Después de las Guerras de Software

Draft: 1.028, 31/01/2011
Words: 38326, Pages: 129

Keith Curtis

I am making this book available as a free digital download.

In general, I tried to get permission for the use of other's information. However, I have over 100 images and it was hard to figure out where to get permission for some of them. For those, I will claim permission under fair use ;-) I am happy to remove any content if its owner objects.

Every movement grows one person at a time.

Keith Curtis
keithcu@gmail.com
twitter: @keithccurtis

ISBN 978-0-578-01189-9

Table of Contents

Batalla del Software Libre

Algunas personas piensan que se requieren computadoras mucho más rápidas para la Inteligencia Artificial, así como nuevas ideas. Mi opinión es que las computadoras de hace 30 años eran lo suficientemente veloces si solo hubiéramos sabido como programarlas.

—John McCarthy, científico de la computación, 2004

Esta Computadora IBM 305 RAMAC, introducida en 1956, fue la primer computadora que contenía un disco duro (de 5 MB) en 24 platos giratorios gigantescos. Hoy en día usted puede obtener 1000 veces más memoria en algo del tamaño de un pulgar.

Dada la tecnología que está disponible actualmente, deberíamos tener automóviles que nos trasladen, de una manera absolutamente segura, mientras holgazaneamos en la parte trasera de este y bebemos champán. Todo lo que necesitamos es una cámara de vídeo en el techo, conectada a un PC, verdad? Todos tenemos el *hardware* necesario, y lo hemos tenido por años, pero aún no tenemos automóviles que se manejen automáticamente porque aún no tenemos el *software*. Este libro explica como podemos construir mejor software y de esta forma obtener todos nuestro propio chofer de alta tecnología.

La llave para un progreso tecnológico más rápido está en el uso extendido del software libre. El software libre contra el propietario (no libre) es similar a dividir la ciencia de la alquimia. Antes de haber ciencia, había alquimia, donde las personas ocultaban sus ideas debido a que aspiraban a una parte del mercado bajo el principio de convertir este avance en oro. Lo negativo de esa "estrategia" es que todo el mundo tenía que darse cuenta por si mismo que beber mercurio era muy mala idea.[1] El fin de la Edad Oscura llegó cuando el hombre comenzó a compartir sus avances en matemáticas y ciencias con otros en pro de su mejora. De hecho, una manera de mirar la historia es dividirla entre períodos de progreso y estancamiento.

Las computadoras son un avance comparado con la invención de la imprenta. Mientras las computadoras y el Internet han cambiado actualmente muchos aspectos en nuestras vidas, aún vivimos en la edad oscura de la computación debido a que el software propietario permanece como el modelo dominante. Uno podría decir que el alquimista más rico que ha existido jamás es mi anterior jefe, Bill Gates. (Larry Ellison, fundador de Oracle, y los cofundadores de Google, Sergey Brin y Larry Page están cercanos a este.)

Este libro discutirá acerca del Software Libre, pero la pregunta es sobre si la investigación científica y otros recursos de información tales como librerías deberían ser libres como se viene pidiendo desde hace tiempo. De hecho, la primera (fundada de forma privada) biblioteca en Estados Unidos fue creada por Benjamin Franklin en 1731, 45 años antes de que la nación misma fuera creada. El lema que se le dio a la biblioteca fue la frase en Latín: "Soportar los bienes comunes es divino." Benjamin Franklin entendía que compartir el conocimiento no tiene nada de negativo.

1 La versión digital de este libro tiene un número de palabras hiper-vinculadas destinadas a llevarte a referencias externas, como este vídeo del escritor Cory Doctorow en el Red Hat Submit.

El conocimiento existe de forma incremental en formato digital, por lo que construir nuevos y mejores modelos son un requerimiento en orden de mejorar el software. Las personas pueden intercambiar ideas solo cuando además comparten el software para mostrarlo y modificarlo. Solo el uso expandido del software libre permitirá a las personas trabajar juntos y lograr progresos. Los casos de estudios que examinamos en este libro nos demuestran que un sistema donde cualquiera pueda editar, compartir, y criticar el cuerpo del trabajo llevarán no solo a algo que funciona, sino que eventualmente a !lo mejor que el mundo pueda lograr! Una mejor cooperación entre nuestros científicos nos llevará a automóviles que se manejan de forma automática, robótica generalizada, inteligencia artificial, y progresos muchos más rápidos en la biología, la que depende en alto grado del software.

Uno de los siguientes capítulos nos describirá las libertades del software en más detalle, y las motivaciones para los programadores para que usen y escriban software libre, pero es importante aclarar aquí que el software libre generalmente significa que el código fuente se hace disponible para sus usuarios. El Internet Explorer de Microsoft no es libre ya que requiere una licencia de Windows, pero más importante aún, porque no puedes descargar su código fuente y saber como este funciona.

Hoy en día, el software propietario es considerado mas valioso que el libre debido a que sus dueños cobran por una caja negra, pero ese tipo de pensamiento es exactamente al revés. El software propietario es menos valioso ya que usted no puede aprender como este funciona, así como mejorarlo. El no puede hacerlo a usted mejor, ni usted mejor a el. Es una realidad que no todo el mundo ejercerá el derecho de leer y cambiar su software, así como no todo el mundo ejerce su derecho a una prensa libre, ¡pero esto no hace que la libertad sea menos valiosa!

Ejército del Software Libre

> Oficiales de justicia argumentan que el poder de Microsoft es impugnable debido a que los consumidores son tan dependientes de Windows. Alzando su voz, Gates exclamó, “Usted deme cualquier silla en la mesa: usando Linux o Java puedo desaparecer a Microsoft”
>
> —*World War 3.0: Microsoft and its enemies,* Ken Auletta

Glenn Reynolds, en su libro *Army of Davids* (Ejército de Davids), habla acerca de como ejércitos, como bloggers en pijamas, están cambiando el periodismo y otros aspectos de nuestras vidas. Este

libro se concentrará en el ejército del software libre, creado por Richard Stallman en 1985. El rango y extensión de este ejército consiste en programadores relacionados de forma distante en lo social y legal, quiénes viven en diferentes países, hablan distintas lenguas madres, que pueden además trabajar en compañías competidoras, o son voluntarios en su tiempo libre, con el fin de establecer su huella en la base de conocimiento del software mundial.

Sourceforge.net, el repositorio de software libre más grande que existe tiene registrado 1,900,000 desarrolladores hoy en día. Aún si dividimos por 50, ya que muchos trabajan solo parte del tiempo, nos quedamos con un ejército de 38,000, tres veces mayor que los equipos de desarrollo de Google y Microsoft unidos. Y SourceForge es solo una de las comunidades de software libre; la mayoría de los equipos más grandes usan sus propios servidores para administrar y organizar su proceso de desarrollo.

La pieza de software libre más importante es el sistema operativo Linux (pronunciado Lin-ex), nombrado en honor a su fundador Linus Torvalds, quién comenzó su codificación en la universidad. Mientras que Linux no es muy utilizado en las computadoras de escritorio hoy en día, el y otras piezas de software libre corren en el 60% de todos los sitios web, un número incremental de teléfonos celulares, y en el 75% de las super-computadoras más rápidas del mundo.

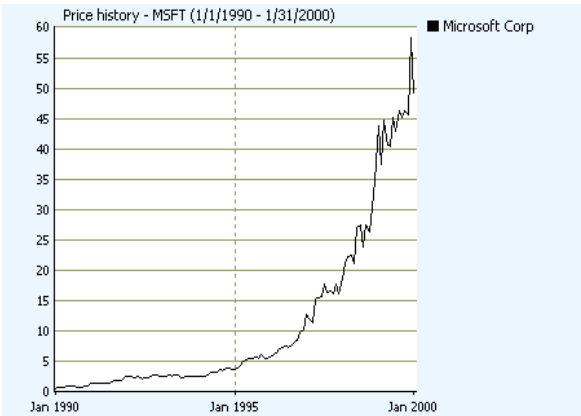
La super-computadora Blue Gene (basada en juego de palabras) de IBM corre un Linux ligero en cada uno de sus nodos, y un Linux completo en sus nodos de administración.

Por su parte, Microsoft a luchado ferozmente contra Linux y la dirección que este marca en el software libre pretendiendo que este es solo otro competidor propietario. Con $28 billones de dólares en efectivo, cuotas de mercado dominantes en Windows, Office e Internet Explorer, y un ejército de miles de programadores experimentados, Microsoft es un enfocado y paciente competidor.

Microsoft es la compañía más grande de software propietario, pero otros han adoptado su filosofía de acaparar todo el conocimiento, sin importar cuan irrelevante sea para ellos o útiles para otros. Google, el jugador dominante en las búsquedas de Internet, depende en gran medida del software libre y lo considera una parte importante de su éxito, pero ellos mantienen en secreto y protegen casi todo el software que *ellos* producen. Ellos son un agujero negro del software libre: la innovación entra pero nunca sale.[2]

Todo esto es perfectamente legal y ético, y el mercado libre le da a todo el mundo el derecho sin restricciones de innovar en cualquier forma, crear cualquier tipo de licencias, y cobrar cualquier cosa por un producto. Pero el software libre no es solo un competidor, es una forma distinta de crear software.

La comunidad del software libre a amenazado desde hace largo tiempo en dominar el mundo. El evangelista Eric Raymond le gritó a un VIP de Microsoft que el era su “peor pesadilla.” Esto fue a mediado de los 90, cuando el precio de las acciones de Microsoft hacía esto:



Precio de las acciones de Microsoft, 1990 – 2000

2 Desde que escribí esto, Google ha comenzado a liberar mas de su software, pero esto es lo menos interesante. Aún peor, en un número de casos ya existe una implementación libre de este. Google en el 2010 piensa que el software libre es solo apropiado para determinados escenarios, y no creen que hubiera sido mejor si hubieran comenzado escribiendo todo su software como libre. Hablaré más acerca de Google en un capítulo posterior.

Un amigo instaló Linux a mediados de los 90 pero se rindió con este debido a que su Backspace no funcionaba. El software libre a recorrido un largo camino desde aquello, alcanzando su masa crítica, si no el dominio del mercado. Este libro discutirá los desafíos técnicos que aún quedan por resolver para lograr dominación mundial, pero la inercia y la ignorancia son los obstáculos mayores.

Mientras que este libro presenta una visión del futuro, yo creo que hubiéramos podido contar con estos avances hace décadas. El éxito paradójico del software libre nos debe causar además que nos hagamos otras preguntas acerca de los derechos de copia, patentes y otros temas que este libro tratará.

iBio

Yo conocí a Bill Gates cuando tenía 20 años. El estaba en los alrededores de su casa del lago en Washington, con una cola dietética en mano (Diet Coke), y una mancha de ketchup en su camisa, la cual nadie tenía el coraje de decirle, y nos contestaba nuestras preguntas, en turnos, como un erudito. Como un interno universitario de verano, yo había planeado un potencial encuentro y me le acerqué con preguntas que me interesaban pero que serían prohibidas para los simples mortales sin conocimientos acerca del tema.[3]

3 Le pregunté acerca del rendimiento de el motor de almacenamiento de base de datos del Microsoft Exchange comparado a la que estaba dentro del Microsoft SQL Server, y sobre la nueva tecnología de clustering de NetWare llamada SST Nivel 3.

Sus respuestas me demostraron que el estaba entre los mejores expertos de software del planeta y me convencieron sería sabio comenzar mi carrera en Microsoft.

Como la carpintería, escribir software es un arte. Mientras usted puede leer libros acerca de lenguajes de programación y algoritmos, usted no puede aprender los incontables detalles de un arte en un libro. Usted debe trabajar con expertos en problemas reales. Antes del software libre, usted debía unirse a una compañía como Microsoft.

Yo me uno a Microsoft en 1993 cuando se caminaba a buen paso. Había recientemente liberado Windows 3.1 y Windows NT, estableciéndose en la ruta de más de una década de dominio del mercado de los sistemas operativos de PC, y de otros muchos mercados que fluyen de este. Yo trabajé como programador por 11 años en una variedad de grupos — en bases de datos, Windows, Office, MSN, movilidad, e investigación.

Un día solo me llegó — debía renunciar. No había grandes razones, solo un montón de pequeñas. Recientemente había lanzado la versión 1 del cliente y el servidor del reloj Microsoft Spot, y mientras este contenía tecnologías sofisticadas, no pensaba realmente que levantaría en el mercado. Yo había ganado cantidad de conocimientos, solo que estos pertenecían solamente al mundo de Microsoft. Estaba haciendo dinero suficiente, pero no tenía tiempo de

disfrutarlo. A pesar que mis jefes estaban felices conmigo, estaba perdiendo motivación de solo mantenerme haciendo la misma cosa que había estado haciendo en la última década. Cuando miré alrededor de la compañía vi grandes cantidades de bases de código antiguas y proyectos no rentables.

Como muchos de mis compañeros en Microsoft, solo tenía una idea muy vaga acerca del software libre cuando me fuí y de forma muy aleatoria decidí mirar aquella cosa que llamaban Linux. En Microsoft, tenía todo el software que quisiera de forma gratuita, y siempre pensé que el software libre siempre estaría detrás del propietario. Por 15 años hice una prioridad aprender tantos aspectos de las tecnologías de Microsoft como fuera posible, y mi oficina contenía filas de libros de todo tipo desde *Windows No Documentado (Undocumented Windows)* hasta *Dentro de SQL Server (Inside SQL Server)*. Corriendo Windows me sentía tan cómodo como Neo en la Matrix, si las balas y el cuero, por lo que cuando quise dar una mirada alrededor, lo hice medio forzado y no quería que este experimento se metiera con mi ambiente computacional.

Cada decisión técnica era importante para mi: ¿que versión de Linux debía probar? ¿Debería coger otra máquina o probar el dual-boot? ¿Podía realmente confiar en que compartiera el mismo disco duro con Windows? Obtuve algunos consejos y confianza en este por parte de un empleado de Microsoft que había recientemente probado Linux, y con eso, y la ayuda de Google, procedí a la instalación del Fedora Core 3 de Red Hat.

Mientras que solo había espacio para quedar impresionado por el propio Fedora, esto comenzaba a ser merecido solo por su proceso de instalación. Este contenía un instalador gráfico que recorría el proceso de instalación en su totalidad, este me re-dimensionó mi partición NTFS — lo que consideraba un milagro menor, estableció el dual-boot, y realmente arrancó el sistema, y me dejó navegar en la Web. No tenía ninguna pista de que hacer después, pero el solo hecho de *funcionar* me dijo más acerca del potencial de Linux que todo lo que había leído hasta ese momento. Usted no puede, por accidente, crear un avión que realmente vuele.

Con el tiempo, lo que más me impresionó acerca de Linux fue su poder. Venía con montones de aplicaciones: Firefox, OpenOffice, GIMP, Audacity, Mono, MySQL, y muchas más que debía descubrir. La interfaz de usuario (UI) era simple, rápida, limpia y configurable. Instalar el servidor web Apache solo me tomó unos segundos y me dio acceso al vasto mundo de PHP. Instalar el blog de WordPress

solo me tomó 15 minutos la primera vez, pero conocí luego cuando me volví mas ágil con este, que lo hubiera podido hacer en uno. Me di cuenta que mas allá de sus mal depurados controladores de dispositivos, una computadora con Windows es un chiste triste. A mediados del 2005, ¡estaba enamorado de las computadoras otra vez!.

Yo he gastado tres años en una paciente investigación en los temas claves de este libro, hablando con miles de programadores, asistiendo a varias conferencias, y leyendo código fuente, revistas, sitios web y libros. Este libro no es realmente tanto acerca de la muerte de Microsoft, sino de como el modelo de desarrollo propietario de Microsoft a pervertido e infectado la computación. Yo tengo cero rencores hacia Microsoft, solo que ahora pienso que ellos están fritos. Yo amé trabajar allí, aprendí en grandes cantidades, y disfruté del privilegio de trabajar con muchas mentes brillantes. Como muchas cosas en la vida, fue divertido mientras duró.

Glosario

Bit: Una muestra de información que puede tomar 2 valores: 1 y 0. Los bits son agrupados en 8 dentro de cada byte, existen caracteres de 2 bytes (Unicode), números de 4 bytes e imágenes con muchos.[1]

Digitalizar: Proceso de convertir algo en 1s y 0s. Una vez que algo se encuentra en un formato digital, puede ser manipulado infinitamente por un ordenador.

Software: Término general usado para describir una colección de programas de computación, procedimientos y documentación que ejecuta tareas en un ordenador.

Función: El bloque básico para la construcción de un software es una función, la cual es un pedazo discreto de código que acompaña una tarea:

```
int NumeroAlCuadrado (int n)
{
   return n * n;
}
```

Lenguaje de máquina: A nivel más bajo, un software es un montón de bits que representan una secuencia ordenada de instrucciones específicas del microprocesador para cambiar el estado del ordenador.

Lenguaje de alto nivel: Un lenguaje de programación que se parece más al idioma inglés.

Compilador: Software que (típicamente) convierte un lenguaje de alto nivel en un lenguaje de máquina.

Núcleo: El nivel más bajo de un sistema operativo que inicializa y administra el hardware.

Hardware: Interconexiones físicas y dispositivos requeridos para almacenar y ejecutar software.

Procesador: Hardware que ejecuta las instrucciones de los programadores.

Disco duro: Platos magnéticos hilados donde los bits son almacenados incluso después de que el ordenador es apagado.

Memoria: Pieza de hardware que brinda acceso rápido a bits de código y a información para el procesador. Un procesador sólo puede manipular información después de que él la ha cargado en memoria desde el disco duro o la red.

URL (Localizador de Recursos Uniforme): La ubicación textual de una página web, imagen, etc. en la Internet. Puedes asociar una URL a cualquier ordenador en el mundo que “entienda la Internet” y esto retornaría la misma cosa. (Se puede notar que prefieres una versión de la página en tu lenguaje nativo.) Una dirección de correo electrónico es también una URL. La única cosa que posee *todo* en Internet es una URL.

