Death Master's FAQ – El Software Libre y GNU/Linux

Revisión 1.1

Introducción

Antes de nada, me gustaría explicar los motivos para realizar este pequeño FAQ (Frecuently Asked Questions o Preguntas Más Frecuentes). El principal motivo es para arrojar un poco de luz sobre un tema del que existe mucha falta de información y, sobre todo, gran cantidad de información incorrecta. Muchos son los mitos y las verdades a medias respecto al tema del software libre.

Por otro lado, siempre he creído que en el conocimiento reside la verdadera libertad. Así pues mi meta primera en la red de redes es difundir el conocimiento a todo aquel que esté dispuesto a aceptarlo. Esto se aplica especialmente al código abierto, pues a veces la mejor manera de animarse a probar algo es que alguien te lo de a conocer.

El último es aclarar los conceptos de ciertas personas en concreto, que por lo que he leído, andan un poco perdidos en estos temas y necesitan leer algo como esto ;-)

1- Aclarando conceptos

¿Linux? Un poco de historia de GNU/Linux

El primer concepto equivocado que voy a citar, y uno de los más extendidos hoy en día es la propia denominación del sistema operativo favorito del tito Death ;-). Bien es cierto que todos cometemos este pequeño fallo por comodidad y por costumbres, pero es bueno saber en qué consite el fallo, aunque se siga cometiendo.

Linux no es más que una parte del sistema operativo que todos conocemos (o deberíamos :-P), concretamente el kernel o núcleo del sistema. La otra parte es GNU, y es el entorno de aplicaciones del sistema operativo, que por tanto se llama GNU/Linux.

La historia se remonta a la creación de ambos componentes. En el caso de Linux (el núcleo o kernel, el corazón de GNU/Linux) nació el 3 de Julio de 1991, con un email de Linus Torvalds pidiendo ayuda al grupo Minix de su ciudad para desarrollar un proyecto que tenía en mente. El email original es este:

From:torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroup: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID: 1991Aug25, 20578.9541@klaava.Helsinki.FI
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki.

Hello everybody out there using minix-

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix; as my OS resembles it somewhat (same physical layout of the file-sytem due to practical reasons) among other things.

I've currently ported bash (1.08) an gcc (1.40), and things seem to work. This implies that i'll get something practical within a few months, and I'd like to know what features most people want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linux Torvalds torvalds@kruuna.helsinki.fi

A partir de ahí lo que comenzó como un “hobby” ha llegado a ser uno de los sistemas operativos más famosos del mundo.
GNU por su parte es el entorno de aplicaciones que rodean al núcleo del sistema operativo. GNU es un acrónimo de definición recursiva. GNU = GNU's Not Unix. Richard Stallman fue el fundador de dicho proyecto, y quería dejar claro desde el primer momento que GNU no era Unix :-).

Cuando ambos proyectos fueron madurando, llegaron a un punto muerto, pues Stallman tenía GNU bajo licencia GPL (más adelante se explicará) pero le faltaba un kernel. Linus tenía su kernel pero no tenía decidido bajo que licencia lo presentaría. Finalmente se decidió (no sin reservas) por la GPL, pero su kernel estaba “huérfano” y no tenía un entorno de aplicaciones. Parecía, y de hecho fue, natural la unión de ambos proyectos en lo que hoy se conoce como GNU/Linux.

Por desgracia está arraigada la costumbre de denominar al sistema operativo Linux, olvidando parte fundamental de su ser (y de paso cabreando sobremanera a Stallman :-P).

¿Linux = Software libre?

Mucha gente cree que software libre es sinónimo del sistema operativo Linux, y eso no es cierto en absoluto. GNU/Linux es un sistema operativo licenciado bajo GPL, pero ni únicamente la GPL representa al software libre, ni Linux es el único sistem operativo libre. Como muestra de otras licencias libres tenemos BSD, LGPL y MPL, aunque existen muchas otras. Otros sistemas operativos libres son FreeBSD y OpenBSD, ambos derivados BSD y compatibles Unix igualmente. Estos conceptos se aclararán más adelante.

¿Software libre = Software gratis?

Esta es otra asociación errónea que hace muchas veces la gente, y es una verdadera pena que se desvirtúe el concepto de software libre por el mero factor económico. Si bien casi todo el software libre es gratuito, existen excepciones como versiones avanzadas de distribuciones de Linux y otros ejemplos particulares.

Este error viene dado por culpa de una ambigüedad en inglés con el término free, que tiene dos acepciones bien distintas: libre y gratuito. Debido a esta ambigüedad se ha pasado a denominar, con mayor acierto, “Open Source” a lo que antes se denominaba como “Free Software”.

Así pues, ¿qué es lo que define al software libre como tal? La disponibilidad de su código fuente.

Como habrá gente que no sabrá lo que es el cógido fuente, hago una ligerísima explicación para que se entienda. El ordenador sólo comprende instrucciones binarias, pero a la hora de programar se usa lo que se denomina “lenguaje de programación” que son ciertas instrucciones que siguen unas reglas y una sintaxis, y que luego el compilador se encarga de traducir a un lenguaje que la máquina comprenda. Sí, los puristas dirán que esto no es ni un ensayo de definición, pero creo que al menos los que no supieran de qué iba la historia estarán enterados.

La disponibilidad del código fuente implica la posibilidad de modificarlo (aunque las licencias puedan decir lo contrario, poseer el fuente nos permite -sino legalmente, sí efectivamente- modificarlo a nuestro antojo. Y otra función importantísima de esa disponibilidad es conocer al detalle el funcionamiento de cualquier software, pudiendo estar seguros de lo que hace y lo que no hace. Pero ya detallaré más adelante este tema...

¿Software libre = Software “amateur”?

Otro de los grandes topicazos respecto al tema del software libre es la idea equivocada de que el Software Libre sólo son unos programas desarrollados por unos cuantos programadores aficionados, y que por tanto no puede esperarse ningún tipo de garantía, fiabilidad o respaldo. Tampoco es así.

