Libertad y colaboración en la construcción del conocimiento

9 de junio 2026

Versión 0.2, convertida por GPT 5.5.

Se puede encontrar la última versión de éste trabajo:
`http://ututo.org/articulos/colabora.html`

Primera versión: 10 de febrero del 2005

Diego Saravia Alía

ORCID: 0000-0002-1868-0409

`http://www.ututo.org/dsa`

`mailto:diego.saravia@gmail.com`.
Dpto. Física; Inenco; Fac. Cs. Exactas; Universidad Nacional de Salta. Hipatia.

Nota: muchas partes de este archivo aparecen como artículos independientes en diferentes sitios de Internet. El trabajo es una compilación de esos artículos con algunos agregados. Luego de muchos años lo reformatee y revise parcialmente con gpt.

> ``*Cualquier tirano puede obligar a sus esclavos a cantar himnos a la libertad.''*
>
> *Mariano Moreno*

> ``*No ser puede ser libre y esclavo a la vez''*
>
> *Simón Bolivar*

> ``*Seamos libres, lo demás no importa nada''*
>
> *José de San Martín*

> ``*Andrea: pobre del paś que no tiene héroes. Galileo: pobre del paś que necesita de héroes''*
>
> *Bertold Brecht.*

> *``Sigan ustedes sabiendo que, mucho más temprano que tarde, se abrirán de nuevo las grandes alamedas por donde pase el hombre libre, para construir una sociedad mejor."*
>
> *Salvador Allende.*

> *``Con paso firme pisa la injusticia en nuestros días. Diez mil años ya tienen detrás los opresores."*
>
> *Bertold Brech.*

> *``La naturaleza ha destinado, los griegos a ser libres y los bárbaros a ser esclavos."*
>
> *Eurípides.*

> *``Es evidente que unos nacen naturalmente libres y otros naturalmente esclavos, y para estos, la esclavitud es tan útil como justa."*
>
> *Aristóteles.*

> *``El único, grande, el verdadero bien a que el hombre debe aspirar es la libertad."*
>
> *Espartaco.*

> *``Un pueblo que oprime a otro pueblo no puede ser libre."*
>
> *Inca Yupanqui.*

> *``Vivir libres trabajando o morir combatiendo."*
>
> *Trabajadores textiles de Lyón.*
> *Lista de Solar.*

Índice General

Contexto de la creación libre y colaborativa del conocimiento

Subsecciones

La revolución de la computación y las comunicaciones digitales

La revolución industrial estableció la propiedad privada sobre los bienes de producción, y con ello el capitalismo, en casi todo el planeta, convirtiéndolo en un gigantesco mercado. [1.1]

Lo accesorio siguió el camino de lo principal. Las ideas y sus representaciones estuvieron vinculadas inseparablemente a bienes materiales. La era industrial formó el concepto de que las ideas o sus representaciones son apropiables exclusivamente y constituyen bienes transables de la economía. [Saravia:REC-04,Saravia:DDS-03,Saravia:EI-03,Saravia:MH-01,Hipatia:SM-04]

La combinación de la computación con las comunicaciones interconecta a los seres humanos cambiando sustancialmente sus formas de relacionarse, permitiendo crear estructuras y organizaciones antes inimaginables.

La revolución computacional y comunicacional plantea una nueva lucha entre el monopolio intelectual y la libre expresión o circulación de las ideas. Gracias a Internet, a la digitalización de ``los contenidos" y a la posibilidad de terminar con la escasez impuesta por el soporte material que contenía históricamente a las ideas, ahora éstas pueden fluir libremente, solo limitadas por las ataduras artificiales de leyes

obsoletas. Así el modelo científico se expande hacia otras ramas del conocimiento y aplicaciones tecnológicas.

Nos encontramos a las puertas de nuevas y diversas culturas y sociedades, que complementan, potencian o cambian las preexistentes. Una revolución tecnológica, comunicacional, social y humana sin precedentes en la historia. Estamos frente a la posibilidad de construir conocimiento colectivo y distribuirlo a toda la humanidad en tiempo real y sin costos marginales.

Las nuevas organizaciones humanas en construcción, o ``Sociedades del Conocimiento", están siendo definidas. La ética que consensuemos, los derechos que nos demos, las normas que instrumentemos y la comunión que logremos, definirán y entornarán en gran medida la cultura humana en los próximos siglos, tanto en lo social, como en lo económico y lo político. Así como las metodologías y los medios definen las organizaciones y su mensaje, los instrumentos que rijan el conocimiento determinarán cómo se construye y crea esta sociedad.

Sobre esta nueva organización se plantean viejas contradicciones. Por ejemplo la idea de la brecha digital. Como siempre ``el progreso" llega primero a los privilegiados. Y como siempre la tarea no consiste en eliminar la brecha digital informatizando la pobreza, sino en eliminar la pobreza, en todo caso apoyándonos en el conocimiento libre.

Internet dio inicio formal a esta revolución que conjuga el poder de procesamiento con la digitalización de la información. Una red de todos con todos, y no solo de a dos (teléfonos) o de a uno a muchos (medios masivos). El espíritu libertario que la impregnó, incubado al amparo de las universidades, acogió a la comunidad de hackers, impulsó al software libre y fue forjado por este movimiento. Filosofía y tecnología crecieron al amparo y en paralelo con la red, nutriéndola en el proceso que todavía define el espacio cultural de la naciente Sociedad de la Información.

¿Economía de la escasez o nueva ciencia de la prosperidad?

En las sociedades del conocimiento el camino del crecimiento pasa por liberar las ideas. Los regímenes legales heredados son anacrónicos y un freno a su crecimiento.

Cuando algo que es escaso comienza a dejar de serlo, cuando la prosperidad aumenta porque un bien deja de ser escaso, disminuye el capital [Saravia:REC-04,Saravia:EI-03] que el bien representa y esto no es negativo sino positivo. Debemos deconstruir la economía o ciencia de la escasez y reemplazarla por otra ciencia de la abundancia, del progreso o de la prosperidad y así ayudar a comprender mejor y poder construir otra ``realidad económica". Una ciencia donde las variables ``consideradas como buenas" aumenten con la satisfacción general.

El progreso

En el análisis marxista de la economía se diagnostica que las relaciones sociales de producción incubadas por el capitalismo, retrasaban el desarrollo de sus propias fuerzas productivas [Marx:MC-48][1.2].

Podemos pensar que los frenos a la libertad del conocimiento retrasan el desarrollo de las sociedades del conocimiento y así, el progreso, avance y la evolución de nuestras sociedades. De dos sociedades: una con conocimiento libre y la otra con sistemas de restricción, la primera crecerá más rápido y más eficientemente.

Es posible criticar esta idea si uno no cree en el progreso, ni en la construcción de un edificio científico que avanza y que cada vez tiene más capacidad de entender, predecir y diseñar el futuro.

Podemos vernos a nosotros mismos como el mecanismo por el cual el universo consigue conciencia. La materia dando forma a la vida. La vida construyendo cuerpos e inteligencia. La inteligencia adquiriendo tecnología y capacidad para manejar mayores cantidades de información. ADN, células, cuerpos, cerebros, lenguaje (ser humano), escritura (historia), herramientas y tecnología (cultura, exo-cuerpos), computación e informática, comunicaciones electrónicas e Internet.

A pesar de su evidencia práctica no parece fácil definir el progreso en términos doctrinarios, y probablemente este sea el mayor desafío teórico pendiente en las ciencias modernas [Wagensberg:P-98]. Un desafío -quizás ilusorio- que debiera llevar a redefinir la economía y buena parte de las ciencias sociales.

Las contradicciones del post-capitalismo infinito de las ideas

Se visualiza un camino alternativo a la construcción económica, política e intelectual de lo que algunos quieren presentar como el sucesor del capitalismo liberal; que no es otra cosa que el imperialismo internacional, la neo-liberalización de la globalización, los monopolios intelectuales o el post-capitalismo infinito.

La capacidad de acumular del capitalismo tradicional está limitada a la cantidad de tierras o la capacidad de construir máquinas que reemplacen obreros. En cambio con el potencial ilimitado de las ideas convertidas en bienes transables, expandibles mediante la copia a costo marginal cero, el capital intelectual no tiene límites y los capitalistas pueden seguir acumulando ad infinitum. Así superan las contradicciones planteadas por Marx al capitalismo. Este encuentra la vía de escape a su final anunciado y se transforma de un sistema de libre competencia en un sistema monopólico altamente concentrado.

Costumbres y Leyes

Las costumbres populares que se van estableciendo en los mundos virtuales nos demuestran que compartir conocimiento es la norma, que es lo justo y que es un error limitarlo. Los sistemas P2P (peer to peer) se multiplican, la gente sigue compartiendo software, música y conocimiento por más que costosas campañas de propaganda los señalen como ``piratas"[1.3].

> No debemos olvidar esta ventaja: esta vez son ``ellos" los que tienen que convencer al mundo de que su idea de sociedad es mejor que la que la gente impone en Internet. ``Information WANTS to be free" escribían algunos hackers en los primeros años 90, y tenían razón [Martin:SRB-03].

No siempre lo legal es justo, y muchas veces lo que lo era deja de serlo. En las nuevas realidades no se pueden aplicar las leyes del pasado. No se puede legislar en contra de las costumbres aceptadas; o ``derecho consuetudinario" de Internet. Hoy, ya, algunas empresas empiezan a reconocer lo ilógico de este sistema [Kantor:OEP-05]. Otros defienden las industrias millonarias del espectáculo. ¿Deberemos sacrificar la libertad del conocimiento para posibilitar la supervivencia de ``Titanic", u otras superproducciones?.

Internet con ``su globalización" de los intercambios de ideas, no acepta funcionar como las multinacionales de las comunicaciones desean. Creyeron que podían transformar una red libre y académica en una autopista para hacer negocios. Sin embargo Internet los está transformando a ellos, estableciendo sistemas de comunicación con costo plano en vez de tarifas por tiempo y distancia. El mundo virtual es mucho más que eso: Humanidad, simplemente Humanidad [Saravia:GI-04].

La importancia del conocimiento funcional y del software

El software es la forma del conocimiento [Wikipedia:Ep,Saravia:QC-05] que se ``ejecuta" [Saravia:MH-01]. Se expresa mediante un lenguaje de instrucciones, que personas, la vida[1.4] o las máquinas pueden instrumentar y transformar así la ``realidad" en la que están inmersas mediante su acción.

Para algunos el conocimiento funcional es aquel que es útil y con ello diferencian el conocimiento que debe ser libre, del que no necesariamente debe serlo. Incluyen además del software a manuales, enciclopedias y otras obras ``útiles".

El código es ley, el código ejecuta la ley

En nuestras sociedades mediatizadas por máquinas, el software aplica por sí la ley: impedir que un auto avance en un semáforo, impedir que un cajero automático que te dé más dinero que el que tienes, etc.. Las máquinas pueden controlar nuestras vidas, muchas veces lo hacen y cada vez lo harán más. En algunos casos este control evitará robos, accidentes y demás. Habrá que balancear lo que se gana y se pierde con cada caso. Se deberá minimizar sus efectos negativos con diferentes medidas como mostrar los códigos que los ejecutan para poder auditarlos.

Sin libertad para conocer los códigos de este control estaremos indefensos e incapaces para discutir políticas [Lessig:C-99]. El código -software- es ley de una forma muy profunda [Dennet:TMI-81]. El software cuando se ejecuta en automatismos de puntos de transacción, reemplaza al humano en la aplicación de la ley. Se ejecuta sin discusión. Esto pasa en todo tipo de barreras y controles, y será más común en el futuro.

Software libre

Objetivo, enemigo y motivos (Stallman Dixit):

> ``¿Hasta dónde puede llegar el software libre? No tiene límites, excepto cuando las patentes lo prohíben. El objetivo final del movimiento es proporcionar software libre para hacer todos los trabajos que los usuarios de computadoras quieran hacer y por lo tanto hacer el software propietario obsoleto" [FSF:HPG].

> ``El enemigo es el software propietario" [Stallman:WFS].

> ``¿Por qué debo escribir GNU? ... para poder usar computadoras sin deshonra, he decidido agrupar un cuerpo suficiente de software libre de tal manera que pueda proseguir sin software que no sea libre" [FSF:MG-85].

La humanidad no se suicida: encontró una salida al problema planteado por la revolución informática, salida que constituye una de las caras de la contradicción fundamental de las sociedades del conocimiento: la libertad del conocimiento, vía el software. Un movimiento vital dio forma a Internet tal como la conocemos y se construyó mediante ella. La construcción de un edificio de software totalmente nuevo para garantizar los derechos humanos en la nueva era.

El movimiento del software libre constituye la mayor aventura colectiva de la humanidad: construye conocimiento colectivo, sin coordinación centralizada o mando único.

Un software es libre si sus usuarios y desarrolladores gozan todas estas libertades [FSF:DSL,Saravia:SLA-03]:

- 0.- (ejecutar) La libertad de usar el programa, con cualquier propósito. (analizar la diferencia entre usar y ejecutar: hay otras formas de usar un programa además de ejecutarlo)
- 1.- (inspeccionar) La libertad de estudiar cómo funciona el programa y adaptarlo a sus necesidades. El acceso al código fuente es una condición previa para esto.
- 2.- (redistribuir) La libertad de distribuir copias, con lo que se puede ayudar a otros.
- 3.- (modificar y redistribuir las modificaciones) La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. El acceso al código fuente es un requisito previo para esto.

Estas libertades corresponden a Derechos Humanos establecidos en la carta universal [Hipatia:SM-04][1.5].

Es legal y conveniente incluir las formas binarias o ejecutables del software si existen, y es obligatorio disponibilizar el código fuente: el original y las modificaciones, de existir. Distribuir programas de modo ejecutable es necesario para que los programas libres sean fáciles de instalar.

Autoría y creatividad: sobre hombros de gigantes.

El software libre refuerza la idea de que la autoría de una obra se basa en muchos autores anteriores. Las personas no crean su obra a partir de cero.

El resultado de la interacción de las viejas ideas con las nuevas experiencias es crear nuevo conocimiento. Este nuevo conocimiento no puede quedar bajo control de una sola persona, por más que sea reconocida como autor. El autor debe tener ciertos derechos a partir de su creación. Más no el conjunto de derechos habitualmente conocido como propiedad, sobre la misma, pues esto frenaría el proceso creativo y forzaría cada vez a reinventar una nueva rueda.

El conocimiento es un edificio colectivo, su desarrollo es comunitario, siempre se está construyendo en algún lado, pero también destruyendo en otro.

Los distintos paquetes de software libre no son un producto o servicio, cuando están disponibles universalmente. Se pueden construir productos y servicios a partir de ellos. Pero a diferencia de los paquetes propietarios no son productos en el sentido del mercado. Una empresa o el gobierno no pueden organizar su sistema de adquisiciones en base a productos de software libre, sino en todo caso a servicios de instalación, configuración, adaptación, etc. basados en ese software libre. En tal sentido no es esperable una competencia directa a nivel de productos.

La reacción

Ciertas formas de hacer negocios están condenadas a desaparecer. Entre otras, las industrias de distribución de conocimiento, información y arte son jaqueadas desde varios flancos. Están dejando de ser necesarias. Cualquiera con un ordenador y una conexión de banda ancha puede actuar como centro de distribución. Cualquiera con un editor de textos, con tarjetas de sonido y algún equipo adicional puede producir libros y composiciones musicales de un aceptable nivel. En el momento en que el flujo de información crece sin parar y dinamiza la economía con más fuerza, paradójicamente se destruyen las empresas que en él se sustentan.

El proceso es irreversible. Así como el automóvil desplazó a los carruajes, la computadora personal e Internet desplazarán a las editoriales y a las compañías musicales, o las transformarán de un modo tal que serán irreconocibles en el futuro. Crecerá el movimiento económico vinculado a los servicios, caerá el capital representado por el conocimiento, al fluir este con libertad, y se desconcentrará su control y el poder que representa[1.6].

Las industrias que otrora fueron adalides de la ilustración, hoy son anacrónicas. Pero aún son poderosas corporaciones capaces de ejercer su poder y poner un freno a un avance, que sin lugar a dudas, las perjudica. La reacción puede imponer regulaciones que construyan un mundo oscuro y cerrado, con prácticas de control centralizadas y totalitarias hoy impensables. Así la Digital Millennium Copyright Act (DMCA) y la Informática Traidora (TCG) [Saravia:DDS-03,Inquirrer:ITW-03,IPJustice:WPA,Eff:TC,p2pnet:TPE,TCG:SW,Anderson:PFI-03,AntiDMCA:S son ejemplos de esta tendencia. Esta reacción, si se impone, puede cristalizar y demorar procesos históricos en marcha, causando una enorme regresión y retraso al crecimiento y progreso de la humanidad.

Tal como se indica en [Saravia:NT-05], existe una clara reacción de Microsoft y otros al Software Libre. Estas reacciones pasan por múltiples andariveles, desde los más burdos intentos de asociar a la libertad del conocimiento con el comunismo [Gates:QCS-04,Raymond:HD-04], hasta los estudios muy serios y profesionales, pseudo-científicos diríamos, de consultoras más o menos independientes en relación al concepto de TCO[1.7] y otras cuestiones más o menos técnicas [DelBianco:FSF-05]. Como una línea intermedia de propaganda ideológica se difunden los conceptos de ``libertad de elección'' [ISC:SW] generalmente en los movimientos sociales, reducidos a ONGs, y se introduce el concepto de ``neutralidad tecnológica'' en los foros públicos [FSF:MCA-05].

Como en todo proceso de cambio histórico, hay intereses en pugna. En este caso nos encontramos con dos objetivos y visiones contrapuestos en la conformación de la Sociedad de la Información. Por un lado, algunos estados y corporaciones, que quieren que Internet sea un mecanismo para reforzar su antigua forma de hacer negocios y su poder. Por el otro, ciudadanos y organizaciones que creemos y queremos que Internet sea un fenomenal medio de comunicación que cambie nuestra forma de relacionarnos y que descentralice la economía y el control planetario.

Sociedad libre vs. sociedad de control, fascismo o dictadura digital

Para controlar la revolución digital e impedir que todo el software sea libre (lo que garantiza a la humanidad la libertad de compartir), han aparecido en los últimos años diversas estrategias para detener lo inevitable, todas ellas centradas en el software. Estas estrategias deben ser absolutamente radicales debido a la profundidad del efecto que necesitan obtener:

1. **Patentes de software.** Dado que el mecanismo previamente usado del copyright binario para impedir la copia está fallando debido al copyleft, la gran industria multinacional está apostando a cambiar de sistema. Las patentes de software, muy caras de obtener (barrera para pequeños creadores) y altamente inconvenientes [Knuth:CSP] se han utilizado hasta hoy en forma defensiva por parte de las grandes industrias. Hoy esperan el resultado del debate europeo para utilizarlas como arma contra el software libre. Sirven para impedir la programación como ``hobby''[1.8].

2. **DRM-TCG.** El Trusted Computing Group (TCG) ex Trusted Computing Platform Alliance (TCPA), desarrolla el concepto de Informática Fiable[1.9] (o Traidora según se mire), Palladium para Microsoft, impulsa tecnologías creadas para tomar el control y apropiarse de las computadoras de la gente.

   De esta forma, si logran evitar que la gente ejecute el software que desea, y solo permiten software certificado, lograrán evitar el software libre por vía del control del hardware. Así impedirán que la gente comparta archivos o haga lo que efectivamente desea, la gente dejará de ser dueña de su computadora y perderá el poder de decidir qué se ejecuta en ella.

   Para impedir que la gente comparta música necesitan impedir que la gente ejecute ciertos programas. Si el conocimiento es controlable puede construirse propiedad con él. Los intentos de constituir propiedad y capital no solo llevan a controlar un bien, sino también las ideas, y con ello se produce un ataque directo a la libertad de expresión justificada en términos económicos [Hipatia:SM-04]. Así desarrollan mecanismos artificiales de restricción de la circulación de las ideas por regiones, tiempos, instancias de uso, etc..

   Para impedir que la gente ejecute ciertos programas en sus computadoras necesitan auditar y controlar cada computadora de la Tierra.

3. **Penalización y profundización del copyright**, penalización de la ingeniería inversa, etc..

   Se han hecho campañas de marketing global para definir como ``pirata'' a aquel que comparte información protegida por copyrights[1.10].

La aberración de distribuir software en formato binario y otorgarle a esta práctica abyecta y a este ``contenido" incomprensible para los humanos, carácter de ``obra intelectual" y protegerlo con copyright y aun con patentes, complicó más la cosa.

Utilizar al estado como recaudador de impuestos y a su policía como garante de negocios en cuanto a licenciatarios.

Imponer mediante acuerdos internacionales como ALCA, lo que no pasaría como legislación local.

Criminalizar a los jóvenes que son capaces de inventar mecanismos que superan las trabas electrónicas ridículas al compartir, en lugar de reconocer el talento que los caracteriza.

4. **Penalización del P2P**. Control del compartir. Inventar delitos informáticos lleva a construir una sociedad de control como nunca la hemos visto.

Todo esto y mucho más se desarrolla sólo para defender monopolios industriales que sencillamente no pueden perdurar porque la humanidad evoluciona en otro sentido.

La historia ha demostrado una y otra vez que lo que debe morir, al fin muere, pese a que se intente todo tipo de resistencias y reacciones. La historia de los cambios tecnológicos es así. Algunas industrias desaparecen mientras otras nuevas se crean. Las avalanchas no se detienen, cuanto más las demoren, más estrepitosa será la caída de los muros que las contengan. La vida, la libertad y la inteligencia siempre encuentran su camino.

Finalmente, en cada caso, algún hacker encontrará una puerta para que pase la luz del conocimiento.

El resultado a futuro es claro, pero la transición es compleja, hay intereses en juego y tareas pendientes.

Ontología de la libertad del conocimiento

Subsecciones

Resumen

Palabras clave: Libre, livre, libertad, free, freedom, Ontología, Ontology, conocimiento, knowledge, software, neutralidad, elección, gobiernos, OSI, FSF, Hipatia, economía, comunidad, información, propiedad, consenso, fundamentalismo, democracia, commons.

Resulta interesante analizar las categorías conceptuales, parámetros y clasificaciones vinculadas a las libertades y derechos que acordamos a las personas con relación al conocimiento que han incorporado o que pueden utilizar. Estas categorías constituyen el sustrato de los debates habituales en el movimiento del software libre y del mismo con sus detractores. Un espacio discursivo riquísimo y sumamente complejo al cual contribuyen muchos tipos de razonamientos y principios y que fortalece a un movimiento cada vez más maduro y diverso.

Se identifican cuatro tipos de fundamentos para la libertad del conocimiento: la naturaleza del mismo (ser), éticos (deber ser) - deducidos de la Declaración Universal de los Derechos Humanos - y su conveniencia tanto por su modelo de desarrollo y construcción, como por el tipo de economía que produce. Para cada uno de estos aspectos se muestran los mensajes y argumentos a favor y en contra.

En particular se discute si las ideas son: libres, compartidas: públicas o comunes, o privadas. Se estudia si la propuesta ética puede ser percibida como una moral fundamentalista. Se analizan distintos conceptos y debates como: neutralidad tecnológica, Creative Commons, OSI - FSF, Bazar - Catedral, construcción comunitaria del conocimiento, tecnología o derecho, gratis vs. libre, el voto tecnológico, y distintas vías de acción para construir un mundo donde sus habitantes tengan el derecho de compartir libremente su conocimiento. Se consideran las dos políticas más usadas por los gobiernos que desean trabajar con Software Libre: ``neutralidad total" y ``libertad del software".

Introducción

Plantear las bases de la ontología [Hartmann:NO-54,Gruber:WIO-93,monografias:TCE,Desconocido:OER,Wikipedia:Ep,Wikipedia:SN,Wikipedia:I[1] de los conceptos construidos por el movimiento que trabaja por la libertad del conocimiento no es simple. Tanto la idea de ``libertad" o ``derecho" como el concepto de ``conocimiento"[2.2] son muy difíciles de categorizar, más aún los conceptos combinados, teniendo en cuenta los debates habituales del movimiento[2.3].

Así, la constitución o no, de capital con ideas y las particularidades económicas de las sociedades del conocimiento, plantean la discusión sobre si el mismo puede tenerse como: libre, público, ``common" o privado, según su esencia. El conocimiento por naturaleza es libre, aunque los sistemas legales heredados de la revolución industrial ya obsoletos, lo conviertan en un bien económico escaso. Los entes se dividirían al menos en dos categorías excluyentes: ideas y cosas. La economía es la ``ciencia" que analiza la ``elección" ante restricciones, en particular la asignación de recursos o bienes económicos en condiciones de escasez. Sólo los entes escasos son su objeto. En particular, no son su objeto, las ideas digitalizadas (sin leyes anacrónicas y artificiales que las aten), ya que son naturalmente libres.

El análisis desde el ``deber ser", más allá de la naturaleza conceptual del conocimiento, nos lleva a la óptica de los derechos humanos y las libertades asociadas, fundamentalmente con la libertad de expresión, la libertad de educar y de conocer lo que se utiliza.

El estudio desde la conveniencia y utilidad, dadas especialmente por el carácter colectivo de la construcción del conocimiento, nos lleva a la socialización del mismo y nos presenta otra visión del fenómeno.

Estas reflexiones pueden ser centrales para algunas comunidades políticas, especialmente aquellas con una visión de izquierda, en las que la palabra ``libre" tiene reminiscencias de defensa del capitalismo, el libre mercado, el libre albedrío basado en el egoísmo, las libertades individuales relativas a la propiedad y la libertad de competir para permitir la evolución -progreso- basado en la selección ``natural"- o no - de los ``mejores".

En estas comunidades -algunas vinculadas con la idea de conocimiento libre por el lado de software pero también por la ecología, los transgénicos, las medicinas, las patentes y la transferencia y dependencia tecnológica- muchos se ilusionan con la aparición en escena de la idea de conocimiento público, especialmente con la idea de ``commons" [Wikipedia:TC,CC:SW].

Por todo ello para ``la izquierda" parece imperioso el ``reapropiarse" totalmente del concepto de la libertad, encerrado por las derechas liberales del planeta. La libertad del conocimiento les da una inigualable oportunidad.

También se puede ver en el software libre a un nuevo paradigma o forma posible y justa de hacer negocios en el campo de las Tecnologías de la Información y/o diferentes metodologías para compartir y construir comunidades con capacidades productivas políticamente correctas, sustentables y tecnológicamente apropiadas.

En este trabajo se pretende iluminar desde varios ángulos estos debates y estudiar las contradicciones comunes en las opiniones de diferentes voces del movimiento. En el centro de este problema se encuentra la discusión habitual sobre si la defensa del software libre se construye sobre la naturaleza del problema, sobre una ética que deriva sus conclusiones de intereses conjuntos de la humanidad que expresan un proyecto universal o si simplemente se fundamenta en una moral auto-referencial reafirmada en creencias sin más sustento que la conveniencia de quienes se aprovechan del mismo.

Discusiones en la comunidad y hacia afuera de la misma

Introducción: ¿por naturaleza, ética, moral fundamentalista, o conveniencia?

Identificamos al menos cuatro conceptos, ideas, enfoques o fuentes doctrinarias para apoyar u oponerse a la libertad del conocimiento. En el análisis de los mismos se encuentra la raíz de gran parte de los debates de la comunidad del software libre consigo misma y entre la misma y sus oponentes.

Estos conceptos parten de distintas categorías de análisis y por ello es complejo analizarlos y estructurarlos sin tener un sustrato ontológico preacordado.

Es interesante la interrelación entre análisis basados en intereses concretos, de los que pueden surgir ideologías, o intereses generales que pueden construir discursos éticos. Otros pueden tomar estos fundamentos como morales [GAUEE:SW,Boff:EM:03,Gutierrez:EMT], los que a su vez también pueden ser fundados en conveniencias [Odum:AES-80]. Cada uno de éstos análisis puede aplicar su propia racionalidad y hasta cierto punto sus herramientas científicas.

Así, estas categorías conceptuales: ideología, moral, ética y los intereses se entrecruzan con la razón, el cientificismo, los consensos y los debates para construir un rico espacio discursivo que cruza las conversaciones sobre las sociedades del conocimiento, mientras las realidades económicas, sociales y políticas dirimen el futuro de la humanidad sobre la base del software y otros tipos de conocimientos.

Clases de argumentos y contra-argumentos:

Tipo	Req. [Saravia:SLA-03]	Cuestión	SI	NO

Naturaleza (ser)		A. Las ideas son libres/ públicas por naturaleza:	A.- Privatizarlo retrasa el progreso.	B.- Para progresar debe privatizárselo mediante la ley y el poder de policía.
Ética o Moral (deber ser)	Transpar., Educación, Sociedad.	C. Compartir, comunicar ser solidarios y conocer son DDHH:	C.- Por razones éticas el conocimiento (funcional) debe ser libre. Solidaridad, libertad.	D.- Esa es una posición fundamentalista de carácter moral, no ético. El derecho de autor limita los derechos humanos. Convivencia, consenso. El SL puede promoverse (o no), pero no imponerse [Romero:OPT-05]. Libertad de elección. Los autores deben retener todo el poder de decidir cómo se usa su obra. Los usuarios deben poder elegir qué usan dentro de las opciones ofertadas por los distintos autores incluyendo las restricciones ofertadas.
Conveniencia	Calidad, Seguridad, Costos	E. Modelo de desarrollo:	E.- El comunitario es más efectivo y eficiente. Compitamos y midamos todos los factores	F.- ``Eso está por verse". Compitamos. Neutralidad Tecnológica. No miremos los factores no técnicos.
Conveniencia	Economía	G. Modelo económico basado en:	G.- los servicios. No al monopolio	H.- la renta del capital intelectual. Monopolio necesario.

La columna requisitos (Req.) vincula estas clases conceptuales con los requisitos que los estados (y en muchos casos otros) deben garantizar con relación al software. Ver el documento 29 requisitos de [Saravia:SLA-03].

- A.- El razonamiento basado en el ``ser" que indica que la naturaleza del conocimiento digitalizado es libre ya que no tiene costo de duplicación y que cada mente que lo posee puede hacer con él lo que desee.[2.4].
- B.- El pensamiento que no debe existir el conocimiento libre, todo debe ser contabilizable y apropiable con exclusividad para tener un sistema económico eficiente. El conocimiento sin copyright o en el ``dominio público" (si se puede utilizar tal expresión para las ideas libres, no protegidas por copyright), sólo es interesante si cualquiera puede apropiárselo en forma exclusiva.

Sólo el conocimiento privativo es económicamente eficiente pues recompensa adecuadamente a sus ``creadores" y ``productores" [2.5] [Abadia:MCU-05].

Una posición intermedia en este sentido plantea que el Software Libre no es perjudicial, como no lo es el de mal llamado ``dominio público", pero sí el copyleft. Las posiciones que sostienen que la licencia GPL es negativa e inconveniente. El copyleft no es aceptable éticamente pues ``destruye la propiedad privada" por su carácter ``virósico". Se acepta el software libre no copyleft, pues puede ser utilizado en forma privada. Esta posición es también un burdo intento de atacar jurídicamente al Software Libre pensando en algunos procesos judiciales e intentando que los gobiernos no licencien a terceros su obra intelectual bajo la GPL.

- C.- El planteo de la Free Software Foundation (FSF), desde el ``deber ser", basado en razonamientos éticos.[2.6] Considera que las ideas funcionales deben ser libres. Construyó un edificio de software libre protegido bajo el ``copyleft" y trata de bloquear las patentes de software. En este marco considera anti-ético al software privativo ya que impide ejercer el derecho de libre expresión, impide ser solidario e impide compartir.

  En última instancia este planteo, sea solo para las obras funcionales o para todas las obras intelectuales puede sustentarse de la Declaración Universal de Derechos Humanos [Hipatia:SM-04,Boll:CDC:03]

- D.- Las posiciones que critican y sindican como ``fundamentalistas" las posiciones éticas y a sus proponentes como ``zelotes, talibanes o chiitas" [Bacon:TZO-05]. El conocimiento libre debe convivir con el propietario. Debe garantizarse la libertad de ``elección". Así se critica la pretensión de acabar con el software privativo, como si fuese una elección de dos alternativas válidas que pueden convivir. Muchos de los proponentes de este concepto comparten también la posición de la OSI: Sincere Choice [Perens:SC]. Son interesantes los conceptos de ``fundamentalismo del software libre" y de ``libre elección".

- E.- La posición de la Open Source Initiative (OSI) que postula que debido a su modelo de desarrollo, el software libre es mejor y por ende ``conveniente", aconsejando por estos motivos, a los diferentes actores a usarlo y promoverlo. No hay crítica ética. El software libre triunfará por sus méritos técnicos.

- F.- Es una cuestión técnica: neutralidad tecnológica. Una posición conceptualmente similar a la de la OSI, pero efectuada desde los adversarios del Software Libre por lo que no parte de considerar al Software Libre mejor técnicamente. Un exponente es ``Initiative for Software Choice" [ISC:SW]. Esta posición no deja de ser propaganda económica para los gobiernos, pero es interesante incluirla en el análisis.

- G.- Se plantean sociedades ``modernas" cuyas economías giran sobre los servicios y no sobre la renta del capital. El software, cada vez más, genera negocios mediante acuerdos de servicios entre usuarios y desarrolladores o integradores [].

- H.- Los propietarios del copyright plantean que les conviene seguir acumulando renta por su capital [Abadia:MCU-05] y que el Software libre perjudica la economía. Un capital cada vez más concentrado. Es un argumento de defensa de sus propios intereses. Son muy pocas y concentradas las compañías que hoy ganan dinero en base a distribuir licencias. No parece probable que nuevos actores puedan entrar en éste mercado.

A. Sobre la naturaleza del concepto de la libertad del conocimiento: libre o público

La clasificación de los bienes

Para aclarar las ideas se presenta la clasificación habitual de los bienes de la ciencia económica.

En la misma los bienes ``incrementan la utilidad" en contraposición a los "malos".

Los mismos se clasifican en [Wikipedia:E,Wikipedia:CG,Wikipedia:GE,Wikipedia:FG,Wikipedia:PG]:

1. libres: hay más de lo que se requiere o demanda, o no son contables.
2. escasos o económicos: hay menos que los que se demanda, se debe decidir cómo o quiénes los usan. El problema que funda la ``ciencia económica": la decisión ante las restricciones.

Los recursos o bienes económicos, tienen un sinfín de criterios de clasificación. Dos particularmente interesantes son:

1. el relativo a su uso: exclusivo o no exclusivo. Si no es posible excluir a una persona de su uso se denominan de uso no exclusivo. Caso contrario son de uso exclusivo.

2. el relativo a si su uso implica competencia o rivalidad: Es decir si una persona lo consume ya no esta disponible para otra (por ejemplo: se agota un poco más).

Este doble sistema de clasificación nos genera 4 tipos de bienes escasos:

BIENES ESCASOS	exclusivo	no exclusivo
rivales	privados	commons
no rivales	club	públicos

Tenemos entonces:

libre:
aire, agua de mar, conocimiento digital;
privados:
pan; el propietario decide;
club:

una escuela privada, todos los partícipes lo usan;

commons:

un recurso natural: petróleo; el que llega primero decide;

públicos:

ejércitos. Toda la comunidad lo usa sin consumirlo.

No se debe confundir público con libre, confusión muy habitual en el análisis de las ideas. El ejército por ejemplo es limitado, no se puede duplicar sin costo, pero todos ``disfrutan la paz que mantiene".

Existe un conjunto de personas que confunden [Stiglitz:KGP] los bienes libres con los públicos. La intención en el fondo es incorporar todo al ámbito de la economía. Personalmente considero mejor poner lo menos posible en el marco de la economía y construir una disciplina científica diferente para las cuestiones vinculadas al crecimiento -donde no exista escasez-, las estructuras ``vivas" y el conocimiento en el universo.

Libertad del conocimiento vs. las cosas compartidas.

La información no es un recurso económico, así no se ``comparte" en el sentido estricto del término, el conocimiento incluido en la misma se duplica infinitamente y sin costo[2.7], mente a mente, cada una con su ``copia".

Si alguien tiene una manzana debe decidir que hacer, la guarda, la planta, la comparte, la cede, etc.. Cuando alguien pasa una idea a otro, ambos la tienen por completo y pueden usarla sin necesidad de tomar decisiones en común sobre la misma. No hay apropiación exclusiva [Saravia:EI-03]. Cada persona además toma a la idea como quiere o puede y la adapta. No hay dos ideas exactamente iguales en las mentes de dos personas.

Las ideas caminan sobre las piernas de los hombres. Así, ``una" idea tiene más fuerza cuando tiene ``muchas piernas".

Las ideas no son cosas, también existen otros entes ``nombrables" como la luz, los abrazos y los besos, que no son cosas [Chaparro:TTN,Judicial:INC]. Como las variables de un lenguaje informático, pueden ni siquiera tener nombre.

Las ideas pueden ser ``apropiadas" por un grupo humano o comunidad, pero eso no pone ningún límite a cualquier otra persona, grupo o comunidad, ya que no es una apropiación exclusiva, característica necesaria para el establecimiento de la propiedad [Wikipedia:P]. Se desprende entonces que el concepto central aquí es LA LIBERTAD DEL CONOCIMIENTO, no la dicotomía entre su APROPIACIÓN COLECTIVA vs. su APROPIACIÓN INDIVIDUAL exclusiva. La apropiación colectiva puede ser un fin loable para las cosas materiales, pero de ningún modo para las ideas.

En las cosas comunes la comunidad decide por algún medio sobre ellas. Por ejemplo una plaza en una ciudad. La ciudadanía a través de su gobierno decide qué hacer con la plaza. En cambio con las ideas no

es así. No hay decisión común. Cada persona tiene su versión de la idea. No hay nada en común a decidir.

No tiene nada que ver la libertad del conocimiento con la propiedad compartida o el bien común.

Tampoco tiene sentido plantearse la dicotomía entre ``derechos" y ``servicios" que se plantea en determinados problemas. Por ejemplo la distribución de agua potable, ¿debe ser un derecho de todos los ciudadanos solventado por el estado, o cada uno debe pagar lo que consume?. Algunos sostienen que se debe contabilizar todo costo y cargárselo directamente al beneficiario, otros que debe ser un derecho general. Dilemas similares se han estado planteando para la educación, etc.. Este planteo lleva de última al concepto de renta universal básica. Para el conocimiento el problema carece de sentido.

Cuando se habla de bienes públicos uno suele referirse a cosas que tienen un dominio comunitario, cosas sobre las cuales se toman decisiones. Y esas decisiones las toma una comunidad, por la vía que esta tenga establecida. Puede ser una persona jurídica ``estrecha y legal" del tipo ``occidental y cristiano" o puede ser una ``comunidad abierta" del tipo ``usuarios y desarrolladores del soft libre" o ``pueblos originarios". Estas cuestiones se vinculan habitualmente con la propiedad ya que la misma no es más que la restricción a terceros para decidir o usar un bien propio.

Hay que evitar a toda costa usar las palabras de ``bien común", ``bien público", ``commons" o cualquier concepto que represente la idea de un ``patrimonio colectivo" para las ideas. Las ideas no son bienes económicos, no son productos, no son patrimonio. Por ende no pueden ser del individuo, pero tampoco de la comunidad en su conjunto o de alguna comunidad en particular. Son libres. Con el conocimiento la humanidad ``progresa" pero no por el lado material.

La cuestión de fondo es que hay alguien: ``individuo o comunidad" que debe tomar decisiones sobre el ``bien económico", lo que no pasa con las ideas.

No son ``bienes económicos", mucho menos propiedad. Nadie puede decidir sobre todas las instancias de esa idea, solo sobre la que tenga en su cabeza. Cualquiera puede hacer lo que quiera con ``su" idea. Por lo tanto no hay una comunidad que deba decidir. No es un recurso, o bien, o propiedad regulado por la escasez.

El conocimiento no es propiedad común, no es propiedad, es libre. Para muchos puede ser agradable asociar el software libre con las ideas de propiedad comúnitaria, pero no es lo mismo.

No es un bien en el sentido económico, ni un recurso económico. No está afectado por la escasez (debido a su intrínseca potencialidad de reproducción ilimitada, como el conocimiento). Queda fuera de la economía o ciencia que estudia la administración de los recursos escasos). Es como la sal a la comida, aporta gusto, pero no llena.

No hay que confundir creación de conocimiento, con producción de múltiples copias de una idea. La creación de una idea tiene un costo independiente del costo marginal nulo de la reproducción (producción) de su expresión digital.

Igualmente erróneo es pensar en la libertad de las cosas. Lamentablemente las cosas son escasas y no pueden duplicarse sin límite y sin costo.

Otra cuestión interesante se da cuando una obra intelectual se libera en el llamado ``dominio público", desde el punto de vista de nuestra clasificación estas obras serían bienes económicos libres al igual que las obras con copyright libre.

Es decir el derecho habiente decide no protegerla con copyright. Desde el punto de vista de las libertades una obra en tal dominio da a las personas los mismos derechos que una obra distribuida en forma libre. Al igual que las obras libres con copyright no copyleft, estas obras pueden ser apropiadas.

Creative Commons y los bienes tipo ``commons''

El conjunto de licencias conocidos como ``Creative Commons" [CC:SW] están brindando un gran servicio a la difusión de la problemática de la libertad del conocimiento entre los artistas e intelectuales, sin embargo su difusión no está libre de problemas.

La palabra ``Commons" alienta la ilusión de que las ideas son compartidas por todos, que son un ``recurso" comunitario.

En este punto es habitual mencionar ``la tragedia de los commons" que se basa en que si un bien aparentemente no escaso es sobreexplotado se convierte en escaso y o bien se agota o bien deja de ser comunitario [2.8]. Muchos piensan que el software libre representa una nueva oportunidad de evitar la tragedia de los ``commons".

El enfoque disperso. No es un tema, sino varios

RMS (Richard M. Stallman) critica duramente la idea de la ``propiedad intelectual" [Stallman:DPI]. Analizamos críticamente esta visión que niega un enfoque unificador.

Él indica sabiamente que el término ``propiedad intelectual" es inapropiado, y lo fundamenta en la falta de sentido de unificar arbitrariamente sistemas legales totalmente diferentes.

Pero parece conveniente complementar con otros conceptos, la lógica interna de su argumentación[2.9].

Si partimos de un estudio analítico, la construcción desde abajo hacia arriba de las ideas de patentes, copyrights, marcas y posiblemente otros conceptos; no parte de una base común. Estos regímenes surgen de diferentes motivaciones, con sistemas legales diferentes y sin ninguna vinculación. Respecto de esto, Stallman tiene razón.

Pero si partimos de un análisis sintético y observamos algunas consecuencias de estos sistemas, llegamos a otra conclusión.

Estos sistemas construyen valor económico sobre elementos inmateriales, al hacer escaso lo abundante y restringir su circulación. Creando derechos individuales y restringiendo derechos colectivos, asignan valor económico a las ideas y sus representaciones. Les designan un titular y las ponen en el mercado, permiten su compra, venta y alquiler o licenciamiento. De esa forma aseguran la apropiación exclusiva de las ideas (como es el caso de las patentes) o sobre sus representaciones (el caso del copyright); durante un largo período de tiempo. [2.10]

``Propiedad Intelectual" es -como dice RMS- una bandera publicitaria, pero también un concepto unificador de varios sistemas legales diferentes en cuanto a su efecto e intencionalidad económica. Sin duda alguna puesto al servicio de los sectores más poderosos y concentrados.

Por lo tanto la posición de RMS es correcta en cuanto a oponerse a la unificación conceptual, desde el punto de vista del derecho, de tres regímenes diferentes. Pero no debemos perder de vista el enorme potencial ideológico unificador del concepto de ``Derecho Intelectual" en términos económicos.

El término: ``Propiedad Intelectual" es un oxímoron por naturaleza autocontradictorio, ya que las ideas no son apropiables con exclusividad.

Pero mediante diferentes regímenes legales aplicados a algunos tipos de ideas o a las representaciones de otros tipos, se consigue el mismo y unificador objetivo que se proyecta a un amplio conjunto de tipos, incluyendo a las ``obras intelectuales", los ``inventos", los ``métodos de negocio", al ``software binario", etc.: construir escasez de la abundancia imponiéndoles un costo artificial y con ello incrementando el ``capital" del planeta puesto en juego en el mercado.

Existe un autor y un derecho-habiente. El derecho-habiente ejerce sus derechos, pero estos no constituyen el conjunto de derechos que habitualmente asignamos a la propiedad de un bien. Propiedad implica muchos derechos, no solamente ser derecho-habiente. Es decir que se puede tener ciertos derechos con respecto a una obra, pero no necesariamente derechos de propiedad.

Estratégicamente, la visión de incidir sobre los regímenes legales por separado, uno por uno, mientras se construye el edificio del software libre, es el mejor camino para evitar el mal mayor.

Vemos entonces, que el término ``Propiedad Intelectual" es inconveniente por donde se lo mire. Para discutir las leyes es mejor seguir a RMS. Para discutir sus efectos sobre la realidad económica es conveniente referirse a esta disciplina legal como ``Derecho Intelectual" o a sus acciones como ``Restricciones al Intelecto".

¿Propiedad, riqueza, progreso o prosperidad?

El conocimiento como riqueza o como impulsor de la prosperidad: ¿músculo o sal?

Para algunos la idea de que lo malo del concepto ``propiedad de las ideas" es que esa propiedad sea individual. En contrapartida serían buenos la riqueza o patrimonio intelectual en tanto sean comunitarios o compartidos.

Esto puede ser beneficioso para cosas materiales, pero las ideas tienen otras características. Éstas no son apropiables y no son escasas; por ello el concepto de riqueza o patrimonio tampoco se les debe aplicar, ni siquiera en su forma comunitaria o compartida.

La propuesta encabezada por la FSF Europa de crear una organización con otro nombre, para reemplazar a la OMPI [FSFE:ADG-04] (Organización Mundial de Propiedad Intelectual [WIPO:SW]) puede servir para cuestionar la situación actual y para debatir y rediscutir sus fines, tal como lo pretenden los gobiernos de Argentina y Brasil.

Mucho más interesante sería hablar de la organización del Progreso o de la Prosperidad Intelectual.

Así la nueva OMRI (La R es riqueza) podría tener como fin la promoción de la creatividad humana -mecenas universal-, aunque para esa tarea tenemos otras organizaciones más apropiadas, como las científicas, las universidades, los colectivos artísticos, los movimientos de software libre, etc. En el corto plazo la OMRI también podría servir para limitar el efecto de las patentes de software sobre el software libre. Si el objetivo de evitar las patentes de software no se logra.

La OMPI tiene como objeto poner en el mercado a las ideas. Así como otros quieren poner en el ``mercado libre" y competitivo a la educación. Y este objeto es perverso.

En el contexto ideal de la ``libertad del conocimiento", la OMPI dejaría de tener sentido y en tal caso debería ser disuelta. [2.11]

El nombre propuesto para la nueva organización carecería del problema del término Propiedad, reemplazándolo por Riqueza.

Esta cuestión requiere una discusión muy de fondo sobre las diferencias conceptuales entre Propiedad, Capital y Riqueza.

En la sociedad capitalista, el valor del capital se mide por su capacidad de generar renta. Si el software se distribuye libremente y no genera renta, no constituye capital, no es apropiable en forma exclusiva. Habría que tener una nueva teoría económica que sustente una definición de riqueza no basada en su capacidad de generar renta para sustentar este nuevo nombre.

Otro mejor nombre sería ``Progreso" o ``Prosperidad Intelectual".

En principio no parece tener sentido una organización internacional en reemplazo de la OMPI. Lo ideal sería disolverla.

La libertad del conocimiento ... ¿solo virtual?

Las ideas son libres en tanto se liberen de la tiranía del medio que las transporta. Así, no es tanto la cuestión de si son obras funcionales o no lo que determina que las ideas deban ser libres [Stallman:NKC-2000] (principio ético propuesto por RMS), sino la realidad más materialista[2.12].

Un libro impreso o un long-play, que tiene un alto costo de impresión o grabación, no podrían ser libres.

Lo que tiene sentido liberar son las expresiones electrónicas o virtuales de una obra, las que no tienen costo de reproducción.

Si está disponible la versión electrónica, quien opte por leer un libro en papel, consume recursos escasos -árboles, transporte- y es justo que pague por ello.

No debiera impedirse que se digitalice o virtualice y redistribuya libremente una obra impresa en forma digital, ya que esto es algo que ayuda a liberar el conocimiento y difundir las ideas. Quien costee la digitalización de una obra, es un héroe, no un villano. Incluso podría obligarse por ley.

En el futuro la industria editorial/musical ``física'' [2.13] quedará reducida a un mercado de lujo. Pero está bien que así sea.

C vs. D. Ética o conveniencia, militancia o marketing, política o comercio

FSF u OSI

En inglés el término ``free'' es confuso, puede significar tanto libre como gratis, por ello un grupo de personas propuso el término alternativo ``open source'', pero es también confuso, pues se refiere a una sola de las cuatro libertades: la visibilidad del código fuente [Wheeler:OSR-04,Wheeler:WOS-04,Wheeler:CSL-01,Stallman:WFS,DiBona:OSV-99,Moglen:FSM-00,OpenSou

Entonces para referirse al mismo software existen en inglés dos nombres.

El problema es que asociada a esta discusión de nombres hay otra discusión política. Dos grupos: la OSI (Open Source Initiative), y la FSF (Free Software Foundation), sostienen cada uno de los nombres y cada organización tiene motivos diferentes para apoyar al mismo software y mantiene una definición con diferente redacción, y una lista separada de licencias ``libres'', en principio la misma [OSI:AL,OSI:SW]. La definición OSI está basada en las guías originales del proyecto Debian [OSI:OSD,Debian:DFSL-98]. La de la FSF es la original del proyecto GNU, muy anterior. En principio no debiera existir ningún software significativo que sea OSI-libre y no FSF-libre, o al revés. Ambas coinciden en muchas prácticas, desarrollan y distribuyen el mismo software bajo los mismos esquemas legales. Las diferencias son filosóficas. Es interesante citar a Perens que cambió de idea luego de fundar la OSI [Perens:TTA-99].

La FSF (Free Software Foundation) indica que la razón fundamental para usar Software Libre es ética.

La OSI (Open Source Initiative) indica que la razón fundamental es la conveniencia. Impulsa el Software Libre en términos pragmáticos más que filosóficos. Sus principales impulsores son Eric Raymond, Bruce Perens y O'Reilly. No todos dan a la OSI el carácter de un movimiento

[Linuxmag:SR-99]. La OSI presenta una estrategia de marketing para el Software Libre. La sustancia no cambia, la actitud sí.[2.14] En castellano la palabra open source tiene los mismos problemas que en inglés. En cambio la palabra libre no tiene problemas. Lo mismo para otros idiomas. Mejor usar las traducciones de ``Software Libre", ya que no hay confusión con gratis. Para evitar identificarse algunos usan, aún en inglés ``Libre Software". También FOSS o FLOSS.

Además la OSI se identifica más con un modelo particular y específico de desarrollo conocido como ``el Bazar". Se puede decir que el ``Software Libre" se sustenta en una filosofía política, y el ``Open Source" en una metodología de desarrollo.

Resumiendo tenemos:

1. dos organizaciones, personas y razones diferentes: OSI, FSF,
2. una comunidad: del software libre,
3. un mismo software: el software libre,
4. varios modelos de desarrollo,
5. muchos nombres en inglés: FLOSS, FOSS, libre software, free software, open source software.
6. un nombre en otros idiomas: software libre y sus traducciones: Francés: logiciel libre, Alemán: freie Software, Ruso: svobodny programy, Chino: zi4you2 ruan3jian4, Japonés: jiyuu [na] sofuto, Esperanto: libera programaro, Sueco: fri programvara, Holandés: vrije software, Portugués: software livre, Danés: fri software, Italiano: software libero, Catalán: software lliure.

Debe además decirse que la OSI rechaza los argumentos éticos. La FSF en cambio no rechaza los otros argumentos. La FSF apoya todo el conjunto argumental del movimiento, indica que lo importante es la libertad pero acepta como un ``Bonus" que el software libre sea más conveniente.

B. Para Bill Gates somos comunistas

Bill Gates actúa de acuerdo a sus intereses cuando dice que algunos del movimiento de Software Libre somos comunistas [Gates:QCS-04], pero eso no significa que no tenga sus fundamentos al decirlo.

Con libertad del conocimiento las ideas y sus expresiones se pueden usar sin pagar derechos. Así se licua el ``capital intelectual" (de Bill Gates y otros), porque los sistemas operativos, aplicativos, música y textos gratuitos están disponibles para todos. Esto significa que socializa el capital intelectual. Entonces al socializar el capital liquida la propiedad privada de los ``bienes de producción" intelectuales. Entonces seríamos comunistas, en el sentido que ``actuamos" en contra de quienes son propietarios de ciertos tipos de bienes de producción.

Así el programa del software libre claramente expresado por RMS [FSF:HPG] de proveer soft alternativo a todo soft privativo es claramente destructor de la propiedad de esos medios de producción. Y por ende, en tal sentido, comunista.

El comunismo no solo es eso, pero esa es la base de su programa y una de las ideas que lo caracteriza: oponerse a la propiedad privada de los medios de producción.

El conocimiento libre es a las sociedades del conocimiento lo que el comunismo fue a las sociedades industriales, un destructor de su ``capital'' intelectual o un constructor de otra forma de capital.

Por otro lado el conocimiento privado constituye un monopolio, entonces la real oposición del conocimiento libre es a los medios privados de producción monopólicos, o con un monopolio artificial creado por el estado. En tal sentido el proyecto del conocimiento libre puede ser compartido no solo por ``los comunistas'' -socializa medios de producción- sino también por ``los liberales'' -evita monopolios-. Otros grupos ideológicos (en realidad todos) también pueden considerarlo como propio [Solar:SLP]. Por diferentes motivos el conocimiento libre apela a principios éticos básicos compartidos por las grandes religiones y grupos políticos: solidaridad, cuidado del ambiente, compartir, libertad, comunidad, etc..

D. Sobre el fundamentalismo

Los zelotes fundamentalistas

> El fundamentalismo es, según su sentido originario, una corriente surgida en el protestantismo norteamericano del siglo XIX, la cual se pronunció contra el evolucionismo y la crítica bíblica y que, junto con la defensa de la absoluta infalibilidad de la Escritura, intentó proporcionar un sólido fundamento cristiano contra ambos. Sin duda existen analogías con respecto a esta posición en otros universos espirituales, pero si se convierte en identidad la analogía, se incurre en una simplificación errónea. De dicha fórmula se ha extraído una clave demasiado simplificada, a través de la cual se pretende dividir el mundo en dos mitades, una buena y otra mala. La línea del pretendido fundamentalismo se extiende entonces desde el protestante y el católico, hasta el fundamentalismo islámico y el marxista. La diferencia de los contenidos no cuenta aquí para nada. Fundamentalista es aquel que siempre tiene convicciones firmes, por ello actúa como factor creador de conflictos y como enemigo del progreso.

> Lo bueno sería, por el contrario, la duda, la lucha contra antiguas convicciones, y con esto, todos los movimientos modernos no dogmáticos o antidogmáticos. Pero, como se desprende del contenido, a partir de un esquema clasificatorio puramente formal no puede interpretarse realmente el mundo. Según mi parecer, se debería dejar a un lado la expresión ``fundamentalismo islámico'', porque oculta, bajo una misma etiqueta, procesos muy diferentes en lugar de aclararlos. Habría que diferenciar, según me parece, el punto de partida del nuevo despertar islámico y sus diversas formas. Ratzinger [Ratzinger:LIV-05]

Libre Elección

Muchos impugnan la pretensión de eliminar el software propietario en base a la necesidad de mantener la ``libre elección"

Al ser humano le fue dada la libertad de elección, o libre albedrío, según diversos y concurrentes pensamientos religiosos. Podemos dar múltiples interpretaciones a esta ``capacidad humana" vinculada según algunos con la voluntad [Schopenhauer:SLV-39,Schopenhauer:MRV-18,Saravia:QC-05],pero muy pocas personas argumentarán que no es esencial a la forma en que nos vemos como seres pensantes y libres.

Esta libertad, asociada a una moral determinada, nos permite optar por el bien o el mal. O simplemente elegir diferentes caminos alternativos. Decidir dentro de ciertos márgenes.

``La libertad de matar no es verdadera libertad sino una tiranía que reduce al ser humano en esclavitud" [Ratzinger:PNM-05].[2.15]

¿Es la cuestión del tipo de licenciamiento del software una cuestión vinculada con la libertad de elección [Jorge:MLE-04]?

En primer lugar el software licenciado en forma privativa está diseñado para quitarle a los programadores y usuarios libertades y con ello libertades de elección. Así un usuario de Windows no puede instalarlo en varias máquinas, o modificarlo, etc.. Su espacio de opciones y alternativas se reduce.

Por otro lado el proyecto del Software Libre propone liberar todo el software. El camino para ello es desarrollar alternativas libres a cada programa propietario [FSF:HPG]. Si este proyecto triunfa, la gente tendrá muchas más opciones; como consecuencia indirecta, no será sustentable el licenciamiento del software como privativo. Microsoft ha salido a plantear la situación como un impedimento a su negocio. En realidad en éste caso, las empresas, incluida Microsoft, podrán distribuir su software como libre adaptando su modelo de negocios. Hoy por hoy, seamos sinceros, hay un monopolio, y lo ejerce Microsoft. En todo caso la existencia de este monopolio es el motivo por el que muchos otros empresarios están impedidos de competir y pierden su espacio de opciones.

Entonces el software libre ¿implica alguna contradicción con la libertad de elección?

Claramente la respuesta es no. No se reduce el espacio de opciones de nadie. Aun en caso de que no quede software privativo en la Tierra[2.16], nadie perdería opciones. Algunos ganarían menos dinero, y sería muy difícil construir monopolios. El software libre y su comunidad ha ampliado extraordinariamente el margen de opciones para todos en cuanto a tecnologías. Está construyendo una alternativa y permitiendo salir de un monopolio. Hoy otro mundo no es solo posible sino que está en construcción.

El software privativo en cambio reduce grandemente el espacio de opciones de desarrolladores y usuarios.

Por ello, lo que los militantes del movimiento afirmamos es que es malo usar software privativo.

Porque usarlo es justamente limitar la libertad de elección y votar por un mundo donde no existe libertad del conocimiento y por muchos otros motivos brillantemente expuestos por los defensores del mismo.

Plantear la cuestión de la Libertad de Elección es conceptualmente ponerse en el mismo bando que los que dicen que el mercado debe imponerse a la Democracia [Chua:MLL-03]. Que el método para decidir que software se usa es el mercado y que los pueblos no pueden usar los mecanismos de decisión democráticos para estas cuestiones. Entonces se usa la forma política-organizativa-de_control de las logias para calificar de fundamentalista la posición ética de conveniencia de las mayorías e imponer la posición de la conveniencia de los mercados. Mayoría de capitales vs. mayoría de votos.

Todos los días uno se obliga a usar determinado software

La gente es obligada a usar determinado software para trabajar, sucede todos los días en las oficinas. También en los telecentros, la gente entra y debe usar lo que está ahí o irse a otro lado, sucede cada vez que alguien envía un archivo adjunto al correo en ``word", con el que se debe usar determinado editor de texto (con algunas alternativas) para verlo.

Creative Commons en el marco de la libertad de elección

Por otra parte ``Creative Commons", (CC) [CC:SW], alimenta la posibilidad de la ``libre elección" presuponiendo:

1. Que todos los regímenes son buenos para obras no funcionales. No todos encuentran que el conocimiento ``no funcional" sea o deba ser libre [Barton:HSS].
2. Que los autores eligen y los ``usuarios" deben acatar.[2.17]

¿Cómo usar Creative Commons?

Usar CC para el software o su documentación[2.18] es un retraso, pero es un avance en relación a la música y los libros. CC plantea el desafío de enseñar con más precisión a todos los involucrados las cuestiones de derecho relativas a las licencias. Hay muchas personas en el movimiento encandiladas con el excelente marketing de CC, y que no han estudiado las cuestiones de fondo.

La ``Free documentation Licence" FDL [Stallman:FDL] lleva asociado el mensaje de la ``libertad del conocimiento" y la CC lleva el mensaje de la ``libre elección" donde todo está bien, que puedes elegir si lo deseas una licencia privativa para los libros y músicas.

Se debe explicar por qué usar para libros y música las licencias CC del tipo libre y copyleft (share-alike) o la FDL [Stallman:FDL] y no las otras.

Intereses, ideologías, religiones y Software Libre

Todas las ideologías, incluso las religiones [Solar:SLP] son compatibles con los principios e ideales del software libre. El único motivo para apoyar al privativo es de intereses. Y los únicos realmente interesados en que siga existiendo software privativo son sus derecho-habientes, cada vez menos y más concentrados.

El esquema privativo es funcional a unas pocas empresas multinacionales que hacen cajitas de colores y las distribuyen por el planeta, contando con la fuerza policial y represiva de los estados del mundo - incluso los pobres a su disgusto, amenazados por distintas vías [Inquirrer:ITW-03]- , un negocio del tipo del recaudador de impuestos. No es funcional para el pequeño y mediano empresario, que vive de la relación directa con su cliente.

Ciencia, tolerancia, consenso y democracia o relativismo; Fe o fundamentalismo

Un debate interesante que cruza muchas de estas discusiones es la cuestión de qué elementos quedan afuera o adentro de los consensos globales. Así se puede vislumbrar un modelo social y político donde muchas ideas diferentes puedan convivir, canalizando conflictos, pero manteniendo una diversidad controlada, que garantice la estabilidad social, los privilegios y las jerarquías sociales [Connif:HNR-02,Waal:SAS-01], es una combinación de:

1. ética protestante del trabajo [Weber:EPE].
2. tolerancia, compasión, desapego, ponerse en el lugar del otro, equilibrios, democracia liberal representativa, voto universal, derechos humanos universales, libertad de prensa (basada en el mercado), control soft, control hard si es necesario.
3. cientificismo [Appleyard:CH-2003].
4. capitalismo económico, propiedad privada, multinacionales no localizadas.
5. organizaciones internacionales: ONU [Annan:QG], etc.. Democracia mundial basada en ONGs., gobiernos y empresas.

Estos conceptos forman la base de la moderna sociedad occidental. Todas las religiones son buenas si son tolerantes, si hay un camino común. La espiritualidad última. Las logias y sociedades secretas y gnósticas. Todas las ideas políticas son buenas, si respetan los derechos humanos universales y participan de la ``democracia".

La diversidad basada en la tolerancia de los demás: no hay absolutos, salvo la democracia y el respeto a las minorías. Toda idea debe ser respetada, en tanto no ponga en riesgo estas ideas políticamente correctas.

La democracia es vista como forma de proteger a las minorías de las mayorías, un sistema de balanzas y contrapesos para proteger a los pocos poderosos de los muchos débiles.

En un mundo donde no hay recursos suficientes para todos [Wilson:B], ni mucho menos recursos para que todos vivan ``bien", es esencial para impedir la emergencia de grupos igualitaristas que pongan en riesgo los privilegios de las jerarquías del momento. Este modelo excluye la existencia de proyectos políticos organizados con proyectos alternativos y en la búsqueda del poder real. Excluye la existencia de proyectos religiosos con una moral autorreferencial - a los que denominan sectarios-. Este proyecto excluye otros planes, es el meta plan, el plan mínimo, el no-plan, al cual todos deben amoldarse.

Solo permite partidos liberales o socialistas reformistas, ambos moderados, sin un plan global que ataque la alternancia, que acepten el plan de la ``democracia liberal" y que no desestabilice el progreso de los ricos. Con una izquierda con un respeto enorme por la ciencia. Este sistema es denominado ``relativismo moral" por sus detractores y los que apoyan al sistema denominan ``fundamentalistas" a los que se salen del mismo.

La cuestión a considerar es hasta qué punto la visión ética del software libre está dentro de este consenso o está fuera. ¿Es tolerable a la democracia occidental el software privativo?. Algunos opinan que no [Saravia:CH,Dutra:SFS-02,Busaniche:SCI,Chua:MLL-03]. Colocarse dentro es asegurar la victoria del movimiento en el mundo occidental moderno. Colocarse afuera es apostar a transformar radicalmente el planeta.

E. Aspectos comunitarios: modelo de desarrollo vs libertad del conocimiento

Calidad y Software Libre

Como el software libre da derecho a compartir y a la solidaridad, permite que se formen comunidades en la red, que lo desarrollan y lo mejoran [Ball:OMS-03,Martin:SRB-03].

En general las capacidades técnicas de un software desarrollado comunitariamente irán mejorando con el tiempo, si no tiene una cosa necesaria, la tendrá, indubitablemente.

La ``ventaja" fundamental de cualquier software libre con relación al privativo es de derechos y por ello permite la formación de equipos de trabajo comunitarios y por ello se va mejorando con el tiempo gracias a los esfuerzos de muchos.

Si bien el software libre se inició con un inmenso voluntariado y en muchos proyectos esta es la norma, en los proyectos centrales como el núcleo la mayoría de los contribuyentes son programadores a sueldo [Cohen:NRG-05].

LIBERTADES - DERECHOS HUMANOS =>(permiten la construcción)=> COMUNIDADES DE DESARROLLO/USUARIOS =>(crean) =>SOFTWARE DE ALTA CALIDAD

Esta cadena hace que desde 1984, cuando rms inicia el proyecto, hasta hoy el edificio del software libre continúe creciendo, cubriendo más aplicaciones y necesidades y mejorando técnicamente, hasta eventualmente llegar a su objetivo final: la supresión de todo software que limite los derechos de usuarios y desarrolladores.

Se identifica el modelo de desarrollo bazar con las virtudes técnicas del software libre en relación al no-libre. Dado que se dice que es mejor porque todos participan y hay una mayor y mejor realimentación con los usuarios.

Por otro lado, en definitiva la ética se fundamenta en que expresa los valores que son convenientes para la humanidad en su conjunto (materialismo).

Podemos explorar estas relaciones en:

Modelo de Desarrollo/Filosofía:	Libre	No - Libre
Bazar - Comunitario	Núcleo Linux	Imposible
Catedral - Vertical	Núcleo Hurd	MS Office

El software privativo es contradictorio con la participación y las comunidades.

El Software Libre remite no solo a sus postulados definitorios basados en las 4 libertades, sino también a las metodologías de desarrollo y a la ingeniería social y económica necesaria para construirlo:

- La creación de Internet usando software libre que a su vez permitió desarrollarlo. Herramientas, como la web, el correo electrónico, los chat, los repositorios de soft, etc.
- El desarrollo de metodologías de desarrollo como ``El bazar" contrapuesto a la ``La Catedral", o las llamadas metodologías ágiles (XP: extreme programming)
- La multiplicación ad infinitum de los LUGs, junto con organizaciones nacionales como Solar, o internacionales como Hipatia.
- Corporaciones y empresas de todo tamaño utilizando el software libre como estrategia comercial e invirtiendo dinero en su desarrollo
- Partidos Políticos como el PT Brasileño y Gobiernos como el de Brasil, Venezuela o Cuba apostando políticamente al mismo.
- Universidades formando desarrolladores e incubando proyectos.
- La revalorización de la persona en el mundo tecnológico. El homo tecnológico no solo consume tecnología informática sino que la puede crear en distintos grados. Se convierte en un ``prosumidor". Las herramientas están disponibles, solo necesita voluntad de aprender.

La libertad no es la única variable a discutir en relación con el conocimiento, también podemos considerar la cooperación. Evaluando la capacidad de generar modelos humanos sustentables de cooperación [Castells:IIT-05,Castells:ILP-05]

Hay otros parámetros o variables para clasificar el software: libertad, cooperación en los modelos de desarrollo, comercial, calidad técnica.

Este análisis deberá ser independiente, pero también podrá preguntarse si hay correlaciones.

¿Es posible un software comunitario no libre?. La libertad del software más el COPYLEFT es lo que impulsa y permite la construccion de una comunidad con la GPL como su constitución.

Así como existe toda una doctrina de la libertad del conocimiento, también puede existir en forma independiente una doctrina de la construcción colectiva y no comercial del conocimiento, pero la segunda reposa en la primera. Ya que si no hay derecho a compartir el conocimiento no se pueden crear comunidades.

Distribución	Libre	Privativo
Comercial	SuSE profesional via CD	MS Office
No comercial	OpenOffice via Internet	MS Explorer

El software libre no es contradictorio con el lucro. La cuestión del lucro es una cuestión independiente de la libertad del software .

Sin embargo no es, en general, sustentable cobrar licencias por software libre, debe financiarse el desarrollo por el lado de los servicios o por consorcios de interesados al estilo de mecenazgos o eventualmente el estado.

F. Neutralidad Tecnológica

Tecnología, derechos, cultura y neutralidad. Política de gobiernos y estados

Las elecciones tecnológicas modifican culturas, las elecciones de las comunidades afectan la tecnología.

Las comunidades no son neutras al elegir tecnología [Quiros:NTA], ni pueden serlo, lo hacen según sus intereses, deseos y límites.

En cuanto a los límites, las normas de ``Propiedad" Intelectual impiden o condicionan seriamente el ejercicio de la libertad de elección. Para la mayoría de la humanidad usar el software de Microsoft es una obligación, es lo que tienen en sus trabajos, en sus ``cibercafés", en sus escuelas, lo que reciben a través de documentos en formatos exclusivos [DiCosmo:TC-98], es lo que el estado les pide para interactuar, etc.. No tienen opción. Esa situación les impone restricciones adicionales y los obliga, muchas veces, a delinquir para poder usar determinado software, en una estudiada metodología de capturar mercados [Gates:ESA] y convertirlos en ``adictos". Así la gente no puede ofrecer resistencia en el marco de la ley, y violar la ley solo ayuda al monopolio. Este software no se puede inspeccionar, no se puede copiar ni menos compartir, no se puede usar libremente. No ofrece ninguna libertad, menos de elección.

Por otra parte, la cuestión del copyright del software y las distintas formas de licenciamiento, o las restricciones a la libertad[2.19] del software, tienen poco que ver con las tecnologías sino con quitarle derechos al usuario.

Las mismas tecnologías se pueden usar bajo cualquier modelo de licenciamiento o con distintos conjuntos de derechos. Existen ``implementaciones" del mismo software distribuidas una bajo modelo privativo y otra bajo licencia libre [Mysql:DLP]. El mismo código puede licenciarse de una y otra forma a ``libre elección", según convengan el derecho-habiente[2.20] y el ``licenciatario". Los usuarios y consumidores, los estados, o las empresas pueden requerir libremente el mismo software bajo modelos diferentes y negociar con los derecho-habientes de los mismos, las condiciones que desean y cuánto deben pagar por ello.

Una cosa es la tecnología de desarrollo o de funcionamiento de un software y otro la forma en que se lo licencia a terceros, en este caso al gobierno. La tecnología no tiene, en una primera aproximación[2.21], relación alguna con las relaciones contractuales entre licenciatario y licenciado.

Una política estatal que indique cuáles son los requisitos que deba tener el software a ser usado en el gobierno en cuanto a las formas jurídicas y económicas de licenciarlo, no impide o limita a ningún proveedor. Pues todos pueden negociar un precio por lo que el gobierno pide y competir libremente por satisfacer el requerimiento público.

Esta es pues la discusión de fondo. Qué modelo de licenciamiento garantiza perdurabilidad, transparencia, independencia del proveedor, seguridad, capacidad de auditoría, [Saravia:SLA-03, Ver 29 requisitos] etc., todos ellos requisitos incuestionables del software para el estado. ¿Deben los consumidores o el estado o las empresas estar atados a las condiciones de licenciamiento de un proveedor cualquiera?, ¿o podrán fijar las condiciones de compra según lo que necesiten?. ¿Deben limitarse a aceptar los contratos de adhesión de las empresas derecho-habientes?

Así pues los entes estatales adoptan varias posiciones: uso ilegal, acuerdos con Microsoft (generalmente ilegal, a veces forzado extrajudicialmente), licitaciones donde especifican Microsoft incluyendo sus marcas (Windows), neutralidad tecnológica, neutralidad completa (considerar condiciones de licenciamiento), política de compras a favor del estado (pedir código fuente, comprar para todo el estado en un solo caso, etc.), decisión por el software libre (normas promocionales u obligatorias).

En general los gobiernos europeos favorables oscilan entre neutralidad tecnológica y completa. Esta es la posición de la OSI. Hay gobiernos sudamericanos que han optado por el software libre como Brasil, Venezuela y Cuba. Hay numerosas ciudades que han hecho lo propio. Estas dos posiciones son las más debatidas entre los proponentes del Software Libre.

Otra tecnología o una cuestión de derechos. El Software libre es un movimiento social

Si bien ha producido muchas tecnologías, el software libre no es una tecnología en sí. Es una forma de distribuir software con el que sus desarrolladores dan derechos a los usuarios y a otros desarrolladores. Es un mecanismo legal para compartir tecnologías y conocimientos. Software libre es más una cuestión política, social y legal que una cuestión tecnológica.

En un mundo donde cada vez se construyen más muros, el software libre construye puentes.

Mediante estos derechos que otorga es posible construir una comunidad. El software libre es una creación político-legal para construir un edificio de software que permite la solidaridad y que puede terminar con los monopolios.

No se puede comprender al software libre si no se lo visualiza como un movimiento social. Como movimiento que es, busca realizar cambios sobre la realidad. Y madura, crece y se desarrolla con su propia identidad, más allá de sus integrantes, por brillantes que sean.

Ya nadie puede hablar de un fenómeno pasajero o una moda. Crece cada vez más dentro de las organizaciones, las universidades, las empresas y el Estado. Así también mayor y más comprometida debe ser la participación de la comunidad, y por ello mismo más abierta e inclusiva.

Monopolios vs. libres mercados

En el caso del estado u otras organizaciones con múltiples personas y oficinas, ¿pueden o no decidir qué quieren instalar el software en todas sus PCs?. ¿Pueden o no decidir qué quieren tener derecho de modificar, copiar o inspeccionar?. En tal caso, en las economías de mercado el precio debería ser fijado por la oferta y la demanda.

Esto no tiene nada que ver con tecnología o con la mentada ``neutralidad tecnológica". Tiene que ver con conveniencia económica y con los derechos de los consumidores y fundamentalmente con terminar con mercados monopólicos.

Donde no hay libre mercado, como en el software de oficina, sino monopolio, ¿cómo puede decidir el mercado?. Dejar que ``el mercado decida" es trabajar para Microsoft, ya que no hay otra decisión posible. En estos casos el papel del estado es construir libres mercados y terminar con los monopolios.

En muchos campos del software existe un monopolio y el único proveedor fija sus condiciones bajo contratos de adhesión. Hoy comienzan a existir alternativas, entonces los gobiernos empiezan a equilibrar el mercado y pueden fijar sus propias condiciones. Son los proveedores los que deben entonces cumplir los requisitos de los estados.

Así es razonable que un gobierno pida que el software que adquiera pueda ser usado e instalado libremente en cualquier computadora de su propiedad. Corresponde al proveedor ofertar un precio para

tal requerimiento. Es razonable que el gobierno pida acceso y derecho a recompilar y modificar el código fuente. Son los distintos proveedores quienes ofertarán el precio justo para este requerimiento.

Lo que sucede hoy con el software es que los usuarios han ganado derechos con el comienzo del fin del monopolio. A partir del software libre, nos empezamos a imaginar condiciones de libre mercado y la competencia permitirá bajar los precios y hacer que los proveedores deban entregar más para permanecer en el mercado.

Que los gobiernos fijen condiciones que amplíen sus derechos, con relación al software, sólo hace más competitivo y transparente al mercado, ayuda a terminar con monopolios [Malone:RIP-05] y favorece la baja de precios. No se excluye a ningún proveedor, solamente se piden los requisitos que el software público debiera tener.

G. Modelos de negocios

Gratis vs. Libre. Aspectos comerciales del Software Libre

El software libre se refiere más a libre expresión que a la cerveza gratis.

Así Richard Stallman (RMS) indica que nada en la filosofía del movimiento impide los negocios ni que este es anticapitalista. El software libre comercial existe y esta bien que así sea.

Por otro lado si bien existirán casos especiales excepcionales, para la inmensa mayoría del software libre sus licencias se obtienen sin costo alguno. Es cierto ``LIBRE'' no siempre es gratis, pero sí lo es en el 99% de los casos prácticos en cuanto al licenciamiento. Si bien se podría cobrar por la licencia, el que la usa puede a su vez redistribuirla, por lo que éste negocio no parece tener sentido para software de uso masivo. Es importante destacar que también es posible vincular la distribución de software libre con diversos servicios comerciales no gratuitos: descarga, soporte, mantenimiento, actualización vía CD-ROMs, etc..

Así cuando se analizan los efectos económicos de las libertades del software se llega a la conclusión que no es posible obtener ``renta'' por el licenciamiento del software libre. Así no es un capital. Su creación no produce renta.

Si cualquiera que lo reciba (aun mediante pago) puede redistribuirlo es razonable -si es útil a muchos- que el precio de su redistribución tienda a cero (o al costo material de la misma). Será entonces posible tener un negocio de redistribución de medios, en el marco del libre mercado, lo cual garantiza márgenes bajos. Los buenos negocios posibles se relacionan con la rama de los servicios. En esta rama se obtiene una paga por el trabajo efectivamente realizado y no una renta. Así el principal medio de apropiación de la riqueza es inefectivo en el mundo del software libre y este proporciona un modelo de negocios justo. No hay renta ni monopolio posible. En una economía de libre competencia la ganancia a esperar es baja.

Nadie puede esperar crear algo y sentarse a cosechar. Debe trabajar. No parece razonable esperar grandes inversiones de capital en el desarrollo del software.

Por otro lado el desarrollo de software puede pensarse en dos niveles: La invención creativa, innovativa y artística que se da en muy pocos casos, cuando se crea algo realmente nuevo: la web, el correo, el protocolo IP, el chat, etc.. Y la cuestión ingenieril, artesanal o técnica de adaptar el software, mejorarlo, etc.. El primer caso implica un modelo artístico, de genios con alguna idea que la realizan en el marco de una inspiración momentánea, y son muy escasos. Un buen mecanismo de selección de proyectos puede llevar a financiar la profundización de estas ideas. Pero al ser el software libre solo se puede esperar financiación pública o del tipo de las ciencias básicas, o un mecanismo de mecenazgo mediante fundaciones de empresas vinculadas por los servicios. Se trata fundamentalmente de ciencia.

Los servicios representan un trabajo tradicional, que puede ser asalariado o profesional y se adapta a reglas y métodos con posibilidad de estandarizar y costear. En tanto sean adaptaciones necesarias caso por caso se puede esperar cierto retorno por realizarlas [Cowley:PCB-02].

Es razonable que los gobiernos inviertan dinero en financiar al software libre a través de los mecanismos que tienen para el desarrollo de las ciencias básicas.

Pero por otro lado la academia llega tarde al software libre. La mayor parte de sus genios son jóvenes que todavía no entraron a la universidad y que no suelen adaptarse a los rígidos métodos de la academia.

Por otro lado Microsoft mantiene cautivas a muchas instituciones educativas pobres, mediante un sistema de licencias permisivo y las introduce en la enseñanza de sus productos como cajas negras. Degradando académicamente a las instituciones que se prestan a este juego bajando el nivel de sus egresados al de meros técnicos operarios.

Hoy la mayor parte del software libre se mantiene y desarrolla mediante consorcios de varias grandes empresas que comparten sus gastos. No lo ven como una inversión que genere renta autónoma sino como una forma de garantizar el control sobre algo que le es útil para su modelo de negocios o simplemente como una forma de promoción de su marca.

El tiempo del joven hacker en su garage ha pasado rápidamente, existe para nuevos desarrollos, pero los grandes proyectos como apache y el núcleo son altamente profesionalizados.

El software libre no es contradictorio con los negocios, pero sirve para cierto tipo de negocios y no otros.

Fundamento económico del modelo

El software libre se basa en un modelo en el que cada persona da uno y recibe mil. No lleva contabilidad de todo[2.22]como en el propietario.

En el libre uno parte de una base común enorme: Sistema operativo, desktop, base datos, etc.. Hace su aporte y puede cobrar por los servicios de integrar todo. En el modelo propietario uno paga por todo lo que usa y puede cobrar por la pequeña porción que aporta.

En el soft privativo no se comparte, uno invierte y después distribuye a miles, millones y acumula fortunas, no distribuye el costo, construye monopolios. O si le va mal y pierde, pierde todo.

No tiene nada que ver lo que finalmente se recauda con lo que costó el software. Es un sistema altamente ineficiente, monopólico.

Con el soft libre se puede hacer consorcios, o no, puede distribuir costos o no, cada actor puede elegir. De hecho suele distribuirse el costo.

Muchos soportan el Sistema Operativo, muchos otros el desktop, otros el motor de Bases de datos. Y el integrador final aporta su granito de arena, hace su microinversión y aporta un pedacito del edificio, a cambio recibe todo el resto.

Conclusiones

Naturaleza

Para analizar la naturaleza del conocimiento podemos pensar tres categorías o bien tomar la posición de RMS:

Concepto	**Típico**	**¿Bien económico?**	**¿Duplicable?**	**Quién decide**	**Ejemplo en el sistema deformado actual**
Lo libre	ideas	No	Sin costo	cada ``poseedor''	Software bajo copyright tipo copyleft
Lo compartido / común / público	bienes comunes	Sí, fuera del mercado	Con costo (artificial para ideas)	el conjunto mediante algún sistema	Estándares: todos usan, nadie cambia por sí, se decide mediante ``representantes''
Lo individual / privado	bienes de cada individuo	Sí, en el mercado	Con costo (artificial para ideas)	el dueño	Software bajo copyright tipo EULA de Microsoft. Patentes.
Disperso (no hay ``Lo'').	No hay un concepto que globalice los derechos de las personas en relación a las ideas o sus expresiones, Posición de RMS. Copyrights, patentes y otros son diferentes.				

La visión ``Dispersa'' nos dice: ``No hay nada de que hablar''. Las otras tres nos dicen que podemos pensar un concepto integrador de la mano de la economía. Y en esta situación vemos dos posibilidades de respuesta a la pregunta: ¿hacemos escasas a las ideas? Si la respuesta es no, estamos en el concepto

de ``Libre". Si la respuesta es sí, estaremos en la situación actual, donde las ideas o sus expresiones son artificialmente convertidas en bienes. Y allí podremos darle la ``propiedad" a la humanidad en su conjunto o a las personas, o pensar caminos intermedios, o dar la posibilidad de la ``libre elección" como en el caso de las Creative Commons.

Tres posiciones típicas en el movimiento: Disperso, Comunitario/Público o Libre. Muchas veces asumidas sin analizar sus diferencias. Las posiciones Libre y Dispersa son altamente compatibles. La visión ``dispersa" se fundamenta en la libertad de expresión y defiende la libertad de las ideas ``funcionales" de los regímenes de copyright abusivos, patentes y sistemas de DRM, aunque no se interese por el conjunto de sus efectos económicos. Ambos sectores coinciden en esta lucha. Pero no todos coinciden con defender la libertad de conocimientos ``no funcionales". La posición ``pública" o ``comunitaria" coincide con la libre en todas sus posiciones, pero sus impulsores no han analizado a fondo la naturaleza de sus concepciones.

En definitiva la visión que puede superar y comprender a las otras es la ``Libre" que parte de la naturaleza del conocimiento y que puede con facilidad interpretar las consecuencias económicas y los debates éticos involucrados.

En tanto los que consideran que el conocimiento debe ser privado lo analizan desde la perspectiva de sus propios intereses o desde la idea que su privatización es conveniente pues alienta ``el progreso" al crear un incentivo económico para la creación de conocimiento.

Así las contradicciones fundamentales de las dos últimas revoluciones que parieron nuevos tipos de sociedad fueron:

- En la revolución industrial la lucha fue entre la propiedad privada y la propiedad común de los medios de producción.
- En la revolución informática y las sociedades del conocimiento, la lucha es entre el conocimiento privado o libre. Propiedad privada o libertad del conocimiento para sus medios de producción y/o creación.

Ética o Moral

La cuestión central de esta discusión es si es aceptable proteger legalmente la existencia de software y conocimiento propietario, al cual la humanidad no puede acceder. A diferencia de la música y los libros, las leyes actuales protegen a las multinacionales de software al punto de alentar la distribución de ``obras intelectuales" en formatos binarios que sus usuarios no pueden leer o conocer.

Si se llega a la conclusión de que es razonable continuar manteniendo estos derechos diferenciales a las empresas de software, ambos tipos de software estarán en el mismo plano ético, y solo podremos hablar

de promover al software libre (o incluso siquiera eso). En tal caso la idea de que todo el software debe ser libre será una cuestión condenada bajo el mote de fundamentalista.