1 Al igual que en varios lugares de este libro, parte de este texto fue tomado de Wikipedia.

Wikipedia

Un buen amigo mío es profesor de Enseñanza Media en Bed-Stuy, Brooklyn – donde abundan los afroamericanos. Traten de imaginar esta aula; se trata de una gran cantidad de verdaderos estereotipos. Pero lo que NO encaja el estereotipo es que él empezó una wiki clase, y todos sus estudiantes contribuyen en la misma. En lugar de un desastre total, en vez de un abuso, el graffiti y el maltrato, se ha elevado el nivel de todos los estudiantes. Es un ambiente parejo: una vez que se vuelve interesante hacerlo bien, para *ser* sincero, el abuso y los problemas se secan casi completamente.

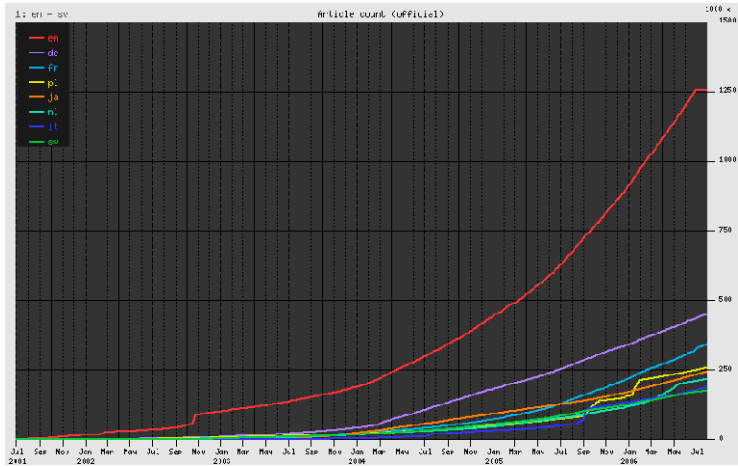
—Comentarista Slashdot.org

Mi escuela bloquea completamente el acceso a Wikipedia. Cuando pregunté por qué, la respuesta fue “cualquiera puede editarla.” ¿A diferencia del resto de la Internet, que está repleta de nada más que la más alta calidad, revisada por contenidos, escrita universalmente por los mejores expertos, seleccionados a mano de todo el mundo?

—Comentarista Slashdot.org

Uno de los grandes movimientos en mi vida entre las personas educadas es la necesidad de encomendarlos a la acción. La mayoría de la gente no está satisfecha con dar dinero; nosotros también sentimos que necesitamos trabajar. Por eso hay un enorme aumento en el número de personal no remunerado, voluntarios. Las necesidades no se irán. El negocio no va a tomar el relevo, y el gobierno no puede.

—Peter Drucker, padre de la administración moderna



Gráfica del número de entradas en las Wikipedias de varios lenguajes. Este crecimiento exponencial es una confirmación de la ley de Metalcafe: cuanto más usuarios haya de Wikipedia, mejor se pone, por eso, úsala más.

Enciclopedia Británica de 1913

En comparación con una enciclopedia de papel, una edición digital tiene ventajas significativas. La mayor es el costo, porque la impresión y el envío de un documento de 50 000 páginas representa un gasto enorme en la producción de una enciclopedia. El ámbito digital tiene otras ventajas significativas: el contenido puede ser actualizado constantemente y las funcionalidades multimedia pueden ser incorporadas. ¿Por qué leer de las fases de un motor de 4 tiempos de combustión interna cuando puedes ver uno en acción?

A mediados de los 1990s, Microsoft creó la Encarta, la primera enciclopedia digital basada en un CD-ROM. Los CDs fueron una evolución natural para Microsoft porque se estaba enviando su software cada vez mayor en un número cada vez más grande de discos floppy. (Windows NT 3.1, liberado en 1993, necesitó de 22 discos floppy. Los CDs se convirtieron rápidamente más rentables, ya que soportan 500 veces más información, y son más confiables y rápidos, Microsoft desempeño un rol importante en la inserción de las unidades de CD-ROM como estándar en los ordenadores.)

Mientras que los CDs soportan más información que los discos floppy y son un adelanto tecnológico importante, este desarrollo pronto fue eclipsado por el rival de la web. Los usuarios podían conectarse a una enciclopedia en constante actualización y de tamaño ilimitado desde cualquier ordenador sin necesidad de instalarla primero.

Desafortunadamente para Microsoft, el equipo de Encarta se demoró en adoptar la Internet porque ellos sintieron que parte de la riqueza de su enciclopedia se perdía en la web. Sin embargo, con imágenes , animaciones y texto, incluso la web en sus inicios era lo suficientemente buena y tenía importantes ventajas sobre una versión en CD-ROM. En el campo de la Internet, sólo necesitas una Wikipedia, aunque corriendo en cientos de servidores, para el mundo entero; incluso no tienes que preocuparte a cerca del costo de “hacer una copia” de una enciclopedia.

Sin embargo, el error más grande que el equipo de la Encarta cometió fue no darse cuenta de que la Internet podía introducir bucles de retroalimentación. Los usuarios de una enciclopedia de Internet pueden además convertirse en potenciadores de esta. Si tengo una pregunta sobre algo que he leído , o pienso que he encontrado un problema, puedo postear la pregunta o arreglar el problema y reportar lo que he llevado a cabo.

Discutiremos más adelante si la posibilidad de que cualquiera edite, potencie o adicione información afectará la calidad, pero es importante recordar que fue la creación de Internet lo que permite que personas en todas las esquinas del mundo trabajen juntas y aprendan unos de otros; una capacidad completamente nueva para el hombre.

De los posibles defectos, Wikipedia se convirtió más grande que la Enciclopedia Británica en sólo 2.5 años. La base de datos ahora contiene más de 15 veces más artículos, y ya es el mejor compendio de conocimiento humano jamás creado. Ninguna corporación invirtió millones de dólares en ingeniería o marketing; esto ocurrió aparentemente por sí misma. Incluso si alguno de esos artículos a cerca de los personajes de Star Trek está mal, muchos no: el artículo de Wikipedia sobre los nanotubos de carbono y muchos otros temas científicos está más detallado y más actualizado que el de la Enciclopedia Británica. La gran profundidad de Wikipedia es además una refutación de sustento o quizás la más grande crítica sobre el software libre y el contenido libre: que nadie va a trabajar en las cosas más arcanas y aburridas. La lección aquí es que las cosas diferentes son interesantes para personas diferentes.

Wikipedia es uno de los 10 sitios más populares en Internet, recibiendo 450 veces el tráfico de la Enciclopedia Británica, y con una colección de artículos que sigue creciendo a un ritmo exponencial.

En Wikipedia se ha avanzado, también ha agregado una colección multimedia, un diccionario, un compendio de citas, libros de texto, y un agregador de noticias — y es sólo el comienzo.

En algunos aspectos, el acceso a un motor de búsqueda podría parecer obviar la necesidad de una enciclopedia. Pero mientras que los motores de búsqueda proporcionan un índice de palabras claves para la Internet, no reemplazan la importancia de una enciclopedia: un comprensivo, coherente y neutral compendio de conocimiento humano.

Imagina que quisiste investigar un tema como el poder nuclear. ¿Dónde irías para obtener una opinión imparcial: al gobierno? ¿Greenpeace? ¿CNN? Algunas escuelas han prohibido la Wikipedia, pero las fuentes secundarias han sido rechazadas. Aún así, el texto del artículo y enlaces a fuentes primarias puede ser un lugar útil para iniciar la investigación sobre un tema.

Mientras que Wikipedia es un recurso poderoso, lo que es más impresionante es que está construida con el mismo excedente de energía intelectual que otros gastan en crucigramas o sudoku. Wikipedia proporciona una salida adicional para la energía de las personas, y algo mucho más grande que cualquier persona, o incluso que una compañía, pudiera lograr.

Un elemento determinante en el éxito de Wikipedia es que sus fundadores crearon una comunidad en la que las personas disfrutaron trabajar, y este factor de disfrute atrajo incluso a más personas. Esto es algo difícil de lograr, y comienza con una visión inspirativa.

No hay ninguna gran corporación multinacional por detrás de Wikipedia. No hubo CEO que golpeara la mesa y dijera que quería crear la enciclopedia más grande que haya existido. Su presupuesto anual es de $5,000,000, la mayoría es usado para financiar el hardware utilizado, ancho de banda y el salario del muy pequeño equipo de operaciones de seis personas que mantienen los pocos cientos de servidores corriendo. Quizás aún no hayas editado la Wikipedia, pero millones de otros miembros registrados, y usuarios no registrados, han creado artículos y la han mejorado a través de los años. Yo he hecho unos pocos arreglos — ¡es muy fácil!

Algunos pueden preguntarse sobre la susceptibilidad a las imprecisiones y al vandalismo de algo tan ampliamente colaborativo como Wikipedia. Afortunadamente, este graffiti digital no amenaza con arruinar las cosas por dos razones importantes: responsabilidad y orgullo. Lo bueno de estos esfuerzos colaborativos es que ellos proveen una forma de apreciar la importancia del prójimo.

Cada cambio hecho a la enciclopedia es registrado permanentemente y fichado públicamente en un mecanismo de control de versiones similar al usado en el desarrollo de software; de hecho, no hay cambios irreversible. El graffiti improbable, el cual puede tomar horas para limpiar, arreglar cambios no deseados o bloquear usuarios no deseados toma meros segundos lo cual es un gran desaliento para otros.

Últimamente, parte de creer en la viabilidad de una enciclopedia libre requiere confiar fundamentalmente en la bondad de la humanidad. Uno debe confiar en que la cantidad de personas en el mundo que disfrutan de hacer una contribución positiva a un producto es muy superior a esos que disfrutan unos pocos segundos de orgullo perverso en cosas temporarias. Además, con millones de usuarios registrados, hay una garantía virtual de que los problemas serán notados.

El vandalismo obvio es fácilmente encontrado y eliminado, pero hay formas más sutiles de vandalismo que son mucho más difíciles de detectar. De hecho, ¿ quién puede decir sí o no cualquier edición es correcta?

Wikipedia ha aislado su producto de imprecisiones implementando tres normas de contenido:

1. **Ninguna investigación original**: Los artículos deben referenciar a fuentes públicas y confiables. El marco de "confiable" es debatible, pero en la práctica, este no es un obstáculo significativo.
2. **Punto de vista neutral**: Un artículo debe de manera justa y sin sesgos, representar todos los puntos de vista importantes que han sido publicados por fuentes confiables.
3. **Verificabilidad**: El marco para la inclusión en Wikipedia es verificable. Verificable significa que un lector debe ser capaz de determinar si el material añadido a Wikipedia ya ha sido publicado por una fuente confiable.

Que la comunidad acepte estos conceptos es clave para el éxito de Wikipedia.

Por hacer de estas normas parte integral de la cultura, Wikipedia creó algo no necesariamente preciso, pero desde recursos reputables y verificables es lo que la hace lo suficientemente buena para que otras personas decidieran que es un mérito leer y contribuir.

Han existido estudios objetivos que demuestran que Wikipedia tiene una alta calidad, comparable a la Enciclopedia Británica. En general, su más grande reto está en los artículos políticos donde las

emociones son altas, y la mayoría de los contribuidores se describirían ellos mismos como liberales. Esto es muy complicado porque muchos factores se disputan: algunos científicos dicen que el Calentamiento Global es un peligro inminente para la humanidad, mientras otros sostienen que esto es un engaño, y Wikipedia no puede resolver esta contradicción entre fuentes públicas y confiables.

Aún para los cínicos que creen que los vándalos pueden todavía vencer, consideran que desde su creación en enero del 2001, Wikipedia ha permanecido mucho más como una enciclopedia que como un experimento tecnológico y social auto organizado. Como Wikipedia evoluciona, las herramientas para detectar y eliminar el vandalismo se están creando, y para etiquetar artículos que no concuerdan con los lineamientos de estilos. Los artículos a veces tienen varias advertencias sobre como un trabajo actual se encuentra en progreso, esta es una advertencia importante. Cada página también tiene una página de discusión donde los puntos discordantes son debatidos antes que el contenido sea por sí mismo actualizado.

Brevemente, Wikipedia es una relación en evolución entre las personas y su software. Por ejemplo, ¿ se debería permitir que los usuarios anónimos editaran ? Muchos creen que ellos no deberían poder hacerlo porque la anonimidad decrementa la responsabilidad. Esta es una discusión continua.

Wikipedia se lee libremente, y un estudio reveló que esta pudiera generar hasta $100 millones por años en ingresos por publicidad. Un día, ellos pudieran elegir, y pudieran usar ese dinero de muchas maneras: desde adquirir contenido propietario como mapas, documentos legales, y plantillas de documentos, y hacerlas libres, para contratar empleados a cargo en áreas insuficiente financiadas por la comunidad.

Eric Raymond, en su libro The Cathedral and the Bazaar, compara el modelo de desarrollo de software libre con una venta benéfica – una conglomeración de de entradas e ideas desorganizadas. Esa es, sin embargo, una imagen insatisfactoria, porque se sugiere algo primitivo y desorganizado. Las catedrales toman cientos de años en construir, pero en menos de 10 años, Wikipedia ha producido un producto más largo y comprensivo que los competidores existentes con anterioridad. Es mejor pensar de este producto de software libre como una catedral muy pulimentada en sus primeros años de desarrollo.

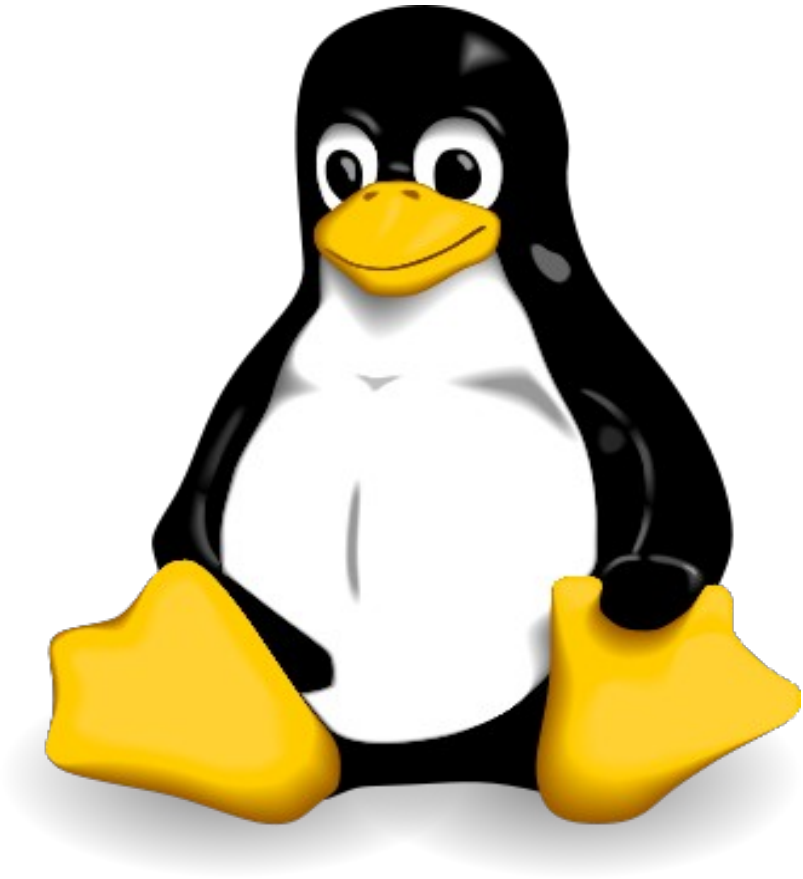
¿Qué más pueden construir las comunidades independientes y altamente cooperativas de software libre? La respuesta es un

número de cosas cercano al infinito. Específicamente, una gran cantidad de software que es tan libre como Wikipedia y utiliza cero código de Microsoft. El software utilizado para hacer funcionar Wikipedia es una idea de último momento para muchos, pero es una parte significativa de la tecnología. Mientras Wikipedia y su software no crearán una gran mella en las ganancias de Microsoft, el núcleo de Linux sí que es una amenaza mortal.

Linux

Realmente, no pretendo destruir a Microsoft. Eso sólo sería un efecto colateral completamente involuntario.

—Linus Torvalds, 2003



La mascota de Linux, Tux, creada por Larry Ewing

El núcleo de un sistema operativo (SO) es el sistema nervioso central de un ordenador. Es la primera pieza de software que ejecuta el equipo, y que gestiona y media en el acceso al hardware. Cada pieza de hardware necesita el correspondiente controlador de dispositivo del núcleo, y se necesita que *todos* los drivers funcionen para poder ejecutar *cualquier* aplicación. El núcleo es el centro de gravedad de una comunidad de software, y la batalla entre el software libre y Windows es en su nivel más bajo una batalla entre los núcleos de Linux y Windows. Microsoft ha dicho que ha

apostado la compañía en Windows, y esto no es un eufemismo! Si el núcleo de Windows es derrotado por Linux, tanto Windows como Microsoft pierden.[1]

El núcleo Linux no es popular en computadoras de escritorio todavía, pero es ampliamente utilizado en servidores y dispositivos empotrados, ya que admite miles de dispositivos y es confiable, limpio y rápido. Esas cualidades son aún más impresionantes si se considera su tamaño: la impresión de las 8.000.000 de líneas de código del núcleo Linux daría lugar a una pila de papel de 30 pies de altura! El núcleo Linux representa 4.000 años-hombre de ingeniería y de 80 empresas diferentes, y 3.000 programadores han contribuido a Linux a través del último par de años.

Esta pila de código de 30 pies de altura es sólo el núcleo básico. Si se incluye un reproductor multimedia, navegador web, procesador de textos, etc, la cantidad de software libre en un equipo con Linux podrían alcanzar 10 veces el núcleo, que requieren 40.000 años-hombre y una pila impresa tan alta como un edificio de 30 pisos.

Este hombre de 40 milenios incluso ignora el trabajo de los usuarios informando fallos, escribiendo documentación, creando obras de arte, traduciendo cadenas, y realizando otras tareas ajenas a la codificación. La pila resultante de software libre basado en Linux es un esfuerzo que es comparable en complejidad al transbordador espacial. Podemos discutir sobre si hay motivos para escribir software libre, pero no podemos discutir sobre que ya existe!