Existen muchas empresas (de la talla de Red Hat Inc., SuSe, Mandrake, IBM, Sun Microsystems...) que desarrollan activamente Software Libre, así como muchísimos programadores que trabajan en compañías dedicadas al software propietario se dedican en su tiempo libre a participar en desarrollos de software libre. Se ha comprobado que no sólo no es un sinónimo de constancia sino todo lo contrario. El software libre está en constante desarrollo y tiene unos tiempos de reacción a los fallos muy cortos, así como sus ciclos de actualizaciones.

Software libre, Freeware, Adware...

Hay mucha gente que no se aclara con estos conceptos, así que daré pequeñas definiciones de estos tipos de programas para que nadie se llame a confusión:

Freeware: Se trata de software gratuito, pero cuyo código fuente NO es conocido ni liberado. Este tipo de software es muy común en Windows, con ejemplos como Winamp, Regcleaner, XPantispy y muchos otros. (Podéis encontrar muchos ejemplos en http://www.donfreeware.com). En Linux también existen programas freeware, con código fuente NO conocido, y que

suelen ser distribuidos mediante paquetes precompilados (rpm, deb...), tarballs de binarios, instaladores (InstallShield...) y otros métodos. Ejemplos de esto son el cliente de TeamSpeak2 para Linux o la máquina virtual de Java de Sun Microsystems.

Shareware: Se trata de programas de evaluación, con ciertas limitaciones. Estas limitaciones a veces comprenden un uso limitado de las funciones del programa, un tiempo máximo de uso (normalmente 30 días) o cualquiera otra. Son muy populares, y lamentablemente es muy fácil convertir un software shareware en completo.

Adware: Este sistema es parecido al freeware pero con una diferencia: instala algún tipo de publicidad, modificación en el navegador de Internet, o la incluye el propio programa a modo de “pago” por el uso gratuito. Es muy típico de aplicaciones dudosas de Windows.

Spyware: Este tipo de software básicamente suele ser freeware pero con algún tipo de módulo encargado de extraer y hacer llegar a alguien información de tu PC: hábitos de navegación, nombre, especificaciones técnicas del equipo... Luego esos datos son vendidos a agencias de publicidad y vuestros buzones de correo se llenan de spam (¿Nunca os preguntásteis de dónde sale?). El propio sistema operativo Windows incluye varios tipos de spyware.

Sobre el software libre ya he hablado y hablaré más así que no doy detalles, pero es importante diferenciar entre software freeware o software libre, porque muchas veces he oido este tema.

El Software libre sólo es para sistemas operativos libres

Otro craso error ;-)

Cada vez es más común que existan versiones de los programas de software libre que funcionen bajo sistemas operativos propietarios. En el caso de los sistemas propietarios Unix, no hay mucho que hacer, pues con un gcc basta para compilarlo. En el caso de Microsoft Windows la cosa no es tan fácil, pues compilar las fuentes requiere más trabajo que, por ejemplo, en un sistema GNU/Linux.

Por suerte para los usuarios de Windows, cada vez hay más desarrolladores de software libre que junto a los tarballs y a los paquetes precompilados para distribuciones de Linux, cuelgan instaladores o versiones precompiladas para sistemas operativos Windows. Os puedo dar ejemplos de grandes programas de software libre que yo mismo tengo instalados en mi Windows XP y uso como principales: Mozilla Firebird (navegador web), Mozilla Thunderbird (cliente de correo electrónico), X-chat (cliente de IRC), OpenOffice.org (suite ofimática), Gimp (tratamiento de imágen), gnuPG (encriptación PGP)... y existen muchísimos otros programas.

Si un usuario no puede o no quiere deshacerse de Windows, no tendrá porqué gastarse más que la licencia de Windows (seremos ingenuos y pensaremos

que somos todos legales), pues el resto puede encontrarlo entre software libre o freeware.

Muchos programas de Windows no tienen equivalente en software libre

Si bien es cierto que existen excepciones rarísimas por las que un software específico de Windows no puede ser encontrado para sistemas Unix, también es cierto que se da el fenómeno contrario, y con mucha más asiduidad, cabe señalar.

Para cualquier usuario medio, el software que pueda necesitar lo encontrará parefectamente para un sistema GNU/Linux. Un buen punto para empezar a buscar es la tabla de los equivalentes (http://linuxshop.ru/linuxbegin/win-lin-soft-en/). Además es curioso ver como por cada 2 ó 3 programas de Windows encontramos al menos 10 para Linux.

Hay que ser un experto informático para usar Linux

Claramente respondo con un rotundo NO. Quizá en los tiempos de SLS la cosa sí fuera así, y es cierto que aún existen distribuciones que para instalarlas es necesario el sacrificio de cabras negras, pintar símbolos raros con su sangre y tocar tambores, pero por norma general no.

Hoy en día la instalación de una distribución de Linux es tan sencilla o más como pueda ser la de un sistema Windows. Autodetección de hardware, configuración de redes y dispositivos...

En la cuestión del entorno de trabajo, hay que apartar de la cabeza a la gente que Linux es sinónimo de tediosas y largas órdenes en el intérprete de comandos (consola o shell). Si bien es cierto que al ir adquiriendo cierta soltura con el sistema operativo se comienza a preferir ese sistema, no es necesario conocer nada de la consola para poder usar un Linux. De hecho, ¿cuántas personas que usen Windows conocen los comandos de MSDOS, digamos, para ocultar un archivo? Seguramente poca gente. Gracias a las X windows y los escritorios como KDE (K Desktop Environment) o gnome, cualquier usuario acostumbrado a Windows no tendrá un trauma en la migración de sistema operativo.

2- Las licencias del Software Libre

A continuación voy a citar algunos ejemplos de las principales licencias de software libre que existen y se usan hoy en día, así como ejemplos de software libre acogido a ellas. También aportaré enlaces a la licencia completa (en inglés, eso sí, por ser su idioma original).

GPL (GNU General Public License)
http://www.opensource.org/licenses/gpl-license.php

Se trata de la licencia “abanderada” del software libre, y es también la más extendida y utilizada. La General Public License no es demasiado larga, para

lo que suelen ser las condiciones de licencia y los diversos tipos de copyright.

Básicamente este tipo de licencia te otorga derecho de uso libre de cualquier programa licenciado bajo GPL. Así mismo, cualquier programa GPL debe ser publicado junto a su código fuente, y cualquiera puede realizar modificaciones o mejoras en estos programas, respetando y citando siempre al autor original y -muy importante- poniendo a disposición de la comunidad las modificaciones realizadas en el código fuente. El uso de software de licencia GPL está sujeto al propio cumplimiento de la licencia.