Si se considera que la ley no debe proteger con su poder de policía al sistema de distribución comercial que oculta la obra intelectual, entonces se deducirá que el software propietario no es ético y por lo tanto sale fuera del marco de consenso racional de las sociedades democráticas. En este caso se seguirá protegiendo el derecho de los autores/editores que publican la obra real. Pero no a quienes la ocultan en formatos binarios.

Por todo ello los peores enemigos del movimiento son aquellos que denominan como fundamentalistas las ideas éticas del Software Libre. Pues lo que pretenden, ni más ni menos, es sacarlo del consenso democrático de la sociedad occidental y cristiana.

Conveniencias

Existen muchas razones que impulsan a pensar que el Software Libre es conveniente ante el privativo en muchos casos, y según múltiples intereses.

La realidad indica que el mercado está mostrando en los hechos estas circunstancias. Por ello ahora las corporaciones de Software privativo están apuntando a los estados para subsistir. Cuando piden neutralidad tecnológica en realidad están pidiendo competir sin que se tengan en cuenta algunas de las abrumadoras ventajas del Software Libre en áreas no tecnológicas.

Votando tecnologías

Cada vez que una persona elige un producto [Saravia:GI-04], está votando en los mercados, cada vez que ejerce una opción tecnológica está votando por ella. Los administradores de redes de las Universidades Americanas votaron por Internet y con esa decisión la impusieron como red mundial global. Perdieron las elecciones Novell con IPX, IBM con SNA, Microsoft con NetBios, etc.

Hoy es la hora del software, cada voto cuenta, cada elección define la batalla por la libertad del Software.

Es también la hora de compartir archivos, cada nuevo desarrollo que permite compartir música y ``contenido" inclina más la balanza hacia un mundo mejor, más próspero [Rehermann:NMP] y con más libertad.

Así tendremos más prosperidad intelectual en un mundo de conocimiento libre [Saravia:REC-04,Saravia:DDS-03,Saravia:EI-03] que en otro cerrado y privativo.

Los caminos del Software Libre

Ante los desafíos planteados hoy la humanidad, con su revolución humanista, centrada en el ser humano, tiene tres caminos para consolidar la liberación del ciberespacio, la revolución computacional y la globalización popular:

- **Político:** Promover cambios en la legislación de copyright, incluso eliminarlo[2.23], evitar el DRM, y la penalización de la copia y en definitiva de Internet. Impedir el establecimiento de patentes de software. Luchar por un desarrollo no signado por la escasez en un ámbito donde es posible. Dictar leyes de uso y creación de software libre en el estado, junto con leyes para administrar las obras intelectuales del estado. Promover decretos ejecutivos y migraciones en el Estado, o en el sistema educativo. Participar de encuentros y eventos de todo nivel en cualquier lugar del planeta. Procesar a Microsoft por acciones monopólicas.

  Una cuestión central es si los gobiernos deben usar Software Libre o si pueden preferirlo o si deben elegir caso por caso [Romero:OPT-05,Saravia:SLA-03].

- **Criminal [Lanier:PF-99]:** Utilizar software ilegalmente, copiar música por Internet. Este es el camino de la mayoría de la humanidad, constituye la costumbre del uso de las TICs en el planeta. Ilegal pero moralmente válido. No es recomendable y es un grave error a largo plazo. Como dijo en su momento Bill Gates [Gates:ESA] y fue citado por Amadeu [Microsoft:ASA-2004], los usuarios primero usan gratuita e ilegalmente su software, luego se convierten en adictos, y a posteriori encontrarán la forma de cobrarles.

  Curiosamente la ley intenta cambiar el uso comúnmente aceptado, mientras que la ley debiera ordenar estas costumbres.

- **Alternativo:** Desarrollar Software Libre. Construir un nuevo edificio. El camino o plan propuesto por Richard Stallman (rms). Para cada aplicación una alternativa. Otro mundo es posible.

  Las patentes de software pueden destruir este camino y en tal sentido no se puede abandonar el camino político. La universalización de las patentes es el plan de las multinacionales de software.

Con relación a otros tipos de conocimiento, las cosas son más complejas. El resultado de la batalla por el software libre determinará en gran medida las otras batallas de la guerra por el conocimiento libre.

Hoy existe un gran movimiento alternativo, que le da a todos una poderosa herramienta de lucha para crear otro mundo, ya no sólo posible, sino uno que estamos construyendo, convirtiendo a cada computador y mente en una trinchera.

Metodologías emergentes para la creación de software

> El vicepresidente de Microsoft Jim Allchin caminó hacia la oficina de Bill Gates en julio de 2004 y le dijo que Longhorn estaba horriblemente retrasado y que nunca se recuperaría. ``No podrá funcionar", dijo, de acuerdo al informe de ``The Wall Street Journal". El problema es que Vista es demasiado complicado y los antiguos métodos de desarrollo de Microsoft no son suficientemente buenos.

> [Richards:MWO-05]

> Las teorías deben ser lo más simples posible, pero no más que eso.

> Albert Einstein.

Subsecciones

Introducción

Este trabajo pretende ayudar a quienes se plantean realizar migraciones hacia el Software Libre en organizaciones en cuanto a las metodologías de desarrollo.

Presenta brevemente el concepto de Software Libre y los modelos de desarrollo emergentes en su construcción. Se plantean algunos patrones considerados claves para el desarrollo de Software Libre.

El trabajo apunta a numerosa bibliografía en la Web que puede ayudar a profundizar cada tema y a conseguir información técnica detallada de cada aspecto en consideración.

El Software Libre

El Software Libre [FSF:DSL] expresa una ética condensada en las 4 libertades que definen las prácticas de sus comunidades: uso, inspección, modificación, distribución libre y en los derechos que sólo estas prácticas garantizan, a la: cultura, educación y comunicación libres [Hipatia:SM-04]. Prácticas que conforman una ética del trabajo muy diferente a la protestante capitalista de la era industrial [Merten:GHS-00,Himanen:EHE-02,Bard:NNP-03].

Su creación fue sustentada por Internet, que fue construida en el proceso. Esta red es la que permite compartir y liberar el conocimiento en las actuales sociedades del conocimiento, que así deja de ser un bien escaso y puede fluir y reproducirse libremente.

Creado mediante una dinámica o metodología colectiva -práctica habilitada por las 4 libertades e Internet- permite a la comunidad de prosumidores[3.1] participar en un marco solidario de construcción del conocimiento [Saravia:MH-01].

El modelo de desarrollo libre, participativo y comunitario

Introducción

La metodología participativa con la que se construye habitualmente el Software Libre se reconoce como técnicamente superior [Ball:OMS-03] a las metodologías antiguas y tradicionales de la ingeniería del software que Raymond [Raymond:CB,Raymond:AUP-03,Raymond:MC] sindica como fuente de su fuerza. La comprensión de esta dinámica, su real alcance, características y uso en las comunidades de desarrolladores es fundamental para tener una visión completa del fenómeno del Software Libre.

Podríamos personificar en Stallman (FSF) vs. Raymond (OSI) la discusión sobre qué es lo que más importa en el Software Libre, si la ética metafísica de sus libertades, o la dialéctica materialista (tanto comunista como capitalista) que lo afianza día a día en las comunidades y mercados del planeta. Pero más allá de ética versus conveniencia o de libertades y derechos versus realidades y prácticas, ambos aspectos están íntimamente vinculados y se refuerzan mutuamente.

Las comunidades del Software Libre han construido poderosas herramientas y técnicas. Internet y sus diversos protocolos son los más notorios derivados de este trabajo. Estas herramientas fueron y son

cruciales y su crecimiento determina el presente y el futuro del software en el planeta. El Software Libre se edificó y conformó a sí mismo construyéndolas. [3.2]Alrededor de cada desarrollo y creación, las comunidades de desarrolladores inventan protocolos y crean software.

Las comunidades suelen funcionar como meritocracias donde el que más trabaja más capacidad de conducir y decidir tiene. En última instancia los desacuerdos y conflictos se resuelven creando ``forks" o nuevos proyectos derivados.

Muchos de los casos paradigmáticos del Software Libre se construyen mediante consorcios de organizaciones, empresas y personas que aportan recursos para sostener el esfuerzo. Así la mayoría de los aportes al último núcleo ``Linux" fue realizado por trabajadores de corporaciones [Jackson:LCB-04].

Algunas de las estrellas típicas del movimiento son: las herramientas ``GNU", por ejemplo el compilador ``GCC", el núcleo ``Linux" [LinuxK:SW,LinuxO:SW,LinuxC:SW], el servidor web ``Apache" [Apache:SW], el servidor de archivos ``Samba" [SAMBA:SW], el directorio ``Open LDAP", el paquete de oficina ``OpenOffice" [OpenOffice:SW], los escritorios ``KDE" [KDE:SW] y ``Gnome" [GNOME:SW], los motores de bases de datos ``MySQL" [MySQL:SW] y ``PostgreSQL" [Postgres:SW], y el navegador ``Mozilla" [Mozilla:SW][3.3].

El movimiento de Software Libre ya nuclea más personas participando de su construcción que el software propietario trabajando[3.4] para ello.

Crítica de las antiguas metodologías para crear software propietario

En los modelos tradicionales de desarrollo e ingeniería de software [Gibbs:SCC-94,Oreilly:I-05,Robles:IS-02] que funcionan para productos licenciados en forma propietaria, el proceso comienza con el análisis de las necesidades de los usuarios por analistas del negocio. El ingeniero de software recibe los requerimientos y prepara las especificaciones. El administrador del proyecto asigna tareas a los desarrolladores distribuyendo el trabajo según las especificaciones. Los desarrolladores escriben el código pasando por las etapas de desarrollo y se prueba hasta que se instala. El usuario certifica y aprueba la funcionalidad. Luego el grupo de desarrollo muta para ser una unidad de mantenimiento que corrige los problemas y genera modificaciones sobre demanda. Para una nueva versión se comienza el ciclo de nuevo.

La estructura de costos de uso de este software pasa por tres componentes: L (licencia), R (renovación y mantenimiento), S (soporte), donde R+S suele ser el 20% de L [Murugan:PSO-05].

Este modelo es lento, tiene altos costos iniciales y de mantenimiento, y el trabajo se repite innecesariamente múltiples veces, ya que todo debe reescribirse de nuevo o en caso contrario pagar licencias a los que lo hicieron antes.

Los desarrolladores están controlados estrechamente y suelen firmar acuerdos de no divulgación. Se dificulta el intercambio de información, lo que suele implicar represión en el lugar de trabajo. Preguntar es un disvalor, pues muestra lo que no se sabe y baja la posición del que pregunta, quien es considerado peste, pues pone en evidencia lo que no se conoce. La lucha por el posicionamiento político suele depender de las relaciones y el manejo de las jerarquías verticales.

Características generales de las nuevas metodologías

En los modelos participativos y comunitarios de desarrollo de software, utilizados generalmente para Software Libre, los administradores organizan el proyecto, y reclutan o aceptan colaboradores, técnicos, revisores, documentadores, especialistas y/o contribuidores esporádicos.

Apenas el proyecto ha creado una base funcional de tests y código, puede comenzar a recibir aportes de terceros, aunque no pertenezcan al grupo que tomó la iniciativa.

Es un modelo dinámico y colaborativo donde el conocimiento colectivo se expande exponencialmente.

Permite que una gran cantidad de personas se involucren eficientemente, sin caída del rendimiento ante el aumento de la escala, como en los proyectos propietarios. A pesar de ser visto como un sistema anárquico, permite un manejo profesional, seguro y apropiado, con mejoras continuas y manejo de calidad. Todos participan de todo el proceso, y no se forman cuellos de botella, pues si un aspecto parece detenido inmediatamente más personas se abocan al problema.

El modelo participativo construye un cuerpo creciente de conocimiento y reduce tiempo y costos. Existen grandes comunidades en línea de donde se obtiene soporte.

En cada proyecto se toma 1000 de la comunidad y se aporta 1 en retorno. Lo que funciona porque todos toman lo mismo -el conjunto- y aportan algo diferente.

Impacto sobre las metodologías de intercambio y la economía

El Software Libre no se constituye como un conjunto de productos. Lo que se ofrece y demanda son servicios. El software suele estar disponible.

El proceso de adquisición de software es muy diferente cuando se usa Software Libre. O los equipos de la organización usuaria lo instalan y lo usan, o se contratan equipos de personas a tal efecto. En este caso el producto es el tiempo del equipo de personas, no el software.

Cuando se piensa en Software Libre, no se puede pensar entonces en comparar ``productos'' de software para elegir y poner una larga lista de softwares propietarios y libres para evaluar, sino que se debe entrenar o ``adquirir'' un equipo y este tomará a su cargo la selección del conjunto de software apropiado, en muchos casos lo modificará, o pedirá su modificación. En muchos casos no hay fronteras claras entre el desarrollo y la instalación o adecuación.

Patrones para el desarrollo de software

Ideas, conceptos y principios para el desarrollo de software, redactado como un ``lenguaje de patrones" [Alexander:ULP-77].

Patrones que reflejan parte de la cultura Unix; ``KISS: Keep it simple stupid"; el arte de hackear por placer, o programar; con las ideas que se han ido creando en la construcción y uso masivo de Internet.

Prácticas que han conformado la más grande aventura colectiva de la humanidad en cuanto a creación de conocimiento.

Los patrones no son absolutos, en algunos casos se adoptan unos y no otros. Algunos contradicen a otros.

Referencias: [Fogel:POS-05].

Colaboración habilitada por tecnologías de Internet

.

Comunidad y cultura a través de Internet.

Del modelo propietario donde cada proyecto es un comienzo a un modelo basado en un edificio creciente de experiencias compartidas. Con enorme potencial de trabajo en paralelo. Los proyectos pueden tardar semanas en tener programas útiles y no meses.

El equipo de desarrollo consiste en administradores, mantenedores, contribuidores de código, técnicos en releases, revisores y testeadores, documentadores, etc.

Los administradores del proyecto controlan el almacenamiento del código en un sistema de control de versiones concurrentes como el ``subversion".

Todo el equipo, ``el público", ve los cambios y puede comentarlos. Todos pueden tener versiones y ``snapshots" del código, probarlos y hacer sugerencias.

Las propuestas de código se envían al administrador del proyecto para su revisión y control. Este puede asignar a otro del equipo la tarea de revisión, o incluso compartirlo en una lista para su revisión por muchos. El código será testeado y revisado y si es aceptado incorporado al proyecto mediante el sistema ``subversion".

Es permanente y constante el envío de mejoras, la revisión y eventual selección de las mismas para su incorporación.

Nadie tiene el control total. Todos tienen los ojos en el código y pueden comentar las acciones de los administradores.

Se trabaja mediante aproximaciones sucesivas e iterativas. Se establecen reuniones IRC (Internet Relay Chat) semanales, diarias o incluso permanentes. Se hacen preguntas y se dan respuestas por correo. Se comparte información, código y protocolos por diversos medios. Se descubren y se forman especialistas. Y se publican y registran los proyectos en sitios web comunitarios.

Es una disciplina particular de manejar proyectos surgida de Internet. Una clave para el desarrollo del Software Libre consiste en que los equipos accedan y aprovechen Internet a fondo. Esto requiere buen y constante ancho de banda. Internet posee un enorme cuerpo de conocimiento para los desarrolladores. Entre otros los archivos con los ``RFC'' de la ``IETF'', las listas de correo de proyectos similares, grupos de google, etc. Participar en los mismos, preguntar y responder, etc. Internet es el ``help desk'' del desarrollo libre. Pertenecer a las listas de correo, grupos de noticias y foros adecuados es esencial.

Para que una organización adopte el Software Libre es fundamental:

1. promover las condiciones para el cambio de la cultura organizacional y de su política de seguridad.
2. garantizar la libre distribución del Software Libre de forma colaborativa y voluntaria, utilizando sitios web específicos, distribuciones en discos compactos específicas, software de manejo de proyectos.
3. apostar a que la gente comparta el conocimiento que vaya adquiriendo, en salas comunes, etc.
4. desarrollar un ambiente colaborativo que promueva la cultura del Software Libre.
5. tener a toda la organización conectada a una buena red con amplio e irrestricto acceso a Internet.

Valores

.

El desarrollo participativo expresa valores entre los que están la máxima simplicidad posible, el diálogo abierto, la realimentación permanente, y el coraje para efectuar cambios.

Modelo de relaciones económicas

.

Cuando se busca migrar una nueva actividad y se necesita para ello crear una gran cantidad de aplicaciones libres, es conveniente involucrar a la comunidad de desarrolladores independientes y habilitarlos económicamente. Tener un plan de acción concreto y pensar en que será lo que reciba la comunidad en contraparte. Así construir un modelo de relaciones económicas apropiadas [Chance:SSO-05] que sustente a esta comunidad de desarrolladores.

Es importante promover y ascender a los empleados que conozcan de Software Libre y colocar a los expertos preexistentes en los niveles jerárquicos apropiados para divulgar, promover y capacitar a los demás funcionarios en su uso y cultura.

Patrones específicos para gobiernos

.

Acciones ejecutivas:

1. Involucrar a la comunidad del Software Libre en el proceso de adopción del Software Libre por el Estado.
2. Ampliar la oferta de servicios prestados al ciudadano a través de Software Libre. Promover la migración y adaptación del mayor número de aplicativos y servicios a plataformas abiertas y Software Libre.
3. Fortalecer y acompañar las acciones preexistentes de Software Libre dentro y fuera del gobierno.
4. Diseminar la cultura del Software Libre en las escuelas y universidades. Realizar convenios de colaboración para desarrollo, implantación, formación de funcionarios y educación en Software Libre. Apoyar la investigación y el desarrollo de Software Libre en el complejo de ciencia y técnica, universidades, etc.
5. Masificar el uso de Software Libre en el Estado, y especialmente en sus proveedores.
6. Realizar campañas de información pública para difundir las características del Software Libre en la comunidad, la sociedad civil y las empresas.
7. Usar Software Libre como base de los programas de inclusión digital.
8. Incentivar y fomentar al mercado nacional para adoptar nuevos modelos de negocios en tecnologías de información y comunicación basados en Software Libre.
9. Promover la capacitación y formación de técnicos y otros funcionarios públicos para el uso de Software Libre.
10. Articular la política de Software Libre con una política de fomento de la industria y expansión de sectores económicos vinculados. Esta política debería incluir créditos a empresas para migración, desarrollo de aplicativos, incentivo a las exportaciones, derivación de impuestos a acciones educativas, etc.

Acciones normativas:

1. Poner en vigencia las normas técnico legales de uso de software en el Estado en los distintos niveles normativos según su nivel: legislativo, ejecutivo, etc.

2. Establecer las condiciones de uso de software en general, haciendo especial énfasis en los requisitos de transparencia, legalidad, accesibilidad a la información, y otros que llevan naturalmente al uso de Software Libre.

3. Establecer límites a las condiciones sobre las licencias que se pueden adquirir. No permitir que las empresas fijen contratos de adhesión. Asegurar que el Estado sea el que fija las condiciones, normas y contratos de adhesión. El propio Estado debe fijar sus esquemas de negociación en materia de Software.

4. Prohibir taxativamente las patentes de software si no lo están. Si lo están, denunciar acuerdos internacionales al respecto.

5. Revisar tratados internacionales en materia de derechos de autor.

6. Eliminar y denunciar acuerdos internacionales que fijen penalizaciones arbitrarias.

7. Impedir y prohibir la adopción de tecnologías que detengan, impidan o condicionen la ejecución de códigos en cada computadora más allá de la voluntad del usuario (Trusted computing, TCG [Anderson:PFI-03,Eff:TC,p2pnet:TPE,TCG:SW]), transfiriéndosela a las corporaciones internacionales.

8. Condicionar las restricciones de copia (copyright) a que se conozca el código fuente, al igual que en los libros y la música. Restringir el alcance del copyright en el caso de código cerrado.

9. Crear legislación específica que constituya una categoría legal diferente para el Software Libre independiente del copyright, las marcas y patentes.

10. Legislar para que el software producido o contratado por el Estado sea GPL [Stallman:GPL].

Patrones para cambios sociales

.

Instalar Software Libre en los hogares y pequeñas organizaciones sociales es un desafío particular. Es conveniente abordarlo con conceptos profesionales de campos como la Antropología y Ciencias Sociales.

Es un cambio humano y cultural. No se trata de cambiar algunas tecnologías por otras. Se habla de formatear cerebros, y no sólo discos duros.

Este tipo de cambios sociales eran antes motorizados con la escuela. Ante la regresión o la inexistencia de la escuela como esquema de adoctrinamiento nacional y social en el Tercer Mundo, se deben buscar nuevas formas de acción social y nuevos agentes de cambios.

Se debe visualizar la inserción social de tecnologías apropiadas como un fenómeno integral y global que comprende entre otras a: las energías alternativas, la cocina alternativa, la informática alternativa -software libre-, los sistemas ambientales alternativos, etc. Es un camino diferente a la

vía ``hippie" de constituir pequeñas comunidades. Cambios globales para las mayorías y no aldeas o grupos especiales. Así el conocimiento libre puede abrazar a otros intereses y no sólo a los vinculados a las patentes medicinales y los transgénicos.

Todo tipo de cambio que transforma una tecnología que requiere grandes flujos comerciales y energéticos centralizados o con control centralizado, grandes inversiones y ganancias para grandes corporaciones por otra que pueda activarse localmente en cuanto a bienes y servicios aunque requiera información global.

Así conocimiento libre globalizado más flujos materiales y energéticos localizados, parece ser la consigna para salvar al planeta y a las sociedades humanas. Para terminar con la globalización neoliberal y el modelo de abrir fronteras para maximizar los flujos comerciales globales. Otra variedad del ``Pensar global, hacer local" [Mitchell:ALA-99].

Exigir o priorizar Software Libre

.

Definir si se establece que se usa Software Libre con exclusividad o si solamente se decide priorizar soluciones, programas y servicios basados en Software Libre que promuevan la optimización de recursos e inversiones en tecnología de información.

Garantizar transparencia

.

Garantizar la auditabilidad plena y la seguridad de los sistemas, respetando la legislación sobre privacidad y seguridad.

Garantizar seguridad

.

En muchos casos el Software Libre permite mejorar notablemente la seguridad, dando a la vez mayor libertad. El Software Libre hace parecer al software propietario como un vector para transportar virus y otras alimañas. Hay muchos otros factores que hacen más seguro al Software Libre, entre ellos la posibilidad de recompilar todo el software y no tener que mantener compatibilidad binaria con antiguos ejecutables.

Es conveniente el cambio de una política de seguridad basada en protecciones o barreras a otra basada en protocolos seguros por naturaleza en cada recurso. Salir del modelo de seguridad mediante la oscuridad. Terminar con la necesidad de poner cercos inútiles que castigan y restringen a quien no es necesario restringir, para que dejen de pagar los santos por los pecadores.

Política con proyectos preexistentes

.

Durante la transición al Software Libre aparecerán nuevos proyectos y se estará en diversos grados de avance en relación a proyectos propietarios predefinidos. Se debe explicitar una política definida con los mismos y un mecanismo para definir excepciones.

Como mínimo se debe restringir el crecimiento de los sistemas preexistentes y nuevos basados en tecnología propietaria.

Se debe buscar interoperatividad con sistemas preexistentes heredados.

Migraciones

.

Se debe evaluar cuándo migrar un sistema, usar el sistema anterior en una nueva base tecnológica, crear un nuevo sistema, etc.

Para la migración de los sistemas propietarios preexistentes se puede graduar la velocidad en relación con la capacitación.

Se puede pasar de una situación con islas de Software Libre, a otra donde existen islas de software propietario inmersas en un mundo de Software Libre.

Crear software universal y gratuitamente disponible bajo licencia GPL [Stallman:GPL]

.

Se debe liberar el software sobre el que se tengan derechos. No sólo se debe hacer Software Libre sino también debe ser gratuito y debe ser liberado en forma universal. El modelo de negocios no pasa por el código sino por los servicios que se construyan sobre el mismo.

También es conveniente establecer que el software creado se libere bajo la GPL [Stallman:GPL] (comunitaria). Utilizar la licencia GPL [FSF:QC-98,Varios:CSC][3.5] permite crear una cultura solidaria sin vampirismo.

Cuando se libera software que antes era privativo se debe buscar problemas de patentes, copyright. [3.6]

Es recomendable consolidar una organización como derecho habiente del código, aún del recibido desde terceros. Esto debe estar claro en las comunidades que participen del desarrollo.

Liberar el software preexistente es un hecho trascendente, debe ser enmarcado mediante una conferencia de la cabeza de la organización anunciando que todo el software del cual es derecho habiente será liberado bajo la GPL [Stallman:GPL], en forma universal.

Esto se podrá hacer uno por uno, por parte de los responsables de cada software, y se irán disponibilizando en el sitio web respectivo.

Se puede dar un plazo a cada responsable, para que se evalúe: a) el estado del código fuente y lo preparen para publicar, b) retiren información confidencial que pueda existir en las mismas, c) evalúen si existe alguna incompatibilidad legal con la GPL [Stallman:GPL] y el código usado y sugieran otra licencia libre en tal caso.

En tres meses todas las aplicaciones debieran tener este trabajo concluido, salvo alguna excepción a ser evaluada.

Se evaluará cuál aplicación debe ser mantenida para el actual nivel de uso y por lo tanto colocada en un sistema ``subversion'' y cuál ya no se toca más el código y queda congelada.

Contribuciones

.

Sea generoso en lo que se acepta en el proyecto, riguroso en lo que emite.

Ciclo de Desarrollo

.

El ciclo de desarrollo suele ser el inverso al del modelo tradicional, con un modelo de recirculación permanente del tipo cascada.

Idea.

Se define la idea del proyecto, se buscan las alternativas preexistentes, y el software sobre el cual se puede fundar la capa lógica a crear o a adaptar.

Tests.

Se debe testear todo permanentemente, lo primero a crear para cada funcionalidad son sus test.

Los programadores deben escribir test unitarios todo el tiempo. Estos deben correr sin problemas en todo el ciclo de desarrollo. Los usuarios pueden escribir tests probando que todo funciona.

El desarrollo es inducido por tests. Primero se escribe el test, luego el código. Hasta que todos los test se pasan no se da por terminada la rutina. Cuando los pasa y no hay más tests posibles, se considera esta lista. Programar nuevas funcionalidades implica realizar más tests.

Los tests son la práctica de la calidad.

Se define y se escribe el código de los test que se van a usar. En las metodologías ágiles los tests son tan importantes como el código y se hacen primero, pues el trabajo principal consiste en que el sistema los pase.

Codificar,

desarrollo propiamente dicho.

Programar y codificar es el trabajo más importante y relevante de todo proyecto. El núcleo y esencia del desarrollo de software.

El diseño se especifica en el código. La estructura se crea en el camino, es parte del código, primero se programa la funcionalidad esencial, luego se la pule, se la amplía y eventualmente refactorea.

Para determinar qué codificar:

1. Buscar el peor problema.
2. Programar el test que determina si se soluciona.
3. Codificar la solución (pasa el test).
4. Cuando deje de ser el peor problema, repetir.

Integración, instalación,

prueba piloto y luego producción.

Es importante disminuir el tiempo en que los usuarios pueden empezar a aprovecharse del proyecto (rápido retorno de la inversión, victorias tempranas).

Esta etapa sigue inmediatamente a la de desarrollo. Incluye los tests de integración y la creación de los de instalación.

Especificaciones.

Una vez que los test están escritos y el código funciona mínimamente, se comienza a preparar las especificaciones trabajando junto a los usuarios. Para ello se utiliza el sistema en producción.

Requerimientos.

Finalmente se escriben los requerimientos respondiendo a cómo está funcionando el software.

El proceso siempre recomienza. Cada elemento lo realimenta. Se parte de la idea inicial y los tests. Cuando ya trabaja la primera funcionalidad útil, se libera el código y se reciben comentarios. Llegado a este punto se puede refactorear, imponer nuevas ideas y crear más tests, en un proceso cíclico continuo.

Las nuevas ideas y los reportes de errores de los usuarios controlan el proceso.

Educación

.

Es el principal requerimiento de un proyecto de Software Libre.

Autoformación

.

Es conveniente preparar material para tener entrenamiento informal y referencia. El software libre ha sido creado por autodidactas en su mayor parte.

Auditorías tecnológicas

.

Realizar permanentemente auditorías tecnológicas y revisar el software con arquitectos y desarrolladores de alto nivel. Realizar las mejoras sugeridas al código o introducir las recomendaciones en un listado del tipo ``TO DO".

Auditoría contable

.

Definir la política de compras de licencias de software. Realizar permanentemente auditorías contables, revisar compras para comprobar que se está cumpliendo la política de compras y que no se adquieren licencias propietarias sin control. Incorporar la cuestión de las licencias a las rutinas de auditoría de los organismos pertinentes.

Gestión

.

Mantener un sistema de gestión tecnológico como (``Objetivos de Control de Información y Tecnologías Relacionadas" COBIT [COBIT:OC-98,COBIT:MR-98,COBIT:PG-00,COBIT:RE-98,COBIT:PO-00,COBIT:AI-00,COBIT:ES-00,C

COBIT ha sido desarrollado como un estándar generalmente aceptado y aplicable a las buenas prácticas de seguridad y control en TIC. Lo que sigue da un breve pantallazo sobre las ideas de COBIT, para su efectiva aplicación utilice los materiales originales mediante auditores certificados.

- Ayuda a salvar las brechas existentes entre riesgos de negocio, necesidades de control y aspectos técnicos. Proporciona prácticas ``buenas" y ``sanas" a través de un Marco Referencial de dominios y procesos y presenta actividades en una estructura manejable y lógica. Las buenas prácticas de COBIT representan el consenso de los expertos.
- Tiene una premisa simple y práctica: con el fin de proporcionar la información que la organización necesita para alcanzar sus objetivos, señala que los recursos TIC deben ser administrados por un conjunto de procesos TIC agrupados en forma natural.
- Está diseñado no sólo para ser utilizado por usuarios y auditores, sino que en forma más importante, está diseñado para ser utilizado como un Check List detallado para los

responsables de cada proceso. proporciona herramientas al responsable de los procesos que facilitan el cumplimiento de esta tarea.

- Es, por lo tanto, la herramienta innovadora para el manejo de TIC que ayuda a la gerencia a comprender y administrar los riesgos asociados con TIC. Con la emisión de este documento ubicamos nuestras actividades actuales en el marco COBIT, y comunicamos el mismo al equipo de gobierno de la universidad, su plantel TIC, y a la comunidad toda, con el objetivo de comenzar la discusión tendiente a implementar una gestión de las TIC moderna, avanzada y racional en la Universidad.

Programar

.

Es la tarea principal y definitoria en la creación de sistemas. La herramienta y capacidad fundamental de un creador de sistemas es su capacidad y nivel como programador. Todo lo otro es secundario [Saravia:CDE-04].

Belleza, arte

.

La creación de software puede ser un arte [Knuth:ACP] y para ello debe ser creativo, inventivo y expresivo de la sensibilidad, no sólo una técnica, ni una ingeniería.

La elegancia y belleza del código es el principal criterio para su evaluación.

Al programar se comprende e instrumenta la cuestión original. Se clasifican con elegancia los posibles casos. El código puede mirarse en función de la belleza con que está redactado.

La capacidad técnica del programador es fundamental en este sentido.

El destinatario del código es otro humano. El código fuente es una obra intelectual -no el binario, que es un subproducto-, debe ser escrito, pensando en que va a ser leído (y no sólo ejecutado por un computador).

Programar es la quinta esencia de la evolución. Es automatizar el pensamiento, o pensar en cómo se piensa y diseñar el pensamiento.

Creatividad

.

La creatividad es el más escaso e importante de los recursos. Las ideas innovativas y generativas, tanto propias como de los interesados (usuarios) son lo que impulsa el ``progreso".

Todo nuevo proyecto se debe basar en ideas originales, precisas y claras de qué se quiere y por qué.

El software debe crearse para resolver problemas nuevos.

Cuando se debe crear un software diferente para tareas similares, se está fallando en el enfoque del problema.

¿Qué origina o impulsa la idea de la creación de un nuevo software y por ello impregnará todo el proyecto de desarrollo? ¿La demanda de un usuario o la genialidad de un creador que percibe un nuevo universo posible?

Se debe evitar reinventar la rueda cada vez, por ello la novedad y creatividad de la idea es lo más importante para comenzar un proyecto, que sea clara, innovativa, que permita la creación artística.

No se trata de programar tareas repetitivas una y otra vez, sino en todo caso crear un lenguaje de especificación que permita contener casos particulares. La instrumentación de estos casos no será una tarea de desarrollo sino simplemente la adaptación e instalación de una instancia de la creación original.

Estos conceptos no deben impedir la creación de un nuevo software para realizar algo que otro ya hace, en tanto y cuanto se haga con nuevas ideas. La diversidad de software para una misma función es positiva. Una de las características importantes del Software Libre es que mantiene para cada categoría distintos softwares similares, el propietario ha creado en cada categoría monopolios.

Niveles de abstracción

.

Para cada nuevo problema se debe pensar y crear un lenguaje con el nivel de abstracción suficiente para manejar todas las posibilidades de uso. De tal forma, cada nueva tarea del mismo estilo, se implementa configurando el software, usando un lenguaje de nivel descriptivo acorde a la tarea.

Programar es instrumentar un lenguaje que permita a partir de la descripción de las tareas hacer funcionar el sistema. Cada tipo de tarea requerirá una descripción específica única para todos los negocios similares.

Todos los restaurantes/puntos de venta/etc del mundo podrían usar el mismo software, configurado de manera particular.

Diseño y análisis

.

Programar, analizar y diseñar son tareas inseparables.

A veces no se analiza, sino se diseña, lo cual siempre es preferible. Esto implica re-ingeniería [Ellison:SW], y nos da la oportunidad de inventar la mejor forma de llevar una actividad y no

congelar en un sistema la forma en que se la lleva. Muchas veces la informatización es una gran oportunidad para repensar la actividad.

Diseño, análisis y programación

.

El diseño se expresa en el código. El código expresa el diseño. Cualquier descripción formal del diseño de un sistema, debe ser parte de su programación. Se debe diseñar con un lenguaje que instrumente este diseño en código ejecutable. Cuanto mayor nivel de abstracción mejor. Pero nunca se debe usar un lenguaje de descripción que no produzca el código, o no pueda ser compilado o interpretado.

Usar descripciones cortas, introductorias y no formales para explicar los conceptos claves. Usar gráficos y esquemas aclaratorios. No usar sistemas formales no codificables.

Diseñar, analizar, codificar y testear es programar.

Comunicación

.

Todos escriben código de acuerdo a reglas que enfaticen la comunicación.

Diversidad

.

Descrea de todos los que impulsan un único camino verdadero. La etapa de los estándares es la muerte de una tecnología. Cuando está suficientemente madura para ser estandarizada seguro que hay algo radicalmente mejor y nuevo en algún lugar.

Usar una tecnología estandarizada para un nuevo sistema es empezar demasiado tarde, para comenzar un nuevo proyecto busque las tecnologías emergentes, que serán maduras cuando su proyecto entre en producción. El mundo avanza muy rápido en desarrollo de software.

Metáforas

.

Se puede utilizar una metáfora simple y compartida sobre cómo trabaja el sistema para guiar el desarrollo. Plantee esquemas conceptuales definidos, por ejemplo el tablero de comandos personalizado

Test

.

Ver Test más arriba en ``Ciclo de Desarrollo'', 3.4.

Estándares

.

Utilice cuando corresponda los estándares del proyecto ``GNU" (GNU no es Unix) [GNU:GCS-04] que fundó el software libre tal como lo conocemos hoy día y los de ``GNU/Linux" [LinuxBase:SW,FHS:SW] que constituye el sistema operativo más usado. En particular en cuanto los programas deben ser multiplataforma, usar autoconf, etc. También es importante apegarse a los estándares POSIX, w3c, y de GNU/linux [FreeStandars:SW], etc. en cuanto correspondan.

Documentación

.

La mejor documentación sobre un sistema es un código bien redactado. El código es el lenguaje humano que describe lo que los sistemas hacen. Los compiladores e interpretadores se encargan de traducir esto a un lenguaje que las computadoras entiendan, pero debe entenderse que el principal destinatario de un código es un humano, los compiladores que se arreglen.

El código que para explicarse requiere textos en lenguaje natural, puede ser visto como un código de baja calidad y confuso.

Tecnología

.

Para crear un sistema se debe en primer lugar comprender, probar y hacer interactuar todas las tecnologías que se piensan usar. Se deben construir tests para probar que funcionan como se piensa.

Los tests se aplicarán a los primeros módulos que se vayan construyendo para utilizar las tecnologías en su bajo nivel. La tecnología determina e impacta profundamente en los diseños por lo que cualquier esfuerzo en clarificar estas interfaces repercute enormemente en todo lo que se haga a posteriori. Con las primeras versiones de estos módulos se puede comenzar a programar los sistemas deseados, implementando tests de los mismos en paralelo con su desarrollo.

Se debe probar que se conocen sus límites de capacidad y rendimiento.

Mensajes y Errores

.

Los errores deben ser controlados por un sistema de alto nivel manejado por política [Jungman:ECC], esta debe decidir qué hacer ante cada error. Así se debe actuar diferente si el error se encuentra en la fase de producción, desarrollo o test.

Cuando algo deba fallar, que lo haga lo antes y más ruidosamente posible, salvo que esté en producción, donde debe haber mucho ruido, pero todo lo que pueda seguir funcionando debe hacerlo.

Se debe coordinar la emisión de diferentes tipos de información extraordinaria para cada suceso y según sea el tipo de trabajo del sistema.

1. Mensajes. Algunos mensajes son normales, realimentación o diálogo con el usuario, información por un canal adicional al normal del programa sobre lo que hace. El nivel de debug indica qué mensajes se tiran, registran o cuáles se registran y se muestran. Los cambios a los datos estables conviene guardarlos como históricos.

2. Errores de la instalación, configuración, suelen reflejar problemas diversos, de organización, etc. En principio se deben mostrar.

3. BUGS: los imprevistos, la idea es que los registre y los muestre al usuario como tales, en forma cruel al programador, pero también que no existan, o sea si aparece avisar al desarrollador de un BUG del programa. Cuando suceden en producción deben ser avisados, pero el programa debe tratar de seguir adelante, dando oportunidad al usuario de salvar datos, o continuar si es pertinente.

WEB

.

Cuando pueda (email) use el modelo de programación web, investigue AJAX [Garret:ANA-05].

Lenguajes

.

Use en general lenguajes tipo Perl, Python, Ruby, use C[3.7] sólo para partes muy especiales y luego de prototipar con los anteriores. Sólo si es imprescindible use c++, o mono (facilita migración de .net).

Plantee sistemas con códigos en diferentes lenguajes, según sus partes. Use el mejor lenguaje para cada diferente parte de su sistema según convenga.

Es mejor mezclar lenguajes que usar un solo lenguaje en áreas para los cuales este es inconveniente.

Use C, cuando no pueda usar Perl, use Perl cuando no pueda usar Bash, Bash cuando Awk no funcione y Awk cuando Sed y otros filtros no puedan hacerlo.

Si Ud. piensa que hay muchas formas hacer lo mismo use Perl [Wall:PP-91] que libera, si cree que hay una forma que es la mejor de todas, use Python [Python:SW] que encasilla. Caos vs. control del pensamiento. O arte creativo vs. ingeniería. La elección es una cuestión de conformación mental de los programadores.

Estudie nuevos entornos de desarrollo como Mozilla [Dumbill:MDP-05], siga con atención a Parrot [Parrot:SW] y Perl6 y estudie los módulos de Apache [Apache:SW].

20-80

.

El 20% de la funcionalidad es el 80% de lo que se necesita.

80% del beneficio viene del 20% del trabajo.

Generación

.

Evite hackear a mano, escriba programas que escriban programas en cuanto lugar pueda.

Optimización

.

Prototipe antes de pulir. Hágalo trabajar antes de optimizar. Optimize cuando ya tenga toda la funcionalidad operativa.

Flexibilidad, y refactoreo

.

Se debe tener el más simple diseño que ejecute la funcionalidad operativa. A medida que hay más funciones se cambia el diseño.

Los programadores reestructuran el sistema sin cambiar el comportamiento: para remover duplicación, mejorar comunicaciones, simplificar o flexibilizar.

No temer al refactoreo. Estos trabajos aumentan la flexibilidad del código.

Obliga a codificar en forma comprensible y elegante. Evita concentrarse en los detalles. Los refactoreos de los proyectos son su principal activador y cuestionador externo.

Redactar rápidamente una función hasta que opere en su nivel mínimo y luego replantearla para un esquema más complejo o más abstracto.

Diseñe para el futuro, en forma extensible, porque está más cerca de lo creíble, pero no se pierda en modelos muy abstractos cuando no los necesita. Pase los test primero, luego abstraiga y generalice el código.

El trabajo de diseño se expresa constantemente en el refactoreo del código. Si es importante, se debe refinar la arquitectura permanentemente.

Expansión - compresión

.

Mientras más funciones se construyen más complejo se hace el código. Pero a medida que se comprende mejor un sistema para una funcionalidad dada, se puede simplificar mediante abstracciones. Es un proceso constante de expansión - compresión.

Cambios y versionado frecuente

.

El cambio permanente y rápido es un muy buen amigo de la programación.

Los cambios se hacen como si se estuviese conduciendo un automóvil. Se debe ir llevando el sistema al objetivo poco a poco. Manteniendo su funcionamiento, mediante micro cambios, que aplicados varios en forma consecutiva producen los grandes cambios. El sistema se mantiene siempre en producción con realimentación permanente.

El mejor sistema es el que se está usando. El software rápidamente debe entrar en producción y mantenimiento. Un sistema que no está en producción es una carga económica y una ineficiencia.

Los sistemas siempre evolucionan, si se quedan mueren. Un software vivo es aquel que tiene programadores mejorándolo inmersos en su propia cultura. Jugar y explorar.

Ponga un sistema simple en producción rápidamente, luego largue nuevas versiones cada poco.

Realimentación rápida, cambios incrementales con pequeñas diferencias, que puedan evaluarse con facilidad.

Publicación temprana y test inmediato con los usuarios, ciclo corto de desarrollo.

El programa y su diseño evolucionan, los cambios no se restringen a un área, los programadores añaden valor al análisis, diseño, instrumentación, y test. Añaden valor a todos los lugares donde es necesario.

Minimizar el costo de cambiar, para cambiar mucho.

Se deben sacar nuevas versiones constantemente. Tiempos de minutos y horas en entre versiones de producción. No semanas o meses. Tener los mecanismos aceitados para poder publicar e instalar nuevas versiones es básico. Si un usuario requiere un cambio y se lo puede poner en minutos mejor.

Gratificación temprana

.

Código pensado para tener excelencia operacional, gratificación instantánea y no quedar atrapado en tecnologías sin salida.

En este modelo de desarrollo el concepto de producir pequeños cambios y que estos sean operativos y ``gratificantes'' en forma inmediata es fundamental. Libere código pronto y en forma frecuente.

Claridad y simplicidad

.

Se debe crear software concreto, intuitivo, fácil de bajar, instrumentar y administrar, asumiendo la simplicidad como valor.

Mantenga el código claro y simple.

Transparencia: diseñar para la visibilidad, para hacer la inspección más fácil.

Robustez: es la hija de la transparencia y la simplicidad.

Modularidad: partes simples conectadas con interfaces claras.

Composición: programas diseñados para conectarse a otros.

Tan simple como sea posible en cada momento. La complejidad extra es removida apenas se la encuentra.

Reuso

.

Deben estudiarse y conocer las herramientas e ideas preexistentes. Antes de comenzar se investiga qué existe.

Consultar software y proyectos preexistentes, relativos a la tarea a asumir, encontrar un vocabulario y un kit de herramientas y abrir las especificaciones al equipo de desarrollo.

Capas para abstraer problemas

.

Si lo que se quiere hacer existe, se lo usa o se lo mejora. Si el proyecto es novedoso no se encontrará nada apropiado, pero si es un proyecto cuya idea está madura, seguramente se encontrarán las capas de abstracción inmediatas sobre las cuales construirlo.

Integración continua

.

Integrar el sistema muchas veces por día, cada vez que se completa una tarea. Testear apenas integrado.

Pequeño

.

Pequeñas herramientas para cada tarea, que funcionan desde línea de comando. Separar GUI (Interfaz gráfica de usuario) de herramientas. La herramienta ideal es el filtro.

Lo pequeño es bonito. Escriba cosas chicas y consistentes que cumplan su función.

Filtros 1

.

Todo lo que pueda ser un filtro debe serlo.

Un filtro es un programa cuya única vía de comunicación con el mundo consiste en ``cañerías" de entrada y salida, y se limita a producir salida en función de lo que va recibiendo en su entrada.

Filtros 2

.

Cuando filtre, nunca tire información que no necesite tirar.

Texto

.

Los ``streams" de datos deben ser textuales

Donde sea posible, las estructuras de datos deben ser textuales, legibles y editables por humanos

Interfaces

.

Las interfaces al usuario deben ser separadas de los ``back ends" complejos

Menor sorpresa

.

Para diseñar una interfaz al usuario, esta hará lo menos sorprendente.

Datos y código

.

Utilice interfaces claras entre el código y los datos. Un lenguaje de alto nivel para administrar los datos o bases (SQL). Cuidado con la programación orientada a objetos. Úsela sólo en los casos en que realmente sea apropiado.

Repositorios centrales para aplicaciones dispersas. No salirse del estándar, todo debe funcionar con todos los motores sin más esfuerzos.

Representación, use datos inteligentes, para que la lógica del código sea estúpida y robusta.

Un viejo aforismo dice ``dame la estructura de datos y te daré el código". Las estructuras de datos inteligentes asociadas a un código torpe funcionan mucho mejor que la alternativa opuesta. ``Enséñame tu código y mantén ocultas tus estructuras de datos, y me seguirás engañando. Muéstrame tus estructuras de datos y normalmente no necesitaré que me enseñes tu código: resultará evidente." [Brooks] [3.8]

Separación

.

Separar la política de los mecanismos. Pensar delicadamente la interfaz de los motores internos.

Procesos

.

Un sistema orientado a procesos mostrará las opciones al usuario según los procesos en marcha y las responsabilidades o atribuciones del rol representado por ese usuario.

En estos esquemas la gestión se organiza en procesos. Los procesos representan conjuntos secuenciados de tareas. Las tareas se ejecutan mediante páginas o pantallas. Para cada proceso se registran las tareas cumplidas exitosamente. Los elementos (archivos, directorios, tablas, etc) son los datos. Los roles se implementan por alias o grupos. Los ACL (listas de control de acceso) vinculan roles con permisos y elementos (datos).

ACL, listas de control de acceso

.

Utilice sistemas de control de acceso en sus políticas de seguridad y visibilidad.

Programación de a pares

.

Cada sección de código se escribe mediante dos programadores trabajando juntos en cada máquina, todo el tiempo. Ver Programación Extrema: [XP:SW,Robles:IS-02,Robles:PES-02].

Los pares revisan permanentemente el código de otros.

Los usuarios revisan el programa en forma permanente y constante.

Rotación

.

La rotación de las personas sobre diferentes partes del código es positiva.

No más de 40

.

No más de 40 horas por semana. Nunca sobretiempo u horas extra.

Metodología para administración de proyectos de software

Divisiones del trabajo

Contribuyentes:

Redactores de código, documentación y tests.

Son las personas que a título individual o trabajando para una tercera organización: empresa o cooperativa crean material para algún proyecto

Cada persona y organización debe registrarse y enviar un compromiso de transferencia de copyright de código

La organización en ese acto se compromete a mantener el código bajo la GPL.

La persona debe informar claramente que no está en dependencia laboral de una tercera o si lo está los responsables legales de tal organización deben autorizar o firmar el acuerdo.

La organización podrá tener a su personal o a personal contratado a ese efecto participando en el desarrollo, este personal tendrá una nómina mensual, por lo que no recibirán un aporte específico por sus aportes, pero estos serán evaluados con la misma rigurosidad por el integrador.

Integrador:

Cada proyecto tiene un integrador, personal de la organización o bajo contrato específico.

Este integrador decide qué porción de código, documentación o código de tests ofrecida por los redactores puede integrarse.

En el software libre se acostumbra a publicar rápido, muchas veces y porciones chicas de código, no funciona de otra forma pues si mucha gente trabaja y se reciben grandes aportes todos juntos, el trabajo que otros hacen es muy difícil de integrar.

Este integrador puede publicar necesidades orientativas para los que busquen aportar código

El integrador podrá tener un equipo de personas evaluando las contribuciones y puliéndolas en lo que sea necesario. Este equipo estará empleado o bajo contrato en la organización.

Responsable institucional:

Es la persona que promueve las vinculaciones entre el proyecto y la comunidad

Sea en proyectos propios o de terceros, en proyectos pequeños puede ser el mismo integrador.

Unidad de contrataciones:

Una vez que a una persona se le acepta el código en el proyecto, se mide con determinada métrica prefijada su aporte y se le paga.

La unidad de contrataciones puede asignar algunas necesidades a redactores específicos e incluso prenegociar el precio de tal trabajo

Usuarios:

La participación de los usuarios es fundamental, en cuanto aportan nuevos desafíos y pedidos de correcciones.

El ciclo de desarrollo de estos modelos colaborativos implican gran velocidad de cambios, estos cambios incluyen a los usuarios, que pueden pedir un cambio y lo tienen operativo en días.

Tipos de proyectos

1. **Tipo 1: Desarrollo de nuevos proyectos**

   Manejo de la cartera de proyectos propios.

   Se aplica cuando no se encuentran equipos que estén desarrollando un software que pueda utilizarse para la tarea en cuestión o cuando se decida que conviene emprender un nuevo proyecto.

   Existirá un único repositorio general de la Organización manejado o controlado desde uno de sus laboratorios.

   Para cada proyecto su integrador decidirá quiénes tienen derecho de ``commit''. Todos tendrán derecho de bajar.

   Cada proyecto será asignado a un integrador bajo las siguientes condiciones:

   1. Mantendrá tres listas de correos abiertas: a) desarrolladores, b) usuarios c) noticias importantes destinadas a todos los otros, la prensa y especialmente a la organización.
   2. Toda decisión de aceptar o no un código por parte del integrador, será evaluada en la lista, el integrador no podrá priorizar desarrollo de sus propios desarrolladores sobre otros.
   3. El código será mantenido en el repositorio creado por la Organización.
   4. El código estará universalmente disponible para terceros mediante la GPL
   5. El integrador mantendrá un sitio web con documentación, los anuncios en blog y links al código.
   6. El integrador mantendrá un sitio wiki con áreas de acceso libre.
   7. El integrador asignará prioridades de desarrollo según un sistema de bugs y reportes.
   8. El integrador tendrá test automáticos exhaustivos para el código.

2. **Tipo 2. Participación en proyectos preexistentes**

   Se aplica cuando se encuentra un Software Libre útil preexistente y un equipo de desarrollo operativo. En este caso la organización se suma a un esfuerzo ya existente.

   En tal sentido hay que analizar cómo funciona el proyecto, cuáles son las mejores formas de cooperar y qué cosas se necesitan agregar a las funcionalidades, documentación, tests, etc de ese software.

   Hay tres acciones que probablemente sean útiles en este sentido:

   1. designar un responsable institucional para interactuar con el proyecto o sus líderes, incluso integrar formalmente los consorcios de desarrollo cuando existan y aportar recursos.
   2. publicar las necesidades de cambio
   3. aplicar el modelo 1, pero la integración la hace el proyecto preexistente. Es decir ofrecer a quien incorpore cambios que la organización necesita un pago similar al que se ofrece en el caso anterior. En este caso el integrador es externo. Cuando este integrador acepte una contribución, la organización paga.

      Si en algún momento el integrador rechaza código que la organización necesita, se puede poner un integrador interno que mantenga un proyecto paralelo que puede compartir con terceros, (un fork)

3. Podrá existir un tercer modelo cuando la organización se asocie con terceras para un nuevo proyecto. En este caso se aplica una combinación de los dos modelos, pues el integrador es designado de común acuerdo entre las firmantes del consorcio que se cree,

Los desarrolladores que pretendan remuneración por sus aportes a estos proyectos se anotarán en un registro especial común a todos los proyectos. Existirá una lista de correo de anuncios para todos ellos.

Una vez que un desarrollador tiene su código aceptado en un proyecto podrá reclamar el pago, sea que haya prepactado su trabajo o no.

Existirá un tablón de anuncios con todos los requerimientos de la Organización en todos los proyectos, propios o no. (podrá incluir valor o no).

Los desarrolladores podrán acordar y tomar alguno de ellos o resolverlo y luego solicitar pago.

Esquemas

proyecto propio	proyecto preexistente
ÚNICA	ÚNICA

Unidad de contrataciones		
Integrador	PROPIO, uno por proyecto	no, lo aporta el proyecto
Responsable Institucional	ÚNICO Optativo, promueve la participación de terceros	ÚNICO, PROPIO, interactúa con el proyecto
Redactores	PROPIOS y de terceros	PROPIOS y preexistentes en el proyecto

Entonces tenemos en la organización una unidad de contrataciones ÚNICA (con su equipo), un responsable institucional, con su equipo de personas distribuido por proyectos, y un integrador por proyecto. En cada proyecto habrá un responsable de proyecto en la organización, que puede jugar el rol de integrador o de responsable institucional o ambos si se lo requiere o tener en proyectos grandes personas con esos roles.

	COORD	LABS	RES. INST.	U.CONTR.	redactores
central	*	*	*	*	
por negocios/ regiones	*	*	*	*	
por área temática	*	analistas/ diseño			
Proyectos	*	Integrador	*		internos externos libres ext. contr. prev

Este esquema administra el trabajo de los contribuidores que pueden ser internos de la organización, sean contratados ad-hoc o parte de AIT preexistentes, o externos a la organización bajo contratos de desarrollo de necesidades o realizando aportes en forma libre.

En los laboratorios por negocio existirán analistas y diseñadores que dividirán el trabajo a realizar en proyectos, detectarán las necesidades y mantendrán la coherencia de estilo y el uso de modelos de datos y desarrollo coherentes, tendrán relación directa con los responsables de los proyectos y sus integradores cuando sean diferentes, mantendrán en línea entre sí a los diferentes proyectos en las áreas de integración.

Tipos de caminos intermedios en proyectos de migración

Tabla 3.1: Caminos posibles para acercarse al objetivo del 3390

	Terminal	Aplicación	Intérprete/ O. Soft.	Sistema Operativo

	M.M.	G/L	Win	Pr	Lb	N.U.	Lb	Pr	Win/O	G/L.E	G/L
Hoy			*	*		*		*	*		
3390	*	*			*	*	*				*
Portar (P/C)									---------->		
Emular									------>		
Cambiar Int.							<----				
Lib./Cambiar				---->							
TS	-----> <-----										

M.M.:
Misma Máquina

G/L:
GNU/Linux

Win:
Windows

O:
Otro

Pr:
Propietario

Lb:
Libre

Lib:
Liberar

Int:
Intérprete

N.U.:
No Usa

G/L.E:
GNU/Linux emulando Windows, Wine, QEMU, VMware, etc.

TS:
Terminal Server, rdesktop

P/C:
Portar propia o adquirir portada, puede o no ser libre.

Cada camino elegido para un proyecto puede cumplir determinado porcentaje de la meta. Se puede decir que emular en terminal remota es un avance del 5%, un 60% a la liberación de la aplicación, un 20% a la liberación del software auxiliar, y un 25% al funcionamiento en Sistema Operativo libre.

Los intérpretes, servidores como Apache, bases de datos, compiladores necesarios para la ejecución y creación del software se consideran como un bloque.

Medios para crear comunidades - Listas

En particular una lista técnica de correo electrónico para la migración. Una de las acciones para migrar las herramientas de trabajo grupal del equipo de migración. Preparar sitio web y repartir información y claves.

1. Es una lista para que el personal de informática de la organización pueda consultar dudas sobre Software Libre.
2. Se invita a la comunidad:
    1. La participación es nominal y real, se debe usar nombres verdaderos.
    2. La información que pueda circular sobre aspectos internos de la organización es confidencial, si bien se intentará limitar esa circulación.
    3. La información técnica sobre SL es libre.
    4. La participación en la lista no implica remuneración ni oferta de empleo, o de contrato de algún tipo, pero eventualmente podrá ser tomada en cuenta en futuros requerimientos como antecedente por diferentes proyectos de la organización que participen en la lista, convocando a personas de su interés.
3. Es una lista con perfil exclusivamente técnico.
4. Sólo los miembros podrán acceder a sus archivos.
5. Se usará lenguaje castellano.
6. Se podrá invitar a técnicos de otros organismos del estado, etc.

Cooperativas y economía solidaria

Subsecciones

Introducción

Las herramientas, prácticas y metodologías del Software Libre constituyen un aporte crucial en la historia del desarrollo del software.

La transición al modelo libre ha ganado un momento irreversible. El paradigma ha cambiado. La duplicación de esfuerzos del modelo privativo es un disvalor en el entorno económico actual. Las organizaciones que no se adapten quedarán estancadas en el pasado.

Un plan masivo de desarrollo de software como el requerido en Venezuela necesita que sus decisores comprendan los modelos de desarrollo que han construido el Software Libre y puedan aplicarlos con éxito al proceso revolucionario que lleva a Venezuela hacia el socialismo.

Planeando el desarrollo masivo de Software Libre en Venezuela

---

Introducción

El efecto del Decreto 3390 [Venezuela:DSL-04] sobre uso de Software Libre en el Estado venezolano no sólo se hará sentir en su destinatario específico, sino en todo el tejido tecnológico venezolano y también en el desarrollo de software de todo el planeta. Sea que se logre el objetivo total o parcialmente, o que se fracase, el impacto positivo o negativo será enorme.

El software siempre avanzó a grandes saltos, este es el momento de salto al Software Libre y Venezuela es la batalla definitiva en este salto.

Una batalla donde la cuestión fundamental es política y no técnica. Con una dirección del proceso desde arriba para abajo[4.1]. Una batalla dinámica con muchos frentes y momentos diferentes. Una lucha donde enormes intereses se oponen mientras aprovechan los últimos espacios de saqueo que Venezuela les ofrece.

En este marco donde los enemigos nunca se rendirán, es esencial la unidad de los revolucionarios en pos del objetivo de dejar de alimentar a las multinacionales del software que viven de apropiarse del conocimiento.

Debe nacer una comunidad de desarrolladores basada en el empleo estatal, en la economía solidaria y cooperativa. La misma debe aprender a construir un modelo de desarrollo de software solidario, libre, colaborativo y participativo. Los responsables de cada institución del Estado deben aprender a gestionar un nuevo sistema, con sus desafíos y oportunidades específicas. Saber redirigir el enorme caudal de recursos que se van en licencias propietarias hacia el desarrollo endógeno de software no es sencillo.

He aquí el desafío del presente trabajo: ayudar a plantear y comprender estas cuestiones.

Plan de desarrollo de aplicaciones

Se trata de pensar un plan masivo de desarrollo de aplicaciones libres:

- Impulsadas desde la demanda (decreto 3390)
- Sostenidas con recursos económicos usando trabajo asalariado y relaciones de economía cooperativa y/o solidaria.

  No se impulsará, aunque tampoco se impedirá la participación de empresas enmarcadas en un mercado capitalista clásico.

- Que tenga en cuenta las metodologías aptas para el Software Libre y que permitan su alto grado de productividad.

Se puede pensar para el desarrollo de aplicaciones específicas en consorcios de interesados con características similares en todo el planeta, y mientras estos se construyen, uno por aplicación o área de interés, puede empezar un demandante. (Es habitual que los consorcios se formen llegado un cierto grado de madurez). [4.2]

Así parece razonable que el impulsor del proyecto construya por sí o mediante algún tercero un sitio específico que apoye el desarrollo, esto es servidores ``subversion", ``wikis", listas de correo, ``jabber", etc. Para proyectos chicos se podrá usar sitios como ``SourceForge". Este sitio podrá ser uno para todo el conjunto de aplicaciones o uno por aplicación. Así un demandante importante debería empezar el trabajo por montar estos sitios y ordenar a los responsables de cada aplicación a que la coloquen en los mismos.

Lo más importante no es el sitio y sus tecnologías, sino la dinámica humana de interacción de los grupos que se formen:

- participantes de muchos lugares que realicen aportes.
- personal contratado a tal efecto, por ejemplo una cooperativa.
- sistemas para financiar contribuciones pre y post efectuadas.
- mecanismos para asociar al proyecto a quienes realicen aportes significativos.
- el líder del proyecto decidirá sobre los aportes de los participantes y casi con seguridad deberá ser rentado para ello o ser empleado de la organización madre del proyecto.
- personas para relevar la calidad, un apropiado sistema de versionamiento y de versiones de producción, prueba, desarrollo, etc.

La ingeniería de este proceso social es la clave para un sistema exitoso de desarrollo masivo de Software Libre. La formación de personas creativas es la esencia para que este proceso refuerce el desarrollo local y endógeno.

El plan Vuelvan Caras, la vinculación de las nuevas universidades venezolanas y su distribución en todos los municipios es fundamental.

No es de esperar que se logre en tan poco tiempo como el requerido por el 3390 la creación de tanto software si se aplican las metodologías tradicionales. Así pensar en entregar cada proyecto a una empresa comercial cerrada o incluso a un solo grupo no es viable ni es la práctica habitual en el mundo del Software Libre, donde el desarrollo debe ser transparente, los ``releases'' abundantes y frecuentes y el trabajo visible por miles.

Es imprescindible:

- empezar a diseñar una metodología de trabajo que permita la apertura del código y los ciclos rápidos de desarrollo propios de las metodologías ágiles (a la XP).
- diseñar una ingeniería económica que permita la justa retribución de los que trabajen en estos proyectos sea en forma cooperativa o en forma individual.
- formar a los líderes de los centenares de desarrollos necesarios y construir alrededor de estos poderosas empresas comunitarias y sociales basadas en el comercio justo y la economía solidaria.

La clave del cumplimiento del 3390 está en diseñar esta nueva ingeniería de contrataciones a escala masiva.

Empresas de producción social

A los efectos de poder construir toda la enorme cantidad de Software Libre necesaria, es pertinente pensar en formar técnicos y desarrolladores a escala masiva. En tal sentido la estructura de la misión Vuelvan Caras es la apropiada.

El gráfico muestra un esquema de incubadoras de cooperativas apto para tal tarea.

Figura 4.1: Incubadora de cooperativas. Vuelvan Caras.

29 Requisitos del Software para una Sociedad del Conocimiento Democrática.

> Con software propietario los Estados deciden si alfabetizan y abren la participación ciudadana a un costo de U$S 500 por persona, en pagos de licencias de software a una multinacional, o los convierten en criminales del delito de compartir.

Subsecciones

Introducción. Razones para el uso del Software Libre y el abandono del Privativo

Un Estado de derecho democrático tiene responsabilidades inherentes a su función.

En particular la obligación de garantizar a sus ciudadan@s una serie de derechos indispensables. El Estado almacena, manipula y transforma información de particulares, que le ha sido confiada por los ciudadanos que, por imperio de la ley, no tienen más alternativa que hacerlo. El Estado tiene la obligación de resguardar esa información y mantenerla en formatos que no lo aten a iniciativas monopólicas privadas. El Estado tiene la obligación de garantizar a sus ciudadan@s el acceso a la información que es pública y la preservación de la información que es privada o que debe ser mantenida en secreto por razones estratégicas de seguridad nacional. El software propietario es incompatible con estas obligaciones del Estado. Los gobiernos deben tomar decisiones pensando en mejorar su sociedad. No basta con que el software que se utilice en el Estado ejecute una tarea de acuerdo a parámetros técnicos, o sea el menos caro, sino que además debe ser políticamente correcto y estratégicamente adecuado, satisfaciendo una serie adicional de requisitos, sin los cuales el Estado no puede garantizar al ciudadano el procesamiento adecuado de su información, ni mucho menos puede construirse un gobierno electrónico o una democracia digital.

Cada decisión referida al software debe evaluarse contrastando sus efectos en diversas áreas. En algunos casos el impacto es directo ya que la decisión del Estado determina resultados en dicha área. En otros

casos la elección del Estado promueve o induce políticas en la denominada sociedad civil, ciudadanía y empresas.

De esta evaluación deberán surgir criterios sobre el uso de software en el Estado. Estos criterios se expresarán en un conjunto de medidas a diferentes niveles normativos y ejecutivos.

Así quedarán reflejados en leyes, decretos y normas técnicas. Por ejemplo normas sobre adquisiciones.

También deberán incorporarse en las normas de cada institución, y cuando se trate de compras, en las condiciones generales y particulares de cada pliego. En estos casos, los responsables de cada adquisición evaluarán la mejor decisión entre las presentadas. Este esquema no excluye a determinados proveedores sino que impone criterios generales y particulares a cumplir por el software. Son los proveedores los que deciden si licencian sus productos al Estado en las condiciones que éste pide o no.

Resumen

Requisitos imprescindibles para el Software del Estado con relación a diferentes temáticas:

Transparencia
: [R01]5.4.1 [R02] [R03] [R04][R05] [R06][R07] [R12][R25]

El Software del Estado debe permitirle a sí mismo, a sus ciudadanos y a la oposición política auditar cuáles son sus reales efectos y funciones. En particular evitar puertas ocultas impuestas sin conocimiento del gobierno, por proveedores, terceros, o agentes extranjeros.

El Estado debe participar en el desarrollo de su software y tomar las decisiones claves. El software debe ser uniformemente usado en todas sus divisiones. Todos los aplicativos deben interoperar entre sí.

El Estado debe usar estándares de almacenamiento y trasmisión, de información y documentos, no sujetos a monopolios particulares. Así garantiza perennidad en la información pública y privada a su cargo, e independencia ante los proveedores.

El Software del Estado debe favorecer la democratización en el acceso a la información y a sus sistemas, facilitando una comunicación multidireccional entre el mismo y su Comunidad. Debe permitir la construcción de sociedades más diversas y justas.

Educación
: [R25][R29]

El Estado debe promover la educación de sus ciudadanos.

¿Cómo explicarle a un niño que su disco compacto con juegos es el único juguete que no debe prestar[030]? ¿Por qué construir un mundo donde el adolescente que encuentra la forma de

cambiar un código secreto y se lo explica al mundo, va preso por ello, en vez de ser considerado un genio?

Es imposible estudiar informática sin acceso al código fuente. Para formar a las nuevas generaciones de técnicos, los niños deben poder ``desarmar'' su software y reconstruirlo como ellos quieran.

El Estado requiere libertad de uso, modificación, y distribución del software, para poder concretar programas de alfabetización informática e inclusión tecnológica para la población utilizando mínimos recursos.

Sociedad

: [R13][R20][R21] [R22][R23][R24] [R25][R26][R27] [R28][R29]

Libertades, Derechos y Futuro: El Estado debe garantizar el derecho a comunicar, el derecho a conocer y manejar la tecnología que se usa, a conocer e interpretar la información que circula por nuestros ámbitos y el derecho a la privacidad de los datos individuales. Debe promover la independencia tecnológica y la innovación. Debe ejecutar políticas que ayuden a la construcción de sociedades más justas y transparentes.

El tipo de código y las redes de comunicación que se usen, determinarán si se impulsa una sociedad abierta y democrática; o cerrada, tecnocrática y estructurada; un ámbito de negocios en libre competencia o un sistema de monopolio absoluto.

Internet representa un ámbito global de interacción de la sociedad humana y las culturas. El software es un lenguaje de articulación de procesos. El Software Libre es un nuevo movimiento social. Internet más Software Libre equivalen a concretar el Otro Mundo Posible, que se empieza a construir desde las trincheras tecnológicas[180].

Calidad y Seguridad

: [R10][R11] [R12]

Se requiere un software sin fallas, estable, seguro, eficiente, que aproveche los pocos recursos disponibles, que permita trabajar a muchos usuarios en diversas terminales, cuyas actualizaciones, correcciones o mejoras no requieran reinstalar todo o cambiar de hardware y que asegure una amplia variedad de software disponible. La seguridad por la oscuridad ha fallado consistentemente en la construcción de una red libre de virus y crackers.

Economía:

[R14][R15] [R16][R17][R18] [R19]

Se requiere un software que apoye el desarrollo económico en forma solidaria y sustentable, que favorezca la libre competencia, que promueva el desarrollo tecnológico, que permita el surgimiento de empresas a distintas escalas para cada tipo de problema a resolver, que cree

puestos de trabajo especializados, que favorezca la aparición de una industria local de desarrollo de software, de generación de servicios, de soporte, etc.

Los modelos de negocio basados en la recaudación, con soporte policial público, de un impuesto por PC o usuario y envío al extranjero no son convenientes.

Costos:

[R08][R09] [R14]

El Estado siempre debe adquirir la opción más conveniente. Sin duda el precio es un factor importante. El costo de las licencias es fundamental para evaluar software, sobre todo si afecta la balanza de pagos.

Usar software propietario, implica también contar el costo administrativo del sistema de licencias, el costo del registro y control de las mismas en cada PC (Computadora Personal) y por cada empleado. Todo ello para evitar que el Estado termine pagando multas, o soportando juicios.

El Software Libre es la única vía para cumplir estos requisitos.

Clasificación de requisitos

- Gestión Propia. Para mejorar la sociedad, el gobierno debe mejorarse a sí mismo. o

  Transparencia. +

  [R01] Auditoría de la función del Software. +

  [R02] Accesibilidad, confidencialidad e Integridad de los datos. +

  [R03] Estándar de formatos. +

  [R04] Perennidad. +

  [R05] Homogeneidad. o

  [R06] Participación. o

  [R07] Interoperabilidad. o

  [TCO]. Costo total de posesión. +

  [R08] Costo de las licencias. +

  [R09] Costos Legales. o

  Calidad. +

  [R10] Técnica. +

[R11] del servicio y soporte. o

[R12] Seguridad. o

[R13] Capacidad de promover el cambio. *

[R14] Economía. o

[R15] Balanza de Pagos. o

[R16] Industria Local. Trabajo, crecimiento y distribución del ingreso. o

[R17] Libre competencia. o

[R18] Sustentabilidad. Software tecnológicamente apropiado. o

[R19] Igualdad. *

Tecnología. o

[R20] Innovación. o

[R21] Independencia y soberanía. *

Derechos Humanos. o

[R22] Seguridad personal, Espacio personal de privacidad. o

[R23] Libertades y Derechos. Derecho de expresión y comunicación. *

[R24] Sociedad. Estructura social. Cultura. Otro Mundo es Posible. Solidaridad o

[R25] Brecha digital. Educación. o

[R26] Género y diversidad. o

[R27] Lenguas y pueblos del mundo. o

[R28] Control. o

[R29] Educación Tecnológica.

Desarrollo

En las próximas secciones se analizan los diferentes criterios de análisis junto a las ventajas y riesgos de cada tipo de Software. El Estado, por su envergadura y por su papel de administrador de los bienes comunes, es particularmente vulnerable a los riesgos del software propietario, a la vez que está en una posición particularmente estratégica para beneficiarse con las ventajas del Software Libre y también para contribuir a su desarrollo. Transparencia

El Estado es el guardián del registro público. Mantiene información actualizada sobre la identidad y el patrimonio de las personas y de su propio accionar. Un Estado de derecho democrático requiere instrumentos tecnológicos adecuados a su misión. Su plataforma tecnológica tiene requerimientos muy estrictos.

El respeto de los derechos del ciudadano es un principio superior a las consideraciones comerciales. El Estado no tiene la misma libertad de acción en materia económica y tecnológica que otros usuarios. El Estado debe respetar normas de contratación más restrictivas.

[R01] Transparencia Auditoría de la función del Software

Democracia o Dictadura Tecnocrática. Estado de Derecho en la Sociedad del Conocimiento. Código legal vs. Código en software. Escrutinio público.

El software trata información y es en sí mismo información. Susceptible de ser interpretada por una máquina para ejecutar acciones, elegir alternativas y llevar adelante procesos.

El Estado usa el software para instrumentar el mandato de la ley. El código en software sustituye en la práctica al código legal como mecanismo de decisión y control. Máquinas automáticas controladas por programa asumen funciones administrativas, de policía y control. El micropoder pasa a las computadoras, que deciden si se accede a un crédito, al voto, a un trabajo, etc. Cuando entre el texto de la ley y la función del programa hay inconsistencias, el ciudadano se encuentra con que el software es más poderoso que la ley, ya que es aquél el que, en realidad, gestiona su trámite.

El Estado requiere mecanismos transparentes en sus acciones. El software usado como soporte operativo de los procesos y procedimientos del Estado, es parte de los mismos, y está sometido al requisito de publicidad de los actos de gobierno.

Que el gobierno, la oposición política, y los administrados, accedan a las fuentes del software es un requisito esencial para confiar en las acciones de un estado tecnológico. Caso contrario será una tecnocracia política, sin control social, operada por las corporaciones, con sus constituciones y representantes electos inefectivos en el mundo real.

Para cumplir principios republicanos básicos el Estado debe usar software cuyo código fuente esté publicado. Al tener la libertad de inspeccionar el mecanismo de funcionamiento del software, la manera en que almacena los datos, y la posibilidad de modificar estos aspectos, queda en manos del Estado la llave del acceso y el control de la información. El uso exclusivo de Software Libre es la única manera de auditar qué regulaciones se establecen.

El software propietario es incompatible con la democracia.

[R02] Transparencia Accesibilidad, Confidencialidad, Integridad de los datos.

Libre acceso del ciudadano a la información pública y respeto por la privacidad de la información particular almacenada en el Estado.

El Estado almacena, manipula y transforma información tanto pública como privada propia y de particulares. Esta última le ha sido confiada por los ciudadanos que, por imperio de la ley, no tienen más alternativa que hacerlo.

Se debe garantizar el libre acceso de los ciudadanos a la información pública, y el respeto a la información particular almacenada en el Estado. El Estado debe extremar las medidas para salvaguardar la integridad, confidencialidad y accesibilidad de esas informaciones.

[R03] Transparencia Formatos Estándares

El uso de estándares es un requisito para una sociedad tecnificada. Las unidades de medida estándares fueron un paso importante en la historia de la humanidad. La posibilidad de interoperación de equipos como teléfonos, radios o televisores, entre otros, se basa en normas comunes de operación. La única manera de garantizar que cualquiera pueda construir aparatos, equipos o software que interactúen unos con otros es el uso de estándares o normas abiertos. Así se posibilita el intercambio y el almacenamiento de la información y los datos entre personas o equipos en diferentes lugares y tiempos.

Es conveniente que los estándares sean fijados por cuerpos emisores públicos con control estatal y democrático. Si los estándares son internacionales, al menos los Estados deben poder decidir si los aceptan o no y tener delegados en los cuerpos internacionales que los emiten. Con más razón cuando es el propio Estado el usuario de los estándares.

Los estándares de datos no pueden estar sujetos a patentes o derechos de autor de ningún tipo. Entre otros se aplica el principio republicano de publicidad de los actos públicos. El gobierno, los ciudadanos y empresas deben poder acceder a la información pública del Estado, entre ella los estándares de interés público, en forma libre.

Resulta indispensable que la codificación de los datos no esté ligada a un único proveedor o poder externo y privado de fijación de estándares. Es fundamental que se pueda acceder a esta información sin necesidad de realizar acuerdos o contratos con proveedores privados de software. A tal efecto se debe tener al menos un Software Libre de referencia para acceder a cada tipo de datos.

Si no existen estándares públicos sobre intercambio de datos, la elección de determinado software propietario por parte del Estado limita la libertad de elección del ciudadano en materia de su propia elección, lo que constituye una violación de la igualdad de los oferentes de software ante la ley.

El software propietario crea estándares que intencionalmente hacen cada vez más complejo el trabajo de la competencia para crear productos que interoperen con otros y usa formatos secretos. El usuario queda atrapado en un determinado proveedor, lo cual es un gran riesgo, ya que es el único que puede ofrecer acceso a ellos. Cuando las decisiones sobre estos estándares son tomadas exclusivamente teniendo en cuenta los intereses privados pueden dejar a los usuarios cautivos, forzándolos a actualizaciones onerosas de software para poder acceder a su propia información.

Dos fragmentos ilustran claramente esta situación.

[103]: Uno de los ejemplos más patéticos de fijación de malas normas puede verse en Argentina. La AFIP (ente recaudador de impuestos) exige a los contribuyentes la presentación de diversas declaraciones en formato digital. La AFIP exige que la presentación sea hecha exclusivamente a través de la ejecución de programas específicos provistos por esa organización. Estos programas son gratuitos, pero entre sus requerimientos de ejecución se incluyen, como sistemas operativos, exclusivamente ``Windows 95, 98 o superior". Es decir que el Estado está exigiendo a los ciudadanos que compren un determinado producto de un determinado proveedor al solo fin de poder cumplir sus obligaciones impositivas. Esto es equivalente a dictar que los formularios no digitales sólo pueden ser completados usando lapiceras fuente marca ``Mont Blanc".

[027]:El país de los tecnocretinos...

Dejemos de lado por un instante a los expertos, para ir a ver lo que pasa en el mundo paralelo imaginario de los TecnoCretinos, en el que una empresa llamada MacroPrensa obtiene poco a poco el control absoluto de todas las imprentas del planeta. Esta no controla directamente los periódicos, pero es la que los imprime con los caracteres MacroPrensa, de los cuales es la única propietaria. Un buen día, tras una gran campaña publicitaria alabando las bondades de un nuevo juego de caracteres que permitirá obtener periódicos más modernos, esta empresa comienza a imprimir todo con caracteres klingonianos (el alfabeto de los Klingons en la famosa serie StarTrek). De esta manera, nadie más puede leer los nuevos libros o periódicos sin recurrir a la Lupa de la MacroPrensa, disponible a la venta en todos los kioscos, donde es distribuida con cargo a los editores de periódicos. El público, encantado de la maravillosa novedad tecnológica, se adapta y compra la Lupa. Envalentonados por el éxito de esta iniciativa, MacroPrensa comienza a cambiar el juego de caracteres periódicamente, todos los años, y después todos los semestres; las viejas Lupas ya no pueden leer los nuevos periódicos y hace falta renovarlas con grandes gastos cada dos o tres meses. Un competidor de MacroPrensa ve ahí una gran ocasión para producir una Minilupa mucho menos costosa que la Lupa Macroprensa, y comienza a venderla en los kioscos. Pero los kioscos tienen un contrato de exclusividad con MacroPrensa y rehusan distribuirla. Peor aún, MacroPrensa demanda al competidor ante la Justicia por violación de los derechos de autor,

pues lo considera culpable de haber analizado los caracteres klingonianos a fin de construir la Minilupa. Y gana...

...no está muy lejos

Pero qué idiotas, dirá usted, quién puede dejarse hacer eso? Pues bien, permítame decirle que el mundo de los TecnoCretinos no está muy lejos. Hace dos años quise presentar a la UE una solicitud de financiación para la visita de un investigador inglés a nuestro laboratorio. Para eso busqué el formulario, y me dijeron que la manera más fácil de proceder era obtenerlo desde el servidor de Web www.cordis.lu de la comunidad europea, ya que el correo normal podía tardar algún tiempo considerable. Di así con un documento que se llamaba machin.doc y que estaba escrito con Microsoft Word para Windows versión vaya-usted-a-saber.

En Klingoniano. No hay problema, me dije a mí mismo, tenemos un MacIntosh en el laboratorio con la lupa Microsoft Word versión 6.0. Esta es de la misma empresa, la más reciente, luego podré leerlo bien. Cuando pensé esto eran las 10 de la mañana. Para mi gran sorpresa, Microsoft Word en MacIntosh, después de una docena de minutos de ``conversión", bloqueó la máquina y me vi obligado a apagar y volver a encender, perdiendo mi trabajo. Así comenzó una verdadera batalla con la Lupa, donde al final salí vencedor pero agotado a eso de las 19 horas, con una versión del formulario rellenado, obtenido imprimiendo las páginas una a una y con manipulaciones complejas en cuyos detalles no entraré. Basta decir que me entraron muchísimas ganas de llevar esto ante la Justicia, pero sin muchas esperanzas de ganar. Todo esto por qué? Por un formulario extremadamente simple con las casillas Nombre, Apellido, etc., que lo habríamos podido preparar muy fácilmente con un formato de archivo libre y público, tal como el HTML que se utiliza desde 1991 en la Web. Han pasado ya dos años y en http://www.cordis.lu nada ha cambiado. El aspecto es muy atractivo, pero los formularios y la documentación que contiene información que debe ser libre y gratuita y que son de importancia vital, están todavía presentados solamente en formato privado, típicamente Microsoft, e, increíble pero cierto, compatible solamente con los productos Microsoft para PC. A causa de esto, nuestro laboratorio pronto comprará un gran PC con Windows 95 y Microsoft Office, solamente para poder leer los documentos de la UE. La Lupa Klingoniana avanza. Además, con esta Lupa el formato de archivos cambia de versión en versión, de tal suerte que Word 5.0 no puede hacer nada con los archivos de Word 7.0, y peor aún, el Word 6.0 en Mac tiene problemas para leer archivos de Word para Windows. Hemos caído en la trampa! No es suficiente con comprar Microsoft Word una vez; deberemos pagar de nuevo cada versión, sólo para poder continuar leyendo los archivos nuevos de otros. Y si por azar habíamos comprado un producto complementario para la versión 5.0, por ejemplo un diccionario en español, habrá que comprar uno nuevo en la nueva versión; la vieja será ``incompatible", aún cuando el español no haya cambiado entretanto. Advertirá que se trata de un verdadero y limpio secuestro de nuestra información: una vez que los datos entran en Word o Money, ya no hay manera fácil de recuperar todo el trabajo que usted ha hecho para transferirlo a otro programa si decide no comprar más productos Microsoft. Se cuidaron muy bien de no suministrar convertidores eficaces hacia otros formatos[+]. Además intentaron varias veces hacer que se aprobaran leyes prohibiendo a los competidores la utilización de sus formatos propios de archivos, o incluso su análisis. Si se llegaran a aprobar estas leyes, una empresa que venda una Minilupa

convertidora sería culpable de violación de las leyes de derechos de autor[+]. Pero son nuestros datos los que están en juego. Bienvenidos al país de los TecnoCretinos!

Prácticas dudosas Resumiendo la técnica es simple: por un lado se hace caer a los consumidores en la trampa secuestrando su preciosa información en un formato propietario, el cual es constantemente ``actualizado". Debido a estas modificaciones, los usuarios se ven obligados a comprar cada 6 ó 12 meses una actualización de todas sus aplicaciones, tan sólo para poder continuar leyendo sus propios datos o acceder a información que (de manera innecesaria) es suministrada bajo este formato privado. Por otro lado, se entrampa a los competidores: no se les da la documentación [del sistema operativo] y se introducen variantes arbitrarias con la única meta de no permitir que los productos que ellos desarrollan funcionen correctamente. Es más, si la competencia llega a descubrir que una de las modificaciones tenía como único fin el hacer funcionar su producto con menor eficiencia que el producto equivalente del monopolista, son condenados por haber hecho ``ingeniería al revés" (reverse engineering), el equivalente informático a desmontar el motor de un Twingo para ver cómo está hecho. Esta última técnica es especialmente poderosa si el editor de software detenta a la vez el sistema operativo (Windows 95) y las aplicaciones (MS Word, Excel, etc). En tal caso es técnicamente posible modificar el sistema para tornar inestables o inutilizables los productos de la competencia, y a la vez mejorar las prestaciones de sus propios productos. Es lo que se ha hecho en Windows NT Workstation, limitando artificialmente a diez los accesos simultáneos a la máquina; esto hace inutilizable el servidor Web de Netscape sobre NT Workstation (ver [13] y [14]). Si quiere resolver esto, deberá comprar la versión Windows NT Server, muchísimo más cara, la cual incluye gratuitamente un servidor de Web de Microsoft. La treta pone fuera de juego a Netscape. Esto es simplemente maquiavélico, y aún más cuando descubrimos que las dos versiones, NT Workstation y NT Server, son prácticamente idénticas y sólo se diferencian en un puñado de líneas, tal y como se documenta en [15] y [16]. El resultado final de estas prácticas dudosas es simple: se impide que el usuario pueda elegir otra cosa que no sea un producto Microsoft. Junto con la reducción a cero de los costos y de los riesgos, tal cual como vimos anteriormente, esto permite al monopolio establecer un verdadero impuesto sobre la información, donde Microsoft es el único beneficiario. Después de todo, si Bill Gates ha sido recibido con honores dignos de un Jefe de Estado en el Elíseo, se debe a que se trata de la visita de la versión ``cyber" del recaudador de impuestos. Un impuesto que no tiene nada de virtual: enormes sumas de dinero salen de la Comunidad Europea cada año en contrapartida por productos de mala calidad que nos vuelven más y más dependientes de la mala tecnología del otro lado del Atlántico. Es más, estos productos se distribuyen en Europa a precios exorbitantes, muy superiores a los precios americanos o canadienses. No se deje engañar por los que le dicen que los programas en Europa son más caros porque necesitan ser traducidos. Si echa un vistazo al servidor Web de Microsoft, se enterará de que consideran ``ilegal" (sic) comprar su software en versión francesa en Canadá (en donde es mucho más barato que aquí) para utilizarlo en Francia [17]. Y el ``libre" mercado? Nos ordeñan como a las vacas lecheras, y la pasividad de los gobiernos europeos, que comienza a parecerse bastante a la cooperación activa si uno piensa en http://www.cordis.lu, es absolutamente inexplicable, visto el tamaño de este verdadero expolio.