Una de las principales razones por las que me uní a Microsoft era que creía que su núcleo Windows NT (New Technology), que todavía está vivo en Windows Vista hoy en día, iba a dominar el cerebro de las computadoras y, eventualmente, los robots. Uno de los golpes más grandes de Bill Gates fue el reconocimiento de que el núcleo original de Microsoft DOS, la fuente de la mayor parte de sus beneficios, y que se convirtió en el núcleo de Windows 9x, no era un esfuerzo de ingeniería digna de mención. En 1988, Gates contrató a David Cutler de Digital Equipment Corporation, un veterano de diez sistemas operativos, para diseñar el producto y dirigir el equipo que construiría el núcleo de Windows NT, que fue lanzado cuando me incorporé en 1993.

1 Aunque cloud computing, el movimiento de un número cada vez mayor de aplicaciones y servicios prestados a través de Internet, es uno de los temas candentes de hoy, no tiene relación con la guerra que está pasando entre Windows versus Macintosh versus Linux. Incluso en un futuro en el que aplicaciones como el tratamiento de textos se realicen a través de Internet, todavía necesitamos un núcleo, un navegador Web, un reproductor de medios, y así sucesivamente.

El núcleo que Cutler y su equipo desarrollaron se veía así:

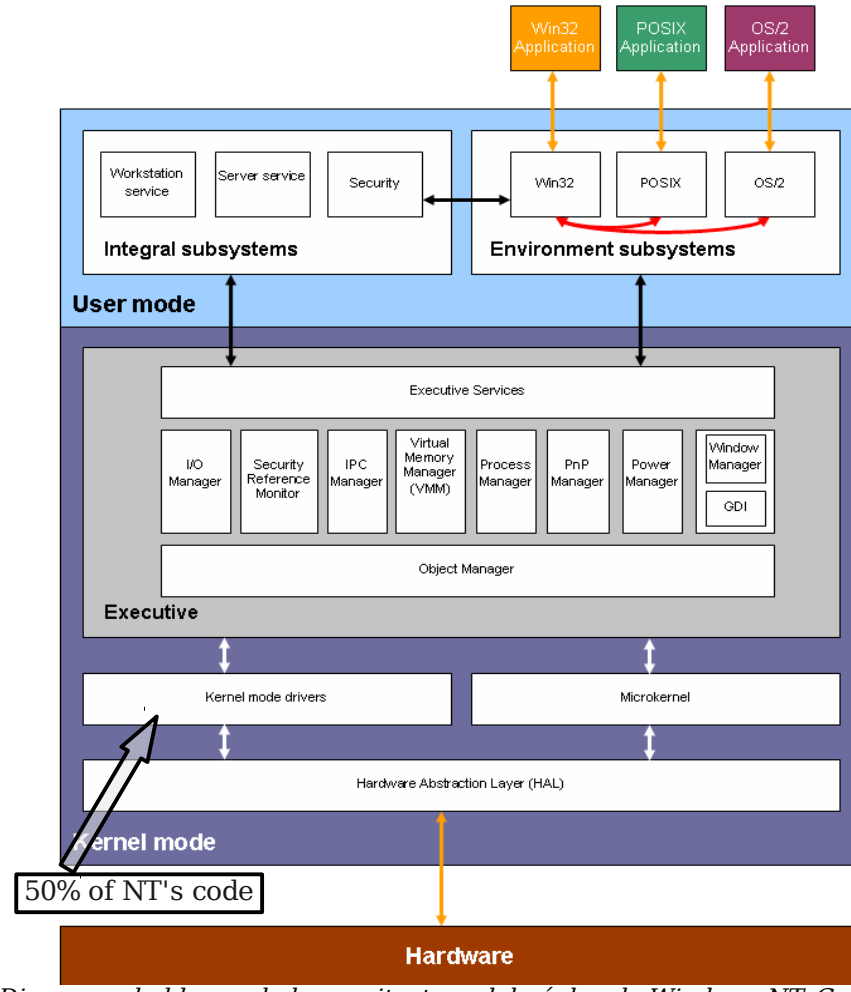


Diagrama de bloque de la arquitectura del núcleo de Windows NT. Cutler tenía un felpudo de Windows 95 fuera de su oficina; se le animaba a la gente a limpiar completamente sus pies antes de entrar.

Por desgracia para Microsoft, el núcleo original sobrevivió a través de Windows 95, Windows 98, y Windows Me. (Microsoft también tenía Windows CE, un núcleo pequeño para dispositivos empotrados. Microsoft tenía tres núcleos separados durante la mayor parte de mi estadía, mientras que el mismo núcleo Linux se utiliza en dispositivos pequeños y grandes.)

Windows se ha convertido en algo popular para servidores y dispositivos, pero nunca logró el mismo dominio que en el PC de escritorio. Tal vez la razón principal es que su código no estaba

disponible para que otros lo puedan ampliar y mejorar. El núcleo Linux despegó porque hay gente de todo el mundo, desde Sony a Cray, que lo ha ajustado para conseguir que se ejecute en su hardware. Si Windows NT hubiera sido libre desde el principio, no habría habido ninguna razón para crear Linux. Sin embargo, ahora que existe el núcleo Linux libre y poderoso, no hay ya ninguna razón, además de la inercia, para utilizar un núcleo privativo.

Hay una serie de razones para la superioridad del núcleo Linux. Pero primero, quiero describir el proceso de desarrollo de software. Cuando usted entienda cómo se construye el núcleo Linux, sus logros técnicos le parecerán más impresionantes y totalmente lógicos.

Desarrollo Distribuido

> En Linux, rechazamos mucho código, y esa es la única manera de crear un núcleo de calidad. Es un poco como la selección evolutiva: increíblemente derrochador e increíblemente eficiente, al mismo tiempo.
>
> —Ingo Molnar, desarrollador de núcleo Linux

Una parte de la lista de oradores para el Linux Kernel Symposium 2006 donde el autor asistió. El desarrollo del núcleo Linux es un esfuerzo distribuido, lo que mejora considerablemente su perspectiva.

Cada libro de gestión del siglo 20 que he leído asume que los miembros del equipo de trabajo están en el mismo edificio y hablan el mismo idioma. La cultura corporativa de Microsoft se basó en la teoría de que el desarrollo de software era un esfuerzo de colaboración que debe ser centralizado para que las personas puedan trabajar juntas. Como resultado, la mayoría de los "Microsofties", especialmente los programadores, se establecían en Redmond, porque era donde se encontraban todos los otros ingenieros.

Microsoft tenía un modelo de desarrollo muy abierto dentro de la empresa: los desarrolladores periódicamente cambiaban de equipo, colaboraban juntos libremente en código inédito, y se reunían en grupos de discusión por e-mail con ingenieros trabajando en un producto similar. Estos recursos de colaboración son algunos de los muchos que no están disponibles para los de afuera.

Un programa de software es básicamente un grupo de archivos de texto legible para humanos, compilado en un lenguaje binario para una máquina específica. Cuando sólo una persona está trabajando en el código base, las cosas son fáciles de manejar, pero cuando hay varias personas, se necesitan tres herramientas:

1. Un sistema de control de código fuente que registre todos los cambios, análogo a la pestaña Historial que existe en la parte superior de cada página de Wikipedia. Esto proporciona muchas capacidades, tales como volver atras para buscar cuándo y por qué se realizó un cambio de código.
2. Un sistema de seguimiento de errores o incidentes mantiene una lista de tareas pendientes. Esto mantiene un seguimiento de los problemas con el tiempo, y a medida que la gente sale.
3. Un mecanismo de comunicación (en persona, correo electrónico, chat o foros), donde los programadores pueden hablar acerca de cómo implementar las características, y trabajar juntos para solucionar problemas.

Equipos más grandes han desarrollado procesos más formales para, por ejemplo, llegar a un consenso sobre el permiso que se da a alguien para hacer cambios al código fuente, pero el proceso básico de desarrollo en el mundo del software libre difiere poco de lo que ocurre en Microsoft.

La Internet, que nació cuando Microsoft era una empresa madura, ha cambiado innumerables aspectos de nuestra vida, incluyendo la forma en que el software puede ser desarrollado. Sin Internet, el software libre no podría existir porque los desarrolladores no serían capaces de trabajar juntos. (Microsoft utiliza Internet para el

desarrollo en otros lugares, aunque todavía es principalmente en Redmond.) Microsoft creció antes del nacimiento de la web, y por lo tanto aún tiene que adoptar este proceso distribuido. Por ejemplo, las bases de datos de errores de Microsoft no están disponibles en Internet. Microsoft no está obteniendo el máximo provecho de los conocimientos adquiridos por sus usuarios, ya que no tiene muchos ciclos de retroalimentación.

Linux ha logrado enormes ganancias en contra de Windows, aun cuando, desde la perspectiva de un administrador del siglo 20, la organización del núcleo de Linux es un ejemplo del peor de los casos para la construcción de una productiva y próspera organización. ¿Dónde están los ejercicios de creación de equipos? Los retiros de planificación de tres años? Debe ser increíble que Linux produzca algo, y menos aún domine el negocio de superordenadores.

Linux tiene éxito, a pesar de los costos de trabajar a distancia por varias razones. En primer lugar, la distancia entre los programadores fuerza a las personas a formalizar las cosas, lo que ayuda a la calidad: las personas se reúnen en conferencias como el Simposio del Kernel de Linux y presentan documentos y obtienen el consenso y la opinión sobre las ideas de una amplia variedad de personas. Además, la capacidad de lograr que cualquier persona de todo el mundo contribuya es un mayor beneficio que el costo de reunir a la gente. No se podría crear un edificio lo suficientemente grande como para acoger a los millones de colaboradores de Wikipedia, por no hablar de los miles de contribuyentes del kernel de Linux.

La frase de Ingo Molnar al comienzo parece contradictoria a primera vista, pero no lo es. Linux recibe un montón de ideas diferentes, y muchas son rechazadas, pero las que quedan son las mejores de todas las ideas. La Internet nos permite evolucionar rápidamente hacia soluciones óptimas con bucles de realimentación de discusiones, pruebas e informes de error. (Las pruebas son muy importantes porque nos dan un número objetivo. Si queremos ver si el nuevo código de caché de disco es más rápido, podemos compilar el kernel, que es una tarea de disco muy intensa, y tomar el tiempo como resultado).

En el movimiento de software libre, las batallas no son entre imperios, sino más bien entre ingenieros luchando sobre detalles técnicos - los VIPs cargados de testosterona son irrelevantes. La comunidad del núcleo Linux ha llevado la idea de una meritocracia al siguiente nivel. Todos los cambios en el kernel oficial de Linux tienen que pasar por Linus, y su lugarteniente, Andrew Morton, y

luego al mantenedor del subsistema correspondiente - pero primero, la modificación propuesta tiene que pasar por todos los demás! Todos los cambios de Linux están en una lista de correo, donde cualquiera puede hacer comentarios y dar opiniones. Linus escribió:

> Los colaboradores para cualquier proyecto son auto-seleccionados. Alguien ha señalado que las contribuciones son recibidas no de una muestra aleatoria, sino de personas que se interesan lo suficiente como para utilizar el software, aprender acerca de cómo funciona, intentar encontrar soluciones a los problemas que encuentran, y en realidad producir una solución aparentemente razonable. Cualquier persona que pasa todos estos filtros es muy probable que tengan algo útil que aportar.

El trabajo principal de Linus es proporcionar experiencia técnica. En una ocasión dijo que su trabajo era mantener fuera mal código, y que eso podría ser suficiente. Dejen que los variados usuarios de Linux lo lleven a lugares nuevos, mientras él se asegurará de que nadie esté arruinando el código existente en el camino.

A Linus se le preguntó si la naturaleza sin pulir de un gran grupo de programadores, con antecedentes dispares, creó una situación tan triste que le dieran ganas de volver a trabajar en privado, y dijo:

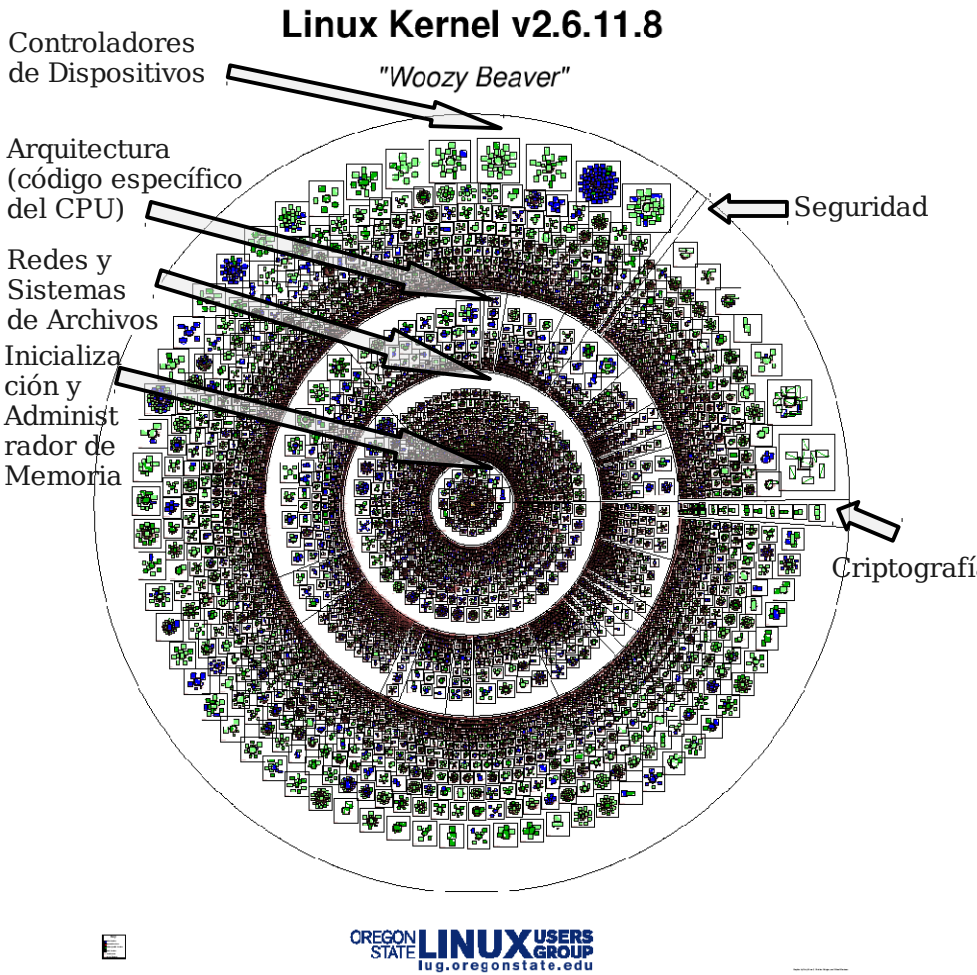
> En realidad me gusta discutir (a veces un poco demasiado), así que el ocasional "flame-fest" realmente no hace nada más que estimularme.
>
> Al mismo tiempo, soy realmente muy bueno para simplemente "soltarlo", una vez que he argumentado lo suficiente y me aburro con el argumento. Parte de eso es también tener que admitir en ocasiones que simplemente te has equivocado, y tener la capacidad de enviar un "mea culpa" por e-mail diciéndolo.
>
> Tiendo a interesarme mucho más en mejorar el modelo general de desarrollo que sobre los detalles de algunos de los subsistemas en particular. De modo que eso tiende a hacer más fácil para mí el "soltar." Voy a exponer mi opinión, pero aunque estoy convencido de que tengo razón, si no estoy realmente dispuesto a escribir el código, al final estaré feliz de ser sobrepasado por las personas que escriben el código.
>
> Esta es, obviamente, en gran medida, una cuestión de personalidad.
>
> Hay cosas en las que tiendo a preocuparme, y que pueden ser muy dolorosas, pero son bastante raras. El ejemplo clásico es el antiguo argumento de que "Linus no escala", que terminó siendo el caso en el que tuve que cambiar fundamentalmente las herramientas que uso y cómo trabajo. Y eso fue mucho más doloroso que preocuparse por el código real.

Superioridad del Núcleo de Linux

He aquí las razones por las cuales el núcleo de Linux es superior al de Windows.

1. Código Refactorizado (Confiabilidad)

He aquí un diagrama del núcleo de Linux:



Capas de la "cebolla" del núcleo de Linux. El núcleo de Linux es 50% controladores de dispositivos y 25% código específico de CPU. Las dos capas interiores son muy genéricos.

Pueden darse cuenta que está construido como una cebolla y está comprendido de varios componentes discretos. La capa exterior del diagrama se compone de controladores de dispositivos (drivers), el cuál es el 50% del código, y más del 75% de este es código especí-

fico de hardware. El diagrama del núcleo del Microsoft Windows NT mostrado páginas atrás, pone todos los drivers en una pequeña caja el la esquina inferior izquierda, ilustrando la diferencia entre la teoría y la realidad. De hecho, si Microsoft hubiera dibujado los drivers en modo núcleo como el 50% del núcleo en el diagrama del Windows NT, ellos hubieran entendido como el núcleo está compuesto mayoritariamente de código específico de hardware, y reconsiderado si este era un negocio en el que querrían meterse.

La refactorización (suavizar, refinar, simplificar, pulir) es realizada continuamente en Linux. Si muchos drivers contienen tareas similares, la lógica duplicada puede ser extraída y puesta en nuevo subsistema que pueda ser usado por todos los drivers. En muchos casos, no está claro hasta que una gran cantidad de código es escrito, si este nuevo subsistema es viable. Existen un número de componentes en el núcleo de Linux que han evolucionado a partir de código duplicado en varios lugares de este. Este flexible pero práctico acercamiento al escribir software ha llevado a Linus Torvalds a describir Linux como “Evolución, no Diseño Inteligente”.

Uno podría argumentar que evolución es signo de mal diseño, pero la evolución en Linux solo ocurre cuando existe una necesidad desconocida por el software actual. Linux inicialmente solo soportaba el procesador de Intel 80386 porque era el que poseía Linus en ese momento. Linux evoluciona, por el trabajo de muchos programadores, para soportar procesadores adicionales — más que Windows, y más que ningún otro sistema operativo halla soportado jamás.