Software típico bajo licencia GPL es el propio kernel de Linux, o los entornos KDE y gnome.

BSD
http://www.opensource.org/licenses/bsd-license.php

La licencia BSD es más breve aún si cabe que la GPL, y es el "corazón" de los sistemas operativos BSD (freeBSD, openBSD...) que son otros sistemas operativos Unix-like, pero que NO tienen relación con Linux.

La licencia BSD se fundamenta en la disponibilidad del código fuente, y la libertad de uso y modificación de este software por cualquier persona. Pero -y aquí es donde se presenta el punto de divergencia- no es obligatorio la publicación y liberación del código fuente por parte del autor de la modificación. Personalmente me parece un atraso y una corriente contraria al software libre...

Como ya he dicho, freeBSD, openBSD y otros sistemas operativos libres BSD son ejemplos de este tipo de licencia. Pero ojo, no todos los sistemas operativos BSD son libres, pues existen algunos propietarios.

LGPL (GNU Lesser General Public License)
http://www.opensource.org/licenses/lgpl-license.php

Hasta hace no demasiado (ahora mismo existe la versión 2.1) se denominaba Library General Public License, pero el uso por otro tipo de aplicaciones a parte de librerías ha provocado la "extensión" de esta definición.

Básicamente se trata de una variación de la GPL original, mediante la que se distribuye el programa libre, así como su código fuente. La principal divergencia es que NO se permite la libre modificación y distribución de este código fuente. Esto no significa que el software no pueda ser modificado por particulares, pero normalmente es regulado a través del desarrollador principal, que acepta sugerencias y código modificado. Lo que no está permitido es la libre distribución de software LGPL modificado.

Típicos ejemplos de LGPL son muchas librerías del sistema GNU, así como la suite ofimática OpenOffice.org (que efectivamente acepta modificaciones de código y sugerencias del público general en su código).

MPL (Mozilla Public License)

http://www.opensource.org/licenses/mozilla1.1.php

Ésta licencia es particularmente larga. Se fundamenta en la disponibilidad libre del software, así como su código fuente. La principal particularidad de esta licencia es que no es obligatoria la publicación de las modificaciones realizadas en el código, y todos podemos ver el caso de Netscape, que usaba código de Mozilla pero sin aportar las modificaciones realizadas y convirtiéndolo en software propietario. Esto puede llevar a que el software propietario se aproveche de las mejoras libres y se guarde las propias, aunque en ese caso particular de Mozilla y Netscape, Mozilla siempre innovaba y Netscape implementaba las mejoras cuando eran “estables”.

Esta licencia es típica del navegador y la suite Mozilla.

ASL (Apache Software License)
http://www.opensource.org/licenses/apachepl.php

Licencia muy corta también, que incluye como todas, la distribución del software, así como su código fuente, y la libre modificación de éste. Las condiciones deben ser nombrar al autor original del código (Apache), así como la imposibilidad de usar Apache como nombre de reclamo comercial, ni incluir la palabra Apache en el nombres del software.

Como es obvio, el servidor web apache es el mejor ejemplo de esta licencia.

AFL (Academic Free License)
http://www.opensource.org/licenses/afl-2.0.php

Licencia muy relajada, destinada al uso académico. Permite prácticamente cualquier tipo de reproducción, modificación y copia del software. Una de las pocas restricciones existentes es la no utilización como reclamo del autor original del código.

Otras licencias importantes

IBM Public License: http://www.opensource.org/licenses/ibmpl.php
Intel Open Source License: http://www.opensource.org/licenses/intel-open-source-license.php
Jabber Open Source License:
http://www.opensource.org/licenses/jabberpl.php
Nokia Open Source License: http://www.opensource.org/licenses/nokia.php
PHP License: http://www.opensource.org/licenses/php.php
Python License: http://www.opensource.org/licenses/pythonpl.php
QT Public License: http://www.opensource.org/licenses/qtpl.php
Sun Public License: http://www.opensource.org/licenses/sunpublic.php

Existen muchas otras licencias de software libre, pero he detallado las más importantes, así como he dado enlaces a otras de mucha importancia también.

2- La dimensión técnica del Software Libre

La primera cuestión que suele plantearse alguien cuando le hablas de proyectos de software libre es su viabilidad. Ya he explicado anteriormente que los desarrollos de software libre no corren necesariamente a cargo de programadores en su tiempo libre. Y sobre todo la idea más equivocada es que los programadores de software libre son “aficionados”. Idea equivocada porque la mayoría de los programas de software libre se dearrollan en lenguaje C (que no C++ o C#), siendo este un lenguaje de programación estructurado (en contraposición a los lenguajes “visuales” que son orientados a objetos) de alto nivel, pero que permite un trabajo a bajísimo nivel. Cualquiera que sepa programar podrá dar fe que un proyecto para ser realizado en C requiere un esfuerzo de programación bastante potente, a cambio de conocer el programa detalle a detalle. En lenguajes como Visual Basic, se desarrollan aplicaciones gráficas con gran facilidad, pero a costa de una ligera pérdida de control sobre lo que hace el programa (por no hablar de las librerías que se usan en este lenguaje...). Por tanto se puede decir que la practica totalidad de los programadores de software libre son muy duchos en su oficio.