[R04] Transparencia - Perennidad

Los datos deben ser almacenados de forma tal que el acceso a ellos por parte de las personas e instituciones autorizadas esté garantizado durante toda la vida útil de la información. En el caso del Estado, cientos de años.

Para garantizar la perennidad de los datos públicos, es indispensable que la utilización y el mantenimiento del software no dependan de la buena voluntad de los proveedores, ni de las condiciones monopólicas impuestas por éstos. Por ello el Estado necesita sistemas cuya evolución pueda ser garantizada gracias a la disponibilidad del código fuente.

Riesgos del Software Propietario:

- La posibilidad de quiebra de las empresas privadas puede dejar inaccesibles sus formatos.
- Los usuarios cautivos de formatos antiguos deben adquirir nuevas versiones para mantenerse al día y comunicarse con otras personas.

Que el software contenga mecanismos temporales de inhabilitación automática o controlada desde el exterior. ``si no pagas, lo apago", ``si me haces la guerra, lo desconecto".

[R05] Transparencia - Homogeneidad

El Estado no es una entidad única, sino que está compuesto de múltiples organismos con diversos grados de autonomía de decisión.

Por eso, se requieren normas generales basadas en la ley que impidan que una decisión discrecional de cualquier funcionario, a cualquier nivel, en cualquier organismo ponga en riesgo la información que pertenece a los ciudadanos.

Todos sus estamentos y niveles suelen requerir soluciones similares y deberían formar un conglomerado para financiar el desarrollo de una solución común a su problemática, y compartirla entre todas. Esto sólo es posible con Software Libre.

[R06] Participación

Toda persona que participa en una actividad necesita tener la libertad de poder involucrarse en ella y regular el grado de involucramiento. Un Estado debe tener el derecho a participar en cualquier actividad de la que dependa su misión fundamental.

El software propietario, obliga al Estado a renunciar a tomar decisiones, ya que éstas le son dictadas por el dueño del programa. Estas decisiones van desde la plataforma de hardware, hasta los programas a usar para tareas relacionadas. El derecho-habiente se asegura de que su software funcione mejor o únicamente, interactuando con otros de su propiedad. Sólo él tiene la facultad de corregir errores, agregar funcionalidad, o quitarla. Además de la obvia e inaceptable dependencia de un único proveedor que esta restricción implica, el Estado queda sin curso aceptable de acción en muchas situaciones

[165]:

- Ausencia, o demora en la corrección de problemas: las prioridades del proveedor de software no son necesariamente las mismas de sus clientes. Si determinado problema no está en la lista de prioridades del proveedor, o si éste se rehúsa a corregirlo o exige una compensación desmesurada para hacerlo, el Estado no tiene siquiera el recurso de utilizar sus propios medios para obtener una corrección por parte de un tercero.
- Incompatibilidad con versiones previas: son conocidos los casos de versiones ``mejoradas'' de programas que tienen problemas leyendo datos de versiones anteriores. El cortísimo ciclo de obsolescencia del software, motivado mucho más por razones de marketing que por efectiva demanda de nuevas funcionalidades, obliga a los usuarios a mantener su software actualizado, y el precio de la actualización incluye (a menudo sin el conocimiento de los usuarios) la renuncia a acceder a datos valiosos almacenados en archivos.
- Desaparición del proveedor o del producto: abundan los ejemplos de programas propietarios cuyos usuarios se vieron obligados a emprender costosísimas migraciones debido a la quiebra del proveedor, a su adquisición por otro más grande, o a la simple discontinuidad del producto por decisión unilateral del autor.
- Dificultades para garantizar la perennidad de los datos, ya que la obligación de acompañar al proveedor en sus decisiones puede llevar a situaciones insostenibles (por ejemplo, a través de una carrera desenfrenada de actualizaciones de hardware).

El Software Libre permite que cualquier Estado o particular se involucre al nivel que desee en su proceso de construcción y desarrollo, permite al usuario corregir y modificar el programa para adecuarlo a sus necesidades.

Esta libertad no está destinada solamente a los programadores. Si bien son éstos los que pueden capitalizarla en primera mano, los usuarios también se benefician enormemente, porque de esta manera pueden contratar a cualquier programador (no necesariamente a la empresa propietaria) para que corrija errores o añada funcionalidad. Las personas que puede contratar no sólo no tienen exclusividad alguna sobre la posibilidad de contratación, sino que tampoco la adquieren a partir de sus modificaciones. De esta manera, el usuario puede asignar sus recursos a resolver sus necesidades de acuerdo a sus propias prioridades, pidiendo varias cotizaciones y quedándose con aquella que le ofrezca mejor relación precio/prestación, sin exponerse a chantajes, extorsiones, monopolios y relaciones comerciales injustas.

Está Roto, y No Se Puede Arreglar [103]

Pero ante esta situación el técnico apenas alcanza a diagnosticar: ``se colgó". Lamentablemente, los profesionales locales no pueden dar respuesta a estos problemas, porque el conocimiento necesario para darla está restringido a los empleados del propietario de los programas en juego. Es cierto: los propietarios ofrecen onerosos cursos en los que capacitan profesionales para resolver problemas, pero ellos dictan la profundidad de esos cursos, nunca revelan todos los detalles, y no proveen ninguna manera de corroborar que lo que enseñan es realmente correcto. En suma, nadie sabe exactamente qué pasa, solamente sospecha. Y aún si una de estas sospechas fuera correcta, aún en el improbable caso de que alguien, fortuitamente, descubriera la causa de un determinado error y pudiera eliminarlo para siempre... ¡Tendría prohibido hacerlo!

El software propietario reduce a los profesionales locales al papel de reinstaladores y reseteadores

---

[R07] Interoperabilidad

Libertad de elección de software, hardware, soporte y formación

Es conveniente que para cada necesidad operativa existan diversas alternativas y que éstas puedan funcionar con los mismos datos y formatos. El Estado tiene la libertad de fijar una política respecto de la tecnología de información que emplea dentro de los límites impuestos por su función. Debe preservar la neutralidad tecnológica.

El mundo del software propietario se encamina cada vez más hacia monopolios en cada segmento y lo que es peor, monopolios verticales que traban la posibilidad de usar software de una empresa en un área y de otra en un área diferente, ya que no se podrían comunicar entre sí.

Mediante la técnica de incorporar programas aislados al sistema operativo, Microsoft va extendiendo su monopolio a otras áreas. Antes el entorno gráfico era un programa, hoy es el propio sistema operativo. El navegador web fue un programa independiente, luego, para eliminar la competencia de Netscape, lo incorporaron a Windows.

En el mundo libre existen alternativas para cada segmento: sistema operativo GNU/Linux, Hurd o BSD; escritorio de trabajo KDE o GNOME; motor de bases de datos MySQL o PostgreSQL, etc. Cada día estas variantes operan mejor entre sí. Por otro lado cada novedad incorporada en uno es rápidamente duplicada en el otro, utilizando las mejores técnicas y sin reinventar la rueda.

[TCO] Costo total de posesión

Cuando se analiza el precio de una solución tecnológica se suele hablar del TCO. El coste del software es especialmente importante en la Administración pública, ya que se habla de dinero público [013]. No

debe tomarse el costo como el motivo central de la opción por el Software Libre, pero sin duda es un motivo importante.

Según un estudio de la consultora Robert Frances Group publicado en el año 2002, el coste total de propiedad del sistema operativo libre Linux era menos de la mitad que el de Windows. En el estudio se analiza el coste de diferentes servidores durante un período de tres años y se constata que gran parte del ahorro proviene de no tener que pagar licencia por el Software Libre y de sus menores costes de administración. Sin embargo, también deben considerarse otros aspectos positivos del Software Libre, como la independencia del proveedor. En el mismo sentido se expresa un estudio realizado por la consultora Consulting Times, en este caso sobre el coste de propiedad de sistemas de correo: también concluye que las soluciones basadas en Software Libre son mucho más económicas en todos los casos planteados.

Existe otro estudio auspiciado por Microsoft que indica que el TCO es mayor con Software Libre. Este estudio se basa en que la mano de obra para el Software Libre es más cara. Desde ya que este estudio toma el valor de la mano de obra en EEUU, mucho más alta que en el tercer mundo, mientras que las licencias tienen precio uniforme. Por otro lado sería realmente bueno que el uso de Software Libre vuelque más recursos a la gente que a las licencias. Pero más allá de esto, el estudio no tuvo en cuenta que un administrador GNU/Linux puede manejar muchos más servidores que uno de Windows, debido a la automatización inherente al Software Libre.

[R08]:[TCO] - Costo total de posesión - Licencias

Se requiere una infraestructura tecnológica con una estructura de costos razonables para poder alcanzar la máxima eficiencia económica. Nada tiene que ver la cantidad de computadoras en que se usa un software o la cantidad de usuarios, con su costo de creación. Esta es una medición ficticia del costo (o de su escasez económica), ya que el costo de creación es indistinto del número de copias. El software, una vez creado no es escaso. Sólo el Software Libre se adapta a esta realidad económica.

El Software Libre no es necesariamente gratis. Puede tener un costo, sobre todo si es necesario desarrollar o modificar un programa, pero una vez creado cualquiera es libre de usarlo en todas sus computadoras.

Según los sistemas instalados, sus costos, y las herramientas disponibles para reemplazarlas, el ahorro en licencias al usar Software Libre puede ser realmente importante.

Los requisitos de hardware necesarios para cada alternativa deben ser incluidos en el TCO. En general GNU/Linux posee unos requisitos de hardware inferiores que Windows, para la misma función.

[R09] [TCO] Costo total de posesión - Legales

La prestación de los servicios del Estado no es optativa, ni admite demoras ni obstáculos. Se requiere un sistema legal que permita usar el software disponible en cualquier lugar y de la mejor forma posible.

El sistema de licencias del software propietario es muy difícil de mantener y asegurar su cumplimiento en todo el Estado. El sistema de compra de los Estados hace casi imposible usar marcas en los pliegos de compras, realizar acuerdos de pago de licencias basados en la cantidad de empleados, adquirir software propietario y mucho menos hablar de pagar multas y otras acciones habituales asociadas al uso de software privativo. Los funcionarios estatales que usan software privativo, sin tomar los costosos recaudos de tener una licencia por cada computadora, están expuestos a acciones legales y problemas patrimoniales de todo tipo.

Los distintos modelos de licenciamiento propietario son fuente constante de confusión acerca de la legalidad de usar cierto programa para determinado propósito, en determinada computadora, por parte de un determinado grupo de usuarios, situación que se complica significativamente cuando hablamos de usar una combinación de varios programas. Esto conforma un grave riesgo ya que un malentendido sobre términos de licenciamiento, un cambio en éstos, su caducidad o una suba de precios pueden llevar a que el Estado deba suspender la prestación de un servicio por carecer de las licencias necesarias. [165]

Los problemas del Software Propietario no tienen su origen en las características técnicas del software, sino en su modelo de licenciamiento propietario que impone prohibiciones expresas, o restricciones de orden práctico insuperables que impiden cumplir las pautas de dichas licencias.

El Software Libre evita este problema y ahorra posibles juicios y controversias contra el Estado y sus funcionarios.

El modelo de licenciamiento es más importante que el precio o la tecnología. Las licencias a usar en el Estado debieran permitir sin límite de tiempo, ni en la cantidad o tipo de computadoras, ejecutar, estudiar, mejorar, y distribuir el programa de acuerdo a las necesidades del Estado, y no las del proveedor. Sólo el Software Libre se licencia de dicha forma.

En cuanto a la producción de software se requiere organizar un sistema de tracking de código fuente, necesario para establecer quién es autor de qué y sobre qué bases. Organizar esto es altamente costoso y complejo, tanto en el Software Libre como en el Propietario, pero sobre todo en el caso de producción de software basado en el modelo bazar.

[R10] Calidad - Técnica

Se requiere un software de calidad, eficiente, sin fallas, productivo, confiable y estable, que funcione las 24 hs., los 365 días, todos los años. Que aproveche los pocos recursos disponibles, que permita trabajar a muchos usuarios en diversas terminales, que mantenga los datos íntegros, cuyas actualizaciones, correcciones o mejoras no requieran reinstalar todo, o cambiar de hardware y que asegure una amplia variedad de software disponible.

En el software propietario, sólo el fabricante puede solucionar los problemas que puedan surgir, y ello dependerá exclusivamente de la capacidad y la disponibilidad de su departamento de desarrollo.

Se Cayó el Sistema [103] Nadie se asombra ya de perder horas de trabajo porque debió reiniciar su sistema, ni de que sus datos desaparezcan (junto con los de varios colegas) debido a la acción de un virus, ni de las colas detenidas porque la computadora no responde, o cuando de repente el programa que opera se ``cuelga" sin razón aparente. El usuario está resignado, y acepta estos problemas como parte del precio a pagar por el uso de la herramienta simplemente oprime el botón de ``reset". Ninguna de estas fallas es inherente a las computadoras: son tan solo la expresión tangible de la impotencia del usuario final ante las fallas de un mecanismo sobre el que no tiene ningún control, y del que depende para poder llevar a cabo su tarea.

El Software Libre, al ser público, está sometido a la inspección de una multitud de personas, que pueden buscar problemas, solucionarlos y compartir la solución con los demás. Debido a esto, y a lo que se llama ``el principio de Linus" (dada la suficiente cantidad de ojos, cualquier error del software es evidente), los programas libres gozan de un excelente nivel de confiabilidad y estabilidad, requerido para las aplicaciones críticas del Estado[148]. Son comunes los casos en que un error de seguridad en GNU/Linux se hace público y se corrige en pocas horas.

El Software Libre se caracteriza por tener ventajas evidentes en cuanto a:

- estabilidad,
- rápida corrección de bugs (errores de programación),
- capacidades de operar en red, como servidor y como medio de comunicación,
- amplia variedad de software disponible,
- existencia de derechos de usuarios distintos, y claves de acceso particulares,
- desaparición de las famosas pantallas azules de error,
- existencia de un administrador del sistema distinto al usuario,
- flexibilidad de configuración remota,
- actualizaciones periódicas,
- la posibilidad de participar e influir en el proceso de creación de software
- menos hardware para la misma tarea,
- eficiencia,

- buenas cualidades multitarea, multiproceso, multiusuario,
- optimización,
- escalabilidad,
- eliminación en la práctica del riesgo de ataques de virus.

[R11] Calidad - Servicio Técnico, soporte.

Es indispensable contar con buen servicio técnico y soporte en cuanto a software. Debe darse lugar a la aparición de empresas independientes de soporte en un mercado en libre competencia.

La comunidad que se forma alrededor del Software Libre ha sido su mayor fortaleza, con la construcción de canales de consultas de diferentes niveles en cada región y país. Y con esto, la posibilidad de acceder en última instancia al autor y al código fuente. Lo que permite un soporte comunitario y la aparición de empresas independientes especializadas en soporte.

El software propietario impide contratar servicios técnicos a otros proveedores, sus servicios técnicos o son muy caros o muy malos.

[R12] Seguridad

Mucha de la información que maneja el Estado es estratégica y puede ser peligrosa si cae en manos incorrectas y no autorizadas.

Es necesario disponer de sistemas seguros y de redes de comunicación que operen sin interrupciones, virus y crackers; evitar la visión de los datos por terceros (Riesgo de filtración), los datos confidenciales deben ser tratados de tal manera que el acceso a ellos sea posible exclusivamente para las personas e instituciones autorizadas; y reducir toda posibilidad de modificación de datos por terceros (Riesgo de manipulación). La modificación de los datos debe estar restringida, exclusivamente a las personas e instituciones autorizadas.

Para garantizar la seguridad nacional, es indispensable contar con sistemas desprovistos de elementos que permitan el control a distancia o la transmisión no deseada de información a terceros (puertas traseras o bombas de tiempo). Esto puede ocurrir con el conocimiento y el consentimiento de la empresa propietaria del software, por accidente, omisión u error, por decisión autónoma de algún empleado o por influencia de agencias de seguridad nacional del Estado en el cual se asienta la empresa.

La seguridad por la oscuridad ha fallado sistemáticamente. Esta filosofía se basa en la idea de brindar seguridad mediante el desconocimiento. En que el atacante no sepa cómo funcionan los mecanismos de protección. Lo ideal es que solo se deba esconder las claves de acceso. Que existan pocos puntos

posibles de ataque. Al emplear mecanismo de seguridad cuyo fundamento es el secreto, se puede acceder descubriendo la clave o descubriendo el funcionamiento del protocolo de seguridad.

Se requieren sistemas cuyo código fuente sea libremente accesible al público para permitir su examen y debate por el propio Gobierno, la oposición, los ciudadanos y expertos independientes en el mundo.

Solo así el Estado puede corregir una falla. Este tipo de inspección sólo es posible con Software Libre. El conocimiento del código fuente eliminará el creciente número de programas con código espía.

La alternativa que ofrece Microsoft a algunos Estados, cuando muestra sus códigos, no es suficiente, pues impide la recompilación del mismo y nada asegura que lo que se muestra sea efectiva y realmente lo que se ejecuta. Por otra parte también la oposición y los ciudadanos deben poder auditar que hace el Estado.

La empresa Mitre ha elaborado un estudio por encargo del Departamento de Defensa de los Estados Unidos donde se analiza el uso de Software Libre y de código abierto en sistemas que se encuentran en producción en este departamento [013]. Las conclusiones son claramente favorables a seguir incrementando el uso del Software Libre, y se destaca la posibilidad que ofrece de solucionar errores de seguridad de forma inmediata sin depender de un proveedor externo. Según este informe, hay más de 115 aplicaciones de Software Libre en uso en el Departamento de Defensa, con más de 250 ejemplos de su empleo en diferentes entornos.

Sólo el Software Libre permite manejar la seguridad en forma abierta y construir un espacio virtual seguro y libre a la vez.

[R13] Capacidad de promover el cambio

Las organizaciones deben ser capaces de adaptarse a las condiciones cambiantes de su entorno para mantener su rumbo, alcanzar sus objetivos y servir sus propios intereses y responsabilidades. Las burocracias estatales son conocidas por su resistencia al cambio.

El uso del Software Libre requiere cambios, formación de personas y políticas de reentrenamiento y adecuación. Requiere superar la comodidad de una cultura habituada a un estado de cosas en que los ciudadanos pagan por el software que se usa y no se informan de lo que sucede tras bambalinas.

[R14] Economía

El software a ser promovido por el Estado debe permitir y contribuir con la creación de una economía sustentable y solidaria.

El modelo económico del Software Libre sustituye el modelo de negocios tipo recaudador de impuestos internacional único (impuesto por Computadora Personal o usuario), con soporte policial público, por otro donde diferentes tamaños de organizaciones y empresas brindan servicios en niveles locales, regionales, nacionales, comunitarios, y globales.

[R15] Economía - balanza de pagos

El software usado por el Estado debe tender a mejorar la balanza de pagos. Cada cajita de colores con una licencia de Microsoft Office que se importa equivale a cientos de horas hombre de trabajo local necesarias para exportar otros bienes.

La mayoría de las ventas de licencias de software del mundo van hacia EEUU. Es el único país que se beneficia de este sistema y que cuenta con balanzas de pago favorables en este rubro. En el resto del mundo, la industria local se basa en distribuir y dar apoyo y formación a productos realizados en EEUU. En el mundo propietario la inmensa mayoría del software es desarrollado en empresas con casas matrices en EEUU, donde están los profesionales más cualificados que desarrollan Software Propietario.

El Software Libre permite un importante ahorro de divisas.

[R16] Economía - Industria local

Trabajo, crecimiento, distribución del ingreso

El software usado por el Estado debe impulsar el crecimiento de la economía local.

En el ámbito de la Administración Pública, una parte importante de la inversión en software se realiza en licencias de sistemas operativos, servidores, bases de datos y paquetes de ofimática, que son producidos en el exterior y que sólo tienen repercusión económica local en los márgenes de distribución.

El software propietario se distribuye uniformemente, y muchas veces no se adapta a las necesidades específicas de empresas y administraciones. Una gran parte de la industria del software se basa en desarrollar proyectos donde se requiere software personalizado.

El Software Libre permite personalizar los programas tanto como sea necesario, gracias a que disponemos del código fuente. La personalización es un área muy importante en que el Software Libre puede responder mucho mejor que el software de propiedad a unos costes mucho más razonables. Es posible desarrollar internamente las mejoras o las modificaciones necesarias, en vez de encargarlas a empresas de otros países que trabajan con licencias propietarias. De este modo, se contribuye a la formación de profesionales en nuevas tecnologías y al desarrollo local bajo los propios planes estratégicos.

El Software Libre mejora la competitividad de la industria local de software, promoviendo el desarrollo tecnológico y la creación de puestos de trabajo más especializados.

El Software Libre mejora la competitividad de la industria local de software, promoviendo el desarrollo tecnológico y la creación de puestos de trabajo más especializados.

En el ámbito de la Administración Pública, una parte importante de la inversión en software se realiza en licencias de sistemas operativos, servidores, bases de datos y paquetes de ofimática, que son producidos en el exterior y que sólo tienen repercusión económica local en los márgenes de distribución.

El software propietario se distribuye uniformemente, y muchas veces no se adapta a las necesidades específicas de empresas y administraciones. Una gran parte de la industria del software se basa en desarrollar proyectos donde se requiere software personalizado. El Software Libre permite personalizar, gracias al hecho de que disponemos del código fuente, los programas tanto como sea necesario hasta que cubran exactamente nuestra necesidad. La personalización es un área muy importante en que el Software Libre puede responder mucho mejor que el software de propiedad a unos costes mucho más razonables. Es posible desarrollar internamente las mejoras o las modificaciones necesarias, en vez de encargarlas a empresas de otros países que trabajan con sistemas de licencia de propiedad. De este modo, se contribuye a la formación de profesionales en nuevas tecnologías y al desarrollo local bajo los propios planes estratégicos.

Por otro lado, todas las mejoras que se realicen no tienen restricciones y se pueden compartir con cualquier otra administración, empresa, institución u organismo que las necesite. En el software de propiedad, estas mejoras o no se pueden llevar a cabo o quedan en manos de la empresa creadora, que normalmente se reserva los derechos de uso y propiedad intelectual y establece en qué condiciones las comercializará.

[R17] Economía - Libre Competencia

Siempre fue función de los Estados capitalistas modernos evitar el monopolio.

En los últimos años el software propietario ha ido concentrándose tanto horizontal como verticalmente. Las técnicas usadas por estas empresas para lograrlo se basaron en las restricciones de libertades que impone el software propietario. Las empresas que desarrollan este tipo de software tienen a su alcance los siguientes mecanismos:

- practicar políticas de precios diferenciales para castigar a los que se oponen; mal informar y explotar a los usuarios;
- usar cláusulas de exclusión en contratos para sostener su censura a la publicación de errores y defectos;

- apagar o bloquear los canales de distribución de competidores legítimos;
- anunciar falsos avances (``vaporware") para evitar la adopción de productos reales competitivos;
- frustrar, oponerse y burlarse de oficiales gubernamentales que protegen el interés público;
- limitar la libertad de elección; producirle confusión y frustración a los usuarios al venderle productos inferiores;
- ocultar el código fuente para mantener a los desarrolladores divididos, privados de derechos y dependientes. Hacer, mediante el ocultamiento del código fuente, que los desarrolladores deban re-inventar la rueda permanentemente.
- tomar estándares abiertos, adoptándolos y extendiéndolos, o contaminándolos de otras formas, para romperlos y apropiárselos;
- Aprovechar estrangulamientos económicos naturales para su propio beneficio;
- manipular y demorar el progreso tecnológico para mantener la supremacía;
- usar contratos excesivamente restrictivos y excluyentes contra competidores menores;
- atar productos inferiores a los dominantes;
- violar y evitar órdenes judiciales de forma desafiante;
- aplastar emprendimientos competitivos y prometedores;
- destruir mercados para eliminar la competencia real;
- transformar a sus clientes en objetos-mercadería a través de la cautividad;
- ejercer comportamiento extorsivo en sus tratos comerciales;
- obligar a los competidores débiles a destruir sus propios productos innovativos para proteger a los ya establecidos;
- tomar innovaciones ajenas como propias
- no responder a las necesidades y pedidos de los clientes en tiempo razonable;
- deshumanizar a los desarrolladores de software tratándolos como ``ingresos" o ``activos";
- forzar a sus usuarios a dejar pasar tecnologías competitivas y prometedoras;
- esconder sus errores de programación, arriesgando la seguridad y estabilidad;
- suprimir la naturaleza abierta, eficiente y libre del método científico a través del ocultamiento del código;

- usar características no documentadas como un dispositivo anti-competencia;
- y realizar otros actos impropios, antisociales - contra el interés público y social - y anticompetitivos, para establecer, mantener y extender sus monopolios de software;
- adquirir empresas innovadoras y discontinuar sus productos.

[R18] Economía - Sustentabilidad

La creación y modificación de software es una actividad constante. Debe asegurarse un esquema que sostenga su desarrollo en forma sustentable contemplando todas sus etapas. Los Estados deben propulsar un Software tecnológicamente apropiado.

El software propietario se basa en un modelo económico centralista que requiere grandes concentraciones de capital para desarrollar nuevo software junto a flujos financieros internacionales considerables y que depende de maquinarias policiales Estatales para recaudar tasas de uso.

La comunidad de Software Libre es conformada por prosumidores, que crean y usan software tecnológicamente apropiado a las necesidades de la ciudadanía digital moderna y cuyo crecimiento, evolución y adaptación es totalmente sustentable tanto en lo económico, en lo social, y en lo ambiental.

El software propietario se basa en un modelo económico centralista que requiere grandes concentraciones de capital para desarrollar nuevo software junto a flujos financieros internacionales considerables y que depende de maquinarias policiales Estatales para recaudar tasas de uso.

[R19] Economía - Igualdad

El Estado debe favorecer la participación igualitaria de todas las personas en la creación de software, siempre que tengan aptitud y deseos para ello.

Crear software propietario implica trabajar para empresas multinacionales o contar con grandes capitales para pagar programadores que reinventen todo el software de base para cada nuevo proyecto, o paguen los royalties correspondientes.

Todos los desarrolladores del mundo pueden participar en la construcción del Software Libre en forma igualitaria. La única inversión de capital es una PC y una conexión a Internet.

[R20] Tecnología - Innovación

Favorecer la innovación tecnológica como mecanismo esencial de progreso es una de las misiones fundamentales de un Estado moderno.

En los últimos años se ha formado una comunidad internacional abierta que comparte conocimiento y de la cual surgen continuamente innovaciones, especialmente en el campo de Internet, que fue moldeada al calor de ésta comunidad. Internet es el marco fundamental para la innovación en cuanto a Software e influye en muchas otras áreas de la ciencia.

El software propietario ha quedado a la saga en cuanto a innovación, intentando permanentemente adaptarse y planeando estrategias para no quedar afuera de la red e impedir que surjan competidores.

En el mundo del software propietario, las licencias de software, la propiedad intelectual y otras herramientas legales y técnicas se utilizan para impedir que terceros participen en ese conocimiento y para que éste continúe siendo patrimonio exclusivo de la empresa que lo creó. La innovación pertenece a una empresa.

El software propietario requiere re-inventar la rueda permanentemente. Numerosas tecnologías se crean y condicionan solamente para poder mantener la compatibilidad de código ejecutable viejo con nuevos procesadores. Se justifican técnicas de objetos, software como java, y otros, en la necesidad de reusar código precompilado. El acceso a las fuentes es la única forma razonable de no replicar y volver a crear código que ya se escribió.

En el mundo del Software Libre, el conocimiento pertenece a la humanidad. Permite construir conocimiento en forma dinámica y en donde siempre se avanza sobre las espaldas de los que dieron los pasos anteriores.

Internet ha sido construida con Software Libre desde sus inicios. Los protocolos que definen la arquitectura de Internet son abiertos y no son controlados por una empresa. Esto es lo que definió Internet como un espacio de Libertad, lo que le dio su carácter y lo que le permitió consolidarse como la red mundial única.

El modelo del Software Libre, donde prima el compartir la información y el trabajo cooperativo, es similar al que tradicionalmente se ha usado en el mundo académico y científico. En estos ámbitos, los resultados de las investigaciones se divulgan en publicaciones científicas, y sirven de base para nuevas investigaciones. Éste es principalmente el modelo sobre el que la humanidad ha innovado y avanzado.

La única opción compatible con la cultura académica es el Software Libre.

[R21] Tecnología - Independencia y soberanía

El Estado debe estar al mando de la gestión de sus procesos informáticos y tener independencia de sus proveedores.

Uno de los grandes problemas del software es la dependencia del usuario con el fabricante. Lo que se acentúa cuando el creador no entrega el código fuente, ya que inevitablemente el usuario queda atado a él, para cualquier mejora que necesite [175].

Con software propietario, quien toma las decisiones es el dueño del copyright, o sea la empresa que lo provee, único capaz de acceder al código fuente. A menudo los proveedores de software propietario se ven obligados a dejar de fabricar un producto por un cambio drástico de las condiciones del mercado, o simplemente porque consideran que ya no podrán rentabilizar la inversión.

El software propietario trabaja habitualmente con formatos propios, cuyos mecanismos de almacenamiento no siempre son públicos, por lo que quizá no sea posible migrar el sistema cuando así se decida.

Estos problemas no son solamente técnicos. Aún cuando el Estado pudiese técnicamente modificar el software, la licencia limitada de uso lo impide.

La naturaleza del software propietario es claramente inaceptable para el Estado. El Estado está expuesto al chantaje (a arbitrarias decisiones de terceros privados) a través de la información que tiene almacenada en formatos propietarios secretos, al sabotaje a través de vulnerabilidades deliberadas o accidentales imposibles de prevenir sin acceso al código.

Es conocido que los programas de computadora a menudo contienen errores, y que no siempre se adecuan perfectamente a las necesidades del usuario. El usuario de software propietario, que ha adquirido meramente una licencia limitada de uso, no tiene otro remedio que recurrir al propietario del programa, en la esperanza de que éste se sienta inclinado a corregir la situación en tiempo y forma (recordemos que la licencia de uso del software es ``tal como está", de modo que el autor no tiene obligación de corregir eventuales errores). A veces, el usuario está en condiciones de incentivar al autor a corregir un error o agregar una función que necesita, pagándole, pero hay obstáculos:

- El usuario suele ser una organización más chica y menos poderosa que el propietario, de modo que su capacidad de incentivarlo es muy limitada (Esto pasa aún con los Estados).
- El propietario es el único que tiene derecho a corregir y modificar el programa, por lo que puede dictar precio, condiciones y plazos a su antojo.
- Aún cuando el usuario pague por el desarrollo del arreglo o de la nueva función, la propiedad intelectual sigue siendo exclusivamente del autor, que seguirá lucrando con la nueva funcionalidad adquirida con el dinero del usuario.

En otras palabras, el usuario está a merced del propietario del software, que puede atender sus necesidades o no de acuerdo a su sola discreción.

Peor aún, al estar los datos codificados en un formato secreto, el usuario depende absolutamente de que el propietario del software continúe permitiéndole el uso de los programas, porque de otra manera no

puede acceder a su propia información. El propietario del software tiene, a todos los efectos prácticos, la potestad de decidir si un usuario determinado puede acceder o no a los datos que él mismo elaboró.

El Software Libre garantiza independencia con respecto al proveedor gracias a la disponibilidad del código fuente. Cualquiera con los conocimientos adecuados, puede seguir ofreciendo desarrollo o servicios para un Software Libre. En el mundo del software propietario, sólo el desarrollador de la aplicación puede ofrecer todos los servicios. Disponiendo del código fuente, cualquier programador puede continuar su desarrollo y sus actualizaciones hasta que el cliente decida que es el momento adecuado de migrar a un nuevo sistema informático. Con Software Libre, la decisión última siempre queda en manos del usuario.

Mediante el uso de Software Libre, el Estado deja de tener sus sistemas controlados por una entidad externa (con frecuencia empresas extranjeras). De esta forma rompe la dependencia tecnológica que lo mantiene actualmente atado y obtiene las libertades que el Software Libre otorga.

[R22] DDHH - Seguridad personal. Espacio personal de privacidad

En muchos países existen leyes que regulan la protección de datos de las personas individuales. El software del Estado debe permitir el control de la información personal que almacene con seguridad y transparencia.

En el software propietario nunca se sabe si los programadores originales introdujeron a título personal, o por encargo de su empresa o agencias de seguridad, puertas traseras que ponen en peligro la seguridad del sistema o la privacidad de los datos. Son atentados a la privacidad: El registro del autor (firma) que hace Office sobre los documentos que escribe, impidiendo o dificultando el anonimato; La existencia de mecanismos como TCPA que autentifican todos los programas que corren en una computadora y los informan a una central, solicitando autorización; La recolección de datos de las películas vistas en un DVD y su acumulación en bases de datos.

Por su carácter abierto, el Software Libre dificulta la introducción de código malicioso, espía o de control remoto, debido a que el código lo revisan muchos usuarios que pueden detectar posibles puertas traseras.

El Software Libre protege la seguridad de los ciudadanos, tanto en su condición de titulares legítimos de la información que suministran al Estado como en su condición de usuarios, mediante una oferta extensa de software desprovisto de potencial código espía susceptible de poner en riesgo la vida privada y las libertades individuales.

[R23] DDHH - Libertades y Derechos. Libertad de expresión y comunicación

El derecho a comunicar y el derecho a conocer la tecnología que se usa permite construir un mundo libre y dinámico. Es esencial para el Estado garantizar al ser humano estos derechos.

El software es un discurso que debe ser comprendido en la libertad de expresión y comunicación. En un mundo cada vez más comunicado y con más redes horizontales, el derecho a la comunicación es parte esencial de las adquisiciones del ciudadano de la red y sus comunidades. Parte de este derecho es la posibilidad y capacidad de mantener el anonimato como contrapeso esencial del poder de las corporaciones en un mundo de comercio global.

Es esencial garantizar al ser humano estos derechos ya que le permiten tener las libertades esenciales de la vida en red. Entre otras cosas así se garantiza el derecho de asociación y el de organizarse libremente, a partir del uso de Internet con estos fines.

El software propietario imposibilita el conocimiento de cómo se ejecutan las acciones y toman las decisiones en un mundo regido por las computadoras. Al exigir el licenciamiento de tecnologías, restringe y limita en exceso las libertades de la gente y sus organizaciones para establecer mecanismos de distribución e intercambio de información impidiendo el derecho de comunicación.

Nunca puede sobrepasar el derecho comercial a los derechos humanos básicos. Es necesario un nuevo equilibrio.

Al ser abierto, el Software Libre garantiza el primer derecho en cuanto a la tecnología que le compete, la informática. También garantiza el segundo derecho brindando las herramientas fundacionales de la red a la humanidad en forma libre.

[R24] Sociedad. Estructura Social. Cultura. Otro Mundo es posible

Internet representa el comienzo de un nuevo ámbito global de interacción de las sociedades humanas. Se requiere un software que soporte las nuevas pretensiones de la humanidad para compartir y crecer solidariamente.

El tipo de código que se use determinará si sus mecanismos llevan a una sociedad abierta y democrática, o cerrada y estructurada, un ámbito de negocios en libre competencia o un sistema de monopolio absoluto.

La prohibición de que la gente comparta el software con sus compañeros o amigos es un grave contrasentido cultural para muchas comunidades.

El uso de Software Libre permite construir sociedades más justas y transparentes. El Software Libre es la única opción ante la luz de todas las ideologías políticas. Las razones para oponerse a su uso no tienen relación con las ideas sino con los intereses. Los únicos interesados en que se use software propietario son los propietarios de estos códigos, es decir, unas pocas empresas multinacionales. Para el resto del planeta, ideas e intereses se alinean en favor del Software Libre.

[R25] Sociedad - Brecha digital. Educación

Se debe poner especial énfasis en la educación como derecho y no como servicio. La educación no debe incluirse en los tratados de la OMC como servicio en ningún nivel, ni para las bibliotecas, los museos y otros reservorios culturales. La educación no debe subordinarse a las leyes del mercado.

La educación para la sociedad de la información incluye no sólo el e-learning y otras formas de educación a distancia, sino y sobre todo, uso y apropiación social de las TIC, complementado con libros y otros medios tradicionales de estudio, uso de la tecnología para afianzar el conocimiento local y regional y para fortalecer el rol del docente como facilitador del aprendizaje.

La educación en todos los niveles, formal e informal, de niños hasta adultos debe promover valores de cooperación, sólo posibles desde el uso de un software que garantice nuestros derechos como ciudadanos.

El Software Libre favorece la democratización en el acceso a la información y los sistemas del Estado, facilitando una comunicación multidireccional entre el Estado y su Comunidad. Asimismo, la libertad de copia y distribución del Software Libre le permiten al Estado la concreción de programas de alfabetización informática e inclusión tecnológica para la población utilizando mínimos recursos.

Compartir el conocimiento es uno de los fundamentos de la educación para la sociedad de la información, por lo que el uso de Software Libre es una condición indispensable.

[R26] Sociedad Géneros Diversidad

El Software Libre puede servir de herramienta en apoyo a la igualdad y contra la discriminación entre hombres y mujeres y entre los seres humanos entre sí.

Si nos planteamos políticas para contrarrestar la llamada ``brecha digital'', uno de los problemas más graves para afrontar en este sentido es la brecha digital desde la perspectiva de género. En América Latina y en otras muchas zonas del planeta las mujeres están doblemente discriminadas -por cuestiones

económicas, culturales y ante las dificultades tradicionales de aproximación a la ciencia y la tecnología y corren el riesgo de quedarse fuera de las políticas de apoyo al acceso a las nuevas tecnologías y a la Sociedad de la Información. Los gobiernos tienen responsabilidad directa en el diseño de políticas para oponerse a esta tendencia.

El Software Libre abre nuevas vías para la incorporación de las mujeres a las TIC. La versatilidad y la posibilidad de modificar el software permite una mejor adaptación a las necesidades específicas en cada caso y facilita el aprendizaje en el uso de las computadoras de quienes tienen mayores dificultades, entre ellos los colectivos de mujeres.

La reducción de costes para la creación de aulas informáticas, la posibilidad de utilizar Software Libre de edición ya diseñado para facilitar al máximo la comunicación de la ciudadanía en la Internet, al que puede accederse de manera gratuita para elaboración de contenidos con mayor facilidad, favorece sin duda una mayor participación de la sociedad y específicamente de las mujeres en esta nueva sociedad que se está conformando y en la que es responsabilidad de todos y de todas que nadie quede fuera.

El Software Libre puede efectivamente ayudar a manejar la diversidad humana; existen innumerables trabajos de investigación en el campo de las personas con capacidades diferentes e incluso distribuciones GNU/Linux que por defecto traen soporte para distintas de ellas.

[R27] Sociedad - Lenguas y pueblos del mundo

Los pueblos del mundo requieren software que funcione con su propia lengua.

Para las empresas productoras de software propietario, esta es una cuestión sólo de costos y negocios y no de cultura. Entonces, suelen exigir a las comunidades con pocos integrantes que paguen el desarrollo de versiones particulares de su software para luego terminar vendiéndoselas. Cada nueva versión requiere el mismo esfuerzo.

En el Software Libre cada comunidad es dueña de producir las modificaciones que requiera y apropiárselas sin más control que su propia voluntad. Los principales proyectos libres, como Open Office, KDE, Mozilla o GNOME, tienen numerosas traducciones, así como muchas otras aplicaciones, gracias a que no precisan autorización de ningún propietario y cualquier persona o institución puede realizarlas. En cambio, en el software propietario sólo la empresa productora posee los derechos para realizar la traducción. Además, si el programa que traducimos no dispone de corrector ortográfico en nuestra lengua podemos desarrollar un corrector particular o adaptarle alguno de los existentes en el mundo del Software Libre.

Por último, cabe destacar que cada vez que se crea un nuevo recurso lingüístico en el ámbito del Software Libre (una traducción, un diccionario, un glosario, etc.), éste, al quedar a la disposición de todo

el mundo puede ser reutilizado en futuras aplicaciones. En el software propietario, cada traducción y recurso lingüístico pertenece al fabricante y está restringido en su uso.

El Software Libre es probablemente la única opción que van a tener muchos países en vías de desarrollo para sumarse a las nuevas tecnologías.

[R28] Sociedad - Control

El software ¿es un bien o es tecnología cultural?, ¿quién debe controlarlo, la gente o las multinacionales?

El software es una lengua, específica para dar órdenes o describir procesos. Como toda construcción cultural humana que cada vez es apropiada y usada por más gente, es inaceptable que su control pase por una multinacional.

Se necesita un software que requiera leyes razonables. Y que garantice su uso conforme a leyes razonables, especialmente penales. No es sustentable un modelo en que la mayoría de los usuarios violan las leyes sistemáticamente. En estos casos la equivocada es la legislación, no la sociedad.

Sistemas previstos por Trusted Computing Group [146] son extremadamente peligrosos por su capacidad de crear una dictadura digital que haría palidecer al mundo de 1984 [072]. Le quitan al usuario el control de su computadora al sólo permitir la ejecución de programas autorizados por la central. Todas las decisiones las toman las compañías productoras de software y el usuario pierde todo derecho.

La existencia de leyes como Digital Millennium Copyright Act (DMCA)[147] hacen muy peligroso trabajar con software propietario, dado que cualquier característica que aparezca en él y que sea una falla de seguridad que permita acceder a datos con copyright, puede ser usada penalmente en contra del usuario.

[R29] Sociedad - Educación Tecnológica

Se debe utilizar un software que permita trasmitir el conocimiento contenido. Especialmente si se desea utilizar como sustancia para la educación. El software que use el Estado debe ser fácil de trasmitir para poder mantenerlo y reproducirlo. Es conveniente que sea material de estudio en las Universidades y ejemplo del estado del arte en cuanto a tecnología.

Es imposible estudiar informática sin acceso al código fuente. Usando software propietario sólo se tiene acceso a cajas negras inescrutables. Quienes así lo hacen devienen en meros técnicos que operan soluciones pensadas por otros y que no pueden alcanzar a conocer o modificar [Saravia:EI-03].

Preconceptos

Transparencia

- ¿No es suficiente con tener la información en formatos públicos?

  El Estado debe hacer pública su información y recibir datos en formatos públicos para garantizar la transparencia de sus operaciones.

  El uso de formatos abiertos en el Estado permite acceder libremente a la información que intercambiamos con el mismo (formularios, informes, boletines oficiales), y evita que el Estado obligue a sus ciudadanos a pactar con una empresa particular. Garantiza su futura accesibilidad y la independencia de los cambios de diseño que pudiera cometer quien controle un formato cerrado o no libre.

  Está muy bien que el Estado use formatos y estándares públicos para almacenar e intercambiar información. Pero si no existiese al menos un sistema libre que pueda acceder a la información almacenada quedaríamos igualmente limitados ya que no habría un sistema de referencia que permita el libre acceso a la información. Al usar software privativo o propietario, aún si este opera sobre formatos abiertos de información, quedan pendientes las cuestiones relativas al control de la información, la seguridad nacional y la independencia tecnológica

  Por lo tanto es necesario usar Software Libre junto con formatos estándares abiertos establecidos por normas públicas para garantizar la transparencia del Estado.

- Ya que cualquiera puede definir formatos, el software libre no provee formatos uniformes y universales como los provistos por Word y Excel (.doc, .xls)

  Nada más variable que los documentos de Microsoft Word y Excel que ni siquiera interoperan con las distintas versiones de su propio software.

  El software libre suele respetar los estándares, y crea los adecuados para cada tarea, asegurando la existencia de numeroso software que puede usarlos. No crea lupas klingonianas [027] especiales para favorecer su monopolio. Los estándares respetados suelen fijarse en comités altamente formalizados como la IETF, ICANN o W3C.

Seguridad

- ¿Existen riesgos en el uso de software libre desde el punto de vista de seguridad?

¿No es necesario ocultar el código en ciertas áreas? ¿La publicidad del código no facilita el acceso indebido a los criminales informáticos?

No hay sistema totalmente seguro. Existe evidencia científica de que, a iguales condiciones, el modelo de desarrollo del software libre favorece la creación de sistemas más seguros. Hay programas libres para usar los mecanismos de seguridad más fuertes conocidos.

Si la seguridad depende de que el atacante conozca o no el mecanismo se está colocando el riesgo en una cuestión ajena al protocolo de seguridad.

El hecho de que sean libres les da una garantía de calidad, ya que su publicidad permite que cualquiera pueda detectar y reparar los fallos y riesgos a la seguridad que contenga. Los expertos en seguridad cuestionan fuertemente el concepto de Seguridad mediante la oscuridad. Cuando se oculta el funcionamiento, sólo aquellos que tienen intenciones de vulnerar esta seguridad se toman el trabajo de desarmarlo y ver cómo funciona, aumentando el riesgo.

Respecto de la seguridad del software en sí, es bien sabido que el software (propietario o libre) contiene errores de programación o "bugs" (en la jerga informática) en sus líneas de código. Los bugs en el software libre se reparan mucho más rápidamente, que en el software propietario.

Corresponde recordar que, en numerosos casos, las condiciones de licenciamiento incluyen cláusulas de No-Divulgación que impiden a los usuarios revelar abiertamente las fallas de seguridad halladas en el software propietario.

Privacidad

- Que el código sea abierto impide el resguardo de los datos privados en forma segura.

  Nada más lejos de la realidad. Los mejores sistemas de criptografía han progresado en el marco del software libre. Hoy es mucho más fácil configurar discos privados seguros con software libre que con propietario. Con software cerrado nunca se sabrá hasta qué punto las agencias de seguridad han conseguido acuerdos con las empresas para tener puertas secretas en sus sistemas. Con software libre esto no es posible pues el código es transparente.

- ¿Es posible usar TCPA con software libre?

  TCPA es un mecanismo que permite asegurar que en una computadora sólo se ejecuta software autorizado por una central. Se basa en el concepto de elementos confiables, o sea los elementos que pueden violar la seguridad del equipo y que por lo tanto se debe confiar en ellos. Es altamente inconveniente un sistema de esta naturaleza pues quita al usuario el control de su computadora. En este contexto el usuario no es confiable y sí lo es la central que lo audita y que generalmente está en las multinacionales creadoras de software. También podría ser usado por gobiernos fascistas. Es posible implementar con software libre esta plataforma, pero no es recomendable su uso. Sugerimos

a los gobiernos prohibir el uso de computadoras con esta tecnología, no sólo en el Estado sino en toda la sociedad.

- Linux es un sistema operativo para hackers, de los hackers y por los hackers, no sirve para usuarios no técnicos.

  Un sistema operativo es un conjunto de herramientas más un núcleo, es lo que permite operar a una computadora y funcionar a sus aplicaciones. Linux es el núcleo del sistema, su código fuente ocupa unos 30 Mb. La mayoría de las herramientas de base para el funcionamiento del sistema fueron concebidas y desarrolladas por el proyecto GNU que tiene varios Gigas de código fuente. Por ello el sistema operativo debe denominarse GNU/Linux.

  En segundo lugar hay que desmitificar la palabra hacker. Los sectores de poder en Internet, y que intentan controlarla, impusieron un significado negativo a esta palabra usándola para designar a quienes violan sistemas o protecciones con el fin de robar o hacer daño. Para la gente que realiza estos actos se utiliza la palabra cracker.

  Hacker se usa para designar a una persona curiosa que con habilidad y talento investiga, crea nuevas herramientas y resuelve problemas. Es un gran mérito que a una persona su comunidad lo considere como hacker o mago.

  No hay que caer entonces en la guerra comunicacional de las corporaciones que intenta asimilar hacker con cracker, y al acto solidario de compartir software con la acción nefasta de asaltar barcos denominada piratería.

  GNU/Linux efectivamente fue concebido por hackers para hackers, pero ha evolucionado y cada día es más fácil de utilizar e instalar, perfectamente puede ser utilizado hoy en día por niños y por personas sin conocimientos informáticos especiales.

Disponibilidad futura y persistencia de la información

- GNU/Linux es un sistema que depende de los hackers para su mantenimiento y evolución. Es un riesgo grande apostar por algo no profesional y sin respaldo empresarial.

  Los hackers crearon Internet a partir de una especificación militar que permitía a las redes sobrevivir aún a ataques nucleares, y mediante ella crearon GNU/Linux, y mucho otro software desde el correo electrónico, navegadores, servidores, etc. Prácticamente toda Internet funciona sobre código hacker. Las empresas comerciales como Microsoft, IBM, etc. intentaron crear sus propias redes, las que finalmente fueron absorbidas por Internet. Entraron tarde a Internet y se tuvieron que adaptar a su filosofía hacker universitaria.

  Las empresas de comunicaciones perdieron la guerra por imponer redes facturables en base a tiempo y distancia (como las telefónicas) y tuvieron que comercializar enlaces para Internet.

Hoy la red ha crecido y su código libertario da cabida a muchos actores que conviven en su ámbito. Así usuarios finales encuentran su lugar, junto a empresas comerciales, estados y corporaciones. Todo ello bajo el espíritu hacker, resguardado por la técnica de base de la red, que facilita el anonimato, la libertad, y la descentralización. Este es el sustento de la Sociedad del Conocimiento y la garantía de que sea una sociedad democrática. En el mundo virtual la ley se expresa mediante el software, que una vez liberado se ejecuta independientemente de la voluntad humana. Los actos en la red son regulados por este código que define y crea el espacio virtual en cuestión. Internet es una de varias redes posibles, definida por su código de base y especificaciones técnicas.

Internet y el software libre que le dio forma son parte de la más grande aventura comunitaria emprendida por la humanidad, miles de personas cooperando para construir conocimiento. Es en definitiva un modelo que derrotó a los modelos comerciales cerrados de creación de conocimiento y que tiene más empuje y vitalidad que estos.

El que haya utilizado Internet ya es usuario de software libre. La mayor parte de la infraestructura de Internet se basa en protocolos abiertos. Más del 60% de servidores web emplean Apache, otro gran número usan SendMail para el envío de correo electrónico y prácticamente la totalidad de los servidores de nombres (DNS), esenciales en el funcionamiento de la Red, utilizan el programa BIND o derivados de su código fuente.

La construcción del edificio del software libre avanza a más velocidad que la del comercial, pues ladrillo que se pone queda, no es necesario reinventar nada y porque los recursos combinados de inteligencia hacker en el mundo no pueden ser superados por ninguna multinacional. Hoy ya se llegó a construir mejores sistemas operativos y aplicaciones de oficina, se está llegando al nivel de las bases de datos comerciales y se sigue avanzando.

Por lo tanto es razonable apostar por un movimiento que viene creciendo y superando a los otros. Los riesgos en este caso son que las corporaciones, particularmente las de edición musical, en alianza con los gobiernos logren alterar mediante leyes como la DMCA, patentes de software y TCPA el espacio libre de la red universal. Cosa que parece difícil que puedan conseguir y cuyo efecto sería solamente hacer más lento un proceso inevitable.

Migraciones

- ¿No es suficiente con usar software libre sólo en los nuevos sistemas?

  Se debe usar software libre en todos los sistemas, nuevos y viejos; ya que los sistemas anteriores seguirían siendo perjudiciales, poniendo en riesgo la seguridad, generando dependencia tecnológica, e impidiendo ejercer control sobre la propia información.

  El software propietario actúa como una drogadicción, es costoso migrar y reeducar a los usuarios, pero cuanto antes se haga mejor.

- ¿No sería costosa una migración si se tiene algo que ya funciona?

  Sí, lo sería. Una migración involucra costos en relevamientos, toma de decisiones para implementar los nuevos sistemas, mano de obra para implementar el cambio, conversión de datos, reentrenamiento del personal, desarrollo y tiempo.

  Todos estos son inversiones fijas, que se pagan una vez. El software propietario también tiene sus costos fijos.

  Pero además de éstos, hay otros costos inherentes en el software propietario: actualizaciones permanentes (acentuadas por un efecto de monopolio autosostenido), pérdida de interoperabilidad, mantenimiento (con las desventajas obvias de ser un cliente cautivo de un contratista con monopolio sobre el mantenimiento, y capaz de cobrar lo que quiera) y por sobre todo, el inmenso costo que tiene para el Estado la pérdida de las libertades que le garantizan el control de su propia información.

  Estos costos son permanentes y crecientes a lo largo del tiempo (incluso si sólo se

  consideran los monetarios), y tarde o temprano, superarán a los costos fijos de realizar una migración. Por lo tanto, dado que la migración, a la larga, nos beneficiará económicamente, conviene llevarla a cabo lo antes posible, en vez de esperar que los costos crezcan hasta volverse incontrolables.

  Es un costo a corto plazo, pero un ahorro enorme a largo plazo. Y que además produce beneficios mayores.

- ¿El costo no sería demasiado alto dada la enorme cantidad de código desarrollado internamente por el Estado?

  Es cierto que mucho código (sobre todo administrativo) está desarrollado internamente, y los problemas de reemplazarlo son complicados, ya que no tienen reemplazo libre por ser algo muy específico y son muy costosos para re-desarrollar.

  Pero dado que el código desarrollado internamente es propiedad del Estado, este puede tramitar su relicenciamiento para declararlo software libre. De esta forma, todo este código se transformaría en programas libres sin esfuerzo técnico, sin necesidad de actualizar sistemas, ni volver a capacitar al personal.

  Existen países (España) que tienen normas que declaran libre a todo software que se usa en el Estado.

  En muchos casos el software realizado por el Estado, no responde a un plan central sino que cada usuario o dependencia lo creó en forma artesanal y sin interoperar con sistemas centrales. En dichos casos puede ser conveniente efectuar un reemplazo.

Cuanto más tiempo se demore la migración a otra tecnología, ésta se tornará más onerosa y al mismo tiempo se irán incrementando los riesgos de seguridad asociados con el software propietario. De esta manera, el uso de sistemas y formatos propietarios va haciendo que el Estado se vuelva cada vez más dependiente de proveedores determinados. Por el contrario, una vez implantada la política de uso de software libre (implantación que, es cierto, implica un costo), la migración de un sistema a otro se hace muy sencilla, ya que todos los datos están almacenados en formatos abiertos. Por otra parte, la migración a un entorno de software abierto no implica más costos que la misma entre entornos distintos de software propietario. Costos que habrá que pagar regularmente cada vez que se deban licitar los sistemas. El software libre interopera entre sí, por lo que cambiar de proveedor no representa un costo.

Soporte Técnico

- ¿De dónde obtendría soporte técnico y mantenimiento? El software de código abierto en su mayoría no ofrece los niveles de servicio adecuados ni la garantía de fabricantes reconocidos para lograr mayor productividad por parte de los usuarios.

  En general todo usuario sabe que es muy difícil obtener respaldo técnico adecuado de las corporaciones. Las mesas de ayuda rara vez funcionan adecuadamente y cuando lo hacen son servicios muy caros. Cuando hay un problema real raramente sirve su ayuda.

  En el mundo del software libre la comunidad ayuda. Es más fácil conseguir la respuesta a un problema simple a través de las listas de los grupos de usuarios. Y si el problema es complejo, siempre está disponible el código fuente y se puede contactar a sus autores que habitualmente suelen responder. La cultura de soporte de la comunidad libre es mucho más eficaz que la del propietario.

  Por otro lado nada impide crear una estructura comercial de soporte para el software libre. Los mismos técnicos recapacitados, podrían seguir cumpliendo sus funciones. El software libre tiene también soporte empresarial, al igual que el propietario. Algunas empresas como IBM dan soporte a software libre y propietario, otras como Red Hat, dan soporte a software libre solamente, y otras desarrollan software libre a pedido.

  Con software libre se puede elegir a quién contratar para soporte, en función de cuán capacitado esté, y cuánto quiera cobrar. De esta forma se impide la extorsión que puede realizar la empresa dueña de un software propietario, aprovechando su exclusividad sobre el soporte y mantenimiento de su software, que le otorga un monopolio.

  Entre las opciones para elegir, en el caso del software libre, se incluirían también como posibilidades a los técnicos y empresas locales, de esta forma fomentando el desarrollo y la economía local. Otro medio disponible son los acuerdos con las universidades, que son fuente de personal capacitado y que pueden colaborar para ofrecer soluciones y desarrollo de sistemas.

Es posible usar software libre sin servicios de soporte (así como sucede también con el software propietario) pero quienes los requieran pueden conseguir soporte y educación por separado, tanto de empresas locales cuanto de corporaciones internacionales, también como en el caso de software propietario.

Imposiciones normativas en el Estado

- ¿Establecer la obligatoriedad de usar exclusivamente software libre en el Estado, es transgredir los principios de la igualdad ante la ley, el de no discriminación y los derechos a la libre iniciativa privada, libertad de industria y contratación protegidos en la constitución?

  Es un principio bien establecido que el Estado no tiene el amplio espectro de libertad

  contractual del sector privado, pues precisamente está limitado en su accionar por el deber de transparencia de los actos públicos; y en ese sentido, la preservación del mejor interés común debe prevalecer cuando se legisla sobre la materia.

  La igualdad ante la Ley no se altera, pues ninguna persona natural o jurídica está excluida del derecho de ofrecer estos bienes al Estado en las condiciones fijadas, sin más limitaciones que las establecidas en las leyes que, sobre contrataciones y adquisiciones que el Estado posea.

  Debe tenerse en cuenta que una política de este tipo no discrimina en contra de software o proveedores específicos, sino contra ciertas prácticas nocivas que involucran el control de la información del usuario por parte del proveedor. Es fundamental que el Estado no se someta a estas presiones. Eso es lo que motiva las restricciones de esta política, que tienen como fin establecer cualidades mínimas para garantizar los derechos de los ciudadanos, la calidad del software y la seguridad de la información. Todas las empresas que acepten proveer su software sin comprometer estos derechos fundamentales no tendrán problema alguno en llegar a ser proveedoras del Estado.

  No hay discriminación alguna, pues sólo se establece cómo han de proveerse estos bienes (lo cual es una potestad estatal) y no quién ha de proveerlos (lo que en efecto resultaría discriminatorio si se impusieran restricciones basadas en origen nacional, raza, religión, ideología, preferencia sexual, etc.) Es más, las condiciones de provisión del software propuestas, impiden a los organismos estatales el uso de programas cuyo licenciamiento incluya condiciones discriminatorias.

  Establecer condiciones para el empleo del software por parte de las instituciones estatales no atenta contra la libre iniciativa privada, pues las empresas pueden elegir siempre bajo qué condiciones distribuir su software, sólo que algunas de estas condiciones serán aceptables para el Estado y otras no.

  Esta libre iniciativa es desde luego, compatible con la libertad de industria y con la libertad de contratación (en los términos acotados en que el Estado puede ejercer esta última). Cualquier sujeto

privado puede desarrollar software en las condiciones que el Estado lo requiere, o puede abstenerse de hacerlo.

Incluso Microsoft podría ofrecer a los organismos del Estado su "suite" de oficina, en las condiciones pedidas y fijando el precio que considere conveniente. Si no lo hiciera, no se debería a restricciones impuestas, sino a decisiones empresariales respecto al modo de comercializar su software, decisiones, en las que el Estado no tiene participación.

De hecho es práctica habitual en muchos gobiernos realizar licitaciones públicas especificando marcas en sus compras, lo que es incorrecto y muchas veces ilegal.

Si se pusiese en las licitaciones solamente las características de lo que se requiere no habría chance para el software propietario. Tampoco habría necesidad de especificar por ejemplo software antivirus. Pues se supone que los sistemas operativos debieran funcionar correctamente y sin posibilidad de estos errores (bugs).

Por otro lado la técnica de comercialización por derrame de copias ilegales y luego realizar acuerdos de legalización bajo las presiones de la BSA o Software Legal, han sido exitosas pero ilegales. Los funcionarios públicos deben violar la ley para adquirir software por acuerdos globales con una sola firma a un precio negociado a puertas cerradas. Sean acuerdos del tipo campus o de licencias por cantidad de personas o sean acuerdos de legalización para el pago de multas, son ilegales y pueden ser perseguidos por los fiscales de cada país.

- ¿Hacer obligatorio el uso de software libre, establecería un tratamiento discriminatorio y no competitivo en la contratación y adquisición de los organismos públicos?

  No se excluye a nadie ``a priori", sino en base a una serie de principios decididos por la voluntad autónoma del comprador, en tanto el proceso se lleve a cabo conforme a la ley. Nadie está excluido de competir en tanto garantice el cumplimiento de los principios básicos.

  Se estimula la competencia, pues alienta a generar oferta de software con mejores condiciones de usabilidad, y a optimizar trabajos ya establecidos, en un modelo de mejora constante.

  De otro lado, el aspecto central de la competitividad es la oportunidad de proporcionar al consumidor mejores opciones. Ahora bien, es imposible desconocer que el marketing no juega un papel neutral a la hora de presentar la oferta al mercado (pues admitir lo contrario habilitaría a suponer que las inversiones que las empresas realizan en marketing carecen de sentido), y por consiguiente un gasto significativo en este rubro puede influir las decisiones del comprador. El software libre no se difunde por marketing sino por militancia comunitaria, si bien algunas empresas últimamente han realizado publicidad con GNU/Linux.

  La elección no debe ser influida por el esfuerzo de comercialización; en este sentido, la competitividad se acentúa, pues el más pequeño creador de software puede competir en un pie de igualdad con la más poderosa de las corporaciones.

Es necesario recalcar que no hay posición más anti-competitiva que la de los grandes

distribuidores de software propietario, que frecuentemente abusan de su posición dominante, porque en innumerables casos proponen como soluciones a problemas planteados por los usuarios: "actualice su software a la nueva versión" (con cargo para el usuario, por supuesto); además, son comunes las interrupciones arbitrarias de asistencia técnica para programas que al sólo juicio del proveedor, son "antiguos" u obsoletos; luego para recibir algún grado de asistencia técnica, el usuario se ve obligado a migrar (con costo no trivial, especialmente porque suele involucrar cambios de la plataforma de hardware) a nuevas versiones. Y como toda la infraestructura está consolidada en formatos de datos propietarios, el usuario queda "atrapado" en la necesidad de continuar empleando los sistemas del mismo proveedor, o realizar el enorme esfuerzo de cambiar a otro ambiente (también probablemente propietario).

- ¿Forzar una política de uso de software libre no es equivalente a forzar el uso de un producto comercial determinado?

  El software libre no es comparable a una marca determinada, ya que para que un programa sea libre basta con que se otorguen las facultades apropiadas al usuario, condición que cualquier empresa nacional o extranjera puede cumplir. Esto es diferente a decir "se exige marca X", que es una condición que solo la empresa X puede cumplir. Además no hay imposición sobre la libertad de decisión de los ciudadanos, ya que una política de uso exclusivo de software libre es una decisión del Estado y para el Estado, es decir, de administración de sus sistemas internos. Es lo que le corresponde a un gobierno: organizarse internamente de la mejor forma posible para defender los derechos de los ciudadanos y proteger su propia seguridad.

  Incluso ni siquiera es necesario especificar una marca como Linux, ya que hay varios sistemas operativos libres como BSD.

- Si el software libre satisface todos los requerimientos de las entidades del Estado ¿por qué se requiere de una Ley para adoptarlo? ¿No debería ser el mercado el que decida libremente cuáles son los sistemas que le dan más beneficios o valor?

  En el sector privado de la economía suele ser el mercado quien decide qué se usa. Pero en el caso del sector público, el razonamiento no es el mismo: como ya establecimos el Estado almacena, manipula y transforma información que no le pertenece, sino que le ha sido confiada por los ciudadanos que, por imperio de la ley, no tienen más alternativa que hacerlo. Como contraparte a esa imposición legal, el Estado debe extremar las medidas para salvaguardar la integridad, confidencialidad y accesibilidad de esas informaciones. El empleo de software propietario arroja serias dudas sobre el cumplimiento de estos atributos, a falta de evidencia concluyente al respecto y por lo tanto no es apto para ser usado en el sector público.

  La necesidad de una ley estriba, por un lado, en la materialización de los principios fundamentales antes enunciados en el campo específico del software. Por otro, en el hecho de que el Estado no es

una entidad ideal homogénea, sino que está compuesto de múltiples organismos con diversos grados de autonomía de decisiones. Dado que el software propietario es inapropiado para ser empleado, el hecho de establecer estas reglas en la ley impediría que la decisión discrecional de cualquier funcionario ponga en riesgo la información que pertenece a los ciudadanos. Y, sobre todo, porque constituye una reafirmación actualizada en relación con los medios de tratamiento y comunicación de información empleados hoy en día, sobre el principio republicano de publicidad.

Conforme a este principio universalmente aceptado, el ciudadano tiene derecho a conocer toda información en poder del Estado que no esté amparada en una declaración fundada de secreto conforme a la ley. Ahora bien, el software trata información y es en sí mismo información. Información en formato especial, susceptible de ser interpretada por una máquina para ejecutar acciones, pero sin duda información crucial porque el ciudadano tiene legítimo derecho a saber, por ejemplo, cómo se computa su voto o se calculan sus impuestos. Y para ello, debe poder acceder libremente al código fuente y probar a su satisfacción los programas que se utilizan para el cómputo electoral o para el cálculo de sus impuestos.

- ¿No es conveniente dejar que se seleccione el mejor software, sea libre o propietario?

Obviamente, al Estado le conviene elegir la solución más apta para sus necesidades. La clave del problema está en el significado de "más apta". No es necesariamente la solución más usada en el mercado; muchas veces la solución más exitosa lo es solamente porque la empresa que la promovió tuvo una mejor campaña publicitaria, porque atrapó al mercado en un monopolio, porque hizo una buena estrategia comercial, etc.

Tampoco la que tiene toda la funcionalidad necesaria; el Estado necesita más que eso: necesita ser independiente tecnológicamente, poder tener control sobre su propia información y poder proteger la seguridad de sus datos. Esas son algunas de las aptitudes que sólo el software libre puede otorgar, aptitudes que el Estado no puede dejar de lado (mientras sí puede dejar de lado, en cambio, funcionalidad no crítica).

Por ello, la solución más apta, sea cual fuere, es un programa libre; los programas no libres tienen características que los hacen completamente inutilizables para un Estado, por someterlo a riesgos importantísimos, aún cuando la solución libre que se otorgue, sea ligeramente menos funcional (mientras no se trate de funcionalidad crítica), o más costosa.

La cuestión de quién elige en el Estado el software es estratégica. Lo hace ¿el Congreso, el Presidente, los Ministros, los Secretarios, en cada oficina, cada usuario?

Cada nivel de responsabilidad tiene su parte. El Congreso deberá fijar las pautas de seguridad, transparencia y durabilidad. El Presidente, a través del área que gestiona las TIC (Tecnologías de la Información y Comunicación.), establecer mecanismos que aseguren un adecuado, eficaz y eficiente cumplimiento de lo anterior. Cada Ministerio, ente descentralizado y de allí para abajo deberá ejecutar estas políticas de acuerdo a los mecanismos decididos.

Economía del software

- ¿No se perjudicaría a la industria de software?

  No, porque la política no tiene nada que ver con poner trabas o prohibiciones a la industria del software, sino fijando como condición necesaria para uso de software en el Estado la característica de "libre" (De la misma manera en que suelen establecerse condiciones necesarias razonables a cumplir en cualquier licitación o contratación que hace el estado).

  De hecho, si en "industria de software" consideramos a la industria local, esta será ampliamente beneficiada, ya que el gobierno puede contratar profesionales locales para modificar y adaptar sus sistemas (incluso aquellos no desarrollados localmente), y de esta forma fomentar la industria tecnológica local, la economía y el empleo.

  Estos beneficios que distinguen al software libre del propietario, provienen de la posibilidad de inspección y modificación libres a cualquier individuo, en vez de estarrestringidos al proveedor, que puede usar esa restricción para monopolizar el soporte técnico

- ¿Al favorecer un modelo de negocios que apoyaría exclusivamente el software libre, se estaría desalentando a las compañías desarrolladoras locales e internacionales? Estas son las que verdaderamente realizan importantes inversiones, crean un significativo número de puestos de empleos directos e indirectos, además de contribuir al PBI vs. un modelo que tiende a tener un impacto económico cada vez menor debido a que crea principalmente empleos en servicios.

  El modelo de servicios, adoptado por gran número de corporaciones en la industria

  informática, es mucho más significativo, en términos económicos y con tendencia creciente, que el licenciamiento de programas.

  Por otra parte, el sector privado de la economía tiene la más amplia libertad para elegir el modelo económico que más convenga a sus intereses, aunque esta libertad de elección quede muchas veces oscurecida de manera subliminal por las desproporcionadas inversiones en marketing de los creadores de software propietario.

  Si el mercado Estatal es crucial e imprescindible para la industria del software

  propietario, a tal punto que la decisión de Estado eliminaría completamente de él a estas empresas, deducimos que el Estado estaría subsidiando a la industria del software propietario.

  En tal caso el Estado tendría el derecho en aplicar los subsidios al área que considere de mayor valor social; resultaría innegable que si el Estado decide subsidiar software debería hacerlo escogiendo el libre por encima del propietario, atendiendo a su efecto social y al uso racional de los dineros de los contribuyentes. Por otro lado esta subvención no violaría los acuerdos TRIPS de la OMC ya que se

crearía software utilizable libremente en cualquier lugar del planeta. Y que no afecta la habilidad competitiva de nadie.

Respecto de los puestos de trabajo generados por el software propietario en países del tercer mundo, estos tratan mayoritariamente tareas técnicas de poco valor agregado; a nivel local, los técnicos que prestan soporte a software propietario distribuido por empresas transnacionales no están en condiciones de solucionar un bug, no necesariamente por falta de capacidad técnica o talento, sino porque no disponen del código fuente a reparar. Con software libre se crea empleo técnicamente más calificado y se genera un marco de libre competencia donde el éxito está sólo vinculado a la capacidad de brindar buen soporte técnico y calidad de servicio, se estimula el mercado y se incrementa el conocimiento común, abriendo alternativas para generar servicios de mayor valor agregado y mejor perfil de calidad beneficiando a todos los actores: creadores, prestadores de servicios y usuarios.

Es un fenómeno común en los países en vías de desarrollo que las industrias locales de software obtienen la mayoría de sus ingresos en el área de servicios, o en la construcción de software "ad hoc". Por lo tanto, cualquier impacto negativo que pueda tener en este sector se verá compensado con creces por un aumento de la demanda de servicios (a condición de que estos sean prestados conforme a altos estándares de calidad). Desde luego, es probable que las empresas transnacionales de software si deciden no competir conforme a estas reglas de juego, sufran alguna disminución de ingresos en términos de facturación por licenciamiento; pero considerando, que estas empresas alegan sostenidamente que mucho del software empleado por el Estado fueron copiados ilegalmente, se verá que el impacto no ha de ser extremadamente serio. Ciertamente, en todo caso su fortuna estará determinada por leyes del mercado, cuyos cambios no es posible evitar; muchas empresas tradicionalmente asociadas con el software propietario ya han emprendido un camino firme (apoyado por cuantiosas inversiones) para prestar servicios asociados con el software libre, lo cual demuestra que los modelos no son mutuamente excluyentes.

Con una política que especifica requisitos de libertad en el software el Estado está decidiendo que requiere preservar ciertos valores fundamentales. Y lo decide en base a sus potestades soberanas, sin afectar con ello ninguna de las garantías constitucionales. Si estos valores pueden ser garantizados sin tener que escoger un modelo económico dado, los efectos de la ley serían aun más beneficiosos. En todo caso debe quedar claro que el Estado no elige un modelo económico; si sucediera que existe un sólo modelo económico capaz de proveer software tal que satisfaga la garantías básicas de estos principios, se trataría de una circunstancia histórica y no de una decisión arbitraria en favor de un modelo dado.

- El uso de Software Libre desincentiva la creatividad de la industria local de software que es una fuente de empleo altamente calificado.

Está claro por demás que nadie está obligado a comercializar su código como software libre. Tan sólo deberá tener en cuenta que, si no lo hace, no podrá venderle al sector público. Este, por otra

parte, no constituye el principal mercado para la industria nacional de software. Ya hemos abordado algunas cuestiones referidas a la influencia del software libre en la generación de empleo técnico altamente calificado y en mejores condiciones de competitividad.

- La licencia GPL destruye valor ya que impide incorporar este software al que se puede producir y vender comercialmente.

  La licencia GPL, efectivamente incorpora el concepto de copyleft que impide su apropiación por el software propietario. Esto es bueno porque empresas comerciales no pueden tomar el software libre y desvirtuarlo.

  El software no se produce sino que se crea. Tampoco se suele vender (esto implica la transferencia del copyright). Muchas veces los autores trabajan a sueldo de una empresa entonces automáticamente el copyright de sus creaciones es de su empleador.

  El software puede ser compartido (libre) o licenciado. De hecho el software libre también se licencia con la anti-licencia GPL (copyleft) que usa la infraestructura legal del copyright (copyrestrictions) para eliminar las restricciones de copia.

  El valor económico del software viene de la escasez artificial creada por las leyes de restricciones de copia y eventualmente las patentes (mecanismo altamente inapropiado para el software) y es reforzado por el poder de policía del estado. Por lo tanto no es una cuestión normal que algo que no es naturalmente escaso tenga valor económico.

  Así las restricciones de copia son una externalidad a la economía, algo que impone un costo adicional y una restricción al libre flujo de ideas en los mercados y comunidades. Creada como medida para facilitar la difusión de la cultura impresa, en la sociedad de la información es anacrónica y una rémora del pasado.

- El software libre es comunista.

  Frase aplicada a todo lo que se detesta por parte de ciertos sectores. Así como suelen también odiar a los gitanos, drogadictos, homosexuales o a los judíos. El software libre puede ser reivindicado por prácticamente todas las ideologías. Se lo mire por donde se lo mire es positivo para los ecologistas, los comunistas, los socialistas, los liberales, los conservadores, los religiosos, etc.

  La oposición al software libre no es una cosa de ideales sino de intereses. Los únicos interesados en el software propietario son sus propietarios. Cada vez menos y más concentrados. El software propietario es un desvalor para todos salvo para sus dueños.

  Curiosamente el modelo del software propietario cada vez más se basa en el poder de policía del estado que en un acuerdo de negocios. Así mediante las presiones de sus ONG: BSA y Software Legal exigen el cobro de un impuesto universal de unos U$S 500 por PC para sistemas operativos, antivirus y paquetes oficina, que van a lo sumo a un par de multinacionales.

Estas presiones muchas veces se ejercen mediante prácticas propias de los regímenes totalitarios como la delación entre empleados y amigos.

El software libre propone un modelo económico basado en los servicios y la economía solidaria.

- El software de código abierto, al poder ser distribuido gratuitamente, no permite generar ingresos para sus desarrolladores por medio de la exportación. Promoviendo el software libre, se debilita el efecto multiplicador de la venta de software a otros países y por lo tanto el crecimiento de esta industria, cuando las normas de un Gobierno deben estimular la industria local.

  Esta afirmación supone que el mercado de cesión de derechos no exclusivos de uso a título oneroso (venta de licencias) es el único posible para la industria informática cuando ni siquiera es el más importante. El surgimiento de una oferta de profesionales más calificados, en conjunto con el incremento de experiencia que significa para los técnicos locales al trabajar a gran escala con software libre en el Estado, los colocan en una posición altamente competitiva para brindar sus servicios al extranjero.

- Es habitual que Microsoft realice donaciones de Software sobre todo a las escuelas.

  Esto ocurre sobre todo cuando están pensando en adoptar software libre. Como regla si necesita software de Microsoft anuncie que usará software libre.

  Si las ideas fuesen bienes económicos el valor total del software libre producido en el planeta sería monstruoso. Por lo tanto la donación que hace el movimiento del software libre a toda la humanidad tiene mucho más valor que la de Microsoft o cualquier otra empresa.

Resistencia al cambio

- La gente se resiste al cambio especialmente en las organizaciones burocráticas como el Estado.

  Esto es cierto, pero tengamos en cuenta que nada cambia tanto como la informática.

  Cada cinco años hay cambios importantes como el del DOS al Windows, o el del Windows al GNU/Linux. La flexibilidad y adaptabilidad de las administraciones es algo necesario si se quiere tecnificar la administración.

Escalabilidad

- El software libre no opera en equipos de gran capacidad para empresas ni es escalable.

  En realidad es todo lo contrario, gracias a la persistencia del proyecto GNU y a su ubicuo compilador gcc, es posible ejecutar software libre desde relojes hasta grandes mainframe. De hecho

las grandes computadoras científicas se están construyendo con clusters bajo GNU/Linux. Raramente el software propietario logra estos extremos. La distribución como código fuente permite recompilar con facilidad el software para cada plataforma.

Garantías

- ¿Existen riesgos en el uso de software libre desde el punto de vista de garantías?

  Leyendo el "End User License Agreement" del software de Microsoft, en la amplísima mayoría de los casos, las garantías están limitadas a la reposición del medio de almacenamiento si este fuera defectuoso, pero en ningún caso se prevén compensaciones por daños directos o indirectos, lucro cesante, etc.

  Si como consecuencia de un error (bug) de seguridad, no oportunamente reparado por una empresa proveedora de software propietario, un atacante comprometiera sistemas cruciales para el Estado: ¿qué garantías, reparaciones y compensaciones proporcionaría la empresa de acuerdo con sus condiciones de licenciamiento? Las garantías del software propietario, en tanto los programas se entregan ``AS IS", es decir, en el estado en que se encuentran, sin ninguna responsabilidad adicional para el proveedor respecto a su funcionalidad, no difieren en modo alguno de las habituales en el software libre.

  Las leyes de protección al consumidor suelen fijar condiciones especiales de garantías y requisitos adicionales en caso de adquisición onerosa. En cuyo caso tanto la empresa proveedora de software propietario como la de libre debieran cubrir.

  En ambos casos estas garantías tienen un costo. Independiente del costo de licenciamiento. Por lo que no hay diferencias entre el software propietario y el libre en este sentido.

Derechos de Autor

- ¿Existen riesgos en el uso de software libre desde el punto de vista de posibles violaciones de la ``propiedad intelectual" de terceros?

  En primer lugar las ideas no son apropiables, por lo que hablar de propiedad intelectual es un intento publicitario/ideológico de quienes quieren asimilar las ideas con las cosas y convertirlas en propiedades.

  Existen los derechos de autor, las patentes, las marcas y otros mecanismos que otorgan derechos, licencias y concesiones a los autores, inventores y otros.

El modelo de software libre no implica en modo alguno desconocer estas leyes y de hecho, la amplísima mayoría del software libre está amparado por el copyright.

La incorporación de otras obras ajenas en obras que luego se atribuyen como propias es muy fácil de descubrir en el mundo del software libre, pues su código está abierto a inspección, casi imposible en el propietario.

Sin embargo hubo casos de copia en el mundo propietario, valga a título de ejemplo la condena de la Corte Comercial de Nanterre, Francia, del pasado 27 de septiembre de 2001 a Microsoft Corp., por 3 millones de francos en concepto de daños e intereses, por violación de la propiedad intelectual (piratería, según el desafortunado término que ésta empresa suele usar en su publicidad).

Recientemente SCO acusó a IBM de incorporar código suyo a algunas funciones especiales del núcleo Linux 2.4. Esta acusación está fundamentalmente relacionada con contratos de confidencialidad firmados entre ambas empresas.

Sin embargo SCO mismo licenció tal código como GPL, en su distribución Caldera.

También hubo un importante litigio entre BSD y los propietarios originales del Unix, que retrasó años la difusión del sistema operativo BSD.

En un mundo con leyes de restricción de copia nadie puede estar seguro, pero sin duda se puede dormir más tranquilo con software libre.

- Los programadores de software pierden sus derechos de autor y su principal fuente de retribución.

  Ningún autor de software libre pierde sus derechos de autor, a menos que por su expresa voluntad desee colocar su obra en el dominio público o los transfiera. El movimiento del software libre siempre ha sido extremadamente respetuoso de sus autores, y les ha generado reconocimiento público extenso. Nombres como el de Richard Stallman, Linus Torvalds, Guido van Rossum, Larry Wall, Miguel de Icaza, Andrew Tridgell, Theo de Raadt, Andrea Arcangeli, Bruce Perens, Darren Reed, Alan Cox, Eric Raymond, y muchos otros, son mundialmente reconocidos por sus contribuciones en el desarrollo de software que hoy es utilizado por millones de personas en todo el mundo, en tanto que los nombres de los autores materiales de excelentes piezas de software propietario, permanecen en el anonimato.

  Por otra parte, afirmar que las regalías por derechos de autor constituyen la principal fuente de retribución de los programadores es en todo caso aventurado, en particular porque no se ha aportado ninguna prueba al efecto ni una demostración de cómo el empleo de software libre por el Estado influiría en estas retribuciones.

- ¿No es perjudicial el software libre para sus autores, quitándoles el incentivo a desarrollar?

  Para contestar esta pregunta, debe aclararse primero que el software difiere de otras creaciones intelectuales (tales como los libros o las obras de arte) en varios aspectos.

Así el software propietario se distribuye sin los códigos fuentes, por lo que no merecería protección de derecho de autor sino de secreto comercial.

El software es más parecido al conocimiento científico, y el desarrollo de software a la investigación. En el mundo científico, las libertades de uso de conocimiento previo, y la libre circulación del conocimiento, son valores importantes, apreciados, y que originan el progreso. Cuando Sir Isaac Newton dijo "Si he visto más lejos, es porque me he subido sobre los hombros de gigantes", estaba valorando estas libertades, otorgadas por los "gigantes", que le permitieron llevar a cabo su desarrollo. De la misma forma, en el mundo del software libre, está disponible el trabajo de "gigantes", como un capital inicial sobre el cual desarrollar y realizar verdadera innovación, en vez de reinventar la rueda una y otra vez. Al igual que los científicos, los desarrolladores de software libre tienen como uno de sus incentivos el prestigio de la creación intelectual, que se logra mediante la publicación del trabajo. De ahí que la libertad de inspección y uso provee un marco para promover el desarrollo y la innovación software, y la libertad de distribución promueve un incentivo.

Además del incentivo personal, existe un incentivo económico. El software, como herramienta, requiere mantenimiento (deployment, adaptación, modificaciones durante su uso, reparaciones de errores). Para realizar adecuadamente ese mantenimiento es necesario saber la forma en la cual opera el software. Cuando el software no es libre, ese conocimiento es exclusivo del autor, y por lo tanto éste tiene un monopolio sobre estos servicios asociados.

Las libertades que otorga el software libre permiten la ruptura de este monopolio, promoviendo la libre competencia, es decir un mercado donde estos servicios pueden ser prestados por cualquier persona capacitada, y donde el precio es fijado por las necesidades reales, no por una decisión monopólica. De esta forma, las libertades en el software crean un incentivo de valor directo económico y puestos de trabajo que además pueden transferirse a la industria local si el software fue desarrollado en el extranjero. La libertad de distribución no actúa en perjuicio de este incentivo económico, ya que el software se desarrolla como un objeto abstracto con un alto valor intelectual, y la realización de copias tiene un costo casi nulo que no es lo que se paga al contratar el desarrollo de software.

- Las leyes como la DMCA son incompatibles con el software libre.

(Digital Millennium Copyright Act)

Estas leyes penalizan fuertemente conductas que siempre fueron aceptadas como válidas en las comunidades virtuales. Creemos que deben ser derogadas. El software libre no es incompatible con estas reglas, si bien la mayoría de los que proponemos el uso de software libre nos oponemos a leyes de este tipo.

Costos

- El software libre no está exento de costo y no es gratuito. Las conclusiones sobre ahorro para el Estado son apresuradas sin ningún estudio que sustente el costo - beneficio de tal política.

  La gratuidad y la libertad son conceptos ortogonales: hay software propietario y oneroso (por ejemplo, Microsoft Office), software propietario y gratuito (Microsoft Internet Explorer), software libre y oneroso (distribuciones RedHat, SuSE, etc. del sistema GNU/Linux), software libre y gratuito (Apache, OpenOffice, Mozilla), y aún software que se licencia bajo diferentes modalidades (MySQL).

  Ciertamente que el uso del software libre no es necesariamente gratuito y tiene diferentes costos.

  En ningún momento las definiciones de software libre se refieren a la gratuidad. Si bien se mencionan las posibilidades de ahorro en términos de lo pagado por licencias de software propietario, lo importante son las garantías fundamentales que se pretende preservar y el estímulo del desarrollo tecnológico local. Puesto que un Estado democrático debe sostener estos principios, no le queda otra solución que emplear software cuyo código fuente está públicamente disponible e intercambiar información sólo en formatos estándares.