He ahí además un ciclo genial: mientras más código es refactorizado, menor es la probabilidad que cambios en el código cause regresiones; mientras más código no causa regresiones, más código puede ser refactorizado. Usted puede pensar acerca de este ciclo de dos formas diferentes: código limpio nos llevará código aun más limpio, y a código más limpio, más fácil sera evolucionar para el sistema, manteniéndose estable. Andrew Morton afirmó que la base de código de Linux mejora de forma estable, aún cuando se ha triplicado en tamaño.

Greg Kroah-Hartman, encargado de mantener el subsistema de USB en Linux, me dijo que los diseños del hardware del USB han evolucionado de la versión original 1.0 a la 1.1 a la 1.2 en la última década, los controladores de dispositivos y la arquitectura del núcleo han cambiado considerablemente también. Ya que los drivers conviven dentro del núcleo, cuando la arquitectura es alterada para soportar los nuevos requerimientos de hardware, los drivers pueden ser ajustados al mismo tiempo.

Microsoft no tiene un solo árbol con todos los controladores de dispositivos. Debido a que muchas compañías de hardware tienen sus propios drivers flotando en los alrededores, Microsoft está obligado a mantener la vieja arquitectura para que así el código antiguo pueda seguir corriendo. Esto incrementa el tamaño y la complejidad del núcleo de Windows, ralentiza su desarrollo, y en algunos casos revela algunos errores o fallas en el diseño que a veces ni pueden corregirse. Esta compatibilidad con código antiguo es una de las mayores razones por la cuál Windows toma años en ser lanzado. El problema no solo existe en la capa de los drivers, sino en todo su software. Cuando el código no está disponible libremente y en un solo lugar, se le hace difícil evolucionar. Microsoft ha acumulado tanto equipaje que podría convertirse en una aerolínea.

Mientras que la lógica *interna* de Linux ha evolucionado bastante en los últimos diez años, las interfaces *externas* de este han permanecido constantes. La llave para interfaces estables es incorporar las abstracciones indicadas. Una de las mejores abstracciones de Linux adoptó de Unix es la abstracción de ficheros. En orden de realizar casi cualquier función en una computadora con Linux, desde leer una página web en un sitio remoto hasta descargar una foto de una cámara digital, es necesario simplemente usar los comandos estándar de archivos: open (abrir) y close (cerrar), read (leer) y write (escribir).

En mi computadora, para leer la temperatura del CPU, solo necesito abrir el archivo de texto (virtual) "**`/proc/acpi/thermal_zone/THM0/temperature`**" y los datos solicitados se encuentran dentro:[2]

```
temperature:             49 C
```

En esencia, el núcleo de Linux es un manojo de controladores de dispositivos que se comunican con el hardware y se revelan a ellos

2 Esto pudo haber sido expresado en XML, pero ya que existe un código común que lee estos valores y los proporciona a las aplicaciones, y porque cada archivo contiene un solo valor, este problema no es muy significativo; ya que la información de configuración del núcleo nunca será parte de la web.

mismos como un sistema de archivos. Así como nuevas características, problemas de seguridad, requerimientos de hardware y escenarios que confronta el núcleo de Linux, el diseño interno evoluciona y mejora, pero la abstracción del sistema de archivos permite al código exterior al núcleo permanecer sin cambios durante mayores períodos de tiempo.

He aquí un ejemplo aleatorio del log de cambios del núcleo de Linux 2.6.14. Como se puede observar, está lleno trabajos de limpieza y corrección de errores de todos tipos.

```
spinlock consolidation
fix numa caused compile warnings
ntfs build fix
i8042 - use kzalloc instead of kcalloc
clean up whitespace and formatting in drivers/char/keyboard.c
s3c2410_wdt.c-state_warning.patch
[SCSI] Fix SCSI module removal/device add race
[SCSI] qla2xxx: use wwn_to_u64() transport helper
[SPARC64]: Fix mask formation in tomatillo_wsync_handler()
[ARCNET]: Fix return value from arcnet_send_packet().
```

Muchos de los cambios en el código del núcleo de Linux son de pulido y limpieza. Código limpio es mas confiable y mantenible, y refleja el orgullo de la comunidad del software libre.

Si usted mira los cambios en el código requeridos para corregir un error, en la gran mayoría de los casos todo lo que se necesita es una revisión de algunas líneas de código en un pequeño número de ficheros. Una guía general que tiene Linux para la corrección de errores es esta: si tu miras el cambio realizado en el código y no puedes probarte a ti mismo que este cambio corrige el problema, entonces quizás el código modificado es confuso, y esta corrección no debe ser añadida cercana al fin de un ciclo de lanzamiento.

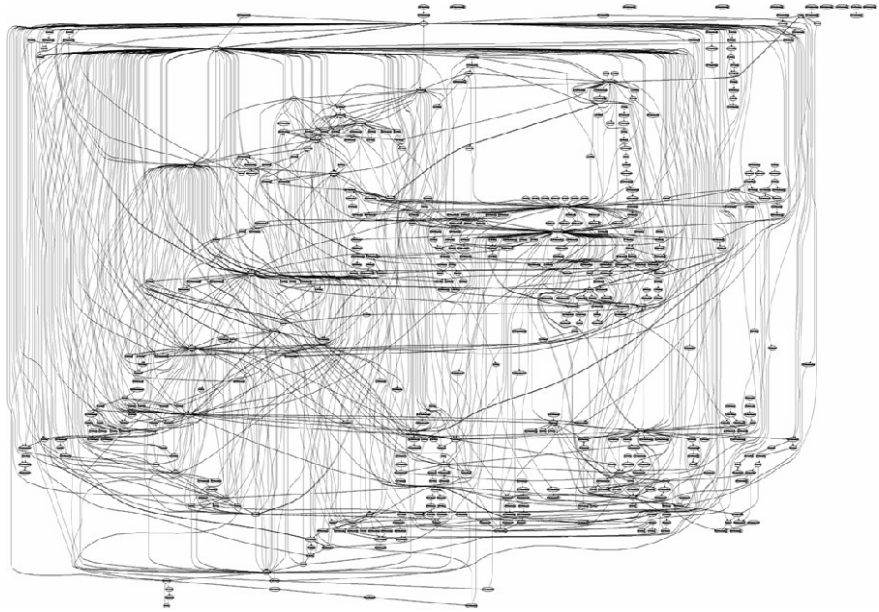
Según investigadores de la Universidad de Standord, el núcleo de Linux tiene .17 errores por cada 1,000 líneas de código, 150 veces menos que la media en el software comercial que contiene de 20 a 30 errores por cada 1,000 líneas.[3] Las bases de datos de errores de Microsoft no están disponibles en Internet, por lo que es imposible

3 Estos estudios tienen un valor limitado ya que sus herramientas generalmente analizan solo algunos tipos de errores de codificación. Como ellos hacen las noticias de IT (Tecnologías de la Información), y corrigen rápidamente debido a la publicidad, esto le que le quita al estudio su significado. Sin embargo, estas herramientas permiten realizar comparaciones entre bases de código. Yo creo que el mejor análisis acerca del número de errores en Linux es que de 1,400 errores en su base de datos, para 8.2 millones de líneas de código hacen .17 errores por cada 1,000 líneas de código. Este es un número pequeño, a pesar de que podría ser otras 100 veces más pequeño. He aquí un vínculo a los errores activos en el núcleo de Linux: http://tinyurl.com/LinuxBugs.

realizar comparaciones, pero aún si Linux no fuera más confiable en la actualidad, está predispuesto a serlo, ya que su código es más simple, correctamente escrito y localizado completamente en el mismo lugar.

Dentro de la comunidad del software libre, diferentes equipos son entidades dispares, así que la idea de mover código arbitrariamente de una parte del sistema a otra no puede sencillamente ocurrir. Dentro de Microsoft no existen fronteras, por lo que el código se mueve en los alrededores por ganancias en el rendimiento a corto plazo al costo de complejidad extra.

He aquí un grafo con todas las llamadas a funciones que el sistema operativo requiere para retornar un simple pedido web. Estas imágenes demuestran la diferencia visual entre la complejidad que existe entre el software libre y el propietario.



Grafo de las llamadas al sistema en el servidor web propietario de Microsoft IIS (Internet Information Services)

Grafo de las llamadas al sistema para retornar una imagen en el servidor web libre Apache.

2. Base de código Uniforme (Confiabilidad, Mantenibilidad, y Usabilidad)

> Mi trabajo es decir que “no”, a algunos elementos. Si usted mantiene las cosas limpias suficientemente y tiene algún tipo de requerimiento acerca de que código puede parecerse, usted está al frente del juego.
>
> —Linus Torvalds

Los ingenieros de Linux han encontrado una vía de correr la misma base de código en una gran variedad de procesadores, en dispositivos desde teléfonos celulares a supercomputadores, un logro sin precedentes. Linux ha sido afinado para en primera, correr correctamente, y luego correr eficientemente en dos, cuatro, ocho, y ahora en máquinas de 1,000 procesadores. El software posee una maleabilidad infinita, por lo que este núcleo universal siempre ha sido posible — esto toma solo un puñado de compañías de hardware trabajando juntas en pos de lograrlo.

Poner todo en una sola base de código ayuda a la confiabilidad. Correr el núcleo de Linux en una computadora de 32 procesadores revela errores multi-hilos mucho más rápido que en una laptop de 2 procesadores. Correr en máquinas de bajas prestaciones mantiene el código pequeño y simple, que hacen correr estos más eficientemente en máquinas de escritorio. Características que primero aparecen en laptops y tablets eventualmente encuentran su camino a incluso dispositivos aún más pequeños donde el código es aun mejor probado y depurado. Los muchos desarrolladores de hardware y servidores que quieren confiabilidad extrema se aseguran que el núcleo de mi PC es tan confiable como para el cliente más exigente.

Linux es mas flexible que el núcleo de Windows NT, aunque ambos son muy limpios y flexibles. Por ejemplo, la Agencia de Seguridad Nacional (NSA) ha creado un componente de software libre llamado SELinux (Mejoras de Seguridad para Linux) que adiciona un fuerte mecanismo de seguridad reforzado conocido como Mandatory Access Control (Control de Acceso Obligatorio).[4] Haciendo

4 Esta es la vía de adicionar seguridad adicional debido a que el sistema operativo puede decir, por ejemplo: Debido a que un reproductor de música o vídeos no tiene razón de escribir ficheros hacia el disco, el sistema le puede quitar este permiso. Antes de que el núcleo trate de realizar algo interesante, el le preguntará al Control de Acceso Obligatorio (MAC) cuando una operación como esa es permitida. La seguridad se chequea en casi cualquier otro sistema operativo simplemente preguntando si a esa *persona* le está permitido realizar algo. Creando una política por defecto adiciona trabajo adicional a los que escriben las aplicaciones, que por si misma no resuelve completamente este problema. Un procesador de texto necesita acceso completo de lectura y escritura, así que ¿cómo usted soluciona el problema de que un virus en la macro de un documento abra todos tus

estos mecanismos públicos nos ayuda a asegurarnos que no existen puertas traseras hacia las computadoras de la NSA. Discutiré en capítulos posteriores porqué los gobiernos pueden adoptar software libre, incluso para escenarios de alta seguridad, concepto que la NSA entiende en la actualidad.

En el mundo de Linux, uno tiene muchas mas posibilidades para encontrar la herramienta que uno requiere para realizar una tarea. Algunos podrán decir que muchas opciones es una mala idea, pero creando muchos componentes obliga a limpiar las fronteras, y la supervivencia al cursar del tiempo del más indicado sobre los demás.

3. Ciclos de Lanzamiento Frecuentes (Mantenibilidad y Usabilidad)

Microsoft tiene un lema: “Lanza temprano, lanza a menudo.” Esta filosofía es sabia para el desarrollo de software debido a que obliga a los equipos a mantener un producto de alta calidad cada día, y a más temprano el lanzamiento, mas pronto usted podrá recibir e incorporar retroalimentación.

Si embargo, esta filosofía solo trabaja cuando se adopta. Desafortunadamente, dos de los productos más grandes de Microsoft, Windows y Office, no siguen esta filosofía. Por supuesto, ¡pagar $240 cada año por la última actualización de Windows “Ultimate” no es aceptable tampoco!

El núcleo de Linux se lanza cada tres meses. Para un producto de esta magnitud y complejidad, la taza de lanzamiento de Linux no tiene precedentes. Este logro ha permitido al núcleo de Linux lanzar drivers antes que Windows, e incluso a veces antes de que el propio hardware halla sido lanzado. Linux soporta USB 3.0 antes de que Microsoft, que no lo incluyó en Windows 7. Debido a que Linux está constantemente cerca de un lanzamiento, puedes obtener cualquier binario aleatorio de la computadora de Linus, ponerlo en un cohete y sentirte bastante seguro de que este no se estrellará.

Una gran parte de la demanda del Departamento de Justicia en contra de Microsoft se enfoca en que la compañía incluyó varios componentes de software en el sistema operativo. El gobierno acusó

---

archivos y escriba basura en estos? SELinux no trata con esta situación debido a que este no posee esta información. En lenguajes de programación con recolección de basura (GC), es posible recorrer la pila y determinar más información acerca de que si la macro, o el propio procesador de texto está solicitando abrir el fichero.

a Microsoft de excluir componentes de terceros suprimiendo de esta manera la competencia. Pero la atadura a Microsoft ha sido a la vez una bendición y una maldición.

La bendición ha sido que teniendo piezas que trabajan en conjunto, ellos pueden rehusar código y ser mas integrado. La maldición es que Microsoft ha creado una situación donde revisa y adiciona nuevas características a todos sus componentes independientes a la vez. Como consecuencia, sus componentes toman años en estabilizarse, y no puedes realizar un lanzamiento hasta que el último de estos esté listo.[5]

En contraste, en un sistema operativo libre, los componentes de software solo dependen de la versiones liberadas. Cada equipo no intenta lanzar el mismo día, por lo que el sistema operativo posee la última versión de todos sus componentes liberados.[6] Organizaciones de desarrollo separadas han delineado fronteras que han simplificado las dependencias, y permitido a todo el mundo moverse hacia sus propios objetivos.

Muchos usuarios se preguntan acerca de que si el software libre algún día será tan bueno como el propietario debido a que los chicos del software libre no pueden pagar estudios de usabilidad. Habiendo visto estudios de usabilidad en Microsoft, cogiendo a personas de la calle para que le dieran algunos bits de retroalimentación en un cuarto con un espejo de dos lados no es necesario. El Internet, y todos los mecanismos de comunicación, nos proporcionan un continuo y rico mecanismo de retroalimentación que usted puede obtener sin necesidad de ningún estudio de usabilidad. Es más, los estudios de usabilidad no interesan si no puedes incorporar estos cambios de forma fácil y rápida. Con bases de código limpias y ciclos de lanzamiento frecuentes, la usabilidad sucederá de forma automática. Yo he empleado gran cantidad de tiempo usando Linux y he encontrado varias aplicaciones perfectamente usables.

Lanzando una nueva plataforma cada cinco años en teoría provee a los compañeros de Microsoft una plataforma estable sobre la cual construir. Si embargo, en realidad, resulta que esto tiene sus fallas. Por ejemplo, yo no fui capaz de instalar el driver de una impresora HP Photodesk 7960 en Windows Server 2003, a pesar de que el driver se instalaba perfectamente en XP. Las sutiles e indocumentadas

5 La alternativa es que por cada componente use la versión previa de cada uno de sus componentes dependientes, lo que significa que las características en el último Internet Explorer no pueda ser mostrada en varios lugares que usen la versión anterior. Sin embargo, ¿el sistema de ayuda necesita la última versión?

6 Algunos componentes contienen múltiples versiones para permitir un periodo de transición.

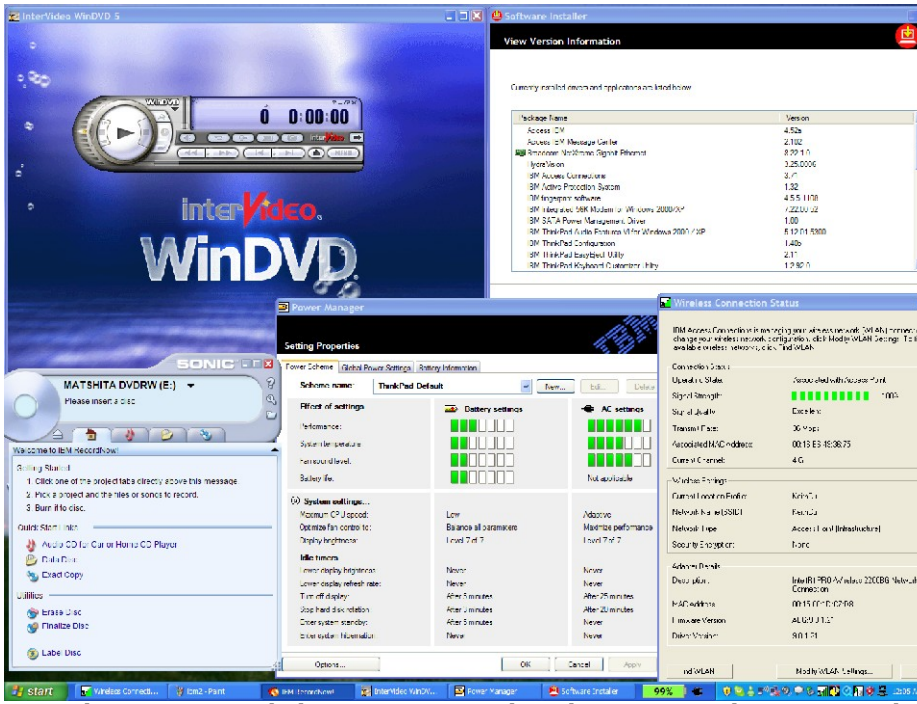
sin intención diferencias entre ambos sistemas, que fueron lanzados con dos años de diferencia, han creado dolores de cabeza de compatibilidad aun cuando Microsoft se supone una plataforma uniforme.

4. Menores Costos de Desarrollo (Mantenibilidad)

Es mucho menos caro para los vendedores de hardware soportar Linux. Si usted quiere construir un controlador de dispositivo, un buen lugar para empezar es mirando los drivers que existen en la actualidad, una oportunidad que Linux ofrece a todo el mundo. Un “Device Driver Toolkit (Herramientas de Controladores de Dispositivos)” propietario con sus códigos de ejemplo nunca es tan bueno como el código de producción. Esas caras herramientas contienen documentación pero no código fuente — por lo que usted tiene que a veces adivinar que es lo que realmente está pasando por debajo de este.