Otro de los grandes miedos de los “profanos” de esta materia es el tema de la seguridad y sus actualizaciones. Las grandes corporaciones como Microsoft se esfuerzan en darnos la idea equivocada de que un producto cuyo código fuente está disponible en Internet es más fácilmente vulnerable por el conocimiento que existe de su forma de trabajar. Esto NO es cierto. Citando un buen ejemplo que leí hace no mucho en una revista “Saber cómo está construida una puerta de acero de un metro de espesor no facilita las cosas a la hora de querer entrar por ella una vez colocada”. Y así ocurre con el software libre. Existen muy buenos softwares de seguridad -incluso para Windows- que son código abierto. El tema de la solución de los posibles fallos encontrados es uno que Microsoft siempre elude, simplemente porque el software libre siempre tiene una capacidad de reacción muchísimo mayor. Hoy día hay millones de desarrolladores de software libre, y cuando un fallo es publicado, la corrección del código puede ser realizada por cualquier persona, incluso por uno mismo si posee la suficiente pericia en programación para hacerlo. En el caso de software propietario, los fallos aparecen de igual manera, y muchas veces se practica ingeniería inversa para ello (aclaración: se conoce como ingeniería inversa la reproducción de algo sin ayuda de documentación, código o modelos de referencia), pero la celeridad a la hora de reaccionar contra esas vulnerabilidades, a pesar de que ciertas corporaciones tienen un volumen impresionante, es bastante más lenta. Eso si hacen caso de los avisos. Recuerdo un caso más o menos reciente, en que un famoso hacker pakistaní descubrió una vulnerabilidad en el sistema de correo electrónico hotmail, y avisó en más de cuatro ocasiones a Microsoft. Como nadie le hacía caso, publicó la vulnerabilidad (que permitía el reseteo de la contraseña y consiguiente inutilización de la cuenta asociada) en Internet. Miles de cuentas inutilizadas, pero en apenas 24 horas Microsoft tenía el problema controlado, aunque podían haberlo hecho de una manera más elegante y sin ese revuelo público.

Otro tema interesante a debatir al respecto del desarrollo del software libre es la vanguardia tecnológica y el desarrollo de nuevas tecnologías. Aquí encontramos bastante heterogeneidad, pues en algunos tipos de software, el

código abierto lleva un atraso importante con respecto al software propietario, y sin embargo en otros aspectos es a la inversa. No debemos olvidar, no obstante, que el software libre tiene alrededor de diez años de vida, mientras que el propietario tiene una amplia historia a sus espaldas (para bien o para mal). Citaré algunos casos de software comunmente utilizado por un usuario de nivel medio:

En software de **navegación web (browsers)**, hace mucho tiempo que Mozilla (http://www.mozilla.org/) lleva la voz cantante en el desarrollo y la innovación, a pesar de que Internet Explorer siga siendo el navegador más usado. Debemos recordar que uno de los navegadores favoritos de los usuarios de Windows es el Netscape Navigator, que es en su base, código de Mozilla ligeramente retocado. El estándar de compatibilidad en web es Mozilla, siendo Internet Explorer un navegador “Mozilla compatible”. El desarrollo de mozilla es hoy día mucho más veloz y eficaz que cualquier otro navegador libre o propietario, y podemos citar por ejemplo a Internet Explorer, Opera, Galeon, Konqueror...

Otra comparación famosa es **la ofimática**. En este campo, por desgracia, Microsoft lleva bastante ventaja aún al software libre -OpenOffice.org (http://www.openoffice.org/)- aunque hay que decir igualmente que la tendencia de esa ventaja es ir reduciéndose conforme pasa el tiempo, pero aún quedan algunos años para poder salvar esa distancia. Un motivo para ello es que Microsoft cuida y desarrolla mucho su suite Microsoft Office, teniendo una de las mejores hojas de cálculo que hay (de la base de datos mejor no hablamos :-P). Aún así, hoy día OpenOffice.org está desarrollado hasta un punto en el que puede satisfacer las necesidades de cualquier usuario e incluso cualquier empresa. Si bien no incluye base de datos, existen muchas opciones tanto libres como propietarias que son muy buenas y que son mucho mejores en todos los aspectos que Access.

El tema del **multimedia** es cada vez tenido más en cuenta por el usuario medio de un PC. Hoy en día todo el mundo quiere poder escuchar sus canciones favoritas en mp3 mientras navega por Internet, o poder ver una película en DVD o DivX cuando tiene un rato libre. Aunque en este campo hay que reconocer que Microsoft no lo ha hecho tan mal, pues el Windows Media Player no es un mal software (el tema de la cantidad de spyware que incorpora lo dejamos de lado :-P). Aún así, la oferta en software libre es bajo mi punto de vista mucho más completa. En reproductores de audio, en Windows se puede decir que el estandar es Winamp (freeware), y además de existir una versión de este último para Linux (liberada hace no demasiado tiempo), en mi opinión X Multimedia System (XMMS) (http://www.xmms.org/) es una opción mucho más potente (al ser software libre, por contra de Winamp). Si hablamos del terreno de reproducción de vídeo, aunque como ya dije Windows Media Player o los distintos reproductores multimedia propietarios que existen para Windows son bastante buenos, no existe color a la hora de la comparación con la -para mí- herramienta más potente de reproducción multimedia: Mplayer (http://www.mplayerhq.hu/homepage/). Con este software he podido reproducir absolutamente todo lo que se me ha pasado por la cabeza: cualquier tipo de archivo de audio, cualquier archivo de vídeo (incluyendo

DVD, VCD, SVCD, xVCD, DivX, XviD, mov...). Aunque el manejo por línea de comandos es mucho más potente y ofrece mayores posibilidades, el propio programa integra de serie una potente GUI (Graphic User Interface ó Interfaz Gráfica de Usuario) denominada gmplayer y que permite usar el programa como usarías cualquier software similar en Windows.

En el terreno de la **mensajería instantánea** también hay representantes para todos los gustos en software libre: la red MSN messenger cuenta con programas como amsn (http://amsn.sourceforge.net), mucho mejor que el cliente oficial en mi opinión, Kmess (http://kmess.sourceforge.net) que es otra gran opción para entornos KDE, o Kopete (http://kopete.kde.org/). La red ICQ tiene también muchos representantes como Licq (http://licq.org/) o Kicq (http://kicq.sourceforge.net/kicq.shtml). Para la red Jabber (creciendo cada día más, es la alternativa de red libre a las demás) tenemos Psi (http://psi.affinix.com/) que también se usa -y yo lo uso- en Windows. Podemos así mismo usar clientes multiprotocolo, como GAIM (http://gaim.sourceforge.net/) o Ayttm (http://freshmeat.net/projects/ayttm/). Para la red de chat IRC tenemos poderosos clientes como X-chat (http://xchat.org/) que yo uso en ambos sistemas operativos, Kvirc (http://www.kvirc.net/), IRSII (http://www.irssi.org), BitchX (http://www.bitchx.org/) y muchos más. Como podréis ver, la oferta está bastante completita.