  Si el Estado no empleara software con esas características, estaría vulnerando principios republicanos básicos. Por fortuna, además, el software libre implica menores costos totales; pero aún en la hipótesis (fácilmente negada) de que costara más que el propietario, la sola existencia de una herramienta de software libre eficaz para una determinada función informática obligaría al Estado a usarla; no por imperio de la ley, sino por los principios elementales que surgen de la esencia misma del Estado democrático de derecho. **El software propietario es incompatible con la Democracia**

- Se estima que el costo de adquisición del software (sistema operativo y aplicaciones) se reduce a sólo 8% del total de costos que las empresas e instituciones deben asumir como consecuencia del uso racional y realmente provechoso de la tecnología. El otro 92% lo constituyen: costos de implantación, capacitación, soporte, mantenimiento, administración e inoperatividad.

  Esto prueba la importancia de los servicios en la economía digital.

  Suponiendo que el costo de software sea sólo el 8% del costo total de utilización, no invalida en forma alguna la conveniencia de usar software gratuito, esto es, aquel cuyo costo de licenciamiento o uso es cero.

  El uso de software libre contribuye significativamente a disminuir los restantes costos del ciclo de vida. Esta reducción del impacto económico de despliegue, soporte, etc. se registra en varios campos; por un lado, el modelo competitivo de servicios del software libre, cuyo soporte y mantenimiento es posible contratar libremente entre una oferta variada que compite en función de la

calidad y el menor costo. Esto es válido para la implantación, la capacitación y el soporte, y en buena medida para el mantenimiento.

La característica reproductiva del modelo, hace que el mantenimiento que se realizó en una aplicación sea replicable muy fácilmente, sin incurrir en mayores costos (es decir, sin pagar más de una vez por lo mismo) pues las modificaciones, si así se desea, quedan incorporadas al patrimonio común del conocimiento.

El enorme costo causado por la inoperatividad ("pantallas azules de la muerte", código malicioso como virus, worms y troyanos, excepciones, fallas generales de protección y otros tantos males conocidos) se reduce significativamente al emplear software más estable; y es bien sabido que una de las virtudes más destacables del software libre es su estabilidad.

En todo caso lo que está en cuestión no es el costo del software, sino los principios de libertad de información, accesibilidad y seguridad.

- También existen estudios que indican que el software propietario es más económico a largo plazo que el libre, teniendo un TCO más bajo y un ROI más rápido.

  Esos estudios pagados por Microsoft se basan en el mayor valor de la mano de obra requerida para manejar software libre. Lo que sólo los hace válidos para economías desarrolladas con altos costos de mano de obra. Esto en realidad es positivo pues refleja que mayor dinero se vuelque a salarios, y eleva el nivel de los técnicos del software.

  Por otra parte no considera que un administrador de sistemas libres puede administrar hasta 4 veces más servidores que uno de sistemas Windows, pues los primeros se ``cuelgan" menos, y su administración remota es más simple y fácilmente configurable y replicable en simultáneo por línea de comandos.

  En resumen, el software libre tiene un menor costo total de operación (TCO), un retorno más rápido de la inversión (ROI) que el propietario, y contribuye mejor a la economía nacional, si bien estas no son sus principales fortalezas.

- Existen modalidades de licenciamiento por volumen que pueden ser sumamente ventajosas para el Estado.

  Ciertamente existen modalidades de licenciamiento por volumen. Ahora estas modalidades exigen comprar software a una empresa particular por decisión política y no mediante procesos de licitación. Exigen optar por esquemas por usuario y no por máquina y entrar en los contratos de adhesión de las corporaciones, nada más humillante para un Estado soberano.

  Sólo apuntan a reducir el impacto de un componente que importa no más del 8% del costo total.

- El Software libre pone en riesgo la compatibilidad y posibilidad de interoperabilidad de las plataformas informáticas dentro del Estado, y entre el Estado y el sector privado, dada la centena de versiones que existen de software libre en el mercado.

  Esta afirmación implica un cierto desconocimiento de los mecanismos de construcción de software libre, en el que no se maximiza la dependencia del usuario respecto de una plataforma determinada, como sucede habitualmente en el campo del software propietario. Aun cuando existen múltiples distribuciones de software libre, y numerosos programas susceptibles de ser empleados para una misma función, la interoperabilidad queda garantizada tanto por el empleo de formatos estándar, como por la posibilidad de construir software interoperable a partir de la disponibilidad del código fuente.

Fracasos

- ¿Por qué fracasó la iniciativa de usar Software Libre en la educación en un país como México?

  en donde precisamente los funcionarios del Estado que fundamentaron el proyecto, hoy expresan que el software libre no permitió brindar una experiencia de aprendizaje a alumnos en la escuela, no se contó con los niveles de capacitación a nivel nacional para dar soporte adecuado a la plataforma, y el software no contó y no cuenta con los niveles de integración para la plataforma que existen en las escuelas.

  Efectivamente, es posible fracasar con software libre. En México se dio marcha atrás con el proyecto Red Escolar. Eso se debió, precisamente a que los impulsores del proyecto mexicano tuvieron al costo de las licencias como principal argumento, en vez de las otras razones que son más importantes. Debido a este error conceptual, y como consecuencia de la falta de apoyo efectivo por parte de la SEP (Secretaría de Educación Pública) se asumió que para implementar software libre en las escuelas, bastaba con quitarle a éstas el presupuesto para software y en cambio enviarles un CD ROM con GNU/Linux. Por cierto, esto falló y no podía ser de otro modo, tal como fallan los laboratorios escolares en los que se usa software propietario si no hay presupuesto para implementación y mantenimiento.

  Los proyectos no se deben limitar a obligar a usar software libre, sino que deben reconocer la necesidad y ordenar la creación de un plan de migración viable, para ordenar una transición técnica para lograr disfrutar de las ventajas del software libre.

  También han fracasado empresas que pretendieron basar su negocio en los servicios de software libre, otras han triunfado. Aquí como en todo negocio, lo importante no es sólo la tecnología sino fundamentalmente el plan de negocios y la capacidad ejecutiva.

  Sería interesante informarse acerca de proyectos de software libre implantados en entidades públicas, que a la fecha hayan sido abandonados en favor del software propietario. Se conocen un

buen número de casos, creciente, en el sentido hacia el software libre, pero carecemos de información respecto a casos inversos.

Planeamiento estratégico participativo de migraciones a Software Libre

Subsecciones

Migrando organizaciones

Migrar es transformar una organización que utiliza software propietario para que adopte la cultura del Software Libre.

Se plantean ideas para la adopción de Software Libre en las organizaciones de las sociedades del conocimiento y sus e-X, enmarcadas en las tecnologías de auditoría y gestión, como COBIT, donde X, puede ser gobierno, salud, etc., según la actual moda.

Algunas referencias en migraciones.[6.1]

Nueve (9) mantras o recomendaciones para migrar estados y organizaciones.

Estos mantras están pensados como guías, puntos de control o patrones para elaborar o monitorear planes específicos[6.2].

Se puede pensar que el Software Libre es un ``problema" que se puede encarar y solucionar al igual que una migración de ``Microsoft Windows 95" a ``Microsoft Windows 2000". Esto no es así por muchos motivos. En particular los ejecutables que funcionan en ``Microsoft Windows" no funcionan directamente en ``GNU/Linux", pero la cuestión va mucho más allá. La migración representa un cambio en la cultura de trabajo de la organización. La cuestión pasa por compartir el conocimiento, con diferente sustento económico y con diferentes prácticas en las tecnologías de comunicación. Implica organizar de otra forma a los departamentos de IT.

Los mantras se pueden pensar como círculos concéntricos y visualizar la migración como una onda que se propaga desde la autoridad máxima, hasta toda la organización. Si bien aparecen como una secuencia, muchos mantras continuarán mientras se activan los subsecuentes.

Figura:
Espiral
de
la
migración:
mantras,
tareas
y
grupos

Tomar decisión política

Es fundamental que la autoridad máxima tome la decisión de migrar y la respalde con hechos ante cada circunstancia y esté dispuesta a contrarrestar las oposiciones, definir las disputas, impulsar el proceso y proveer recursos económicos.

La decisión formal debe:

1. definir la población y/o instituciones objeto;
2. dar instrucciones claras sobre el alcance de la medida;
3. liberar el software del cual se es derecho-habiente,
4. resolver sobre los proyectos previos que estén en proceso,
5. establecer la actitud ante la resistencia y
6. especificar los plazos.

Podría pensarse una etapa previa consistente en sensibilizar y convencer a la autoridad unitaria o colectiva que toma la decisión. También podría pensarse que la decisión sólo se tomará cuando toda o una buena parte de la organización esté sensibilizada, de acuerdo e incluso migrada. Este último sería un esquema de migración de abajo hacia arriba. Un esquema intermedio sería que la autoridad ``promueva" pero no ordene la migración. Uno más débil que solamente ``autorice" el uso de Software Libre.

Definir el responsable

El responsable debe:

1. ser o tener autoridad y mando sobre las funciones vinculadas a las TIC.
2. reportar directamente a la máxima autoridad del área objeto, por la importancia estratégica de su función. Y en las cuestiones TIC debe tener precedencia sobre sus pares que dirigen otras áreas.
3. contar con respaldo político y autoridad suficiente para cumplir su misión.
4. tener amplio presupuesto y facilidades para ejecutarlo sin trabas burocráticas, y poderes especiales cuando corresponda.
5. organizar acciones y definir cuestiones fundamentales para unificar criterios de desarrollo e implementación de políticas de TICs en todas las áreas objeto. En el caso del Estado sus Ministerios, Poderes y estructuras.
6. tener respaldo tanto en el oficialismo como en la oposición, cuando aplique.

7. estar profundamente consustanciado con la política del Software Libre y ser inmune a las ofertas inapropiadas.
8. gestionar el plan, los recursos y definir la estructura administrativa apropiada según la cultura de gestión.
9. migrar de inmediato y a su equipo. No sería creíble si no lo hace.

Elaborar plan estratégico

Como expresión de la política del organismo para el software. La política del software debe ser una política de Software Libre.

El plan de gestión para el área TIC puede utilizar las ideas de COBIT [COBIT:OC-98,COBIT:MR-98,COBIT:PG-00,COBIT:RE-98,COBIT:PO-00,COBIT:AI-00,COBIT:ES-00,COBIT

El plan puede prever subplanes por organismos internos y puede dar plazos adicionales para la confección de cada uno de ellos, a nivel ministerios, empresas descentralizadas, u otras estructuras organizativas.

El plan se elabora y revisa continuamente, se va perfeccionando en tanto se va avanzando en las etapas, se conforman las estructuras funcionales pertinentes para el planeamiento y se cuenta con más información.

Esta política expresada en un plan estratégico participativo debe contener: organigrama, metas, indicadores, tareas, presupuesto, etc.

Sensibilizar

Es fundamental sensibilizar e involucrar a la organización en las consideraciones filosóficas, dialécticas y políticas del Software Libre en forma temprana.

En particular se debe actuar sobre los niveles superior, pares y primeras líneas ejecutivas inferiores del responsable de la migración[6.3].

La sensibilización debe incluir un programa de información, formación y capacitación orientado a los diferentes niveles de decisión para que todos los órganos brinden apoyo, colaboración y consenso a políticas de desarrollo sustentable en materia de TICs y Software Libre.

Apenas estén las áreas de IT, de ayuda y soporte capacitadas se debe migrar a los directivos. Las primeras computadoras a migrar son las del Jefe de Estado, los Ministros y demás cabezas de los

poderes públicos, para que comprendan la naturaleza del problema y brinden soporte y seguridad al resto.

Deben generarse consensos en todos estos niveles y ampliar significativamente la adhesión y el compromiso de funcionarios públicos con el Software Libre, facilitando el proceso de migración de todos los usuarios en todos los niveles de la Administración Pública.

Es importante que la comunidad ``sensibilizada'' tenga tareas concretas que realizar como parte de este proceso. La sensibilización debe traducirse en hechos concretos y auditables, como instalar Firefox, u otra tarea simple que dé una sensación de victoria temprana.

Construir la ingeniería técnica del proyecto y realizar pilotos

En el marco de las estructuras funcionales de desarrollo y laboratorio, se deben comenzar a probar las tecnologías previstas y desarrollar las áreas de vacancia (donde no hay Software Libre disponible), definiendo y empaquetando el software que se establecerá terminada la migración. Así se tendrá el modelo tecnológico destino al que debe llevar la migración.

Esto realimentará el plan y permitirá establecer fechas de migración ciertas una vez que se prueben en condiciones ``duras'' los nuevos sistemas.

Se debe involucrar o crear equipos técnicos, consiguiendo las personas suficientes y necesarias dentro o fuera de la organización y contar con el tiempo suficiente para entrenarlos.

El establecimiento de laboratorios de desarrollo y referencia que apoyen a las áreas técnicas preexistentes es fundamental y debe darse en las primeras etapas del proyecto.

Estos equipos se deben automigrar, y de la experiencia ir tomando nota para la planificación e ingeniería del proyecto.

Inmediatamente se debe expandir esta migración al resto de los equipos TIC.

Es fundamental adquirir personas capacitadas y/o prever su capacitación.

Establecer e iniciar monitoreo con diagnóstico de estado inicial

Se debe establecer el monitoreo y el esquema de auditoría dependiente de la máxima autoridad [Viera:RR-03], antes de comenzar los cambios. Es conveniente integrarlos en los mecanismos de auditoría previamente existentes en la organización. Deben contar especialmente con personal especializado para auditar las licencias de software y el grado de avance del proceso de migración.

Entre otros debe:

1. Realizar un diagnóstico, inventarios, y foto previa o evaluación diagnóstico antes de la planificación y antes de comenzar el despliegue. A partir de allí mantener vigilancia permanente.
2. Instruir a los responsables de la revisión de cuentas del organismo para que verifiquen que se cumplan las normas de adquisición de software propietario (restricciones) aprobadas para la migración.
3. Relevar el parque informático e inventariar el hardware donde se migrará el software. **Algunas Referencias**[6.4].
4. Inventariar aplicaciones.
5. Verificar situación de legalidad imperante, si la cuestión legal es un problema. Inventariar todas las licencias para confirmar su existencia y correcta adquisición para evitar posibles problemas legales. Para la paz, prepararse para la guerra. Preparar un registro permanente de licencias y normatizar su uso.
6. Identificar las acciones preexistentes de Software Libre dentro de la organización, la comunidad, el gobierno y en la sociedad.

Modificar la cultura de gestión y de desarrollo, Educación

En muchos casos se deberá modificar la cultura de gestión para adaptarla a un modelo de gestión del conocimiento participativo y libre. Especialmente en las áreas de desarrollo de software que deberán cambiar radicalmente sus formas de trabajo.

Se debe proveer la infraestructura que habilita la cultura del Software Libre a toda la organización: Internet, redes, computadoras y buenos equipos, conocimiento, contactos con la comunidad y soporte. Las medidas de seguridad no deben impedir SSH, jabber, http, etc.

Todo esto debe asegurarse en las primeras etapas al menos a los equipos de migración y pensar en que posteriormente toda la organización debe migrar sus actitudes, cerebros y cultura organizacional. La migración pasa por ``formatear los cerebros'' de los participantes.

Así todas las áreas deben estar dispuestas a autorizar y apoyar los procesos administrativos de la migración sin retaceos.

Es fundamental la educación masiva en cuanto a todo proceso de cambio cultural.

Aprobar la legislación, presupuesto, y normas. Tomar las decisiones necesarias

En diferentes momentos del proceso será necesario preparar y aprobar nueva normativa, permisos, presupuestos, decisiones, adquisiciones, contratación de personal, obras o servicios, y diversas actividades en los niveles necesarios.

Cada una de estas acciones pone a prueba la voluntad política de los responsables institucionales.

Iniciar los procesos, tareas y actividades de ejecución en el resto de la organización

No hay peor plan que el que no se ejecuta. El responsable debe realizar en tiempo y forma cada una de las acciones ejecutivas pautadas en el plan, tomar las decisiones que corresponda en cada momento y remover los obstáculos que se vayan presentando.

Los auditores deben evaluar y monitorear permanentemente el avance del plan y el cumplimiento de sus premisas.

Es el momento de comenzar a migrar los sistemas en producción de la organización.

Modelo de plan estratégico participativo

Misión y objetivos estratégicos

1. Desarrollo endógeno, sustitución de importaciones.
2. Soberanía Tecnológica, eliminar debilidades en cuanto a seguridad, no depender de empresas externas.
3. Participación del trabajador en las decisiones relativas a su trabajo. Auto-gestión.

Visión, organización, estructura, interacción, roles y responsabilidades

Se visualiza un proyecto de migración con un equipo de trabajo central que interactúa con la organización. Este proyecto tiene varias funciones. Estas funciones comprenden tres grandes agrupamientos: ejecutivo, planeamiento y monitoreo.

Se visualiza a cada organización como un conjunto de grupos objeto en los que se proyectan metas y se definen tareas para cada función del proyecto.

El proyecto de migración tendrá un equipo central, y equipos regionales y en cada localidad, distribuidos por área, para dar soporte a las actividades de los grupos objeto en su área.

El responsable y ejecutor de la migración es el gerente de cada grupo objeto. El equipo de migración es responsable de proveer soporte y dinamizar el proceso, pero en última instancia cada área es responsable de su propio cumplimiento de los objetivos de la migración.

El proyecto de migración debe disponer de un espacio físico definido que contenga su equipo central y debe contar con gran movilidad y flexibilidad operativa. Debe tener acceso a todas las máximas instancias ejecutivas de la corporación para participar de todas las decisiones relacionadas.

Factores Críticos

Es importante identificar los factores que son críticos a la migración.

En este sentido es útil revisar los patrones y adaptarlos a la situación concreta y deducir de allí los factores críticos.

Objetivos de Control de COBIT

En el marco del proceso de Gobernanza TIC se debe plantear cómo impacta la migración a cada uno de los Objetivos de Control TIC de la organización que se corresponden con las actividades permanentes, las funciones, el organigrama y las tareas.

1. Planificación y Organización
    1. Plan estratégico
    2. Crear un modelo de arquitectura.
    3. Dirección Tecnológica
    4. Estructura Organizacional
    5. Administrar la Inversión en TICs
    6. Comunicar las directivas y objetivos de las autoridades
    7. Administrar al plantel TIC
    8. Asegurar el cumplimiento de normas externas.
    9. Evaluar riesgos.
    10. Administrar proyectos.
    11. Administrar calidad.

2. Generación de Nuevos Sistemas y cambios a los actuales - Desarrollo y Despliegue
    1. Identificar necesidades de automatización
    2. Identificar, adaptar, o crear software de aplicaciones
    3. Adquirir y mantener infraestructura tecnológica
    4. Desarrollar y mantener procesos
    5. Instalar y acreditar sistemas de información
    6. Administrar cambios
3. Mantenimiento y soporte, Funcionamiento día a día
    1. Definición de Niveles de servicio
    2. Servicios de terceros
    3. Administrar desempeño y calidad
    4. Asegurar continuidad
    5. Garantizar seguridad
    6. Identificar y asignar costos.
    7. Educar y capacitar
    8. Asistir y aconsejar a los usuarios
    9. Administrar la configuración
    10. Administración de problemas e incidentes
    11. Administrar los datos.
    12. Administrar las instalaciones.
    13. Administrar la operación.
4. Monitoreo
    1. Monitorear los procesos.
    2. Evaluar lo adecuado del control interno
    3. Obtención de Garantías
    4. Auditoría Externa
    5. Auditoría Informática Interna.

Medidas para el control

Indicadores

Es importante establecer dos clases de indicadores, unos para definir la eficacia del proceso y hasta qué punto se han alcanzado las metas y otros para definir la eficiencia o cómo se alcanzan las mismas, en tal sentido es importante evaluar qué recursos se han asignado o consumido en el proceso.

Establecer indicadorEs de grado de avance y metas, con evaluación periódica.

Por ejemplo:

1. Número / Porcentaje de personal capacitado (por entidad, y tipo de uso).
2. Satisfacción/conocimiento de los funcionarios en relación al Software Libre.
3. Cantidad, relevancia e impacto económico de proyectos desarrollados con el modelo colaborativo.
4. Cantidad, relevancia e impacto económico de soluciones desarrolladas en Software Libre.
5. Porcentaje de comunicación institucional usando estándares abiertos.
6. Cantidad y relevancia de los sistemas usando estándares de interoperabilidad que permitan el uso de Software Libre.
7. Porcentaje de reducción de gastos en licencias y derechos.
8. Porcentaje de nuevas/viejas estaciones usando Software Libre.
9. Porcentaje de licencias propietarias sustituidas.
10. Porcentaje de estaciones usando ``Firefox", ``OpenOffice" y ``GNU/Linux".
11. Porcentaje de sistemas y servicios desarrollados en Software Libre.
12. Cantidad de proyectos de gran impacto social implantados.
13. Nivel de cumplimiento de las etapas de emisión de normas y reglamentos.
14. Nivel de cumplimiento de las etapas del plan de migración de los sistemas heredados.
15. Cantidad y relevancia de los aplicativos y servidores objetos de migración.
16. Cantidad y relevancia de servicios ofrecidos al público basados en Software Libre.

Metas

Algunas metas ejemplo:

1. Reemplazo Explorer por grupo objeto.
2. Reemplazo Office por grupo objeto.
3. Reemplazo ``Microsoft Windows" por grupo objeto.
4. Reemplazo de servicios como el Active Directory, DHCP, DNS, impresión, archivos, etc.
5. Reemplazo de aplicaciones por aplicación.
6. Reemplazo de bases de datos, constitución de repositorio.
7. Apertura de redes de telecomunicaciones.
8. Cambio de enrutadores con software propietario como los equipos marca ``Cisco" u otros.

9. Cambio del paradigma de seguridad.
10. Todo funcionario debe tener acceso a una terminal, Internet, email, con los puertos elementales abiertos y funcionando sin restricciones.
11. Todos deben estar interconectados.
12. Todos deben tener email y utilizar los sistemas de trabajo colaborativo, especialmente los jefes.
13. Toda actividad debe reflejarse en la web: resoluciones, documentos, formularios
14. Todos usan documentos y comunicación electrónica basados en estándares libres y abiertos definidos por cuerpos colectivos suficientemente amplios y participativos.

Se puede organizar un sistema de premios por meta cumplida, destinados a cada grupo objeto dentro de una clase de los mismos.

Tableros de Control y Mando

Conjunto de indicadores cuyo seguimiento periódico permite conocer el estado de la organización según un modelo determinado.

- Global (integral)
- Estratégico de TIC (largo plazo)
- Desarrollo de TIC (resultado de las actividades)
- Operacional de TIC (diario, decisiones del día a día)

Se debe tener en cuenta, el período cubierto por cada dato, la forma de desagregarlo en los ámbitos, la frecuencia de actualización, la referencia o base histórica, los niveles de alarma, y la presentación gráfica.

Criterios para la información requerida

Requisitos de calidad

confiabilidad
: (fidelidad) de la información: calidad y entrega. Se refiere a la provisión de información apropiada para la administración con el fin de operar la entidad y para ejercer sus responsabilidades de reportes financieros y de cumplimiento.

eficiencia
: de las operaciones, costo. Se refiere a la provisión de la información a través de la utilización óptima (más productiva o a menos costo) de recursos.

Requisitos fiduciarios

efectividad

de las operaciones, confiabilidad de la información. Se refiere a que la información relevante sea pertinente para el proceso, así como a que su entrega sea oportuna, correcta, consistente y de manera utilizable.

cumplimiento:

leyes, regulaciones y acuerdos, muchas veces externos. Se refiere al cumplimiento de aquellas leyes, regulaciones y acuerdos contractuales a los que el proceso de negocios está sujeto, por ejemplo, criterios de negocio impuestos externamente.

Requisitos de seguridad

confidencialidad:

Se refiere a la protección de información sensible contra divulgación no autorizada. integridad: precisión y suficiencia. Se refiere a la precisión y suficiencia de la información, así como a su validez de acuerdo con los valores y expectativas del negocio.

disponibilidad de la información.

Se refiere a la disponibilidad de la información cuando ésta es requerida por el proceso de negocio ahora y en el futuro. También se refiere a la salvaguarda de los recursos necesarios y capacidades asociadas.

El cumplimiento de estos requisitos asegura una institución transparente. Cada uno de estos criterios impacta en cada proceso en forma primaria o secundaria

Según el grado de afectación al proceso se los puede clasificar:

Primario

es el grado al cual el objetivo de control definido impacta directamente el requerimiento de información de interés.

Secundario

es el grado al cual el objetivo de control definido satisface únicamente de forma indirecta o en menor medida el requerimiento de información de interés.

No se aplica

, podría aplicarse pero los requerimientos son satisfechos más apropiadamente por otro criterio en este proceso y/o por otro proceso.

Niveles de Madurez

Cada actividad puede evaluarse y ser ubicada en un determinado nivel de madurez. Características de cada nivel:

0.
No hay proceso reconocible, la institución ni siquiera reconoce la cuestión.

1.
La institución reconoce el problema, hay enfoques ad-hoc para el caso por caso, pero no hay enfoque global coordinado, y el plantel está desorganizado.

2.
Los procesos similares se llevan en forma similar por diferentes personas con la misma tarea. No hay comunicación o entrenamiento formal, las responsabilidades quedan a cargo de cada individuo. Se depende del grado de conocimiento de cada individuo.

3.
Los procesos están documentados y comunicados mediante entrenamiento. Sin embargo queda a cargo de cada individuo el seguir los procesos. Los procedimientos no son sofisticados pero las prácticas están formalizadas.

4.
Se puede medir y monitorear el cumplimiento de los procesos y se toman acciones cuando hay desviaciones. Los procesos están bajo constante mejora. Se usan herramientas automatizadas en forma fragmentaria

5.
Los procesos están en el nivel de mejor práctica. Se usa la tecnología para automatizar el flujo de trabajo, se tiene herramientas para mejorar la calidad y la eficiencia, la organización se adapta rápidamente.

Contrastar:

- Estado actual
- Estándar Internacional
- Mejores prácticas
- Estrategia Institucional

Análisis del gap:

Qué proyectos son necesarios para pasar de un nivel a otro de madurez. Para cada objetivo de control se puede evaluar la relación riesgo/beneficio y el impacto.

Grupos objeto

Se deben definir los grupos objeto de la migración, las tareas y actividades de cada uno y las fechas previstas para cada meta.

Cada grupo objeto es responsable de ejecutar las tareas definidas por el proyecto de migración y pautadas por este con los responsables de cada grupo objeto.

Se deben organizar pruebas piloto, y avanzar por áreas y en cada ciclo a más profundidad. El tamaño de la mesa de ayuda condiciona la velocidad de la migración, pues según la posibilidad de responder las consultas es que se podrá migrar más o menos máquinas en forma simultánea.

Se puede dividir cada clase de grupos objeto en pilotos y por diferentes áreas geográficas en función de los resultados de los inventarios de PC resultantes de las primeras tareas del monitoreo.

Equipo del proyecto
: de migración.

  Es el primer grupo a migrar, como su primera tarea. Tanto los equipos centrales como los localizados o específicos. No es creíble un equipo que plantea el cambio y no ha sido capaz de cambiar. Ni siquiera podrá comunicarse con la comunidad.

  Trabajarán en su propia migración F+OO+G/L, y luego apoyarán las diferentes actividades de su zona de acción.

Mesa de ayuda:
: Debe estar preparada para responder las consultas de los sucesivos grupos que se van migrando. El trabajo aumenta enormemente inmediatamente luego de una migración, va decayendo hasta estabilizarse en valores de la mitad de los que tenían antes de la migración, ya que no hay virus y el sistema es más estable.

  Al menos una parte de la misma debe ser el segundo grupo a ser migrado.

Grupo piloto usuarios:
: El tercero a migrar. Cada tarea puede tener un grupo diferente. Algunas tareas pueden tener secuencias de varios pilotos.

Gerentes:
: Cuarto a migrar.

Usuarios:
: Quinto a migrar. Estos tres grupos: Piloto de usuarios, gerentes y usuarios, pueden ciclarse primero en el área AIT y luego en el resto. En muchos casos habrá una transferencia de conocimiento entre el equipo técnico de migración y el grupo que mantiene esa aplicación en AIT.

Familias, hijos de personal:

Se debe disponer un equipo de soporte y divulgación en el área formativa y de sensibilización preparado para dar soporte a los trabajadores y su familia, dado el impacto del proyecto en las TI personales de los trabajadores y ambiente social.

Otros:

entes estatales/misiones, escuelas, población venezolana/mundial. Evaluar papel de la organización en estas tareas.

Los usuarios básicos incluyen a los usuarios básicos de AIT. Los especialistas y otros grupos también recibirán formación como usuarios básicos, sólo su especialidad será aparte.

Recursos

En el marco del plan estratégico se debe evaluar el impacto del Software Libre en cada recurso.

Datos:

Los elementos de datos en su más amplio sentido, (por ejemplo, externos e internos), estructurados y no estructurados, gráficos, sonido, etc. Se empezó a crear los primeros diccionarios de datos: lugares, organizaciones, personas, bienes, etc.

Aplicaciones:

Se entiende como sistemas de aplicación la suma de procedimientos manuales y programados.

Tecnología:

La tecnología cubre hardware, software, sistemas operativos, sistemas de administración de bases de datos, redes, multimedia, etc.

Instalaciones:

Recursos para alojar y dar soporte a los sistemas de información.

Habilidades de la planta TIC:

conocimiento, conciencia y productividad para planear, organizar, adquirir, entregar, soportar y monitorear servicios y sistemas de información.

Habilidades de los usuarios

para adaptarse y usar las herramientas TIC.

Presupuesto económico

Se debe calcular el ROI (retorno de la inversión) de la migración. Para ello calcular el TCO (Costo total de propiedad), antes y después [Dell:CTO-03,Hayes:DRT-04,Andriole:GGT-03,Orzech:TCO-02,Villoslada:ELM-04,Hall:TCO-02,IDC:W2L-02].

El presupuesto debe contemplar la creación y el sostenimiento de una infraestructura de técnicos para todas las computadoras objeto y acciones de desarrollo, difusión y educación de todas las personas objeto.

Debe ser aprobado por las instancias adecuadas. En el caso de los Estados, puede requerir aprobación parlamentaria, por lo cual se debe planificar con mucha antelación.

Impacto en el Gobierno de las TIC

La migración de las TIC a un nuevo paradigma comunicacional y tecnológico, como lo es el Software Libre, impacta profundamente en la gobernanza de las TIC. En tal sentido es prudente analizar toda la estructura de la gobernanza TIC en el nuevo marco del Software Libre.

El gobierno o gobernanza se define como la estructura de relaciones y procesos que dirigen y controlan la organización en orden de conseguir sus objetivos institucionales añadiendo valor y balanceando riesgos contra beneficios.

El enfoque del control en TIC se lleva a cabo visualizando la información necesaria para dar soporte a los procesos y considerando a la información como el resultado de la aplicación combinada de recursos relacionados con la TIC que deben ser administrados por procesos de TIC.

El marco referencial conceptual puede ser enfocado desde tres aspectos estratégicos que impactan en la eficiencia:

1. Recursos de TI,
2. Metas (objetivos), requerimientos institucionales basados en criterios para la información basados en niveles de madurez requeridos (balanceado con costos).
3. Dominios, procesos, actividades o tareas de TI.

Los procesos pueden ser aplicados a diferentes niveles dentro de una organización. Algunos de estos procesos serán aplicados al nivel institucional, otros al nivel de la función de servicios de información, otros al nivel del responsable de cada proceso.

Figura:
Tareas,
Metas
y
Grupos,
reemplazando
grupos

por
recursos
se
agrega
otra
dimensión
y
se
tiene
el
cubo
COBIT.

División del trabajo: Actividades permanentes

- Prever equipos humanos centrales en grandes organizaciones, el despliegue de actividades concretas a nivel local o regional.
- Así, se generan y coordinan lineamientos en el equipo central. Se realizan reuniones periódicas, se organizan mesas de trabajo y se comparten los avances conformando comunidades del conocimiento.
- Las soluciones, según inventario y racionalización, se distribuyen para su desarrollo en diferentes equipos y laboratorios distribuidos. Cada equipo genera su plan detallado de operaciones.
- Se colabora con la comunidad y otras organizaciones.
- Se trabaja con Cooperativas y empresas de producción social, se utilizan convenios internacionales, se incorporan universidades, se trabaja con la comunidad y se adquieren nuevos equipos.

Figura:
Organigrama
del
proyecto
de
migración

Las actividades permanentes (en contraste con las tareas):

Administrar.
(coordinar, mandar) para que el proyecto avance.

En principio debe reportar a la cabeza de la organización.

Conduce el proceso, toma decisiones, evalúa logro de objetivos.

Comprende:

Coaching.

Asegurar que todos: tomen buenas decisiones; expliquen el proceso a sus superiores; ayuden a los programadores con habilidades técnicas como tests, formateo, refactoreo; vean objetivos a largo plazo; impulsen pequeños cambios; sean un buen compañero de programación, particularmente para los nuevos programadores.

Adquirir juguetes, regalos cuando corresponda y comida es su tarea, con el fin de mantener un buen espíritu de colaboración en los equipos de trabajo.

Intervención.

Muchas veces los programadores u otros no tienen respuestas a algún desafío o no ven los problemas; se requiere alguien que decida. Intervención a tiempo. Cambios de personal.

Opciones.

Abandonar, cambiar la dirección, diferir, crecer.

Factores económicos.

Flujos de fondos y retorno de lo invertido, mortalidad de proyectos.

Variables en el desarrollo.

Costo, tiempo, calidad, ámbito.

Ejecutar.

Funciones y subáreas del tipo ejecutivas:

Sensibilización:

Comunica el plan a los distintos grupos objeto. Coordina primeras acciones de los grupos, coordina formación relativa a las primeras acciones con las áreas pertinentes. Coordina que el laboratorio tenga estos materiales. Provee materiales gráficos y documentales relativos a su área.

Laboratorio

de referencia:

Misión:

Desarrolla la ingeniería del proyecto de migración, y brinda soporte técnico de máximo nivel, constituyendo un laboratorio de referencia de alto nivel, que vincule a los expertos de la organización entre sí y con el resto de la comunidad nacional e internacional.

En el laboratorio se dispondrá de equipos y red en forma libre, fuera de todo tipo de restricciones internas.

El o los laboratorios podrán estar física y organizacionalmente distribuidos pero deberán estar virtual y conceptualmente vinculados.

Funcionará como punto de encuentro y trabajo en la migración de los responsables usuarios de cada aplicación con los expertos de software y en particular de software libre.

Funciones y actividades del laboratorio

1. Busca, prueba y evalúa software
2. Prueba, especifica y certifica hardware
3. Adapta y configura software
4. Realiza pruebas e implementaciones piloto.
5. Asesora y da soporte técnico de último nivel (previo al externo)
6. Coordina desarrollo de software
7. Mantiene servidores, repositorios y sistemas colaborativos
8. Gestiona desarrollo por terceros de software, evaluando su trabajo.
9. Coordina participación en proyectos de terceros
10. Crea comunidades del conocimiento sea por función operativa de la organización, o por técnicas del software. En particular una lista técnica de correo electrónico para la migración. Una de las acciones es migrar las herramientas de trabajo grupal del equipo de migración. Preparar sitio web y repartir información y claves.
    1. es una lista para que el personal de informática de la organización pueda consultar dudas sobre Software Libre
    2. Se invita a la comunidad, con algunos puntos
        1. la participación es nominal y real, se debe usar nombres verdaderos
        2. la información que pueda circular sobre aspectos internos de la organización es confidencial, si bien se intentará limitar esa circulación.
        3. la información técnica sobre SL es libre.
        4. la participación en la lista no implica remuneración ni oferta de empleo, o de contrato de algún tipo, pero eventualmente podrá ser tomada en cuenta en futuros requerimientos como antecedente por diferentes proyectos de la organización que participen en la lista, convocando a personas de su interés.

3. es una lista con perfil exclusivamente técnico

4. sólo los miembros podrán acceder a sus archivos
5. se usará lenguaje castellano

6. se podrá invitar a técnicos de otros organismos del estado, CONATEL, INCE, etc.

11. Tareas varias:

1. adquirir biblioteca

2. establecer canal de TV y radio web educativos

3. adaptar políticas de seguridad al SL.

Contrataciones:

Unidad central de contrataciones. Contrata equipos/cooperativas de desarrollo de aplicaciones externas. Coordina con ellos repositorios ``subversion" y grado de avance. Aprueba pagos. Promociona y ayuda a la creación de cooperativas.

Formación:

Provee formación. Plan de formación por cada grupo objeto. Interacción con otras organizaciones participantes.

Crea material de autoformación/formación asistida local/web.

Coordina con academias y universidades.

Apoyo:

adquisiciones, secretaría, logística, etc.

Mesa de Ayuda.

Algunas referencias: Information Resource Manager: [IRM:SW].

Legal.

Asesoría legal para cuestiones de licenciamiento.

Monitoreo.

Comprende:

Monitoreo, información, seguimiento y control. (COBIT [COBIT:OC-98,COBIT:MR-98,COBIT:PG-00,COBIT:RE-98,COBIT:PO-00,COBIT:AI-00,COBIT:ES-00,C

Analiza la realidad, crea bases de datos e inventarios, releva información, analiza y sintetiza, evalúa grado de cumplimiento del plan. Inventarios. Sigue tanto las actividades continuas como las tareas.

Al relevar parque de PCs, indicando lugar geográfico, negocio, dependencia funcional, nivel y tipo de escritorio, aplicaciones, etc

Así se puede determinar los ``grupos objeto".

Se puede crear una mini aplicación web para esto.

Escuchar:
porque si no, no se sabe qué codificar o testear.

Analizar:
los datos relevados y preparar informes.

Auditoría externa:
El sistema debe mantener auditorías externas.

Realizar métricas:
Mantener una cartelera con las métricas elegidas y actualizarla todos los días. Por ejemplo incluir el número de tests realizados. No incluir más de 4 medidas, e ir cambiando cada tanto las medidas. Los números son una forma de comunicar el cambio. Incorporar semáforos. Es de grado de avance, con evaluación periódica.

Planificar.
Crea, y mantiene el plan maestro y sus especificaciones detalladas, junto a las normas técnicas que se vayan dictando.

Previsión de tareas en un cronograma y actividades permanentes. Definir y mantener la hoja de ruta. Se deben incluir acciones a un nivel detallado, en la política y plan.

Comprende:

Diseñar
para mantenerse codificando, testeando y escuchando en forma permanente. Planeamiento y diseño incremental durante todo el ciclo de vida. El juego de planeamiento.

Determinar
rápidamente el ámbito de la próxima versión y cambios combinando prioridades y estimados técnicos. A medida que la realidad se impone sobre el plan, cambiar el plan.

Roles: usuario, programador, tester, tracker, coach, consultores, jefe, etc.

[PDVSA:I-05,PDVSA:PMS-05,Genovese:PMS-05].

División y organización del trabajo

Actividades y funciones permanentes

Estas funciones podrán distribuirse entre varias personas o cargos.

Laboratorio:

Descritos previamente

Control, Seguimiento y Gestión, (Planeamiento y Monitoreo):

Función que existirá a nivel de ``Equipo Nacional'' y en los equipos de cada negocio.

Planifica las tareas y las monitorea, adquiriendo información, y realizando seguimiento y control de las tareas. (COBIT [COBIT:OC-98,COBIT:MR-98,COBIT:PG-00,COBIT:RE-98,COBIT:PO-00,COBIT:AI-00,COBIT:ES-00,C

Analiza la realidad, crea bases de datos e inventarios, releva información, analiza y sintetiza, evalúa grado de cumplimiento del plan. Inventarios. Sigue tanto las actividades continuas como las tareas.

Releva el parque de computadoras y procesos, indicando lugar geográfico, negocio, dependencia funcional, nivel y tipo de escritorio, aplicaciones, etc., lo cual impactará directamente en la determinación precisa de los grupos objeto.

Formación:

Es responsable de las tareas relativas a la educación del personal en las técnicas requeridas para cada nuevo software que se instale.

Interactúa con otras organizaciones que migran en el estado.

Coordina con academias y universidades.

Sensibilización:

Es responsable de las tareas relativas a la sensibilización del personal y otros actores en la problemática del conocimiento propietario y de difundir las razones y motivaciones del cambio surgido del decreto 3390. Trabajará para difundir la nueva cultura de gestión del conocimiento en la organización.

Contrataciones:

En el nivel central, la función es clave para diseñar políticas y mecanismos de contratación en un entorno colaborativo.

Unidades de contrataciones en cada laboratorio regional. Contratan equipos y cooperativas de desarrollo de aplicaciones. Coordina con ellos repositorios ``subversion'' y grado de avance. Aprueba pagos. Promociona y ayuda a la creación de cooperativas.

Legal:

Asesoría legal para cuestiones de licenciamiento.

Manejo del Cambio (Entrega, migración propiamente dicha):

Entrega, instalación y soporte en sitio del software evaluado o desarrollado.

Relaciones Institucionales:

Coordinación de actividades con proyectos de software libre, la comunidad y sus actores.

Tareas

Figura:
Gantt
de
la
Migración
-
Resumen
1

Cada tarea inmersa en la división funcional de la actividad permanente correspondiente. Las tareas se aplican a cada grupo objeto.

Figura:
Gantt
de
la
Migración
-
Resumen
2

Decisión Política:

La autoridad de la institución decide migrar.

Responsable y equipo de trabajo:

Inicial:

En los primeros días luego de la decisión, el responsable debe armar el equipo inicial para comenzar el planeamiento, monitoreo y demás trabajos.

Completo:

El equipo se va completando a medida que el plan se desarrolla. En particular en las áreas de laboratorio se incorpora gente una vez definidos con claridad los proyectos de desarrollo.

Tareas de Sensibilización:

Para involucrar y sensibilizar a la población objeto mediante un plan masivo de difusión por TV, radio, periódicos y conferencias en todos los ámbitos.

Se prevén varias etapas según los grupos objeto:

Gerencia:

(deciden) Para las gerencias. Contrarrestar el miedo con relación a la continuidad operativa. Prometer planificación profesional y cambiar cuando la situación lo permite.

AIT - SL:

(ejecutan) para los grupos de AIT involucrados en la migración en sus etapas tempranas. Contrarrestar el miedo a la falta de capacidad prometiendo cursos de formación.

AIT completo:

(ejecutan) para el resto de IT.

Masivo:

(usan) para los usuarios. Contrarrestar miedo a no poder funcionar con la promesa de apoyo y soporte en el cambio.

Externo, proveedores familia:

masivo, para el resto de la sociedad. Explicar motivos del cambio y motivos del decreto.

Este plan debe estar coordinado con formación y migración de Firefox, OpenOffice y escritorio. La acción de sensibilización va vinculada a que el grupo a sensibilizar migre Firefox, OpenOffice y eventualmente escritorio. El resultado de la acción es ese y debe ser medido en esos términos. Así pues el proyecto plantea esto y espera esa respuesta del responsable del grupo objeto sensibilizado.

Para cada lugar y grupo objeto definido donde se da la conferencia, se instala el software previsto y se dan los cursos sobre este software instalado. No hace nada sin tener prevista la tarea reacción del grupo objeto. Siempre la demanda estará desde el proyecto de migración al grupo y no al revés.

Debe incluir propaganda, etc. como parte del trabajo de sensibilización y estar centrado en una buena comunicación.

Se deben preparar los medios de la sensibilización: textos, web, gráficas, y el software de difusión. Incluso livecd.

Se debe asentar la sensibilización el cumplimiento del decreto, y a partir de allí explicar sus razones.

A cargo de Sensibilización.

Planeamiento:

A cargo de Planeamiento y Monitoreo.

Preliminar:

Redacción del Plan Estratégico Inicial. Se trazan las líneas gruesas del proyecto de migración, se dividen tareas, se plantean fechas de referencia.

Permanente:

Se ajusta el plan a medida que se va desarrollando. Se incorporan precisiones y se profundiza en cada etapa.

Monitoreo

Foto e inventario inicial:

Se establece la situación antes de comenzar a actuar. Se realizan los inventarios de base. Se definen precisamente los grupos objeto.

Control:

Se mantiene un monitoreo permanente de las tareas.

Figura:

Gantt
de
la
Migración
-
Decisión,
Equipo,
Sensibilización,
Planeamiento
y
Monitoreo

Aplicaciones Estructurales:

Como todas las aplicaciones deben tener en cuenta la cultura de desarrollo de los ambientes UNIX [Seebach:AWL-04], y ``GNU/Linux" y el software libre en particular [Kroah:LKD,Adelstein:ULO-05,Chance:SSO-05].

Figura:
Gantt
de
la
Migración
-
Aplicaciones
Estructurales

Inventario, análisis y diseño.

Integrador y Diferenciador.

Equipo de análisis, diseño, e integración de aplicaciones. Para dividir el trabajo, estudiar cada nueva aplicación e integrarla al esquema de trabajo global. Este grupo será permanente y quedará operando luego de la aplicación.

Para muchas tareas se deberán armar equipos de programadores para empezar a construir una plataforma abierta de desarrollo de software para las aplicaciones. En algunos casos estos equipos programarán, en otros coordinarán la interacción con contratistas/comunidad.

Designar un responsable por cada nuevo desarrollo que se vaya a emprender, designar un responsable de participar en proyectos externos preexistentes que sean de interés. (IRM, Apache, Mozilla)

Este equipo monitoreará a los otros.

Estudiar cómo consolidar las aplicaciones preexistentes, y definir de cuáles se usarán los códigos viejos, de cuáles se comenzará un nuevo proyecto y para cuáles se usará software ya existente.

Decidir cómo se distribuyen las funciones requeridas entre aplicaciones, cómo se construye cada aplicación -bajo Web o de escritorio- y quién se hace cargo de la misma. Decidir esquema de distribución de aplicaciones entre responsables y regiones.

Se deberá evaluar cómo la organización contribuirá con cada proyecto, por ejemplo aportando fondos para salarios de programadores en Venezuela. Esto también puede ser un anuncio público importante. Por ejemplo se anuncia que usará Apache, y que contratará 2 personas locales para que se integren al desarrollo de Apache, lo que se conversará con el consorcio Apache, el cual se puede integrar.

Evaluar qué personas o cooperativas están contribuyendo significativamente con los proyectos. Se podrá analizar, establecer contratos y diseñar una métrica para los pagos.

Liberación de aplicaciones propias

Se anunciará la decisión. Se estudiará legalmente cada aplicación para determinar la factibilidad de liberarla. Se estudiará desde el punto de vista del contenido de información cada aplicación para evitar liberar datos internos de la organización. Se evaluará el impacto en la seguridad de liberar aplicaciones que hayan funcionado con un modelo de seguridad vía oscuridad. Se liberarán las aplicaciones que vayan pasando estas revisiones.

Marco de Acceso a Bases de datos:

Se crearán librerías comunes en diferentes lenguajes para acceder a estas BD. Una vez construida esta capa se podrá conectar con las nuevas aplicaciones.

Marco de Aplicaciones Web:

Se priorizará la plataforma Web en el desarrollo de sistemas e interfaces de usuario, LAMP + AJAX. Todos los sistemas posibles serán de este tipo.

Se revisará el uso de los sistemas de autenticación y ACL (listas de control de acceso) .

Crear y/o definir las librerías, plataforma, herramientas y marco de trabajo para aplicaciones WEB.

Seleccionar entre Perl, PHP, Python, Ruby, etc.

Hacer que todos estos sistemas cumplan las normas del consorcio W3C

Dentro de cada aplicación en servidor se trabajará en el sistema operativo y hardware óptimos bajo los cuales funciona.

Se evitarán plataformas cliente-servidor no web, si es necesario se incorporarán en este ítem.

Marco de Aplicaciones de Escritorio:

Incluye las herramientas de desarrollo para aplicaciones de Escritorio. Las librerías locales, GTK, Qt, wx, etc.,

Crear y/o definir las librerías, plataforma, herramientas y marco de trabajo para aplicaciones de escritorio. Armar equipo de programadores.

En esta área estarán los proyectos de GIS.

Desarrollo: [Mono:SW,RealBasic:RPA,wxWindows:SW,IBM:PMA].

Portando aplicaciones: ``Visual Basic'' con ``Real Basic'' [RealBasic:RPA], Marco de software libre para programación GUI (interfaz gráfica de usuario) en plataformas cruzadas C++ [wxWindows:SW], MFC (Microsoft foundation clases) a ``GNU/Linux'' [IBM:PMA].

``GNU/Linux registry'': [LRP:SW].

Sistemas de configuración: [UniConf:SW].

Marco de Aplicaciones Cluster:

Desarrollo de aplicaciones de clustering [OpenMosix:SW], grid y visualización.

Hardware como en software para las aplicaciones de cómputo o visualización intensiva.

Marco de Aplicaciones de Automatización:

Desarrollo de aplicaciones de Automatización. Proyecto SCADA, telemedida y control.

Centro de Bases de Datos:

Seleccionar e instalar motores de Bases de Datos con postgres [Postgres:SW], mysql [MySQL:SW], R, MAXDB (sap mysql), Firebird, etc.

Figura:
Gantt de la Migración - Centro de Bases de Datos, Seguridad y Autenticación

Se pensara en establecer repositorios centralizados con redundancia regional para las Bases de Datos. Estos repositorios estarán normalizados y responderán consultas SQL a cualquier aplicación y usuario con derecho.

Autenticación, seguridad y monitoreo:

Se probará ``Kerberos'', ``heimdal'' o equivalente, junto con ``OpenLDAP''. Eliminar ``active directory'' y sus sistemas de dominios.

Se realizará toda la ingeniería de seguridad de redes y usuarios. Se revisarán sistemas de control y auditoría.

Monitoreo y Control: [Nagios:SW].

Se verificará la conectividad de las estaciones preexistentes en Windows a estas nuevas redes y mecanismos de autenticación, durante la etapa de transición.

Redes, routers

Cambiar tecnología de red, enrutamiento: DNS, DHCP. Capa de IP y DNS.

Figura:
Gantt
de
la
Migración
-
Redes

En el laboratorio se constituirá un piloto de ``nueva red'' donde se integrarán routers, DNS, DHCP. Para cada grupo de trabajo. En esta línea de trabajo también se evaluará el monitoreo, vía SNMP.

Herramienta de colaboración:

Software de trabajo en Grupo. Comenzará como un trabajo de Integración y Consolidación de muchas herramientas, convendrá desarrollar una sola para todas estas funciones.

Figura:
Gantt
de
la
Migración
-
Colaboración

1. correo de escritorio y vía web [Openwebmail:SW] y/o Thunderbird, Evolution, KMail,
2. listas de correo, mailman,
3. conversación, jabber,
4. publicación de noticias, blogs,
5. servidores web, publicación de trabajos (Apache, CMS), wikis (mediawiki),
6. servidores de archivos compartidos, ``Samba'', ``NFS'', u otros,
7. impresión en red,

8. versionado, ``subversión" para liberar proyectos propios cerrados
9. repositorio de software, Establecer repositorios y formas de distribución de software, crear CD-ROM. (incluso de Software Libre para win en las primeras etapas), rsync, p2p, etc.
10. respaldo,
11. reserva de recursos
12. calendario y agenda,
13. directorio (integrado con autenticación),
14. manejadores de proyectos [opensourcecms:SW], phpproject.
15. VOIP. Instalar Asterisk, contratar servicio externo de ruteo de llamadas.
16. Inventarios y tickets de soporte.

Preparar entorno colaborativo y cambiar cultura de cooperación y trabajo. Compartir experiencias, mantener comunicación permanente.

Se trabajará con los servidores y cuando sea necesario con los clientes.

Algunas referencias:

Impresión: [Easy:PPS,Adobe:TWP,CUPS:SW].

Samba: [SAMBA:SW].

GAIM: [GAIM:SW].

Escritorio

Se prepara el escritorio de uso habitual en la organización.

Figura:
Gantt
de
la
Migración
-
Escritorio
1

Hay varias subtareas:

Adecuar Hardware:

Se debe investigar la compatibilidad del hardware preexistente y definir requisitos y listados detallados de hardware compatible para compras futuras. Establecer especificaciones de compra. Requerir drivers libres para cada dispositivo particular.

Establecer criterios mínimos para proceder a diferentes formas de migración según el Hardware disponible: aumentar memorias a 128MB, evaluar PXE o LTSP para máquinas más chicas.

Algunas referencias:

Hardware: [Ndiswp:SW,Winmodems:SW,Gphoto:SW,RedHat:LHP].

Scanners: [SANE:SW].

Smartcards: [MUSCLE:SW,USDP:SW].

Autenticación de ``GNU/Linux'' sobre las redes actuales:

Se debe verificar y establecer los mecanismos para autenticar las estaciones de trabajo GNU/Linux a las redes actuales, estudiar los problemas y las metodologías de actualización y uso de los pilotos que se vayan construyendo en las condiciones actuales.

Navegador:

Migrar Firefox y adaptar páginas y sistemas web,

Se instalará en todos los escritorios. Se instruye al centro de soporte o mesa de ayuda para recibir llamadas de gente avisando que determinadas páginas no son visibles, este centro avisará a los responsables para que las cambien (internas, externas), estos centros además verificarán cumplimiento de W3C. Se prepara esto en el laboratorio.

Oficina:

Migrar software de oficina,

Preparar un escritorio estándar con aplicativos libres de oficina y acceso a

Figura:
Gantt
de
la
Migración
-
Escritorio
2

Proceder a notificar en legal forma a los empleados que esas son las herramientas apropiadas, oficiales y legales para su trabajo diario.

Desinstalar software innecesario. Definir necesidades de diferentes aplicativos.

Evaluar OOO, abiword, u otros.

Pensar en un sistema inteligente de gestión de documentos orientado a procesos.

Para el OOO:

- se intentará instalar en muchos escritorios, preparar versión especial, diccionarios y ``accesos rápidos''.
- se instalará en forma escalonada por grupos objeto y evaluando reacción. cuando un grupo esté listo, se pasará a otro. Se evaluará en cuáles se pudo y en cuáles no funciona. Se mantendrá en paralelo ``Microsoft Office'' hasta que una buena parte de los grupos objeto estén migrados.
- Una vez que un buen porcentaje esté cubierto, especialmente todos los secretarios y gerentes, se dispondrá el uso de ``.odt'' como formato de intercambio estándar y se prohibirá el uso de ``.doc'' en forma interna. Se seguirán recibiendo ``.doc'' de afuera, pero no se emitirán más. Para afuera se podrán emitir docs de OOO, pero sólo a pedido, por defecto se enviará ``.odt''.
- La mesa de ayuda recibirá pedidos de ayuda con migración de datos (planillas y presentaciones) Se prepara esto en el laboratorio y se regula la velocidad de avance de la migración según el grado de capacidad de la mesa de ayuda.

Algunas referencias:

OpenOffice.org: [OpenOffice:SW].

Integración Mozilla-ooo: [polinux:KOI].

Otro

software de escritorio: GIMP, CAD, etc. Buscar software libre para Windows que tenga versiones en GNU/Linux.

The Gimp: [GIMP:SW].

Sistema Operativo para escritorio:

``GNU/Linux''.

Pensar en dual boot para primeras pruebas. Pensar en PXE, LTSP o similares, rdesktop en casos extremos.

Figura:
Gantt
de
la
Migración
-

Escritorio
3

Se construirá un piloto de estación de trabajo con ``GNU/Linux". Se prestará especial atención al funcionamiento de hardware especial, y de software corporativo. Se dispondrá un método de actualización central basado en la autenticación. Automatización de instalación y actualización de escritorios.

Se instrumentará el sistema de instalación centralizado de las aplicaciones de oficina ya probadas en ``Windows" bajo ``GNU/Linux". Se evaluará koffice , LaTeX/XML para casos especiales y se pensará en un sistema moderno de gestión de documentos.

Algunas referencias:

Equivalencias de ``Microsoft Windows" a "GNU/Linux" [Kachurov:LES-03]

SVG: [OCA:SVG,SVG:SW].

Distribuciones: Que ``distro" usar es el problema del novato, cualquier profesional debiera poder usar diferentes formas de instalar el mismo software, sea cual sea el medio o la paquetería usada, sobre todo en instalaciones masivas, de cualquier forma es interesante listar diferentes ``distros":
[Ututo:SW,RedHat:SW,Suse:SW,Debian:SW,Gentoo:SW,Slackware:SW,LFS:SW,Mandriva:SW,Wik

Usabilidad: [Horstmann:SSL-03,Turner:GUS-04].

Varios:

[iFolder:SW,RSYNC:SW,Kiosk:SW,Novell:EC,Webmin:SW,SuperKaramba:SW,KDE:SW,KDEDoc

Configura servidores para FF y w3c:

Apenas se comienza a instalar el Firefox se recibirán informes de páginas de la organización y externos con problemas. Se debe ir adaptando los servidores de la organización a las normas W3C y para que funcionen con el w3c. Se deberá conversar con las organizaciones externas para que cumplan los estándares.

Repositorio SLW:

Preparar SLW (cdrom y repositorio).

Software Libre que funciona tanto en ``Windows" como en ``GNU/Linux", se instala primero en ``Windows", para iniciar el entrenamiento y que las personas vayan migrando sus datos a estas aplicaciones. Luego será más simple migrar el sistema operativo. Paquetes para oficina, gráficos, clientes de correo, mensajería, calendario.

Distribuir estos aplicativos mediante CDROM a todas las Oficinas, otras entidades relacionadas, e incluso público en general.

Todo esto en ``Microsoft Windows", como primer paso.

Aplicaciones de cada Negocio:

Un equipo por aplicación específica a crear o mantener.

Se deberá conversar con los mantenedores de las aplicaciones para integrarlos a los nuevos equipos de desarrollo.

Figura:
Gantt
de
la
Migración
-
Aplicaciones
de
cada
negocio

La idea es tener un esquema donde existan responsables por aplicación, otros por arquitectura común, y otros por aplicación externa preexistente, que concentren los trabajos en cada aspecto común. Cada uno de estos responsables decidirá qué contribuciones se aceptan en cada proyecto y cuáles no están todavía listas.

Aplicaciones de AIT:

Ídem para estas aplicaciones.

Figura:
Gantt
de
la
Migración
-
Aplicaciones
de
AIT

Subtareas particulares para proyectos de software

No todos los proyectos constarán de todas estas tareas.

Etapas de cada tarea o línea de trabajo del laboratorio

Figura:
Tareas en cada línea de trabajo, L: laboratorio, F: Formación, AIT: Aut. Inf. y Tel., RU: Responsable Funcional.

EvaTest - Evaluación y test:

> **Laboratorio**
>
> Para cada requerimiento, se debe investigar la existencia de software libre disponible, definir el inicio de un nuevo proyecto o participar en proyectos preexistentes para su mejora.
>
> Se debe comenzar de inmediato el proceso de evaluación y prueba en los laboratorios, si es posible usando test automáticos, apenas el proyecto tenga sus primeras líneas de código. (Primero los Test.)
>
> Los laboratorios incluyen el diseño y análisis funcional de las necesidades de cada área en forma conjunta con el responsable de negocio, la investigación de alternativas libres y la toma decisiones sobre las opciones de desarrollo.

Des - Desarrollo

> **Laboratorio**

Son varias las posibilidades para trabajar, según cada caso.

1. **Diseñar, analizar, y/o desarrollar** un nuevo proyecto, por sí o mediante terceros.
2. **Liberar** un software propio o de terceros (previa negociación) y/o posiblemente **portarlo** a una plataforma libre.
3. **Sumarse** a un proyecto preexistente con terceros.
4. **Usar** software libre preexistente tal como está.

Este punto y el anterior están íntimamente vinculados.

Val - Validación :

Responsable Funcional

Una vez que determinado paquete ha sido aceptado en el laboratorio, corresponde que sea evaluado y validado por el responsable usuario de la aplicación.

PINU - Prácticas de instalación y normas de uso

Laboratorio

Definición de prácticas de instalación y normas técnicas, habitualmente considerados como estándares [Saravia:RR-03,Saravia:NT1-03,Saravia:NT2,Saravia:NT3,Saravia:NT4]. Así se deben adoptar estándares abiertos en el desarrollo de tecnologías de información y comunicaciones y en el desarrollo multiplataforma de servicios y aplicativos.

La fijación de estándares no debe frenar nunca el proceso creativo, la discusión de alternativas no debe nunca impedir el avance. El software libre es creativo y está en un proceso permanente de cambio y renovación.

El órgano competente debe estar atento al proceso y dictar las normas que surjan como necesarias durante la realización del plan.

1. Políticas de Hardware.

   Normar la adquisición de hardware compatible con las plataformas libres. Modificar normas técnicas del estado/organización.

   Establecer normas de adquisición de nuevo hardware, pedir compatibilidad 100% con software libre [Saravia:GAS-05].

2. Políticas de software.

   Fijar política clara en relación a requisitos para uso, manejo de inventario, y compra/alquiler y tipo de licencias privativas, lo ideal es fijar condiciones de licenciamiento de software

coincidentes con la definición de software libre [Saravia:RRP-01,Saravia:NT-05,Saravia:GAS-05,Saravia:NT2].

Fijar política clara en relación a proveedores de software en cuanto a que el derecho habiente de software es el que lo demanda y que este es distribuido bajo la GPL [Stallman:GPL] universalmente.

3. Definir cómo se administran las obras intelectuales propias, cómo se maneja con subcontratistas y se consigue ser derecho habiente de todo ello. Definir registro. Definir como libre GPL [Stallman:GPL] universal ese software, poner repositorios web. Autorizar a empleados a participar en proyectos de software libre externos, Financiar esa actividad.
4. Definir formatos de documentos e intercambio de datos

   Definir estándar de documentos, abiertos, en principio preparar sistema de documentos orientados a la Web. HTML. (mejor pdf que doc, mejor html que pdf)

   Definir e implantar estándares de interoperabilidad, archivo, e intercambio de documentos no sujetos a decisiones de corporaciones monopólicas. No tener como única forma de comunicación con los ciudadanos documentos Word, Excel, o aplicativos cerrados para un solo sistema operativo. Garantizar al ciudadano el derecho de acceso a los servicios públicos sin obligarlo a usar plataformas específicas.

   Un buen estándar es el oasis, ``.odt'' del OpenOffice.

5. Establecer normas de redes
6. Establecer normas de Bases de datos y repositorios centralizados.
7. Definir lenguajes y estilos de programación, interfaz web, etc.
8. Definir sistemas operativos, distribuciones, repositorios, etc.

EIT - Entrenamiento de IT

Formación

EMA - Entrenamiento de la mesa de ayuda

Formación

PIL - Pilotos

AIT

- Instalación
- Cursos
- Adaptación

EM - Entrega a la organización y mantenimiento:

AIT

Entrega masiva por grupo, según la sobrecarga que pueda atender la mesa de ayuda.

- Instalación
- Cursos
- Adaptación

Subtareas de las tareas de despliegue

Instalación:
Desplegar el software en las máquinas de los grupos objeto.

Cursos:
Formar a los usuarios para que puedan utilizar el software que se les instaló.

En la tarea participarán expertos en documentación, traducción y multimedias (video). Se creará un servidor con cursos y material informativo y formativo de autoformación y formación asistida tanto local como vía web en ``moodle". Se preparará material y textos educativos [Westermann:CA-04]. Funcionará en forma coordinada con el laboratorio.

Es importante formar a la planta de IT en la cultura hacker [Raymond:CSH].

Adaptación:
Dar soporte especial a los nuevos usuarios hasta que se adapten al nuevo software y puedan ser eficientes con su uso.

Guía de adquisición de software para gobiernos y grandes organizaciones

Subsecciones

Necesidad

Ya que los gobiernos:

- adquieren software para muchas computadoras,
- respetan normas y procedimientos formales de adquisiciones,
- su funcionamiento debe ser extremadamente transparente y garantizar condiciones de seguridad, privacidad, permanencia en el tiempo de la información, etc. Ver Software Libre en el Estado, preparado por Hipatia para el ICA [Saravia:SLA-03, 29 requisitos]

Parece necesario discutir cuáles son las pautas para la adquisición de software en este tipo de organizaciones. Esta guía pretende ser una ayuda para aplicar correctamente las normas públicas en la adquisición de software.

El software -que no sea del derecho público o sin copyright- se adquiere mediante contratos entre el usuario y el derecho-habiente del mismo. Es fundamental definir claramente este tipo de contrato en los pliegos de adquisiciones. Los estados no tienen por qué aceptar contratos de adhesión, tal cual es la costumbre ahora. A medida que se termine con el monopolio de Microsoft -tarea de los gobiernos- será posible mejorar las condiciones de adquisición.

Adquirir software no implica necesariamente una compra: puede adquirirse sin necesidad de ningún trámite, como suele suceder con el Software Libre. Habitualmente se lo baja de Internet. Otras veces se tendrá, incluso para Software Libre, que contratar servicios para grabar CDs, empaquetar manuales, instalarlo, configurarlo, adaptarlo, formar a su gente, etc.

Se debe tener claro y separar correctamente que son varios los ítems habitualmente involucrados en la adquisición de software: adquisición de derechos de uso (para el Software Libre, habitualmente se lo adquiere por el hecho de bajarlo de Internet), contratación de servicios diversos vinculados: instalación,

medios de distribución (CDs, DVDs, etc.), manuales, configuración y adaptación, cursos de formación, etc.

Entonces, ante la necesidad de software, la primera consideración es si es necesario realizar un trámite formal de adquisición de derechos, sea por pedido de precios o licitaciones, o si directamente se instruye a las oficinas técnicas correspondientes para utilizar Software Libre disponible en repositorios públicos.

Sólo cuando se necesite software no disponible por esta vía, será pertinente recurrir a procesos formales de adquisición en cuanto a los derechos de uso. Así podrá decidirse crear software nuevo, ya sea por personal propio o contratado, o adquirir derechos para usar, inspeccionar, modificar y redistribuir software preexistente. Sólo en este último caso será necesario un proceso formal de adquisición.

Debe tenerse en cuenta el costo de este proceso: convocatoria, compulsa, impugnaciones, y luego el costo de mantener la infraestructura legal y de auditoría del correcto uso del software con restricciones privativas.

Leyes y normas

No siempre será necesaria una ley especial para regular la adquisición de software por los estados. Puede ser recomendable modificar las normas vigentes sobre compras, fijar una política de estado, adaptar las normas técnicas, o ``instruir" a los funcionarios. El instrumento a adoptar dependerá de la cultura política de cada país: una ley, un decreto, una resolución, o simplemente una formación adecuada para los responsables de las adquisiciones en cada oficina.

Han existido casos de planteos judiciales de inconstitucionalidad sobre normas legislativas que disponen el uso obligatorio de Software Libre, o simplemente lo promueven. El fundamento de tales planteos es que las normas discriminarían a las ``empresas de software privativo". En realidad estos planteos no tienen fundamento, pues cualquier proveedor puede ofertar su software con la licencia que le pidan. Ni las empresas de software, ni el mismo software, son libres o privativos por sí. Es cada contrato entre cada usuario y cada derecho-habiente el que fija cómo se licencia cada software en determinada situación.

Pero, por otro lado, plantear las leyes de esta forma es invitar a una acción de inconstitucionalidad. La recomendación es que las normas -sean leyes o normas técnicas u otras- no especifiquen sus requerimientos imponiendo un modelo de licenciamiento concreto o incluso una categoría o conjunto de modelos concretos, sino que se especifiquen los requerimientos legales que sean convenientes para el estado: indicando libertad de instalación en todas las máquinas, libertad de inspección, modificación, compilación, distribución, etc. Lo mismo en cuanto a los efectos, pero redactado en forma diferente.

Sí es necesaria una ley para administrar el derecho intelectual del estado, los copyrights que este tiene y el tipo de restricciones que establece a terceros sobre los programas de los que es derecho-habiente. Así debe fijarse por ley:

- que el estado es derecho-habiente de todo software que cree, aún si los autores no están bajo relación de dependencia, por ejemplo mediante contratos de obras o de servicios,
- que todo el software del cual es derecho-habiente se licencie universalmente a terceros bajo la GPL y que el estado mantendrá repositorios públicos bajo sistemas web y/o tipo CVS. Así cumplirá el requisito de libre acceso a la información del estado y todos podremos usar en condiciones de igualdad el esfuerzo de nuestros estados. Y nadie podrá apropiárselo y usufructuarlo sin aportar su parte (vampirismo).

Requisitos en los pliegos de condiciones y/o normas técnicas

1. Comprar las mayores cantidades posibles. Un principio de la compra pública indica que se deben juntar los requerimientos de muchas oficinas en un solo pedido para mejorar las condiciones de compra.

   Como existe software que debe ser usado en ``todo el estado", conviene adquirirlo de una sola vez para cada ``estado", en cada período de tiempo.

   Se recomienda especificar en el contrato que regule la adquisición (en particular el licenciamiento) que el software podrá ser instalado en todas las computadoras del estado sin límite y que el precio ofertado incluye esa posibilidad.

2. No adquirir hardware y software juntos o en forma combinada.

   Un buen software se caracteriza por ser compilable en muchas arquitecturas. Adquirir software que pueda funcionar en muchos tipos de máquinas, permitirá al gobierno que, cuando cambie la tecnología de hardware, sus inversiones en software puedan seguirse usando, con mínimos cambios.

   Los pliegos de condiciones de las licitaciones de hardware indicarán que los equipos se entregarán sin software y que el software será instalado por el estado o mediante contratos separados bajo estricta supervisión del estado.

   Esto garantiza que se recibe y resguarda todo el software necesario para operar y para reinstalar si se requiere. No existirán drivers ocultos que luego no se puedan conseguir.

   En las adquisiciones de licencias de software se debe especificar que éste deberá operar en toda la gama de equipos que el estado usa y que puede llegar a usar. Se debe pedir código fuente que pueda ser recompilado en diferentes arquitecturas. Siempre será más fácil añadir una nueva arquitectura a un software que funciona en varias, que convertir un software mono-arquitectura en multi-arquitectura.

   Así se podrá mover el software de una computadora a otra.

Durante la vida útil de un equipo suele ser necesario instalar varias veces diversas versiones del mismo software.

Este es un principio de toda compra pública, separar lo más posible las compras, no favorecer las combinaciones de productos y la formación de monopolios cruzados del estilo Intel-Microsoft.

3. Conviene separar la prestación de servicios de mantenimiento o instalación de la adquisición de licencias. De esta forma muchos proveedores podrán competir en este rubro. Se propenderá a la existencia de proveedores que puedan manejar entornos diversos de software.

4. Se debe especificar que se debe proveer el código fuente y el derecho a adaptarlo y recompilarlo. Esto es necesario por varios motivos, en primer lugar para saber qué hace el software en cuestión y poder auditar su real función. Luego para poder modificarlo y adaptarlo en condiciones cambiantes, sin depender de contratos particulares con la empresa proveedora. Luego para poder adaptarlo a nuevas arquitecturas, de haber obsolescencia tecnológica.

5. Se debe especificar que el estado debe poder redistribuir ese software a otras personas jurídicas. Así podrá usarlo en programas de gobierno electrónico, educación pública e interacción con los ciudadanos.

6. Las licencias no pueden caducar en determinado período de tiempo.

7. Se deben inventariar las licencias de software por separado del hardware y llevar un estricto control de dónde y cómo se usa cada licencia privativa. Un ejemplo: ``Registro de programas de uso restringido, UNSa" [Saravia:RRP-01].

8. Los estados deben designar a los responsables de mantener a resguardo los medios de instalación, las facturas y las obleas o certificados cuando se compran licencias privativas. Se deberá prever si los manuales quedan con el usuario o van a bibliotecas comunes.

9. Se debe instruir a las auditorías y organismos de contralor para que informen sobre incumplimientos a estas normas, realicen planes sistemáticos y periódicos de control e impulsen los sumarios correspondientes.

   Este trabajo es muy costoso, pero indispensable en caso de usar software privativo.

10. Se debe indicar claramente a los funcionarios del estado que usar software ilegal implica sumarios, y que no se debe regularizar software por acuerdos privados con empresas, sin iniciar causas penales a los responsables de la violación de la ley en primer lugar. Tampoco es legal contratar planes globales o realizar acuerdos de este tipo sin licitación pública.

11. No se deben indicar marcas en las especificaciones de los pliegos o normas técnicas. Windows es una marca, GNU/Linux es una marca. Se deben especificar requerimientos necesarios o convenientes, sean técnicos, económicos o contractuales de licenciamiento. Todo lo que se pida debe ser directamente vinculable a las necesidades del organismo.

12. No se deben solicitar productos que sólo son necesarios para una marca en particular. Se debe pedir que estos productos sean parte del ítem particular en que esa marca puede concurrir. Así, se deben solicitar sistemas operativos razonablemente seguros y que garanticen un normal funcionamiento sin verse inundados de virus. Entonces, quien provea Windows deberá entregar y hacerse responsable de solucionar estos problemas típicos de esta plataforma entregando sistemas antivirus y firewalls.

Neutralidad tecnológica, un ridículo y patético oxímoron.

¿Neutralidad tecnológica o tecnología apropiada?

Subsecciones

Una táctica más de Microsoft para frenar el avance del Software Libre

Microsoft y otros están reaccionando fuertemente contra el Software Libre. Esta reacción activa múltiples tácticas. Desde los más burdos intentos de asociar la libertad del conocimiento con el comunismo [Gates:QCS-04,Raymond:HD-04], hasta los estudios muy serios y profesionales, pseudo-científicos diríamos, de consultoras más o menos independientes, en relación al concepto de TCO[8.1] y otras cuestiones más o menos técnicas [DelBianco:FSF-05].

Como una línea intermedia de propaganda ideológica se difunden los conceptos de ``libertad de elección" [Hipatia:SM-04] generalmente en los movimientos sociales, vistos simplemente como ONGs, y se introduce el concepto de ``neutralidad tecnológica" en los foros públicos y gobiernos [Evangelista:AC-04]. Analizaremos este último.

Tecnología, derechos, cultura y neutralidad

Las elecciones tecnológicas modifican culturas, las elecciones de las comunidades afectan la tecnología.

Las comunidades no son neutras al elegir tecnología [Quiros:NTA], ni pueden serlo, lo hacen según sus intereses, deseos y límites.

En cuanto a los límites, las normas de ``Propiedad" Intelectual impiden o condicionan seriamente el ejercicio de la libertad de elección. Para la mayoría de la humanidad usar el software de Microsoft es una obligación, es lo que tienen en sus trabajos, en sus ``cibercafés", en sus escuelas, lo que reciben a través de documentos en formatos exclusivos [DiCosmo:TC-98], es lo que el estado les pide para

interactuar, etc. No tienen opción. Esa situación les impone restricciones adicionales y los obliga, muchas veces, a delinquir para poder usar determinado software, en una estudiada metodología de capturar mercados [Gates:ESA] y convertirlos en ``adictos". Así la gente no puede ofrecer resistencia en el marco de la ley, y violar la ley sólo ayuda al monopolio. Este software no se puede inspeccionar, no se puede copiar ni menos compartir, no se puede usar libremente. No ofrece ninguna libertad, menos de elección.

Por otra parte, la cuestión del copyright del software y las distintas formas de licenciamiento o las restricciones a la libertad[8.2] del software, tienen poco que ver con las tecnologías sino con quitarle derechos al usuario.

Las mismas tecnologías se pueden usar bajo cualquier modelo de licenciamiento o con distintos conjuntos de derechos. Existen ``implementaciones" del mismo software distribuidas una bajo modelo privativo y otra bajo licencia libre [Mysql:DLP]. El mismo código puede licenciarse de una y otra forma a ``libre elección", según convenga al derecho-habiente[8.3] y al ``licenciatario". Los usuarios y consumidores, los estados, o las empresas pueden requerir libremente el mismo software bajo modelos diferentes y negociar con los derecho-habientes de los mismos las condiciones que desean y cuánto deben pagar por ello.

Una cosa es la tecnología de desarrollo o de funcionamiento de un software y otra la forma en que se lo licencia a terceros, en este caso al gobierno. La tecnología no tiene, en una primera aproximación[8.4], relación alguna con las relaciones contractuales entre licenciatario y licenciado.

Una política estatal que indique cuáles son los requisitos que deba tener el software a ser usado en el gobierno en cuanto a las formas jurídicas y económicas de licenciarlo, no impide ni limita a ningún proveedor, pues todos pueden negociar un precio por lo que el gobierno pide y competir libremente por satisfacer el requerimiento público.

Esta es, pues, la discusión de fondo: qué modelo de licenciamiento garantiza perdurabilidad, transparencia, independencia del proveedor, seguridad, capacidad de auditoría [Saravia:SLA-03, Ver 29 requisitos] etc., todos ellos requisitos incuestionables del software para el estado. ¿Deben los consumidores o el estado o las empresas estar atados a las condiciones de licenciamiento de un proveedor cualquiera? ¿O podrán fijar las condiciones de compra según lo que necesiten? ¿Deben limitarse a aceptar los contratos de adhesión de las empresas derecho-habientes?

Monopolios vs. mercados libres

En el caso del estado u otras organizaciones con múltiples personas y oficinas, ¿pueden o no decidir que quieren instalar el software en todas sus PCs? ¿Pueden o no decidir que quieren tener derecho de modificar, copiar o inspeccionar? En tal caso, en las economías de mercado el precio debería ser fijado por la oferta y la demanda.

Esto no tiene nada que ver con tecnología o con la mentada ``neutralidad tecnológica". Tiene que ver con conveniencia económica y con los derechos de los consumidores y fundamentalmente con terminar con mercados monopólicos.

Donde no hay libre mercado, como en el software de oficina, sino monopolio, ¿cómo puede decidir el mercado? Dejar que ``el mercado decida" es trabajar para Microsoft, ya que no hay otra decisión posible. En estos casos el papel del estado es construir mercados libres y terminar con los monopolios.

En muchos campos del software existe un monopolio y el único proveedor fija sus condiciones bajo contratos de adhesión. Hoy comienzan a existir alternativas; entonces los gobiernos empiezan a equilibrar el mercado y pueden fijar sus propias condiciones. Son los proveedores los que deben entonces cumplir los requisitos de los estados.

Así es razonable que un gobierno pida que el software que adquiera pueda ser usado e instalado libremente en cualquier computadora de su propiedad. Corresponde al proveedor ofertar un precio para tal requerimiento. Es razonable que el gobierno pida acceso y derecho a recompilar y modificar el código fuente. Son los distintos proveedores quienes ofertarán el precio justo para este requerimiento.

Lo que sucede hoy con el software es que los usuarios han ganado derechos con el comienzo del fin del monopolio. A partir del software libre, nos empezamos a imaginar condiciones de libre mercado y la competencia permitirá bajar los precios y hacer que los proveedores deban entregar más para permanecer en el mercado.

Que los gobiernos fijen condiciones que amplíen sus derechos, con relación al software, sólo hace más competitivo y transparente al mercado, ayuda a terminar con monopolios [Malone:RIP-05] y favorece la baja de precios. No se excluye a ningún proveedor, solamente se piden los requisitos que el software público debiera tener.

Votando tecnologías

Cada vez que una persona elige un producto [Saravia:GI-04], está votando en los mercados; cada vez que ejerce una opción tecnológica está votando por ella. Los administradores de redes de las universidades americanas votaron por Internet y con esa decisión la impusieron como red mundial global. Perdieron las elecciones Novell con IPX, IBM con SNA, Microsoft con NetBIOS, etc.

Hoy es la hora del software, cada voto cuenta, cada elección define la batalla por la libertad del software.

Es también la hora de compartir archivos, cada nuevo desarrollo que permite compartir música y ``contenido" inclina más la balanza hacia un mundo mejor, más próspero [Rehermann:NMP] y con más libertad.

Así tendremos más prosperidad intelectual en un mundo de conocimiento libre [Saravia:REC-04,Saravia:DDS-03,Saravia:EI-03] que en otro cerrado y privativo.

Gobiernos e Internet

Subsecciones

Motivos

> **La cuestión no es darle computadoras e Internet a los pobres, la cuestión es erradicar la pobreza.**

> **Informatizar la pobreza con subsidios gubernamentales y fondos de ``igualdad digital'' de la mano de Intel, Microsoft y las compañías telefónicas es un excelente negocio.**

En el marco de la Cumbre Mundial de la Sociedad de la Información (CMSI/WSIS) se debate con especial fuerza la cuestión del gobierno/gobernanza de Internet. Este documento analiza críticamente algunas posiciones frecuentemente planteadas en el debate.

Ideas centrales:

- La cuestión no es quién o cómo se gobierna Internet, sino cuál es la agenda de los gobiernos territoriales y las agencias internacionales en los que estos participan, con relación a Internet.
- Se ha impuesto en las agencias internacionales la participación mediante un esquema que legitima a las empresas multinacionales y clasifica a las personas según su interés [Annan:QG]. Para pensar correctamente debemos hablar apropiadamente. No debemos usar el término ``interesados'' -multistakeholders- sino ``Ciudadanos de Internet''. Pongamos fin a la clasificación arbitraria y corporativa de personas entre civiles, empresarios y funcionarios. Pongamos fin a las entelequias organizativas, cada persona un ``voto''.

  Debemos dejar de usar la expresión ``Gobierno de Internet'' y reemplazarla por ``Gobiernos e Internet''.
- The Internet interprets censorship as damage and routes around it.

  John Gilmore

- In the long run, Internet radio will succeed. Instant messaging systems will interoperate. Dumb companies will get smart or die. Stupid laws will be killed or replaced. But then, as John Maynard Keynes also famously said, ``In the long run, we're all dead."

  ...

  Internet no es una cosa, es un acuerdo para transportar mensajes:

  - nadie es su dueño;
  - tod@s la pueden usar;
  - tod@s la pueden mejorar.

  A World of Ends.

- Internet es ``la" red nerviosa que vincula las sociedades del conocimiento, y sus personas y organizaciones. Mediante ella se genera un nuevo territorio virtual, un nuevo planeta, donde se radican infinidad de construcciones de sus ciudadanos o seres pensantes.

Internet, ¿es gobernable?

Gobierno, gobernanza, administración, control, o legislación son cuestiones similares para Internet. Esas palabras pueden tener significados diferentes en otros ámbitos. Internet es tan ajena a estas cuestiones que las diferencias entre las palabras anteriores no son significativas. Normalmente se las asocia con la toma de decisiones, en general restrictivas y limitantes y la cuestión fundamental a considerar es: ¿qué debe ser gobernado?, ¿quiénes lo hacen? y ¿cómo imponen y ejercen sus decisiones? Veremos que, en nuestro objeto de estudio, Internet, el concepto de gobierno y similares no se aplica.

¿Qué es Internet? Internet es un acuerdo. Un protocolo para conectar redes, que las partes deben respetar para poder comunicarse. Internet ha sido diseñada como una red dispersa. Sus nodos y enlaces no requieren una infraestructura central y tienen miríadas de ``propietarios", o decisores: sus usuarios -ciudadanos-, que deciden sobre sus equipos, software y conexiones. Internet representa un esfuerzo colaborativo, donde cada uno debe costear su conexión a la misma, y luego la inter-red lleva y trae sus ``mensajes" a cualquier otro nodo a cargo de los otros nodos. El diseño implica reciprocidad y aceptación de esta regla para poder participar.

No depende de los gobiernos. Si bien los gobiernos podrían regular ciertos aspectos vinculados a algunos tipos de enlaces, siempre los ``mensajes" podrán ser ruteados por otros, libres de la regulación legal o impuesta por los estados o en todo caso empresas. En términos humanos, ¿cómo sería el acuerdo Internet?, muy simple. A cambio de participar y poder enviar y recibir mensajes, cada persona acuerda transmitir los mensajes que reciba que no estén destinados a ella. Así cuando Perico recibe un mensaje para Juan, si lo conoce se lo lleva, si no lo conoce se lo da al amigo que tenga más posibilidades de conocer a Juan -que esté más cerca en algún sentido-, por ejemplo Pedro. Y Pedro hace lo mismo, y así sucesivamente hasta que llegue a Juan. No importa si usan teléfonos, correo, la radio, la televisión,

palomas mensajeras, satélite, boca a boca, email, mensajeros, etc. No importa si el mensaje tiene dirección de origen, no importa si el mensaje es un texto, un programa, una foto, una canción, si está encriptado, etc.

Hay muy pocas cosas a decidir en Internet. No es gobernable, como un país, o eventualmente el planeta, porque no hay cosas que decidir en forma conjunta, cada nodo toma sus decisiones.

Por otra parte los ciudadanos del mundo no delegaron a sus gobiernos nacionales autoridad constitucional, legal o consuetudinaria sobre sus mensajes en Internet o sobre Internet misma. Para que un gobierno democrático pueda atribuirse legitimidad de origen sobre su porción de Internet (y costaría mucho definir tal cosa) debería hacer un plebiscito. Es difícil ver, por otro lado, el interés social de este proceso. Por un lado los ciudadanos virtuales -usuarios- no tienen un peso proporcional al que les daría la representatividad de su gobierno. Es más, el concepto de peso relativo, esencial a la idea de democracia y gobierno le es ajeno.