Encontramos actualmente en Windows que los fabricantes de hardware han duplicado gran cantidad de la funcionalidad que Windows provee pero que no se ajusta perfectamente a sus necesidades. Por ejemplo, IBM incluye su propio applet e icono de estado para la Internet inalámbrica, por lo que el Windows XP en hardware de IBM posee dos de estos. Presumiblemente ellos no quedaron satisfechos con las características que Windows les brindaba, y no fueron capaces de arreglarlas. ¡Por lo que tuvieron que construir nuevos applets desde cero! Es lo que además le da a Windows la sensación de un confuso grupo de componentes solapados entre ellos.

He aquí cinco de los 100 applets que IBM adiciona a Windows:



Windows XP con 5 de los 100 extra applets de IBM. Vean la gran cantidad de iconos de estado en una instalación casi virgen.

Construir todos estos applets, diseñando interfaces de usuario multilenguajes, brindando los medios para instalar y configurar, etc, es diez veces más trabajo que simplemente escribir el controlador de dispositivo, mejorando otros drivers, y adicionándolos a la base de código oficial. Mi impresora Photodesk 7960 trabaja bien en Windows XP, pero no en Windows Server 2003 debido a que el código de instalación explota — por el cual HP no tenía que preocuparse en primer lugar.

5. Seguridad (Fiabilidad y Mantenibilidad)

> Para arruinar una máquina con Linux, tienes que trabajar en ello; para arruinar una máquina con Windows sólo necesitas trabajar en esta.
>
> —Scott Granneman

Intentar comparar la seguridad de los sistemas operativos es un esfuerzo complicado porque hay un infinito número de riesgos. Es

como preguntar si un carro es más seguro que otro. Contra qué: ¿Chocar por el frente? ¿Volcarse y dar vueltas? ¿ Tener el tanque de gas agujereado ? ¿Su habilidad para romperse?

Además, ni Windows ni Linux son sistemas operativos perfectamente seguros; ambos están propensos a virus de sobrecarga de memoria, un asunto que discutiré en el capítulo de las herramientas. También, nuevas amenazas aparecen todo el tiempo,y por eso cada avance infame requiere de nueva lógica para defendernos contra él. Dadas estas advertencias, todavía es posible hacer algunas comparaciones.

Algunas de las ventajas anteriores de Linux, tales como su simplicidad, modularidad, soporte para SELinux, etc. todo ayuda con su seguridad. Además, la naturaleza de cómo Linux es desarrollado también ayuda. Un documento emitido por el Ejército de los Estados Unidos decía:

> **El software de código abierto está potencialmente sujeto al escrutinio de muchos ojos**
>
> Por eso los bugs, fallos de seguridad, y el diseño pobre no se pueden esconder por mucho tiempo, por lo menos cuando el software tiene una comunidad de programadores para brindarle soporte. Y desde que corregir el código no depende de un único proveedor, los parches de seguridad son a menudo distribuidos mucho más rápido que los parches de seguridad de software de código cerrado.
>
> **Puede incrementarse la calidad y la seguridad del código fuente**
>
> Con software de código cerrado, muchas veces es difícil de evaluar la calidad y la seguridad del código. Además, las compañías de código cerrado tienden a retardar el anuncio de las fallas de seguridad o bugs en sus productos. A menudo, esto significa que sus clientes no se enteran de las fallas de seguridad hasta semanas o meses después de que fuera conocida internamente.

Otra gran diferencia entre Linux y Windows es que Linux fue adaptado desde Unix, el cual tenía un correcto diseño multiusuario desde el principio. En Windows, los usuarios han tenido históricamente acceso de Administrador completo del ordenador, incluyendo la posibilidad de sobrescribir los archivos del sistema. Cuando el ordenador es atacado por un virus, el virus puede ganar los mismos privilegios que el usuario y por tanto esconderse a sí mismo dentro de los archivos del sistema, lo cual lo vuelve muy difícil de eliminar. En Linux, sólo puedo escribir en mis propios archivos y tener sólo

acceso de lectura a todos los demás. Linux es por naturaleza multiusuario, y por eso se enfoca en permisos mínimos y se minimiza cualquier daño.

Un estudio encontró que hay alrededor de 60,000 virus conocidos para Windows y sólo unos 40 para Linux. Otro estudio realizado por *Evans Data* mostró que el 8% de los desarrolladores de Linux dicen que sus ordenadores han sido infectados por código malicioso, comparado al 60 % de los ordenadores con Windows.

Brian Krebs del *Washington Post* encontró que el código para explotar fallos en Internet Explorer 6 existió por 284 días en el 2006, mientras Firefox estuvo en riesgo por sólo 9 días. El experto de seguridad Bruce Schneier recomendó en diciembre del 2004 que las personas no deberían usar el Internet Explorer. Algunos argumentan que Linux y Firefox tienen menos virus porque ellos tienen menos usuarios, pero Apache es bien respetado por su seguridad.

Durante casi toda mi estancia en Microsoft, nos preocupamos por las funcionalidades, la fiabilidad, y el rendimiento, no por la seguridad. El funcionario Jefe de Investigación y Estrategia de Microsoft, Craig Mundie, dijo en el 2002:

> Mucho de los productos que diseñamos en el pasado han sido menos seguros de lo pudieran haber sido debido a que estábamos diseñando con las funcionalidades en mente en vez de la seguridad.

Microsoft ha incrementado grandemente su enfoque en la seguridad en los últimos años, y estoy seguro que la seguridad de cada producto mejora con cada liberación, pero el código innecesario presente en su código base es un impedimento continuo.

Habiendo buscado a través de las fuentes a un número de aplicaciones para Linux, uno puede decir que las bases de código libres son típicamente más limpias que sus homólogos de Windows, lo cual las hace más fácil de hacer seguras. La distribución de Linux por defecto orientada a servidores Debian 4.0 necesita unos meros 170MB de espacio en disco, mientras Windows Server 2003 necesita 3 GB. Siendo igual todas las demás cosas, el hecho de que Debian es 17 veces más pequeño significa que será más seguro. La base de datos libre MySQL tiene unos meros 26MB al descargar; el código es limpio y pequeño, y por eso es mucho más probable de que sea fiable y seguro.

Otra ventaja de Linux es que todas las aplicaciones en un sistema operativo basado en Linux reciben actualizaciones de seguridad. En el mundo de Microsoft, sólo el código de Microsoft está protegido por el servicio de Windows Update.

Mientras Linux es libre para adquirir, este puede además ser menos costoso para ejecutar y mantener que Windows debido a su mejor seguridad. La ciudad de Manchester en Inglaterra gastó $2 millones en el 2009 para eliminar el gusano Conflicker de sus computadoras..

6. Linux ha aprendido de Windows

Mientras el kernel de Windows NT fue lo más grande en la tecnología cuando fue liberado en 1993, muchas de sus mejores ideas han sido bien aprendidas y absorbidas, en respuesta al hecho de que el código nunca ha sido liberado.

Por ejemplo, el kernel de Linux soporta E/S(entrada/salida) asíncrona, una forma innovadora para hacer lecturas y escrituras sin tener que levantar “hilos” de recursos. Esta fue una innovación hecha extendida primeramente en Windows NT.

La habilidad de cargar código dinámicamente es otra importante característica que el kernel de Linux adoptó desde NT y otros. Plug and play y suspender e hibernar fueron el resultado de una colaboración entre Microsoft y las compañías de hardware en la década de 1990, y Linux ahora soporta está característica.

A lo largo de todo el software libre, los desarrolladores han incorporado buenas ideas provenientes del mundo exterior. No existe el síndrome de Aquí Nada Inventado; una buena idea es una buena idea, y el código existente es incluso mejor. En el software de hoy en día, el impedimento más grande para compartir ideas no es el ego, sino los acuerdos de licencias.

El kernel de Linux ha incluso aprendido de los errores de Microsoft. Por ejemplo, una funcionalidad añadida al kernel de Window NT 4.0 fue colocar el código que dibuja widgets dentro del mismo kernel. Mientras esto puede mejorar el rendimiento gráfico, también significa que un bug en el código de un botón tiene la capacidad hacer que todo el sistema falle. La mejor manera de mantener un sistema seguro y fiable es mantener tanto código como sea posible en modo de usuario por encima del kernel, y Linux sigue esta estrategia.[7]

7 En Windows Vista, Microsoft movió *algunos* de los controladores de dispositivos al modo de usuario pero deberían haber mantenido los controladores de dispositivos pequeños, simple, y en el kernel y en cambio haber movido los widgets y las cosas sin importancia al modo de usuario.

La Carrera de las Funcionalidades

Uno de los mejores argumentos de Microsoft contra el software libre a través de los años ha sido promocionar sus nuevas funcionalidades y usar eso como “prueba” de que el software libre siempre estará por detrás del software propietario. En el momento en que un producto de Microsoft sale al mercado, siempre ofrece funcionalidades que nadie más tiene. Sin embargo, la mayoría de las funcionalidades son de hecho unas que otros sí poseen, y Microsoft se está poniendo al día. Casi toda funcionalidad nueva es un paso de avance evolutivo en un lugar en que Microsoft sintió que necesitaba trabajar.

Pero como Microsoft, cada equipo en la comunidad de software libre se encuentra mejorando su código cada día. De hecho, como Microsoft se demora tanto entre versiones, la comunidad de software libre frecuentemente ha añadido nuevas funcionalidades antes que Microsoft. El sitio web http://kernelnewbies.org muestra la lista de las últimas novedades añadidas desde la anterior liberación de hace 3-4 meses antes, ¡ y este es típicamente del tamaño de 15 páginas! Por ejemplo, aquí está la lista de las novedades de driver incorporadas a la versión 2.6.26 del kernel de Linux, el cual tuvo un ciclo de desarrollo de 3 meses.

Linus 2.6.26 driver workitems

4.1. IDE/SATA

IDE

Add warm-plug support for IDE devices
Mark "idebus=" kernel parameter as obsoleted (take 2)
Remove ide=reverse IDE core
Add "vlb|pci_clock=" parameter
Add "noacpi" / "acpigtf" / "acpionboot" parameters
Add "cdrom=" and "chs=" parameters
Add "nodma|noflush|noprobe|nowerr=" parameters
Add Intel SCH PATA driver
Add ide-4drives host driver (take 3)
gayle: add "doubler" parameter
Remove the broken ETRAX_IDE driver

SATA

sata_inic162x: add cardbus support
libata: prefer hardreset
ata: SWNCQ should be enabled by default
Make SFF support optional
libata: make PMP support optional
sata_mv: disable hotplug for now, enable NCQ on SOC, add basic port multiplier support
sata_fsl: Fix broken driver, add port multiplier (PMP) support

4.2. Networking

ssb: add a new Gigabit Ethernet driver to the ssb core
Add new qeth device driver,
Add new ctcm driver that reemplaces the old ctc one,
New driver "sfc" for Solarstorm SFC4000 controller.
Driver for IXP4xx built-in Ethernet ports
Add support the Korina (IDT RC32434) Ethernet MAC
iwlwifi: Support the HT (802.11n) improvements,,,, add default WEP key host command, add 1X HW WEP support, add default WEP HW encryption, use HW acceleration decryption by default, hook iwlwifi with Linux rfkill, add TX/RX statistics to driver, add

4.6. Video

cx88: Add support for the Dvico PCI Nano, add xc2028/3028 boards, add support for tuner-xc3028
saa7134: add support for the MSI TV@nywhere A/D v1.1 card, add support for the Creatix CTX953_V.1.4.3 Hybrid
saa717x: add new audio/video decoder i2c driver
Support DVB-T tuning on the DViCO FusionHDTV DVB-T Pro
Add support for xc3028-based boards
ivtv: add support for Japanese variant of the Adaptec AVC-2410
Add basic support for Prolink Pixelview MPEG 8000GT
bttv: added support for Kozumi KTV-01C card
Add support for Kworld ATSC 120
CX24123: preparing support for CX24113 tuner
Added support for Terratec Cinergy T USB XXS
budget: Add support for Fujitsu Siemens DVB-T Activy Budget
Support for DVB-S demod PN1010 (clone of S5H1420) added
Added support for SkyStar2 rev2.7 and ITD1000 DVB-S tuner
em28xx-dvb: Add support for HVR950, add support for the HVR-900
Add support for Hauppauge HVR950Q/HVR850/FusioHDTV7-USB
HVR950Q Hauppauge eeprom support
Adding support for the NXP TDA10048HN DVB OFDM demodulator
Add support for the Hauppauge HVR-1200
pvrusb2-dvb: add DVB-T support for Hauppauge pvrusb2 model 73xxx
Add support for Beholder BeholdTV H6
cx18: new driver for the Conexant CX23418 MPEG encoder chip
s5h1411: Adding support for this ATSC/QAM demodulator

4.7. SCSI

zfcp: Add trace records for recovery thread and its queues, add traces for state changes., trace all triggers of error recovery activity,register new recovery trace., remove obsolete erp_dbf trace,

debugfs to iwl core, enables HW TKIP encryption, add led support, enables RX TKIP decryption in HW, remove IWL{4965,3945}_QOS

ath5k: Add RF2413 srev values, add RF2413 initial settings, identify RF2413 and deal with PHY_SPENDING, more RF2413 stuff, port to new bitrate/channel API, use software encryption for now

pasemi_mac: jumbo frame support, enable GSO by default, basic ethtool support, netpoll support

rt2x00: Add per-interface structure, enable master and adhoc mode again, enable LED class support for rt2500usb/rt73usb

e1000e: Add interrupt moderation run-time ethtool interface, add support for BM PHYs on ICH9

niu: Add support for Neptune FEM/NEM cards for C10 server blades, add Support for Sun ATCA Blade Server.

gianfar: Support NAPI for TX Frames

ehea: Add DLPAR memory remove support

sfc: Add TSO support

b43: Add QOS support, add HostFlags HI support, use SSB block-I/O to do PIO

S2io: Multiqueue network device support implementation,, enable multi ring support, added napi support when MSIX is enabled.

ixgbe: Introduce MSI-X queue vector code, introduce Multiqueue TX, add optional DCA infrastructure, introduce adaptive interrupt moderation

uli526x: add support for netpoll

fmvj18x_cs: add NextCom NC5310 rev B support

zd1211rw: support for mesh interface and beaconing

libertas: implement SSID scanning for SIOCSIWSCAN

ethtool: Add support for large eeproms

The scheduled bcm43xx removal

The scheduled ieee80211 softmac removal

The scheduled rc80211-simple.c removal

Remove obsolete driver sk98lin

Remove the obsolete xircom_tulip_cb driver

4.3. Graphics

radeon: Initial r500 support,,

intel_agp: Add support for Intel 4 series chipsets

i915: Add support for Intel series 4 chipsets

Add support for Radeon Mobility 9000 chipset

fb: add support for foreign endianness

pxafb: preliminary smart panel interface support,

Driver for Freescale 8610 and 5121 DIU

intelfb: add support for the Intel Integrated Graphics Controller 965G/965GM

Add support for Blackfin/Linux logo for framebuffer console

4.4. Sound

hda-codec - Allow multiple SPDIF devices, add SI HDMI codec support, add support for the OQO Model 2, add support of Zepto laptops, support RV7xx HDMI Audio, add model=mobile for AD1884A & co, add support of AD1883/1884A/1984A/1984B, add model for cx20549 to support laptop HP530, add model for alc883 to support FUJITSU Pi2515, add support for Toshiba Equium L30, Map 3stack-6ch-dig ALC662 model for Asus P5GC-MX, support of Lenovo Thinkpad X300, add Quanta IL1 ALC267 model, add support of AD1989A/AD1989B, add model for alc262 to support Lenovo 3000, add model for ASUS P5K-E/WIFI-AP, added support for Foxconn P35AX-S mainboard, add drivers for the Texas Instruments OMAP processors, add support of Medion RIM 2150, support IDT 92HD206 codec

ice1724 - Enable AK4114 support for Audiophile192

ice1712: Added support for Delta1010E (newer revisions of Delta1010), added support for M-Audio Delta 66E, add Terrasoniq TS88 support

Davinci ASoC support

intel8x0 - Add support of 8 channel sound

ASoC: WM9713 driver

Emagic Audiowerk 2 ALSA driver.