Si hablamos de **peer to peer** estamos en el mismo punto que en el apartado anterior, poseemos clientes para cualquier red usada hoy día, a saber: en la red eDonkey (la que usa el eMule :-P) está el propio eDonkey para Linux (http://ed2k-gtk-gui.sourceforge.net/download.shtml) o Xmule (http://www.xmule.org). Para la red Overnet existe el propio cliente nativo de Linux para Overnet (http://www.overnet.com/download.html). En el caso de la red Bittorrent existn clientes como Snark (http://freshmeat.net/projects/snark/). Para el protocolo de winMX existe Loophole (http://www.loopholesoftware.com/). Para el protocolo Gnutella existen LimeWire (http://www.limewire.com/), GTK-Gnutella (http://gtk-gnutella.sourceforge.net/) o Qtella (http://www.qtella.net/). Para el protocolo Soulseek tenemos PySoulSeek (http://www.sensi.org/%7Eak/pyslsk/). Así mismo, y como en el caso anterior, tenemos la posibilidad de usar un cliente multiprotocolo, como por ejemplo Mldonkey (http://www.freesoftware.fsf.org/mldonkey/).

Creo que con esto queda claro que alternativas para software libre no faltan en ningún caso a un usuario de nivel medio que desee adentrarse en este mundo. Y espero también que poco a poco se vaya disipando esa “duda” sobre la disponibilidad de software para código abierto. Ahora toca el turno del desarrollo tecnológico, una parte quizá un poco más “friki” que la anterior, pero que nadie se me pierda. ;-)

Podemos empezar por ejemplo por el propio diseño del núcleo del sistema operativo o **kernel**. En el caso de Windows ha sido tradicionalmente un sistema de kernel monolítico, aunque en los sistemas NT se ha cambiado por un sistema de microkernel. El quid de la cuestión es que este kernel es intocable, es un todo que se instala en tu sistema, habiendo sido previamente

precompilado en otra máquina (de especificaciones inferiores). El kernel cargará todas sus funciones independientemente de lo que necesites. En el caso de los sistemas GNU/Linux, la concepción del kernel es mucho más versátil. El kernel de Linux es de diseño totalmente modular, y puede ser modificado y compilado en cada sistema. Si hace años que sustituiste tu escáner e impresora de puerto paralelo por USB y no necesitas usar el puerto paralelo, ¿por qué debería tu sistema cargarlo con el consiguiente desperdicio de memoria y recursos? Pues muy sencillo: no lo cargues. Esto es aplicable a cualquier parte del kernel. Y existe un concepto más allá aún en su modularidad: si tú por ejemplo quieres de vez en cuando navegar con el protocolo IPv6 en lugar del tradicional IPv4, quizá cargarlo para no usarlo muy a menudo sea un gasto inútil de recursos... pero no cargarlo te imposibilitaría su uso. Por otro lado, andar compilando el kernel cada vez que queramos usarlo sería tedioso en extremo. La solución también la da el kernel de Linux: compílalo como módulo. Cuando arranque el sistema, no estará cargado en memoria, y cuando lo necesites, simplemente cargas ese módulo del kernel. Y si entramos en el tema de la compilación, creo que si tienes por ejemplo un Pentium4 o un Athlon XP, preferirás un kernel compilado para el juego de instrucciones de TU procesador antes que, por ejemplo, un Windows 2000 compilado con el juego de instrucciones de un Pentium2. También hay que tener en cuenta que en los sistemas Linux podemos actualizar el núcleo del sistema operativo como si de un software más se tratara. Echad un vistazo a http://www.kernel.org y veréis las versiones disponibles.

Hay una tecnología, en desarrollo aún, que también está comenzando a usarse por empresas, gobiernos y curiosos (como yo xD) y que de aquí a unos años se convertirá en el nuevo estándar de protocolo de Internet: **IPv6**. No quiero entrar en explicaciones técnicas porque es un tema ligeramente complejo de explicar en toda su extensión, pero hoy en día estamos todos acostumbrados al IPv4, a todos nos suena lo que es una IP, como por ejemplo 127.0.0.1 (loopback). Ese número es de 32 bits e identifica unívocamente a una máquina en la red. El nuevo protocolo viene dado por la extenuación del antiguo, nos quedamos sin Ips :-P. Con el nuevo protocolo, una IP tendría una forma más o menos así: 2002:1322:a42c:231d:39ab:55de:acb1:3521. Vale, ahora bajad las manos de la cabeza. Tú, no te rasgues las vestiduras :-D. Esto es un ejemplo para que comprendáis el cambio de protocolo. Las nuevas IPs tendrán 128 bits y permitirá que exista un número de IPs tan elevado que no podrán agotarse (aunque nunca es bueno decir de este agua no beberé). Toda esta pequeña introducción técnica sirve para exponer un hecho: la implantación de nuevas tecnologías en los sistemas operativos. Para hacer honor a la verdad, Linux no fue el primer sistema operativo en incorporar este nuevo protocolo. El primero fue freeBSD, y tras él, GNU/Linux. Pasado bastante tiempo, con Windows 2000 SP1 y Windows XP, Microsoft incluyó la posibilidad de usar este protocolo. Con el tema de sistemas de ficheros ocurre algo similar, pues mientras Windows compatibiliza con sistemas FAT16, FAT32 y NTFS (que por cierto vaya chapuza el intento de compatibilidad con Mac, eso de los ficheros Stream tiene más peligro...). Linux incluye de serie compatibilidad -entre otros- con ext2, ext3, FAT16, FAT32, así como en los nuevos kernels NTFS de escritura y lectura (antes era sólo de lectura).

Otro tema importante es la cuestión de la propia comunicación del entorno de

aplicaciones con el kernel (que a su vez se “comunica” con el hardware). Esta comunicación se realiza a través de la shell. En el caso de los sistemas Unix-like, y por consiguiente de los BSD y GNU/Linux, la separación entre shell y kernel es bastante notable, permitiendo independencia entre ambos. No estamos limitados a un mismo kernel, pudiendo cambiarlo en el momento que queramos, y tampoco estamos limitados a usar una misma shell. Esto nos permite correr varias combinaciones de shell y kernel en un mismo sistema sin necesidad de instalaciones separadas. En el caso de los sistemas Windows esta separación es meramente conceptual, pues al tratarse de un sistema cerrado y monolítico, estamos obligados a un uso predeterminado de ambos componentes.