Tengo cuentas de mails ubicadas en varios países, páginas web distribuidas en diferentes continentes, amigos por todos los países, uso servicios y servidores de mensajería que no sé dónde están, subo fotos a vaya a saber dónde. ¿Cómo puede el gobierno del país donde voto, o cualquier otro, decir que me representa en Internet? Soy ciudadano del universo sin gobierno de Internet, participo de Internet, a veces soy usuario de algunos servicios. Mi gobierno tiene poco y nada que ver en ello.

Más alejado de la realidad es el intento de las NNUU y la UIT de ``tener algo que ver con Internet", o con lo que ellos denominan su gobierno y mucho más con el esquema corporativo de gobiernos, empresas y civiles de la WSIS. ``Multistakeholders", interesados en reemplazo de ciudadanos. Intereses en lugar de ideales. Puedo tener una empresa, ser funcionario público, ciudadano, participar en varias ONG y movimientos, ¿cómo pueden dividirme y clasificarme? Soy ciudadano argentino, no me defino como ``interesado" en Argentina, soy ciudadano de Internet, por derecho propio, no por ser argentino.

Las decisiones en Internet las toman los dueños de sus nodos. Es decir sus ciudadanos o usuarios. Lo único que hay que consensuar es la distribución de las direcciones IP y las rutas, que con IPv6 dejarán de ser escasos y por ende no habrá necesidad de tal administración. Cada ciudadano puede definir su DNS, no es necesario optar por las zonas oficiales de la ICANN. ICANN, de hecho, parece más un mecanismo automático que una administración. Los ciudadanos virtuales acceden a él a través de formularios Web.

Internet es producto de la decisión de estas personas, que eligieron el protocolo IP (de los hackers) sobre los otros disponibles: IPX de Novell, DECNET de Digital, SNA de IBM, SMB de Microsoft, etc. Incluso triunfó sobre protocolos como el OSI, impuestos como obligación por el gobierno de EEUU y aprobados por organismos de telecomunicaciones internacionales que fijan estándares (¿un déjà vu de los intentos de controlar Internet?), un gobierno de Internet de la ONU o ITU, ¿no intentaría imponer un protocolo adecuado para las empresas de comunicaciones? Esa elección fue tomada sobre la base de la interoperabilidad, simpleza y la libertad de los protocolos y el software libre de referencia, que la construyó y se construyó en el proceso. Todos sus protocolos esenciales son libres, definidos por consenso técnico con la idea de la libertad. Idea ratificada por el voto de los usuarios -ciudadanos- que la

prefirieron. Es decir los técnicos crearon el protocolo IP, la gente lo eligió como sistema de comunicación global.

En estos días Microsoft intenta imponer su sistema ``Sender-ID'' para bloquear el spam, uno de los organismos frecuentemente sindicados como Gobierno de Internet, la IETF, habitual impulsor y emisor de estándares, ha rechazado este protocolo. ¿Qué hará Microsoft? Intentará imponerlo. ¿Existe un gobierno? ¿Quiénes resolverán esta cuestión? La resolverán los administradores de los principales gestores de correo. ¿Podría la gente hacer algo? Sí, usar gestores que no acepten el sistema. ¿Tendrá éxito Microsoft? Se aceptan apuestas.

Supongamos que se instalase una agencia de las NNUU a cargo del ``Gobierno de Internet'', con ``plenos poderes''. ¿Cómo obligaría a Microsoft? ¿Mandaría funcionarios de Interpol a los servidores a cambiar su configuración? ¿Mediante qué tribunal? ¿Con una multa? Hemos visto en decenas de casos que las medidas de los tribunales son lentas y no efectivas en estas cuestiones.

Internet es el resultado de una competencia darwiniana de normas técnicas y estructurales de redes. Es un universo evolutivo nuevo por derecho propio y en el que la gente define su propio espacio normativo. Vencedor entre otros mundos paralelos posibles. Internet se define al adaptar las nuevas tecnologías a lo que el ser humano y sus sociedades pueden aceptar y disfrutar. Tiene sus genes (código IP) y un cerebro automático determinado por los genes, las posibles decisiones a tomar son pocas e irrelevantes. Es producto de aplicar nuevas ideas al filtro de las sociedades humanas. En este sentido Internet es ``la más humana de las redes posibles''. Intentar plantear un control legal, ideológico controlante, y/o interesado externo al producto de un exquisito equilibrio de relaciones humanas y tecnología es equivocar el camino, porque surgirá otra red que será rápidamente adoptada por los humanos. Cada vez que un ciudadano virtual ejecuta un programa, lo elige y vota por él, que la gente no sea consciente de ello, es tan poco importante como que los votantes de Bush supieran que este iba a empezar una guerra en Irak.

En este proceso el gobierno estuvo ausente, y eso es razonable y debe continuar así. Las empresas globales estuvieron opuestas pero tuvieron que construirlo. El modelo de costo fijo en vez de uno operado por distancia y tiempo les arruina su negocio.

Los seres humanos no podemos, al menos hasta ahora, cambiar el valor del número Pi, en nuestro universo, pero podemos crear muchos universos virtuales, cada uno con protocolos diferentes, e intentar conectarlos. Cuando hagamos esto veremos que la forma que la gente elige para ello, es la más simple, la del diseño de IP planteado en ``World of Ends''. El protocolo que naturalmente aparece es el Pi de la simplicidad y de la libertad: IP. Ningún intento de gobiernos u organizaciones podrá convencer a la gente para que se conecte a una red global sin las libertades del protocolo IP. No podrán cuadrar el círculo de Internet. Todavía ninguna red de control o grupo o red conspirativa o secreta pudo asumir esta totalidad, como muchos dicen que ha sucedido en las democracias estatales. Los intentos de establecer un ``gobierno'' de Internet impulsados desde distintos frentes podrían estar centrados en el accionar de estas redes jerárquicas y de control por círculos ascendentes, pero están destinados al fracaso.

Internet es y debe seguir siendo una red universal, redundante, descentralizada, con ciudadanos con la posibilidad de anonimato o identidad virtual, como prefieran. Habrá servicios que los gobiernos darán a sus ciudadanos mediante Internet, pero el ``gobiernode la misma'' no es un servicio útil, ni siquiera definible.

Internet es un nuevo espacio mundial no sujeto a fronteras nacionales ni a corporaciones globales, sino a la voluntad dispersa y en votación continua de sus ciudadanos que deciden qué protocolo, redes, hardware, y software utilizan para sus mensajes.

No se ata a la soberanía de los estados y regiones y plantea un nuevo esquema de relaciones humanas que sólo rige en su interior. Podemos construir muchos mundos en su interior, o incluso intentar crear otra red con otras normas.

Internet no es gobernable, es un nuevo y bravo universo que merece ser explorado y vivido. Cada uno de los ``países'' que se constituyan en su interior tiene libertad para fijar sus normas, y los humanos tendremos el derecho de ser sus ciudadanos o no.

Deberes de los gobiernos en relación a Internet

Los gobiernos tienen una agenda de Internet: deberán regular los enlaces y la última milla, garantizar acceso mediante la eliminación de la pobreza, establecer penas por algunas conductas, garantizar la libertad del conocimiento como derecho humano, pero no gobernar Internet. Tienen tareas, pero no control. Deben trabajar por socializar la red y promover su buen uso.

Cada país tiene su legislación penal, y debe crear ámbitos para coordinarlas en cuanto a delitos y contravenciones que puedan cruzar las fronteras nacionales usando como medio a Internet (spam, lectura de mensajes privados, terrorismo de estado y espionaje a las personas por los estados, etc.)

Cada país participa de ámbitos internacionales donde se acuerdan las cuestiones vinculadas a los derechos de autor, las patentes y las marcas; deben actuar en esos ámbitos para adecuar estas normativas a la disminución de los costos e inversiones necesarias para difundir el conocimiento, liberando las restricciones de una época donde eran necesarias, disminuyendo los cuerpos represores del estado para evitar que nuestros ciudadanos compartan sus creaciones.

La cuestión es qué hacen los gobiernos nacionales o el ``mundial'' con Internet y no cómo se gobierna Internet: qué responsabilidades asumen, qué servicios prestan, cómo colaboran. Hay, pues, que dejar de hablar del gobierno de Internet y empezar a hablar de la agenda de los gobiernos en relación a Internet. El núcleo del problema

Luego de escribir el texto original busqué otros trabajos similares. Entre lo mucho de interés que encontré destaco el trabajo ``A World of Ends''. Como está publicado con una licencia libre, lo he incorporado con varias modificaciones y agregados a este trabajo.

1. Internet es una red de redes.

Es un medio concebido para vincular entre sí diferentes redes preexistentes: informáticas, de televisión, radio, telefónicas, etc., de la forma más simple, redundante, transparente y descentralizada posible. No es su cableado, no son sus computadoras, no es el contenido que fluye en su interior. Es realmente un medio para conectar y permitir que distintas redes puedan intercambiar mensajes. Es una red de redes.

La forma más simple, económica y segura de llevar un mensaje desde un punto a otro.

Internet, siendo sólo un mecanismo abierto que lleva mensajes desde un origen a un destino, construyó a su alrededor una entelequia social, política y económica.

Se aplican a Internet las cuestiones vinculadas a los derechos humanos relativos al conocimiento y la libertad de expresión.

2. Internet no es una cosa, es solamente un acuerdo o protocolo.

   Un protocolo es un acuerdo que establece cómo se trabaja en conjunto. Justamente Internet es el conjunto de redes interconectadas respetando el denominado ``Protocolo de Internet" o IP. Un protocolo que establece cómo hace un ``nodo" para intercambiar mensajes con otro.

   Internet no es sólo su foto actual. La red de redes podría haberse construido de muchas formas y sin duda su naturaleza irá cambiando con el tiempo. Los protocolos que hoy se utilizan son el resultado del acuerdo y la elección de sus ciudadanos de entre muchos posibles. Al aparecer la tecnología para hacerlo, de alguna forma se iban a interconectar todas las redes; lo interesante es que la sociedad decidió hacerlo de esta forma libre y no de otra. Cada nuevo protocolo propuesto, más específico que el IP, como TCP, DNS, HTTP, HTML, DHCP, LDAP, Kerberos, etc., es aceptado o no por grupos de personas en diferentes porciones de Internet; algunos fenecen, otros resultan universales. Algunos son propuestos por IETF, W3, DMTF; incluso ICANN habilita números de puertos. Otros son propuestos por empresas, como Sender-ID para correo, Messenger o ICQ. Pero quien decide, quien realmente gobierna Internet, es cada persona, cada red, que decide si los usa o no. Obviamente, la interoperabilidad de los protocolos, y para ello que sean públicos o libres y no estén sujetos a patentes y copyrights, ayuda a su adopción. Es una discusión donde los derechos de los consumidores priman sobre la ganancia empresarial.

   El protocolo no especifica qué puede o no hacer la gente con la red, qué puede construir en sus bordes, qué mensajes puede transmitir. Internet no exige identificarse, tampoco lo prohíbe. Puedes conectar tu computadora, tu celular, tu heladera, sólo tienes que estar de acuerdo con usar el protocolo IP. Puedes intentar cambiar este acuerdo y armar con otras personas una red con otro protocolo, si logras que muchos se te unan y finalmente si todas las redes del mundo te siguen, habrás cambiado el protocolo de Internet. No es una cuestión de votar a través de algún sistema de gobierno, o de que alguien lo imponga, cada uno se comunica con quien quiere, con el protocolo que quiere. Esa es la regla.

Hoy el protocolo más usado es IP. Son decisiones INDIVIDUALES, no hay una cosa sobre la que decidir. Cada uno decide sobre lo suyo. No hay lugar para GOBIERNOS.

3. Internet es estúpida. El sistema telefónico es inteligente, sabe mucho. Sabe quién te llama, desde dónde, cuánto cuesta la llamada, cuánto dura, etc. Provee muchos servicios: identidad del que llama, llamada en espera, etc.

   Internet es estúpida. A propósito. Sus diseñadores tuvieron la genialidad de darse cuenta de que la red que incluya a todas debiera ser la que menos supiera de las características especiales de cada integrante.

   No sabe de identidades, permisos, prioridades, etc. Sólo sabe que un mensaje debe moverse de un nodo a otro.

   Hay muy buenas razones para ello. La estupidez es un buen diseño. Es robusto: Si un ruteador falla, los mensajes irán por otro lado. Por su estupidez es fácil añadir nuevos dispositivos: cámaras, grabadoras, teléfonos, etc., que viven en sus nodos. Por ello crece en todas direcciones.

4. Añadir inteligencia y capacidades a Internet baja su potencial. Si uno optimiza una red para una aplicación, la desoptimiza para otra. Si se prioriza el tráfico de vídeo, las otras aplicaciones deben esperar.

   Algo simple para todos sería transformado en algo complejo para un solo propósito. No sería más Internet.

5. Lo importante de Internet está en sus extremos o nodos. Si Internet fuese inteligente, sus diseñadores hubiesen anticipado el potencial de los motores de búsqueda y hubiesen puesto la búsqueda como un servicio de la red misma. Como fueron inteligentes, la red es estúpida. La búsqueda la puede ofrecer cualquier nodo, de los millones conectados. Como a la gente le interesa encontrar cosas, los servicios de búsqueda prosperaron y compiten produciendo innovación, generando trabajo y dando opciones a la gente.

   Como la Internet es tan estúpida y sólo mueve mensajes, los innovadores pueden imaginar cientos de servicios y aplicaciones contando con que Internet moverá los mensajes que ellos necesiten.

   No necesitan el permiso de los administradores de Internet, o el gobierno de las Naciones Unidas o de quien sea. Tienes una idea. Ponla en tu nodo. Si a alguien le interesa la usará. Y cada vez que así sea la importancia de Internet crecerá.

   El potencial y la realidad de Internet reside en esta posibilidad de libertad.

6. La riqueza se mueve en los nodos. Si toda la riqueza está en los nodos, la conectividad de Internet es un ``commodity". Así debe ser. Todo intento de añadir funciones al protocolo básico de Internet debe ser resistido. No debe añadirse a los protocolos básicos la capacidad de identificar las personas, la

capacidad de priorizar tráfico, etc. Estos serán servicios separados y cada uno maximizará su valor en forma independiente.

7. ¿El fin del mundo?, no, un mundo de fines. En Internet todos se pueden conectar con todos. ¿Y qué hacen sus nodos? Todo lo que puede hacerse intercambiando mensajes. Todo y Tod@s. Es el producto de una arquitectura libre y estúpida.

   Como es un acuerdo, no pertenece a nadie. No a las compañías que proveen los enlaces, no a los ISP que dan conectividad a la gente, no a las compañías que ofrecen hospedaje de sitios, no a las asociaciones industriales que creen que lo que hacemos les perjudica, no a los gobiernos sin importar cuán sinceramente crean que intentan llevar seguridad a su gente.

   Conectarse a Internet es acordar llevar el valor a tus nodos. Y algo interesante pasa, todos estamos conectados casi igualitariamente. No importa la distancia. No importa nuestro poder. Somos cada uno un nodo, simplemente uno más. Estas barreras caen por primera vez en la historia de la humanidad.

8. Tres virtudes de Internet atadas a su naturaleza.

   1. nadie es su dueño.

      1. No es una cosa, no puede ser apropiada en forma exclusiva. Es un acuerdo. No sólo está en el dominio público, es de dominio público.
      2. Es un recurso confiable. Funciona, no nos obliga a hacer upgrade, no aumenta su precio cuando ya lo hemos comprado, ni puede ser comprado por un competidor.
      3. No hay problemas de que unas partes funcionen con un proveedor y otras con otros, o con una empresa sí y con otra no.
      4. El mantenimiento está distribuido entre sus participantes. No está concentrado en las manos de una empresa que puede quebrar.
   2. tod@s la pueden usar. Está pensada para incluir a todos en el planeta. Habrá que resolver problemas de pobreza o geografía, pero nada impide que solucionados los problemas estructurales de la humanidad, todos la puedan usar. Para muchos de nosotros es un recurso natural.
   3. tod@s la pueden mejorar. Todos pueden contribuir y hacerla un lugar mejor, contribuyendo con algo. Es muy difícil empeorarla.

      Hay dos formas: poner servicios en algún nodo o inventar un nuevo protocolo que permita nuevos servicios. Para que el protocolo prospere debe ser abierto, libre y sin dueños, para todos.

9. Si es tan simple, ¿por qué tanta confusión? Porque las tres virtudes son la antítesis de cómo los gobiernos y corporaciones ven al mundo.

   ◦ Nadie es dueño: las corporaciones son definidas por lo que poseen. Los gobiernos, por lo que controlan.

- Todos pueden usarla: Para los negocios, vender significa transferir derechos de uso exclusivos. En los gobiernos, hacer leyes implica imponer restricciones a la gente.
- Cualquiera puede mejorarla: En los negocios y los gobiernos, existen roles y jerarquías. Cada uno tiene una tarea.

Los gobiernos y las empresas por su naturaleza están predispuestos a malinterpretar Internet. Por otro lado, las grandes corporaciones preferirían que sigamos pensando que Internet es sólo una televisión lenta.

10. Errores que debemos dejar de cometer
    - Las compañías que distribuyen contenido en formas que el mercado no desea (industria musical), pueden darse cuenta de que la información no es como la materia. No queremos que impidan que copiemos información. ¿Por qué no nos dan razones para preferir comprar nuestra música de ustedes?
    - Los gobernantes que confunden Internet con sus contenidos, se darán cuenta de que jugar con el núcleo de Internet es bajar su valor.
    - Tenemos un sistema que transporta mensajes igualitariamente, sin censura gubernamental o de la industria: es la más poderosa fuerza surgida en la historia para profundizar la democracia y la libertad.
    - Las compañías telefónicas se darán cuenta de que la inter-red estúpida fagocitará su negocio de redes inteligentes y que su futuro es ofrecer la última milla y los enlaces troncales de Internet. Ahorrarán billones si evitan pelear por lo inevitable.
    - Aquellos que censuran las ideas se darán cuenta de que Internet no discrimina un mensaje ``bueno'' de uno ``malo''. La censura no puede ocurrir en el interior de Internet, podrá darse en los nodos, cuando estos lo deseen. Sólo quienes tengan el deseo de aplicarse censura a sí mismos lo harán.
    - Las compañías que creen que pueden obligarnos a leer o escuchar sus mensajes se darán cuenta que podemos evitarlas. Internet no es como la radio o la televisión. Uno elige lo que hace.

La verdadera pelea, ¿qué hacer?

Si el gobierno de Internet es imposible, si podemos dejar a todas esas organizaciones y gobiernos discutiendo y dando vueltas sobre indeseables utopías irrealizables, ¿para qué preocuparnos?

Si en Internet las decisiones las toma la gente, lo importante es orientar a la gente para que tome buenas decisiones.

Lo esencial es el convencimiento, no el control. Ni siquiera la pueden controlar las multinacionales de la música o los gobiernos que censuran o espían.

Entonces debemos trabajar para convencer. No sólo debemos discutir sobre qué o cómo controlar o gobernar; sí debemos generar mejores alternativas.

Además de luchar contra el copyright del software, patentes, DMCA, ALCA, la informática traidora y todo lo demás, debemos generar alternativas como primera tarea.

Internet surgió y dio lugar al movimiento de Software Libre, sin él no existiría la opción de la libertad, ni Internet tal como la conocemos.

Si bien ya nadie jamás va a poder gobernar/controlar la informática, debemos trabajar para que la gente pueda seguir eligiendo.

El riesgo está en la privatización del software, en el establecimiento de patentes, en el control de las PCs con TCG, y en la prohibición de compartir en Internet, y con ello un mundo cerrado y opaco. Debemos entonces dar la única pelea que importa, que la gente elija dejar de usar software propietario. Si se impone el software privativo, puede cambiar nuestra historia. Si la privatización del conocimiento gana, perdemos. Es la tarea defensiva que hacemos. El que controla lo que se pone en las PC controla el mundo. Por ende nadie debe controlar nada. Todo debe ser libre.

Es la forma de distinguir amigos de adversarios. Quien está en contra de liberar el conocimiento en forma consciente y con conocimiento de causa, es un adversario. Debemos buscar aliados en la temática del gobierno de Internet entre quienes apoyan al Software Libre. Toda la temática de las TICs está cruzada por el problema de la libertad. Hay una sola contradicción: la libertad del conocimiento, o la propiedad intelectual, y esta disyuntiva se aplica a campos diversos, como el espectro radial. Todas las otras discusiones son funcionales y derivan de esta. Saber vincular cada disyuntiva a esta temática es la llave para clarificar las ideas en la cuestión TIC.

La decisión sobre el futuro de la humanidad está allí, en la mente de cada persona, en sus discos rígidos y sus futuras decisiones. Son el verdadero gobierno de Internet y del futuro de las sociedades humanas. Trabajemos porque sigan siéndolo. Los dados están echados, la obra se está rodando. Somos parte del juego y tenemos muy poca capacidad de control o gobierno (y está bien que así sea).

En el largo plazo la victoria seguramente será nuestra, pero también es cierto que estaremos muertos; nuestro trabajo es evitar el sufrimiento intermedio adelantando el proceso de cambio.

Referencias y apuntes adicionales

Segundo Manifiesto de Hipatia, derechos humanos y libertad del conocimiento [Hipatia:SM-04]. Documentos en Hipatia: `http://docs.hipatia.info/`. Manifiesto de Hipatia [Saravia:MH-01]. Posición institucional de APC [APC:PI]. Internet, general [WIP:TI,Gillmor:CWC-02,Lessig:OAF].

Centro del dominio público [CPD:SW]. Música [Orlowski:TSM-04,EGA:OPF-05,Intellpuke:SCR-05]. ICANN Montevideo [Bergara:NDM-04]. Contacto [Sagan:C-85]. Manifiesto Cluetrain [Locke:TCM-99]. Artículos en NYT [NYT:A-99]. Red estúpida [Isenberg:RSN-97,Isenberg:TPB]. Internet [Wikipedia:IP]. Varios [TBCIS:SW-00,TBCIS:JPB-00,TBCIS:IS99-00,Zittrain:BCA-00,McCullagh:SUN-04,TIM,HIP:TIM-97,Jonsson:N

La sección ``El núcleo del problema'' deriva del trabajo a ``World of Ends'' [Searls:WOE-03,Saltzer:ETE-84].

La primera versión de este trabajo es [Saravia:GIV-04].

Redes

- ¿Qué es una red? Un conjunto de nodos y enlaces. Los enlaces conectan a los nodos. Los nodos emiten mensajes para otros nodos. Los enlaces los trasmiten.
- ¿Cómo se conecta una red con otra? Compartiendo un nodo entre ambas. Ese nodo actúa como enrutador. Puede haber varios enrutadores entre dos redes. Hay muchos caminos alternativos para que un mensaje viaje de una red a otra.
- ¿Qué es Internet? Un acuerdo para intercambiar mensajes, un protocolo de comunicaciones. Se acuerda dar un número a cada nodo y se establece un mecanismo para los ruteadores. Eso es todo. Cada mensaje sabe el número del nodo al que debe llegar. Cada nodo conoce al menos un ruteador para acercar el mensaje a su nodo destino.

Meritocracia

> El uso del término suele ser muy cuestionado en ciertos ámbitos, pero las realidades no pueden ser escondidas, los programas y protocolos que mueven Internet fueron desarrollados por las mentes más brillantes -hackers- de los últimos 50 años en un proceso de ingeniería social y realimentación muy fuerte. Es parte de la mayor aventura colectiva de la humanidad, la creación de un edificio de software, que satisfaga nuestras necesidades y esté libremente disponible para todos y todas.``Software doesn't come from lists, but from people opening up their emacsitors or viitors and actually hacking [Welton:C-00]."

Represión digital o judicializando el compartir

> Las compañías musicales no logran, mediante la represión, parar a la gente que comparte [Orlowski:TSM-04,Intellpuke:SCR-05,EGA:OPF-05].

ICANN

> Es un MECANISMO, bastante automatizado para otorgar números y nombres. Junto con IETF y otros órganos, generan protocolos y normas que son sólo propositivos; la gente los acepta o no.
>
> Ninguna de estas tareas puede ser vista como un gobierno, ni siquiera como administración.

Definir una zona no se hace mediante un mecanismo, es cierto, pero en estos casos la gente puede o no aceptar esta propuesta. En realidad esa decisión la toma cada uno cuando pone en sus DNS los servidores centrales a los que apunta. Si bien poca gente los cambia, podrían hacerlo; entonces la decisión la toma quien distribuye el soft y quien los preconfigura con determinados servidores raíz.

Bastaría que alguien creara una nueva zona, y preconfigure las computadoras para leerla.

Commons

O bienes públicos. Cuando se habla de ``commons" en general se habla de cosas comunitarias, es decir que tienen un dominio comunitario, o sea que son cosas sobre las cuales se toman decisiones, y esas decisiones las toma una comunidad, por la vía que tenga establecida. Puede ser una persona jurídica ``estrecha y legal" del tipo ``occidental y cristiano" o puede ser una ``comunidad abierta" del tipo ``usuarios y desarrolladores del soft libre" o ``pueblos originarios"

Habitualmente este tipo de decisiones se vinculan con la propiedad, aunque no tiene por qué ser así.

De cualquier forma la cuestión de fondo es esa: hay alguien ``individuo o comunidad" que debe tomar decisiones sobre el ``bien" o la ``cosa". Para Internet no. En principio no creo que sea un ``bien", mucho menos propiedad, pero la idea es que ningún individuo o colectividad organizada tenga posibilidades de decidir. Cualquiera puede hacer lo que quiera con él.

Ver la clasificación de los bienes en [Saravia:OLC-05]: libres o escasos, entre los escasos: públicos, club, commons y privados.

Stanford, África e IPs

Que digan que Stanford tiene más IPs que toda África es un comentario que asusta, y que está basado en el hecho, cruel pero no menos real de que probablemente haya más computadoras en Stanford que en África. Y la solución a esto no pasa por regular Internet, sino por eliminar la pobreza y la explotación en África y en todo el planeta.

Como siempre **la cuestión no es darles IPs o computadoras a los pobres, la cuestión es erradicar la pobreza. Ya sabrá una persona con un nivel de recursos digno, educación y libertad si desea computadoras o IPs, o si requiere otras cosas.**

Informatizar la pobreza con subsidios gubernamentales de la mano de Intel, Microsoft y las compañías telefónicas es un excelente negocio.

Transparencia, anonimato

La cuestión importante de Internet, que garantiza su transparencia, anonimato, etc., está en su estructura técnica constitutiva. Ya se probó que redes con otras estructuras constitutivas

son desplazadas por un sistema libre al estilo IP. La potencia de Internet está en la combinación de libertad de uso y de creación. Es motor del desarrollo de Software Libre y su principal usuario. Define el espacio donde todos se miran y donde se establece la competencia, el campo de batalla de las nuevas tecnologías.

Normas técnicas de Internet

El mecanismo actual de definición de normas basado en RFC es correcto, no hay nada que cambiar allí. La meritocracia técnica que perfecciona las reglas técnicas ha demostrado su valor como generador de propuestas que son aceptadas por la sociedad. Cuando las normas técnicas no son útiles a la humanidad, como pasa muchas veces, la sociedad no las adopta.

Diccionario conceptual del conocimiento libre

Este trabajo toma como base el informe Hipatia para ICA [Saravia:SLA-03].

Se sugiere consultar además el Diccionario Hacker [Raymond:DH], la Wikipedia [Wikipedia:SW], el diccionario técnico de Lucas-Giait [Giait:DII], sin olvidar a Google [Google:SW].

Las definiciones de las versiones HTML se encuentran rotuladas para un simple referenciamiento desde otras páginas o documentos.

Subsecciones

Software y derechos de usuarios y programadores

SOFT

Software

En este diccionario se describen distintas formas de distribuirlo, exponiendo los derechos que los usuarios y programadores tienen con respecto a los mismos. No siempre son categorías excluyentes ni totalizadoras. Muchas veces se instrumentan mediante licenciamientos basados en copyright o contratos privados de sus derecho-habientes con los usuarios o desarrolladores. El copyright -legislación pública penal- actúa mediante mecanismos de restricción impuestos por las legislaciones estatales y tratados internacionales, que en algunos casos se refuerza con leyes tipo DMCA y tratados internacionales.

GNU

es un acrónimo recursivo, una palabra autorreferencial que significa ``GNU no es Unix".

Ver SISGNU y SGNU.

SL

Software Libre

Un software es libre si sus usuarios y desarrolladores gozan de todas estas libertades:

- 0.- (ejecutar) La libertad de usar el programa, con cualquier propósito. (analizar la diferencia entre usar y ejecutar: hay otras formas de usar un programa además de ejecutarlo)
- 1.- (inspeccionar) La libertad de estudiar cómo funciona el programa y adaptarlo a sus necesidades. El acceso al código fuente es una condición previa para esto.
- 2.- (redistribuir) La libertad de distribuir copias, con lo que se puede ayudar a otros.
- 3.- (modificar y redistribuir las modificaciones) La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. El acceso al código fuente es un requisito previo para esto.

Estas libertades se corresponden con las libertades humanas fundamentales de conocer y aprender la tecnología que se usa, comunicar el conocimiento del que se dispone y ser solidario con los demás [Hipatia:SM-04].

En otros idiomas:

Francés:
logiciel libre

Alemán:
freie Software

Ruso:
svobodny programy

Chino:
zi4you2 ruan3jian4

Japonés:
jiyuu [na] sofuto

Esperanto:
libera programaro

Sueco:
fri programvara

Holandés:
vrije software

Inglés:
free software. Para evitar la ambigüedad de libre con gratis del idioma inglés se empieza a hablar de ``Libre Software" aún en esa lengua. Con el mismo fin se propuso el uso de ``Open source software", que luego se convirtió en una identificación política dentro del movimiento que identifica a quienes prefieren promover el Software Libre desde el punto de

vista del marketing y la conveniencia comercial. Esta corriente se nuclea alrededor de la OSI y es muy fuerte en Estados Unidos. A partir de ese momento muchos identifican el uso del término ``Free Software" en inglés con los postulados de carácter ético de la ``Free Software Foundation". También se utiliza FOSS o FLOSS como forma de referirse al Software Libre en inglés evitando identificarse con la corriente ética o la corriente marketinera.

Portugués:

software livre

Danés:

fri software

Italiano:

software libero

Hindi:

swatantra software

Catalán:

software lliure

Software Libre significa la libertad de distribuir copias, sea con o sin modificaciones, en forma gratuita o cobrando un monto por la distribución, a cualquiera y en cualquier lugar.

Software Libre significa, entre otras cosas, que no hay que pedir o pagar permisos.

Software Libre significa la libertad de hacer modificaciones y utilizarlas de manera privada en trabajo u ocio, sin siquiera tener que anunciar que dichas modificaciones fueron realizadas. Cuando se publican y difunden los cambios no hace falta avisar a nadie en particular ni de ninguna manera en particular.

Software Libre no significa la obligación de distribuir el software o sus modificaciones; es un derecho. Incluso los derecho-habientes pueden distribuir a unos en forma libre y a otros en forma privativa.

El software libre debe incluir tanto las formas binarias o ejecutables del programa (si existen) como su código fuente, sean o no versiones modificadas. Distribuir programas de modo ejecutable es necesario para que los programas libres sean fáciles de instalar.

Para que las libertades de hacer modificaciones y de publicar versiones mejoradas tengan sentido, se debe tener acceso al código fuente del programa. Por lo tanto, la posibilidad de acceder al código fuente es una condición necesaria para el Software Libre.

Para que estas libertades sean reales, deben ser irrevocables mientras no se haga nada incorrecto; si el desarrollador del software tiene el poder de revocar la licencia aunque no haya motivos, el software no es libre.

Son aceptables, sin embargo, ciertos tipos de reglas sobre la manera de distribuir Software Libre, mientras no entren en conflicto con las libertades centrales. Por ejemplo, copyleft ``izquierdo de copia" (expresado muy simplemente) es la regla que implica que, cuando se redistribuya el programa, no se pueden agregar restricciones para denegar a otras personas las libertades centrales. Esta regla no entra en conflicto con las libertades centrales, sino que más bien las protege.

Puede que se haya pagado por obtener copias de Software Libre, o que se haya obtenido sin ningún costo, pero independientemente de la forma de adquisición u obtención de las copias, siempre se mantiene la libertad de copiar y modificar el software e incluso vender nuevas copias.

``Software Libre" no significa ``no comercial". Un programa libre debe estar disponible para uso comercial, desarrollo comercial y distribución comercial. El desarrollo comercial del Software Libre ya es muy común; el software comercial libre es muy importante.

Es aceptable que haya reglas acerca de cómo empaquetar una versión modificada, siempre que no bloqueen a consecuencia de ello la libertad de publicar versiones modificadas. Reglas como ``Si haces disponible el programa de esta manera, debes hacerlo disponible también de esta otra" pueden ser igualmente aceptables, bajo la misma condición. Este tipo de reglas permiten decidir si se publica o no el programa. También es aceptable que la licencia requiera que, si se distribuye una versión modificada y el desarrollador anterior solicita una copia, sea un deber enviársela [FSF:QC-98,FSF:DSL].

DESARROLLADORES

¿Quién escribe el Software Libre? [Mas:SLS-03]

La mejor forma de responder esta cuestión es clasificar los proyectos del mundo del Software Libre según su liderazgo. Esta clasificación no pretende ser exhaustiva pero podemos diferenciar claramente tres grupos de comunidades que han liderado los principales proyectos libres en los últimos años:

Proyectos vinculados a empresas, como Sun Microsystems, que mantiene OpenOffice.org; la fundación Mozilla, hasta hace poco directamente dependiente de America Online, que mantiene el proyecto Mozilla.org, o Ximian, que mantiene el sistema de correo electrónico Evolution. También hay proyectos, como Apache, que están representados por fundaciones y que reciben ayuda de empresas como IBM.

Proyectos que han sido desarrollados en universidades, algunas veces con financiación de empresas o del Gobierno. Uno de los más representativos es la familia de sistemas operativos BSD, en concreto NetBSD y FreeBSD, desarrollados en la Universidad de Berkeley, en California.

Proyectos liderados por grupos de voluntarios, como por ejemplo Debian, GNU o Abiword, que son desarrollados por voluntarios de todo el mundo. Estos proyectos son, sin duda, los más altruistas desde un punto de vista ético, ya que no siempre cuentan con apoyo financiero y muchas veces se construyen desde el puro voluntariado. La motivación principal de los colaboradores de este tipo de proyectos es pensar que el Software Libre es un progreso para la humanidad.

COPYLEFT

[FSF:QC-98]

Subversión del copyright [Varios:CSC]. Uso de la fuerza del sistema del copyright para lograr el objetivo contrario, como en el Judo.

El software protegido con copyleft es Software Libre cuyos términos de distribución no permiten a los redistribuidores agregar ninguna restricción adicional cuando éstos redistribuyen o modifican el software. Esto significa que cada copia del software, aun si ha sido modificado, debe ser Software Libre.

En el Proyecto GNU, se protege mediante copyleft casi todo el software que se escribe, porque el objetivo es dar a cada usuario las libertades que el término ``Software Libre'' implica.

Copyleft es un concepto general. Para proteger actualmente un programa con copyleft, se necesita usar un conjunto específico de términos de distribución. Hay muchas maneras posibles de escribir términos copyleft de distribución.

SLSCL

Software Libre no protegido con copyleft

El Software Libre no protegido con copyleft viene desde el autor con autorización para redistribuir y modificar así como para añadirle restricciones adicionales.

Si un programa es libre pero no protegido con copyleft, entonces algunas copias o versiones modificadas pueden no ser libres completamente. Una compañía de software puede compilar el programa, con o sin modificaciones, y distribuir el archivo ejecutable como un producto propietario de software.

El Sistema X Window ilustra esto. El Consorcio X libera X11 con términos de distribución que lo hacen Software Libre no protegido con copyleft. Se puede obtener una copia que tenga esos términos de distribución y es libre. Sin embargo, hay versiones no libres también, y hay estaciones de trabajo populares y tarjetas gráficas para PC para las cuales las versiones no libres son las únicas que funcionan.

SGPL

Software GPL

La GPL (General Public License/Licencia Pública General) de GNU es un conjunto específico de términos de distribución para proteger con copyleft a un programa. El Proyecto GNU la utiliza como los términos de distribución para la mayoría del software GNU

El sistema mundial de copyright se basa en acuerdos internacionales de reconocimientos, pero en cada país el alcance y restricciones impuestas por este sistema y otras normas, como las de protección al consumidor, pueden modificar la aplicabilidad real de cada término específico de la GPL, dependiendo, por ejemplo, de si el software ha sido transferido al usuario a título oneroso o no.

También puede cambiar en cada país el sentido que la legislación de copyright le da a los aportes o contribuciones de diferentes personas a un proyecto bajo la GPL. En todos los casos es conveniente asegurar que el código sumado a un proyecto GPL proviene de personas con derecho a hacerlo. Por ejemplo: si los programadores trabajan en relación de dependencia deberán contar con autorización de su empleador. Esto es un requisito muy importante.

La GPL constituye un excelente paraguas protector para el Software Libre en todo el mundo, pero algunas cláusulas pueden no ser válidas en todos los países y en todos los casos.

La GPL se distribuye exclusivamente en inglés, y los acuerdos internacionales garantizan su validez en todo el planeta, independientemente de donde se origine el programa o sus modificaciones. Las traducciones se distribuyen sólo a título aclaratorio.

SGNU

Software GNU

El Software GNU es software liberado bajo el auspicio del Proyecto GNU. La mayoría del software GNU está protegido con copyleft, pero no todo. Todo el software GNU es Software Libre.

Parte del software GNU es escrito por el personal de la Free Software Foundation, pero otra parte es aportado por voluntarios. La Fundación para el Software Libre es derecho-habiente de parte del software aportado; otra parte es de quienes lo aportaron o escribieron.

SISGNU

El sistema GNU

El sistema GNU es un sistema operativo libre, con estilo similar al Unix.

Lo bueno de la filosofía de diseño del Unix es que consiste en muchos programas independientes que cooperan entre sí. La FSF estuvo acumulando componentes para este sistema desde 1984; la primera liberación de prueba de un ``sistema GNU completo'' fue en 1996, utilizando el núcleo

Linux desarrollado por Linus Torvalds en Finlandia y liberado en 1991. Al conjunto se lo denomina GNU/Linux.

El sistema GNU incluye todo el software GNU, así como muchos otros paquetes tales como el Sistema X-Windows y TEX que no son software GNU.

Debido a que el propósito de GNU es ser libre, cada componente individual en el sistema GNU tiene que ser Software Libre. No todos tienen que estar protegidos con copyleft, y se puede incluir legalmente cualquier tipo de Software Libre si ayuda a alcanzar metas técnicas, como el Sistema X-Windows.

En el proyecto GNU, se utiliza ``copyleft'' para proteger de modo legal estas libertades para todos. Pero existe Software Libre sin ``copyleft'' desarrollado por otros proyectos. Existen razones importantes por las que es mejor usar copyleft, pero los programas son Software Libre sin necesariamente tener copyleft. En cualquier caso, se pueden usar.

A veces las normas de control de exportación de los gobiernos y acuerdos internacionales [Wassenaar:SW] junto a las sanciones mercantiles, pueden restringir la libertad de distribuir copias de los programas a nivel internacional. Los desarrolladores de software no tienen el poder de eliminar o sobrepasar estas restricciones, pero lo que pueden y deben hacer es rehusarse a imponerlas en las condiciones de uso del programa. De esta manera, las restricciones no afectarán a actividades y gente fuera de las jurisdicciones de los gobiernos que las impongan.

Si un programa es libre, entonces puede ser potencialmente incluido en un sistema operativo libre tal como GNU, o sistemas GNU/Linux que son libres.

Estas libertades pueden construirse legalmente de diversas formas. En el marco de las leyes de copyright, se creó una licencia específica, la General Public Licence (GPL) para instrumentar el concepto de copyleft.

Algunas personas consideran que estas libertades y la comunidad de desarrolladores y usuarios formada a su alrededor utilizan la GPL como un acuerdo constitutivo, y que el código así producido requiere, merece, o se beneficiaría con un encuadre en un marco legal especial diferente de las leyes de copyright. De hecho esta comunidad se ha caracterizado por garantizar sus acuerdos por el consenso más que por el uso de los tribunales, como medio de resolución de conflictos.

En ambos casos lo importante es hacer hincapié en la libertad y no en el esquema legal para su consecución, sea éste una licencia como la GPL u otro mecanismo legal.

En las referencias se listan diferentes licencias de cada categoría analizada [FSF:VLC,FSF:DSL,Stallman:CIP,FSF:CSL].

SOL

Sistemas Operativos Libres: GNU/Linux, BSD, Hurd

Distribuciones Debian, SuSE, Red Hat, Mandrake, Ututo, Ubuntu, etc.

GNU/Linux es el sistema operativo más conocido del mundo libre; otros son BSD y Hurd. Linux es un proyecto colaborativo llevado adelante por miles de desarrolladores en el Mundo [LinuxK:SW]. Hoy en día la mayor parte de los aportes al núcleo provienen de desarrolladores contratados por grandes empresas a tal efecto.

Existen distribuciones de GNU/Linux en CD-ROM o DVD, otras disponibles para bajar de la red; algunas comunitarias como Debian y otras comerciales como Red Hat y SuSE.

OSS

Open Source Software

Forma de denominar al Software Libre, en inglés, con mucho uso en EEUU [OSI:SW].

Existen al menos dos tendencias políticas principales en el mundo del Software Libre. Ambas coinciden en las prácticas, desarrollan y distribuyen el mismo software bajo los mismos esquemas legales. Las diferencias son filosóficas. El primero indica que la razón fundamental para usar Software Libre es ética y su referencia es la FSF (Free Software Foundation).

El segundo indica que la razón fundamental es la conveniencia. Impulsa el Software Libre en términos pragmáticos más que filosóficos. Sus principales impulsores son Eric Raymond y O'Reilly. Tienen su propia definición basada en las guías originales del proyecto Debian [OSI:OSD].

En inglés la OSI ha concebido una forma diferente de denominar al Software Libre para identificarse: OSS.

El argumento original para el cambio de nombre se refiere a que el término ``free'' en inglés es confuso, si bien la propuesta de abierto es todavía más confusa, pues referencia sólo a una de las cuatro libertades. La OSI (Open Source Initiative), al igual que la FSF, mantiene una lista de licencias aprobadas que cumplen con sus requisitos [OSI:AL]

La Open Source Initiative es una estrategia de marketing para el Software Libre. La sustancia no cambia, la actitud sí. En febrero de 1998, Netscape anunció el lanzamiento de su navegador como Software Libre. Un grupo de personas con base en Palo Alto y Silicon Valley, propuso empezar tal campaña a favor del Software Libre mediante el término ``Fuente Abierta''. El objetivo fue lograr la aceptación de compañías y capitalistas en el boom de la nueva economía. Se eligió dejar de lado las cuestiones de largo alcance como economía, ética y sociales, pensando que estas obstaculizaban la rápida aceptación por parte de las empresas. El foco estaba en las ventajas técnicas. Muchos hoy usan el término para referirse a software con condiciones intermedias entre Libre y Propietario.

En otros idiomas es mejor usar las traducciones de Software Libre, ya que no hay confusión con gratis. Últimamente se empieza en inglés a usar ``Libre Software''. También FOSS o FLOSS.

SA

Sistemas Abiertos

Como reacción a los sistemas cerrados, primero de IBM y luego de Microsoft, surgió la preocupación de permitir que las computadoras interoperen en base a estándares.

En general esta idea se materializó con servidores Unix y los estándares POSIX y han quedado completamente superadas por el movimiento de Software Libre que es la única vía real de satisfacer las necesidades expresadas en este movimiento.

SC

Software Comercial

El software comercial es distribuido por una entidad que tiene la intención de hacer dinero del uso del software. ``Comercial'' y ``propietario/privativo'' no son la misma cosa. La mayoría del software comercial es propietario, pero hay Software Libre comercial y hay software no libre no comercial.

Por ejemplo, Ada de GNU siempre se distribuye bajo los términos de la GPL de GNU y cada copia es Software Libre; pero los desarrolladores venden contratos de soporte. Cuando sus vendedores les hablan a los clientes potenciales, algunas veces el cliente dice ``Nos sentiríamos más seguros con un compilador comercial.'' Los vendedores responden, ``Ada de GNU es un compilador comercial; sólo que es Software Libre.''

SDP

Software de dominio público

Es software que no está protegido con copyright. Esto implica que terceros pueden apropiarse del mismo y realizar versiones modificadas que pueden ser o no libres.

Algunas veces la gente utiliza el término ``dominio público'' de una manera imprecisa para decir ``libre'' o ``disponible gratis''. Sin embargo, ``dominio público'' es un término legal y significa de manera precisa ``sin copyright''. Por claridad, se recomienda el uso de ``dominio público'' para ese significado solamente y el uso de otros términos para transmitir los otros significados.

SSL

Software semilibre

El software semilibre es privativo, pero tiene menos restricciones que el privativo clásico.

El software semilibre es mejor que el software propietario, pero aún plantea problemas y no se puede usar en un sistema operativo libre.

SG

Software gratuito, regalado, Freeware

Software que se distribuye sin costo. No es lo mismo ni es sinónimo de Software Libre.

Ese término específico significa que tiene precio cero. Software Libre es una cuestión de libertad y de derechos, no de precio.

Las copias de Software Libre frecuentemente están disponibles libres de cargo -por ejemplo, para ser descargadas vía FTP (File Transfer Protocol). Pero las copias de Software Libre también están disponibles por un precio en CD-ROM. Las copias de software propietario ocasionalmente están disponibles libres de cargo en promociones y algunos paquetes propietarios están normalmente disponibles sin cargo alguno para ciertos usuarios.

El término ``freeware" no tiene una definición clara aceptada, pero es usada comúnmente para paquetes que permiten la redistribución gratuita, pero no la modificación (y su código fuente no está disponible). Estos paquetes no son Software Libre, por lo tanto no es correcto usar la palabra ``freeware".

El término ``freeware" fue utilizado con frecuencia durante los años 80 para hacer referencia a programas sacados al mercado sólo como ejecutables, con el código fuente no disponible. Hoy no se tiene una definición particular aceptada.

Para otros idiomas, se debe evitar palabras tomadas del inglés como ``free software" o ``freeware".

Un programa distribuido como Software Libre no es un regalo. El uso del término regalar tiene el mismo problema que el concepto de ``software gratuito": expresan un atributo basado en el precio, no en la libertad.

SW

Shareware

El shareware es software con autorización para que la gente redistribuya copias, pero bajo la condición de que aquel que continúe usando la copia deberá pagar un cargo por licencia.

El shareware no es Software Libre:

1. En la mayoría del shareware, el código fuente no está disponible, de esta manera se limita el derecho básico de modificar el programa.
2. El shareware no viene con autorización para hacer una copia e instalarlo sin pagar una cantidad por licencia, ni aún para particulares involucrados en actividades sin ánimo de lucro.

(En la práctica, la gente a menudo hace caso omiso a los términos de distribución y lo hace de todas formas, pero las cláusulas no lo permiten).

SP

Software propietario, privativo o cerrado

El software propietario es software que no es libre ni semilibre. Su código suele estar oculto. Su uso, redistribución o modificación suele estar prohibida o requiere de autorización.

Los usuarios de este software no suelen poder acceder al código del mismo, por lo tanto se ven privados del conocimiento real de las utilidades y funciones que el software realiza. Los usuarios de software propietario suelen estar privados del derecho a compartirlo, modificarlo, adaptarlo o corregir sus errores. Los usuarios de software propietario se convierten en clientes cautivos de las empresas que los desarrollan y de sus servicios técnicos autorizados y pierden así sus derechos básicos como consumidores.

El software propietario se caracteriza por privar de derechos a sus usuarios, por lo tanto es correcto y mejor denominarlo privativo.

El software cerrado es aquel cuyo código no se puede inspeccionar.

El mismo software que se distribuye como libre puede ser distribuido por los derecho-habientes como privativo a personas particulares, generalmente por una fuerte compensación económica. Esto no presenta ningún problema en tanto y cuanto esté disponible universalmente el mismo software en forma libre.

CND

Contratos de no divulgación

Muchas veces los programadores y otros trabajadores se sujetan a contratos de no divulgación para poder programar. Los programadores acceden al código fuente del software, lo pueden usar para sus propios sistemas, pero no pueden redistribuir el código fuente o sus aportes al mismo.

Estos contratos exigen renunciar al derecho de compartir para poder programar.

LIBREELECCION

Al ser humano le fue dada la libertad de elección, o libre albedrío, según diversos y concurrentes pensamientos religiosos.

Esta libertad, asociada a una moral determinada, nos permite optar por el bien o el mal. O simplemente elegir diferentes caminos alternativos. Decidir dentro de ciertos márgenes.

¿Es la cuestión del tipo de licenciamiento del software una cuestión vinculada con la libertad de elección [Jorge:MLE-04]?

En primer lugar el software licenciado en forma privativa está diseñado para quitarle a los programadores libertades y con ello libertades de elección. Así un usuario de Windows no puede instalarlo en varias máquinas, o modificarlo, etc. Su espacio de opciones y alternativas se reduce.

Por otro lado el proyecto del Software Libre propone liberar todo el software. El camino para ello es desarrollar alternativas libres a cada programa propietario [FSF:HPG]. Si este proyecto triunfa, como consecuencia indirecta, no será sustentable el licenciamiento del software como privativo. Microsoft ha salido a plantear esto como un impedimento a su negocio. En realidad, en este caso las empresas podrán distribuir su software como libre adaptando su modelo de negocios. Hoy por hoy, seamos sinceros, hay un monopolio y lo ejerce Microsoft; en todo caso la existencia de este monopolio es el motivo por el que muchos otros empresarios están impedidos de competir y pierden su espacio de opciones.

Entonces el software libre ¿implica alguna contradicción con la libertad de elección?

Claramente la respuesta es no. No se reduce el espacio de opciones de nadie. Aún en caso de que no quede software privativo en la Tierra[10.1], nadie perdería opciones, algunos ganarían menos dinero, sería muy difícil construir monopolios. El software libre y su comunidad ha ampliado extraordinariamente el margen de opciones para la gente en cuanto a tecnologías. Está construyendo una alternativa y permitiendo salir de un monopolio. Hoy otro mundo no es sólo posible sino que está en construcción.

El software privativo en cambio reduce grandemente el espacio de opciones de desarrolladores y usuarios.

Por ello, lo que los militantes del movimiento afirmamos es que es malo usar software privativo.

Usarlo es justamente limitar la libertad de elección y votar por un mundo donde no existe libertad del conocimiento y por muchos otros motivos.

El software privativo es una lacra de la humanidad, hay que cambiar las condiciones que permitieron la existencia de licenciamientos que restrinjan la libertad de usuarios y programadores, que construyeron monopolios y que nos llenaron de virus.

Entonces consideramos que usar software libre es un deber MORAL o en realidad ÉTICO, pues no se basa en una idea religiosa adquirida por fe, sino una decisión fundada en razones que se basan en los intereses de la humanidad[10.2].

Pero, de forma similar a las religiones, existe para el movimiento algo BUENO: usar software libre, y algo MALO: usar privativo. Y en definitiva cada ser, usando su libre albedrío podría decidir de qué lado del software ubicarse.

No se puede elegir entre ser esclavo o libre. Al menos nuestro sistema legal no lo permite y por buenas razones, nadie puede venderse, nadie puede ceder esa libertad. Por ningún motivo, siquiera para salvar la vida de un hijo se permite la venta de una persona. ¿Por qué permitir que se cedan las libertades del conocimiento?

En organizaciones colectivas, ¿quién decide?, ¿quién elige? Por ejemplo en el estado. ¿Debe ser el parlamento?, ¿debe ser el presidente? ¿Cada ministro? ¿El responsable de TI de cada oficina? ¿Cada empleado? ¿Esto afecta la libertad de elección?

PLAN

El plan del software libre

> *¿Hasta dónde puede llegar el software libre? No tiene límites, excepto cuando las patentes lo prohíben. El objetivo final del movimiento es proporcionar software libre para hacer todos los trabajos que los usuarios de computadoras quieran hacer y por lo tanto hacer el software propietario obsoleto. Adaptado de RMS [FSF:HPG]*

Hoy la humanidad tiene tres caminos para consolidar la revolución informática y alentar la globalización popular:

- Político. Promover cambios en la legislación de copyright y patentes, evitar el DRM, y la penalización de la copia. Dictar leyes de uso y creación de software libre en el Estado. Promover decretos ejecutivos y migraciones en el Estado, o en el sistema educativo. Participar de encuentros y eventos de todo nivel en cualquier lugar del planeta. Procesar a Microsoft por acciones monopólicas.
- Criminal [Lanier:PF-99]. Utilizar software ilegal, copiar música por Internet. Éste es el camino de la mayoría de la humanidad, constituye la costumbre del uso de las TICs en el planeta. Ilegal pero moralmente válido. No puedo recomendarlo y es un grave error a largo plazo. Como dijo en su momento Bill Gates [Gates:ESA] y fue citado por Amadeu [Amadeu:SLI-03], los usuarios primero usan gratuita e ilegalmente mi software, luego se convierten en adictos.

  Curiosamente la ley intenta cambiar el uso comúnmente aceptado, mientras que la ley debiera ordenar estas costumbres.
- Alternativo. Crear Software Libre. El camino propuesto por Richard Stallman (RMS). Para cada aplicación una alternativa. Otro mundo es posible. [10.3]

LOP

Teorema de exclusión del software libre y privativo.

Tesis: si se apoya el software libre se ataca al privativo.

Definición: el software que no es licenciado libre, es privativo.

Demostración:

En cada momento cada procesador de computadora de la Tierra que esté trabajando, puede estar ejecutando una instrucción libre, privativa legal o privativa ilegal.

En cada momento la cantidad de procesadores del mundo es fija.

Entonces si aumenta el uso de software libre disminuye el de privativo.

Por lo tanto cualquier acción que tienda a incrementar el uso del software libre disminuye el uso del privativo.

Entonces en cada momento, si se trabaja a favor del software libre se trabaja en contra del privativo.

VOTAR

Votando tecnologías

Cada vez que una persona elige un producto [Saravia:GI-04], está votando en los mercados; cada vez que ejerce una opción tecnológica está votando por ella. Los administradores de redes de las universidades americanas votaron por Internet y con esa decisión la impusieron como red mundial global. Perdieron las elecciones Novell con IPX, IBM con SNA, Microsoft con NetBIOS, Digital con DECNET, etc.

Hoy es la hora del software, cada voto cuenta, cada elección define la batalla por la libertad del software. Es la hora de compartir archivos, cada nuevo desarrollo que permite compartir música y ``contenido'' inclina más la balanza.

MERTONIAN

Principios de la ciencia [Mertonian:SN,Cole:MSB-92]:

Comunalismo:

Los resultados de la ciencia son conocimiento libremente y universalmente disponibles para todos. Son aceptados como literatura primaria en el registro público sólo como el resultado de la revisión por pares. El secreto no es posible en el sentido que no tiene peso o crédito en el discurso científico. La deshonestidad no es tolerada y la confianza mutua y personal es la norma. La ética del comunalismo ayuda a explicar la importancia del empirismo y la intersubjetividad en la epistemología científica. Las estrategias epistemológicas incluyen cuantificación, medida, estandarización (y calibración), instrumentación, experimentación y verificación.

Universalismo:

Universalism in science has two faces: first, contributions to science can not be excluded because of race, nationality, religion, social status, gender, sexual preference, or other irrelevant criteria (i.e., science is multicultural); science strives to be a meritocracy; the scientific community attempts to be democratic and fair to its citizens; second, empirical facts have to be consensible and the scientific theories that interpret them have to be consensual; the categories of fact, (representation of experience), taxonomy (classification of facts), and theory (explanation of the classification) have to be shared among the scientists

Desapego:

When scientists present themselves or their work to the scientific community, they try to take a backseat to their own observations and results and to previously reported observations and results; scientific papers are written in a neutral, impersonal, passive voice to minimize the personal and social context of the research; academic science tries to attain consensual objectivity (intersubjectivity) by merging individual interests into a collective enterprise; the purpose of the knowledge is often said to be for knowledges sake, as if there were no interest in the uses to which the knowledge could be put; all reference to economic, political, religious or other social interests is routinely excluded; an impression of humility is conveyed by systematically citing formal scientific sources for everything that is not entirely their own work; on the other hand, although impersonally written, research papers are never anonymous; credibility is the prime personal asset and reliability the prime social asset; thus one role science has to offer a democratic society is as independent arbiter of social questions (of a scientific nature) that otherwise could only be resolved by reference to political, religious, or economic institutions

Originalidad:

Scientists are self-reliant, independent thinkers who are trained to be original; whether choosing a research question, deciding on an approach to address it, or finding a way to convince others of their findings, scientists must display novelty in order to gain maximum credit; when they publish, something about the research has to be new; thus, freedom or independence is a necessity in science (in the academy we call it academic freedom); also, the right to dissent (see below) is absolutely critical; this need for originality explains the emphasis on research rather than, say, scholarship and the drive towards specialization; research areas (requiring projects and proposals) have to be formulated and in some sense discovered

Escepticismo:

Scrutiny of research claims is a hallmark of good science; peer review allows the most important moment for skepticism to be exercised and is the key institution of the scientific culture; this allows scientists to be held accountable to a community, rather than a superior; the scientific culture is an institutionalized context for argumentation; peer review occurs at both ends of the process: in funding research proposals and in accepting for publication

research reports; credibility and relevance are two important criteria when reviewing research; verification (via reproducibility) is a triple point, where psychological, social, and epistemic considerations come together in a critical arena; an additional degree of confidence is achieved by validation by triangulation when different approaches give the same answer

Formatos de almacenamiento e intercambio de datos

FAID

Son las convenciones o formatos que se usan para intercambiar información en el tiempo (almacenamiento), a la distancia, o entre personas usando distintas aplicaciones o programas. Frecuentemente los tipos de archivos se vinculan con distintos formatos. Así, PDF, DOC, XLS, SXW, RTF y otros, representan distintas clases de formatos [Mas:SLS-03].

Si estas convenciones o estándares, usados para contener nuestra información, son abiertos, conocidos por todos y no tienen restricciones de uso, cualquiera podrá consultarlos y realizar programas para acceder a estos datos. Los formatos libres garantizan la libertad de los usuarios para intercambiar información con todo el mundo independientemente de la aplicación que utilicen, ya que permiten a cualquier programador desarrollar software que trabaje con estos formatos.

Lamentablemente los formatos propietarios en ocasiones pueden tener absurdas limitaciones de uso o simplemente no estar documentados. Si, por ejemplo, utilizamos Microsoft PowerPoint para enviar una presentación, sólo tienen garantizado el acceso con todas las particularidades del documento los usuarios de este programa.

En Internet a menudo nos encontramos con creadores de sitios que sólo prueban sus páginas con ``Internet Explorer'' de Microsoft, muchas veces por desconocimiento de la existencia y del grado de implantación de otros navegadores o, en ocasiones, simplemente porque no consideran la compatibilidad como un tema importante.

La comunidad de Internet ha creado sus propios mecanismos para evitar este tipo de situaciones. En 1994 se creó el World Wide Web Consortium (W3C), que agrupa a los principales fabricantes de software de Internet, con la misión principal de definir y promover la creación de estándares para la Web. En realidad, cuando hablamos de estándares web nos referimos casi siempre a las definiciones y recomendaciones de este consorcio, que ha conseguido que prácticamente todos los navegadores modernos funcionen en un grado aceptable con los estándares más recientes.

La restricción en el acceso de la información a un determinado navegador o formato representa una discriminación contra los usuarios de los otros navegadores o aplicaciones.

Las administraciones públicas deben velar por no discriminar contra ninguna plataforma del usuario y no favorecer a ningún fabricante en especial.

Se pueden clasificar los estándares de la misma forma que el software según los derechos que sus usuarios tienen.

Aspectos políticos, legales, penales y comerciales

LIC

Licencias, contratos privados, relación con el copyright

La licencia es el tipo de contrato privado regido por las leyes del copyright que regula el uso de determinado software.

Sus partes son un derecho-habiente-editor que distribuye su material al usuario. Le concede ciertos derechos e impone restricciones. En general son contratos de adhesión. El contrato muchas veces debe ser ratificado cuando se instala un programa, apretando el botón correspondiente, o por el simple hecho de abrir el sobre que lo contiene.

Además estos acuerdos están respaldados por la legislación sobre copyright que penaliza sus violaciones y establece derechos por defecto. La legislación de protección al consumidor muchas veces limita el tipo de contratos posibles en el caso de que sean onerosos.

En el caso del Software Libre, los autores o editores indican que se distribuye bajo una licencia particular que otorga amplios derechos a los usuarios.

En el caso de dominio público sus autores especifican que su software no está sujeto a ninguna restricción y ceden todos sus derechos. Esto es inconveniente porque terceros pueden tomar el software, modificarlo y cerrarlo. Tienen hasta el derecho de cerrarlo y eventualmente obtener copyright sobre el mismo. El Software Libre bajo licencias no copyleft puede ser reutilizado en software privativo, pero no puede obtenerse copyright sobre el mismo.

La mayoría de las licencias de software propietario prohíben técnicas de ingeniería inversa, descompilar, desensamblar el producto e, incluso, su traducción a otras lenguas. Básicamente deniegan el derecho a cualquier modificación o mejora del mismo, lo que queda exclusivamente en manos del fabricante. La mayoría de las licencias tampoco permiten distribuirlo o usarlo en más de un PC.

En un mundo regido por el copyright (o restricciones de copia ``Copyrestrictions''), el software, aún el binario, se considera igual que un libro o una canción. Lo cual es incorrecto pues el software binario no es comprensible por los humanos, siendo un subproducto, no la obra intelectual.

El software propietario se licencia de tal modo que sólo puede ejecutarse en un determinado tipo de computadora, o sistema operativo. La elección de qué combinación de computadora y sistema operativo queda enteramente a discreción del propietario del programa: nadie puede obligarlo a hacer que su programa esté disponible para tal o cual plataforma, y como la licencia es meramente de uso, y no de modificación, tampoco es posible llevarlo uno mismo a la plataforma de elección. Este hecho, combinado con el monopolio inevitable del que goza Microsoft, hace que la elección del usuario esté dictada por las decisiones del propietario del software dominante, en vez de ser hecha en base a sus propias necesidades. Esto es claramente visible en el mercado de computadoras de escritorio, en el que la predominancia de Windows ha convertido a las computadoras basadas en procesadores de la familia Intel en la única alternativa viable.

DH

Derecho-habiente

Es aquel que ostenta los derechos de copia según la legislación. Autor es quien creó la obra. El derecho-habiente de una obra creada por un empleado es su empleador.

VS

Vender Software

La venta de un software es asimilable a la transferencia de los derechos de autor. Habitualmente el software no se vende, incluso, lo más común es que los programadores trabajen en relación de dependencia, por lo que su software es originariamente apropiado por sus empleadores.

Los derecho-habientes-editores generalmente licencian el software a sus usuarios, mediante un contrato particular que transfiere derechos de uso limitados y condicionados. Muchas veces es mejor referirse a la ``distribución de copias de un programa" sea por un monto de dinero o no.

PI

Propiedad Intelectual

Se suelen englobar en esta categoría regímenes muy diferentes como las patentes, marcas, secretos industriales o derechos de autor, los cuales son más diferentes que similares. Mezclar estos sistemas legales y sacar conclusiones sin ser consciente de ello, lleva a generalizaciones incorrectas.

Existe una campaña tendiente a establecer la categoría de ``propiedad intelectual" como unificador de estos sistemas. Esta concepción intenta establecer y asume que las ideas son apropiables, que lo incontable y copiable infinitamente a costo marginal tendiente a 0 es escaso y por lo tanto un bien económico. El concepto de Propiedad Intelectual es un término cargado de ideología y conlleva como presunción escondida, que la forma de pensar más natural sobre la copia está basada en una analogía con objetos físicos y nuestras ideas de ellos como propiedad.

Pero esta analogía esconde la diferencia crucial entre objetos materiales e información: la información puede copiarse y compartirse casi sin esfuerzo, mientras que los objetos materiales no pueden serlo. Usar este término es ignorar la mencionada diferencia.

Incluso el sistema legal de los Estados Unidos de América no acepta por completo esta analogía, debido a que no trata los ``derechos de copia" (copyrights) sólo como derechos de propiedad de objetos físicos.

No debemos usar esta terminología sin saber o ser conscientes de los conceptos que esconde; es mejor evitar el término ``propiedad intelectual" en las palabras y los pensamientos.

Es mejor utilizar los términos de copyright o copyrestrictions o derechos de autor junto con patentes y marcas, o el sistema legal específico del cual se esté hablando.

Es correcto hablar de derechos emergentes de la autoría de la expresión de las ideas, pero no de propiedad de las mismas.

El concepto es un oxímoron, por naturaleza autocontradictorio, ya que las ideas no son apropiables con exclusividad. Pero mediante diferentes regímenes legales aplicados a algunos tipos de ideas o a las representaciones de otros tipos, se consigue el mismo objetivo unificador que se proyecta a un amplio conjunto de tipos, incluyendo a las ``obras intelectuales", los ``inventos", los ``métodos de negocio", al ``software binario", etc.: crear escasez de la abundancia imponiéndoles un costo artificial y con ello incrementando el ``capital" del planeta puesto en juego en el mercado.

DA

Derechos de Autor, copyrights o copyrestrictions

Son un conjunto de concesiones que los Estados del mundo le dan a los derecho-habientes-editores de obras intelectuales con respecto a las mismas. Es un sistema que restringe el derecho natural de copiar la expresión de las ideas que es formativo de las culturas humanas, mediante el imperio de la ley y la policía del Estado. Este sistema se justificó en algún momento para permitir la financiación de las editoriales e imprentas. Hoy, en el mundo digital es anacrónico y se constituye en un freno a la cooperación y el crecimiento, ya que cualquiera con muy pocos recursos puede editar libros y canciones y distribuirlas en segundos a todo el planeta.

La propaganda de las empresas asigna estas concesiones a derechos de copia para la gente, cuando en realidad son restricciones de copia para favorecer a los derecho-habientes o editores de las mismas. Mientras en general los autores cobran un monto pequeño por su obra, las editoriales se benefician enormemente con este sistema quedándose con la ganancia del falso ``capital intelectual que poseen".

Estas externalidades a las economías que restringen la ubicuidad de las ideas, dan un valor económico a elementos artificialmente escasos y naturalmente ilimitados.

En cuanto a la aplicación de este mecanismo de apropiación de ideas para el software es doblemente chocante cuando se usa para software binario, ya que al contrario de los libros y la música no se accede al lenguaje humano con que fueron creados.

El hecho de aplicarse a código cerrado lo pone más cerca de los secretos industriales que de obras intelectuales utilizables por la humanidad.

El software debería considerarse más como idea científica y regirse por sus cánones de conocimiento libre universalMERTONIAN. El Software Libre pone las cosas en su lugar, al proveer una alternativa que poco a poco logrará que el software sea un commodity libremente disponible sin costo de licenciamiento, creando una opción de mayor calidad y menor costo y por lo tanto, con obvias ventajas de mercado.

También se usa el término ``protección" para describir el ``copyright". Esta palabra expresa de cierta forma ``prevención de la destrucción o el sufrimiento"; por lo tanto, impulsa a la gente a identificarse con el propietario y con el publicista, quienes se benefician del ``copyright", en lugar de identificarse con los usuarios quienes son los restringidos por éste.

Es fácil evitar el término ``protección" y utilizar términos neutrales en su lugar. Por ejemplo, en lugar de ``La protección del copyright está vigente por un tiempo prolongado", se puede decir, ``El copyright está vigente por un tiempo prolongado".

Para criticar el copyright en lugar de apoyarlo y ajustarlo a su real funcionalidad, se puede emplear el término de restricciones de copia (copyrestrictions).

H

Hurto, Robo

Los apologistas del copyright emplean con frecuencia palabras como ``robo" y ``hurto" para describir la violación del copyright. Al mismo tiempo, nos piden tratar el sistema legal como una autoridad en ética: si copiar está prohibido, debe estar equivocado.

Así es pertinente mencionar que muchos sistemas legales rechazan la idea de que la violación del copyright es ``hurto". Los abogados del copyright que usan términos como ``robo" están malinterpretando la autoridad que ellos mismos ejercen.

La idea de que las leyes deciden qué está bien o mal es una equivocación en general. Las leyes son, en su mejor concepto, un esfuerzo por lograr justicia; decir que las leyes definen justicia o conducta ética es dar vuelta a todo.

P

Piratería, o compartir las ideas, adicción y software-traficantes.

Los publicistas de las multinacionales frecuentemente se refieren a la copia prohibida como ``piratería". De esta forma, ellos expresan indirectamente que hacer copias ilegales es éticamente equivalente a atacar barcos en alta mar, secuestrar y asesinar a la gente que viaja en ellos. Han instalado en el sentido común y en el lenguaje la palabra piratería, con toda la carga ideológica que esta conlleva.

La copia ilegal no es como secuestrar y asesinar, entonces no se debe usar la palabra ``piratería" para describirla. Términos no cargados ideológicamente como ``copia prohibida, no autorizada, o ilegal" están disponibles para utilizar, en lugar de ello. Inclusive es preferible utilizar un término positivo tal como ``compartir información con tu vecino".

Bill Gates al respecto, el 20 de Julio del 1998, en la Universidad de Washington dijo [Gates:ESA,Microsoft:ASA-2004]:

> *A pesar de que se venden 3 millones de computadoras cada año en China, las personas no pagan por el software. Algún día ellos pagarán. Nosotros queremos que ellos nos roben. Ellos se tornarán adictos, y entonces, de alguna forma nosotros descubriremos cómo cobrarles en algún momento de la próxima década.*

PS

Patentes de software

La aplicación de las patentes al software puede constituir un desafío importante al movimiento de Software Libre. Patentar es caro, y los grupos de desarrollo libre no podrían competir con las multinacionales que ven a las patentes como un mecanismo de defensa y patentan miles de ideas ridículas por año.

En una decisión histórica, la Unión Europea ratificó la no patentabilidad del software luego de manifestaciones y una acción concertada de los activistas del Software Libre, cambiando el sentido del proyecto originariamente presentado a su parlamento. Posteriormente algunos en la Comisión Europea, intentaron de cualquier forma, revertir esta decisión poniéndose a prueba la democracia europea en una historia épica.

Se reproduce una carta del Profesor Knuth a la oficina de patentes de su gobierno que resume de manera admirable lo ridículo del concepto de patentes de software. [Knuth:CSP].

> *Estimado Comisionado:*

> *En nombre mío y de muchos otros científicos informáticos, quisiera solicitarle que reconsidere la política actual de otorgar patentes de procesos computacionales. Noto una ansiedad considerable en la comunidad informática debido a que las decisiones de las cortes de patentes y de la Oficina de Patentes y Marcas Registradas están haciendo que la vida de los programadores sea mucho más difícil.*

> *En el período comprendido entre 1945-1980 se aceptaba que la ley de patentes no aplicaba al software. Ahora algunas personas han recibido patentes por algoritmos de importancia práctica -por ejemplo, compresión Lempel-Ziv y la encripción de llave pública RSA-, y están utilizando las leyes para evitar que otros programadores los usen.*

> *Este es un cambio enorme con respecto a la política anterior, que permitió la revolución computacional, y temo que perjudicará a la sociedad. Para mí es clarísimo que este cambio habría tenido un efecto profundamente negativo en mi trabajo: Por ejemplo, desarrollé un software llamado TeX que actualmente es usado para producir más del 90% de todos los libros y revistas matemáticos y físicos, y para producir cientos de miles de reportes técnicos en todas las disciplinas científicas. Si las patentes de software hubieran sido frecuentes en 1980, no habría podido crear tal sistema; probablemente ni siquiera habría considerado hacerlo y tampoco puedo imaginar a nadie que se hubiera atrevido a hacerlo.*

> *Me han dicho que las cortes están tratando de hacer una distinción entre los algoritmos que son matemáticos y los que no lo son. Para un científico de la computación esto no tiene sentido: nada es más matemático que un algoritmo. Un algoritmo es un concepto abstracto, que no tiene ninguna relación con las leyes físicas del universo.*

> *Tampoco es posible distinguir entre algoritmos ``numéricos'' y ``no numéricos'', como si los números tuvieran alguna característica que los distinguiera de las otras formas de información precisa. Todos los datos son números y todos los números son datos. Los matemáticos trabajan mucho más con entidades simbólicas que con números.*

> *Por este motivo, la idea de aprobar leyes que afirmen que algunas clases de algoritmos pertenecen a las matemáticas y otras no me parece tan absurda como los intentos de la legislatura de Indiana en el siglo XIX de aprobar una ley diciendo que la relación entre la circunferencia y el diámetro de un círculo es exactamente 3, y no aproximadamente 3.1416. Es como la iglesia medieval decretando que el sol da vueltas sobre la tierra. Las leyes hechas por el hombre pueden ser muy útiles pero no cuando contradicen verdades fundamentales. El congreso decidió sabiamente hace mucho tiempo que las entidades matemáticas no pueden ser patentadas. Claramente, nadie podría aplicar las matemáticas si fuese necesario pagar un valor por una licencia cada vez que se use el teorema de Pitágoras. Las ideas algorítmicas básicas, que hoy en día muchos están patentando, son tan fundamentales que las consecuencias amenazan con ser equiparables a las que se tendría si se permitiera a los autores patentar individualmente palabras y conceptos. Los novelistas y columnistas no podrían escribir historias excepto en los casos en que las editoriales fueran autorizadas por los propietarios de las palabras. Los algoritmos son tan básicos para el software como lo son las palabras para los escritores: son las piezas fundamentales que se necesitan para armar productos interesantes. ¿Qué pasaría si los abogados pudiesen patentar sus métodos de defensa o si los jueces de la Corte Suprema pudiesen patentar sus antecedentes?*

> *Soy consciente de que las cortes de patentes hacen sus mejores esfuerzos para servir a la sociedad cuando formulan las leyes de patentes. La oficina de patentes ha cumplido su misión de manera admirable con respecto a aspectos tecnológicos que involucran leyes físicas concretas en vez de leyes abstractas del pensamiento. Yo mismo tengo patentes en algunos dispositivos de hardware. Pero creo firmemente que la tendencia reciente de patentar algoritmos solamente beneficia a un reducido número de abogados e inventores mientras que perjudica seriamente a la mayoría de las personas que quieren hacer cosas útiles con computadores.*

> *Cuando pienso en los programas para computadoras que necesito a diario para trabajar me doy cuenta de que ninguno existiría si las patentes de software hubiesen sido comunes en los 1960s y 1970s. Cambiar las reglas ahora tendrá el efecto de congelar el progreso en su estado actual. Si las tendencias actuales continúan, el único recurso para la mayoría de los brillantes desarrolladores de software de Estados Unidos será dedicarse a otras labores o emigrar. Los Estados Unidos perderán pronto su posición dominante.*

> *Por favor hagan lo que puedan para retroceder esta alarmante tendencia. Hay formas mucho mejores para proteger los derechos de propiedad intelectual de los desarrolladores de software que quitarles el derecho a usar las piezas fundamentales con las que se construye el software.*

> *Sinceramente,*

> *Donald E. Knuth*
>
> *Professor Emeritus*

Y Bill Gates lo ratifica diciendo a los empleados de Microsoft en 1991: [Gates:QCS-04]

> *Si la gente hubiera entendido cómo las patentes iban a ser otorgadas, cuando la mayoría de las ideas de hoy fueron inventadas, y las hubiesen patentado, la industria estaría totalmente congelada hoy en día... Una futura empresa en surgimiento, que no tuviera patentes, se vería forzada a pagar cualquier precio que los gigantes eligieran imponerle.*

IT

Informática traidora o TCPA, TCG, Palladium, NGSCB, DRM, Longhorn, Centrino [TCG:SW,Anderson:PFI-03,p2pnet:TPE,Anderson:ESR]

> *1.- ¿Qué es esta historia de la ``Informática Fiable''?*
>
> *El Grupo para la Informática Fiable (Trusted Computing Group (TCG) es una alianza de Microsoft, Intel, IBM, HP, y AMD que promueve un estándar para un ordenador ``más seguro''. Su definición de ``seguridad'' es controvertida; las máquinas construidas según sus especificaciones serán más fiables desde el punto de vista de los proveedores de software y la industria del contenido, pero menos fiables desde el punto de vista de los dueños. De hecho, las especificaciones TCG transfieren el control último del ordenador a quienquiera que escribiera el software que éste ejecuta. (Sí, incluso más que ahora) Al proyecto TCG se le conoce por una variedad de nombres. ``Informática Fiable'' (Trusted Computing) fue el original, que todavía usa IBM, mientras que Microsoft lo denomina Trustworthy Computing (NT: en castellano tendría un sentido similar a informática fiable) y la Free Software Foundation (FSF) lo llama Informática Traidora treacherous computing. De aquí en adelante, lo llamaré simplemente TC. Entre otros nombres también puedes encontrar TCPA (el nombre original del TCG), Palladium (el antiguo nombre de Microsoft para la versión que se publicaría en el 2004) y NGSCB (el nuevo nombre de Microsoft). Intel en este momento lo empieza a denominar ``Informática más segura'' (Safer computing). Muchos observadores creen que esta confusión es deliberada*
>
> *Los promotores quieren distraer la atención sobre lo que TC hace realmente.*
>
> *2.- ¿Qué hace TC, en castizo?*

> *TC aporta una plataforma informática donde las aplicaciones no pueden ser alteradas o modificadas, y donde éstas se pueden comunicar de forma segura con su fabricante y entre sí. Su aplicación original era el Control de Derechos Digitales (digital rights management - DRM): Disney podrá venderte DVDs que sólo se descodifiquen y ejecuten en plataformas TC, pero que no podrás copiar. La industria discográfica podrá venderte descargas de música que no podrás intercambiar. Podrán venderte CDs que sólo podrás oír 3 veces, o solamente el día de tu cumpleaños. Toda una nueva gama de posibilidades en marketing a su alcance. TC hará también muy difícil ejecutar software sin licencia. En la primera versión de TC, el software pirateado se podía detectar y borrar de forma remota. Desde aquello, Microsoft ha denegado en algunas ocasiones que quisiera que TC hiciera esto, pero en el WEIS 2003 un director senior se negó a desmentir que luchar contra la piratería es una meta: ``Ayudar a que la gente ejecute software robado simplemente no es nuestra meta en la vida''. Los mecanismos ahora propuestos son más sutiles, sin embargo. TC protegerá los procesos de registro de las aplicaciones, de tal forma que el software sin licencia esté bloqueado. Además, las aplicaciones TC funcionarán mejor con otras aplicaciones TC, de tal forma que ya no merecerá la pena usar aplicaciones no-TC (incluyendo las piratas). Del mismo modo, algunas aplicaciones TC podrán rehusar abrir ficheros de viejas aplicaciones cuyos números de serie hayan sido prohibidos. Si Microsoft opina que tu copia de Office es pirata, y tu gobierno local se cambia a TC, entonces los documentos que intercambies con ellos se pueden volver ilegibles. TC también hace más fácil alquilar software que comprarlo; si dejas de pagar el alquiler, no solamente el software dejará de funcionar, sino probablemente también los ficheros creados con él. Así que si dejas de pagar actualizaciones para el Media Player, puede que pierdas acceso a todas las canciones que compraste con él.*

> *Durante años, Bill Gates ha soñado con una forma de hacer pagar a los chinos por el software que usan; ésta puede ser la respuesta a sus plegarias.*

> *Hay muchas otras posibilidades. Los Gobiernos serán capaces de ajustar las cosas de tal forma que todos los documentos de Microsoft Word creados en PCs de funcionarios civiles sean 'clasificados' y no se puedan filtrar electrónicamente a la Prensa. Los sitios Web de subastas pueden hacer obligatorio el uso de TCPA/Palladium para pujar. Hacer trampas en juegos en línea también puede ser más difícil. También hay desventajas. Por ejemplo, puede que haya censura remota. En su forma más simple, se pueden diseñar aplicaciones para borrar música pirateada bajo petición. Si se extrae una canción protegida de una plataforma TC comprometida (crackeada) y es puesta en la Web como un fichero MP3, el reproductor de ficheros compatible con TC puede que lo detecte usando una marca de agua, informe de ello y sea informado de que debe borrarlo (del mismo modo que cualquier otro dato que viniera de esa fuente comprometida). Este modelo de negocio, denominado traitor tracing (búsqueda del traidor) ha sido investigado extensivamente por Microsoft (y otros). En general, los objetos digitales creados usando sistemas TC permanecen bajo control de sus creadores, en lugar de bajo control de los dueños de las máquinas que los albergan (como sucede en la actualidad). Entonces, el autor de un escrito designado como difamatorio por un tribunal, puede ser requerido a censurarlo y la empresa dueña del procesador de textos usado, ordenada a borrar el fichero si el autor se niega. Dadas estas posibilidades, debemos esperar que TC sea usado para suprimir cualquier cosa desde pornografía a textos que critican a líderes políticos.*

> *La desventaja para las empresas es que los proveedores de software pueden hacer muy difícil cambiarse a un producto de un competidor. Por ejemplo, Word podría cifrar todos sus documentos con claves a las que sólo tendrían acceso otros productos de Microsoft; de tal forma que sólo tendrías acceso usando programas de esa compañía, y no con ningún otro procesador de textos de la competencia. Un bloqueo tan clamoroso probablemente sea prohibido por las autoridades antimonopolio, pero hay estrategias de bloqueo más sutiles que son mucho más difíciles de regular.*

> *Sigue...*

DMCA

EUCD, ALCA

La European Copyright Directive (EUCD) y la Digital Millennium Copyright Act (DMCA) [AntiDMCA:SW], implementan el tratado sobre Copyright de diciembre de 1996 de la Organización Mundial de la Propiedad Intelectual. (OMPI o WIPO)

Existe enorme inquietud por los efectos de restricción de libertades que su aplicación conlleva.

El software propietario pone un área que antes estaba regulada por representantes electos democráticamente en manos de las corporaciones, estableciendo una tecnocracia multinacional.

El ALCA prevé incorporar medidas similares en América Latina [Inquirrer:ITW-03].

Economía y ecología

BIEN

¿Es el conocimiento un bien? ¿Es un bien común? ¿O es por naturaleza libre?

> *Si hay algo en la Naturaleza que sea menos susceptible de propiedad exclusiva que todo lo demás, es la acción del poder intelectual llamada ``idea'', la cual un individuo puede poseer exclusivamente mientras se la guarde; pero el momento en que se divulga, se convierte por fuerza en la posesión de todos, puesto que el receptor no puede desposeerse de ella. Quien recibe de mí una idea recibe instrucción sin disminuir la mía; igual que quien enciende su vela con la mía recibe luz sin oscurecerme. Que las ideas deberían difundirse libremente entre las gentes por todo el globo, para la instrucción moral y mutua de la humanidad, y la mejora de su condición, parece algo diseñado de forma peculiar y benevolente por la naturaleza cuando las hizo, como el fuego, expandibles por todo el espacio, sin perder densidad en ningún punto, y como el aire que respiramos, en el que nos movemos y tenemos nuestro ser físico, incapaces de confinamiento o apropiación exclusiva. Los inventos no pueden así, por naturaleza, ser sujetos a propiedad.*
>
> *Thomas Jefferson.*

Si alguien tiene una manzana debe decidir qué hacer, la guarda, la planta, la comparte, la cede, etc. Cuando alguien da una idea a otro, ambos la tienen por completo y pueden usarla sin necesidad de tomar decisiones en común sobre la misma. No hay apropiación exclusiva [Saravia:EI-03], salvo mediante leyes artificiales, útiles cuando las ideas no estaban digitalizadas y dependían de su soporte material.

Las ideas no son cosas, también existen otros ``nombrables'' como la luz, los abrazos y los besos, que no son cosas [Chaparro:TTN,Judicial:INC].

Sin embargo existen sistemas legales artificiales que limitan la libertad del conocimiento.

Si partimos de un estudio analítico, la construcción desde abajo hacia arriba de las ideas de patentes, copyrights, marcas y posiblemente otros conceptos; no parte de una base común. Estos regímenes surgen de diferentes motivaciones, con sistemas legales diferentes y sin ninguna vinculación.

Pero si partimos de un análisis sintético veremos que estos sistemas legales artificiales construyen valor económico sobre elementos inmateriales, al hacer escaso lo abundante y restringir su

circulación. Al crear derechos individuales y restringir derechos colectivos, asignan valor económico a las ideas y sus representaciones. Les designan un titular y las ponen en el mercado, permiten su compra, venta y alquiler o licenciamiento. De esa forma aseguran la apropiación exclusiva de las ideas (como es el caso de las patentes) o sobre sus representaciones (el caso del copyright) durante un largo período de tiempo. [10.4]

Las ideas pueden ser ``apropiadas" por un grupo humano o comunidad, pero eso no pone ningún límite a cualquier otra persona, grupo o comunidad, ya que no es una apropiación exclusiva, característica necesaria para el establecimiento de la propiedad. Se desprende entonces que el concepto central aquí es LA LIBERTAD DEL CONOCIMIENTO, no la dicotomía entre su APROPIACIÓN COLECTIVA vs. su APROPIACIÓN INDIVIDUAL exclusiva. La apropiación colectiva puede ser un fin loable para las cosas materiales, pero de ningún modo para las ideas.