Add PC-speaker sound driver

oxygen: add monitor controls

virtuoso: add Xonar DX support

soc - Support PXA3xx AC97

pxa2xx-ac97: Support PXA3xx AC97

4.5. Input

Add support for WM97xx family touchscreens

WM97xx - add chip driver for WM9705 touchscreen, add chip driver for WM9712 touchscreen, add chip driver for WM97123 touchscreen, add support for streaming mode on Mainstone

wacom: add support for Cintiq 20WSX

xpad: add support for wireless xbox360 controllers

add trace records for recovery actions.

qla2xxx: Add support for host supported speeds FC transport attribute., add FC-transport Asynchronous Event Notification support., add hardware trace-logging support., add Flash Descriptor Table layout support., add ISP84XX support., add midlayer target/device reset support.

iscsi: extended cdb support, bidi support at the generic libiscsi level, bidi support for iscsi_tcp

scsi_debug: support large non-fake virtual disk

gdth: convert to PCI hotplug API

st: add option to use SILI in variable block reads

megaraid_sas: Add the new controller(1078DE) support to the driver

m68k: new mac_esp scsi driver

bsg: add large command support

Add support for variable length extended commands

aacraid: Add Power Management support

dpt_i2o: 64 bit support, sysfs

Firmware: add iSCSI iBFT Support

4.8. WATCHDOG

Add a watchdog driver based on the CS5535/CS5536 MFGPT timers

Add ICH9DO into the iTCO_wdt.c driver

4.9. HWMON

thermal: add hwmon sysfs I/F

ibmaem: new driver for power/energy/temp meters in IBM System X hardware

i5k_amb: support Intel 5400 chipset

4.10. USB

ISP1760 HCD driver

pxa27x_udc driver

CDC WDM driver

Add Cypress c67x00 OTG controller core driver,,

Add HP hs2300 Broadband Wireless Module to sierra.c

Partial USB embedded host support

Add usb-serial spcp8x5 driver

r8a66597-hcd: Add support for SH7366 USB host

Add Zoom Telephonics Model 3095F V.92 USB Mini External modem to cdc-acm

Support for the ET502HS HDSPA modem

atmel_usba_udc: Add support for AT91CAP9 UDPHS

4.11. FireWire

release notes at linux1394-user

4.12. Infiniband

IPoIB: Use checksum offload support if available, add LSO support, add basic ethtool support, support modifying IPoIB CQ event moderation, handle 4K IB MTU for UD (datagram) mode

ipath: Enable 4KB MTU, add code to support multiple link speeds and widths, EEPROM support for 7220 devices, robustness improvements, cleanup, add support for IBTA 1.2 Heartbeat

Add support for IBA7220,,,,,,,,

mthca: Add checksum offload support

mlx4: Add checksum offload support, add IPoIB LSO support to mlx4,

RDMA/cxgb3: Support peer-2-peer connection setup

4.13. ACPI and Power Management

ACPICA: Disassembler support for new ACPI tables

eeepc-laptop: add base driver, add backlight, add hwmon fan control

thinkpad-acpi: add sysfs led class support for thinklight (v3.1), add sysfs led class support to thinkpad leds (v3.2)

Remove legacy PM

4.14. MTD

m25p80: add FAST_READ access support to M25Pxx, add Support for ATMEL AT25DF641 64-Megabit SPI Flash

JEDEC: add support for the ST M29W400DB flash chip

NAND: support for pxa3xx

NOR: Add JEDEC support for the SST 36VF3203 flash chip

NAND: FSL UPM NAND driver

AR7 mtd partition map

NAND: S3C2410 Large page NAND support

NAND: Hardware ECC controller on at91sam9263 / at91sam9260

4.15. I2C

Add support for device alias names

Convert most new-style drivers to use module aliasing

Renesas SH7760 I2C master driver

New driver for the SuperH Mobile I2C bus controller

Add PS/2 serio driver for AVR32 devices	Convert remaining new-style drivers to use module aliasing
aiptek: add support for Genius G-PEN 560 tablet	**4.16. Various**
Add Zhen Hua driver	MMC: OMAP: Add back cover switch support
HID: force feedback driver for Logitech Rumblepad 2, Logitech diNovo Mini pad support	MMC: OMAP: Introduce new multislot structure and change driver to use it
4.6. Video	mmc: mmc host test driver
V4L2 soc_camera driver for PXA270,,	4981/1: [KS8695] Simple LED driver
Add support for the MT9M001 camera	leds: Add mail LED support for "Clevo D400P"
Add support for the MT9V022 camera	leds: Add support to leds with readable status
Add support for the ISL6405 dual LNB supply chip	leds: Add new driver for the LEDs on the Freecom FSG-3
Initial DVB-S support for MD8800 /CTX948	RAPIDIO:
cx23885: Add support for the Hauppauge HVR1400, add generic cx23417 hardware encoder support	Add RapidIO multi mport support
Add mxl5505s driver for MaxiLinear 5505 chipsets, basic digital support.	Add OF-tree support to RapidIO controller driver
	Add serial RapidIO controller support, which includes MPC8548, MPC8641
	edac: new support for Intel 3100 chipset
	Basic braille screen reader support
	ntp: support for TAI
	RTC: Ramtron FM3130 RTC support

No te preocupes si no entiendes que significan estas cosas como yo tampoco las entiendo todas. Es importante comprender que incluso el hardware de los ordenadores es demasiado grande y complicado para una compañía vigilar su desarrollo.

Esta es sólo una porción de los cambios de código para esa versión de Linux, y no incluye el trabajo en los sistemas de archivos, gestión de redes, rendimiento, trabajo específico de arquitectura, y así sucesivamente.

El software libre ha incorporado incontables funcionalidades antes de Microsoft. Sin embargo, muchas de estas funcionalidades son tan poco visibles que tus ojos se vidriarían leyendo la lista. Cualquiera que concentre su atención en las funcionalidades mostradas en las últimas liberaciones de Microsoft se pierde la pintura más larga: a pesar de que estas características son convincentes, te convierten en muchas ocasiones en alguien más atado profundamente al mundo del software propietario y hacen más duro el proceso de migración.

El software nuevo de Microsoft siempre brinda nuevas funcionalidades, pero usualmente es sólo una porción de código para un escenario. Linux soporta 60 sistemas de archivos, incluyendo muchas soluciones de clúster, eso permite escalar sin límites el almacenaje de disco entre un grupo de servidores. Microsoft soporta sólo uno, el cual está muy apretadamente atado a su sistema, lo que limita su flexibilidad. El software libre también ofrece soluciones desde diferentes direcciones. Hay muchas formas de atacar la virtualización y Linux está trabajando sobre ellas en paralelo y encontrando que es común entre ellas.[8]

8 Dos de las más grandes diferencias en estrategia es Linux en Modo de Usuario(siglas UML en inglés) el cual cambia a Linux para correr como una aplicación por encima de otra instancia de Linux, y la virtualización estándar, la cual corre el Sistema Operativo invitado en modo kernel, aunque esta actualmente no se comunica con el hardware. El kernel de Linux está evolucionando hacia mejo-

Linux está Inexorablemente Ganando

El hecho de que el kernel de Linux tiene estas varias ventajas sobre Windows significa dos cosas. Primero, el argumento de que los ingenieros del software libre son incapaces de innovar y sólo copian el trabajo de otros no es verdad.

El líder de Windows Server recientemente dijo que el software libre, “por su propia naturaleza, no permite que exista la propiedad intelectual.” Esta declaración es incorrecta: el kernel de Linux está haciendo cosas que nunca antes un kernel había hecho, tal como Wikipedia está haciendo cosas que ninguna enciclopedia había hecho anteriormente.

Segundo, incluso si Microsoft diera el código fuente del kernel de Windows, la comunidad existente de software libre lo desecharía. Una comunidad mundial ha colocado a Linux en la forma que este debería ser. Linus llama Windows “brujería”, por eso ¿ por qué trabajar sobre una base de código inferior y que sus detalles hasta ahora han sido opacos ?

rar la arquitectura, y lo que es compartido entre las diferentes estrategias.

El cobro por un Sistema Operativo

Un sistema operativo Linux es una bestia enteramente diferente comparado a un sistema operativo de Microsoft. Microsoft estaba debatiendo constantemente sobre cuanto invertir en Windows, y cuanto dejar de lado a los ingresos por licencias adicionales en otros productos. Windows Vista tiene cinco versiones diferentes, (¡originalmente anunciaron ocho!), cada una con básicamente el mismo código, pero dramaticalmente con distintos precios :

Producto (Amazon.com)	**Actualización**	**Nuevo**
Windows Vista Ultimate	$243	$350
Windows Vista Business	$176	$260
Windows Vista Home Premium	$140	$219
Windows Vista Home Basic	$85	$157
Windows Vista Enterprise	Licencia personalizada	

Microsoft cobra de $85 a $350 por Windows Vista, pero el código fuente en cada versión es 99% el mismo.

La meta personal de Microsoft es colocar tantas nuevas y persuasivas funcionalidades dentro de versiones de gama alta y de alta rentabilidad, a pesar de que los precios no están relacionados con el trabajo necesario para construir las tantas versiones de Vista.

Crear múltiples versiones es un truco porque si las aplicaciones de terceros dependen de funcionalidades que están en una versión particular de Windows, entonces las aplicaciones no correrán, y la marca Windows se debilita. Por eso, Microsoft incorporaría algunas veces lógica para estropear el rendimiento de funcionalidades de gama alta en versiones de gamas bajas. En el mundo de software libre, nadie estropea deliberadamente su propia creación.

Cuando trabajé en Microsoft, hubieron numerosas guerras de posiciones. Por ejemplo, el equipo de Word luchó con el equipo de Windows sobre si WordPad, una diminuta applet de Windows, debería tener la habilidad de leer archivos DOC de Word. El equipo de Windows quiso crear un sistema operativo con la habilidad de mostrar cientos de millones de archivos DOC, pero el equipo de Word no quería crear una razón para que las personas no compraran Word. Hubieron también batallas entre los equipos responsables de Out-

look y Outlook Express, Exchange y SQL, Works y Word, FoxPro y Access, Access y VB, SQL Server y Access, PC y XBox. Cada equipo temía de que otro equipo agregara funcionalidad que desalentara a alguien a comprar su producto.

Microsoft también se debatía en si agrupar características importantes en el sistema operativo, tal como hicieron con los buscadores web, la mensajería instantánea y multimedia, y dejar funcionalidades fuera para ganar más dinero luego. Windows no trae un diccionario porque Office trae uno. Windows no viene con herramientas de desarrollo porque esas son partes del negocio de Microsoft Visual Studio. Con Linux, cualquier cosa libre es bienvenida en el sistema operativo.

Además de las decisiones estratégicas de Microsoft de excluir ciertas prestaciones, también hay casos de negligencia benigna. El Grabador de Sonido en Windows XP te permite grabar solamente 1 minuto, una limitación que existe desde los días de Windows de 16 bits que nadie se molestó en corregir. La solución oficial de Microsoft para el cliente es comprar Office OneNote.

Las applets, las herramientas de línea de comandos, y muchas otras importantes pero poco atractivas partes de un sistema operativo siempre fueron asignadas a recursos muy limitados. Pregúntale a Steve Ballmer por los recursos para el poco importante Grabador de Sonido, y recibirás una indecente e insultante respuesta. En el modelo de software libre, cualquiera, en su calendario, puede mejorar una porción de código que sea útil o interesante para ellos, siendo “estratégico” o no. Eric Raymond denomina este fenómeno un desarrollador “rascándose su propia picazón.” Wikipedia está construida casi completamente desde este mecanismo, y sólo el software libre puede captar cada pequeño avance.

Complejidad de los Acuerdos de Licencia

> Sería posible financiar la construcción de todas las carreteras con peaje. Esto implicaría tener casetas de peaje en todas las esquinas de las calles. Este sistema proporcionaría un incentivo para mejorar las carreteras. También tendría la virtud de causar que los usuarios de cualquier carretera pagaran por esa carretera. Sin embargo, es una obstrucción artificial para suavizar el manejo—artificial, porque no es una una consecuencia de cómo las carreteras o los carros trabajan.
>
> —Richard Stallman

Incluso si crees que es perfectamente correcto cobrar por el software, es difícil de saber que se le cobra a los cliente por el software. Microsoft acostumbró a cobrar una cantidad fija por un producto de

servidor. Sin embargo, un día se dieron cuenta que los clientes que tenían 10 usuarios conectados a un servidor deberían estar pagando memos que esos que tenían 100. Esta observación dio lugar a la creación de licencias de acceso de clientes(CALs en inglés); un concepto que requería el pago por cada usuario individual de un software de servidor, el cual es una carga adicional sobre el cliente. Si tienes 1,000 empleados accediendo a 1,000 servidores, necesitas llenar mucho formularios.

Esto era todo antes la invención de la Internet por el cual el número de usuarios de un servidor podría fácilmente estar en miles, esto hizo el uso de CALs caro e insostenible. Por eso, Microsoft promovió un modelo donde el costo estaba basado en el número de procesadores en el ordenador de manera que las cajas pequeñas costarían menos que las cajas grandes.

Este modelo funcionó hasta que Intel presentó el concepto de “hyperthreading”, lo que creó el pensamiento de que dentro de un ordenador hay dos procesadores, pero el cual adiciona sólo de un 15-30% más rendimiento. Los clientes de Microsoft estuvieron por supuesto molestos ante la idea de comprar un montón de nuevas licencias para mejorar en un desempeño tan pequeño, por eso Microsoft terminó dando licencias gratis para procesadores con soporte para hyperthreading.

Entonces, fue creada la virtualización:

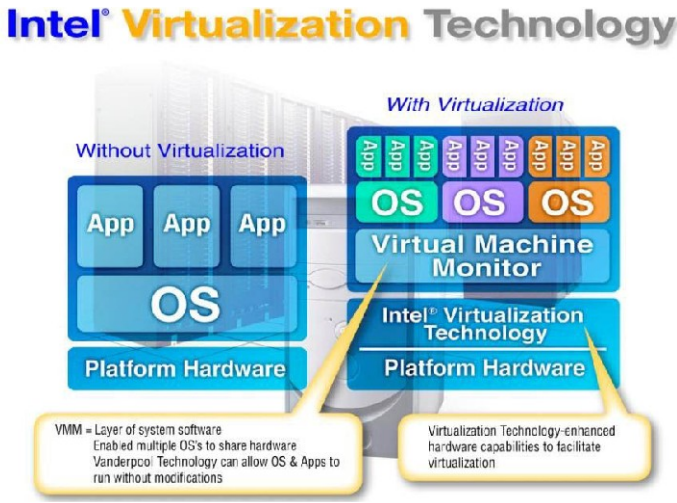


Diagrama de marketing de la virtualización de Intel: La virtualización permite la habilidad de correr múltiples sistemas operativos en una misma computadora: cada instancia piensa que controla la máquina. Los mejores usos para la virtualización es el alojamiento web, dentro de los centro de datos corporativos, y para los desarrolladores de software.

La virtualización permite que diferentes aplicaciones sean completamente aisladas unas de otras dentro de diferentes instancias completas de un sistema operativo, pero comparten a la misma vez el CPU y los recursos de hardware. El aislamiento es bueno porque incluso dentro de un simple centro de datos de una compañía, diferentes departamentos no quieren correr su código en la misma máquina. Si Hotmail se fuera abajo, ellos no quisieran que fuera la culpa de la página web de Microsoft Bob.

Se decía que dentro de los grandes centros de datos de una compañía Fortune 500, los ordenadores usaban como promedio sólo el 15% de su capacidad de CPU. Cada departamento mantenía su propio hardware, el cual habían construido fuera para manipular la máxima carga posible del CPU; esta idea es tan tonta como utilizar sólo el 15% de un edificio en construcción.

La virtualización te ofrece el aislamiento de software pero te permite compartir el hardware. Sin embargo, cuando instalas más software propietario, las cuestiones de licencia aumentan. Si coloco tres copias de un servidor de bases de datos en instancias separadas de virtualización sobre un ordenador de cuatro procesadores, bajo muchos modelos de licencia de Microsoft tendría que comprar licencias suficientes para 12 procesadores — aún si la computadora tuviera sólo 4. Tal como hyperthreading, la virtualización es otra tecnología que no fue concebida cuando Microsoft creó su modelo de licencias por procesador.

En un ambiente de software libre, puedes agregar hardware nuevo y adicionar o eliminar aplicaciones sin pagar o tener que llevar un seguimiento de cualquier cosa. Escribiré más tarde sobre aspectos económicos, pero por ahora sólo es importante que entiendas que el software libre te evita estas molestias, las que son casi siempre muy penosas para las empresas.

Un interesante dilema moral para los vendedores de software propietario es si se deben permitir las copias pirateadas para recibir actualizaciones de seguridad.

El Software Libre solamente cuesta ordenadores

La creación de una base de software industrial unificado, construido en torno al software libre no solamente significará ordenadores más potentes, sino también nos permitirá impulsar su inteligencia a cualquier parte, desde automóviles hasta equipamiento médico. El software libre será construido porque es muy valioso para los negocios; todos querrán entrar al juego y obtenerlo libremente.

El modelo de software propietario ha dañado muchos de los pequeños mercados de software. De hecho, excepto para el negocio de los juegos que trata tanto de talento artístico como de software, Microsoft es casi la última compañía de software propietario que se mantiene en pie. Cuando trabajaba en Microsoft, nuestros mayores competidores eran compañías como Borland, WordPerfect, Corel, Lotus, Netscape y Sybase — nombres que no se oyen nunca más.

No existe una Microsoft de software educativo vendiendo el software usado en cada escuela. Existen varias compañías vendiendo productos privativos, quizás obteniendo suficientes ingresos como para permanecer en el negocio, pero el modelo de software privativo les ha evitado alcanzar las multitudes.

Recientemente, he visto un anuncio para un pequeño software, algo trivial, un programa capaz de convertir un DVD al formato de video del iPod. Microsoft se las ha ingeniado para convencer a todos de que vale la pena vender cualquier pequeño software. Ahora vemos porque la analogía de Stallman sobre el software privativo como barreras de peaje, funciona como obstáculos permanentes, retrasando el progreso al software que fue escrito años antes. El software libre fluye con muchísima menos fricción. De hecho en el mundo del software libre, la definición del sistema operativo de un ordenador cambia completamente.

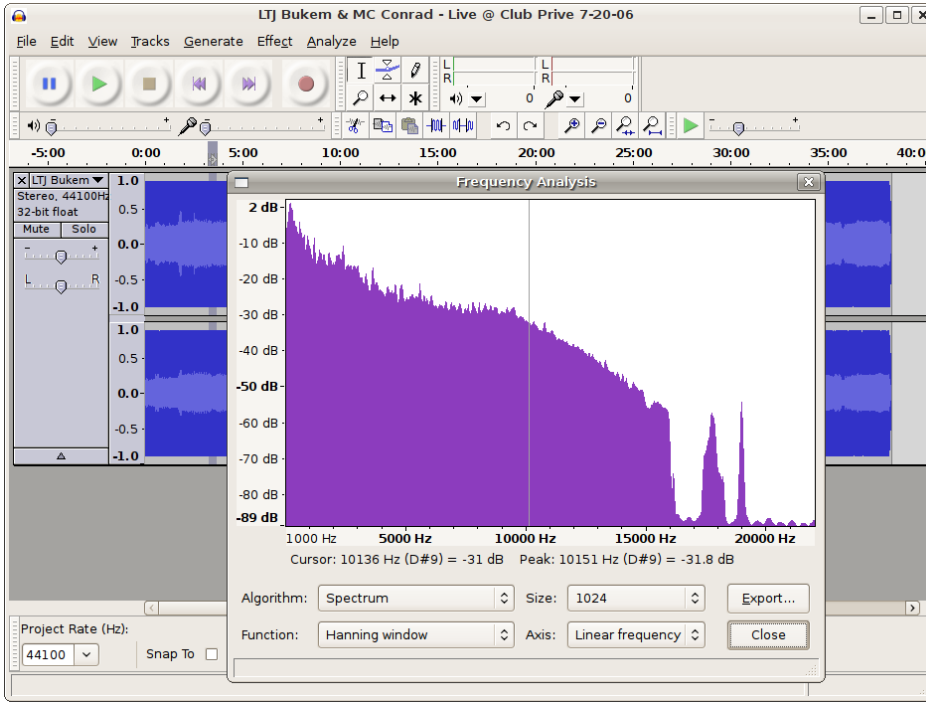
Un Sistema Operativo Libre

> Nuestro sistema operativo competidor más potente es Linux y el fenómeno alrededor del código abierto y el software libre. El mismo fenómeno potencia los competidores de todos nuestros productos. La facilidad de escoger Linux para aprenderlo o para modificar algún componente es muy atractiva. La comunidad académica, compañías en ascenso, gobiernos extranjeros y muchas otras regiones están poniendo sus mejores esfuerzos en Linux.
>
> —Bill Gates

La OLPC tiene tanta potencia de cómputo como un ordenador de 1991, pero no sería nada más que una caja brillante sin el software libre y el contenido.