Otros conceptos técnicos a tener en cuenta son los **multiusuario** y **multitarea**. Cuando hablamos de sistemas multiusuario no nos referimos a la capacidad de hacer “login” con distinto nombre, sino a la posibilidad real de trabajo simultáneo de varios usuarios. Unix desde siempre ha sido un sistema multitarea real, al igual Linux o BSD. En el caso de Windows, la funcionalidad multiusuario apareció con los sistemas New Technology (NT, 2000, XP, 2003), y aún así el trabajo simultáneo entre usuarios no es todo lo efectivo que nos gustaría que fuera. Multitarea es un concepto más complicado. Si miramos en la wikipedia, se define multitarea (http://www.wikipedia.org/wiki/Multitasking) como la aparente simultaneidad de dos o más tareas en una única máquina, y nos remite para más información a Multitarea computacional (http://www.wikipedia.org/wiki/Computer_multitasking). No quiero aburrir con asuntos técnicos así que lo resumiré diciendo que la multitarea consiste en la coejecución de procesos en tiempo de proceso del microprocesador. El sistema más sencillo (y cutre, porqué no decirlo) es por el que una aplicación “cede” parte de su tiempo en ejecución a otros procesos para que pueda existir esa simultaneidad. Este proceso se denomina multitarea cooperativo. En máquinas rápidas ya no se nota diferencia prácticamente, pero este sistema está bastante cogido por los pelos. Sistemas típicos de este tipo de multitarea son los Windows 9x (95, 98, 98se, ME). Hoy día se habla de multitarea “real” cuando esta funcionalidad está perfectamente controlada y definida por el kernel del sistema operativa, y auxiliada por el sistema de interrupciones de entrada y salida. Los sistemas Unix-like (Linux y BSD) están entre estos sistemas de multitarea “real”. Si queréis probar, podéis ejecutar en Windows las siguientes tareas: dos sesiones de compilación en C, suite ofimática, un analizador de procesos, disco, red y ejecución del sistema, tres navegadores web, un reproductor de mp3, tres clientes de mensajería instantánea, un cliente de correo electrónico, un cliente de IRC, dos intérpretes de comandos, un explorador de archivos y un gestor de descargas FTP a la vez (sin contar los procesos residentes). No me miréis así, lo estoy haciendo yo ahora mismo y esto va de maravilla :-D.

La propia concepción de la estructura de comunicación con hardware de cada sistema operativo es llamativamente distinta. Mientras en sistemas Windows tenemos una serie de APIs y de librerías dinámicas (DLLs) que como ya sabemos resultan inestables en exceso, en los sistemas Linux tenemos la filosofía heredada de Unix de “todo es un archivo”. Así, un comando en consola de “cat /proc/cpuinfo” nos dará la información del procesador, simplemente volcando al buffer de pantalla la información contenida en el

fichero. La filosofía de funcionamiento del software también es radicalmente distinta. En sistemas Linux existen mutlitud de librerías y funciones comunes que usan todos los programas (típicamente GTK y librerías gráficas) por lo que el programa se limita a incorporar el código justo y necesario. Así podemso encontrar casi cualquier tipo de programa con entre 0,5 y 5 Mb de espacio. En Windows la filosofía es no compartir nada, cada programa que se busque la vida. Y así nos duran los discos duros lo que nos duran. Como curiosidad diré que con un software similar, y eliminando archivos de audio y vídeo, así como juegos, mi entorno Linux ocupa unos 5,5 Gb, y mi entorno Windows XP ocupa unos 20 Gb.

Por último voy a hablar de un tema muy importante según mi concepción de la informática: **la seguridad**. Aunque en este caso la comparación no puede ser tan general como “software libre contra software propietario” pues ocuparía horas y horas, reduciéndola a “Windows vs Linux” podemos citar algunos puntos por encima y sin entrar en mucho detalle técnico para no aburrir al personal. Lo primero que hay que tener en cuenta cuando queremos una mínima seguridad en un sistema (además de los fallos evidentes) es que no tengamos puertos abiertos. En el caso de Linux, no habrá más puertos abiertos que los de aquellos procesos de inicio que tú decidas ejecutar. Con un rcX.d limpio, tu sistema arrancará con los puertos totalmente cerrados (podéis comprobarlo con el comando netstat y una sintaxis para que muestre conexiones de red y elimine sockets de Unix, por ejemplo “netstat -tupan”). En sistemas Windows, por desgracia y nunca lo eliminan por muchos años que pasen, se empeñan en que tu sistema tiene que tener abiertos varios puertos. Un clásico en los sistemas Windows es el protocolo netbios y SMB. Desde siempre, aunque no usaramos la funcionalidad de compartir archivos e impresoras a través de redes, teníamos abierta una lista importante de puertos (139, 135 y 137. En sistemas NT, además 445). Esto plantea un grave problema, pues esos puertos abiertos es la única invitación que necesitan ciertas conexiones para poder comunicar con su ordenador. Claros ejemplos de esto que estoy citando son el exploit de denegación de servicio (DoS) que afecta a los sistemas NT, 2000 inferior a SP2 y XP inferior a SP1 por su protocolo SMB. Cualquier sistema con el puerto 445 abierto y sin el debido parche, puede ser reiniciado remotamente. Si el sistema tuviera el puerto cerrado, aunque no estuviera parcheado, no existiría el problema. Otro caso más reciente y más llamativo aún es el del gusano MSblaster. El exploit de desbordamiento de buffer del protocolo RPC se ejecutaba a través de los puertos 135 abiertos (osea, cualquier Windows sin proteger) y luego se abría otros enlaces al exterior (4444 para la shell y la conexión trivialFTP). Como veréis, estos fallos podrían haber sido evitados con las debidas precauciones. Muchos tendréis ya en la cabez auna palabra: Firewall. Efectivamente, un cortafuegos puede cerrar todos esos puertos y filtrar el tráfico, pero no deja de ser un añadido al sistema operativo, mientras que en sistemas Linux no es necesario (y de serlo poseemos en el propio sistema Iptables incluído en el kernel). Podemos mencionar también el tema de las escaladas de privilegios, del compromiso de la información, contraseñas de usuarios, etc etc etc. Pero como ya dije, no es mi intención crear un artículo técnico.