Cuando se habla de ``commons" suele hablarse de cosas comunitarias, es decir que tienen un dominio comunitario, o sea que son cosas sobre las cuales se toman decisiones, y esas decisiones las toma una comunidad, por la vía que ésta tenga establecida. Puede ser una persona jurídica ``estrecha y legal" del tipo ``occidental y cristiano" o puede ser una ``comunidad abierta" del tipo ``usuarios y desarrolladores del soft libre" o ``pueblos originarios". Estas cuestiones se vinculan habitualmente con la propiedad, aunque no tiene por qué ser siempre así.

Hay que evitar a toda costa usar las palabras de ``bien común", ``commons" o cualquier concepto que represente la idea de un ``patrimonio colectivo" para las ideas. Las ideas no son bienes, no son productos, no son patrimonio. Por ende no pueden ser del individuo, pero tampoco de la comunidad en su conjunto o de alguna comunidad en particular. Son libres. Con el conocimiento la humanidad ``progresa" pero no por el lado material.

En cambio con las ideas no es así. No hay decisión común. Cada persona tiene su versión de la idea. No hay nada en común a decidir.

No tiene nada que ver la libertad del conocimiento con la propiedad compartida o el bien común.

La cuestión de fondo es que hay alguien ``individuo o comunidad" que debe tomar decisiones sobre el ``bien", lo que no pasa con las ideas. No es un ``bien", mucho menos propiedad, pero nadie puede decidir sobre todas las instancias de esa idea, sólo sobre la que tenga en su cabeza. Cualquiera puede hacer lo que quiera con la idea. Por lo tanto no hay una comunidad que deba decidir. No es un recurso, o bien, o propiedad regulado por la escasez.

El conocimiento no es propiedad común, no es propiedad, es libre. Sé que para muchos es agradable asociar el software libre con las ideas de propiedad comunitaria, pero no es lo mismo.

Concepto	**Típico**	**¿Bien económico?**	**¿Duplicable?**	**Quién decide**	**Ejemplo intelectual en el sistema deformado actual**

Lo libre	ideas	No	Sin costo	cada ``poseedor''	Software bajo copyright tipo copyleft
Lo compartido	bienes comunes	Sí, fuera del mercado	Con costo (artificial para ideas)	el conjunto mediante algún sistema	Estándares: todos usan, nadie cambia por sí, se decide mediante ``representantes''
Lo individual	bienes de cada individuo	Sí, en el mercado	Con costo (artificial para ideas)	el dueño	Software bajo copyright tipo EULA de Microsoft. Patentes.
Disperso (no hay ``Lo'').	No hay un concepto que globalice los derechos de las personas en relación a las ideas o sus expresiones, Posición de RMS. Copyrights, patentes y otros son diferentes.				

La visión ``Dispersa'' nos dice: ``No hay nada de qué hablar''. Las otras tres nos dicen que podemos pensar un concepto integrador de la mano de la economía. Y en esta situación vemos dos posibilidades de respuesta a la pregunta: ¿hacemos escasas a las ideas? Si la respuesta es no, estamos en el concepto de ``Libre''. Si la respuesta es sí, estaremos en la situación actual, donde las ideas o sus expresiones son artificialmente convertidas en bienes. Y allí podremos darle la ``propiedad'' a la humanidad en su conjunto o a las personas, o pensar caminos intermedios, o dar la posibilidad de la ``libre elección'' como en el caso de las Creative Commons.

CC

Creative Commons [CC:SW]

Conjunto de licencias que permiten a las personas optar. Pensadas para música, material bibliográfico y otras expresiones. No deben aplicarse al software . ``Creative Commons'' alimenta la posibilidad de la "libre elección" suponiendo que todos los regímenes son buenos, que los autores eligen y los ``usuarios'' deben acatar.

Se argumenta que la libertad debe darse solamente para el software o en general para todo conocimiento funcional, pero que con la música, literatura fantástica u otras obras intelectuales dedicadas al placer no es necesario. Debe tenerse en cuenta que la libertad en discusión es para quien ``usa'' la obra (libertad, no obligación). La cuestión central entonces es: quien conoce algo, ¿tiene la libertad de difundirlo a otros? ¿El software es diferente de una canción, en esto? ¿El placer es menos serio que el software?

CC sería un retraso si se aplicara al software y es un avance con relación a las prácticas mayoritarias en música y material bibliográfico.

La FDL lleva asociado el mensaje de la libertad y la CC lleva el mensaje de la libre elección donde todo está bien y donde se puede elegir, si se lo desea, una licencia privativa para los libros y música.

LNP

Libertad, no precio

El ``Software Libre'' es un asunto de libertad, no de precio. Para entender el concepto, es necesario pensar en ``libre'' como en ``libertad de expresión'', no como en ``consumo libre''.

Libertad de expresión y no cerveza gratis.

En inglés una misma palabra (free) significa tanto libre como gratis, lo que ha dado lugar a cierta confusión.

Cuando se habla de Software Libre, es mejor evitar términos como: ``regalar'' o ``gratis'', porque esos términos implican que lo importante es el precio, y no la libertad. En cada idioma es fundamental usar las palabras que refieran directa y concretamente al concepto de libertad, y no simplemente usar y repetir de forma mecánica alguna traducción errónea de un concepto misterioso de mercados extranjeros.

MESL

Modelo económico del Software Libre, economía solidaria

El Software Libre permite construir una economía solidaria que opere bajo el concepto de la prestación de servicios y donde los ingresos se relacionan directamente con las horas de trabajo efectivo y no con los derechos, potencialmente infinitos, de la explotación comercial de una licencia de software.

Dicha economía es de escala múltiple: en ella tienen lugar las grandes corporaciones internacionales para proyectos de gran magnitud y las empresas unipersonales que prestan servicios a individuos; todos los puntos intermedios son válidos. Esto contrasta de manera notable con el esquema de las pocas multinacionales actuales del soft que se constituyen en recaudadoras de impuestos en todo el mundo por el uso de sus programas en cada PC bajo el poder de policía del Estado y la protección legal de las restricciones de copia (Copy-restrictions).

El acceso al código fuente, libremente visible y modificable permite construir una cultura de la cooperación donde todos participan y contribuyen en la medida de sus posibilidades y necesidades a la construcción de un acervo de conocimiento universal que va creciendo, como un edificio a partir de sus cimientos.

No hay arquitectos centrales, más allá de lo aceptado por todos, ya que es simple y está bien visto crear desarrollos nuevos a partir de proyectos con cuyas decisiones no se acuerda. Esta meritocracia del consenso y con disenso aceptado es el motor del cambio. Esta es la más grande aventura colectiva de la historia mundial de la humanidad en la construcción de otro mundo más real que posible.

Si las ideas fuesen bienes económicos el valor total del Software Libre producido en el planeta sería monstruoso. Muchísimo más alto que lo que Microsoft hace figurar como donaciones -fijando precios como quiere y eludiendo impuestos- a causas públicas.

El punto central de este modelo es que cada partícipe da uno y recibe mil. En el software privativo debe pagar cada código que usa y cobrarlo en su producto combinado. En el modelo libre no. Se disminuye el capital falsamente creado y circulante.

ET

Ética del trabajo

El movimiento hacker posee una cultura del trabajo diferente de la cultura protestante definida por Weber [Weber:EPE].

El trabajo se considera más un placer que una obligación. No se siguen horarios ni rutinas. La actividad consiste en la creación de nuevas herramientas que permiten definir nuevos espacios y mecanismos de comunicación. [Himanen:EHE-02]

ESL

Ecología y Software Libre [Ourproject:ECL]

PROS

Prosumidores

Productor-consumidor: término tomado de la economía solidaria y los movimientos de trueque. No es utilizable para el software, ya que el software no se produce, pero sí para los servicios relacionados. Refleja la idea de comunidad que crea, mejora, comparte y usa su conocimiento con un modelo económico sustentable.

BD

Brecha Digital

Expresa la preocupación mundial relacionada con la ubicuidad creciente de las tecnologías de la información y las comunicaciones, y las consecuencias negativas aparejadas por un acceso diferencial a las mismas. Los sectores marginados por motivos estructurales, económicos, geográficos, culturales y sociales entre otros empiezan a ser considerados analfabetos digitales.

Los ciudadanos de países en vías de desarrollo donde la renta per cápita es muy baja, deben trabajar meses y dedicar su salario completo para pagar una licencia de Microsoft Office y Windows XP. No es de extrañar que los índices de uso ilegal de software sean tan altos. Lo que está mal es la legislación, no las costumbres. Esta clase de legislación impuesta a posteriori, crea

de la noche a la mañana miles de criminales y no satisface los mínimos criterios para ser considerada una legislación legítima.

El concepto se está utilizando para conseguir que los fondos para el desarrollo se vuelquen a pagar conectividad, computadoras y software privativo, incluso vía exenciones impositivas. Si la idea es eliminar la pobreza y no informatizar a los pobres, habrá en cada caso que evaluar si no hay formas mejores de invertir dichos fondos con el fin de eliminar la pobreza.

Modelos de desarrollo de software

BC

Bazar y Catedral

Se refiere a dos modelos de desarrollo de software, uno típico del Software Libre como el Bazar y otro usado tanto en Software Libre como en el propietario denominado la Catedral. Existen muchas vinculaciones con la metodología de desarrollo ``Programación Extrema" (XP).

El bazar es un método dinámico, no estructurado y comunitario, muy apto para formar comunidades en la red. [Raymond:CB]

> *El modelo de cooperación utilizado por el Software Libre no es nuevo. Sin ir más lejos, el principal diccionario de referencia en lengua inglesa, el Oxford English Dictionary (OED), fue desarrollado de forma cooperativa siguiendo un modelo muy similar al usado por el Software Libre. A finales del siglo XIX, James Murray, el primer editor del OED, pidió ayuda públicamente para completar el diccionario. Casi 400 personas le enviaron información sobre palabras de la lengua inglesa y ejemplos ilustrativos de su uso de forma totalmente altruista, un material que fue recogido en dicha obra. Hoy en día, el OED sigue admitiendo colaboraciones [Mas:SLS-03].*

Organizaciones, grupos, listas y comunidades

FSF

[FSF:SW] La Free Software Foundation (FSF) es el organizador del Proyecto GNU. Creada a tal efecto y punto de partida del movimiento de software libre, es la principal responsable del desarrollo de Software Libre en el mundo y la principal aportante de coherencia al movimiento.

Su referente es Richard Stallman. Existen además fundaciones similares en Europa e India. La fundación europea involucra organizaciones no solamente en Europa como la FSF Japón y Vía Libre en Argentina.

HIPATIA

(Hypatia) [Hipatia:SW]. Es una organización mundial, popular y democrática que promueve la adopción de políticas públicas, junto a conductas humanas y sociales, que favorecen la libre disponibilidad, sustentabilidad y socialización de la tecnología y el conocimiento, su uso solidario y la viabilidad del modelo económico y social que lo construye en términos de igualdad e inclusión de todos los seres humanos y los pueblos del mundo.

Considera que el Software Libre es un camino hacia un conocimiento socialmente justo, tecnológicamente sustentable y económicamente viable [Saravia:MH-01,Hipatia:SM-04].

Participa activamente en Foros Sociales Mundiales en la construcción del ``Otro Mundo'', si bien ha cuestionado su política de manejo de la información [Hipatia:PPL-04]. En el foro 2005, se ha incrementado notoriamente el uso de Software Libre en sus procedimientos internos.

ON

Existen diferentes movimientos nacionales con sus respectivas organizaciones en diferentes países, destacamos:

- Hispalinux: Organización Española de Software Libre [Hispalinux:SW].
- PSL: Proyecto Software Libre Brasil [Psl:SW].

  Movimiento que impulsa el desarrollo del Software Libre, primero en Porto Alegre, Río Grande do Sul y ahora en todo Brasil. Recientemente fue creada una asociación con personería jurídica.
- SOLAR: Organización Argentina de Software Libre [Solar:SW].

PROP

Proposición [Proposicion:LC]

Lista originaria de Argentina donde se discute la legislación para el uso de Software Libre en el Estado.

LUG

Linux Users Groups o Grupos de Usuarios Linux

Son las asociaciones de usuarios del sistema GNU/Linux. Fueron los primeros núcleos duros de hackers que trabajaron y trabajan con Software Libre.

La asociación Linux Internacional [Li:SW] suele agruparlos y contiene listados de ellos.

HL

Hacklabs

Son comunas organizadas alrededor de laboratorios participativos, donde se difunden las ideas libertarias de Internet. Es un movimiento principalmente vivo en Europa. Muy activas en Software Libre y enlaces inalámbricos de Internet (Wi-Fi).

HC

Hacker y Cracker

Los enemigos del movimiento intentan establecer que los hackers son peligrosos y están al margen de la ley. Nada más falso, el movimiento hacker se caracteriza por tener personas inquietas y curiosas que aportan conocimiento en forma solidaria a su comunidad. La palabra correcta para describir conductas inapropiadas de violación de información personal es Cracker.

OEKONUX

Project [Oekonux:SW]

Radicado fundamentalmente en Alemania, se interesa por las formas políticas y económicas del Software Libre.

Varios

TCO

Total cost of ownership, Costo total de posesión.

Mide los costos de IT, la complejidad y las mejores prácticas en el transcurso del tiempo. Incluye los costos directos (costo de adquisición, hardware, licencias, tasas, manejo, operación) e indirectos (operación de usuarios finales, tiempo fuera de servicio, etc.) [Dell:CTO-03]

Microsoft usa mucho este criterio de evaluación.

ROI

Retorno de la Inversión

Criterio para evaluar proyectos de inversión. Recomendamos usarlo para cualquier proyecto que involucre migraciones.

Economía de la escasez

> *Pretender que el conocimiento es un recurso económico más de la producción, es intentar asimilar lo nuevo a las viejas teorías. Y lleva a conclusiones nefastas. Necesitamos otras teorías.*

Subsecciones

Introducción

La economía [11.1] es la ciencia [11.2] que estudia los fenómenos influidos por la escasez. Se aplica a un sistema cuando sus flujos de energía, materia e información, junto a sus estructuras disipativas [Prigogine:OOC-84], consumen bienes escasos. (Apéndice: 11.4.2.)

Dada la definición precedente [11.3] el título del trabajo parece redundante. Sin embargo casi nunca se profundiza en las consecuencias de esta definición. A partir de su análisis crítico, y formalización, el presente trabajo pretende explorar y generalizar la economía en dos direcciones.

Por un lado se intenta contribuir al análisis contable del crecimiento[Odum:AES-80,Rosnay:MHV-77] y el ``progreso"[Wagensberg:P-98]. Existirá una relación importante entre la ciencia de la escasez (economía) y la ciencia que contabilice el progreso (general), pero la primera es sólo una parte de la segunda. Para salir de la visión ``escasa" se deben comprender las condiciones que la ciencia de la escasez (economía) impone a una ciencia más general que la contenga y clarificar qué papel juega la economía en ella.

Por otro lado estudiar los sistemas ``vivos" o ``auto-reproductivos": sociedades prehistóricas, sistemas biológicos o ecológicos, y no sólo las sociedades capitalistas y humanas, actuales o históricas.

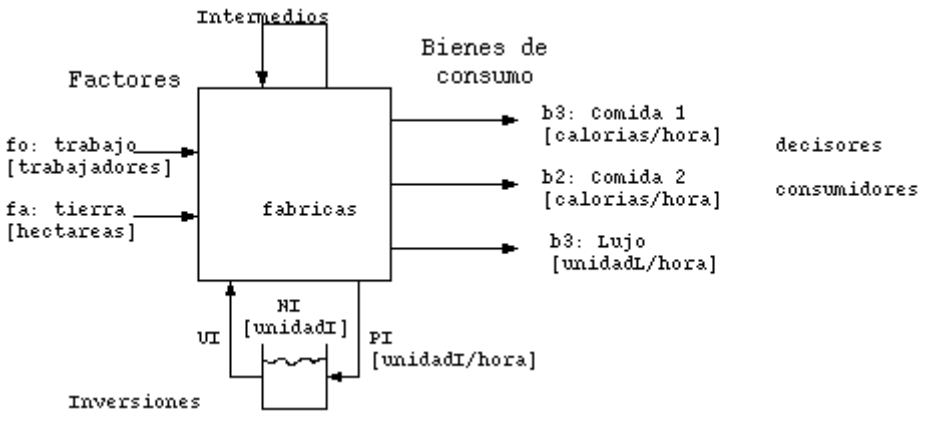
Se pone el acento en los flujos de bienes de las realidades económicas y no en las cuestiones o mecanismos internos particulares de algunos sistemas, como el dinero, o la propiedad en relación al trabajo como mecanismo de distribución de lo producido.

Este trabajo intenta ser útil para comprender las consecuencias de tomar el conocimiento digitalizado como un elemento no escaso.[11.4] Cuestión que se plantea con la posibilidad que nos da Internet de distribuir contenido digital a un costo marginal nulo. Un ejemplo concreto y actual de lo que pasa cuando un recurso deja de ser escaso. El trabajo pretende ser simple y explicativo como para alimentar la discusión de políticas sobre el oxímoron ``propiedad intelectual". [11.5]

``Defender la alegría"[Benedetti:DA-85], es una premisa útil para los que deseamos construir un mundo mejor. Pensar un camino y ser optimista sobre el mismo, ayuda a lograr el apoyo necesario y suficiente para recorrerlo. [Postcarcity:SW] [11.6]

Desarrollo

Conceptos primitivos. Modelo económico básico



Esquema de la economía

Conceptos primitivos de una realidad económica:

Bienes económicos,
o directamente bienes: todos ellos escasos y contables. Los bienes económicos tienen un ciclo de vida. Luego de su creación, adquisición, extracción, producción o generación, se consumen o se usan, sea para su disfrute o para producir otros bienes. En el proceso circulan[11.7] y se transforman.

Se sistematizan las hipótesis del modelo en postulados, así:

Postulado Cero: *Los bienes económicos son escasos.*

Los bienes se pueden clasificar en cuatro tipos:

Factores:
se usan sólo para producir otros bienes. Tienen topes, máximos o límites a su uso; externos a la realidad económica.[11.8] Supondremos que no son necesarios otros bienes para su extracción o generación.[11.9]

En este trabajo consideraremos dos factores: trabajo (F_o) y tierra (F_a) [11.10]

Consumibles,
o bienes de consumo: en nuestro modelo se consumen -destruyen- apenas fabricados. Producir estos bienes es el objeto de la realidad económica.

Bienes intermedios:
se producen para ser utilizados en la fabricación de otros bienes.

Inversiones,
son bienes intermedios que se introducen para modelar la acumulación.

Fábricas:
donde se producen los bienes. Cada fábrica produce un flujo de bienes a partir de flujos de factores o bienes intermedios.

Consumidores:
quienes consumen bienes. [11.11]

Decisores:
los que deciden cuáles de las alternativas posibles se efectivizan. Sin elección no hay ciencia económica. [11.12]

Bienes intermedios e inversiones

1. La posibilidad de acumular bienes, y los efectos de las variaciones en su acumulación se representan con un bien intermedio especial, denominado inversión. Cualquier bien que se acumule será modelado como una producción intermedia del ``bien inversión" y su ``reconversión al consumible deseado" será a la tasa exacta de su uso. Se modela con este bien cualquier otro que habitualmente se considere inversión. Estos bienes se irán depreciando a medida que se ``usan" para fabricar otros bienes.

   Así la única posibilidad de acumulación y fenómenos transitorios del modelo queda en la variable $\mathbf{I}$. En cada momento su cantidad depende del pasado y en esos términos es ``escasa" y no modificable instantáneamente. Por ello, habitualmente, se toma como factor este ``bien".

   Primer Postulado: *La acumulación se representa mediante un bien especial denominado inversión. No se acumula en los flujos.* [11.13]

2. Se elimina la acumulación transitoria. [11.14]

   Todo lo que se invierte eventualmente se utiliza o consume (deprecia). El único efecto de las inversiones es permitir una acumulación temporal y definir un nivel de inversión. En algunos casos interesará el nivel de inversión de una realidad económica o incluso sus variaciones o efectos transitorios.

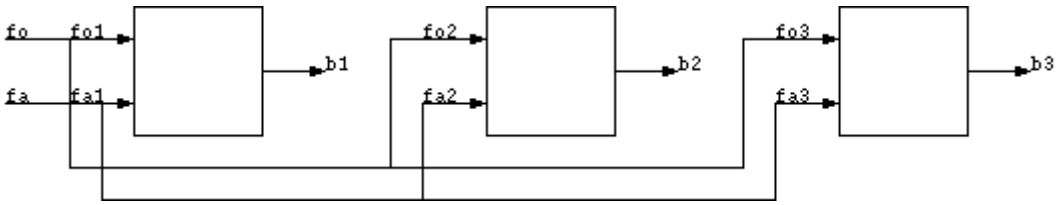
   En este trabajo se elimina la posibilidad de variar el nivel de inversión y con ello estudiar esos fenómenos transitorios. Supondremos que se produce la ``cantidad" necesaria de inversiones para reponer su consumo o uso. En tal sentido serán como cualquier otro bien intermedio. Daremos por dado el nivel de inversiones que se quiera proponer y no estudiaremos los efectos de su variación. [11.15]

3. No se incorporan al modelo los bienes intermedios. [11.16]

   Aumentan el número de ecuaciones e incógnitas por igual, sin aportar nuevos comportamientos de interés.

Relaciones de producción

Podemos pensar que las fábricas de bienes son independientes. Para producir cada bien se utiliza independientemente de los otros, parte de cada uno de los factores.



Fábricas.

`F_o = F_{o1} + F_{o2} + F_{o3}; F_a = F_{a1} + F_{a2} + F_{a3}`

podemos definir coeficientes mu_{ij}, tales que:

`F_{ij} = mu_{ij} F_i; sum_j mu_{ij} = 1`

i puede ser: o u a, y j: 1, 2 o 3.

En términos matriciales:

`mu_o = (mu_{o1}, mu_{o2}, mu_{o3}); mu_a = (mu_{a1}, mu_{a2}, mu_{a3})`

`Factores = F = (F_o, F_a)`

Segundo Postulado: *Se usan cantidades constantes*[11.17] *de cada factor para producir cada bien.*

Para cada tasa de flujo de un bien, necesitaremos usar determinadas tasas de cada factor. [11.18]

El trabajo usa una ``función" de producción por bien, compuesta por tantas ecuaciones como factores existan. [11.1911.20]

Tendremos entonces una ecuación por bien y factor:

`B_1 = c_{o1} F_{o1} = c_{a1} F_{a1}; B_2 = c_{o2} F_{o2} = c_{a2} F_{a2}; B_3 = c_{o3} F_{o3} = c_{a3} F_{a3}`

`c_o = diag(c_{o1}, c_{o2}, c_{o3}); c_a = diag(c_{a1}, c_{a2}, c_{a3})`

`Flujo de consumibles = B = (B_1, B_2, B_3)`

`B = F_a c_a mu_a = F_o c_o mu_o`

definimos además el:

`flujo de bienes = E = (B, F, PI, UI, I)`

El modelo simple: 2 factores, 3 consumibles

El modelo tiene 2 factores y 3 bienes.[11.21] Carece de sentido analizar menos elementos. No es necesario analizar más.

Si producimos un solo bien, éste llegará a consumir totalmente un factor; del otro sobrará y no será factor.

Si producimos dos bienes, la cantidad de cada uno estará totalmente determinada, se consumirá todo lo que se pueda de ambos o sobrará de algún factor. No hay decisión. En ese caso no podemos hablar de economía.

Recién con tres bienes podremos decidir qué cantidad de bienes producir. Aparece el problema económico fundamental: elegir opciones ante recursos escasos.

Un modelo con menos de dos factores no permitiría comparar entre ambos.

Es el modelo más chico que tiene sentido estudiar.

Definición del costo

El costo de un bien es lo que se emplea de cada factor para producir un bien. Se determina inspeccionando las ecuaciones de producción, en nuestro caso, las matrices c_o, c_a.

Es un costo vectorial, pues tiene componentes en cada factor.

Se mide en unidades de factor, por unidad de bien. [11.22]

Decisiones y la preferencia

En cada realidad económica se toman decisiones ante la escasez de recursos y las ``preferencias'' de sus ``actores''. Existirán decisores que ante distintas circunstancias y alternativas ejercerán su cuota del ``poder''. En este trabajo se trabaja como si hubiese un decisor único y global. [11.23] Estas elecciones se modelan con una función $\mathrm{Preferencia}$ de las tres tasas de flujo de bienes. Aplicando la función a cada par de alternativas posibles, se sabe cuál es preferible. [11.24]

La función $\mathrm{Preferencia}$ queda pues expresada[11.25]:

```
Preferencia(B_1, B_2, B_3)
```

Tercer Postulado: *Existe una función $Preferencia$ de los consumibles que representa con mayores valores las elecciones preferidas por los decisores de una realidad económica.*

La visión ultra-simple y la definición de riqueza

Podemos reemplazar en la función de $\mathrm{Preferencia}$ los bienes por las funciones de producción.

Eliminamos los bienes de consumo. Reemplazamos las cantidades de los bienes de consumo por las cantidades de los factores necesarias para generarlos. Y la maximizamos en las variables libres que quedan.

Si reemplazamos los bienes por los factores aplicando las ecuaciones de producción y maximizamos sobre las variables libres nos queda una función a la que llamamos $\mathrm{Riqueza}$: [11.26]

```
P(B_1, B_2, B_3) = Riqueza(F_a, F_o) = R(F_a, F_o)
```

donde P maximiza $\mathrm{Preferencia}$, para cada $\mathrm{F\_a}$ y $\mathrm{F\_o}$ satisfaciendo las ecuaciones de producción.

La $\mathrm{Riqueza}$ está expresada en función de los factores. Y el modelo está simplificado al máximo. Nada se ha perdido pues siempre se puede obtener las variables relativas a los bienes a partir de los factores. Aquí la elección económica se esconde en el paso anterior: la maximización. Sigue existiendo pero en este momento no la vemos.

Precio o renta de los factores

Ya se definieron los costos de los bienes en función de las cantidades consumidas de los factores. Pero ¿cuál es el ``costo" relativo de un factor con otro? ¿Tiene sentido esa pregunta y generalizar el concepto de costo? ¿Es posible encontrar una sola variable escalar que nos indique el ``precio" de un bien? En definitiva, su ``valor" de intercambio. La cuestión del valor es uno de los problemas centrales de la economía. ¿Es posible tener una variable o medida única para comparar el ``valor" de diferentes bienes? ¿Podemos comparar peras con manzanas, en cuanto a su ``valor"?

Para dar una respuesta positiva a estas preguntas plantearemos una cuestión previa que constituye su núcleo: la renta o los precios de los factores.

Para estudiar el precio relativo de los factores calcularemos cómo compensar la caída de un bien con la subida de otro en términos de mantener la función $\mathrm{Riqueza}$ (preferencia).

Se expande como serie la $\mathrm{Riqueza}$:

```
R(F_o,F_a) ~= R(F_{o0},F_{a0}) + (dR/dF_a) dF_a + (dR/dF_o) dF_o
```

Y calculamos cuanto deberemos aumentar un factor, para ``compensar'' la caída de otro en términos de Riqueza.

```
R - R_0 = 0 = (dR/dF_a) dF_a + (dR/dF_o) dF_o
```

donde:

```
R = R(F_o,F_a); R_0 = R(F_{o0},F_{a0})
```

Esto abre la puerta al concepto de intercambio[11.27] y nos da una herramienta para comparar ``peras'' con ``manzanas'', o en nuestro caso ``tierra'' con ``trabajo'', desde la ``visión económica''.

Así la suma de los productos del precio por la cantidad se debe conservar (en balance) para producir un intercambio ``justo'', que mantenga la función Riqueza constante.

```
cte = sum p_i F_i; 0 = sum p_i dF_i
```

Se definen los precios de los factores[11.28] como:

```
p_i = dR/dF_i
```

Sólo indirectamente están relacionados estos precios relativos de los factores con los ``rendimientos'' de las funciones de producción. En primera instancia están claramente determinados por la función Riqueza . Es el hecho supuesto de que la gente se ``sacia'' con la saturación de un bien lo que hace que su precio caiga con el volumen y no los rendimientos decrecientes -que podrían contribuir-.[11.29]

Generalización del precio para todos los bienes

Considerando los costos de los bienes en función de los factores y el precio de los factores, podemos generalizar el concepto de precio y definir un precio para cada bien, sumando el producto del precio de cada factor por la cantidad que el bien lleva de ese factor.

Definición de capital

Podemos definir el Capital que representa determinado flujo o almacenamiento de un bien como el producto de la cantidad de bienes por su precio.[11.30]

Tiene sentido sumar el Capital para distintos bienes y considerar el Capital de un conjunto de bienes o de flujos de bienes (flujo de capital).

También se puede considerar la posibilidad de que se conserve ante intercambios.

La parte variable de la primera aproximación de la Riqueza como una serie es el Capital :

```
R ~= R_0 + p_a dF_a + p_o dF_o
```

```
C = sum p_i F_i
```

Teorema 1: Existe una variable, denominada precio, para cada bien económico, que representa su valor relativo con relación a otros bienes. La suma para todos los bienes de sus flujos por sus precios permanece invariante para intercambios que conservan la Preferencia .

Modelos económicos con consumo acotado

Se puede imaginar una realidad económica donde no tiene sentido consumir más allá de un tope de cada bien.[11.31]

Así la función Riqueza dejará de crecer -o crecerá cada vez menos- a partir de algún valor de cada recurso.[11.32]

A medida que se incrementa la cantidad de factor, se acerca al tope, el precio del factor tiende a cero y éste deja de ser escaso.

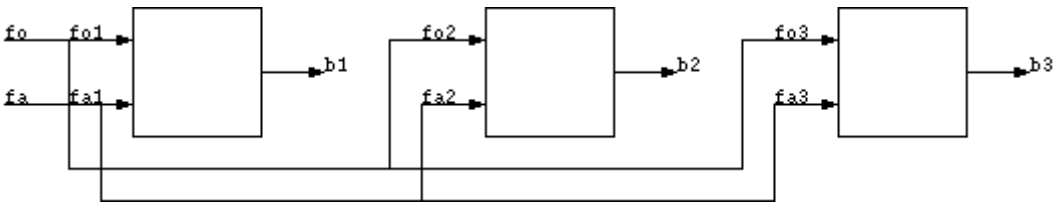
Una función que tiene un comportamiento similar y sirve de ejemplo es:

```
R(F_o,F_a) = 1 - e^{-(F_o+F_a)}
```

A medida que crece cada factor la función tiende asintóticamente a 1.

Para interpretar mejor la función se cambia de coordenadas, utilizando una variable Z, así:

```
R(Z) = 1 - e^{-Z}; p_z(Z) = dR/dZ = e^{-Z}; C(Z) = Z e^{-Z}
```



Cuarto Postulado: *La derivada de la función Preferencia disminuye con incrementos de sus variables.*

La escasez

Teorema 2: En modelos donde el consumo se satura, a medida que un bien resulta menos escaso:

- la Riqueza aumenta, primero rápido, luego cada vez más lento.
- el precio del bien disminuye.
- el capital que constituye, primero aumenta y luego disminuye.

Si no hubiese escasez de un bien, inmediatamente dejaría de serlo; en coherencia con el postulado cero.

Si consiguiésemos energía gratuita -sin límites- a partir del aire, el valor del petróleo, de las compañías eléctricas, etc., desaparecería, una formidable desaparición de enormes volúmenes de capital en el planeta.

Sin embargo tendríamos energía ilimitada y seríamos mucho más ``ricos" en el sentido real de la palabra.

En el planeta, el capital se crea y se destruye permanentemente. Estamos programados para lamentar su destrucción y felicitarnos por su creación y acumulación. Pero esto no tiene por qué ser siempre así.

Sea por mejoras de la productividad [11.33], o por el encuentro de fuentes abundantes, o por una baja de la demanda[11.34], en determinadas condiciones disminuir la escasez, implica disminuir el capital.

Se suele pensar que ``aumentar la circulación de capital" mejora la ``situación de la realidad económica", pero eso sólo es verdad en determinadas situaciones: en entornos de flujos pequeños que no cambian el entorno local de la realidad económica, lejos de la saturación o ante cambios lentos. Con la aceleración de la creación de nuevas tecnologías y en otros casos se deben considerar otras posibilidades. En un contexto de rápido cambio tecnológico y de situación económica, las cosas pueden ser muy diferentes. El Capital ya no mide la Riqueza . Y no siempre el objetivo debe ser aumentar el Capital . La sociedad ideal sería una sin escasez donde todos puedan tener lo que deseen. La sociedad de ``viaje a las estrellas" [Postcarcity:SW]

Por ejemplo, si en las sociedades del conocimiento[11.35] el copyright deja de ser operativo por la libre circulación de contenido [11.36] por Internet, muchas corporaciones verán cómo se evapora su ``capital invertido". El recurso no es más escaso y deja de existir como tal. Pero todos tienen acceso al conocimiento. Ya no es un bien escaso. Lo que antes era escaso, la información, que dependía de un sustrato material para su distribución, hoy ya podría ser libre y no escaso, el único límite es legal y artificial, un sistema legal perimido. Entonces la tecnología ayuda a eliminar escasez, y crea sociedades más ricas, con menos capital. [Moglen:ATS]

Conclusiones

- Es posible sentar las bases para la construcción de una economía política formal:
  - sin considerar criterios de distribución;
  - sin pensar en la propiedad de los bienes, en particular las inversiones, ni en su renta;
  - sin pensar a las ``familias humanas" como fábricas que consumen bienes equivalentes a los que producen con la renta de su trabajo y que coincide con lo estrictamente necesario para reproducirse;
  - sin cerrar sobre sí misma la maquinaria de la ciencia económica; [11.37]
  - sin apelar a los rendimientos decrecientes (pp: ❑) o la idea ``marginalista" para explicar los precios relativos de los factores, estos influyen pero no los determinan. Los precios son construidos principalmente mediante la elección de los decisores;
  - sin considerar a las inversiones como un factor adicional (primera aproximación).
- Retrasar el estudio de las cuestiones distributivas permite tener un marco común para interpretar los planteos conceptuales de las diferentes corrientes ideológicas de la economía política.
- El concepto de la escasez es poderoso para ayudar a interpretar el significado real de la economía neoclásica y entender sus alcances. Comprendiendo los límites de la economía de la escasez podremos plantear una ciencia más general.
- El precio (valor) precisa el sentido del concepto de escasez de un bien y es la variable matemática ``económica" que la representa. Variable común[11.38] a todos los flujos de bienes, se determina a partir de las preferencias de los decisores.
- El trabajo permite pensar como objetivo para las políticas públicas el de disminuir la escasez. Así el mayor progreso sería la no economía, la no existencia de escasez, la superabundancia. Construida a la vez mediante los **tres caminos del progreso económico** unificados e integrados mediante el planteo de este trabajo:
  1. el **gandhiano-ecologista**: usar sólo lo que se necesita. [11.39] Este es el camino ``espiritual" de tener la voluntad de no consumir en exceso, y de mantener el consumo en lo estrictamente razonable.
  2. el **científico-tecnológico**: lograr bienes y servicios, con la mejor eficiencia que el conocimiento permita, concentrando recursos, inversiones y habilidades. Investigar, desarrollar. [11.40] Este es el camino del progreso científico.

3. el **expansivo:** ampliar el uso y la disponibilidad de los factores. Si falta algo, producir más. Pero como esto causa muchas veces daño ambiental, es un camino que cada vez más estará y/o debería estar vedado. [11.41]

Debemos reinterpretar la relación entre las enseñanzas espirituales ecologistas gandhianas, la ciencia, la tecnología y la economía. Las presentes ideas pueden ser útiles a tal fin.

Todo esto parece obvio, sin embargo parece representar una posición muy radical. [11.42]

Apéndices

Definiciones de escasez

- `http://www.auburn.edu/~johnspm/gloss/scarcity.html`
- `http://www.amosweb.com/cgi-bin/gls.pl?fcd=dsp&key=scarcity`
- `http://en.wikipedia.org/wiki/Scarcity`

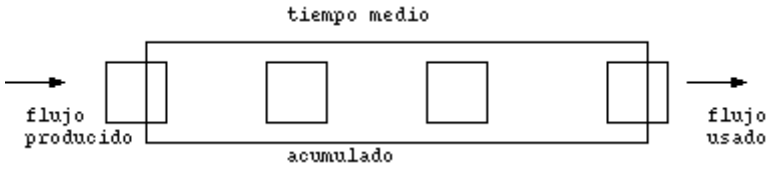
Definiciones de economía

- `http://www.auburn.edu/~johnspm/gloss/economics.html`
- `http://www.google.com/search?hl=en&lr=&ie=UTF-8&oi=defmore&q=define:economy`
- `http://en.wikipedia.org/wiki/Economics`
- `http://en.wikipedia.org/wiki/Political_economy`
- ``La Economía es la ciencia que analiza el comportamiento humano como una relación entre fines dados y medios escasos que tienen usos alternativos". Lionel Robbins, 1932.

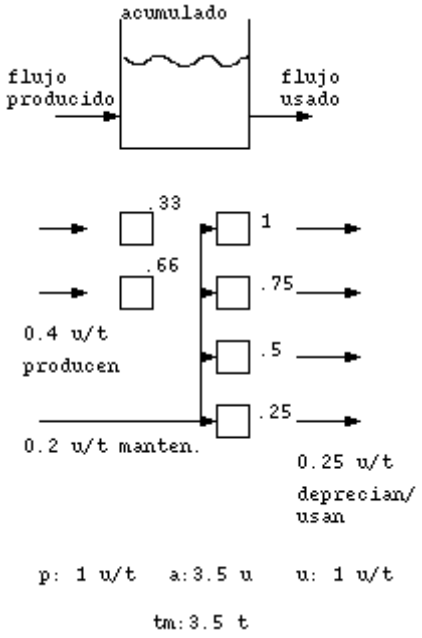
  `http://www.wordiq.com/definition/Lionel_Robbins`
- Mi definición: La economía neoclásica es la ciencia que estudia los comportamientos influidos por la escasez. [11.43]

Flujos, tasas, acumulación y transitorios

El flujo, o cantidad de bienes que circula por unidad de tiempo, se caracteriza por una tasa. Se expresa en unidades de bien por unidad de tiempo.



Flujos y acumulados



Flujo, otra vista

Se puede pensar en el flujo de unidades producidas y en el flujo de unidades utilizadas o consumidas. Si entre la producción y el uso o consumo pasa un tiempo, existirá una acumulación transitoria de los bienes.

```
Flujo * tiempo_medio = Acumulado
```

Si las tasas de los flujos producidos y usados son diferentes, la acumulación variará, así:

```
f_p - f_u = d(Acumulado)/dt
```

Clásicos y neoclásicos

Neoclásicos http://cepa.newschool.edu/het/essays/margrev/ncintro.htm Los economistas de la escuela clásica intentaron fundarla en el concepto de costos. Marx, Stuart Mill, Say, Ricardo, Malthus, Smith, Sraffa

A partir de considerar un solo factor se pretendía calcular el concepto de valor basado en el costo, derivando todos los precios en relación a la cantidad de trabajo acumulado.

Esta idea no funciona, primero porque hay más factores en la economía, existen otros límites y no sólo el trabajo, pero además porque el capital acumulado en cada momento no se puede modificar sino que está predeterminado por la acumulación y la historia y en tal sentido es otro factor escaso, y por ello independiente.

Los neoclásicos inventaron el concepto del ``marginalismo" Gossen, Jevons, Menger, Clark, Robbins, Friedman, Lucas, Walras, Samuelson,

En tal sentido identificaron que las decisiones no dependen solamente del costo de un elemento sino también de la ``utilidad"[Gossen:DLE-54] o el deseo de que ese elemento exista en determinada cantidad. Es decir, la economía no se cierra en sí misma sino que la voluntad de los decisores es importante.

El problema fundamental de esta concepción es considerar que los precios de los bienes están determinados por las derivadas parciales de la función Utilidad, aquí denominada preferencia. El error es que si varía la cantidad consumida de un bien, varía la de los otros, ya que no son variables independientes.

Este trabajo postula que la preferencia determina los precios de los factores independientes, no de los bienes directamente; estos precios surgen de los costos en función de la cantidad de factores de cada uno de ellos.

Así se conjugan aspectos de las teorías clásicas, con la decisión expresada en la función preferencia y sus variaciones marginales referidas a los factores.

Hipatia: Segundo Manifiesto

> OTRO MUNDO ES POSIBLE, y lo estamos construyendo.

> Una sociedad donde se respete la dignidad de las personas requiere que el conocimiento se pueda difundir solidariamente.

Subsecciones

HIPATIA, en su Segundo Manifiesto, reclama la libertad del conocimiento

Y exige que sean respetados los derechos humanos, en particular:

1. el derecho a la cultura libre[12.1],
2. el derecho a la educación[12.2],
3. el derecho a la libre comunicación o expresión[12.3],

derechos cuyo ejercicio es impedido[12.4] por los sistemas normativos de patentes y derechos de autor vigentes [Stallman:APC,Vi:PI-98][12.5].

La creciente preponderancia de estos sistemas normativos frente a los derechos humanos a la cultura, la educación y la comunicación, debe ser limitada debido al interés público y la función social que cumplen estos últimos[12.6], para impedir ponerle un freno al progreso de la Humanidad[12.7].

Debemos construir una sociedad del conocimiento solidaria y sustentable

Corresponde modificar el sistema legal adecuándolo a la realidad, a la conveniencia de la sociedad y a los nuevos usos y costumbres de la red, poniendo en vigencia el derecho a la libertad del conocimiento, conforme a lo establecido por [DUDDHH][12.8].

Así se consolidan los principios éticos que le permiten a la persona difundir su conocimiento, para ayudarse a sí misma, ayudar a su comunidad y al mundo entero, con el fin de que la sociedad toda sea cada vez más libre, equitativa, sustentable y solidaria.

Conforme a esto, Hipatia:

- invita a todas las personas a trabajar para que todas las instituciones, entidades privadas, públicas y especialmente los gobiernos del mundo se manifiesten, participen en la creación y establezcan un marco legal [Ciurcina:NWN-04], adecuado a la realidad, a la conveniencia de la sociedad y a los nuevos usos y costumbres de la red, que permita gozar de la libertad del conocimiento a todas las personas, tal como lo establece la Declaración Universal de Derechos Humanos.

Manifiesto anti-distros

Subsecciones

Introducción

La mayoría de las distros -tal como las conocemos- son al software libre como los cuadros a la pintura y los temas de 7 minutos a la música.

Una forma o formato de distribución diseñado para convertir al Software Libre en producto y capitalizarlo.

Lo que ocasiona la aparición de burocracias, especialmente las comerciales, con metas innecesarias salvo para ellas mismas, y que como toda burocracia busca formas de hacerse imprescindible e impone normas que retrasan o ponen la innovación real en los márgenes del sistema.

El problema de qué distro usar, es lo primero que se plantea una persona u organización que se acerca al movimiento. Un problema que suele ocasionar luchas comerciales y discusiones religiosas. Un problema que crea empresas de servicios que sólo son aptas para trabajar con una distro.

Y es un problema artificial que sólo aparece en aquellas personas que se acercan al movimiento, pero no que debiera ser un problema para el técnico competente.

Los problemas de elección reales e importantes son las otras cientos de opciones a tomar, como qué escritorio usar, qué sistema de archivos en red, qué paquete de oficina, etc.

Una vez decididas, usar una distro u otra no tiene la más mínima relevancia, tanta como elegir de dónde instalar los paquetes.

Discutir qué distro usar es entrar en el juego perverso de las empresas comerciales que asocian su distro con dinero y servicios.

Debemos insistir en que la distro es lo de menos, lo que importa es que la gente conozca los conceptos y que se pueda interoperar de la mejor forma.

¿Es el trabajo de hacer distros o subdistros malo? No, por supuesto que no. Es muy útil tener repositorios (distros) particulares, específicas para cada empresa o adaptadas a los médicos o a las escuelas. Debemos facilitar este trabajo y que cualquiera puede hacerse su distro. Pero esto no tiene por qué generar ramas incompatibles de software.

Me parece genial que cada empresa usuaria, cada organización, cada ``Linux User Group" (LUG), cada persona, tenga su distro, con su menú, su selección de paquetes y sus fondos de pantalla. Ese no es el problema.

Por lo tanto debemos rediseñar el concepto de distro.

Definiciones y elementos

REPOSITORIOS:
: consistentes en uno o más medios conteniendo paquetes. Deben contener el paquete necesario para ejecutar un sistema autónomo y todos los necesarios para ejecutar lo que se desee para el objeto del repositorio.

DISTROS:
: mecanismos, formas, organizaciones creadas para distribuir los paquetes de los repositorios. Materializadas al menos en un medio (en CDROM, disquete o Internet) con un conjunto de paquetes conteniendo ejecutables y archivos de configuración. Probablemente esté preparado para ejecutarse e instalarse.

  Así pues el trabajo que caracteriza a una distro es la compilación de paquetes propios, la forma de empaquetarlos y cómo subdivide el software en paquetes.

SUBDISTROS:
: distros derivadas de otras, que toman los paquetes del repositorio de la distro madre, constituyendo otros repositorios y alterando algunos paquetes, a veces solamente el fondo de pantalla, otras veces scripts, menús, etc.

GENERADORES DE DISTROS:
: mecanismos, procedimientos, protocolos, normas (como las ``Guidelines" de ``Debian" [Debian:SW,Nguyen:CCD-04]) o software para generar distros, como ``Linux from Scratch" [LFS:SW] o ``Gentoo" [Gentoo:SW] y hasta cierto punto ``Metadistros", o el paquete ``apt-get" [Welton:C-00] source de ``Debian".

PAQUETE:
: conjunto de software que no se puede o conviene subdividir, cada paquete tiene ejecutables, archivos de configuración, librerías provistas y requeridas, además de otras cosas, pero estos

estarán identificados a los efectos de su incorporación en los menús y en los instaladores, actualizadores y retardadores

DEPENDENCIAS:

archivos que necesita tener un paquete pero que están en otros, para funcionar (ver dependencias de compilación).

Lo interesante es que la división del software necesario para una computadora en paquetes trae el fenómeno de la dependencia. Entonces no es importante hablar sólo del paquete sino de la forma en que se subdivide el software y se arma la red de dependencias.

La forma de dividir los paquetes es también característica de las distros.

MANEJO DE PAQUETES:

sistema que permite manejar los paquetes en un menú, ejecutar y configurar lo que deba hacerse además de instalar, actualizar y retardar un paquete

SISTEMA AUTÓNOMO:

es un sistema distribuido en un paquete que contiene: núcleo, imagen y software adicional que se instala y puede auto-manejarse y repararse.

SISTEMA QUE COMPILA:

sistema que tiene los paquetes ejecutables que permiten crear paquetes ejecutables o simplemente paquetes. Un directorio donde se agrupan las fuentes y un superrepositorio donde se colocan los paquetes producidos.

SUPER-REPOSITORIO:

de un sistema que compila (o de varios sincronizados) conteniendo todos los paquetes en muchas versiones.

FUENTES:

URL del proyecto original de cada soft, pueden adicionarse parches con lo necesario para que cada paquete funcione como aquí se indica. Siempre se harán esfuerzos para que estos parches sean incorporados al proyecto original. No hay unicidad entre paquetes fuentes y paquetes ejecutables. Es decir 3 fuentes pueden producir 8 ejecutables.

SOFTWARE LIMITADO A UNA DISTRO:

como parte del mecanismo perverso mencionado algunos han confeccionado software con la idea de que sólo funcione en una distro. Por ejemplo Yast. O el conjunto de scripts que inicia una computadora en muchas distros. O la forma de hacer discos compactos ``vivos", o el sistema ``apt-get" en Debian. Por suerte ya hay gente intentando que Yast funcione en Debian o Ututo [Ututo:SW] por ejemplo. Pero hay que entender que este software no es más que paquetes. Artificialmente limitado, pero un paquete más al fin.

Se suele asociar una distro con estos paquetes, pero esto es un error. Un error más inducido por este modelo.

Discusión y propuesta

¿Cuál es el problema con las distros tal como las conocemos?

1. Que favorecen la aparición de un trabajo adicional e independiente centrado en la distro y no en el software original: ``el empaquetamiento". Así, se alienta a un conjunto de desarrolladores a centrar su esfuerzo en el concepto de distro en vez de concentrarse en mejorar cada paquete. Con lo cual se desperdicia y se multiplica el trabajo sin mayor diversidad real.

   ¿Está mal que empaquetar sea importante? No, es un trabajo muy importante. El problema es que se use y se enfoque en separar a la gente por distros. Es importante que se enfoque en cada paquete. Me gustaría hablar de los empaquetadores del Open Office, y no de los empaquetadores Debian o SUSe [Suse:SW].

2. Que tras el concepto de distro se construye software que sólo opera en el marco de esa distro.

3. Algunas distros comerciales, en su afán de vender CDROMs, hacen mucho más compleja la actualización entre versiones, lo que muchas veces resulta en la necesidad de construir rpm específicos por cada versión de la distribución.

4. Que imponen al usuario novato un problema innecesario como esencial: ¿qué distro usar?

La solución

¿Es la solución elegir una distro y abandonar las otras?

No, la solución es cambiar el modelo actual de distros. Sacar el foco de allí y concentrarse en los paquetes. Pensar la forma de disolver el concepto actual de distros y hacer más fácil la vida a la gente.

Hacer que el empaquetamiento sea un trabajo vinculado estrechamente con el desarrollo del software que empaqueta.

Concentrarse en mejorar los miles de paquetes originales para que interoperen mejor entre sí y se instalen directamente desde cada uno de sus sitios. ¿Para qué tener 1000 personas trabajando en paquetes Debian, cuando directamente podrían mejorar los ``Makefile" de los originales? Así devuelven al proyecto lo que usan; me parece más racional y funcional al modelo de software libre. Lo mismo para las otras distros. Los ``Makefile" de cada proyecto pueden producir los paquetes necesarios para que se usen, tales como deb, rpm, tgz, sh o incluso paquetes completos o que separen documentación, ejecutables y desarrollo.

Armar repositorios que los agrupen y discos compactos que distribuyan esos repositorios, originados en el sitio base de cada proyecto.

Hacer que la diferencia entre un paquete deb y un rpm sea sólo de formato, como puede ser la diferencia entre un gz y un bz2. Ambos se abren con un tar -algo[13.1].

Hacer un ``particionador'' general, un instalador general y un vivificador (para CDROMs vivos) general.

Lo ideal es que el paquete pueda generarse e instalarse con los ``Makefile'' de cada software original. Así cada desarrollador y los empaquetadores vinculados producirán los ``Makefile'' apropiados para que se generen los paquetes que se instalen apropiadamente en una computadora.

Diseñar instaladores, sistemas de menús y demás que puedan interoperar con los paquetes tal como estén en sus repositorios.

Respuestas a las preguntas frecuentes. FAQ

¿Entonces cobrar es malo? ¿Qué hace la gente con las facturas a final de mes? La misma competencia se puede dar entre cooperativas. ¿O es que el Software Libre sólo puede serlo gracias al subsidio estatal?

Para nada es malo. La cuestión es que las formas que se usen para cobrar no aten a la gente.

El concepto de distro actual está comercialmente pensado para atar a la gente con el proveedor de esa distro. En cierta forma el concepto de producto armado de esa forma no es demasiado compatible con la filosofía del software libre. Pues te ata a determinada empresa o grupo.

La filosofía de prestación de servicios en un entorno libre implica que puedas cobrar (mucho o poco) por tu servicio, pero que ese servicio no obligue al servido a seguir atado a ti.

La división de comunidad-empresas en grupos excluyentes, cada una con una distro es absolutamente artificial y la podemos superar. Es tan difícil aprender Debian, si sabes bien SUSe, como aprender ``postfix'' si sabes ``sendmail''. O sea el contenido de conocimiento adicional que representa una distro es poco, comparado con el conocimiento de base que uno puede tener con relación al sistema GNU/Linux.

Coincido contigo, ahora, lo cierto es que elegir una distribución u otra tendrá fuertes implicaciones. No en balde los creadores de las distribuciones vieron cada uno un beneficio en ``forkear'' y crear una propia.

No es ``la elección de distro'' la única opción a tomar, diría que es la menos importante.

Es falso el problema de elegir ``la distro''. Es originado por aquellos que hacen de ``la distro'' su producto.

Para instalar un sistema con GNU/Linux, se deben tomar cientos de decisiones. Elegir ``la distro'' no te hace avanzar mucho en ello.

Qué usar: ``KDE'', ``GNOME'', ``Blackbox''?, ``sendmail'' o ``postfix''?, qué particiones usar?, qué fondo de pantalla?, qué tiene el menú?, qué paquetes selecciono para cada estación y servidor?, ``nfs'', ``samba'', ``openafs'' o ``gfs''?, ``nis'', ``ldap'', ``kerberos'' o una combinación?, ``openwebmail'' o ``squirrel'' u otros?, ``firefox'', ``konqueror'' u otros?, ``evolution'', ``kmail'', o sólo usar ``webmail''?, instalo las máquinas desde un repositorio o preparo un CDROM?, y cientos de posibilidades más.

No estoy de acuerdo, la variedad de distribuciones es justamente un punto fuerte en GNU/Linux. Me parece bárbaro crear estándares, consorcios y ese tipo de cosas. Pero matar la diversidad, sería un error estratégico muy grande. Sí creo que se deberían crear estándares tan buenos que permitan convertir un RPM a un DEB de manera que se pueda usar para fines productivos, por poner un ejemplo.

Entonces estamos de acuerdo.

Si eso pasara y tuviésemos conversión transparente de deb a rpm y de rpm entre sí y eventualmente deb entre sí (Ubuntu [Ubuntu:SW] - Debian por ejemplo), las distros no serían relevantes, y ese es el punto, justamente.

Y no planteo crear un consorcio, solamente meter la idea en las cabezas de los desarrolladores y empaquetadores para que poco a poco la cosa vaya por ese camino, bueno creo que ya está pasando, a lo sumo estoy dándome cuenta de una tendencia, no sé si otro lo habrá pensado en estos términos o descripto. Nada nuevo bajo el sol, sólo que puesto en palabras fuertes para hacer un poco de olas.

La aparición del concepto de distro fue un aporte muy grande al movimiento, un paso necesario, pero creo que ya podemos abandonarlo.

Como en la selección natural (explosión cámbrica), se produjo una explosión de diversidad, apenas se vio la importancia del empaquetamiento y se usó ese fenómeno para crear marcas comerciales: RedHat [RedHat:SW]. La respuesta de la comunidad no se hizo esperar y fue excelente: Debian.

Pero llega un momento cuando ya se conoce mucho sobre un tema que la innovación se mueve a otro lugar y es razonable que la diversidad ceda y queden las buenas prácticas y los estándares sobrevivientes. Esto empieza a pasar en el área del empaquetamiento. A nadie se le ocurriría plantear un sistema de paquetes que no sea tan útil como el ``apt-get'', simplemente es hora de que esos sistemas, los varios que puedan quedar, manejen todos los formatos de paquetes e interoperen con cualquier repositorio.

Frente al muchas veces trillado cuestionamiento de ¿para qué otra?, la respuesta es: ¿por qué no? Si la distribución es mala o poco popular, desaparecerá o la usará poca gente. Por otro lado nadie puede asegurar que un fulanito que hace una distro XYZ quiera sumarse a una única distribución, así que no se pierde nada al fin y al cabo. La libertad está en poder tener la fuente, la diversidad hace a la acción de libertad.

De acuerdo, pero si la base de las distros es interoperable todos ganaremos mucho. O sea, lo que combato es el concepto de paquetes no interoperables, más que lo que tú llamas distro y yo subdistro, y que en el futuro pueda quedar como distro.

Los humanos poco a poco sacamos conclusiones sobre cuáles son las ``buenas prácticas'' y tendemos a ellas y creo que una buena práctica para la comunidad sería trabajar en un esquema de paquetes compatibles entre los diferentes sistemas. ¿Es esto muy difícil de lograr? ¿Requiere consensos, discusiones, consorcios? NO, ESO ES LO BUENO, sólo requiere centrar el empaquetamiento en el desarrollo de cada paquete original. Está sucediendo naturalmente.

No entendí muy bien el tema de convertir RPM a DEB y supongo que viceversa.

Los dos tipos de paquetes básicamente guardan la misma información.

El problema es cómo dividas el universo de software en paquetes y cómo organizas las dependencias.

Eso hace que sea difícil convertir un paquete en otro y que un rpm no sea compatible incluso con otro. Los de RedHat con los de SUSe por ejemplo.

Otro factor que entorpece el trabajo es la necesidad de las distros comerciales de vender CDROMs, que hace que no provean formas fáciles de actualizar todo a las nuevas versiones, sino que proveen parches dentro de cada versión. Esto crea una nueva barrera artificial paquetes para la SUSe 9.0 y otros para la 9.3, incompatibles entre sí.

Debian lo pudo resolver sin problemas con sus ``upgrades'', pues no tiene interés en vender CDROMs.

La otra cuestión fundamental es que en las dependencias de los paquetes se ponen otros paquetes, en vez de archivos reales, lo que facilitaría buscar dependencias en paquetes de otra distro.

¿Y perder lo mejor de Debian? El problema de ``Makefile'' es que no verifica y baja dependencias, es una herramienta ``estática'' (si las cosas no están donde vos pedís, las cosas no andan ... sí, sí, usa ``autotools'' de última

¿Usar debs de otras distros? ¿Cómo cuáles? Si todos usan los de Debian. Podrías decir Ubuntu, pero no aplica. El ``main'' de Ubuntu es propio de Ubuntu, si bien está basado originalmente en paquetes de Debian, siguen su propio camino. En el caso de Universe se toma de Debian y se compila. Si se agregan cosas nuevas, ahora está Debian Ubuntu que pretende hacer lo contrario, tomar lo que tiene Ubuntu y aplicarlo en Debian.

No, no. Me refiero poner en los ``Makefile'' originales la posibilidad de hacer los paquetes deb o rpm o tgz o todos ellos.

ahh, creo que ahora voy entendiendo a dónde apuntas. Sería algo como hacer : ``make deb'', ``make rpm''. Eso suena bien y razonable. Un poco complejo tal vez para el programador, debería

haber herramientas (macros ``m4'', ``alien'') que funcionen más ``inteligentemente'', entre otras cosas).

Exacto. Lo pueden hacer los empaquetadores también. Lo ideal sería ``make dist-rpm" o ``make dist-deb"

Incluso una opción intermedia sería ``dist-rpm-RedHat", ``dist-rpm-SUSe" o ``dist-deb-Ubuntu" llegado el caso, aunque sin duda lo mejor sería un rpm y un deb que hagan paquetes ``iguales" en el sentido de las dependencias y totalmente transformables uno en otro.

Tu idea se llama Gentoo

Quizás se llame ``autotools" y ``make dist".

Estoy de acuerdo en que Gentoo dio un paso en esa dirección, un paso más que ``Linux from scratch".

Y Ututo complementó la idea pues provee lo que Gentoo no quiso, ejecutables. Así Ututo es una distribución y Gentoo un mecanismo para hacer distribuciones.

Pero creo que hay margen para avanzar más y si muchos nos metemos el tema en la cabeza podemos llegar en algún tiempo a anular la diversidad que es como azúcar sintáctica, es decir diversidad inconducente.

Toda esta historia empezó con los rpm, gran invento, pero que permitió separar a RedHat y construirla como empresa, lo que derivó en la respuesta deb y el ``apt", base de identidad de Debian y su gran invención.

Ahora, sería bueno apuntar a consolidar esas ideas y absorberlas en el movimiento global, de tal forma que en el futuro sea indiferente que instales un paquete desde deb o rpm, así como es indiferente que instales desde un tar con bz2 o gzip, o que podés usar en una consola gráfica GNOME y en otra KDE en la misma computadora.

También es interesante apuntar a consolidar los instaladores, que puedan interoperar Yast con Webmin y otros en cualquier distro.

Es decir ortogonalizar el espacio decisorio.

En cierta forma sería justo decir que mi idea se llama slackware [Slackware:SW] o incluso SLS [Wikipedia:SLSD]. Tomar los ejecutables tal como los generan los desarrolladores, empaquetarlos y meterlos a un repositorio.

El concepto de paquete con dependencias es una sofisticación importante pero debió haber seguido en los paquetes originales y no como capa que justifique el concepto de distribución.

Si cada paquete expone los archivos de librerías y de configuración que provee y los que requiere ... Cada repositorio podría tener su base de datos con esa información, que podría ser consultada por los

manejadores de paquetes. Nada nuevo bajo el sol. Podría también combinarse con los ./configure ... Es decir de la misma manera que el programador del paquete programa las ``autotools'' para saber que ``include'' y librerías pedir, puede esa misma información usarse a la hora de instalar el paquete.

Así no es necesario indicar el paquete X requiere el paquete Y. Sino X requiere yyy.1.25.so por un lado y por otro Y provee yyy.1.25.so.

Entonces los repositorios serían mucho más versátiles.

Buscar una solución intermedia a corto plazo no me parece mal, como es el caso de tener mantenedores de paquetes.

Desde mi punto de vista lo ideal es no concentrar en distros la capa de mantenedores de paquetes, sean deb o rpm, sino incorporar ese trabajo en los paquetes originales.

Y que Debian y cualquier distro se arme en base a los ``Makefile'' de los sitios originales o incluso los deb o los rpm de otras distros.

¿Y que ``ese'' GNU/Linux tenga repositorios con todos los paquetes y fuentes GNU que andan por ahí? Si mi interpretación es correcta ... entonces

Según lo que entendí en otro correo, no es del todo correcta. El plantea que el paquete de ``upstream'' (el que hace el programa) ya esté listo para poder meterse en cualquier distro.

Concentrar el esfuerzo en las fuentes originales.

Lo que traerá como consecuencia que las distros dejen de ser relevantes y sean sólo repositorios distintos con cambios menores y selecciones de diferentes paquetes. Eventualmente alguna especializada en un procesador o arquitectura.

Siempre será bueno tener un CDROM listo para instalar una distro musical o una interesante para los médicos, o una compilada para 486, qué sé yo.

Pero todo tenderá a estandarizarse más, al menos en la base.

El problema es que ``upstream'' no suele empaquetar, por costumbre o desinterés. Es por eso que los mantenedores aparecen. No sé si está bien o mal, pero a mí personalmente me agrada andar haciendo paquetes, hay desafíos muy interesantes.

Sin duda, entonces devolvamos esas contribuciones al paquete original.

La voluntad dentro de Debian no falta, para hacer lo que pides, los parches o ``bugs'' son reportados a ``upstream'', el problema suele ser que hay ``upstreams'' y ``upstreams''. Algunos son menos ``buena onda'' y otros son copados y ayudan aplicando estos parches, otros se resisten. Como ejemplos se me ocurre ahora GTK/GNOME (también vale aclarar que varios ``Debian

developers'' (DD) son ``GNOME developers'' y como mal ``upstream'' está el de ``centericq'' por ejemplo (tipo totalmente ególatra que no acepta un solo parche).

Hay muchas veces que los ``Makefile'' rompen los ``policy'' (acá entraríamos en otro tema, ``un-generic policy'' por ejemplo :D). Un caso que estoy tratando justo ahora es un paquete para Ubuntu. El autor tiene en el ``Makefile'' que instale los documentos ``INSTALL'', ``ChangeLog'' , etc en /usr/share/nombre. El ``policy'' de Debian dice que ``INSTALL'' no debe copiarse, pues no tiene sentido. ``ChangeLog'' debe ir comprimido con gzip o bzip2, por lo que tuve que parchar el ``Makefile'' para que el paquete cumpla dicho ``policy''. Este es sólo un pequeño ejemplo y fácil de resolver porque usa ``autotools'', pero hay casos donde el autor provee directamente el ``Makefile'' hecho a mano y es mucho más difícil sacar un paquete que cumpla todo al pie de la letra.

¿Por qué no mejorar los ``Makefile'' de cada proyecto?

Sería bueno generar un parche con todos los cambios propuestos, incluso un ``Makefile'' decente, cuando no exista. Entonces no existiría un paquete Debian fuente, sino un parche Debian fuente. Si el autor original no lo acepta, seguirán estando esos dos un buen tiempo.

Ojo, los DD no sólo hacen paquetes, también hacen la distro completa (son los únicos autorizados a hacer ``commits'' directos de códigos del proyecto). Lo que sí hacen los DD es aprobar los ``uploads'' de aquellos que no son DDs, para garantizar su calidad.

Claro, pero ¿qué es una distro si no la suma de paquetes? Para lo que tú dices, ¿por qué no hacer paquetes generales?

Estoy de acuerdo en la idea, sería ideal (y hasta utópico) poder plantearlo a todas las distros y que sea aceptado (hasta diría que sería una mini-guerra civil :D). Desgraciadamente no lo tenemos y personalmente creo que nunca lo tendremos (o por lo menos no creo vivir para verlo :).

Hay un sitio que busca hacer rpm genéricos, sería interesante plantear una cooperación Debian y ese sitio para sacar rpm y deb con las mismas dependencias y forma de subdividir un paquete.

[OpenPKG:SW]

Opinión personal: las cosas están como están porque a cada uno le gusta hacerlo a su manera.

Y sí.

Ese es el punto, que la gente ha focalizado la ``marca'' en la cuestión distro, las comerciales por cuestiones de dinero, pues instalaron la idea de distro-producto.

Y Debian como respuesta a ese fenómeno concentró su esfuerzo en la propia marca comunitaria. Si 1000 personas se agruparon en Debian con su idea, también pueden ponerse de acuerdo en otras.

Creo que hoy en día se podría concentrar el esfuerzo en disolver ese foco y plantearlo en otro lado, un buen instalador general -algo como 123L que propuse- algún manejador de paquetes genérico, metadistros, etc

Es cuestión de ir proponiendo la idea y poco a poco irá sucediendo -si la idea sirve, claro- a medida que más gente ``compre'' esa idea.

pero coincido que en general todas las subDebian no son más que eso: subdistros, quizás Ubuntu pueda llegar a ser otra distro.

Knoppix se pondría como otra distro bastante alejada de Debian. Si bien usa muchísimo, el autor ha hecho grandes cambios en la parte de ``booteo'' (mucho relacionado con la detección de hardware principalmente).

Claro, y sería bueno que eso sea -y probablemente lo sea- un paquete más que en algún momento pueda adoptar Debian u otra distro.

Así como alguien quiere portar Yast para Debian y Arturo lo está haciendo para Ututo.

Una idea interesante es ``sourceinstall'' [Fontana:GSI-05], aunque si se leen sus contras se nota que sería bueno que interopere con un repositorio y que sea automatizable.

tal como hace ``apt-get source''

Aún si hubiese un solo particionador y un solo instalador y un solo todo, habría distros.

Depende de qué llames como distro.

Yo en general llamo distro a un conjunto de paquetes ejecutables. Así por ejemplo Debian es una distro, con cientos de variantes que se diferencian en al menos un paquete: el fondo de pantalla, el menú y algunos agregados.

Es decir hay tres o cuatro distros importantes que concentran trabajo real al menos de compilación: SUSe, RedHat, Debian, Ututo, Ubuntu, Mandriva [Mandriva:SW] quizás alguna más.

Algunas de ellas además se toman el trabajo de hacer y mejorar muchos paquetes, pero eso lo tomo como trabajo de hacer un paquete. SUSe con Yast, o los scripts de arranque, etc. O Debian con ``apt''. Sería bueno que los desarrolladores de esos paquetes se saquen su ``distro'' de la cabeza y los piensen para cualquier PC con GNU/Linux. Su aparición fue necesaria, no había otra forma de hacerlo, ahora hemos progresado mucho.

Debian, SUSe, RedHat, Gentoo, Ututo y seguramente otras, tienen además el trabajo de empaquetado, que mantienen y construyen, lo que es un trabajo enorme, lo que propongo es que ese trabajo se vuelque en los paquetes originales y sea compartido por todos.

¿Qué quedaría para las distros? El trabajo de compilar, y decidir qué paquete meten en el CDROM y hacer los menús.

O sea un trabajo duro: ``compilar'', pero relativamente simple y un trabajo estético y de toma de decisiones.

Entonces las distros tendrían mucha menos relevancia.

Por un lado aparecerían miles de distros, una por cada empresa por ejemplo.

Pero por otro lado todas las distros serían mucho más compatibles. Hasta es probable que sólo quede un conjunto de rpms y otro de debs y que finalmente sean equivalentes incluso a nivel de dependencias.

Si yo quiero un SO ``listo para instalar'' que tenga básicamente software para oficina, y vos querés hacer uno centrado en software para programadores, ya tenemos dos distros...

Claro, lo armás y listo, o te lo bajás armado,

O repositorios con diferentes paquetes en un CDROM. Hoy ya con cualquier distro podés hacer eso.

Repositorio podría haber uno solo (difícilmente, pero bueh imaginemos), pero la idea de distro está bastante asociada a un medio de distribución físico: no puedo meter en un solo CDROM (o DVD) todo el repositorio de todo el software libre que existe, y aún si distribuyera no sé cuántos DVDs con todo, no sería práctico.

Claro, la cuestión es no centrar el asunto en la ``distro'', que deje de ser relevante.

Obviamente que todo sería mucho más cómodo si esa fuese la ÚNICA diferencia entre distros (y no existiesen las otras: escritorio GNOME/KDE/etc, paquetes dpkg/rpm/tar, etc)

En varias distros actuales tenés eso como opciones, son parte de los cientos de opciones que tenés al instalar un equipo.

Sobre la inutilidad de la guerra de las distros

Con respecto al tema guerra de distros, creo que estas discusiones son muy pertinentes.

Ayudan a todos nosotros a conocer las impresiones que podemos tener sobre cada distro y sus fortalezas y debilidades. Así podemos lograr sacar las buenas ideas y utilizarlas en todas.

Instalador independiente de la distro. Sistema 123L

Especificaciones estándar para distribuir sistemas GNU/Linux

El proceso de instalación de GNU/Linux ha sido siempre uno de los factores más complejos a la hora de su difusión. Se propone un esquema que solucione muchos de esos problemas.

Características

- Modular, con pasos o módulos cortos, ortogonales y fácilmente repetibles e investigables si surgen problemas, con test que permitan determinar si el paso se cumplió y por qué no, si hubo fallos.
- 3 pasos fundamentales, por lo que se puede denominar 1 2 3 y listo, o PIA: ``particionar", instalar, arrancar.
  - **PENSAR**.
  - 1: **``Particionar''**.
  - 2: **Instalar** GNU/Linux autónomo.
  - 3: **Arrancar**, instalar el arranque (``boot") e iniciar el sistema.
  - 4. **LISTO**, usar y/o hacer crecer GNU/Linux autónomo hasta la configuración deseada.

  Los 3 pasos, controlados por un ``Makefile".
- Uso mínimo de intérpretes y librerías, para que pueda funcionar en sistemas GNU/Linux chicos.

  Debe poder ser ejecutado desde cualquier tipo de sistema y circunstancia. Desde distros vivas completas, o una distro viva instaladora específica, o un GNU/Linux ya instalado en la máquina para instalar el GNU/Linux objetivo o para reparar o modificar algo.

  El sistema debe poder instalar cualquier GNU/Linux autónomo.

  ¿Qué es un GNU/Linux autónomo?

  Un sistema GNU/Linux que tenga lo mínimo imprescindible para ``bootear" y administrarse solo, en particular este ``Makefile".

  Consistirá en un paquete: núcleo + imagen inicial + programas básicos.

  El núcleo, la imagen, y los programas básicos deberán caber en un disquete cada uno, y deberán poder ser copiados de allí. O sea el paquete núcleo+imagen+pb debe poder estar en tres disquetes o menos.

  Todo estará contenido en algún paquete, la distro viva instaladora, sus árboles, etc. Todo se debe poder constituir agrupando paquetes y procesándolos con scripts autocontenidos.

  La instalación del GNU/Linux autónomo consiste en copiar el núcleo, la imagen e instalar en las particiones designadas los programas mínimos para su funcionamiento y autonomía.

  Este conjunto es un paquete y se actualiza junto. Cada vez que se instala, actualiza o retrasa se hace con el ``Makefile" del sistema 123L.

El núcleo+imagen inicial+pb: sistema GNU/Linux autónomo es un solo paquete.

Los paquetes son las unidades mínimas que deben instalarse sin poder separarse, si se instala, actualiza o retrasa se hace en bloques.

El sistema ``vivo'' instalador también. Puede instalar su propio núcleo+imagen+pb, o instalar otras de otros disquetes. Esto significa que el núcleo del disquete y su imagen del sistema vivo instalador deben ser en sí un sistema GNU/Linux autónomo.

El sistema podrá montar un conjunto de árboles opcionales (directorios y puntos de montaje) de existir.

Este sistema autónomo ofrece siempre múltiples consolas, una de ellas ingresa automáticamente siempre un usuario (se puede además hacer manual) que despliega un programa básico de instalación, actualización y retraso de paquete. Este programa básico usa como dato el nombre del paquete (el mismo que se utiliza en el menú, para agrupar a todos los ejecutables y archivos de configuración del mismo) y sin información adicional debe poder ejecutar el programa predesignado del mismo, o instalar el paquete si no lo está. Si hay en repositorios disponibles una actualización ofrecerla. Con comandos adicionales se podrá volver a una versión anterior, o eliminarlo. Cualquiera de estas opciones debe llevar a una actualización o retraso de sus dependencias.

Pueden existir varios paquetes con el mismo nombre; además de las versiones, pueden tener características diferenciales, como procesador para el que están optimizados, o diferentes opciones (``ldap'' con ``kerberos'' por ejemplo)

- Una vez instalado (en el paso listo) el sistema ya funciona y tendrá un sistema de incorporación de paquetes y actualización (o retraso). Este sistema funcionará contra cualquier repositorio, sea por red o por CDROM.
- El sistema 123L se podrá instalar en cualquier GNU/Linux.
- En todo momento el sistema debe saber qué repositorios tiene disponibles. Si coloco un CDROM con un repositorio el sistema debe notarlo.

  Los repositorios serán coherentes, es decir tendrán paquetes que interoperarán entre sí y todo paquete tendrá en el repositorio todo lo que necesita para funcionar. Los repositorios podrán ser múltiples, es decir basarse en varios medios. Un CDROM puede basarse en un repositorio en la red. Si pongo un CDROM de este tipo y no hay red, ese repositorio no se activa.
- El sistema de actualización debe copiar todos los paquetes necesarios para dejar operable al sistema en un caché local seguro antes de proceder. (no red, no CDROM) Si debe instalar más que lo que puede almacenar el caché deberá dividir la tarea adecuándose al tamaño del caché.

Así:

- Eliminar los paquetes que no se vayan a usar.
- Copiar los primeros paquetes coherentes e instalarlos.
- Copiar otra tanda y volver a repetir.

Nunca el sistema debe quedar con su sistema autónomo incompleto o con paquetes no ejecutables.

Las tareas pautadas de este tipo deben quedar en un ``Makefile", cosa de que cualquier problema se pueda retomar desde el paso faltante.

Así, si se determinan 10 operaciones de caché e instalación, cualquiera de las 10 deben poder retomarse después de una colgada.

Todos los archivos de configuración (los indicados o los modificados a mano) de un paquete desinstalado deben preservarse, eventualmente el paquete podrá tener medios de decidir cómo autoconfigurarse con esto.

Existirá un directorio con estos archivos viejos, clasificados por nombre del super-repositorio origen, nombre del paquete, versión del paquete y versión del cambio de la configuración usando un sistema simplificado a la ``cvs".

Así estas distros tendrán un sistema elemental de control de cambios.

- Los datos de usuario: ``home", ``var", ``root", ``srv", ``usr/local" y archivos de configuración alterados, se preservan en todo cambio.

  Para sistemas vivos se pueden preparar esquemas especiales para almacenar estos directorios.

- Los paquetes deben tener deltas. No hay versiones de los sistemas. El super-repositorio contiene todos los paquetes en todas las versiones y los provee junto con sus deltas. Es posible construir muchísimos sistemas diferentes.

Sistema compilador

Cada sistema compilador debe imponer un nombre a sus paquetes, de haber varios coordinados pueden usar el mismo nombre. Por defecto un sistema genera paquetes con su propio nombre.

Cada repositorio usa paquetes producidos en sistemas con mismo nombre. Cada GNU/Linux instalado usa paquetes con el mismo nombre.

Cambiar el nombre implica eliminar todos los paquetes incluido el base e instalar otro.

Los archivos de configuración son preservados en el directorio especial de las particiones.

Cuando se producen paquetes para autoconsumo el super-repositorio local es un medio más del repositorio en uso y se producen paquetes del mismo nombre.

Las dependencias deben ser a archivos mínimos de librerías, algo que permita la ejecución del programa, luego los problemas deben ser resueltos por el sistema de errores y de configuración de cada programa.

El sistema de paquetes debe entonces instalar cada paquete y los paquetes con sus librerías. Nada más.

Un sistema que administra Linux es otra cosa y para cada función debe instalar los paquetes para su operación y elegirlos entre las alternativas. Esto está a un nivel superior.

El sistema de paquetes debe poder instalar paquetes de cualquier distro en cualquier instalación; para ello debe conocer los paquetes disponibles en cada repositorio habilitado (incluso el filesystem de otra distro) y tener una BD de los paquetes del repositorio, conteniendo qué librerías provee cada uno y qué necesita.

Cada instalación debe tener también una base de datos de paquetes instalados diciendo qué paquete modificó cada archivo.

Por otro lado debe tener un svk de la configuración de la máquina. Rama estable, rama test.

Sistema de menú

Cada paquete contendrá la información sobre en qué rama del menú colocar sus entradas.

Son entradas, una por defecto con el programa principal.

Otras con otros ejecutables del paquete

Otras con los archivos de configuración editables, enlazadas con el editor de referencia. Lo ideal sería que todos los archivos de configuración de todos los paquetes tengan un formato único al estilo de variables ``bash"[13.2]. Mientras tanto un sistema de administración puede tener un conjunto de estos archivos en paralelo con medios de edición de los mismos y de generación-lectura de los archivos de configuración reales.

Otra con la posibilidad de remover el paquete.

Otra con la posibilidad de actualizarlo cuando exista.

Cada paquete además indicará qué librerías provee.

Así listará: librerías provistas, ejecutables provistos, archivos de configuración provistos.

Un sistema para manejar menús podrá acomodar todo el software de los repositorios disponibles. Si se pide ejecutar algo no instalado se instala.

El menú podrá ser ejecutado en un navegador ``web", adaptado para ejecutar, y en terminales de texto o gráficas.

El problema fundamental con los rpm y deb es que mezclan una especificación de contenido con una especificación de formato.

El tar.gz o bz2 puede contener todas las especificaciones y shells que se necesiten.

La misión del instalador es que el ejecutable corra; es misión del desarrollador que, una vez que se ejecute, tenga todos los archivos y genere mensajes de error; es misión del administrador del sistema instalar el conjunto de paquetes adecuado para una tarea.

download, patch and untar compconfigure (leer script a partir variable) ejecuta configure make make pkg make install (aunque no use paquete toca base de datos) make execonfig (a partir variables bash)

urpmi Mandrake

dpkg-deb dpkg dselect apt

apt4rpm apt-rpm

ebuild un paquete emerge dependencias portage sofisticado

Referencias varias: [Wikipedia:LD] [LinuxISO:SW] [Wikipedia:Ptg] [FHS:SW] [Wikipedia:CLD] [Distrowatch:SW] [LWN:LDL] [Distromania:SW] [LinuxQuestions:DR] [FrozenTech:LL] [LMP:SW] [LDC:SW] [FreeStandars:SW] [LinuxBase:SW] [Gentoo:WWP] [Gentoo:EH] [Gentoo:API] [Hamilton:SUT] [Bailey:MR-97] [Vaughan:GAA-00] [Kitenet:ACD] [Cortes:ACP] [apt4rmp:FAQ] [Kojima:RPA] [Bodnar:WWR] [Raymond:SRP] [FreeBsd:FPH] [Miller:RMC] [Crasta:EPW] [GNU:GL] [Uekawa:DLP:02] [LinuxQuestions:LRC] [Vairagade:MPS-03] [Poirier:PSR-01] [Autoconf:AMA] [Yamato:AP] [Sweet:SDU-99] [rpmorg:SW] [Banks:M] [Worth:SAT] [Knight:SUG-04] [Debian:HMP] [Jackson:DPM-96] [Benson:P]

Guía de (auto) aprendizaje en Software Libre

Subsecciones

Introducción: Currículo para la formación profesional

Se presenta un currículo, constituido por módulos, que tiene como objetivo ser útil en la (auto) formación de sus estudiantes en la cultura, valores, filosofía, ética, principios, técnicas, prácticas, ideas y conocimientos en general, desarrollados por la comunidad del software libre en estas últimas décadas y que se consideran necesarios para la efectiva inclusión, participación y colaboración en la construcción de la misma.

El currículo se encuentra estructurado en módulos (una clase) que pueden ser usados para construir cursos, los que pueden organizarse en diferentes planes de estudio. Se presenta un conjunto posible de cursos. Cada curso puede tener módulos optativos, sea para dar contenidos diversos o de nivel especial.

Está previsto que cada módulo pueda dictarse en un tiempo entre 30 y 45 minutos, más un tiempo de práctica asistida, práctica a solas y evaluación.

Cada módulo se describe mediante párrafos, que tienen conjuntos de sentencias. Cada sentencia, párrafo o módulo puede tener su propia bibliografía.

Cada módulo tiene un código e indica de qué otros módulos es correlativo. Lista los comandos, programas o archivos especiales involucrados. Indica actividades especiales y contiene notas.

Se puede variar el dictado de cada módulo para hacer cursos acelerados o para profundizar en algunos de sus aspectos. También puede darse todas las alternativas para cada función o sólo algunas.

Cada módulo contiene temas que se disponen cada uno en un párrafo en la presente. Algunos párrafos pueden ser opcionales, otros pueden contener diversas alternativas que no es necesario cubrirlas todas. Otros párrafos pueden ser dados sólo en cursos especiales en los casos en que se aspira a profundizar la formación. Estas situaciones se indican con diferentes leyendas.

Filosofía

Todo el contenido se basa en los conceptos teóricos que lo sustentan y en la capacidad práctica y efectiva de instrumentarlos. Se hace énfasis en la relación causa - efecto y en el análisis de las razones de los hechos. La correcta e inteligente aplicación de los fundamentos a las necesidades prácticas es sin duda uno de los fuertes de la cultura UNIX. No se trata de adquirir conocimientos especiales o esotéricos mediante recetas, sino aplicar la lógica y la experiencia al conocimiento de los fundamentos y la filosofía subyacente.

Una de las características de los sistemas libres es que exponen todo su funcionamiento a la vista, por lo que cada suceso puede ser rastreado a sus consecuencias. Esto hace que un mayor conocimiento permita mejorar considerablemente la calidad de su utilización.

Independencia

El cumplimiento de los estándares POSIX es lo que da a los diferentes UNIX y GNU/Linux existentes una gran unicidad en su inherente diversidad. Este currículo no hace distinción entre distribuciones.

Valores, prácticas y principios del currículo

Para transmitir los conocimientos que hacen del software libre una comunidad vibrante y del GNU/Linux un sistema potente:

- aprender haciendo,
- desafiante y de alto valor,
- enfoque en los conceptos, y de allí en las técnicas, herramientas y prácticas,
- capacidad autodidacta y autoformativa, pero colaborativa, para una cultura no habituada al sistema educativo convencional.

Organización de módulos

Listado, códigos y correlatividades de los módulos.

El número corresponde al curso propuesto para incluirlos, cuando tiene un valor luego del punto, este es un módulo optativo para el curso en cuestión.

```
/usr/doc/.TeX/listcur
```

Contenido de los módulos

Software Libre, sus comunidades, filosofía y prácticas

Código: **SLC** Previos: **NIN**.

Computadoras: software y hardware. El modelo de Von Neumann y el concepto de programa almacenado. Procesador, memoria, conceptos de datos y direcciones. Dispositivos de entrada y salida. Controlador de interrupciones. Almacenamiento (persistencia), el archivo.

Unix, sus principios de diseño [Raymond:AUP-03]. Historia del Unix [Criado:CIG-00,Bolsky:USH-82]. Sistemas multiusuario, multitareas, seguridad.