La mayor diferencia entre Windows y Linux es que el software libre contiene miles de aplicaciones, instalables con un sólo click, y administradas como un conjunto. Un sistema operativo Linux incluye todas las cosas obvias como una hoja de cálculo, navegador web, mensajería instantánea, pero también incluye herramientas para hacer imágenes, música, aplicaciones para niños, software de servidor, la Biblia, herramientas de desarrollo y mucho más.

Audacity es el editor libre más popular en Linux. No tiene un clip de papel que habla: “Tal parece que usted está tratando de agregar eco. ¿Le gustaría algo de ayuda?” Pero sí provee un conjunto de características bien definidas, y múltiples efectos para la manipulación de audio.



Una herramienta para editar audio amplia y confiable

Audacity desde luego, no sufre de una limitación de un minuto de grabación como Windows XP. Su característica más importante son sus plugins para importar, exportar y agregar efectos — la extensibilidad es una de las mejores características del software libre.

Lista de Funcionalidades del Editor de Audio Audacity

Grabación

Audacity puede grabar audio en vivo a través de un micrófono, un mezclador o digitalizar grabaciones desde cintas, vinilos o minidiscos. Con algunas tarjetas de sonido puede además capturar un flujo de audio en transmisión.
Grabación desde micrófono, línea de entrada u otras fuentes..
Mezcla sobre pistas existentes para crear grabaciones multipistas.
Graba hasta 16 canales a la misma vez (requiere hardware multicanal).
Los medidores de nivel pueden monitorear el volumen antes, durante y después de la grabación.

Importación y exportación

Se pueden importar archivos de sonido, editarlos y combinarlos con otros archivos o nuevas grabaciones. Exporta tus grabaciones en varios formatos de archivo comunes.
Importa y exporta archivos en los formatos WAV, AIFF, AU y Ogg Vorbis.
Importa audio MPEG(incluyendo archivos MP2 y MP3) con libmad.
Exporta MP3s con el codificador opcional LAME.
Crea archivos WAV o AIFF listos para ser grabados en CD.
Abre archivos de audio en crudo(sin cabezeras) usando la opción “Importar datos en bruto”.
Nota: Audacity no soporta actualmente WMA, AAC, ni la mayoría de los otros formatos propietarios o restringidos.

Edición

Fácil edición mediante Cortar, Copiar, Pegar, y Eliminar.
Uso ilimitado de Deshacer (y Rehacer) para regresar cualquier número de pasos.
Edición muy rápida de archivos grandes.
Edita y mezcla un número ilimitado de pistas.
Usa la herramienta de dibujo para alterar puntos de muestra individuales.
Desvanece el volumen suavemente con la herramienta Envolvente.

Efectos

Cambia el tono sin cambiar el ritmo, o viceversa.
Elimina sonidos estáticos, silbidos, zumbidos u otros ruidos de fondo constantes.
Modifica las frecuencias con la Ecualización, Filtro FFT, y los efectos Realce de Graves.
Ajusta los volumenes con Compresión.., Amplificar...y los efectos de Normalizado...
Otros efectos incorporados:
Eco...
Cambio de Fase...
Wahwah ...
Revertir...

Calidad de sonido

Graba y edita muestras de 16-bit, 24-bit y 32-bit(coma flotante).
Grabaciones de hasta 96 KHz.
Las frecuencias de muestreo y formatos son convertidos empleando remuestrado y fusionado de alta calidad.
Mezcla pistas con diferentes frecuencias de muestreo o formatos, y Audacity los convertirá automáticamente en tiempo real.

Plug-Ins

Añada nuevos efectos con LADSPA plugins.
Audacity incluye algunos plugins para muestras de Steve Harris.
Cargue plugins VST para Windows y Mac, con el opcional VST Enabler.
Escriba nuevos efectos con el lenguaje de programación interno Nyquist.

Análisis

Modo Espectrograma para la visualización de frecuencias.
Opción "Análisis de espectro..." para un análisis de frecuencia detallado.

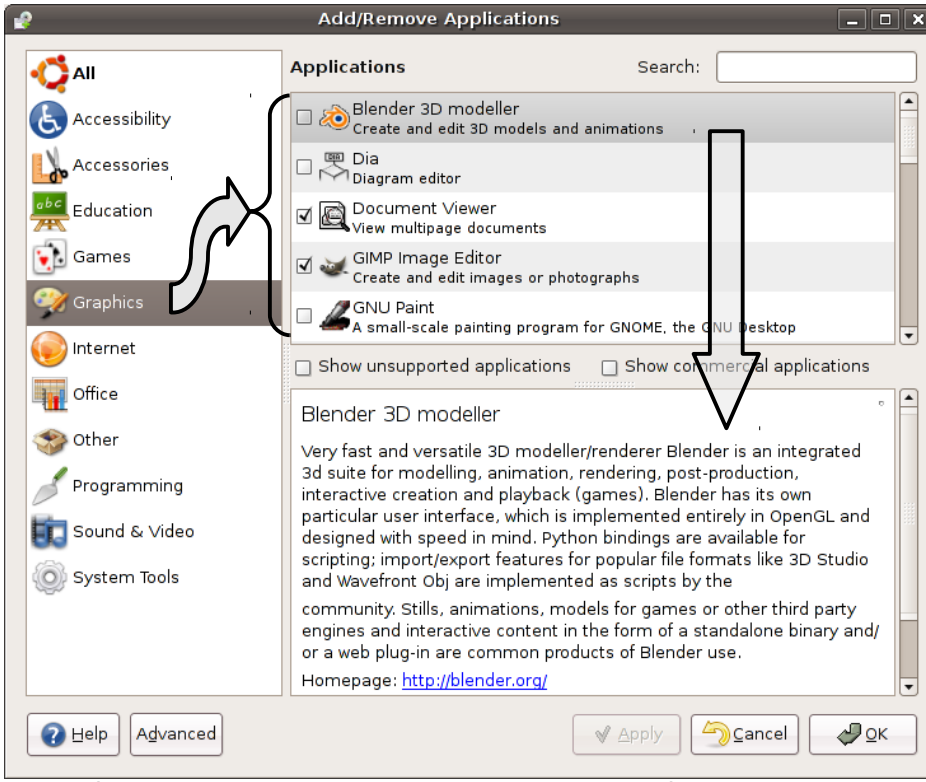
Libre y Multiplataforma

Licenciado bajo la GNU General Public License (GPL).
Corre sobre Mac OS X, Windows, y GNU/Linux.

Los desarrolladores de aplicaciones software libre tienden a construir plugins de extensibilidad como vía fundamental de escribir software porque ellos saben que sus herramientas nunca serán

capaces de hacer por si solas todas las cosas que las personas querrán hacer. Un plugin proporciona un límite entre las cosas que manejan los datos, y las cosas que los manipulan. Los plugins más populares finalmente se convierten en parte del sistema base, pero al ser construidos por separado, han obligado a limpiar dichos límites y la modularidad.[9]

Cada aplicación que Linux tiene y Windows no es una funcionalidad que en Windows no está presente:



La visión del software libre de Richard Stallman se dió cuenta de algo: Un sistema operativo Linux libre contiene todo un almacén de aplicaciones libres disponibles con solo un click, y desarrollados para trabajar en juntos.

9 Yo argumento en otra parte del libro que el software no tiene límites claros. Lo que quise decir era que verdaderamente uno nunca sabe precisamente cual debería ser la interfaz entre el gestor y el manipulador. Para archivos de audio, el límite parece claro: aquí hay algunos datos de audio, masticalos. Sin embargo aún tendrás que preguntar: ¿que APIs DSP están disponibles para los plugins? De lo contrario, !cada plugin necesitarán un montón de código duplicado que el gestor probablemente ya tiene! Es la capacidad del hardware nuevo la que crea la necesidad de hacer cambios en este límite. La lección aquí es mantener los límites simples, pero asumiendo que puede que necesites cambiarlos.

El tener tantas herramientas a tu disposición hace a las computadoras más personales, potentes, productivas y agradables. Tu experiencia con las computadoras se ve solo limitada por tu creatividad.

Un sistema operativo libre es donde el software cumple la ley de Metcalfe: a mayor cantidad de personas usando software libre, mayor cantidad de aplicaciones serán desarrollados para ello. Un único programa software libre no es útil, pero con todo un conjunto de ellos, podemos entrar en una era brillante.

Pasar de los 20 millones de usuarios de Linux que hay actualmente hasta los mil millones previstos significa un potencial de 50 veces mas recursos. El movimiento de software libre tiene varios retos que discutiré a lo largo de este libro, pero es importante mencionar aquí que pocas aplicaciones en el cuadro de diálogo están tan pulidas o fiables como Firefox. Sin embargo, muchas son muy potentes, y lo suficientemente buenas como para depender de ellas en un negocio.

Mientras Linux aún necesita trabajo, Windows no es un día en la playa. Aquí está un e-mail de Bill Gates contando su experiencia al instalar Microsoft MovieMaker en Windows:

From: Bill Gates

Sent: Miércoles, Enero 15, 2003 10:05 AM

To: Jim Allchin

Cc: Chris Jones (WINDOWS); Bharat Shah (NT); Joe Peterson; Will Poole; Brian Valentine; Anoop Gupta (RESEARCH)

Subject: Degradación de la Usabilidad de Windows

Estoy realmente decepcionado sobre como la usabilidad de Windows ha estado retrocediendo y los grupos de gestión de programas no se preocupan por los problemas de usabilidad.

Les contaré mi experiencia de ayer.

Decidí descargar (Moviemaker) y comprar el paquete Digital Plus ... por lo que fui a Microsoft.com. Tienen un sitio de descargas por lo que hacía allí fui.

Las primeras 5 veces que usé el sitio fallaba su carga por demora tratando de mostrarme la página de descarga. Luego de un retraso de 8 segundos pudo cargar la página para continuar.

El sitio es tan lento que es inutilizable.

No estaba en los primeros 5 por lo que expandí los otros 45.

Estos 45 nombre son totalmente confusos. Estos nombres hacen que cosas como: C:\Documents and Settings\billg\My Documents\My Pictures parezcan claras.

Ellos no son filtrados por el sistema ... y muchas de las cosas son extrañas.

Intenté filtrar las cosas de Media. Aún sin aparecer el moviemaker. Teclié movie. Nada. Escribí movie maker. Nada.

Entonces me dí por vencido y envié un correo a Amir diciéndole - ¿donde está la descarga del Moviemaker? ¿Existe?

Entonces ellos me dijeron que usar la página de descarga para descargar algo no era algo que habían previsto.

Ellos me dijeron que fuera al botón buscar de la página principal y tecleara movie maker (!No moviemaker!)

Intenté eso. El sitio estaba patéticamente lento pero luego de 6 segundos de espera por fin apareció.

Tomé por seguro ahora que vería el botón para hacer la descarga.

De hecho es como un rompecabezas que quieres resolver. Me dijo que fuera a Windows Update e hiciera un manojo de encantos.

Esto me pareció totalmente extraño. ¿Porque debería tener que ir a algún lugar más y realizar una búsqueda para descargar el Moviemaker?

Entonces fuí a Windows Update. Windows Update decidió que yo necesitaba descargar un montón de controles. (No) solo una vez sino varias veces donde conseguí ver extrañas cajas de diálogo.

¿No sabe Windows Update alguna forma de conversar con Windows?

Entonces hize la búsqueda. Esto tomó realmente bastante tiempo y me dijeron que era crucial para mi descargar 17 megas de cosas.

Esto es después de que me dijeron que estábamos haciendo parches delta para estas cosas, pero en cambio solo para conseguir 6 cosas que estaban etiquedas de la manera más espantosa posible tuve que descargar 17 megas.

Entonces realizé la descarga. Esa parte fue rápida. Entonces quiso hacer una instalación. Esto tomó 6 minutos y la máquina se puso tan lenta que no pude usarla para nada más durante ese tiempo.

¿Qué diablos estuvo pasando durante esos 6 minutos? Eso es una locura. Esto es luego de completada la descarga.

Luego me dijo que reiniciara mi computadora. ¿Porque debería hacer eso? Yo reinicio mi máquina todas las noches - ¿porque debería hacerlo en este momento?

Entonces reinicié la computadora porque el programa insistió en ello. Por supuesto eso significó deshacerme de todo mi estado de OutLook.

Entonces regresé y fui de nuevo a Windows Update. Olvidé por completo por qué estaba en Windows Update desde que todo lo que quería era conseguir el Moviemaker.

Entonces regresé a Microsoft.com y miré en las instrucciones. Tuve que dar click en una carpeta llamada WindowsXP. ¿Porque debería hacer eso? Windows Update sabe que estoy en una PC con Windows XP.

¿Qué significa tener que dar click en esa carpeta? Entonces conseguí un montón de cosas confusas pero lo suficientemente seguro que una de ellas era Moviemaker.

Entonces hago la descarga. La descarga es rápida pero la instalación toma muchos minutos. Es asombroso cuan lento es este proceso.

En algún momento me dicen que yo necesitaba conseguir Windows Media Series 9 para descargarlo.

Entonces decidí que haría eso. Esta vez obtuve diálogos diciendo cosas como "Abrir" o "Guardar". Sin guía en las instrucciones sobre qué hacer. No tenía idea sobre qué hacer.

La descarga es rápida y las instalación se toma 7 minutos para esto.

Entonces ahora creo que voy a tener Moviemaker. Voy hacia mi opción Agregar/Quitar programas para asegurar de que está allí.

No está allí.

¿Qué hay ahí? La siguiente basura está allí. El paquete de pruebas Microsoft Autoupdate Exclusive, el paquete de pruebas Microsoft Autoupdate Reboot, el paquete 1 de pruebas de Microsoft Autoupdate, el paquete 2 de pruebas de Microsoft Autoupdate, el paquete 3 de pruebas de Microsoft Autoupdate.

¿Alguien decidió botar en la basura la única parte de Windows que era útil? El sistema de archivos ya no es utilizable. El registro no es utilizable. Este listado de programas eran los únicos lugares sanos pero ahora está todo hechado a perder.

Pero esto es solo el comienzo de la basura. Después de que listé cosas como Windows XP Hotfix vi Q329048 para más información. ¿Qué es Q329048? ¿Por qué están estos seriales de parches listados aquí? Algunos de los parches son solo cosas como Q810655 en vez de decir ver Q329048 para más información.

Un absoluto enredo.

Moviemaker no estaba allí para nada.

Entonces desistí del Moviemaker y decidí descargar el Paquete Digital Plus.

Me dijeron que necesitaba entrar a un montón de información sobre mi mismo.

Escribí todo ahí y porque él decidió que yo me había equivocado escribiendo algo tuve que intentar de nuevo. Por supuesto, borró la mayor parte de lo que había escrito.

Intenté escribir las palabras correctas en 5 intentos y solo se seguían borrando las cosas para que yo las tecleara de nuevo.

> Así, después de más de una hora de locura y de hacer mi lista de programas basura y de comenzar a asustarme al ver que Microsoft.com es un sitio web terrible, que no pude conseguir el Moviemaker y que no conseguí tampoco el paquete plus.
>
> La falta de atención a la usabilidad representados por estas experiencias impactó mi mente. Pensé que nosotros habíamos alcanzado un pico bajo con los “sitios de red Windows” o con los mensajes que me aparecieron cuando intenté usar 802.11. (¿No les encanta el mensaje de certificado raíz?)
>
> Cuando realmente consiga usar esas cosas estoy seguro que tendré más comentarios.

Distribuciones de Linux

Con Linux, cada distribución del sistema operativo crea con gran esfuerzo un marco para conocer las necesidades de sus usuarios. Existen versiones especializadas de Linux conteniendo softwares educativos, herramientas para músicos, versiones dedicadas a dispositivos embebidos o de gama baja, y versiones regionales producidas en países como España y China.

Las diferentes distribuciones tienen mucho en común, incluyendo el kernel Linux, pero usan distintos softwares libres y mecanismos de instalación. Una distribución llamada Gentoo descarga solo un binario, un compilador de autoarranque. El resto de sus entregables son el código fuente de los componentes que ellos ofrecen. Esto da al usuario la posibilidad de construir un sistema altamente optimizado para su hardware.

Algunas distribuciones están optimizadas para ejecutarse sin problemas en equipos antiguos y que quepan en CDs con el tamaño de una tarjeta de crédito:

Damn Small Linux es el Linux más popular para computadoras antiguas y cabe en CDs de 80x60 mm.