3- La dimensión económica del Software Libre

Aunque en principio pueda parecer que el tema económico y el software libre están separados, no es tanto así. No puedo entrar en grandes detalles con respecto a este tema porque la economía no es mi fuerte, pero daré unas pinceladas generales que sirvan para aclarar conceptos.

Ya aclaramos al principio que el software libre no siempre es software gratuito, pero ahora amplío el concepto un poco más: el software libre y gratuito no tiene porqué ser desarrollado por programadores que no cobran por ello. Vale, parece un trabalenguas pero es fácilmente explicable mediante un ejemplo cercano a todos nosotros y relativamente reciente: LinEx. Sobre el nacimiento de este sistema operativo GNU/Linux hablaré más en el apartado de la dimensión social del software libre, pero ahora vamos a lo que nos ocupa. LinEx ha sido desarrollado en España, concretamente en Extremadura. Es un sistema operativo libre y gratuito, pero los desarrolladores que han trabajado en él (bueno, no podemos hablar en términos absolutos, pero en general sí) han cobrado y han comido de esas líneas de código que han escrito. ¿Que cómo es eso posible? Pues porque se trata de proyectos financiados. En el caso concreto de LinEx, fue financiado por la Junta de Extremadura. ¿Perdieron ese dinero? No, para nada. La alternativa era pagar una millonada en licencias a Microsoft. Sale más barato, es un dinero invertido y no pagado, permite ceder también conocimiento a otras personas, y lo más importante (y enlazando con otro tema de este apartado) no manda ese capital a una empresa, sino que lo reinvierte en el propio país. Si tú pagas las licencias de Microsoft, estarás enriqueciendo a una compañía privada, que puede ser que luego invierta en investigación, que done dinero a ONG's o que compre millones de patitos de goma, pero no tendrá nada que ver contigo, al menos de forma directa. El financiar un proyecto con programadores de tu país o región evita esa “fuga” de capital y lo reinvierte en la propia economía sin debilitarla, pues ese dinero que cobran los programadores sigue en la economía local. Este tipo de proyectos financiados son cada vez más comúnes. Algún tipo de asociación se organiza para financiarse su propio sistema operativo o cualquier otro tipo de software libre, y luego cuando lo tiene lo ofrece libremente al resto de la comunidad para que pueda usarlo de base para nuevos proyectos (de hecho ahí tenemos GuadaLinEx basado en LinEx).

Así mismo hay casos parecidos en muchas Universidades, cada vez más en regiones del tercer mundo, y últimamente en regiones del primer mundo, lo cual está significando los primeros síntomas de que esto realmente está cambiando. Brasil es uno de los sitios con mayor conciencia de software libre. En europa, Alemania y España son ejemplos claros de esta tendencia, con la inclusión de software libre en la administración. Por cierto, y para orgullo personal, España es el país con un mayor desarrollo y peso en la comunidad de software libre europea.

Otro tema interesante son las compañías que desarrollan software libre y ganan dinero con ello. ¿Ganar dinero? Sí, pero no con la venta. Hoy día, y excepto Microsoft, la mayoría de las grandes compañías de software no obtienen sus mayores ingresos mediante la venta de licencias de su software, por lo que cada vez es más común la venta de software con licencia ilimitada, esto es, de uso en un número de máquinas no determinado por la licencia del fabricante. La mayor parte de los ingresos hoy en día se obtienen mediante

soporte de esos productos de software. Y no hablamos de compañías locales o pequeñas. Hablamos, por ejemplo, de IBM o Sun, que de pequeñas tienen muy poco. IBM y Sun participan activamente en el desarrollo de software libre, y no por eso dejan de ser empresas rentables. La idea es esta: yo creo un sistema operativo libre y lo ofrezco de varias formas. Tú puedes descargártelo de Internet o copiarlo a tu antojo, así como usarlo cómo y donde quieras. A cambio, tú no recibes más que el software y si quieres aprender a usarlo, búscate la vida. También puedes pagar por el software una licencia de usuario, con lo cual recibirás el software y documentación pertinente, pero si quieres saber más, tendrás que arreglártelas tú solo. O por último puedes comprar el software y pagarme un soporte técnico, con lo cual cualquier problema que tengas, te lo solucionaremos nosotros y tú podrás trabajar tranquilo. Como se puede ver, esta situación es ideal porque permite a un usuario con limitados recursos económicos disponer del software de forma gratuita a cambio de tener que aprender por su cuenta los pormenores (dicho sea de paso, en Internet hay documentación sobre cualquier cosa que se quiera aprender ;-P). Si tienes una empresa y deseas usar software libre (es mucho más seguro conocer el código del que depende tu empresa) puedes comprarlo y pagar a un equipo técnico de forma que jamás estarás descubierto, y podrás mantener tus equipos en funcionamiento siempre que quieras. Y el precio de ese soporte es muchísimo menor que lo que podría suponer el soporte de Microsoft. ¿Una idea alocada? Pues Red Hat Inc., SuSe, o Mandrake están viviendo de eso. Y funciona. ¿Sorprendidos?

Hay multitud de “teorías” económicas sobre el software libre, así como sobre la propiedad intelectual y las patentes, pero son teorías y la economía no es lo mío, así que sólo daré mi opinión al respecto: El conocimiento nos hace libres. Conocer lo que nos rodea nos da una perspectiva distinta y podemos comprender cómo funciona todo a nuestro alrededor. Así mismo, creo que nadie puede “apropiarse” del conocimiento. Puedes patentar un invento, una solución a un problema concreto, pero no soluciones abstractas. Creo firmemente que el software y el conocimiento han de ser libres. Antiguamente el soporte de la información tenía un precio, los libros no son gratuitos y todos lo sabemos; pero hoy en día el coste marginal de ese traspaso de información, gracias a Internet, es 0. Vale, de acuerdo, el ancho de banda tiene un coste, pero es despreciable respecto a esa información. Un simple algoritmo que pueda ocupar 100 Kb no tiene un coste marginal de transporte que merezca la pena tener en cuenta.