Definición de Software Libre [Saravia:SLA-03,FSF:DSL,OSI:OSD] e historia [FSF:HPG]. Comunidades y prácticas del Software Libre, Internet. Derechos Humanos [Hipatia:SM-04], ética, libertad y sustentabilidad del conocimiento [FSF:FPG]. Software libre y copyright: la informática antes, durante y después. La libertad del conocimiento [Saravia:MH-01,Hipatia:SW,Saravia:OLC-05]. Riesgos, tres estrategias contra el movimiento: patentes de software, penalización: DMCA, y automatización del control: TCGA [Saravia:DDS-03].

Proyecto GNU [FSF:MG-85,FSF:SW]. Copyleft [FSF:QC-98]. GPL [Stallman:GPL].

El núcleo Linux [LinuxK:SW,LinuxO:SW,LinuxC:SW], sus orígenes e historia [Criado:CIG-00]. Linux y BSD. Versiones y distribuciones GNU/Linux. El impacto de GNU/Linux.

Actividad: Ver película de Extremadura relativa a Linex.

Ingreso y exploración del sistema

Código: **IES** Previos: **SLC**.

Presentación del sistema funcionando [Pentima:IUG-00]. Guía para la supervivencia [Belkin:MCN-05]. Usuarios y claves. Sesiones e ingreso al sistema, consolas virtuales, de texto y gráficas. Sesiones gráficas: KDE [KDEDocTeam:KUG-04], GNOME, otras, sesiones gráficas múltiples.

Menús e Interfaces gráficas de usuario (GUI) de texto y gráficas. Comandos y su ejecución en terminales (ls, less, mc) y entornos gráficos. Atajos (shortcuts) típicos: CTRL ALT ESC, CTRL ALT DEL, ALT + -, CTRL + -.

Conceptos elementales de archivo y de directorio. Navegadores de archivos gráficos en KDE, GNOME y otros.

La Web, idea. Concepto elemental de URL y página web. Navegar en la Web con los navegadores de archivos. Subir archivos a sitios FTP.

Ayuda y cooperación

Código: **AYC** Previos: **IES**.

Soporte, ayuda y cooperación: Man, Info, -help, whatis, usage. Libros, manuales, tutoriales, manuales de las distribuciones, howtos, FAQs, /usr/share/doc.

Web. Google. LDP. Grupos de Usuarios. Herramientas y prácticas colaborativas en Internet: Wiki, Email, Documentos Web, IRC, Mensajería/Jabber, Listas/Foros, noticias (news), BBS, Blogs. Repositorios: CVS, subversion.

Bibliografía Gral: [Bartsh:SMU]

Nota: se dicta sin requerir los conocimientos del módulo 14.3.14.

Archivos, directorios y programas

Código: **ADP** Previos: **AYC**.

Listando contenidos de directorios. ¿Qué es un directorio? Nombres de directorios y archivos. Caminos absolutos y relativos. Exploración, conceptos y árbol jerárquico estándar de directorios. Metacaracteres simples.

Directorios importantes y razones: `/usr, bin/sbin, lib/include, /home, /var /etc`, etc.; local/remoto, para superusuario/usuario común, etc. *Contenido más avanzado.*

Consola básica. Directorio de trabajo actual, cambiando directorios: `cd, pwd`. Copiando, moviendo, renombrando, creando y borrando archivos y directorios: `cp, mv, rm`. *Contenido más avanzado.*

Editores: `pico`, `nano`, `e3`, `emacs`, `vi`. Abrir, cerrar, escribir y abandonar archivos. *Contenido más avanzado.*

`cat, grep y sed`. Programación elemental con `bash, at y crontab`. *Contenido más avanzado.*

Procesadores de texto, Planillas de Cálculo, Presentaciones

Código: **PTP** Previos: **AYC**.

`OpenOffice (ooo), KOffice, Abiword, Gnumeric`. Características generales del ooo. Writer y Calc. Gráficas y sonidos Impresión. Ortografía. Auto-completar y formato automático. Plantillas. Piloto automático en calc. Impress.

Formato Oasis, zip y xml. Intercambio, importación y exportación a otros formatos: .doc, pdf.

Dibujando

Código: **DIB** Previos: **AYC**.

`Gimp, Dia, OpenOffice Draw`. *Se puede o no dar.*

Encriptación

Código: **ENC** Previos: **ADP**.

`GPG`. Concepto de encriptación. Claves públicas y privadas. Concepto de confianza. Uso en clientes de correo. *Contenido más avanzado.*

Editando y mirando páginas web

Código: **EMP** Previos: **AYC**.

`Firefox, Nvu, Quanta, OpenOffice`. Generar y mirar pdf.

Comparación entre formatos `.odt`, `.doc`, `.html` y `.pdf`.

Programas varios

Código: **PRV** Previos: **AYC**.

Música: Juk. *Se puede o no dar.*

Instalación con HGA

Código: **INH** Previos: **AYC**.

Instalación de GNU/Linux con HGA. Particionamiento. Instalación de sistemas de booteo. Copia de núcleo e Imagen. Partición raíz y árbol de directorios. El concepto de paquete. Instalación y configuración de paquetes básicos.

Instalación de software

Código: **INS** Previos: **INH**.

Instalación con sistema de paquetes: rpm, deb, tgz. Actualización diaria y mantenimiento en línea.

Administración elemental

Código: **ADE** Previos: **INS,ADP**.

Presentación del HGA para la configuración elemental.

Montando sistemas de archivos: mount, supermount, fstab. Sistema X/Xorg, configuración por HGA. Inicio del X desde consola. Usuarios, claves y permisos de archivos a nivel elemental.

Instalación desde las fuentes con configure y make: mplayer.

Periféricos

Código: **PER** Previos: **INS**.

Sonido. Escaneo, ocr, `sane`. Impresoras: detección, configuración, `cups, foomatic` (web). Teclados. Ratones. Cámaras de video y fotos. Dispositivos USB, placas PCMCIA.

Conectarse a Internet

Código: **CAI** Previos: **AYC**.

Internet, Protocolo IP, número o dirección IP. DNS. Configuración interfaz con HGA. Ruteo, y su configuración. Hora: ntpd. Redes Ethernet, teléfono, modems, ADSL, wifi. Proxy.

Usar Internet

Código: **USI** Previos: **CAI**.

Clientes web: Firefox, Konqueror, navegadores (browsers) para consola: w3m, lynx, links. Configuración del Firefox, privacidad y seguridad, plug-ins.

Fish, sftp con `Konqueror, Nautilus, mc`.

Correo, configuración del cliente de correo, `Evolution, Kmail, Thunderbird`.

Mensajería IRC e IM: `gaim, kopete, irc`,

Migración

Código: **MIG** Previos: **USI**.

Pasos para la migración individual de PCs y de cerebros. Formateando cerebros. Software libre para Windows.

Internet y consola

Código: **IYC** Previos: **USI,ADP**.

Acceso remoto, bajando, transfiriendo e intercambiando archivos: `scp, ftp, gFTP, NcFTP, lftp, wget, rsync`, p2p, BitTorrent. *Se puede optar por uno o varios de los sistemas alternativos.*

`GnuPG, mail, mutt`.

`Screen`.

Nota: se facilita el dictado si se vio previamente el módulo 14.3.7.

Redes locales y terminales remotas

Código: **RLT** Previos: **IYC**.

`SSH, telnet` y servicios ``r''. `X` en red, clientes Xorg, variable DISPLAY, ssh -X. Cliente NIS, LDAP. Montar y desmontar sistemas remotos con NFS y Samba, smbclient. Configuración básica de Samba, `/etc/exports`.

Bibliografía Gral: [Peek:UPT-93,Stonebank:UTB-01].

Incorporar la PC a la Web

Código: **IPW** Previos: **USI**.

Uso del `Apache`. Instalación y configuración elemental. Colocando páginas Web con `konqueror` y similares a sitios FTP. Compactar y comprimir directorios.

Desarrollo comunitario del Software Libre

Código: **DCS** Previos: **SLC**.

Modelos de Desarrollo del Software Libre [Ball:OMS-03]. Modelo Bazar [Raymond:CB]. Programación ágil [Beck:EPE-01].

¿Qué es un Hacker? [Raymond:CSH,Himanen:EHE-02], ¿cómo serlo?

Actividad: Ver película ``The Code".

La computadora

Código: **CMP** Previos: **ADP**.

Información analógica y digital. Los sistemas numéricos. Códigos: ASCII, Unicode.

Máquinas de Turing. El modelo de Von Neumann II, explicado en detalle. Instrucciones de máquina principales. Interrupciones de hardware II.

El concepto de archivo. Todo es un archivo.

La terminal de texto. Los comandos. Filtros. Conceptos de interfaces de usuario, de caracteres y gráficas.

Consola y Shell

Código: **CYS** Previos: **CMP,RLT**.

Introducción a los shells: `bash, sh, csh, tcsh`, etc. Bash y su uso eficiente [Criado:CIG-00,Buanzo:IC,Rivero:IPB]: introducción, globbing, metacaracteres, tilde, comillas, tilde inverso. Búsqueda y edición en la historia, history, auto-finalización, TAB, atajos.

Variables de entorno: `set, export`.

Plomería Unix

Código: **PLU** Previos: **CYS**.

Canales de comunicación estándar: entrada, salida y salida de error [Criado:CIG-00]. Flujos. Redirección. Sobreescribir o anexar. Cañerías. Objetivos útiles. `tee`. Salida de un programa como argumentos de otro: `xargs`.

`touch`.

Directorios, buscando y mostrando archivos

Código: **BMA** Previos: **PLU**.

Estadísticas en relación a los archivos [Criado:CIG-00]. Buscar en el sistema de archivos. Listar y trabajar con archivos y directorios. `ls, find`: ejemplos básicos, operadores lógicos, criterios numéricos, tiempos de acceso, ejecutar comandos.

Herramienta de búsqueda del KDE, GNOME, Mono (`Beagle`), otros entornos de desarrollo.

Espacio ocupado por archivos listados en directorios: `du`.

Procesando archivos y cadenas. Filtros. Expresiones Regulares

Código: **PAC** Previos: **BMA**.

Procesando archivos. Utilidades básicas para filtros: `cat, grep, head, tail -f, wc, sort, cut, paste`.

Versionado: `diff y patch`.

Expresiones regulares. Buscando en archivos: `less, slocate`. `sed, awk y bash`. Editando archivos con `sed y awk`.

Usuarios y grupos

Código: **USG** Previos: **IES,ADP**.

El modelo de seguridad de GNU/Linux. Usuarios, Grupos. Autenticación (login) [Criado:CIG-00].

Cambiando la clave: passwd.

Procesos y terminales

Código: **PRT** Previos: **USG**.

Procesos. Comandos: `ps -uax, top` [Criado:CIG-00].

Creación y estado de los procesos. Estadísticas de un proceso activo. Número de proceso, número de usuario del proceso. Ancestros. Señales. Matando procesos: `kill, killall`. Alterando la prioridad de los procesos. Comandos de administración de procesos interactivos. Sistema `/proc`. Demonios.

Procesos y terminales. Control de trabajos. Foreground, background: `bg, fg`. Suspendiendo procesos. Listando procesos suspendidos y en background. Volviendo al foreground. Reasumiendo procesos suspendidos. Comandos compuestos. &.

Mensajes a terminales: `write, talk, ytalk`.

Administración de memoria: `free`.

Archivos

Código: **ARC** Previos: **CMP**.

Conceptos. Inodos. directorios: `cp, mv y rm` con relación a los inodos. Links simbólicos: `ln` [Criado:CIG-00]. Links reales. Archivos, directorios, enlaces simbólicos, dispositivos de caracteres, de bloques, pipes y sockets. Sus funciones. Comandos: `mknod, mkfifo`.

Bibliografía Gral: [Tamara:AAL-03].

Dueños y permisos de archivos

Código: **DPA** Previos: **USG,BMA**.

Esquema de seguridad de archivos de GNU/Linux [Criado:CIG-00]. Permisos de archivo: dueños y tipos de permisos. Examinando e interpretando permisos: simbólico, numérico: chmod, chown, chgrp. Permisos de archivos especiales: setuid, setgid, sticky. Directorios y ejecutables. Seguridad de procesos ejecutables. Permisos por defecto, máscara.

Criterios de permisos en el `find`.

Empaquetado y compresión

Código: **EMC** Previos: **BMA**.

Empaquetando: tar, cpio, zip. Utilidades de compresión: compress, gzip, bzip2. Crear, extraer, inspeccionar: `mc`. `tar` hacia floppies no formateados. Selección de archivos, selección incremental.

Edición de archivos

Código: **EDA** Previos: **CMP**.

Uso del Emacs [Glickstein:GEE-77,Craig:CTE-99] y del vi [Criado:CIG-00]. Modos. Buffers. Pantallas. Sistemas de codificación. Búsqueda. Copiar y Pegar.

Aportar y dar soporte. LaTeX. Docbook. Versionado

Código: **ADN** Previos: **EDA,DPA,DCS**.

Escribir documentación. LaTeX[Wilkins:GSL-95,Green:HHW]. XML, SGML, docbook [Davila:TDB-02], estándar OASIS [OASIS:DXB-05] y OOO 2.0.

Control de versiones. CVS, subversión.

Participar en la comunidad redactando, brindando soporte e instrucción.

La programación

Código: **LPR** Previos: **PAC**.

Lenguajes imperativos. Tipos. Instrucciones. Control de flujo. Funciones. Stack. Variables, concepto, ubicación, nombre y tipo, cadenas, números, vectores, locales, globales, objetos. Entradas y Salidas. Archivos. Programación literata, funcional, estructurada, etc. Análisis, diseño, metodologías tipo bazar y catedral, ágiles.

El cálculo numérico y científico. Modelos. Instrumentos de control y medida.

Programación shell

Código: **PRS** Previos: **PAC**.

Filosofía Unix [Raymond:AUP-03].

El concepto de script `bash` [Criado:CIG-00]. Principios de la programación shell. Ejecución de scripts y permisos. Variables y parámetros. El entorno heredado. Trabajando en el entorno raíz. ``." Creando programas. Generando salida. Manejando la entrada, read. Estatus de salida. Estructuras de control. Ejecución condicional. Estructuras de selección: if, else, case. Tests: archivos, cadenas. Loop, for,

secuencias, while, continue y break. Parámetros posicionales. Argumentos. Manejando parámetros con espacios. Funciones. Scripts en la línea de comandos.

Introducción al Perl

Código: **INP** Previos: **PRS**.

Redactar scripts Perl simples [Wall:PP-91]. Error por falta de módulo. Módulos Perl. Instalar módulos desde CPAN, perl -MCPAN -e shell, Usar awk y sed en scripts.

Introducción a las bases de datos

Código: **IBD** Previos: **CMP**.

Bases de datos: Postgres, MySQL. Instalación, inicialización, puesta en funcionamiento. Consolas SQL.

Programas y desarrollo de sistemas

Código: **PDS** Previos: **INP,IBD**.

Introducción al C y al `gcc`, compilación [Outaline:PCP-91,Kernighan:EPU,Kernighan:LPC]. `Make` II e introducción al `autoconf`.

Tipos de datos y operadores. Sentencias y control de flujo. Declaraciones e inicialización. Funciones y estructura de programa. Operaciones de entrada y salida. Cadenas. Punteros. Asignación de memoria. Llamadas al sistema.

KDE, GNOME, Mono: como entornos de desarrollo, interfaces de caracteres. Combinar filtros con interfaces.

LAMP [LaMonica:OSL:05]: servidor `Apache`, bases de datos. Programación web. Ejemplos simples.

Desarrollo, creación, producción. Análisis, diseño, síntesis, programación.

*Actividad:*Instalación de Openwebmail.

Bibliografía Gral: [Loukides:PGS-97].

Núcleo Linux: configuración, compilación, instalación y manejo de módulos

Código: **NLC** Previos: **ADE**.

El núcleo. Módulos: `kmod, lsmod, insmod, rmmod, modprobe, modconf`. Agregar soporte a otros dispositivos. Archivos relacionados con el núcleo y los módulos: `/etc/modules, modules.conf`. Descarga, compilación, configuración básica e instalación del núcleo.

Hardware, arquitectura y dispositivos

Código: **HAD** Previos: **ARC**.

Configuración del BIOS.

Tarjetas de expansión, `ISA, PCI, pcmcia, hotplug: lspci, lsusb, pnpdump, isapnp, usbmodules, lsusb, /proc`.

Dispositivos en `/dev`.

Dispositivos en serie. Dispositivos de comunicaciones. Placas de red, NIC. Modems. Placas de sonido. `ALSA`. Impresoras. Teclados. Monitores. Detección `DDE`.

Dispositivos en bloques. IDE, disquetes, viejos CD-ROMs, otros. Geometría de los discos. Dispositivos SCSI.

Dispositivos PCMCIA. Configurar GNU/Linux para soportarlos. Configurar adaptadores Ethernet para que se autodetecten cuando se insertan. /etc/pcmcia/, *.opts, cardctl, cardmgr

Dispositivos USB

Particiones

Código: **PAR** Previos: **HAD,**.

Particionamiento [Criado:CIG-00]. Dispositivos particionables en `/dev` Definir y modificar una tabla de partición: `fdisk, ntfsresize, parted`. Diseño del particionado de discos y el layout del disco.

Formatos de particiones y sistema de archivos, `ext2`, `journaling: ext3, reiserfs, xfs`. Formatear particiones: `mkfs`. Comprobar y reparar un sistema de archivos: `fsck`. Formateando disquetes. Espacio libre: `df`.

Montando y desmontando sistemas de archivos: `mount, mountall, umount`, opciones. Punto de montaje. Sistemas de archivos montables y desmontables. Archivos `/etc/mntab y /etc/fstab`. Diversos tipos de sistemas de archivos. Dispositivos removibles: disquetes, CD-ROMs, DVDs, USB. Remount. Archivos (-o loop). Sistemas comprimidos. Sistemas encriptados. `Mtools`. `Autofs`. `Supermount`.

El proceso de arranque (boot)

Código: **EPA** Previos: **PAR**.

`Lilo` [Criado:CIG-00], `grub`. 4 etapas del encendido: arrancar (boot) con lilo-grub, cargar núcleo e imagen, descubrimiento del hardware y carga de dispositivos y módulos, inicialización con init y demonios. Instalación de lilo-grub. Adecuación del núcleo. Directorios que contienen el núcleo, la imagen inicial y los módulos. Inicialización. Llamado a init.

Inicio, niveles de ejecución y apagado

Código: **IEA** Previos: **PRS**.

Init. Niveles del funcionamiento del ``System V''. Archivo `/etc/inittab`. Pasos del proceso init. Directorios que llevan a cabo el control de los scripts utilizados para detener e iniciar procesos y servicios del sistema. Control de servicios y demonios: `/etc/init.d`. Pasos para agregar un nuevo script. Comandos para apagar el sistema: `init, shutdown, halt, reboot`.

Instalar y administrar software: binarios y fuentes

Código: **IAS** Previos: **EMC,PDS**.

Describir un paquete de software. Información contenida en un paquete de software, visualización. Administrar librerías dinámicas. Sistemas y programas de administración y mantenimiento de paquetes: tar, rpm, apt-get, dselect, alien. Consultas. Agregar un paquete de software. Remover un paquete de software.

Compilar e instalar un paquete desde la fuente: make, configure, ebuilds.

Parches y actualizaciones.

Registros (Logs)

Código: **REG** Previos: **IBD**.

Archivos y demonios de registro (logs), uso, manejo y configuración. syslogd, configuración: syslog.conf. klogd, logger

Ejecución programada de comandos

Código: **EPC** Previos: **PRS**.

Automatizar las tareas administrativas. Programar la ejecución automática de comandos, programas o scripts en determinado momento y en forma cíclica. `at y crontab`, archivos vinculados. Formato `crontab`. Crear y ejecutar un trabajo at. Pasos para crear, visualizar, editar, y para remover un archivo crontab. Usar cron. Directorios diarios, semanales y mensuales en /etc.

Administración de usuarios y grupos

Código: **AUG** Previos: **DPA**.

El superusuario root [Criado:CIG-00]. Otros usuarios del sistema. Números de identificación de usuarios y grupos. Archivos `/etc/passwd`, `/etc/shadow`, y `/etc/group`. Encriptamiento. Ocultamiento de las claves. Regular el acceso a la cuenta root, cambiando la identidad, grupo `wheel`. `su` y `sudo`.

Administrar cuentas. Crear y mantener cuentas de usuario y grupos. Comandos: useradd, userdel, groupdel.

Información sobre usuarios.

El proceso de `login` y la autenticación. `PAM`. Clientes `NIS, LDAP, Cyrus SASL`.

Envejecimiento de contraseñas.

Impresión

Código: **IMP** Previos: **PAC**.

Comandos de usuario habituales para impresión: lpr, lpq, lprm [Criado:CIG-00].Filosofía, administración, configuración y comparación de los sistemas de impresión: `lpd`, `LPRng`, `CUPS`. El

subsistema de impresión GNU/Linux. Colas. Utilidades para conversión de formatos. text, ps, pdf, etc. Añadiendo impresoras locales y remotas. Autodetección.

Respaldos (backup)

Código: **RES** Previos: **EMC**.

Estrategias y software de Respaldo. Tareas y operaciones básicas de respaldo y restauración.

X

Código: **XXX** Previos: **IYC,EDA**.

El sistema de ventanas X [Criado:CIG-00]. XFree86, X.org, XFree4, XFree3. Configuración del servidor X. Modularidad del X.

El protocolo X. Transparencia hacia las redes. X y ssh.

Establecer xdm. X remoto: XDMCP, VNC, DISPLAY, uso y configuración. Adecuación de sesiones X y aplicaciones. Terminar aplicaciones X. Seguridad en X.

Administradores de ventanas y de entornos. Conceptos del ``Display Manager'': KDE, GNOME, BlackBox, otras, instalación y configuración.

Inicio, configuración. El servidor de fonts. Instalación y mantenimiento de fonts.

Terminales remotas: ltsp, pxes, varios monitores con varias placas PCI, computadoras multipersonales con teclados y ratones USB.

Sonido y video

Código: **SYV** Previos: **HAD**.

Ondas sonoras, micrófonos y parlantes. Fuentes de audio. Digitalización: conversión A/D, muestreo. Archivos y formatos: wav, midi, ogg. Tarjetas de sonido.

Video, archivos, calidad de imagen, barrido, pixels, formas de caracterizarlo, color, ancho de banda. Video digital. Interfaces.

Percepción Visual. Luz, espectro electromagnético. Percepción visual. El sistema visual humano. Luminancia, Brillo y contraste. Modelo de visión monocromática. Representación de color. Sistemas de Coordenadas de color.

Tratamiento digital de imágenes Concepto de imagen. Elementos de un sistema de imágenes. Digitalización. Muestreo y cuantificación. Estructuras de datos. Relaciones básicas entre pixeles. Mapas de transición de luminancias. Operaciones físicas. Filtros espaciales. Filtrado de frecuencias. La convolución. Aplicaciones. El histograma.

Captura y transmisión de imágenes. Sistemas de captura de imágenes: sensores CCD. Sistema de representación de imágenes, tubos de rayos catódicos. Televisión blanco y negro, señal de video luminancia, espectro de señales de video, descripción del receptor de televisión. Televisión color, señal de crominancia, ejes de color, sistema NTC, sistema PAL, receptor, Televisión digital.

Almacenamiento de imágenes. Sistemas de almacenamiento de información, cintas magnéticas, cabezales magnéticos, sistemas VHS, sistema SVHS.

La imagen en la computadora. Sistema de toma de imágenes en computadora, conversión A/D, procesadores de imágenes.

Entorno del usuario

Código: **EUS** Previos: **PRS**.

Configurando el shell bash. Variables. Variables locales. Exportando variables. Variables comunes: PS1, TERM, PATH. Comandos de configuración. Aliases. Expansión de la línea de comando. Scripts de inicio del shell. Orden de ejecución. Shells de login. false. /etc/profile /etc/profile.d /.bash_profile y /.bashrc /.bash_logout .profile .bashrc

El entorno inicial del usuario. /etc/skel/... y directorios de usuarios /home, .

Visualizando, configurando y estableciendo cuotas: `quota` .

Fundamentos e introducción a las redes

Código: **FIR** Previos: **HAD**.

Bibliografía: [Kirch:LNG-00,Hunt:TNA-92].

Qué es una red: nodos, vínculos e interfaces. Redes locales, topología. Ethernet. Lan/Wan. Cableado. Tipos y normas.

Comunicaciones. Concepto de canal. Señales. Medida. Ancho de banda. Tipos de transmisión, modulación. Transmisión asincrónica y sincrónica. Errores. Modems.

Interfaces. Configurando interfaces de red. Módulos. Configurar dispositivos para redes locales. Configurar dispositivos para redes wan.

Redes IP. Modelo IP y modelo OSI. Redes abiertas vs. propietarias: IPX, SNA, NetBIOS, etc.

Números y direcciones: interfaz, red, broadcast y máscara, clases. Protocolo arp. Herramientas: ping, ifconfig, netstat, arp, arpwatch, directorios de configuración en /etc. Protocolos, capas, IP, TCP/UDP. Puertos.

Estableciendo clientes DNS y DHCP. Configurar y usar PPP. Diagnosticando situaciones y problemas de configuración de red.

Configuración del cliente: `/etc/hosts`, `/etc/resolv.conf`, `/etc/host.conf`, `/etc/nsswitch.conf`, `DHCP` y `DNS`. Configurar `ntp` como cliente.

Ruteo sin políticas: `route`. Comunicar entre subredes.

Configurar registros (logs) de TCP/IP.

Introducción a los servicios de red, enrutado y proxy

Código: **ISR** Previos: **FIR,PRS**.

Internet.

Configuración básica de correo electrónico, smtp, pop3, imap, seguros; sendmail/postfix/exim

Configuración básica de servidores caché DNS, DHCP, Apache y sshd.

Servicios inetd/xinetd. Inicio, habilitacion, /etc/services. Derechos. /etc/host.equiv, /.rhosts

Servidor NIS/YP y LDAP; NFS [Stern:NAN-91] y Samba [Ts:US-03,Vernooij:SDG-03].

Configurando un router básico: NAT, ip_forwarding, squid, proxy transparente.

Verificar puertos abiertos con nmap.

Seguridad

Código: **SEG** Previos: **PRS**.

Tareas del administrador de sistemas. Tareas básicas de la administración y vigilancia de seguridad. Seguridad a nivel de servidor y a nivel de usuario.

Actualizando los servidores.

Bibliografía Gral: [Bodamer:SMA].

Núcleo Linux: especializar, actualizar, adaptar

Código: **NLE** Previos: **PDS,NLC**.

Componentes del núcleo. Utilizar los necesarios para determinado hardware, drivers, recursos del sistema o requerimientos del sistema. Opciones para dispositivos UDMA66 y grabadores de CD IDE (2.4/2.6).

Instrumentar diferentes tipos de imágenes del núcleo. Identificar núcleos y parches estables y en desarrollo. zImage y bzImage.

Compilando el núcleo. Incluir o deshabilitar características específicas, configuración, opciones. Optimización. Personalización.

Realizar actualizaciones. Encontrando los cambios en un nuevo núcleo. make config, xconfig, menuconfig, oldconfig, mrproper. modules, modules_install, /usr/src/linux/, /etc/lilo.conf,

Aplicando patchs al núcleo Actualizar. Corregir errores. Añadir soporte. Remover patchs de un núcleo en producción. patch Makefile, gzip, bzip.

Adaptar un núcleo: por compilación, aplicando parches, editando archivos de configuración. Discusión.

Construir y configurar módulos del núcleo. patch, make, /usr/src/linux, /proc/sys/kernel, modprobe, /etc/conf.modules, /etc/modules.conf insmod, lsmod, kmod, kerneld.

Examinar los parámetros del núcleo en /proc.

Personalización del arranque e inicialización

Código: **PAI** Previos: **EPA,IEA**.

Personalización del inicio del sistema y de los procesos de encendido. Modificación del lilo y grub.

Configuración del ``System V" de inicio. Administrando el inicio de los demonios de systema

Adaptando el inicio. Scripts. Configurar los procesos de inicio y los niveles de ejecución estándares. Interactuar con los niveles de ejecución.

Crear imágenes personalizadas. `/etc/init.d`, `/etc/inittab`, `/etc/rc.d`, `mkinitrd`.

Sistema de archivos, RAID, LVM

Código: **SAR** Previos: **PAR**.

Funcionamiento y mantenimiento del sistema de archivos GNU/Linux. Crear y configurar las opciones del sistema de ficheros.

Archivo `/etc/fstab`, cómo reparar el archivo cuando el sistema falla en el encendido.

Dispositivos de bloques y otros dispositivos. Direccionamiento de discos y particionado.

Configuración, administración y uso de RAID y LVM. Usar ACLs.

Conceptos de Raid. Raid por software vs. por hardware. Configurar e instrumentar RAID por software. Raid: 0, 1, 5. mkraid, /etc/raidtab

Uso de LVM (Logical Volume Manager) para administrar discos y particiones.

Herramientas para interactuar con parámetros de los discos duros: hdparm, tune2fs, /proc/interrupts, sysctl, fsck , badblocks, Mke2fs, Dumpe2fs, Debuge2fs, Tune2fs.

Particionado de servidores, diseño.

Configurar y montar varios tipos de sistemas de archivos. Manipular sistemas de archivos para ajustar el requerimiento de espacio libre y para adicionar dispositivos. /etc/mtab, sync, swapon y swapoff, /proc/mounts

Opciones de los sistemas de archivos Automount, automont para redes, para dispositivos. IVMAN. Sistemas no ext2. Sistemas para cdroms. /etc/auto.master, /etc/auto, mkisofs, dd, mke2fs.

Hardware

Código: **HAR** Previos: **HAD,NLE**.

Conocimiento básico del hardware. Arquitecturas Intel y clones. Ajustando IRQs para los puertos y slots estándares.

Configurar subsistemas de discos: IDE, EIDE, SCSI.

Instalación de hardware. Compatibilidad. Dispositivos internos, externos, nuevos discos internos: scsi, terminales tontas, dispositivos seriales de UPS, tarjetas seriales multipuerto, paneles LCD, setserial.

Configuración de software y del kernel. Configuración de dispositivos PCMCIA y USB, hotplug.

Soporte de CPUs y SMP. Detección de nuevo hardware. Soporte Plug and Play.

Dispositivos periféricos y su configuración. lsdev, lspci, usbview, /proc/bus/usb

Archivos compartidos, red local: DHCP, NIS/LDAP/Kerberos/ Cyrus, NFS, Samba

Código: **ACR** Previos: **FIR,PAR,ARC**.

DHCP

Fundamentos. Configuración y monitoreo del servidor. Hosts estáticos y dinámicos. `dhcpd.conf`, `dhcpd.leases`. Esclavos. Distintas alternativas de configuración. Control por dirección de hardware de las placas de red.

NIS, LDAP, Kerberos, cyrus sasl

Revisión de la configuración de un cliente.

Fundamentos de NIS. Cliente. Servidor. nsswitch.conf. Utilidades YP. /etc/nis.

Servicios LDAP. Esquemas. Referenciando las entradas LDAP. Seguridad LDAP.

Implementar y configurar el servidor OpenLDAP. Archivos LDIF. Configuración del usuario. slapd y slapd.conf

Parámetros globales. Restringiendo el acceso. Configuración e indexado de las bases de datos. Consultando las bases LDAP.

Cyrus SASL [CSSA].

Instalar y configurar kerberos.

NFS

Configuración de un servidor NFS. Crear archivo /etc/export y especificar directorios a exportar, exportfs. Restringir hosts en cada entrada. Especificar opciones. showmount. Configurar el mapeo de ID de usuarios.

Configuración de los clientes NFS. Montar un recurso NFS en un cliente. Subredes o grupos de redes. Especificar opciones del mount para reintentos soft o hard. Manejo de señales. Tamaño de bloque. Configurar wrappers para asegurar el NFS. nfsstat.

Samba

Samba 3. Teoría del protocolo SMB. NetBIOS y NetBEUI. Nombres NetBIOS. Establecer un servidor para varios clientes, smbd. Servidor nmbd. Wins. Configuración de un servidor. Swat. HGA. Cambiar el grupo de trabajo del servidor. Compartiendo archivos, directorios e impresoras con Samba. Definir un recurso compartido en smb.conf: directorio, impresora.

Herramientas para los clientes. Establecer un script para ingreso de clientes. Montando recursos compartidos SMB. Usar nmblookup para testear la funcionalidad WINS. Usar smbmount para montar un recurso en un cliente. smbstatus, smbtestparm, smbpasswd, lmhosts.

Cuestiones relativas a las contraseñas encriptadas de Windows y Samba. Registry de Windows para contraseñas sin encriptar.

OpenAFS Conceptos. Instalación y configuración de kerberos y OpenAFS.

Construcción de paquetes, distribución, repositorios

Construir paquetes de software. Estrategias de división de paquetes. Distribución de paquetes. Paquetes Debian. /debian/rules. Reconstruyendo un .deb. Reconstruyendo paquetes .rpm. Reconstruyendo RPMs, rpm, formato de archivo SPEC.

Configuración post-instalación.

Instalación automatizada desde repositorios

Registros (logs), contabilidad, performance

Código: **RCP** Previos: **REG,PRS**.

Configuración y análisis de registros (logs) del sistema. Configurar syslogd como servidor central de registro (logs). Configurar syslogd para enviar a un registro (log) central. Conexiones de registros (logs) remotos. Uso de grep y otras utilidades de texto para automatizar el análisis de registros (logs). `syslog.conf`, `sysklogd`, `/etc/hosts`.

Contabilidad de procesos. Límites a los procesos. Análisis y ajuste de performance.

Respaldo (backup) fuera del sitio

Código: **RFS** Previos: **RES,PRS**.

Planes de respaldo y recuperación fuera del sitio.

Automatización de tareas

Código: **AUT** Previos: **EPC,RCP**.

Planificación de tareas administrativas. (uso de cron, anacron y at). Manejo de procesos que se ejecutan en el sistema.

Scripts para probar la ejecución de procesos. Generación de correos con alertas. Generar alertas SMS ante la muerte de procesos. Programar scripts para analizar y ``parsear'' registros (logs) para generar alertas y correos a los administradores. crontab.

Sincronizar archivos entre máquinas con rsync.

Monitorear cambios de archivos e ingreso, salida de usuarios y generar alertas.

Resolución de problemas

Código: **REP** Previos: **EPC,AUT**.

Conceptos para solucionar problemas en GNU/Linux. Análisis de registros (logs) para identificar problemas. Uso de herramientas de ayuda a nivel de sistema para solucionar problemas.

Niveles de ejecución de rescate.

Uso de entornos de rescate de GNU/Linux. Creación de discos de recuperación. Recuperación del sistema. Disco o CD-ROM estándar para reparación y recuperación. /usr/sbin/rdev, /sbin/lilo, /bin/dd, /sbin/mke2fs, /etc/fstab, /etc/inittab, /usr/bin/chroot. LPD bootdisk Howto.

Recuperación. Manipular un GNU/Linux durante el proceso de encendido y en el modo de recuperación. La utilidad init. La opción init=kernel. LILO, init, inittab, mount, fsck.

Solución de problemas del lilo y grub. L, LI, LIL, LILO. /boot/boot.b /boot/boot Identificación de las 4 fases del encendido.

Solución de problemas generales. Archivos con los mensajes durante arranque. Utilizar mensajes para diagnosticar errores. Identificar y corregir cuestiones comunes de hardware. Determinar si el problema es de hard o soft.

dmesg, syslogs del núcleo en los registros (logs), registros (logs) del núcleo y demonios en /var/log, /sbin/lspci, /usr/bin/lsdev, /sbin/lsmod, /sbin/modprobe, /sbin/insmod, /bin/uname.

Ubicación de los archivos del núcleo y módulos, /boot, /lib/modules.

El sistema /proc, strace, strings, ltrace, lsof.

Solución de problemas con los recursos del sistema. Identificar, diagnosticar y reparar el entorno local. /etc/profile y /etc/profile.d/, /etc/bashrc y otros /etc/init.d/. /etc/rc.*. /bin/ln. /bin/rm. /etc/ld.so.conf. /sbin/ldconfig. /sbin/sysctl, /etc/sysctl.conf.

Solución de problemas de configuración del entorno. Técnicas comunes de reparación. /etc/inittab, /sbin/init, /etc/passwd, /etc/shadow, /etc/group, /etc/profile, /etc/rc.local o /etc/rc.boot, /usr/sbin/cron, /usr/bin/crontab, /var/spool/cron/crontabs/, /etc/SHELL_NAME, /etc/login.defs, /etc/syslog.conf

Recuperación de sistemas de archivos corruptos.

Puntos de revisión en X, servicios, redes, encendido.

Bibliografía Gral: [Liu:MII-94].

DNS

Código: **DNS** Previos: **ISR**.

DNS historia y teoría [Albitz:DAB-92]. El espacio de nombres. Delegación de zonas. Resolviendo nombres y búsquedas inversas.

Bind sólo caché. Configurando bind: /etc/named.conf. Configurando zonas directas e inversas. Zonas especiales. Archivos de Zonas. Zonas primarias y secundarias. Creación y mantenimiento de zonas DNS. Registros SOA, NS, MX, A, CNAME, otros.

Recargar bind con kill o ndc.

Jerarquías de DNS, delegación de subdominios.

Seguridad en un servidor DNS. chroot jail, non-root. DNSSEC Registros (logs) del DNS.

Vistas. Restringiendo las consultas y las transferencias de zonas. DDNS y nsupdate, dnskeygen

Clientes: dig, nslookup, host.

Archivos de inicio del System V.

Superdemonio inetd

Código: **INE** Previos: **ISR**.

Wrappers de TCP, configuración, tcpd. Seguridad. Servicios inetd/xinetd. Inicio, habilitación, /etc/services. Derechos. /etc/host.equiv, $home/.rhosts /etc/inetd.conf, /etc/hosts.allow, /etc/hosts.deny

---

FTP (Protocolo de transferencia de archivos)

Código: **FTP** Previos: **ISR**.

Servidores ftp y ftp seguros

SSH

Código: **SSH** Previos: **ISR**.

Fundamentos. OpenSSH. Utilidades del cliente.

Configuración del servidor. Métodos de Autenticación. sshd. Generación de llaves. Redireccionamiento de puertos.

Redireccionamiento de X.

Servidor Web (apache)

Código: **SEW** Previos: **ISR**.

Teoría del protocolo HTTP. Historia y estatus. Arquitectura.

Configuración e Instalación del Apache. Archivos de configuración. Usando módulos: proxy. Registros (logs) y análisis. Monitorear carga y performance.

Contenido dinámico. CGI. Perl. PHP. Tomcat. Seguridad.

Servicios FTP.

Autenticación. Restringiendo usuarios. Configurar mod_perl y mod_php.

Pedidos máximos. Mínimos y máximos de clientes y servidores.

Registro (log) de inicio: rcapache.out, access.log, error.log, .htaccess, mod_auth, htpasswd, htgroup, httpd.conf.

Hosting virtual.

Cuestiones de seguridad. Criptografía, uso de HTTPS. Certificados SSL. Definiciones y configuración. OpenSSL. Instrucciones de redireccionado en httpd.conf.

Caché y proxy web (Squid)

Código: **PRW** Previos: **ISR,SEW**.

Introducción.

Instalar y Configurar un proxy squid.

Políticas de acceso. Autenticación. squid.conf. ACLs y jerarquías. http_access.

Medición de ancho de banda y monitoreo.

Correo electrónico y noticias

Código: **EMA** Previos: **ISR,SEW**.

Teoría del SMTP. SMTP con sendmail. Configuración. Lenguaje de macros m4. Archivo senamil.mc. Otros archivos de configuración. ESMTP AUTH y encriptación.

SMTP con Postfix. Configuración. ESMTP AUTH y encriptación.

Exim: exim.conf.

Configuración de las listas de correo, aliases, /etc/aliases. Quotas Uso de Sendmail. Controlando el tráfico. Monitoreando servidores SMTP.

Dominios virtuales. Mails relays.

Servicios: POP3 y IMAP4 Encriptando el acceso de clientes.

Filtro de correos: virus y Spam, ordenamiento. Monitoreo de correo ingresante.

procmail .procmailrc Procmail server-side.

Clientes de mail vía web: openwebmail, squirrel.

Servicio de noticias.

Mailman.

Políticas en el Ruteo

Código: **POR** Previos: **ISR**.

Ruteo por políticas, route, programa ip.

Ruteo Dinámico

Código: **RUD** Previos: **ISR,POR**.

routed, zebra.

Netfilter

Código: **NEF** Previos: **RUD**.

Puertos. iptables. Configuración de netfilter/iptables. Reglas de filtrado.

Manteniendo las reglas. Ejemplos.

NAT con netfilter. Tracking de conexiones.

Configuraciones comunes en /proc/sys/net/ipv4.

Configurar un router básico: NAT, ip_forwarding, squid, proxy transparente.

Resolución de problemas en redes

Código: **RPR** Previos: **NEF**.

Herramientas de diagnóstico e información: tcpdump, nmap, ntpd, finger, snmp, netstat, registros (logs) del sistema, ifconfig, /etc/network, /etc/sysconfig/network-scripts/, ping, /etc/resolv.conf, /etc/hosts, /etc/hosts.allow, /etc/hosts.deny, /etc/hostname, /etc/HOSTNAME, traceroute, nslookup, dig, host, dmesg.

Seguridad en Redes

Código: **SER** Previos: **RPR**.

Tareas de seguridad, asegurando servicios. Configuración PAM. Firewalls.

Sistemas de detección de intrusos.

Nessus.

Establecer alertas/alarmas de seguridad.

Cerrar relays abiertos de email.

Escaneo de puertos con nmap.

Configurar dispositivos de red para autenticación. VPN.

Presentación e instalación de sistemas LAMP

Código: **PIL** Previos: **SEW,PDS**.

Instalación de Apache, Postgres, MySQL, Perl, PHP, Python, módulos y otros necesarios. Descarga desde Internet. Compilación y configuración.

Trabajo en equipo

Código: **TEQ** Previos: **SEW**.

Subversion. SVK. CVS. GIT.

Bases de datos

Código: **BDD** Previos: **IBD**.

Introducción al SQL. SQL: Elementos Básicos del lenguaje. Objetos, operadores y expresiones. Definiciones de datos: Tablas base. Manipulación de datos: Operaciones: Select, insert, update, delete. Expresiones de tablas, condicionales. Vistas. Integridad. Nulos. Transacciones. Agregados. Joins. OID. Secuencias. Tipos de datos. Manejo de tablas. Restricciones. Funciones y disparadores. Transacciones.

Modelos entidad relación. Normalización. Nulos. Algebra relacional. Tuplas.

Postgres y MySQL. Terminal SQL. Instalación. Administración. Sintonización. Respaldo. Replicado. Fortalezas y debilidades en relación a otros motores. Sesiones. Comandos.

Perl, Python y PHP

Código: **PPP** Previos: **PDS,BDD**.

Programación en lenguajes dinámicos: elementos básicos en los tres lenguajes. Variables y constantes. Ámbito. Globales, locales, léxicas. Cadenas, números, listas, hashes. Expresiones Operadores Estructuras de Control. Formas de Ejecución. Subrutinas. Expresiones Regulares. Archivos.

Perl. Presentación. Variables. Particularidades. Módulos y Objetos en Perl.

Programación para redes

Código: **PPI** Previos: **PPP,SEW**.

Uso de sockets. Acceso a servidores. Envío de Email. Acceso a HTTP, FTP, SMTP

Programación de Bases de Datos

Código: **PBD** Previos: **PPP,BDD**.

Perl DBI. Uso de Módulos. Programación. Interacción con la Base de Datos. Comprobación de errores. Consultas iterativas. Subrutinas básicas.

HTML, CGI, JavaScript

Código: **HCJ** Previos: **SEW,PPP**.

HTML. Protocolo HTTP. URLs. Introducción al HTML. Estructura. Texto. Imágenes. Enlaces. Listas. Tablas. Formas. Marcos.

Programación CGI. Página dinámica. Idea. Entorno. Parámetros. Get y Post. Procesando y generando con el mismo script. Sesiones y Cookies.

CGI.pm: Introducción. Funciones.

JavaScript: introducción, estructura léxica. Tipos de datos, valores y variables. Expresiones y operadores. Sentencias. Funciones. Objetos. Arrays. JavaScript en Navegadores. Ventanas y Marcos. DOM. Eventos. Formas y elementos. Cookies. Ajax.

Estructura y división de la currícula en cursos. Otras currículas y certificaciones.

En principio cada curso está pensado para durar 5 días, cada uno de ellos con 4 horas de clase dirigida, dos horas de práctica individual tutelada y dos horas de actividad individual.

El currículo está pensado para facilitar su reducción o alargue, eliminando algunas partes o profundizando en otras, alterando el orden, la organización, la duración y la puesta en práctica concreta. Está pensado también como una guía de auto-aprendizaje y un compendio de la bibliografía relevante.

Es recomendable no ver todas las alternativas de cada aplicación por cada función. En cada instancia de su uso es razonable optar por alguna de ellas según las necesidades de los organizadores de la misma, o preferencia de los estudiantes.

Se propone un plan de estudios constituido con un conjunto de cursos. Cada curso enfoca un perfil especial y se va construyendo sobre el conocimiento contenido en el anterior. Al terminar cada curso el estudiante completa el perfil de ese curso.

- El **curso 1** está destinado a todas las personas y sirve para completar su formación cultural, presentándoles las herramientas TIC, y dándoles una base para reproducir en otras personas estos conocimientos, ayudando a romper el fenómeno conocido como brecha y el analfabetismo digital [Abismo:SW]. Incluye también los conceptos básicos de la programación de ordenadores que deben empezar a formar parte de la cultura general de una humanidad [Saravia:PHC-05] cada vez más compañera de ruta de ordenadores y automatismos.

- El **curso 2** está destinado a quienes desean comenzar el camino hacker en informática [Raymond:CSH,Himanen:EHE-02]. Incorpora el conocimiento básico para producir y vivir en una computadora moderna y seguir formándose por su cuenta. Supone conocido el contenido del curso 1. Es similar al curso de Red Hat: RHA 030 [RedHat:RAC].

- El **curso 3** agrega los conocimientos necesarios para administrar una estación de trabajo. Supone conocido el contenido del curso 2.

- Los **cursos 2 y 3** suman el conocimiento básico necesario para la certificación LPIC 1 (Junior Level Administrator) que se obtiene rindiendo dos exámenes LPI 101 y 102 [Dan:LCN,Hunt:LPI,Dulaney:ENL-02]. Los contenidos de cada examen están distribuidos en los dos cursos. Estos dos cursos suelen ser similares en contenido y nivel a los cursos: Sair 1 [Sair:SLC], Novell Suse 103 Novell 3036 SuSE Linux Fundamentals, Novell 3037 SuSE Linux Administration - 3036 [Novell:SLT], GBDirect: Linux and Unix Fundamentals [GBDirect:LUF]. Varios: [Incasol:SW,UOC:ML,IBM:CDL,IBM:TPL,GuruLabs:LSAC].

- El **curso 4** aporta los conocimientos necesarios para administrar un servidor grupal. Supone conocido el contenido del curso 3. Incluye los contenidos de la certificación LPI 201 [Dan:LCN,Hunt:LPI,Dulaney:ENL-02]. Y de los siguientes cursos: 3038 Advanced SuSE Linux Administration [Novell:SLT], GBDirect: Advanced System Administration [GBDirect:ASA]. El curso 130 de Red Hat cubriría el contenido de este curso, y del 3 [RedHat:RAC],

- El **curso 5** aporta los conocimientos necesarios para administrar servicios de Internet. Supone conocido el contenido del curso 4. Este curso da los contenidos de la certificación LPI 202 [Dan:LCN,Hunt:LPI,Dulaney:ENL-02]. GBDirect: Running Linux in the Enterprise: Network Administration [GBDirect:RLE] RHA 230 Red Hat Linux Network Applications [RedHat:RAC]

- El **curso 6** aporta los conocimientos necesarios para realizar, instalar y configurar sistemas LAMP.

Las equivalencias no son exactas, ya que no todas las organizaciones distribuyen el contenido en la misma secuencia. Terminar un curso de este currículo, en general significa que se cubren los cursos o exámenes que se reputan como similares, quizás en exceso.

Si bien se ha intentado no repetir los contenidos entre los cursos, muchas veces es conveniente proceder como en una espiral que a la vez que crece vuelve a recorrer algunos ``ángulos" del círculo a mayor nivel. En particular esto sucede con algunos protocolos de red que se ven primero como clientes y luego se profundiza en sus fundamentos y en sus servidores.

Cursos

Curso 1: Usuario TIC

TIC: Tecnologías de la Información y Comunicaciones.

Descripción

Curso que pretende formar una persona partícipe de las TIC con autosuficiencia y capacidad de reproducción de sus conocimientos en su comunidad.

Contiene lo necesario para instalar y operar un escritorio GNU/Linux en entornos hogareños, de pequeñas organizaciones y en conexión con las comunidades internacionales.

Se hace hincapié en la capacidad de comprender el significado de la programación de ordenadores y las técnicas colaborativas y de construcción grupales, especialmente aquellas que aseguren anonimato y seguridad en la red como encriptación.

Se aspira a contribuir a formar personas tecnológicamente aptas para convivir, participar, militar y ganarse la vida en un mundo cada vez más conectado y mediatizado por computadoras.

Requisito de conocimientos previos

Ninguno

Objetivos y contenidos mínimos

Uso autónomo de las TIC, participación en comunidades y capacidad de sensibilizar y ayudar a los demás.

Trabajar efectivamente con:

- Las comunidades del movimiento del Software Libre. Historia. Fundamentos políticos, filosóficos, éticos y prácticos de la libertad del conocimiento. Cómo participar.
- Escritorio gráfico del sistema GNU/Linux.
- Aplicaciones de oficina, grupales, y de Internet. Operar los aplicativos tradicionales tanto en entorno gráfico como en consola.
- Habilidad para editar archivos en consola. Abrir, editar y salvar archivos de texto con nano/pico, vi, e3 y emacs.
- Instalación del sistema desde CDROMs o Internet.
- Hardware. Impresoras. Equipos multimedia. Sonido. Cámaras. Dispositivos USB, placas pcmcia.
- Conexión a red (IP): placas Ethernet, modems, placas Wifi y otros dispositivos. Acceder como cliente a una red. Internet y sus servicios.
- Administración básica con una herramienta gráfica avanzada (HGA).
- Particiones, sistemas de archivo, directorios, archivos. Árbol jerárquico de directorios estándar del sistema GNU/Linux.
- Conceptos de programación.
- Encriptación de mensajes.

HGA: Webmin, Yast, otras.

Contenido

Módulos: 14.3.1, 14.3.2, 14.3.3, 14.3.4, 14.3.5, 14.3.6, 14.3.7, 14.3.8, 14.3.9, 14.3.10, 14.3.11, 14.3.12, 14.3.13, 14.3.14, 14.3.15, 14.3.19, 14.3.18, 14.3.16.

Curso 2: Usuario técnico avanzado

Descripción

Curso de formación tendiente a que los estudiantes dominen, puedan usar a profundidad su sistema GNU/Linux y puedan seguir adquiriendo la variedad de conocimientos necesarios para progresar solos.

Requisito de conocimientos previos

Curso 1.

Objetivos y contenido sintético

Obtener un sólido fundamento en conceptos técnicos de GNU/Linux y el Software Libre junto a sus fortalezas y debilidades.

- Modelos de desarrollo basados o usados en el software libre: bazar, métodos ágiles, etc.
- Expresiones regulares: grep y programación con la filosofía unix: sed, awk y bash.
- Redirección de I/O, pipes.
- Manipular archivos y directorios.
- Administrar y mantener los siete tipos fundamentales de entradas, permisos/derechos y propietarios.
- Administrar la memoria y los procesos.
- Uso eficiente del emacs.
- Preparar documentación: LaTeX, html, pdf, docbook, oasis.
- Desarrollo a nivel elemental: bash, perl y c.
- Repositorios: CVS, subversion, git, gnuarch.

Contenido

14.3.20 14.3.21 14.3.22 14.3.23 14.3.24 14.3.25 14.3.26 14.3.27 14.3.28 14.3.29 14.3.30 14.3.31 14.3.32 14.3.33 14.3.34 14.3.35 14.3.36 14.3.37

Curso 3: Instalación y mantenimiento de estaciones de trabajo

Descripción

Curso de formación básica tendiente a que los estudiantes puedan configurar y mantener su estación de trabajo.

Requisito de conocimientos previos

Curso 2.

Objetivos y contenido sintético

Obtener un sólido fundamento en el funcionamiento del sistema operativo GNU/Linux.

- Conocer y saber utilizar los comandos principales esenciales. Proficiencia en la línea de comandos. Cada tarea, cuando se puede, se analiza a 3 diferentes niveles de abstracción: archivos/memoria, scripts de línea de comando y HGA. Eventualmente a nivel de librerías de c.
- Administrar autenticación, usuarios y grupos.

- Particionamiento. Entender y manipular los sistemas de archivos de GNU/Linux: estructura y funcionamiento, crearlo y mantenerlo.
- Instalar, actualizar y sacar software organizado en paquetes. Consultas a las bases de datos de paquetes. Dependencias. Usar make y configure para instalar paquetes de fuentes.
- Entender, instalar, configurar, prender y apagar servicios.
- Administrar el subsistema de impresión.
- Automatizar tareas con cron y at.
- Utilizar y configurar el hardware habitual en las computadoras personales.
- Mantener e interpretar los registros (logs) del sistema.
- Realizar y mantener respaldos (backups).
- Compilar el núcleo, incorporar módulos, interactuar con lilo y grub.
- Configurar el sistema X. KDE, GNOME, otros escritorios.
- Conceptos básicos sobre redes. Configurar e integrar una máquina en su red ejecutando los servicios habituales y activando los clientes NIS, DHCP y DNS.
- Configurar servidores a nivel básico: NFS, Samba, etc.
- Configurar seguridad esencial a nivel de host y de red.
- Solucionar problemas básicos típicos.

Contenido

14.3.38 14.3.39 14.3.40 14.3.41 14.3.42 14.3.43 14.3.44 14.3.45 14.3.46 14.3.47 14.3.48 14.3.49 14.3.50 14.3.51 14.3.52 14.3.53 14.3.54

Curso 4: Administración de servidores

Descripción

Curso orientado a la administración de servidores grupales para múltiples usuarios y estaciones de trabajo en una red local.

Requisito de conocimientos previos

Contenidos del Curso 3.

Objetivos y contenidos mínimos

El objetivo de este curso es mostrar la configuración de un servidor para múltiples usuarios, en especial sirviendo a varias estaciones de trabajo en una red local.

Es importante la capacidad de planear, instrumentar, mantener, corregir y solucionar errores. Diseñado para dar experiencia práctica en estas cuestiones junto a los fundamentos de cada tema.

- Adaptar el núcleo Linux a las necesidades de cada configuración de servidor.
- Inicialización de servicios
- Administración y mantenimiento de particiones y sistemas de archivos: RAID, LVM.
- Servicios para una red local: DHCP, NIS/LDAP/Kerberos/Cyrus, archivos compartidos: NFS, Samba, OpenAFS
- Automatización de tareas
- Identificación y solución de problemas.

Contenido

14.3.55 14.3.56 14.3.57 14.3.58 14.3.59 14.3.61 14.3.62 14.3.63 14.3.64

Curso 5: Servicios Internet, enrutado y netfilter

Descripción

Curso para enseñar a establecer servidores y routers, tanto a nivel de redes locales como de Internet. Se da especial importancia a los conceptos por sobre las recetas y se hace énfasis en la seguridad y en la capacidad de solucionar errores y problemas, encontrando sus causas y detectándolos en forma temprana.

Requisito de conocimientos previos

Curso 4.

Objetivos y contenidos mínimos

- Conocimientos profundos sobre los protocolos de interconexión de redes IP y el funcionamiento de Internet y los vinculados al mismo.
- Redes locales y sus servicios típicos de autenticación y archivos.
- Servicios de Internet
- Ruteo, políticas de ruteo, ruteo dinámico.
- Filtrado y acciones a nivel de puerto

Contenido

14.3.65 14.3.66 14.3.67 14.3.68 14.3.69 14.3.70 14.3.71 14.3.72 14.3.73 14.3.74 14.3.75 14.3.76

Curso 6: Desarrollador LAMP

Descripción

Sitios web dinámicos provistos por bases de datos.

Curso para enseñar a instalar, programar, especificar, diseñar, analizar y configurar sistemas LAMP.

Requisito de conocimientos previos

Curso 5.

Objetivos y contenidos mínimos

- Aprender a programar.

Contenido

14.3.77 14.3.78 14.3.79 14.3.80 14.3.83

Índice de Materias

Fundamentos e introducción a las redes

/etc/shadow

Administración de usuarios y grupos

/proc

Procesos y terminales

/usr, bin/sbin, lib/include, /home, /var /etc

Archivos, directorios y programas

ALSA

Hardware, arquitectura y dispositivos

Apache

Incorporar la PC a la Web | Programas y desarrollo de sistemas

at y crontab

Ejecución programada de comandos

autoconf

Programas y desarrollo de sistemas

Autofs

Particiones

bash

Programación shell

bash, at y crontab

Archivos, directorios y programas

bash, sh, csh, tcsh

Consola y Shell

Beagle

Directorios, buscando y mostrando archivos

bg, fg

Procesos y terminales

bienes de consumo

Conceptos primitivos. Modelo económico básico

Bienes económicos

Conceptos primitivos. Modelo económico básico

Bienes intermedios

Conceptos primitivos. Modelo económico básico

Capital

Definición de capital

cat, grep, head, tail -f, wc, sort, cut, paste

Procesando archivos y cadenas. Filtros. Expresiones Regulares

cat, grep y sed

Archivos, directorios y programas

cd, pwd

Particiones

kill, killall

Procesos y terminales

kmod, lsmod, insmod, rmmod, modprobe, modconf

Núcleo Linux: configuración, compilación, instalación y manejo de módulos

konqueror

Incorporar la PC a la Web

Konqueror, Nautilus, mc

Usar Internet

less, slocate

Procesando archivos y cadenas. Filtros. Expresiones Regulares

Lilo

El proceso de arranque (boot)

ln

Archivos

login

Administración de usuarios y grupos

lpd

Impresión

LPRng

Impresión

ls, find

Directorios, buscando y mostrando archivos

Make

Programas y desarrollo de sistemas

mc

Empaquetado y compresión

mkfs

Particiones

mkinitrd

Personalización del arranque e inicialización

mknod, mkfifo

Archivos

mount, mountall, umount

Particiones

Mtools

Particiones

nano

Archivos, directorios y programas

NIS, LDAP, Cyrus SASL

Administración de usuarios y grupos

ntp

Fundamentos e introducción a las redes

OpenOffice (ooo), KOffice, Abiword, Gnumeric

Procesadores de texto, Planillas de Cálculo, Presentaciones

PAM

Administración de usuarios y grupos

pico

Archivos, directorios y programas

Precio

Precio o renta de los factores

Preferencia, función

Decisiones y la preferencia

ps -uax, top

Procesos y terminales

quota

Entorno del usuario

Relaciones de producción

Relaciones de producción

Riqueza, función

La visión ultra-simple y la definición de riqueza

route

Fundamentos e introducción a las redes

sane

Periféricos

scp, ftp, gFTP, NcFTP, lftp, wget, rsync

Internet y consola

Screen

Internet y consola

sed y awk

Procesando archivos y cadenas. Filtros. Expresiones Regulares

sed, awk y bash

Procesando archivos y cadenas. Filtros. Expresiones Regulares

set, export

Consola y Shell

SSH, telnet

Redes locales y terminales remotas

su

Administración de usuarios y grupos

sudo

Administración de usuarios y grupos

Supermount

Particiones

sysklogd

Registros (logs), contabilidad, performance

syslog.conf

Registros (logs), contabilidad, performance

tar

Empaquetado y compresión

tee

Plomería Unix

touch

Plomería Unix

valor

Precio o renta de los factores

vi

Archivos, directorios y programas

wheel

Administración de usuarios y grupos

write, talk, ytalk

Procesos y terminales

X

Redes locales y terminales remotas

xargs

Plomería Unix

Anexos: Derechos y estándares

HTML5 cumple con el formato HTML declarado por `<!doctype html>`.

Bibliografía

MySQL:SW
: MySQL AB.
Sitio web.

`http://www.mysql.com/company`.

Abismo:SW
: El Abismo.
Sitio web. un observatorio de la brecha digital.

`http://www.el-planeta.com/abismo`.

Amadeu:SLI-03
: S. Amadeu, Cassino, Lima, et al.
Software Livre e inclusao digital.
Conrad, 2003.
ISBN 8587193961.

Adelstein:ULO-05
: Tom Adelstein.
Understanding the linux and open software development model, 2005.

`http://lxer.com/module/newswire/view/47182/`.

Adobe:TWP
: Adobe.
Try a windows ppd file provided.

`http://www.adobe.com/products/printerdrivers/winppd.html`.

EGA:OPF-05
: Economist Global Agenda.
Online pirates forced to walk the plank, Junio 2005.

`http://www.economist.com/agenda/displaystory.cfm?story_id=4124724&fsrc=nwl`.

Alexander:ULP-77
: Christopher Alexander, Sara Ishikawa, y Murray Silverstein.
Un lenguaje de patrones.

Gustavo Gill, 1977.
ISBN 84-252-0985-4.

Albitz:DAB-92
Paul Albitz y Cricket Liu.
Dns and bind, 1992.
ISBN 1-56592-236-0.

Alemania:MM
Alemania.
Manual de migración.
`http://www.olea.org/cursos/realizados/2004-06-UJI_CENT-curso-avanzado-publicacion-electronica/practica/libro/index.html`.

Anderson:ESR
Ross Anderson.
Economics and security resource page.
`http://www.cl.cam.ac.uk/users/rja14/econsec.html`.

Anderson:PFI-03
Ross Anderson.
Preguntas frecuentes sobre informática fiable, 2003.
`http://linuca.org/body.phtml?nIdNoticia=207`
`http://www.cl.cam.ac.uk/users/rja14/tcpa-faq.html`.

Andriole:GGT-03
Steve Andriole.
Getting a grip on TCO and ROI, 9 2003.
`http://itmanagement.earthweb.com/columns/bizalign/article.php/3076031`.

Annan:QG
Kofi Annan.
¿Qué es la globalización?, 2001.
`http://www.analitica.com/va/internacionales/fuentes/4782010.asp`.

Apache:SW
Apache.
Sitio web.

`http://www.apache.org`.

APC:PI
: APC.
Posición institucional.

`http://www.apc.org/english/rights/governance/`.

Appleyard:CH-2003
: Bryan Appleyard.
Ciencia vs Humanismo.
El Ateneo, 2003.
ISBN 950-02-5890-0.

apt4rmp:FAQ
: apt4rpm.
Apt-rpm questions and answers.

`http://apt4rpm.sourceforge.net/faq.html`.

OCA:SVG
: Open Clip Art.
SVG clip art library.

`http://www.openclipart.org`.

Autoconf:AMA
: Autoconf.
The autoconf macro archive.

`http://ac-archive.sourceforge.net/`.

Bacon:TZO-05
: Jono Bacon.
Trust and zeal in open source advocacy, 2005.

`http://www.linuxdevcenter.com/pub/a/linux/2005/04/21/advocacy.html`.

Bailey:MR-97
: Edward C. Bailey.
Maximum rpm, taking the red hat package manager to the limit, 1997.

`http://www.openpkg.org/doc/book/maximum-rpm.html/`.

Ball:OMS-03

Philip Ball.
Openness makes software better sooner.
Nature, 2003.

`http://www.sci-tech-today.com/perl/story/22862.html`

`http://www.nature.com/news/2003/030616/pf/030623-6_pf.html`

`http://www.nature.com/nsu/030623/030623-6.html`

`http://noti.hipatia.info/modules.php?op=modload&name=News&file=article&sid=1403`

`http://bioinformatics.org/forums/forum.php?forum_id=1954`

`http://opensource.mit.edu/pipermail/discuss/2003-July/000275.html`.

Banks:M
Greg Banks.
Maketool.

`http://www.alphalink.com.au/~gnb/maketool/index.html`.

Barton:HSS
Mat Barton.
Hackers, slakers and shackles. the future of free software games development.

`http://www.armchairarcade.com/aamain/content.php?article.78`.

LinuxBase:SW
Linux Base.
Sitio web.

`http://www.linuxbase.org/`

`http://en.wikipedia.org/wiki/Linux_Standard_Base`.

RealBasic:RPA
Real Basic.
Realbasic for porting visual basic applications.

`http://www.realbasic.com/vb`.

Bates:SW
John Bates Clark.
Sitio web.

`http://cepa.newschool.edu/het/profiles/clark.htm`.

Boll:CDC:03
Fundación Heinrich Böll y Sociedad civil alemana.
Carta de los derechos civiles para una sociedad del conocimiento sostenible, 2003.

`http://www.worldsummit2003.de/download_de/Charta-Flyer-espanol.pdf`.

Beck:EPE-01
Kent Beck.
Extreme programming explained, 2001.
ISBN 0-201-61641-6.

Belkin:MCN-05
Sergio Belkin.
Manual compacto para nuevos usuarios de sistemas linux, el uso de sistemas linux en pcs de escritorio, 2005.

`http://www.escritorioya.com.ar/modules/Downloads/Manual/manual.html`.

Benson:P
Dave Benson.
Pkgwrite.

`http://ffem.org/daveb/pkgwrite/`.

Benedetti:DA-85
Mario Benedetti.
Defensa de la alegría.
En El Sur también existe. Joan Manuel Serrat, 1985, 1978.

Benner:MWL-04
Ryan Benner.
Migration from windows to linux saves thousands, 2004.

`http://www.itmanagersjournal.com/software/04/01/09/2231250.shtml`.

Bergara:NDM-04
M. Bergara, A. Harris, O Robles, C. Afonso, R. Echeberria, C. Handley, y P. Twomey.
Notas desde la mesa redonda Gobierno de Internet, Montevideo, abril 2004.

`http://www.latinoamericann.org/modules.php?op=modload&name=News&file=article&sid=552&mode=thread`.

Bass:UCB
Dina Bass y Benedikt Kammel.
Último capítulo de la batalla de munich, Junio 2004.
`http://www.lapastillaroja.net/archives/000449.html`.

Bartsh:SMU
Jörg Bartsch y other.
Suse linux, manual de usuario.

Bodamer:SMA
Frank Bodammer y other.
Suse linux, manual de administración.

OASIS:DXB-05
Michael Brauer y Lars Oppermann.
Developing an xml-based file format specification for office applications, open document format for office applications (opendocument) .odt, 2005.
`http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office`.

Bodnar:WWR
Ladislav Bodnar.
What's wrong with rpm?, 2002.
`http://distrowatch.com/dwres.php?resource=article-rpm`.

Boff:EM:03
Leonardo Boff.
Ética y moral, 2003.
`http://servicioskoinonia.org/boff/articulo.php?num=020`.

Bolsky:USH-82
Morris I. Bolsky.
The unix system user's handbook, 1982.
ISBN 0-13-937764-6.

Boulton:FFS-05
Clint Boulton.
The future of free software lies in the past, 2005.
`http://www.internetnews.com/dev-news/article.php/3508051`.

Boyle:SEM-03

James Boyle.
The second enclosure movement and the construction of the public domain.
66 Law & Contemp. Probs., 2003.

`http://www.law.duke.edu/journals/lcp/articles/lcp66dWinterSpring2003p33.htm`.

Brooks
Fred Brooks.

Bard:NNP-03
Alexander Bard y Jan Söderqvist.
La Netocracia. El nuevo poder en la red y la vida después del capitalismo.
Pearson Educación - Prentice Hall, 2003.
ISBN 84-205-3586-9.

Buanzo:IC
Arturo Buanzo.
Introducción a la consola.

`http://www.buanzo.com.ar/lin/sololinux.html`.

Burcet:SRD-02
Josep Burcet.
Analfabetos digitales, 2002.

`http://www.burcet.net/escenarios/segunda_ruptura.htm`.

Busaniche:SCI
Beatriz Busaniche.
El software cerrado es incompatible con la democracia. entrevista a diego saravia.

`http://www.caminandoutopias.org.ar/contenidos/notas/entrevistas/0014.php`.

Castells:IIT-05
Manuel Castells.
Innovation, information technology and the culture of freedom: the political economy of open source.
Softwarelivre.org, 2005.

`http://www.softwarelivre.org/news/3636`.

Castells:ILP-05

Manuel Castells.
Innovation, libertad y poder en la era de la información.
Softwarelivre.org, 1 2005.

`http://www.softwarelivre.org/news/3635`

Catalan: `http://www.softcatala.org/articles/article53.html`.

CEPA:NTD
CEPA.
The neoclassical theory of distribution.

`http://cepa.newschool.edu/het/essays/margrev/distrib.htm`.

Criado:CIG-00
Sebastian D. Criado y Emiliano Gavilan.
Curso de introducción a gnu/linux. historia, filosofía, instalación y conceptos básicos, 2000.

`http://www.lugro.org.ar/biblioteca/cursos/curso_intro/book1.html`.

DPCCI
la Ciencia y la Cultura Conferencia General de la Organización de las Naciones Unidas para la Educación.
Declaración de los principios de la cooperación cultural internacional, 1966.
Proclamada por la conferencia en su reunión 14, 4 de noviembre de 1966.

`http://www.unhchr.ch/spanish/html/menu3/b/n_decl_sp.htm`.

Chaparro:TTN
Enrique Chaparro.
Izena duen guzia omen da (todo lo que tiene nombre existe).

Chance:SSO-05
Tom Chance.
The social structure of open source development.
Newsforge, 2005.

`http://programming.newsforge.com/programming/05/01/25/1859253.shtml?tid=25&tid=89&tid=91`.

LDC:SW
Linux Distribution Chooser.
Sitio web.

`http://www.zegeniestudios.net/ldc/index.php`.

Chua:MLL-03
Amy Chua.
El mundo en llamas.
Sine Qua Non, 2003.
ISBN 84-406-9739-2.

Ciurcina:NWN-04
Marco Ciurcina.
New world, new laws, new strategies. hacking the international legal system on copyright and patentes., 2004.

`http://bo.unsa.edu.ar/docacad/softwarelibre/articulos/ddhh/`.

Clarke:IGF-05
Gavin Clarke.
Ibm gets own facts out for linux v windows, 2005.

`http://www.channelregister.co.uk/2005/09/01/ibm_windowsvlinux/`.

Locke:TCM-99
Locke Christopher, Rick Levine, Doc. Searls, y David Weinberger.
The cluetrain manifesto, Abril 1999.

`http://www.cluetrain.com/`.

Cohen:NRG-05
Stuart Cohen.
No renegade group behind linux, 2005.

`http://www.newsfactor.com/story.xhtml?story_id=34392`.

Cole:MSB-92
Stephen Cole.
Making Science: Between Nature and Society.
Harvard University Press, 1992.

`http://www.rdg.ac.uk/RevSoc/archive/volume10/number2/10-2c.htm`.

CC:SW
Creative Commons.
Sitio web.

`http://www.creativecommons.org/`

`http://www.freesoftwaremagazine.com/free_issues/issue_01/`.

Connif:HNR-02
Richard Connif.
Historia Natural de los Ricos.
Taurus, 2002.
ISBN 950-511-825-2.

Copani:RDS-01
María Copani.
Reportaje a Diego Saravia, de la UNSa, creador del Ututo. una batalla para abrir las puertas a más gente en internet, 7 2001.

`http://old.clarin.com/diario/2001/07/13/s-285112.htm`

`http://bo.unsa.edu.ar/sct/gestion/docs/softlibre/clarin2.jpg`.

Corazzon:OAR
Raul Corazzon.
Ontology. a resource guide for philosophers.

`http://www.formalontology.it/`.

Cortes:ACP
Carlos Cortes.
Alien: Conversor de paquetes deb, rpm, tgz y slp en linux.

`http://bulma.net/body.phtml?nIdNoticia=1186`.

Cowley:PCB-02
Stacy Cowley.
Panelists consider the 'business case' for open source, Octubre 2002.

`http://www.infoworld.com/articles/hn/xml/02/10/01/021001hnbizopen.html?s=IDGNS`.

Crasta:EPW
James Crasta.
Elf prelinking and what it can do for you.

`http://www.crast.us/james/articles/prelink.php`

`http://linuxcommand.org/man_pages/prelink8.html`.

CSSA
Cyrus sasl for system administrators.

`http://www.sendmail.org/~ca/email/cyrus/sysadmin.html`.

Brasil:GM
Gobierno de Brasil.
Guía de migración brasil.
http://www.hispalinux.es/informes/brasil/E15_469guia_libre_formatado.pdf.

Brasil:PPI-03
Gobierno de Brasil.
Presenta plan para implantación de software libre, 10 2003.
http://noti.hipatia.info/modules.php?op=modload&name=News&file=article&sid=1441&mode=thread&order=0&thold=0
http://www.softwarelivre.org/index.php?menu=mais_noticias2&cod=1065457458&tab=1
http://www.iti.br/twiki/pub/Main/PressRelease2003Oct02B/PlanejamentoSwLivre.pdf
http://www.iti.br/twiki/bin/view/Main/PressRelease2003Oct02B.

Cornell:PC
Universidad de Cornell.
Publicación cientifica.
http://arxiv.org.

Gphoto:SW
Project Gphoto 2 Supporting digital cameras.
Sitio web.
http://www.gphoto.org.

DiCosmo:TC-98
Roberto Di Cosmo.
Trampa en el cyberespacio, 1998.
http://www.pps.jussieu.fr/~dicosmo/Piege/trampas.

COBIT:MR-98
Comité Directivo de COBIT, Information Systems Audit, y Control Fundation.
Cobit, marco referencial, Abril 1998.
2nd ed.

COBIT:OC-98

Comité Directivo de COBIT, Information Systems Audit, y Control Fundation.
Cobit, objetivos de control, Abril 1998.
2nd ed.

COBIT:RE-98
Comité Directivo de COBIT, Information Systems Audit, y Control Fundation.
Cobit, resumen ejecutivo, Abril 1998.
2nd ed.

Incasol:SW
Instituto de Capacitación en Software Libre.
Sitio web.

`http://www.incasol.org.ar/index.php`.

CCTJE:L
Consejeria de Ciencia y Técnica de la Junta de Extremadura.
Linex.

`http://www.linex.org`

Aulas virtuales literarias: `http://www.juntaex.es/consejerias/ect/gaceta/712003/paginas/sinformacion26y27.html`

Experiencias con Linex: `http://es.openoffice.org/unbranded-source/browse/~checkout~/es/www/lecturas/exper-1.html`.

COBIT:AI-00
Comité de Dirección de COBIT y IT Governance Institute.
Cobit v3, adquisición e implementación, Julio 2000.
3er ed.

COBIT:A-00
Comité de Dirección de COBIT y IT Governance Institute.
Cobit v3, apendice i, Abril 2000.
3er ed.

COBIT:ES-00
Comité de Dirección de COBIT y IT Governance Institute.
Cobit v3, entrega y soporte, Abril 2000.
3er ed.

COBIT:M-00

Comité de Dirección de COBIT y IT Governance Institute.
Cobit v3, monitoreo, Abril 2000.
3er ed.

COBIT:PG-00
Comité de Dirección de COBIT y IT Governance Institute.
Cobit v3, pautas de gestión, Julio 2000.
3er ed.

COBIT:PO-00
Comité de Dirección de COBIT y IT Governance Institute.
Cobit v3, planificación y organización, Julio 2000.
3er ed.

PTCIDH
Conferencia Internacional de Derechos Humanos en Teherán.
Proclamación de teherán, 1968.
Proclamada el 13 de mayo de 1968, note=

`http://www.unhchr.ch/spanish/html/menu3/b/b_tehern_sp.htm`.

Dan:LCN
Jeffrey Dean.
Lpi certification in a nutshell, 2001.
ISBN 1-56592-748-6

`http://www.lpi.org/en/lpic.html`

`http://www-106.ibm.com/developerworks/edu/l-dw-linux-lpir21-i.html`

`http://www.vue.com/lpi/`

`http://www.vue.com/servlet/vue.web2.core.Dispatcher?webViewApp=TestCenterLocator&countryList=VEN&webViewID=10028252&Continue`

Debian:HMP
How to make deb packages.

`http://linuxdevices.com/articles/AT8047723203.html`.

Debian:FAI
Debian.
Fai (fully automated installation).

`http://www.informatik.uni-koeln.de/fai`.

Debian:SW
: Debian.
Sitio web.

`http://www.debian.org/`.

Debian:DFSL-98
: Proyecto Debian.
What does free mean? or what do you mean by free software?, the debian free software guidelines dfsg, 1998.

`http://www.debian.org/intro/free`

`http://www.debian.org/social_contract#guidelines`.

Dell:CTO-03
: Dell.
Costo Total de Operación. ¿Que es realmente?, 2003.

`http://www.taxadmin.org/fta/meet/tw03_pres/barr2.pdf`.

DelBianco:FSF-05
: Steve DelBianco.
Free software isn't free, 2005.

`http://www.heartland.org/Article.cfm?artId=16181`.

Desconocido:OER
: Desconocido.
Ontología y epistemología / realismo e idealismo / sujeto y objeto.

`http://www.filosofia.org/filomat/df087.htm`.

Dennet:TMI-81
: Daniel C. Dennet y Douglas Hofstadter.
The mind's i: Fantasies and reflections on self and soul, 1981.

`http://www.amazon.com/gp/reader/0553345842/ref=sib_rdr_next1_1/103-0117524-7288640?`.

Giait:DII
: Grupo de Informática Aplicada al Inglés Técnico (g.i.a.i.t.).
Diccionario informático inglés castellano.

`http://es.tldp.org/htmls/otros.html`

`http://es.tldp.org/Otros/diccionario-us-es/diccionario-us-es-0.1.6/`.

Abadia:MCU-05

Abadia Digital.

Microsoft considera que el uso del software libre perjudica a la economía, 2005.

`http://www.abadiadigital.com/modules.php?op=modload&name=News&file=article&sid=545`.

Distromania:SW

Distromanía.

Sitio web.

`http://www.distromania.com/`.

Distrowatch:SW

Distrowatch.

Sitio web.

`http://distrowatch.com/`.

PIDCP

Asamblea General de la NNUU.

Pacto internacional de derechos civiles y políticos.

Adoptado y abierto a la firma, ratificación y adhesión por la Asamblea General en su resolución 2200 A (XXI), de 16 de diciembre de 1966. Entrada en vigor: 23 de marzo de 1976, de conformidad con el artículo 49.

`http://www.unhchr.ch/spanish/html/menu3/b/a_ccpr_sp.htm`.

DUDDHH

Asamblea General de las NNUU.

Declaración universal de derechos humanos, 1948.

Adoptada y proclamada por la Asamblea General en su resolución 217 A (III), de 10 de diciembre de 1948.

`http://www.unhchr.ch/udhr/lang/spn.htm`.

PIDESC

Asamblea General de las NNUU.

Pacto internacional de derechos económicos, sociales y culturales, 1966.

Adoptado y abierto a la firma, ratificación y adhesión por la Asamblea General en su resolución

2200 A (XXI), de 16 de diciembre de 1966 Entrada en vigor: 3 de enero de 1976, de conformidad con el artículo 27.

`http://www.unhchr.ch/spanish/html/menu3/b/a_cescr_sp.htm`.

DUPCTP
: Asamblea General de la Organización de las Naciones Unidas.
Declaración sobre la utilización del progreso científico y tecnológico en interés de la paz y en beneficio de la humanidad, 1975.
Resolución 3384, 10 de noviembre de 1975.

`http://www.unhchr.ch/spanish/html/menu3/b/70_sp.htm`.

DDD
: Asamblea General de la Organización de las Naciones Unidas.
Declaración sobre el derecho al desarrollo, 1986.
Resolución 41/128 del 4 de diciembre de 1986.

`http://www.unhchr.ch/spanish/html/menu3/b/74_sp.htm`.

AntiDMCA:SW
: Anti DMCA.
Sitio web.

`http://anti-dmca.org/`.

PDVSA:I-05
: Equipo de migración de Software Libre.
Inventario de aplicaciones, 2005.

PDVSA:PMS-05
: Equipo de migración de Software Libre.
Plan de migración al software libre, 2005.

UNYB:RSL-03
: Universidad de Nueva York en Buffalo.
Resolución sobre el software libre, 2003.

`http://orange.math.buffalo.edu/csc/resolution2_april2003_approved.html`.

Oviedo:IEC
: Universidad de Oviedo.
Incorpora a sus estatutos la cuestión de la igualdad del acceso y el software libre.

`http://noti.hipatia.info/modules.php?op=modload&name=News&file=article&sid=1331`.

DiBona:OSV-99
Chris DiBona, Sam Ockman, y Mark Stone.
Open Sources: Voices from the Open Source Revolution.
O'Reilly, 1999.

ISBN 1-56592-582-3 `http://www.oreilly.com/catalog/opensources/book/toc.html`

`http://www.sindominio.net/biblioweb/telematica/open-sources-html`.

Pentima:IUG-00
Lucas Di Pentima y Nicolás Cesar.
Introducción al uso de gnu/linux, 2000.

`http://g.unsa.edu.ar/lucas/basico/curso.html`.

Dulaney:ENL-02
Emmett Dulaney.
Examining the new lpi level one exams, certification: Exploring the new lpi 101 exam, 2002.

`http://www.unixreview.com/documents/s=7695/uni1036090509292/`

`http://www.unixreview.com/documents/s=7750/uni1038932969999/`.

Dumbill:MDP-05
Edd Dumbill.
Mozilla as a development platform: An interview with axel hecht, 2005.

`http://www.oreillynet.com/pub/a/network/2005/09/02/axel-hecht.html`.

Dutra:SFS-02
Olivio Governador Dutra.
Softwares fechados são incompatíveis com a democracia, 2002.

`http://www.softwarelivre.org/news/385`

`http://www.softwarelivre.org/news/393`.

Venezuela:DSL-04
Gobierno de Venezuela.
Decreto sobre el uso del software libre en la administración pública, 3390, 2004.

`http://www.sapi.gov.ve/web/index.php?option=com_content&task=view&id=64&Itemid=92`.

Davila:TDB-02
: Jaime Irving Dávila.
Tutorial de docbook, un enfoque integrado y a través de ejemplos, 2001.

`http://es.tldp.org/Tutoriales/DOCBOOK/doctut/single-html/dbktut.html`.

Easy:PPS
: EasySW.
Purchase printing software supporting linux.

`http://www.easysw.com/printpro`.

Linuxmag:SR-99
: Editors of Linux Magazine.
Saint richard, 1999.

`http://www.linux-mag.com/1999-07/stallman_01.html`.

Alaco:SW
: Alaco: especialistas en migraciones.
Sitio web.

`http://www.alaco.com`.

Versora:SW
: Versora: especialistas en migraciones.
Sitio web.

`http://www.versora.com`

`http://www.free-press-release.com/news/200506/1119550516.html`.

Eff:TC
: Eff.
Trusted computing.

`http://www.eff.org/Infrastructure/trusted_computing/`

`http://www.eff.org/Infrastructure/trusted_computing/20031001_tc.php`.

Ellison:SW

Lary Ellison.
Softwar, oracle.

Evangelista:AC-04
Rafael Evangelista.
Omc: Organización mundial de comercio. apelando al capataz, Diciembre 2004.

`http://www.redvoltaire.net/article3304.html`.

FSFE:ADG-04
FSFEuropa et al.
Apoyo a la declaración de génova, 2004.

`http://mailman.fsfeurope.org/pipermail/press-release-es/2004q4/000004.html`

`http://fsfeurope.org/documents/wiwo.es.html`.

Fedora:ISL
Fedora.
Introduction to stateless linux, proposal for the fedora project.

`http://fedora.redhat.com/projects/stateless/`.

Craig:CTE-99
Craig Finseth.
The craft of text editing. emacs for the modern world, 1999.

`http://www.finseth.com/craft/`.

MUSCLE:SW
M.U.S.C.L.E. Project for integrating SmartCards.
Sitio web.

`http://www.linuxnet.com`.

TBCIS:JPB-00
The Berkman Center for Internet & Society, 2000.
The Debate Over Internet Governance: A Snapshot in the Year 2000. title=John Perry Barlow,

`http://cyber.law.harvard.edu/is99/governance/barlow.html`.

TBCIS:IS99-00
The Berkman Center for Internet & Society.
I&s 99 course description, 2000.
The Debate Over Internet Governance: A Snapshot in the Year 2000.

`http://cyber.law.harvard.edu/is99/desc.html`.

TBCIS:SW-00
: The Berkman Center for Internet & Society.
Sitio web, 2000.
The Debate Over Internet Governance: A Snapshot in the Year 2000.

`http://cyber.law.harvard.edu/is99/governance/introduction.html`.

Fogel:POS-05
: Karl Fogel.
Producing Open Source Software. How to Run a Successful Free Software Project.
2005.

`http://producingoss.com/producingoss.html`.

Fontana:GSI-05
: Claudio Fontana.
Gnu source installer, 2005.

`http://www.gnu.org/software/sourceinstall` .

FSF:SW
: Free Software Foundation.
Sitio web.

`http://www.fsf.org/`.

FSF:MCA-05
: Free Software Foundation.
Richard stallman dice que microsoft compra el apoyo de estados e instituciones, 2005.

`https://e.ututo.org.ar/xp/modules/news/article.php?storyid=153`.

FrozenTech:LL
: FrozenTech.
Livecd list.

`http://www.frozentech.com/content/livecd.php`.

LFS:SW
: Linux from scratch.
Sitio web.

`http://www.linuxfromscratch.org/`.

ISC:SW
: Initiative for Software Choice.
Sitio web.

`http://www.softwarechoice.org/`

`http://www.softwarechoice.org/download_files/CPR.Comments.pdf`.

FSF:CSL
: FSF.
Categorías de software libre y no libre.

`http://www.gnu.org/philosophy/categories.es.html`.

FSF:DSL
: FSF.
Definicion de software libre.

`http://www.gnu.org/philosophy/free-sw.es.html`.

FSF:FPG
: FSF.
Filosofía del proyecto gnu.

`http://www.gnu.org/philosophy/philosophy.es.html`.

FSF:HPG
: FSF.
Historia del proyecto gnu.

`http://gnu.open-mirror.com/gnu/gnu-history.es.html`.

FSF:VLC
: FSF.
Various licenses and comments about them.

`http://www.fsf.org/licenses/license-list.html`

`http://www.gnu.org/licences/licences.es.html`.

FSF:MG-85
: FSF.
Manifiesto gnu, 1985.

`http://www.gnu.org/gnu/manifesto.es.html`.

FSF:QC-98

FSF.
¿que es copyleft?
Boletin GNU, 1(6), 1998.

`http://www.gnu.org/bulletins/bull6.html`.

CPD:SW
Center for the Public Domain.
Sitio web.

`http://www.centerforthepublicdomain.org/`.

GAIM:SW
GAIM.
Sitio web.

`http://gaim.sourceforge.net`.

Garret:ANA-05
Jesse James Garrett.
Ajax: A new approach to web applications, 2005.

`http://www.adaptivepath.com/publications/essays/archives/000385.php`

`http://developer.mozilla.org/en/docs/AJAX`.

Gates:ESA
Bill Gates.
Ellos serán adictos, Bill Gates, adicción y distribución ilegal en china, 7 1998.

`http://www.softwarelivre.org/news/2549`

`http://www.softwarelivre.org/news/2557`

`http://64.233.161.104/search?q=cache:http://archives.cnn.com/2000/TECH/computing/02/23/microsoft.china.idg/`.

Gates:QCS-04
Bill Gates.
Los que quieren cambiar el sistema de derechos intelectuales son comunistas, 2004.

`http://news.com.com/Gates+taking+a+seat+in+your+den/2008-1041_3-5514121-4.html?tag=st.num`

`http://www.smaldone.com.ar/documentos/docs/gates_comunistas.html`

`http://news.com.com/Bill+Gates+and+other+communists/2010-1071_3-5576230.html`

`http://webmastercristiano.com/articulo/2/171`

`http://www.pilas.net/?p=51`

`http://www.diarioti.com/gate/n.php?id=7955`.

GBDirect:ASA
: GBDirect.
Advanced system administration (lpi 201 certified) - a 5 day course.

`http://training.gbdirect.co.uk/courses/linux/LPI_201_linux_certified_system_administration.html`.

GBDirect:LUF
: GBDirect.
Linux and unix fundamentals (lpi 101/102) - a 5-day course.

`http://training.gbdirect.co.uk/courses/linux/fundamentals.html`.

GBDirect:RLE
: GBDirect.
Running linux in the enterprise: Network administration (lpi 202) - a 4-day course.

`http://training.gbdirect.co.uk/courses/linux/running_linux_in_the_enterprise.html`.

Gentoo:EH
: Gentoo.
Ebuild howto.

`http://www.gentoo.org/proj/en/devrel/handbook/handbook.xml?part=2&chap=1`.

Gentoo:API
: Gentoo.
A portage introduction.

`http://www.gentoo.org/doc/en/handbook/handbook-x86.xml?part=2&chap=1`.

Gentoo:SW
: Gentoo.
Sitio web.

`http://www.gentoo.org/`.

Gentoo:WWP
: Gentoo.
Working with portage.

`http://www.gentoo.org/doc/en/handbook/handbook-x86.xml?part=3`.

Genovese:PMS-05
: María Amparo Genovese.
Presentación, proyecto de migración al software libre, 2005.

Gibbs:SCC-94
: Walter Gibbs.
Software's chronic crisis, 1994.

`http://www.cis.gsu.edu/~mmoore/CIS3300/handouts/SciAmSept1994.html`.

Gillmor:CWC-02
: Dan Gillmor.
10 choices that were critical to the net's success, Setiembre 2002.

`http://www.siliconvalley.com/mld/siliconvalley/business/columnists/4029770.htm`.

GIMP:SW
: The GIMP.
Sitio web.

`http://www.gimp.org`.

Glickstein:GEE-77
: Bob Glickstein.
GNU Emacs Extensions.
O'Reilly, 1977.
ISBN 1-56592-261-1.

GNOME:SW
: GNOME.
Sitio web.

`http://www.gnome.org`

Significant GNOME desktop applications

`http://www.gnomefiles.org`,

GDesklets

`http://gdesklets.gnomedesktop.org/`.

GNU:GL
GNU.
Gnu libtool.

`http://www.gnu.org/software/libtool/manual.html`.

GNU:GCS-04
GNU.
Gnu coding standards, 2004.

`http://www.gnu.org/prep/standards_toc.html`.

Google:SW
Google.
Sitio web.

`http://www.google.com`.

Gossen:DLE-54
Heinrich Hermann Gossen.
The development of the laws of exchange among men and of the consequent rules of human action, 1854.

`http://cepa.newschool.edu/het/profiles/gossen.htm`.

Green:HHW
Sheldon Green.
Hypertext help with 1995.

`http://www-h.eng.cam.ac.uk/help/tpl/textprocessing/teTeX/latex/latex2e-html/`.

IPJustice:WPA
Robin D. Gross.
Ip justice white paper on the draft intellectual property rights chapter in the free trade area of the americas treaty: ``ftaa: A threat to freedom and free trade".

`http://ipjustice.org/FTAA/IPJ_FTAA_White_Paper_r_1_2.html`

`http://ipjustice.org/FTAA/release20031020.shtml`.

TCG:SW
Trusted Computing Group.
Sitio web.
`http://www.trustedcomputinggroup.org/home`.

Gruber:WIO-93
Tom Gruber.
What is an onthology, 1993.
`http://www-ksl.stanford.edu/kst/what-is-an-ontology.html`.

Gutierrez:EMT
Claudio Gutiérrez.
Ética y moral: teorías y principios.
`http://www.claudiogutierrez.com/Introduccion_a_la_etica.html`.

Hall:TCO-02
Mark Hall.
Tco: Linux delivers on big iron, 2002.
`http://www.computerworld.com/hardwaretopics/hardware/story/0,10801,70944,00.html`.

Hamilton:SUT
Bruce Hamilton.
A sysadmin's universal translator (rosetta stone).
`http://bhami.com/rosetta.html`.

Hartmann:NO-54
Nicolai Hartmann.
La nueva Ontología.
Editorial Sudamericana, 1954.

RedHat:LHP
Red Hat.
Linux hardware probing library, kudzu.
`http://rhlinux.redhat.com/kudzu`.

RedHat:RAC
Red Hat.
Redhat academy linux curriculum.

`https://www.redhat.com/training/academy/curriculum.html`.

Hayes:DRT-04
Ian Hayes.
Determining ROI, TCO and other key financial metrics, 2004.
`http://www.clarity-consulting.com/DeterminingROI&TCO.pdf`.

Himanen:EHE-02
Pekka Himanen.
La ética del hacker y el espíritu de la era de la información.
Destino, 2002.
Prólogo de Linus Torvalds, Epílogo de Manuel Castells, ISBN 8423333906.

Hipatia:SW
Hipatia.
Sitio web.
`http://www.hipatia.info`
`http://en.wikipedia.org/wiki/Hipatia`.

HIP:TIM-97
The internet manifesto, 1997.
`http://hippy.com/manifesto.htm`.

Hipatia:PPL-04
Hipatia.
Proposta de uma politica livre para ti no fórum social mundial, 2004.
`http://www.softwarelivre.org/news/3245`.

Hipatia:SM-04
Hipatia.
Segundo manifiesto, 2004.
`http://www.hipatia.info/index.php?id=manifesto2_es`.

Hispalinux:IEE
Hispalinux.
Informes de estudio y evaluación de software libre.
`http://www.hispalinux.es/informes/index.html`.

Hispalinux:SW

Hispalinux.
Sitio web.

`http://www.hispalinux.es`.

Horstmann:SSL-03
Jutta Horstmann, Jan Muehlig, Eva Brucherseifer, y Ralf Ackermann.
User experience architecture, success story of linux usability on the desktop, Agosto 2003.

`http://www.linux-usability.de/download/linux_usability_report_en.pdf`.

Hunt:LPI
Kenneth Hunt.
Lpi certification 101 exam prep. ibm has some good resources for preping for your lpi.

`http://kennethhunt.com/archives/001329.html`.

Hunt:TNA-92
Craig Hunt.
Tcp/ip network administration, 1992.
ISBN 0-937175-82-X.

IBM:CDL
IBM.
Course description: Linux lpi level 1 certification preparation workshop.

`http://www-304.ibm.com/jct03001c/services/learning/ites.wss/us/en?pageType=course_description&courseCode=QLX37`.

IBM:IRU
IBM.
Ibm redpaper using asset depot for inventory management.

`http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/redp3763.html?Open`.

IBM:LIS
IBM.
Linux at ibm solutions page.

`http://www.ibm.com/linux/solutions`.

IBM:LCM

IBM.
Linux client migration cookbook. a practical planning and implementation guide for migrating to desktop linux.

`http://www.redbooks.ibm.com/redbooks/SG246380/wwhelp/wwhimpl/common/html/switch.htm` ISBN 0738491527, IBM Form Number: SG24-6380-00.

IBM:PMS
IBM.
Pc model types suitable for linux.

`http://www.ibm.com/pc/support/site.wss/MIGR-48NT8D.html`.

IBM:PCS
IBM.
Personal computing support: Linux for ibm personal systems.

`http://www-306.ibm.com/pc/support/site.wss/MIGR-48NT8D.html`.

IBM:PMA
IBM.
Porting mfc applications to linux: A step-by-step guide to using wxwindows.

`http://www-106.ibm.com/developerworks/linux/library/l-mfc`.

IBM:TPL
IBM.
Training path, preparation for lpi certification.

`http://www-304.ibm.com/jct03001c/services/learning/ites.wss/us/en?pageType=page&c=a0000592`.

IDA:MM
IDA.
Manuales de migración.

`http://europa.eu.int/idabc/en/document/1921`.

IDC:W2L-02
IDC.
Windows 2000 versus linux in enterprise computing, 2002.

`http://www.microsoft.com/windows2000/docs/TCO.pdf`.

iFolder:SW

iFolder.
Sitio web.

`http://www.novell.com/products/ifolder`.

Iglesias:ISE-98
Enrique Iglesias.
Impacto del software empacado en las economías latinoamericanas.
Technical report, Informe de Price Waterhouse para la BSA, 1998.

`http://global.bsa.org/usa/globallib/econ/laspanish98.pdf`.

IHRTIC
International human rights treaties and internet censorship.

`http://www.earlham.edu/~pols/ps17971/asia1/mayro.html`.

Li:SW
Linux International.
Sitio web.

`http://www.li.ar`.

Intriligator:OMT
Michael Intriligator.
Optimización matemática y teoría económica.
Prentice Hall.

Intellpuke:SCR-05
Intellpuke.
Supreme court rules against file-sharing websites, Junio 2005.

`http://freeinternetpress.com/modules.php?name=News&file=article&sid=3895`.

Isenberg:RSN-97
David Isenberg.
Rise of the stupid network.
Computer Telephony, pages 16-26, Agosto 1997.

`http://www.isen.com/stupid.html`

`http://www.rageboy.com/stupidnet.html`.

LinuxISO:SW

Linux ISO.
Sitio web.

`http://www.linuxiso.org/`.

CRIS:SW
Communication Rights in the Information Society (CRIS).
Sitio web, campaña.

`http://www.crisinfo.org/`

`http://www.democraciadigital.org/2003/0521/opinion/derecho_a_comunicarse.html`.

Isenberg:TPB
David Isenberg y David Weinberger.
The paradox of the best network.

`http://www.netparadox.com/netparadox.html`.

Jackson:LCB-04
Joab Jackson.
Linux now a corporate beast, 2004.

`http://www.gcn.com/vol1_no1/daily-updates/26641-1.html`,

`http://www.linuxjournal.com/node/7725/print`.

Jorge:MLE-04
Jorge.
Mactivismo: la libertad de elección, 2004.

`http://www.faq-mac.com/mt/archives/007919.php`.

Jackson:DPM-96
Ian Jackson y Christian Schwarz.
Debian policy manual, 1996.

`http://www.debian.org/doc/debian-policy/`.

Jonsson:NCC-96
Peder Jonsson y Frank Sharman.
Nobody can control the internet, 1996.

`http://hem.passagen.se/peder/essay1.htm`.

Judicial:INC

Fallo Judicial.
Las ideas no son cosas.

`http://www.tectimes.com/secciones/notas.asp?codnota=13962`.

Jungman:ECC
G. Jungman.
Exceptional conditions and contextual information in numerical computer.

`http://t8web.lanl.gov/people/jungman/except.pdf`.

Kachurov:LES-03
Valery V. Kachurov y Nesov Artem.
Lista de equivalencias de software: Windows - gnu/linux, 2003.

`http://linuxshop.ru/linuxbegin/win-lin-soft-spanish/index.shtml`,

`http://www.linuxrsp.ru/win-lin-soft/index-spanish.html`.

Kantor:OEP-05
Damián Kantor.
Otra estrategia contra la piratería.

`http://www.clarin.com/suplementos/economico/2005/05/08/n-00401.htm`.

Kirch:LNG-00
Olaff Kirch y Terry Dawson.
Linux network administrator's guide, 2000.
ISBN 1-56592-400-2

`http://www.faqs.org/docs/linux_network/`

`http://es.tldp.org/Manuales-LuCAS/GARL2/garl2/`.

KDE:KSA
KDE.
Kde for system administrators guide.

`http://www.kde.org/areas/sysadmin/`.

KDE:SW
KDE.
Sitio web.

`http://www.kde.org`.

Kroah:LKD
Greg Kroah-Hartman.
Howto do linux kernel development - take 2.
`http://lwn.net/Articles/160191/`.

Kiosk:SW
Kiosk.
Sitio web.
`http://extragear.kde.org/apps/kiosktool.php`
`http://webcvs.kde.org/cgi-bin/cvsweb.cgi/kdelibs/kdecore/README.kiosk`.

Kitenet:ACD
Joey Kitenet.
A comparison of the deb, rpm, tgz, and slp package formats/comparing linux/unix binary package formats.
`http://debian-br.sourceforge.net/txt/alien.html`
`http://www.kitenet.net/~joey/pkg-comp/`.

Knight:SUG-04
Steven Knight.
Scons user guide 0.96, 2004.
`http://www.scons.org/doc/HTML/scons-user/book1.html`.

Knuth:ACP
Knuth.
The art of computer programing.

Knuth:CSP
Donald Knuth.
Carta sobre las patentes.
`http://bachue.com/colibri/patentes/knuth-pto/knuth-pto.es.txt`
`http://www.progfree.org/Patents/knuth-to-pto.txt`.

Kojima:RPA
Alfredo K. Kojima.
An rpm port of apt, 2000.

`http://freshmeat.net/articles/view/192/`.

Kernighan:EPU
: Brian Kernighan y Rob Pike.
El entorno de programacion unix.

Kernighan:LPC
: Brian Kernighan y Ritchey.
El lenguaje de programacion c.

GuruLabs:LSAC
: Guru Labs.
Linux systems administrator courses.

`http://www.gurulabs.com/training/courses.php`.

LaMonica:OSL:05
: Martin LaMonica.
Open-source lamp a beacon to developers, 2005.

`http://news.com.com/Open-source+LAMP+a+beacon+to+developers/2100-7344_3-5744767.html?tag=sas.email`.

Lanier:PF-99
: Jaron Lanier.
Piracy is your friend, 1999.

`http://www.maui.net/~zen_gtr/zgzinepg4.html`.

Lenin:CTN-19
: Vladimir Lenin.
El comunismo y la transformación de la naturaleza por el trabajo. Inauguración del plan general soviético de electrificación, 1919.
El comunismo es los soviets mas la electricidad. ``software libre + internet = sociedad del conocimiento'', ``ideología+tecnología=nuevo mundo''.

`http://www.perspectivamundial.com/2001/2509/250908.shtml`

`http://www.ainfos.ca/00/mar/ainfos00080.html`.

Leontief:AEI
: Wassily Leontief.
Analisis Económico Input-Output.
Ariel.

Lessig:C-99

Lawrence Lessig.
El Codigo.
Taurus es digital, 1999.
ISBN 8430604286,

`http://www.lessig.org/`.

LinuxK:SW
Kernel Linux.

`http://www.kernel.org/`.

LinuxC:SW
Linux.com.

`http://www.linux.com/`.

LinuxO:SW
Linux.org.

`http://www.linux.org/`.

LinuxQuestions:DR
LinuxQuestions.
Distro reviews.

`http://www.linuxquestions.org/reviews/index.php`.

Psl:SW
Proyecto Software Livre.
Sitio web.

`http://www.softwarelivre.org`.

Lessig:OAF
Lessig y Lemley.
Open access to the fcc.

`http://cyber.law.harvard.edu/works/lessig/cable/fcc/fcc.html`.

Loukides:PGS-97
Mike Loukides y Andy Oram.
Programming with gnu software, 1997.
ISBN 1-56592-112-7.

Liu:MII-94

Cricket Liu, Jerry Peek, Russ Jones, Bryan Buus, y Adrian Nye.
Managing internet information services, 1994.
ISBN 1-56592-062-7.

LWN:LDL
LWN.
The lwn.net linux distribution list.

`http://old.lwn.net/Distributions/index.php3`.

Openwebmail:SW
Open Web Mail.
Sitio web.

`http://openwebmail.org/`.

Malone:RIP-05
Michael S. Malone.
R.i.p. microsoft? after dominating the technology industry for years, is microsoft poised to collapse?, 2 2005.

`http://www.googlecommunity.com/about2839.html`.

Mandriva:SW
Mandriva.
Sitio web.

`http://www.mandriva.com/`.

Maquiavelo:P
Maquiavelo.
El príncipe.

Martin:SRB-03
Mike Martin.
Scientific research backs wisdom of open source, 2003.

`http://www.sci-tech-today.com/perl/story/22862.html`.

Mattelart:HSI-02
Armand Mattelart.
Historia de la Sociedad de la Información.
Paidos Comunicación, 1 edition, 2002.
ISBN 9501275329.

McCullagh:SUN-04

Declan McCullagh.
Should the united nations run the internet?, Marzo 2004.

`http://news.com.com/Should+the+United+Nations+run+the+Internet010-1028_3-5181327.html`.

Mello:SIH
Leonardo Mello.
Sistema de inventario de hardware y software. cacic.

`http://www.cetico.org/cacic/doku.php`.

Mertonian:SN
Mertonian.
Science norms: Cudos, the mertonian sociological tradition of open science.

`http://bio-oak.mercer.edu/courses/huber%20sci_copy(1)/Mertonian%20norms.doc`.

Merten:GHS-00
Stefen Merten.
Gnu/linux: Un hito en el camino hacia la sociedad gpl, 2000.

`http://www.oekonux.org/texts/meilenstein/spanish.html`.

Marx:MC-48
K. Marx y Engels F.
Manifiesto del partido comunista, 1848.

`http://www.ucm.es/info/bas/es/marx-eng/47mpc/`.

Microsoft:ASA-2004
Microsoft.
Microsoft acusa a Sergio Amadeu de difamación, 2004.

`http://portal.softwarelivre.org/news/2607`

`http://www.softwarelivre.org/news/2479`.

Mas:SLS-03
Jordi Mas i Hernàndez.
Software libre en el sector público, 2003.

`http://www.uoc.edu/dt/20327/index.html`.

Miller:RMC

Peter Miller.
Recursive make considered harmful.

`http://www.pcug.org.au/~millerp/rmch/recu-make-cons-harm.html`.

MIT:LDC
MIT.
La libre difusión de sus conocimientos.

`http://news.com.com/2100-1023-961563.html?tag=m`.

Mitchell:ALA-99
Si Mitchell.
A local alternative, si mitchell questions the wisdom of export driven economies, 1999.
http://www.urban75.com/Action/seattle7.html.

Moglen:ATS
Eben Moglen.
Anarquismo triunfante. software libre y la muerte del copyright.

`http://old.law.columbia.edu/my_pubs/anarchism.html`.

Moglen:FSM-00
Eben Moglen.
Free software matters: Free software or open source?, 2000.

`http://emoglen.law.columbia.edu/publications/lu-07.html`.

Mono:SW
Mono.
Sitio web.

`http://www.mono-project.com`.

monografias:TCE
Monografias.com.
Teoría del conocimiento / epistemología.

`http://www.monografias.com/trabajos/epistemologia2/epistemologia2.shtml`.

Mysql:DLP
Mysql.
Licensing policy.

`http://www.mysql.com/company/legal/licensing/`.

Nagios:SW
: Nagios.
Sitio web.

`http://sourceforge.net/projects/nagios`

`http://www.nagios.org/`.

Nguyen:CCD-04
: Bihn Nguyen.
A constructive critique of debian linux, 2004.

`http://desktoplinux.com/articles/AT7588639943.html`.

Novell:EC
: Novell.
Exchange connector.

`http://www.novell.com/products/connector/`.

Novell:NM
: Novell.
The novell migration to linux white paper.

`http://www.novell.com/collateral/4621400/4621400.html`.

Novell:SLT
: Novell.
Suse linux training.

`http://www.novell.com/training/train_product/suse.html`.

Novell:ZW
: Novell.
Zenworks 6.5 linux management.

`http://www.novell.com/documentation/zenworks65/index.html`

`http://www.novell.com/products/zenworks/linuxmanagement/`

`http://www.novell.com/products/zenworks/eval.html`.

nomachine:SW
: NoMachine NX.
Sitio web.

`http://www.nomachine.com`.

Odum:AES-80
H. T. Odum.
Ambiente, Energía y Sociedad.
Blume, 1980.
ISBN 84-7031-237-5 .

Oekonux:SW
Oekonux.
Sitio web.

`http://www.oekonux.org`

`www.oekonux.de`.

OpenOffice:SW
Open Office.
Sitio web.

`http://www.openoffice.org`.

OpenMosix:SW
OpenMosix.
Sitio web.

`http://www.openmosix.org`.

OpenPKG:SW
OpenPKG.
Sitio web.

`http://www.openpkg.org/`

`http://www.openpkg.org/doc/articles/sysadmin/article.html`.

opensourcecms:SW
OpenSourceCMS.
Sitio web.

`http://www.opensourcecms.com`.

Oreilly:I-05
Tim O Reilly.
Interview, 2005.

`http://news.bbc.co.uk/1/hi/technology/4372728.stm`.

Orlowski:TSM-04
Andrew Orlowski.
Triple setback for music giants' global jihad, Abril 2004.
`http://www.theregister.co.uk/content/6/36712.html`.

Orzech:TCO-02
Dan Orzech.
Linux TCO: Less than half the cost of windows, 10 2002.
`http://www.cioupdate.com/article.php/10493_1477911`.

OSI:SW
OSI Open Source Initiative.
Sitio web.
`http://www.opensource.org/`.

OSI:AL
OSI.
The approved licenses.
`http://www.opensource.org/licenses`.

OSI:OSD
OSI.
The open source definition.
`http://www.opensource.org/docs/definition.php`.

Ourproject:ECL
Ourproject.
Ecohacktivismo: Conocimiento libre ecológico como otro modelo de activismo.
`http://ourproject.org/cgi-bin/moin.cgi/Versi_f3n_20para_20hackers`.

Outaline:PCP-91
Steve Outaline.
Practical c programming, 1991.
ISBN 1-56592-035-X.

p2pnet:TPE
p2pnet.
Tcpa - palladium, the end? or the beginning?

`http://www.p2pnet.net/issue04/issues.html`.

GAUEE:SW
: Grupo para asesorar a la UE sobre aspectos éticos de la ciencia y la tecnología.
Sitio web.

`http://europa.eu.int/comm/european_group_ethics/index_en.htm`.

Murugan:PSO-05
: Murugan Pal.
Proprietary software to open source - migration approach, 2005.

`http://www.onlamp.com/pub/wlg/8510`.

Parrot:SW
: Parrot.
Sitio web.

`http://www.parrotcode.org`.

Perens:SC
: Bruce Perens.
Sincere choice.

`http://www.sincerechoice.org/`.

Perens:TTA-99
: Bruce Perens.
It's time to talk about free software again, 1999.

`http://lists.debian.org/debian-devel/1999/02/msg01641.html`.

R2000SPPDH
: Subcomisión para la Promoción y Protección de los Derechos Humanos del Alto Comisionado de las Naciones Unidas para los Derechos Humanos.
Intellectual property rights and human rights. resolucion 2000/7.

`http://www.unhchr.ch/Huridocda/Huridoca.nsf/0/c462b62cf8a07b13c1256970046704e?Opendocument`.

R2001SPPDH
: Subcomisión para la Promoción y Protección de los Derechos Humanos del Alto Comisionado de las Naciones Unidas para los Derechos Humanos.
Intellectual property rights and human rights. resolucion 2001/21.

`http://www.unhchr.ch/Huridocda/Huridoca.nsf/.2.RES.2001.21.En?Opendocument`.

Poirier:PSR-01
: Dan Poirier.
Packaging software with rpm, 2001.

`http://www-128.ibm.com/developerworks/library/l-rpm1/index.html`

`http://www-128.ibm.com/developerworks/library/l-rpm2/index.html`

`http://www-128.ibm.com/developerworks/library/l-rpm2/index.html`.

polinux:KOI
: Polinux.
Kde openoffice integration and mozillux projects.

`http://www.polinux.upv.es/mozilla`

`http://kde.openoffice.org`.

Peek:UPT-93
: Jerry Peek, Tim O Reilly, y Mike Loukides.
Unix power tools, 1993.

ISBN 0-679-79073-X.

Postgres:SW
: Postgres.
Sitio web.

`http://www.posgressql.org`.

Postcarcity:SW
: Postscarcity.
Sitio web.

`http://c2.com/cgi/wiki?PostScarcity`.

Pourailly:WSU-04
: María José López Pourailly.
Web semántica, unión europea persigue producir el tránsito de la sociedad de la información a la sociedad basada en el conocimiento, 2004.

`http://apc.reuna.cl/rml.shtml?http://apc.reuna.cl?AA_SL_Session=297251ddea54a21666226de3e9eaf3a6&x=3783`.

XP:SW

Extreme Programing.

Sitio web.

`http://extremeprogramming.org/` `http://en.wikipedia.org/wiki/Extreme_programming`.

LRP:SW

Linux Registry Project.

Sitio web.

`http://registry.sourceforge.net`.

Ndiswp:SW

Ndiswrapper project.

Sitio web.

`http://ndiswrapper.sourceforge.net`.

LMP:SW

The Linux Mirror Project.

Sitio web.

`http://www.tlm-project.org/`.

USDP:SW

Unix SmartCard Driver Project.

Sitio web.

`http://smartcard.sourceforge.net`.

WIP:TI

Washington Internet Project.

The internet.

`http://www.cybertelecom.org/internet.htm`.

Proposicion:LC

Proposición.

Lista de correo para discutir y proponer el uso de tecnologías libres en el estado.

`http://www.proposicion.org.ar`

Documentos Gubernamentales: `http://proposicion.org.ar/doc/gob/index.html.es`

Referencias legislativas: `http://proposicion.org.ar/doc/referencias/`

Documento Varios `http://proposicion.org.ar/doc/rel/`

Noticias: `http://proposicion.org.ar/doc/noticias-web.html`.

FreeBsd:FPH
: The FreeBSD Documentation Project.
Freebsd porter's handbook, 2000.

`http://www.freebsd.org/doc/en_US.ISO8859-1/books/porters-handbook/index.html`.

Prigogine:OOC-84
: Illia Prigogine y E. Stengers.
Order out of chaos.
New York: Bantam Books, 1984.

CMSI:DSC-03
: Adoptada por unanimidad en Plenaria por la sociedad civil de la CMSI.
Declaración de la sociedad civil en la cumbre mundial sobre la sociedad de la información, Diciembre 2003.

`http://www.caminandoutopias.org.ar/cumbre/declaracion.php`.

PXELinux:SW
: PXELinux.
Sitio web.

`http://syslinux.zytor.com/pxe.php`.

Python:SW
: Python.
Sitio web.

`http://www.python.org`.

LinuxQuestions:LRC
: Linux Questions.
Library related commands and files.

`http://wiki.linuxquestions.org/wiki/Library-related_Commands_and_Files`.

Quiros:NTA

Bharley Quirós.
La neutralidad tecnológica no debe asociarse a la indecisión. jerarcas respaldan equidad en software.

`http://www.conicit.gov.cr/boletin/boletin28/conatic.shtml`.

Rasch:BHF-00
Cris Rasch.
A brief history of free/open source software movement, 2000.

`http://www.openknowledge.org/writing/open-source/scb/brief-open-source-history.html`.

Ratzinger:PNM-05
Joseph Ratzinger.
El papa no es un monarca absoluto, 2005.

`http://www.clarin.com/diario/2005/05/08/elmundo/i-02801.htm`.

Ratzinger:LIV-05
Joseph Ratzinger.
La iglesia que viene: ``el fundamentalismo islámico", 2005.

`http://salta.batcave.net/temp82.htm`.

Raymond:CSH
Eric Raymond.
Como ser un hacker.

`http://www.sindominio.net/biblioweb/telematica/hacker-como.html`.

Raymond:MC
Eric Raymond.
El caldero mágico.

`http://www.alanta.info/MagicCauldron.html`.

Raymond:DH
Eric S. Raymond.
Diccionario hacker.

`http://catb.org/esr/jargon/`.

Raymond:CB
Eric S. Raymond.
La catedral y el bazar.

`http://www.sindominio.net/biblioweb/telematica/catedral.html`.

Raymond:SRP
: Eric Steven Raymond.
Software release practice howto, 2000.

`http://en.tldp.org/HOWTO/Software-Release-Practice-HOWTO/`.

Raymond:AUP-03
: Eric Raymond.
The art of unix programing, 2003.
ISBN 0131429019

`http://www.faqs.org/docs/artu/`.

Raymond:HD-04
: Eric Raymond.
The halloween documents, 2004.

`http://www.opensource.org/halloween/`

`http://www.opensource.org/halloween/faq.html`

`http://www.catb.org/~esr/not-the-osi/halloween-rant.html`

`http://it.wikipedia.org/wiki/Halloween_Documents`.

Romero:OPT-05
: Cristiano Romero y Juliano Basile.
Órgão público terá de usar o software livre, 2005.

`http://clipping.planejamento.gov.br/Noticias.asp?NOTCod=189996`

`http://www.softwarelivre.org/news/3882`.

rdesktop:SW
: rdesktop.
Sitio web.

`http://sourceforge.net/projects/rdesktop/`.

RN:PSL-03
: Argentina Rectores del Norte Grande.
Propuesta generada en la reunión de catamarca sobre software libre.

`http://bo.unsa.edu.ar/sct/informes.previo/unsainfo/siu.html`.

RedHat:SW
RedHat.
Sitio web.
`http://www.redhat.com`
Red Hat Network Architecture
`http://www.redhat.com/software/rhn/architecture/`
RHN
`http://www.redhat.com/software/rhn`.

Rehermann:NMP
C. Rehermann.
No me protejas.
`http://www.henciclopedia.org.uy/autores/Rehermann/ProtecAutores.htm`.

Robles:PES-02
Gregorio Robles y Jorge Ferrer.
Programación extrema y software libre, 2002.
`http://es.tldp.org/Presentaciones/200211hispalinux/gregorio2/progm-ext-soft-libre-html/`.

Richards:MWO-05
David Richards.
Microsoft windows officially broken, 2005.
`http://digg.com/software/Windows_Is_Offically_Broken_-_Microsoft_has_admitted_it`
`http://www.schoolforge.org.uk/index.php/Microsoft_Windows_Officially_Broken`.

Rivero:IPB
Gonzalo Rivero.
Introduccion a la programacion en bash.
`http://www.softlibre.salta.org.ar/docs/descarga/2003/curso/pdfs/bash-intro.pdf`.

Robles:IS-02

Gregorio Robles.
Ingeniería del software libre, 2002.

`http://es.tldp.org/Presentaciones/200211hispalinux/gregorio/ing-soft-libre-html/`.

Rosnay:MHV-77
Joël de Rosnay.
El Macroscopio. Hacia una visión Global.
AC, 1977.
ISBN 84-7288-017-6.

rpmorg:SW
rpmorg.
Sitio web.

`http://www.rpm.org/`.

RSYNC:SW
rsync.
Sitio web.

`http://samba.anu.edu/rsync`.

SANE:SW
Scanner Access Now Easy (SANE).
Sitio web.

`http://www.sane-project.org`.

Sagan:C-85
Carl Sagan.
Contacto.
Emece, 1985.

ISBN 950-04-1781-2.

Sair:SLC
Sair.
Sair linux certification.

`http://www.linuxcertification.org/roadmap.htm`

`http://www.linuxcertification.org/`.

SAMBA:SW

SAMBA.
Sitio web.

`http://samba.org`

`http://samba.org/samba/docs/Samba-HOWTO-Collection.pdf/`,

`http://us5.samba.org/samba/`.

Saravia:CH
Diego Saravia.
Congreso hispalinux: Sin software libre imposible soñar democracia electrónica.

`http://www.noticiasdot.com/publicaciones/2003/0903/2409/noticias240903/noticias240903-6.htm`.

Saravia:NT2
Diego Saravia.
Norma técnica 2. política y normas de uso, renovación y adquisición de bienes y servicios informáticos, y derechos de uso de software.

`http://softwarelibre.unsa.edu.ar/res/propNT2.html`.

Saravia:PET
Diego Saravia.
Plan estratégico de las tecnologías de la información y comunicaciones (tic), en la unsa.

`http://bo.unsa.edu.ar/coorinfo/`.

Saravia:DDS-03
Diego Saravia.
Democracia y dictadura en la sociedad de la información.
In *Information Technology for all*. Dip. Bizkaia; UE; ONU. Bilbao., 2 2003.

`http://weblog.educ.ar/sociedad-informacion/archives/000672.php`
`http://bo.unsa.edu.ar/docacad/softwarelibre/articulos/defasoco/`.

Saravia:EI-03
Diego Saravia.
Economía de las ideas/economía del software.
Hipatia, Agosto 2003.

`http://bo.unsa.edu.ar/docacad/softwarelibre/articulos/ecosoft/`

`http://www.apesol.org/news/52`

Artículo en Brecha: `http://www.brecha.com.uy/hnnoticiaj1.cgi?1758,53,0,0,`.

Saravia:RA-03
: Diego Saravia.
Recomendaciones de acción, parte del informe ica, Octubre 2003.
parte de Saravia:[Saravia:SLA-03].

`http://www.hipatia.info/docs/ica/acciones.html`.

Saravia:CDE-04
: Diego Saravia.
Criterios de desarrollo del ekeko, 2004.

Saravia:GI-04
: Diego Saravia.
Gobiernos e Internet.
Hipatia, 2004.

`http://bo.unsa.edu.ar/docacad/softwarelibre/articulos/internetg/`.

Saravia:GIV-04
: Diego Saravia.
Governo de internet, versión inicial, 2004.

Castellano: `http://mailman.greennet.org.uk/public/lac/2004-September/002575.html`

Portugués: `http://mailman.greennet.org.uk/public/lac/2004-September/002629.html` o

`http://www.softwarelivre.org/articles/59`

Discusión: `http://lac.hipatia.info/index.php/Documentos`.

Saravia:REC-04
: Diego Saravia.
Sobre la riqueza, la escasez y el capital. Hacia una ciencia contable de la vida, la evolución y sus estructuras disipativas. Más allá de la economía.
Hipatia, 2004.

`http://bo.unsa.edu.ar/docacad/softwarelibre/articulos/economia/`.

Saravia:GAS-05

Diego Saravia.
Guía de adquisición de software para gobiernos y grandes organizaciones.
Hipatia, Marzo 2005.

`http://www.hipatia.info/docs/comprasg/`

`http://bo.unsa.edu.ar/docacad/softwarelibre/articulos/comprasg/`.

Saravia:NT-05
Diego Saravia.
Neutralidad tecnológica.
Hipatia, 2005.

`http://bo.unsa.edu.ar/docacad/softwarelibre/articulos/neutro/`.

Saravia:OLC-05
Diego Saravia.
Ontología de la libertad del conocimiento.
Hipatia, 2005.

`http://bo.unsa.edu.ar/docacad/softwarelibre/articulos/ontologia/`.

Saravia:PHC-05
Diego Saravia.
Programar es una habilidad cultural fundamental, 2005.

`http://lists.ourproject.org/pipermail/solar-general/2005-June/019672.html`.

Saravia:QC-05
Diego Saravia.
¿Que es el conocimiento?
Hipatia, 2005.

`http://bo.unsa.edu.ar/docacad/softwarelibre/articulos/conocimiento/`.

Schopenhauer:MRV-18
Arthur Schopenhauer.
El mundo como representación y voluntad.
CAYFOSA-QUEBECOR, 1818.
ISBN 84-473-2848-1.

Schopenhauer:SLV-39

Arthur Schopenhauer.
Sobre la Libertad de la Voluntad.
Alianza, 1839.
ISBN 84-206-3922.

Schumpeter:CSD
Schumpeter.
Capitalismo, Socialismo y Democracia.
Harper & Brothers, 1942.

Saravia:SLA-03
Diego Saravia y Comunidad de Hipatia.
Software libre en la administración pública: Desafíos y oportunidades, 2003.

`http://bo.unsa.edu.ar/docacad/softwarelibre/articulos/ica`

`http://www.hipatia.info/docs/dsl`.

Seebach:AWL-04
Peter Seebach.
The art of writing linux utilities, 2004.

`http://www-128.ibm.com/developerworks/linux/library/l-util.html`.

Saravia:MH-01
Diego Saravia, Juan Carlos Gentile, Gonzales, y Mario Tessa.
Manifiesto de Hipatia, 2001.

`http://www.hipatia.info/index.php?id=manifesto_es`.

FreeStandars:SW
Free Standards Group y Wikipedia.
Sitio web.

`http://www.freestandards.info/`

`http://freestandards.org/docs/FSG_ISV_WP_public.pdf`

`http://en.wikipedia.org/wiki/Free_Standards_Group`.

Sindominio:SW
Sindominio.
Sitio web.

`http://copyleft.sindominio.net/`.

Slackware:SW
Slackware.
Sitio web.
`http://www.slackware.com/`.

Samuelson:E-99
Samuelson y Nordhaus.
Economía.
Mc Graw Hill, 16 edition, 1999.
ISBN 84-481-2314X.

Solar:SW
SOLAR.
Sitio web.
`http://www.solar.org.ar`.

Solar:SLP
SOLAR.
Software libre desde la política y religión.
`http://ourproject.org/cgi-bin/moin.cgi/Software_20Libre_20y_20Pol_edtica`.

OpenSource:M
Open Source.
Marketing.
`http://www.opensource.org/advocacy/case_for_hackers.php#marketing`.

OpenSource:WFS
Open Source.
Why ``free'' software is too ambiguous.
`http://www.opensource.org/advocacy/free-notfree.php`.

Saltzer:ETE-84
J. H. Saltzer, D. P. Reed, y D. D. Clark.
End-to-end arguments un system design.
ACM Transactions in Computer Systems, (2):277-288, Noviembre 1984.
`http://www.reed.com/Papers/EndtoEnd.html`.

Stallman:APC

Richard Stallman.
Algunas palabras y frases confusas que vale la pena evitar como piratería y propiedad intelectual.

`http://www.gnu.org/philosophy/words-to-avoid.es.html`.

Stallman:CIP
Richard Stallman.
Copyleft: Idealismo pragmático.

`http://www.gnu.org/philosophy/pragmatic.es.html`.

Stallman:DPI
Richard Stallman.
¿dijiste ``propiedad intelectual"?, Es sólo un espejismo seductor.

Traducido por Martín Olivera

`http://www.solar.org.ar/article.php3?id_article=207`.

Stallman:FDL
Richard M. Stallman.
Fdl, licencia para documentos libres.

`http://www.fsf.org/licenses/fdl.html`.

Stallman:GPL
Richard M. Stallman.
Gpl, licencia para programas libres.

`http://www.fsf.org/licenses/gpl.html`.

Stallman:WFS
Richard M. Stallman.
Why ``free software" is better than ``open source".

`http://www.gnu.org/philosophy/free-software-for-freedom.html`.

Stallman:NKC-2000
Richard Stallman.
New kinds of copyright.
Copyright versus community in the age of computer networks, 2000.

`http://www.carnall.demon.co.uk/stallman/kinds.html`.

Stallman:PSO-02

Richard M. Stallman.
Patentes de software, un obstáculo para el desarrollo de software, 2002.

`http://www.cl.cam.ac.uk/~mgk25/stallman-patents.html`.

Inquirrer:ITW-03
Inquirrer Staff.
International treaty will force 34 democracies to change copyright, ip laws: Alca, 2003.

`http://www.theinquirer.net/?article=12219`.

Stern:NAN-91
Hal Stern.
Nfs and nis, 1991.
ISBN 0-937175-75-7.

Stiglitz:KGP
Joseph E Stiglitz.
Knowledge as a global public good.

`http://www.worldbank.org/knowledge/chiefecon/articles/undpk2/`.

Stonebank:UTB-01
M Stonebank.
Unix tutorial for beginners, 2001.

`http://www.ee.surrey.ac.uk/Teaching/Unix/`.

SuperKaramba:SW
SuperKaramba.
Sitio web.

`http://netdragon.sourceforge.net/`.

Suse:SW
Suse.
Sitio web.

`http://www.suse.com`.

Saravia:NT3
Diego Saravia y Victor Omar Viera.
Norma técnica 3.

Saravia:NT4

Diego Saravia y Victor Omar Viera.
Norma técnica 4.

Saravia:RRP-01
Diego Saravia y Victor Omar Viera.
Reglamento de registro de programas de uso restringido de la universidad nacional de salta, unsa, Agosto 2001.

`http://bo.unsa.edu.ar/dr/R2001/R-DR-2001-0326.html`.

Saravia:RR-03
Diego Saravia y Victor Omar Viera.
Creación del sistema de normas técnicas y ampliatoria, resolución r-dr-2003-0186, unsa, Mayo 2003.

`http://bo.unsa.edu.ar/dr/R2003/R-DR-2003-0186.html`

`http://bo.unsa.edu.ar/dr/R2003/R-DR-2003-0194.html`.

Saravia:NT1-03
Diego Saravia y Victor Omar Viera.
Norma técnica 1. redes informáticas, 2003.

`http://softwarelibre.unsa.edu.ar/res/R-DR-2003-0412.html`.

SVG:SW
W3 standard SVG.
Sitio web.

`http://www.w3.org/Graphics/SVG/`.

Searls:WOE-03
Doc Searls y David Weinberger.
World of ends. what the internet is and how to stop mistaking it for something else.

`http://www.worldofends.com/`.

Sweet:SDU-99
Michael Sweet.
Software distribution using the esp package manager, 1999.

`http://www.easysw.com/epm/documentation.php`.

CUPS:SW
Common Unix Printing System.
Sitio web.

`http://www.cups.org`.

Tamara:AAL-03
: Vladimir Támara, Jaime Irving Dávila, Pablo Chamorro, y Igor Támara.
Aprendiendo a aprender linux, 2003.

`http://structio.sourceforge.net/guias/AA_Linux_colegio/AA_Linux_colegio.html`.

KDEDocTeam:KUG-04
: The KDE Documentation Team.
The kde user guide, 2004.

Ts:US-03
: Jay Ts, Robert Eckstein, y David Collier-Brown.
Using samba, 2003.
ISBN 0-596-00256-4.

Mozilla:SW
: Mozilla Thunderbird.
Sitio web.

`http://www.mozilla.org`.

TIM
: The internet manifesto.

`http://www.internetmanifesto.org/internetmanifesto/manifesto.txt`.

NYT:A-99
: New York Times.
Archivos sobre ``internet governance", 1999.

`http://www.nytimes.com/library/tech/reference/index-domain.html`.

Wassenaar:SW
: Tratado sobre control de exportación de armas de Wasenaar.
Sitio web.

`http://www.wassenaar.org/`.

Turner:GUS-04
: Jonathan Turner.
Gnome 2.6 usability study and review, Julio 2004.
`http://www.userinstinct.com/viewpost.php?postid=gnome26review`.

Ubuntu:SW
Ubuntu.
Sitio web.
`http://www.ubuntulinux.org/`.

Uekawa:DLP:02
Junichi Uekawa.
Debian library packaging guide, 2002.
`http://www.netfort.gr.jp/~dancer/column/libpkg-guide/libpkg-guide.html`.

UNESCO:SC-02
UNESCO.
La sociedad del conocimiento.
Revista Internacional de Ciencias Sociales, (171), Marzo 2002.
`http://www.unesco.org/issj/rics171/fulltext171spa.pdf`.

UniConf:SW
Configuration System UniConf.
Sitio web.
`http://open.nit.ca/wiki/`.

UOC:ML
UOC.
Materiales libres.
`http://www.uoc.edu/masters/softwarelibre/esp/materiales.html`.

Ututo:SW
Ututo.
Sitio web.
`http://www.ututo.org/`.

Vairagade:MPS-03
Mugdha Vairagade.
Manage packages using stow, stow offers a flexible, capable alternative to rpm, 2003.
`http://www-128.ibm.com/developerworks/linux/library/l-stow/`.

Varios:CSC

Varios.
Copyleft como subversion del copyright.

`http://www.onnirik.org/onnirikarchives3/art_copyleft_es.htm`

`http://www.subvertise.org/search.php?q=copyleft`

`http://javarm.blogalia.com/historias/1239`.

Varios:TWS
Varios.
Theme web sites.

`http://themes.freshmeat.net`

`http://www.customize.org`

`http://www.kde-look.org`

`http://art.gnome.org`

`http://www.crystalgnome.org`.

Venezuela:LA
Venezuela.
Libro amarillo.

`http://www.mct.gov.ve/uploads/biblio/amarillo2.pdf`.

MigrandoVenezuela:SW
Migrando Venezuela.
Sitio web.

`http://www.migrandovenezuela.org`.

Vernooij:SDG-03
Jelmer R. Vernooij.
Samba developers guide, 2003.

Vaughan:GAA-00
Gary Vaughan, Ben Elliston, Tom Tromey, y Ian Taylor.
Gnu autoconf, automake and libtool, 2000.
ISBN 1-57870-190-2

`http://linux.duke.edu/~mstenner/free-docs/autobook-1.3/autobook.html`

`http://sources.redhat.com/autobook/`.

Viera:RR-03
: Victor Omar Viera y Hector Alfredo Flores.
Creación auditoría informática. resolución r-dr-2003-0673, unsa, Noviembre 2003.

`http://softwarelibre.unsa.edu.ar/res/R-DR-2003-0673.html`.

Villoslada:ELM-04
: Benjamín Villoslada.
La estrategia linux de microsoft, 2004.

`http://bulma.net/body.phtml?nIdNoticia=2102`.

Vi:PI-98
: Viñameta.
Propiedad intelectual, 1998.
ISBN 968-24-0854-7.

Wagensberg:P-98
: J. Wagensberg, Jordi Agusti, Alberch, Goodwin, Hull, et al.
El Progreso, un concepto acabado o emergente.
Tusquets, 1998.
ISBN 84-8310-569-1.

Waal:SAS-01
: Frans de Waal.
El simio y el aprendiz de sushi. Reflexiones de un primatólogo sobre la cultura.
Paidos, 2001.
ISBN 84-493-1325-2.

Wall:PP-91
: Larry Wall, Tom Christiansen, y Randal Schwartz.
Programming perl, 1991.
ISBN 1-56592-149-6 .

IRM:SW
: Sitio Web.
The information resource manager.

`http://www.stackworks.net/view.php/irm/index.html`.

Linuxcompatible:SW
: Sitio Web.
Linuxcompatible.org.

`http://linuxcompatible.org/`.

Webmin:SW
: Webmin.
Sitio web.

`http://www.sourceforge.net/projects/webadmin`

`http://www.webmin.com`.

Weber:EPE
: Max Weber.
La ética protestante y el espíritu del capitalismo.
Ed. Península/Biblos, 17 edition, 1904.
ISBN 84-8307-025-1

`http://usuarios.lycos.es/politicasnet/autores/weber.htm`.

Welton:C-00
: N. Welton, David.
Comentario sobre apt-get, 2000.

`http://lists.debian.org/debian-devel/2000/debian-devel-200012/msg00969.html`.

Westermann:CA-04
: Westermann Werner.
Curso de alfabetizacion digital, 2004.

`http://ciberania.dnsalias.org/moodle14/course/view.php?id=4`.

Worth:SAT
: D.J. Worth y C. Greenough.
A survey of available tools for developing quality software using fortran 95, 2005.

`http://www.sesp.cse.clrc.ac.uk/Publications/tools_report/`.

Wheeler:CSL-01
: David A. Wheeler.
Counting source lines of code (sloc), 2001.

`http://www.dwheeler.com/sloc`.

Wheeler:OSR-04
: David A. Wheeler.
Open source software / free software (oss/fs) references, 2004.

`http://www.dwheeler.com/oss_fs_refs.html`.

Wheeler:WOS-04
: David A. Wheeler.
Why open source software / free software (oss/fs, floss, or foss)? look at the numbers!, 2004.

`http://www.dwheeler.com/oss_fs_why.html`.

Wikipedia:CG
: Wikipedia.
Common good.

`http://en.wikipedia.org/wiki/Common_good`.

Wikipedia:CLD
: Wikipedia.
Comparison of linux distributions.

`http://en.wikipedia.org/wiki/Comparison_of_Linux_distributions`.

Wikipedia:E
: Wikipedia.
Economics.

`http://en.wikipedia.org/wiki/Economics`.

Wikipedia:Ep
: Wikipedia.
Epistemology.

`http://en.wikipedia.org/wiki/Epistemology`,

`https://es.wikipedia.org/wiki/Epistemología`.

Wikipedia:FG
: Wikipedia.
Free good.

`http://en.wikipedia.org/wiki/Free_good`.

Wikipedia:GE
: Wikipedia.
Good (economics).

`http://en.wikipedia.org/wiki/Good_%28economics%29`.

Wikipedia:IP

Wikipedia.
Internet.

`http://en.wikipedia.org/wiki/Internet`.

Wikipedia:TC
Wikipedia.
La tragedia de los commons.

`http://en.wikipedia.org/wiki/Tragedy_of_the_commons`.

Wikipedia:LD
Wikipedia.
Linux distribution.

`http://en.wikipedia.org/wiki/Linux_distribution`.

Wikipedia:O
Wikipedia.
Ontology.

`http://en.wikipedia.org/wiki/Ontology`.

Wikipedia:OCS
Wikipedia.
Ontology in computer science.

`http://en.wikipedia.org/wiki/Ontology_(computer_science)`.

Wikipedia:POS
Wikipedia.
Philosophy of science.

`http://en.wikipedia.org/wiki/Philosophy_of_science`.

Wikipedia:Ptg
Wikipedia.
Portage.

`http://en.wikipedia.org/wiki/Portage_(software)`.

Wikipedia:P
Wikipedia.
Propiedad.

`http://en.wikipedia.org/wiki/Ownership`

`http://www.webster-dictionary.org/definition/property+right`.

Wikipedia:PG
: Wikipedia.
Public good.

`http://en.wikipedia.org/wiki/Public_good`.

Wikipedia:SN
: Wikipedia.
Semantic network.

`http://en.wikipedia.org/wiki/Semantic_network`.

Wikipedia:SW
: Wikipedia.
Sitio web.

`http://en.wikipedia.org`.

Wikipedia:SLSD
: Wikipedia.
Sls distribution.

`http://en.wikipedia.org/wiki/Softlanding_Linux_System`.

Wilson:SNS-75
: Edward O. Wilson.
Sociobiology: The new synthesis.
Cambridge, MA: Belknap Press, 1975.

`http://www.ship.edu/~cgboeree/sociobiology.html`.

Wilson:HN-78
: Edward O. Wilson.
On Human Nature.
Hardvard University Press, 1978.
ISBN 0-674-63442-X .

Wilkins:GSL-95
: David R. Wilkins.
Getting started with 1995.

`http://www.maths.tcd.ie/~dwilkins/LaTeXPrimer/`.

Wilson:B

Edward O. Wilson.
The bottleneck.
Scientific American, Febrero 2002.

`http://www.sciam.com/article.cfm?articleID=000E5878-3E45-1CC6-B4A8809EC588EEDF`.

Winmodems:SW
Winmodems.
Sitio web.

`http://linmodems.org`.

WIPO:SW
WIPO.
Sitio web.

Info: `http://www.wipo.int/about-wipo/en/gib.htm`

Convention: `http://www.wipo.int/treaties/en/convention/trtdocs_wo029.html`

Plan: `http://www.wipo.int/about-wipo/en/dgo/pub487.htm`.

Wong:FOS-03
Kenneth Wong.
Free open source software and governments: A survey of foss iniciatives in governments, Agosto 2003.

`http://opensource.mimos.my/fosscon2003cd/paper/full_paper/kenneth_wong.pdf`.

wxWindows:SW
wxWindows.
Sitio web.

wxWindows provides an open source C++ GUI framework for cross-platform programming -

`http://www.wxwindows.org`

Looking through wxWindows: An introduction to the portable C++ and Python GUI toolkit

`http://www-106.ibm.com/developerworks/library/l-wxwin.html`.

Yamato:AP

Masatake Yamato.
Autopack.

`http://www.gyve.org/~jet/autopack/`

`http://lists.gnu.org/archive/html/automake/2001-11/msg00004.html`

`http://lists.debian.org/debian-devel/2005/03/msg00433.html`

`http://autoconf-archive.cryp.to/ax_dist_rpm.html`.

FHS:SW
Filesystem Hierarchy Standard y Wikipedia.
Sitio web.

`http://en.wikipedia.org/wiki/Filesystem_Hierarchy_Standard`

`http://www.pathname.com/fhs/`.

Zapata:IGL-02
Marcos Zapata.
Infoz, gnu/linux para inventariar pcs, Noviembre 2002.

`http://g.unsa.edu.ar/softlibre/infoz/index.html`.

Zittrain:BCA-00
Jonathan Zittrain.
Balancing control and anarchy on the internet, Octubre 2000.

`http://chronicle.com/free/v47/i07/07b02001.htm`.

Notas

1.1

Hoy una gran parte del capital productivo del planeta es privado (corporaciones habilitadas por el estado, personas jurídicas), junto con otra buena proporción pública en manos de los estados.

1.2

Comunicación Personal Eduardo Letelier.

1.3

Término a evitar [Stallman:APC]

1.4

El código genético es software.

1.5

Libre significa el derecho de distribuir copias, sea con o sin modificaciones, en forma gratuita o cobrando un monto por la distribución, a cualquiera y en cualquier lugar y momento.

Software Libre significa, entre otras cosas, que no hay que pedir o pagar permisos.

Software Libre significa la libertad de hacer modificaciones y utilizarlas de manera privada en trabajo u ocio, sin siquiera tener que anunciar que dichas modificaciones fueron realizadas. Cuando se publican y difunden los cambios no hace falta avisar a nadie en particular ni de ninguna manera en particular.

Software Libre no significa la obligación de distribuir el software o sus modificaciones. Es un derecho. No una obligación, menos de distribución universal. Incluso los derecho-habientes pueden distribuir a unos en forma libre y a otros en forma privativa. En el caso del ``software libre-copyleft'' los licenciatarios sólo pueden redistribuir en forma libre, o con los mismos derechos que recibieron ``share-like'', en términos de las CC [CC:SW]. Los autores no tienen los mismos derechos que los usuarios.

1.6

Otra cuestión en que la industria editorial está siendo reemplazada es la de selección de material. En un universo de escasez, era importante decidir qué se imprimía o distribuía. Hoy cualquiera publica y ``Google'' en un sentido y ``Wikipedia'' en otro permiten la búsqueda y calificación de los trabajos.

1.7

Que no es más que un burdo reemplazo de las herramientas de evaluación de proyectos de inversiones basadas en el retorno de la inversión.

1.8

Microsoft ha liderado la tendencia de crear herramientas de programación cada vez más complejas, caras y sofisticadas, alejadas de los sistemas simples y aptos para niños, jóvenes o hobistas que iniciaron la popularización de la computación.

1.9

Un concepto proveniente de las agencias de inteligencia por el cual algo fiable es algo de lo que necesitas cuidarte.

1.10

Entre otras cosas se ha llegado a fomentar la denuncia entre pares en diferentes ámbitos, recordando así las peores prácticas de los regímenes fascistas.

2.1

Se utiliza el término Ontología en el sentido dado en la Wikipedia: [Wikipedia:O].

2.2

El conocimiento es información con sentido [Saravia:QC-05]

2.3

La curva de aprendizaje que le espera al novato en el campo del Software Libre en cuanto a su filosofía es apabullante.

2.4

Es interesante destacar la Constitución Bolivariana (Venezuela) que sindica a la cultura como libre.

2.5

Recuérdese diferenciar la creación de una obra intelectual de la producción de sus copias en forma masiva.

2.6

La FSF aplica estos razonamientos a sólo una clase de obra intelectual, la denominada ``funcional". Así se opone a la restricción de circulación de sus expresiones mediante licencias limitantes basadas en el copyright. También se opone al patentamiento de estas ideas funcionales - o software- supuestamente asimilables a inventos o métodos de negocio.

2.7

El tiempo requerido para el aprendizaje o conversión de información en conocimiento o asimilación, está fuera del mercado, y no influye necesariamente en la cuestión del valor económico.

2.8

El mismo Boyle lo indica, los ``commons" no son bienes libres. ``On the other hand, however, it fit very well into a new literature on governing the commons from Elinor Ostrom, Robert Keohane, Margaret McKean, and many others. This literature was able to show that not every commons was a tragedy. But the literature also showed that successful commons were not entirely free - they ran on layers of norms that were frequently invisible to the legal system, but which nevertheless served to avoid the various paradoxes of collective action. Whether the examples

were Japanese herdsmen or Silicon Valley programmers, the literature seeks to show just how the commons was, and should be, governed" [Boyle:SEM-03].

2.9

También debe notarse que RMS habla solo de software libre (y de obras intelectuales funcionales en general) y no de otras categorías de conocimiento, información, ideas o sus expresiones [Barton:HSS]. Sólo para las obras funcionales él defiende la libertad del conocimiento. RMS no expresa una posición basada en los efectos económicos sino se centra exclusivamente en la cuestión de la libertad de expresión.

2.10

No necesariamente todos los sistemas vinculados con los ``derechos intelectuales" son iguales, por ejemplo no parece tener sentido pensar en socializar las marcas, pero podría tener sentido con los diseños industriales.

2.11

Según el profesor Mark Lemley, actualmente en la Escuela de Leyes de Stanford, el extendido uso del término ``propiedad intelectual" es un capricho originado en la creación en 1967 de la Organización Mundial de la ``Propiedad Intelectual" (World ``Intellectual Property" Organization), y sólo se hizo más común en estos últimos años. (OMPI (WIPO) es formalmente una organización de la ONU, pero en realidad representa los intereses de los derechohabientes de derechos de autor, patentes y marcas registradas.)

Quienes prefieren analizar estos temas según sus méritos deberían rechazar un término desviado para nombrarlos. Muchos me han pedido que proponga algún otro nombre para esta categoría - o proponen alternativas ellos mismos. Las sugerencias incluyen PMIs, de Privilegios Monopólicos Impuestos, y MOGSL, de Monopolios Originados por el Gobierno Sostenidos Legalmente (en inglés GOLEMs, Government-Originated Legally Enforced Monopolies). Algunos hablan de ``regímenes de derechos exclusivos", pero esto parece referirse a las restricciones como derechos, lo que también genera malentendidos.

Pero es un error reemplazar ``propiedad intelectual" con cualquier otro término. Un nombre diferente podría eliminar la desviación, pero no soluciona el problema más profundo del término: la sobregeneralización. No hay tal cosa unificada llamada ``propiedad intelectual". Es un espejismo, que parece tener existencia coherente sólo porque el término lo sugiere. RMS [Stallman:DPI]

2.12

Se dice que toda ética se basa en la realidad, o que los ideales subliman la realidad

2.13

Si se inventan dispositivos agradables de lectura digital, como papel auto imprimible

2.14

En febrero de 1998, Netscape anunció el lanzamiento de su navegador como Software Libre. Un grupo de personas con base en Palo Alto y Sillicon Valley, propuso empezar una campaña a favor

del Software Libre mediante el término de Fuente Abierta. El objetivo fue lograr la aceptación de compañías y capitalistas en el boom de la nueva economía. Se eligió dejar de lado las cuestiones de largo alcance: económicas, éticas y sociales, pensando que estas obstaculizaban la rápida aceptación por parte de las empresas. El foco estaba en las ventajas técnicas y el modelo de desarrollo.

2.15

¿Es la libertad hacer todo lo que se nos ocurra?. ¿Termina la libertad donde empieza la del otro?

2.16

Victoria absoluta del movimiento.

2.17

Algunos argumentan que la libertad debe darse solamente para el software o en general para todo conocimiento funcional, pero que con la música, literatura fantástica u otras obras intelectuales dedicadas al placer no es necesario. Debe tenerse en cuenta que la libertad en discusión es para quien ``usa" la obra (libertad, no obligación). La cuestión central entonces es, el que conoce algo, ¿tiene la libertad de difundirlo a otros? ¿El software es diferente que una canción, en esto? ¿El placer es menos serio que el software?

2.18

Los manuales son parte del software y deben licenciarse igual que éste.

2.19

La cuestión es relativa al licenciamiento en el marco de las leyes de copyright vigentes, en general es una cuestión de libertades y derechos.

2.20

No siempre es el autor el derecho-habiente de un software, por ejemplo cuando existe relación de dependencia el copyright es del empleador, no del autor.

2.21

Hay modelos de desarrollo como el Bazar que sólo son posibles con Software Libre.

2.22

Sueño del modelo de Banco Mundial de contabilidad de costos para todo

2.23

Si no hay copyright el software -incluso el libre- podría ser binarizado y se podrían montar empresas que distribuyan binarios. Ahora estos binarios no tendrían protección legal, que impida su copia. Si podrían depender de un esquema tipo TCG/DRM/llaves físicas, pero sin copyright tal sistema difícilmente podría crearse y sostenerse.

3.1

no es muy relevante separar a la comunidad en creadores y usuarios, pues todos en cierta forma participan y aportan.

3.2

No se puede hablar de Software Libre sin un amplio y libre acceso a Internet por parte de los desarrolladores, y usuarios, sin restricciones de ningún tipo.

3.3

Existen muchas discusiones sobre si estos softwares fueron creados utilizando el método bazar o un método con alguna de sus características y hasta qué punto

3.4

Sourceforge tiene 35% más programadores involucrados que Microsoft [Boulton:FFS-05].

3.5

Si algún proyecto se basa en código libre pero incompatible con la GPL, esto puede no ser posible y se deben buscar alternativas.

3.6

Ver software como Blackduck y Palamida.

3.7

GNU recomienda usar C por defecto [GNU:GCS-04].

3.8

Él decía ``organigramas" y ``tablas" [Raymond:CB].

4.1

Por decreto.

4.2

En cada caso primero habrá que relevar que no existan proyectos previos y en tal caso cuál es su grado de avance y velocidad de crecimiento.

6.1

Software Libre en los Estados, ICA: [Saravia:SLA-03].

Plan Estratégico TIC UNSa: [Saravia:PET].

Especialistas: [Versora:SW,Alaco:SW].

Empresas y guías de migración, soporte, hardware:
[Novell:NM,IBM:LCM,IBM:LIS,IBM:PMS,IBM:PCS]. países:

Brasil: [Brasil:GM,Brasil:PPI-03],

Venezuela: [Venezuela:DSL-04,MigrandoVenezuela:SW,Venezuela:LA],

Europa: [IDA:MM],

Alemania: [Alemania:MM,Bass:UCB].

Ahorros de la migración: [Benner:MWL-04,Clarke:IGF-05].

6.2

El texto está basado en el trabajo [Saravia:RA-03] al cual eventualmente reemplazará. Su versión original fue publicada como wiki en Hipatia y en Sapi.

6.3

Puede definirse una etapa previa para sensibilizar a las autoridades para que tomen la decisión política. Algunos podrán ver esto como primer mantra. O puede realizarse una acción de difusión entre responsables de diferentes niveles y que cada uno comience en su área de autoridad a migrar.

6.4

Pueden ser útiles: INFOZ [Zapata:IGL-02] y también linuxcompatible.org [Linuxcompatible:SW], CACIC [Mello:SIH] y [IBM:IRU].

8.1

Que no es más que un burdo reemplazo de las herramientas de evaluación de proyectos de inversiones basadas en el retorno de la inversión.

8.2

La cuestión es relativa al licenciamiento en el marco de las leyes de copyright vigentes, en general es una cuestión de libertades y derechos.

8.3

No siempre es el autor el derecho-habiente de un software, por ejemplo cuando existe relación de dependencia el copyright es del empleador, no del autor.

8.4

Hay modelos de desarrollo como el Bazar que sólo son posibles con Software Libre.

10.1

Victoria absoluta del movimiento.

10.2

No es posible deducir el software libre de la nada, sino que su fundamento parte de los intereses de las mayorías frente a los intereses de los derecho-habientes de unos pocos códigos privativos. En definitiva el movimiento del ``Free software", debe tomar sus fundamentos del movimiento ``Open Source". Otros actores en la comunidad adoptan otra respuesta a Microsoft: [Perens:SC]

10.3

¿Hasta dónde puede llegar el software libre? No tiene límites, excepto cuando las patentes lo prohíben. El objetivo final del movimiento es proporcionar software libre para hacer todos los trabajos que los usuarios de computadoras quieran hacer y por lo tanto hacer el software propietario obsoleto. Adaptado de [FSF:HPG]

10.4

No necesariamente todos los sistemas vinculados con los ``derechos intelectuales" son iguales, por ejemplo no parece tener sentido pensar en socializar las marcas, pero podría tener sentido con los diseños industriales.

11.1

Capitalista, oficial, occidental, neo-clásica.

11.2

Una ciencia para la cual hacemos modelos que representan la ``realidad económica".

11.3

Suele presentarse en las primeras páginas de los tratados de economía.

11.4

¿Qué papel juega el conocimiento en la economía? ¿Qué papel juega la acumulación de estructuras? ¿Cuál es el capital? ¿Se puede definir un precio en este marco, una variable común que represente el valor de cualquier bien? ¿Es posible un mundo sin escasez que no afecte negativamente nuestro medio ambiente, impulsado por la ciencia y la tecnología -``tecno-utopía"-? Más allá de la cuestión ``tecno", una utopía social de seres con iguales derechos, ¿es factible? ¿con qué límites? ¿Qué cuestiones se pueden esconder en estos planteos, y cómo afectarían la distribución del goce en el planeta? ¿Es posible una ``utopía" de los derechos? ¿Servicios o Derechos? ¿Podemos ser nosotros optimistas, o creemos que los que trabajan por la concentración y el control tienen razones para ser optimistas? ¿Cómo se vincula la economía con las jerarquías sociales en los sistemas biológicos? ¿Qué sucede con el salario ante un mundo con propiedad intelectual?, ya que el ingreso que produce el trabajo permanece constante ante el ``capital intelectual" que crece indefinidamente y sin límites. ¿Cómo se distribuyen los bienes físicos realmente escasos entre trabajo y capital en estas circunstancias?

11.5

El ALCA[Inquirrer:ITW-03] -es sólo un ejemplo- forzará a 34 democracias a cambiar sus leyes de protección a los derechos de copia. Se prohíben los backups de los DVD, se encarcelará a quienes compartan archivos en la red.

11.6

La utopía tecnológica (en su versión informática):[Mattelart:HSI-02]

> La alborada de la sociedad de la información - Desafíos y oportunidades.[Iglesias:ISE-98]

> La sociedad naciente de la información -la culminación del desarrollo de la tecnología de la información y del campo de las comunicaciones- creará una nueva revolución industrial tan significativa y tan llena de consecuencias como la que se produjo en el pasado. Los sistemas de producción, los métodos y relaciones de trabajo, la organización corporativa, los programas de capacitación y educación, los esquemas de consumo y la comunicación humana básica están sufriendo cambios dramáticos. Se trata de una revolución basada en la información, esto es, en el conocimiento, la innovación y la creatividad humana. Y esta es la revolución que nos permitirá procesar, almacenar, recuperar y comunicar la información cualquiera que sea la forma que ésta tome -oral, escrita o visual- liberada de la mayoría de las limitaciones de distancia, tiempo y volumen."

11.7

En el Apéndice: 11.4.3 se describen aspectos particulares de los flujos de bienes.

11.8

Es más simple imaginar topes rígidos, que costos de extracción crecientes. Es interesante notar que no es necesario suponer rendimientos decrecientes en la obtención de factores para pensar la

economía, basta un tope. Pensar en costos de extracción variables, también nos lleva a construir un modelo similar al presente.

11.9

Primera simplificación u ordenamiento, de suponer la necesidad de otros bienes, habría que prever una fábrica adicional por factor.

11.10

Si se sigue a Marx, se podría considerar que toda la economía se genera en función de un único factor: el trabajo humano. Aquí se generaliza el análisis para comprender otros recursos limitados -no el capital- y permitir la combinación de sus ideas con las cuestiones ambientales

11.11

En los modelos tradicionales el ``factor trabajo" se toma como un bien intermedio ``producido" por las ``familias" vistas como fábricas que utilizan los bienes de consumo como intermedios. Esto le da un carácter circular a la economía, inverso al flujo de dinero. Incluso se ``olvida" el ``consumo" de recursos naturales escasos. En este trabajo el objetivo que tiene la realidad económica, no es la acumulación sino la satisfacción de las preferencias de consumo, y contabilizar los recursos usados, por eso es más ``interesante" desacoplar el modelo circular que presupone un esquema de distribución capitalista y abrirlo.

11.12

No se estudia el impacto de los mecanismos de decisión en la realidad económica, por ejemplo si están o no concentrados.

11.13

Segunda simplificación, u ordenamiento.

11.14

Tercera simplificación, propia del ejemplo usado en éste trabajo.

11.15

Esta simplificación hace al concepto de inversión paralelo a la producción. Al suponer que no hay variaciones de lo acumulado, pensamos en tiempos medios invariantes para cada bien. Y por lo tanto, en estas condiciones, la inversión y la producción global van de la mano, y su estudio diferenciado no aporta nuevas ideas.

11.16

Cuarta simplificación, propia del ejemplo usado en éste trabajo.

11.17

Nada impediría un modelo más complejo, con variables en vez de constantes.

11.18

Quinta simplificación, es con respecto a la economía clásica. El trabajo se aleja aquí considerablemente de la teoría económica clásica, que utiliza funciones de producción que modelan realidades más complejas (tres en vez de seis),

```
B_1 = fpc(F_{o1}, F_{a1}); B_2 = fpc(F_{o2}, F_{a2}); B_3 =
fpc(F_{o3}, F_{a3})
```

permitiendo cantidades variables de cada bien. Este trabajo usa ecuaciones o relaciones conceptualmente más simples.

Con las funciones de producción clásicas se vincula lo producido con cantidades variables de factores (o bienes intermedios), es decir se puede vincular la cosecha de trigo con cantidades variables de tierra y campesinos. Por ejemplo, pueden trabajar más o menos campesinos por hectárea y variará el rendimiento.

En nuestro modelo este comportamiento tendremos que reproducirlo con muchas funciones de producción diferentes: una por cada relación distinta del número de campesinos y hectáreas. Deberemos pensar en que cada bien se puede producir de muchas maneras, ``eligiendo" en cada caso hacerlo por la que más convenga.

Una cuestión interesante es que no se piensa en los rendimientos decrecientes. En este trabajo las ``fábricas" son lineales.

11.19

Por ejemplo, si se fabrica un auto se necesitan 5 ruedas y un chasis, no es posible alterar las proporciones, sin cambiar el auto (por ejemplo, venderlo sin rueda de auxilio).

11.20

Estas ecuaciones definen una matriz de producción, similar a las utilizadas por Leontief[Leontief:AEI,Intriligator:OMT] en su análisis Input-Output.

11.21

Sexta simplificación.

11.22

Por ejemplo el costo de los automóviles será un chasis por auto más cinco ruedas por auto

11.23

No se plantean actores internos y cómo distribuyen sus capacidades de decidir.

11.24

Para una alternativa de tasas posible, la función dará un valor. Para otra, otro valor. Comparando ambos valores y eligiendo el mayor, sabremos cuál alternativa es preferida.

11.25

Esta función es conocida como Utilidad. He preferido cambiarle de nombre en este trabajo para evitar que se asocie estas conclusiones a las de la economía marginalista. Ver apéndice.

11.26

Planteamos riqueza como algo diferente al capital -ver más adelante-, es una diferencia importante en relación a la economía clásica.

11.27

Por ejemplo entre dos realidades económicas. Si los precios en ambas favorecen el intercambio.

11.28

Multiplicados por una constante arbitraria. Son ecuaciones homogéneas. Debe notarse que esta constante arbitraria puede ser la misma para todo el modelo y por ello tienen sentido comparar precios de distintas situaciones de una misma realidad económica.

11.29

La economía neo-clásica usa la ``Teoría de la productividad marginal" [CEPA:NTD,Bates:SW,Samuelson:E-99] pp225. Capítulo 12. B, para definir el valor relativo de los factores. Aquí se verá claramente que no es lo determinante.

11.30

Podemos hablar indistintamente de capital como de producción, a los efectos del razonamiento presente es la misma cosa.

11.31

Realista en su simplicidad.

11.32

En casos de economías humanas podríamos pensarla en base a cada ser humano: nadie puede consumir más allá de una cantidad del recurso. Por ejemplo, no se puede comer carne en forma sostenida más que a un ritmo de 1kg por hora por decir una exageración.

11.33

Por ejemplo, aprendiendo mejores métodos productivos.

11.34

Como muchas veces, Gandhi tenía razón: si cada uno usa solo lo que necesita no habría pobreza.

11.35

Supuestamente nuestro futuro. Se podría decir que las sociedades del conocimiento son ``software libre + internet", así como Lenin[Lenin:CTN-19] decía que El comunismo es los soviets más la electricidad, o en general: ``ideología+tecnología=nuevo mundo".

11.36

Aumento espectacular de su ``productividad".

11.37

Este trabajo abre los modelos económicos a sus flujos externos, y no los considera cerrados en sí mismos, al estilo de un ``equilibrio general" autocomplaciente. Considera que existen bienes terminales cuyo destino es satisfacer a los decisores y así rompe este círculo y permite vincular los recursos escasos con los recursos naturales.

11.38

Permite ``comparar y sumar peras con manzanas" desde la visión económica.

11.39

¿Y esto sería superabundancia? Claro, si sólo necesito 1500 calorías por día, disponer de 2000 sería superabundancia. Si quiero comer caviar en Salta, todos los días y esto representa 25000 calorías/día sólo en traerlo del mar Negro, y mi sociedad tolera estos comportamientos, esto no es superabundancia, es super-estupidez.

Cuanto más necesite hacer llamadas internacionales más ``ricas'' serán las empresas telefónicas, pues más escasez de enlaces estaré produciendo y se sabrá que más ``dinero'' deberé darles y el valor de sus acciones se incrementará.

11.40

Poder alimentar a la humanidad sin agotar sus tierras, poder darnos energía sin destruir el planeta: conocimiento sin que sus creadores pasen hambre, etc.

11.41

Como lo ha indicado Wilson [Wilson:B,Wilson:HN-78,Wilson:SNS-75], la humanidad se aproxima rápidamente a un cuello de botella, sobre todo si queremos mejorar nuestro nivel de vida promedio. Se requerirían 4 planetas Tierra, para llevar a toda la humanidad al nivel de consumo medio de los EEUU. Siempre la ciencia y la tecnología nos permitieron seguir disminuyendo la escasez sin llegar a los límites de Malthus, pero no es claro si esto podrá seguir eternamente de esta forma.

11.42

``Nada hay más difícil de emprender, más peligroso de liderar o más incierto de lograr, que un nuevo orden de cosas.'' [Maquiavelo:P]

11.43

O de cómo un sistema es determinado por sus recursos escasos. En los sistemas económicos se toman decisiones, por ello se habla de comportamientos más que de fenómenos. Estos comportamientos pueden ser llevados adelante por máquinas, inclusive humanas.

12.1

Establecido en el párrafo 1 incisos a) y b) del artículo 15 del [PIDESC], y en el párrafo 1 del artículo 27 de la [DUDDHH]: ``Toda persona tiene derecho a tomar parte libremente en la vida cultural de la comunidad, a gozar de las artes y a participar en el progreso científico y en los beneficios que de él resulten.''

12.2

Establecido en el artículo 13 del [PIDESC] y en el artículo 26 de la [DUDDHH]: ``Toda persona tiene derecho a la educación. La educación debe ser gratuita, al menos en lo concerniente a la instrucción elemental y fundamental...''.

12.3

Establecido en el párrafo 2 del artículo 19 del [PIDCP] y en el artículo 19 de la [DUDDHH]: ``Todo individuo tiene derecho a la libertad de opinión y de expresión; este derecho incluye el no ser molestado a causa de sus opiniones, el de investigar y recibir informaciones y opiniones, y el de difundirlas, sin limitación de fronteras, por cualquier medio de expresión''. Este derecho debe ser profundizado y ampliado notablemente en las sociedades del conocimiento [CRIS:SW].

12.4

en el marco de las sociedades del conocimiento, con su nueva base tecnológica y mecanismos de comunicación [UNESCO:SC-02,Bard:NNP-03]

12.5

Relaciones entre derechos:

1. Los tres derechos humanos ya citados: cultura, educación y comunicación, (así como los otros) tienen valor preeminente frente (y limitan) la normativa que se desprende del derecho del autor a beneficiarse de su creación, reconocido en el párrafo 1 inciso c) del artículo 15 del [PIDESC] y en el párrafo 2 del artículo 27 de la [DUDDHH]: `` Toda persona tiene derecho a la protección de los intereses morales y materiales que le correspondan por razón de las producciones científicas, literarias o artísticas de que sea autora";

2. La aplicación de éste derecho está limitada por el interés público y por su función social, lo que es reafirmado en el los documentos citados.

3. Los derechos relativos al conocimiento no limitan otros derechos humanos, en particular el derecho a la privacidad establecido al artículo 12 de la DUDDHH [DUDDHH]. La libertad del conocimiento no obliga a nadie a difundir una información, ni la hace públicamente disponible, sólo le da al que la conoce, el derecho de difundirla, no la obligación.

4. La libertad del conocimiento permite a la persona ejercer una solidaridad hoy prohibida. Por defecto, sin que el autor diga nada, y utilizando el derecho público, hoy, las expresiones de las ideas de un tercero no se pueden difundir. Con libertad, la expresión de una idea conocida, si la fuente no dice lo contrario expresamente, se puede difundir.

5. Los derechos del conocimiento están profundamente relacionados entre sí ya que no es posible ejercerlos aisladamente, no se puede comunicar sin conocer, y conocer sin haber sido comunicado o educado. Educar para la sociedad altamente sofisticada de hoy en día implica tener acceso a todo el conocimiento disponible, con libertad, desde los primeros niveles de formación, con contenidos y abstracciones apropiadas a cada nivel. Esto no es posible si se prohíbe difundirlo.

6. La ampliación del sistema de patentes [Knuth:CSP,Stallman:PSO-02] para incluir software impone dos tipos diferentes de normativa sobre este tipo esencial de conocimiento. Restringiendo aún más su difusión y uso.

12.6

Lo que es reafirmado en: los párrafos 13 y 18 de [PTCIDH]; el párrafo 1 de la [R2000SPPDH] y los párrafos 4, 5, 6, 7 y 8 de la [R2001SPPDH]; la [DUPCTP] y el artículo 6 de [DDD]; la [DPCCI]

12.7

Los efectos de los cambios tecnológicos:

1. El tránsito desde la sociedad industrial a la sociedad de la información implica cambios profundos.

2. Las reglas de apropiación exclusiva aplicables a los bienes materiales no son extensivas a la información y al conocimiento, porque:
    1. los bienes materiales [Judicial:INC] son "escasos" (la manzana: o es tuya o es mía) y sobre ellos se construye la economía;
    2. el conocimiento digitalizado se multiplica hoy sin costo marginal (si tengo una idea y te la comunico, los dos tenemos una idea), dado que se independizó económicamente de la base material que lo transporta y circula libremente por Internet [Saravia:EI-03,Schumpeter:CSD].
3. Que la información y el conocimiento no sean ya bienes escasos representa un progreso para la Humanidad; incorporar forzadamente la información a la economía mediante externalidades frenaría este avance benéfico.

4. El conocimiento es incontable, y cada persona que recibe una idea la tiene en forma independiente del que se la dio. El conocimiento es libre y se difunde o transmite. Cada persona puede difundirlo como quiera y en ese proceso, se va modificando. En tal sentido no es necesario compartir los derechos o usos sobre determinado conocimiento, ya que no hay un elemento sobre el cual tomar decisiones entre todos.

5. Internet, como red libre adoptada por las personas, entre varias otras redes cerradas que estuvieron disponibles, ofrece hoy nuevas posibilidades, para ejercer no sólo el derecho al conocimiento, sino también: a educarse y educar, a dar y recibir información, a comunicarse y sentirse comunicado con los demás, en una verdadera comunión del conocimiento.

6. La formación de comunidades virtuales, como las del movimiento por el software libre [Himanen:EHE-02,Saravia:SLA-03,Wong:FOS-03,FSF:SW,Stallman:GPL,Stallman:FDL,FSF:CSL,DiC
que constituye la herramienta cultural de la era, que construyó y fue construido por Internet, y que demuestra que se puede crear en libertad, es sólo el inicio de los cambios que se producen al conectar a las personas sin límites. El software es el código bajo el cual se ejecutará la ley provista por los automatismos y es esencial que sea libre [Lessig:C-99].

7. Prohibir a las personas que difundan su conocimiento solidariamente [Cornell:PC,MIT:LDC], porque otro retiene derechos sobre el mismo, conlleva el peligro de reforzar la pobreza con las profundas desigualdades de acceso a las nuevas tecnologías que ya han consagrado una brecha digital y un nuevo tipo de analfabetismo [Burcet:SRD-02,Abismo:SW,Copani:RDS-01].

8. Un desarrollo completo de la sociedad del conocimiento libre surge de la plena vigencia de estos derechos mediante el uso irrestricto de todas las posibilidades ofrecidas por Internet.

12.8

Es necesario modificar el derecho:

1. La legislación del derecho de autor, de patentes y todos los monopolios legales sobre creaciones intelectuales deben incentivar la difusión del conocimiento. Los cambios tecnológicos hicieron que los sistemas que incentivaban esta difusión hoy la frenan. El marco

jurídico vigente, consolidado en la era industrial con el fin de favorecer la difusión de la información y el conocimiento, resulta hoy anacrónico e injustificado [Saravia:MH-01].

2. Impedir el flujo de una información, que se activa cada vez que alguien que la conoce decide difundir su expresión en libertad, perjudica a las personas y a la sociedad, y beneficia sólo a intereses particulares minoritarios (que no necesariamente coinciden con los de los autores).

3. Imponer el marco vigente requiere hoy excesiva regulación estatal y poderes dictatoriales, control policial, ingresar a hogares e instituciones para controlar licencias, reemplazo del comercio libre por impuestos, penalización de costumbres arraigadas, arrestar personas en conferencias sobre criptografía, impedir que países fabriquen medicinas baratas, vigilancia invasiva, autorización centralizada y en línea de ejecución de software, alteraciones en el hardware de las computadoras, la criminalización de Internet, y hasta considerar pecado el difundir el conocimiento [Saravia:DDS-03,Inquirrer:ITW-03,IPJustice:WPA,Eff:TC,p2pnet:TPE,TCG:SW,Anderson:PFI-03,Anti

13.1

Sería bueno que haya un flag que autodetecte qué es.

13.2

Tarea enorme.