4- La dimensión social del Software Libre

Este es un tema que por desgracia mucha gente está abandonando al hablar de software libre, pero que en mi opinión es tan importante o más como el aspecto técnico. Al fin y al cabo el ser humano no es más que un animal social. ¿En qué términos defino el aspecto social del software libre? En los aspectos que permiten al software libre ayudar a la sociedad y hacer un bien colectivo desde algo que tradicionalmente ha sido exclusiva e irremisiblemente técnico.

En primer lugar el software libre puede ser un método de despegue social para una determinada comunidad de personas. Vuelvo a dar el ejemplo de LinEx, esta vez con más detalle y hablando sobre su nacimiento. En

Extremadura existía un grave problema con el sistema de enseñanza. Se hacía imprescindible una reforma educativa, y se decidió que esta pasara, en parte, por Internet, la ventana a la información que esto supone, e inevitablemente por los ordenadores. Evidentemente la reforma educativa contemplaba muchos más aspectos, pero para este artículo sólo nos interesan esta cuestión. Se compraron equipos informáticos de forma que cada dos alumnos existiera un ordenador a su disposición. Sólo les faltaba el sistema operativo y el software necesario. Como todo buen hijo de Windows, lo primero fue preguntar a Microsoft, que pedía cerca de 5.000 millones de pesetas por las licencias de todo el software requerido. Tras el desembolso en materia informático, ese nuevo desembolso era impensable, e irónicamente supondría mayor gasto que la compra de equipos informáticos. La solución se la dio la asociación HispaLinux: ellos gastaban un poco de dinero para financiar la creación de su propio sistema operativo, pero luego tendrían un software que podrían usar en cualquier situación, y no sólo ellos sino cualquier persona de la población o del mundo, que a su vez ayudarían a desarrollar el software. Una idea simple pero eficaz. Ha pasado el tiempo, y ahora LinEx está consolidado. A Microsoft le ha entrado el miedo, y celebró hace no demasiado tiempo (verano 2003) una reunión para “evitar que volviera a pasar lo de Extremadura”. Pero ya es tarde para Microsoft porque lo que en un principio parecía un proyecto de locos, ha resultado funcionar y con espléndidos resultados. Ahora Andalucía y Valencia se unirán en breve, así como otras administraciones del resto del mundo. Curiosamente hace unos meses Microsoft ofreció al Ministerio de Educación de España licencias gratuitas de su software para institutos, escuelas y universidades. Afortunadamente esta conciencia social ha hecho despertar a la gente, pues nos damos cuenta que aunque en la escuela tengamos Windows gratis, no es más que una trampa para que sigamos con el tedio y la fuerza de costumbre de utilizar su sistema operativo, y sea lo único que queramos usar. Así de simple, pero así lleva siendo desde hace tiempo, es lo que implica un monopolio de este tipo.

Brasil es otro claro ejemplo de esto mismo, pues hace ya tiempo que las universidades comenzaron a trabajar con software libre de forma casi exclusiva, desarrollando también su propio software. Con la crisis económica de Brasil y el nuevo gobierno de Lula, se dio un impulso mayor aún si cabe al software libre. Allí la situación es grave, la población muere de hambre. El gobierno de Brasil destina poco más de 400 millones de dólares para paliar el hambre de la población, y sin embargo la cifra total de licencias de software sólo para Microsoft que abandona el país es superior a los 1200 millones de dólares. ¿Es social o moralmente aceptable que se pierda (y digo pierda porque no es una reinversión, sino una entrega total a una corporación privada) el triple de dinero del destinado para paliar el hambre en licencias de software? Yo creo que no. ¿No sería más correcto autofinanciarse y reinvertir ese dinero en desarrollo propio de software, de forma que el gasto fuera infinitamente más bajo e invertir el resto del dinero en paliar las carencias sociales? Yo creo que sí. Y así se está haciendo. Es una idea que de simple, asusta. EL software libre está permitiendo que países en vías de desarrollo no pierdan la estela de desarrollo informático del resto del mundo a la vez que invierten su propio dinero en cubrir las necesidades de su población. No se trata de ningún movimiento político (hay quien acusa estas ideas de socialistas, comunistas, etc). A mí la política me la trae floja (con perdón)

como los que me conocen saben, pero me parece de sentido común ese rumbo que está tomando el software libre. Como ya dije, el conocimiento debe ser libre para que TODA la población pueda acceder a él. No nos damos cuenta de lo que perdíamos hasta que lo recuperamos, y al igual que en Extremadura un cambio educacional radical fue posible y en Brasil la erradicación del hambre está avanzando, esto mismo puede aplicarse al resto del mundo. No es necesario que se trate de un país del tercer mundo.

5- Enlaces de interés sobre Software Libre

- Open Source Initiative http://www.opensource.org/
- Free Software Foundation Europe http://www.fsfeurope.org/
- GNU http://www.gnu.org/
- Linux.org http://www.linux.org/
- Kernel.org http://www.kernel.org/
- Linux ISO http://www.linuxiso.org/
- FreeBSD http://www.freebsd.org/
- openBSD http://www.openbsd.org/
- Tux http://www.tux.cl/historia.php
- HispaLinux http://www.hispalinux.es/
- El Rincón de Linux ES http://www.linux-es.com/
- Debian GNU/Linux http://www.debian.org/
- Gentoo Linux http://www.gentoo.org/
- GNU/LinEx http://www.linex.org/
- Linux Mandrake http://www.linux-mandrake.com/es/
- Red Hat Linux http://www.redhat.com/
- SuSe Linux http://www.suse.com/index_us.html
- The Slackware Linux Project http://www.slackware.com/

6- Autoría, distribución y licencia de este documento

Este documento se distribuye en formato de texto maquetado para phpBB e Invision Power Board, así como en formato libre no propietario .sxw de OpenOffice.org, formato compatible .doc de Microsoft Word realizado bajo OpenOffice.org, y formato .pdf. Este artículo se ha realizado desde OpenOffice.org 1.10 rc4 bajo GNU/Linux.
Revisión 1.1 realizada desde OpenOffice.org 1.1 Final (rc5) bajo GNU/Linux.

Un saludo.

End Of File