Linux es muy popular en servidores, que requieren un atención adicional en el rendimiento, fiabilidad y seguridad. Una de las maneras más sencillas de disminuir la sobrecarga y los riesgos de seguridad es suprimir la interfaz gráfica:

```
top - 12:54:17 up 62 days, 20:14, 2 users,  load average: 0.16, 0.42, 0.43
Tasks: 127 total,   1 running, 126 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.2% us,  0.3% sy,  0.0% ni, 98.5% id,  0.2% wa,  0.7% hi,  0.2%
si
Mem:    514248k total,   489360k used,    24888k free,    79128k buffers
Swap:  1020088k total,    18416k used,  1001672k free,   177528k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2041 root      10  -5     0    0    0 S    0  0.0   0:11.74 usb-storage
    1 root      16   0  1564  528  460 S    0  0.1   0:01.09 init
    2 root      RT   0     0    0    0 S    0  0.0   0:00.01 migration/0
    3 root      34  19     0    0    0 S    0  0.0   0:14.63 ksoftirqd/0
    4 root      RT   0     0    0    0 S    0  0.0   0:00.00 watchdog/0
```

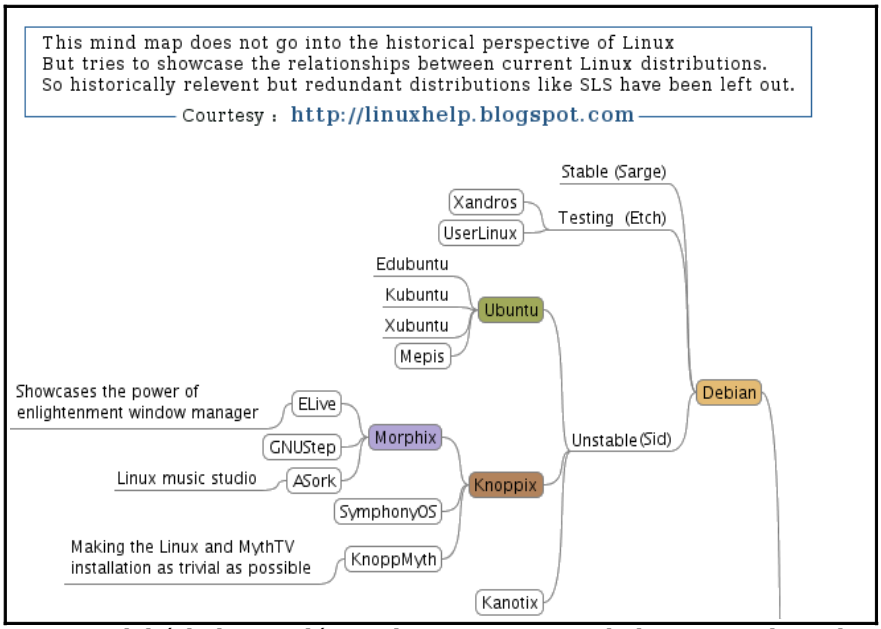
Captura de pantalla de un visor de procesos de Linux basado en texto. Solo porque la computadora muestre solo texto no siginifica que no es moderno, hay todo un sistema operativo sólido como las rocas debajo de eso.

Es tan difícil escribir interfaces gráficas de usuario (GUI) para un servidor como lo es escribir el servidor en si mismo: cada nueva funcionalidad en el servidor necesita la mejora correspondiente en la intefaz de administración. Si tu servidor puede generar 1,000 registros por segundo, entonces tu interfaz de usuario tiene que mejorar para ser capaz de manejar esa situación sin colapsar.[10]

Mientras que la mayoría de los dispositivos no necesitan un GUI, los usuarios de desktop si, y como todo en el mundo del software libre de hoy, existen varias buenas opciones.

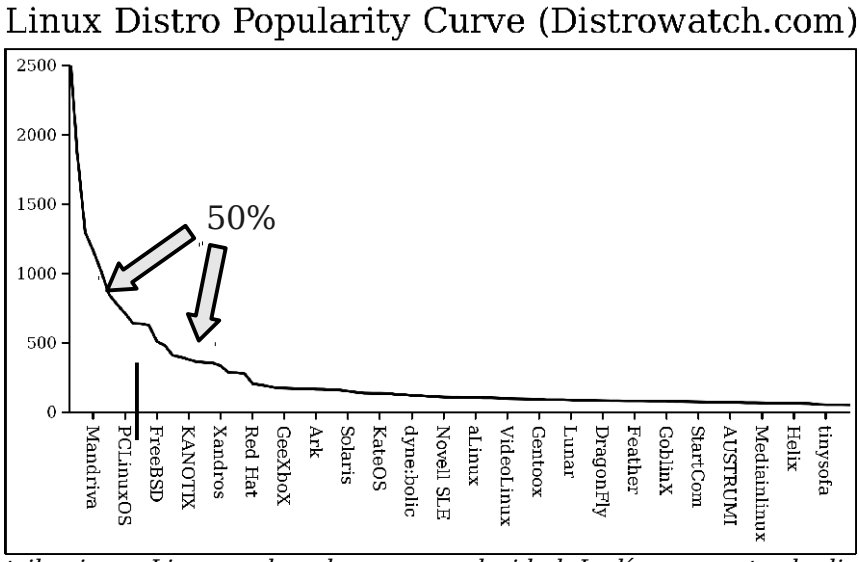
10 Muchos escenarios embebidos no incluyen GUI porque no existe aún ningún estandar. La mayoría de las GUIs son demasiado complicadas y lentas para escenarios embebidos. La caja registradora nueva en mi local de Starbucks tiene una pantalla táctil con una interfaz de texto.

El siguiente mapa muestra la rama Debian del árbol de familia de distribuciones Linux; ellas se derivan unos de otras, como en un ecosistema biológico:



Una parte del árbol genealógico de Linux mostrando la rama Debian, la mayor distribución software libre.

Aquí está un gráfico que muestra la popularidad relativa de las distribuciones Linux:



Distribuciones Linux, ordenadas por popularidad. La línea muestra la división entre ambas mitades de la curva de popularidad.

Lo que ves aquí es una curva casi perfectamente lisa que ilustra una idea realtivamente nueva llamada la "Larga Cola". Una manera de pensar sobre esta idea es mirar la lengua inglesa. Palabras como "the" son usadas con gran frecuencia, pero muchas más palabras como "teabag" son usadas con poca frecuencia. Hay una larga cola de palabras inglesas de poco uso, y simplemente ignorarlas sería desechar muchas de las que hacen que nuestro lenguaje sea tan especial.

La lección de la cola larga en los negocios es la importancia de atender a los clientes con intereses especiales. La larga cola de las distribuciones Linux significa que la creación de un ecosistema de software libre no significa el fin del libre mercado o de la competencia.

Wikipedia y el kernel Linux son dos de los mejores ejemplos del hecho de que el software libre y el libre intercambio de ideas pueden crear un producto superior sin costo por licencias. La mera existencia de estos productos de primera, sin una gigantesca compañía detrás de ellos, es un prueba de que el modelo de desarrollo privativo está sentenciado.

IA y Google

> El futuro esta en hacer todo código abierto
>
> —Linus Torvalds

> Ese conocimiento se ha convertido en *el* recurso, más que en *un* recurso, es lo que hace a nuestra sociedad pos-capitalista.
>
> —Peter Drucker, 1993

Imagínese 1,000 personas, separadas en grupos de a 5, trabajando en 200 enciclopedias separadas, en contra de ese mismo número de personas trabajando en una enciclopedia. ¿Cuál será la mejor? Esto suena como una tonta analogía cuando lo describimos en el contexto de una enciclopedia, pero es exactamente lo que está sucediendo con las investigaciones sobre la Inteligencia Artificial (IA) hoy en día.[1] Algunos dicen que el software libre no funciona en teoría, pero si que funciona en la practica. En verdad, "funciona" en proporción al número de personas que trabajan en conjunto, y su eficiencia colectiva.

En anteriores borradores de este libro, yo había puesto este capítulo detrás del que explicaba las implicaciones económicas y legales alrededor del software libre. Sin embargo, pienso que es importante discutir la inteligencia artificial primero y de forma separada, ya que la IA es el santo grial de la computación, y la razón por la que no hemos solucionado todavía la IA es porque no existen bases de código libre que hayan alcanzado su masa crítica. Más que suficientes personas existen allá afuera, pero generalmente se encuentran trabajando en equipos de una o dos personas, o en bases de código propietarias.

Deep Blue ha sido Profundamente-Sellada

> Algunas personas temen que la inteligencia artificial nos hará sentir inferiores, pero entonces, nadie en su sano juicio deberá sentir complejo de inferioridad cada vez que mira una flor.
>
> —Alan Kay, computer scientist

El código fuente de la Deep Blue de IBM, la primera máquina de ajedrez en derrotar al en aquel entonces Campeón Mundial Gary

1 Un sitio web documenta 60 piezas de código fuente que realiza transformaciones de Fourier, que es un importante bloque para la construcción de software. La situación es la misma para las redes neuronales, visión computacional, y muchas otras tecnologías avanzadas.

Kasparov, fue construido por un equipo de alrededor de cinco personas. Ese código ha estado lamentablemente en una bóveda en IBM desde el hecho debido a que no fue creado con una licencia que permitiera su uso posterior por cualquiera, ni siquiera cuando IBM no está intentando hacer dinero con este o usando su código para algo.

El segundo mejor motor de ajedrez del mundo, Deep Junior, tampoco es libre, y a su vez desarrollado por un equipo muy pequeño. Si solo tenemos pequeños equipos de personas atacando la IA, o escribiendo código y luego cerrándolo, no realizaremos progresos en aras de lograr software verdaderamente inteligente en un futuro cercano.

Las computadoras de ajedrez de hoy no tienen verdadera inteligencia artificial en ellas; simplemente realizan movimientos, y luego usan un análisis creado por humanos para medir el resultado. Si usted fuera a afinar el valor de la computadora de cuanto vale una reina comparado con un peón, la máquina comenzará a perder y ni siquiera entenderás porqué. Ella resulta ser inteligente solo porque tiene a expertos en ajedrez muy inteligentes programándola de forma precisa en cuanto a como analizar los movimientos, y evaluar la importancia relativa de las piezas y sus localizaciones, etc.

Deep Blue puede analizar doscientos millones de posiciones por segundo, en comparación con los grandes maestros que solo pueden analizar solo 3 por segundo. ¿Quién pudiera decir donde estaría ese código si los aficionados a la IA del ajedrez hubieran estado mejorándolo en los últimos 10 años?

Gran Desafío DARPA

> Los desarrolladores de software propietario tienes las ventajas que el dinero brinda; los desarrolladores de software libre necesitan darse ventaja unos a los otros. Yo espero que un día tengamos una gran colección de librerías libres que no tengan paralelo disponible para el software propietario, brindando módulos útiles que sirvan como bloques de construcción en el nuevo software libre, y adicionando una mayor ventaja al futuro desarrollo del software libre.
>
> ¿Qué es lo que necesita la sociedad? Necesita información que sea verdaderamente disponible para sus ciudadanos—por ejemplo, programas que las personas puedan leer, corregir, adaptar, y mejorar, no solo operar. Pero lo que los dueños de software típicamente entregan es una caja negra que no podemos estudiar o cambiar.
>
> —Richard Stallman

Los desafíos computacionales más difíciles que enfrentamos son hechos por el hombre: lenguajes, carreteras y correo basura. Tomen, por ejemplo, automóviles-robot. Podríamos hacer esto sin un sistema de visión, solo modificando cada carretera en el planeta adicionándole un raíl u otras guías para los automóviles-robot, pero es mucho más barato y seguro construir un software para que los autos viajen en las carreteras existentes hoy — un desastre caótico.

En la conferencia anual de la Asociación Americana para los Avances de la Ciencia (AAAS) en febrero del 2007, el “consenso” entre los científicos fue que tendremos automóviles sin chofer para el 2030. Esta predicción no tiene sentido ya que los que trabajan en el problema no lo hacen de manera conjunta. Además, como investigador estadounidense del cáncer Sidney Farber dijo, “Cualquier hombre que pronostique la fecha para un descubrimiento ya no es científico.”

Hoy, Lexus tiene un automóvil que se puede estacionar paralelamente el mismo, pero su sistema de visión solo necesita una idea muy vaga de los obstáculos que lo rodean para completar esta tarea. El desafío en construir automóviles-robot queda en crear un sistema de visión que se de cuenta de la líneas pintadas, los signos de la autopista, y otros obstáculos en la vía, incluyendo a los que no respetan “las reglas”.

La Agencia de Proyectos de Investigación Avanzada de Defensa (DARPA), que no como Al Gore, realmente inventó el Internet, ha patrocinado varios concursos para construir automóviles-robot:

Stanley, la entrada ganadora de la Universidad de Stanford para el desafío del 2005. Quizás no atropelle a un signo de Alto, pero no sabría que este significa parar.

Como el escenario de estacionar en paralelo, el Gran Desafío de DARPA del 2004 requería solo de un sistema simple de visión. Los automóviles competidores viajaban mayormente sobre una carretera limpia y donde se les proporcionaba un detallado conjunto de puntos de un mapa. Aun así, muchos de ellos no finalizaban, o funcionaban de manera confiable. Hay una expresión en ingeniería llamada "basura entrando, basura saliendo"; como esta, si un automóvil ve "muy poco", este no tendrá muchas esperanzas.

Lo que fue decepcionante acerca del primer desafío fue que una gran cantidad de software fue escrito para operar estos vehículos, ninguno de los cuales fue liberado (especialmente los sistemas de visión) para que otros lo criticaran, comentaran, mejoraran, etc. Yo visité el sitio del Stanley de Stanford y no pude encontrar ningún vínculo al código fuente, ni siquiera información acerca de en qué lenguaje fue escrito.

Algunos podrán preguntarse de porqué las personas deben trabajar en conjunto en un concurso, pero si todos los carros usaron gomas, procesadores de Intel y el núcleo de Linux, ¿Podría decir que no estuvieron compitiendo? Sí es una carrera, con el hardware más

rápido y estilo de manejo ganando al final. Trabajando juntos en una parte del software, los ingenieros pueden enfocarse más en el hardware, el cual es la parte divertida.

Lo siguiente es una descripción de la tubería de visión computacional requerida para operar de forma satisfactoria automóviles sin chofer. Mientras que el equipo de software del Stanley involucraba a solo 12 personas que trabajaban solo una parte del tiempo, el software de visión solo, es un problema tan complicado que tomará un esfuerzo comparado en complejidad con el núcleo de Linux para construir este:

Adquisición de Imágenes: Convertir la entrada de los sensores de 2 o más cámaras, radar, calor, etc. en una secuencia de imágenes tridimensional.

Pre-procesamiento: Reducción de ruido, mejorado del contraste

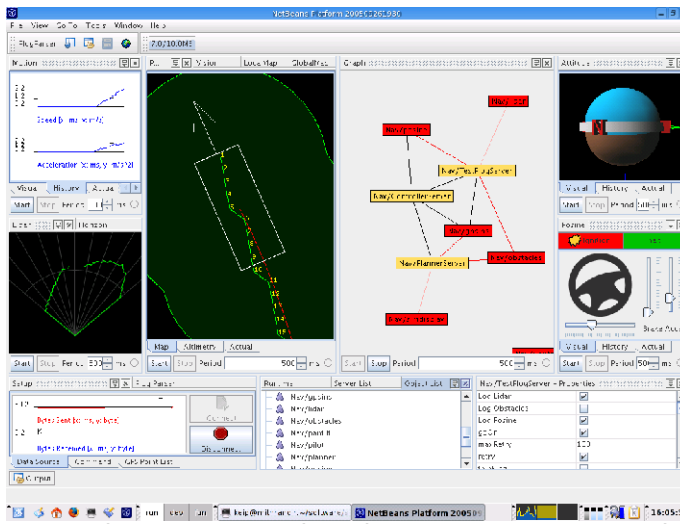
Extracción de características: Líneas, bordes, formas, movimiento

Detección/Segmentación: Encontrar porciones de las imágenes que necesiten un análisis posterior (signos de la autopista)

Procesamiento de Alto Nivel: Verificación de datos, reconocimiento de texto, análisis de objetos y categorización

Las 5 etapas de una tubería de reconocimiento de imágenes.

Una gran cantidad de software necesita ser escrito para soportar tal sistema:



La tubería de visión es la parte más difícil para crear un automóvil-robot, pero incluso este software de diagnóstico no es trivial.

En el 2007, hubo un desafío Urbano de DARPA. Este es un ejemplo de la información que se les brindó a los concursantes:

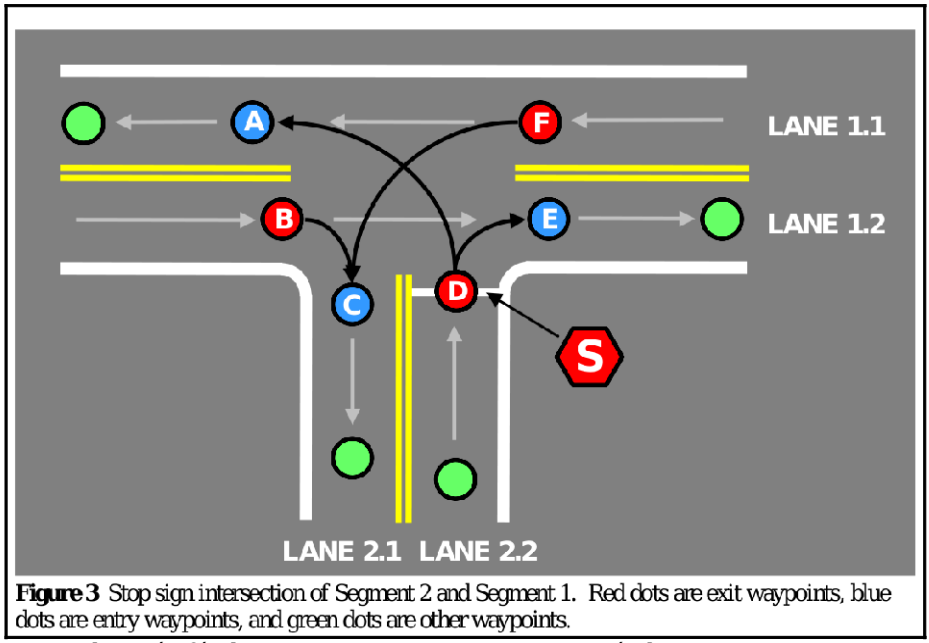


Figure 3 Stop sign intersection of Segment 2 and Segment 1. Red dots are exit waypoints, blue dots are entry waypoints, and green dots are other waypoints.

Es mucho más fácil y seguro programar un automóvil para que reconozca una señal de Alto que decirle la ubicación de todas ellas.

Construir una tubería de visión que pueda manejar en un ambiente urbano presenta un problema de software mucho más difícil. Sin embargo, si usted mira los requerimientos de visión necesarios para resolver el Desafío Urbano, está claro que reconocer formas y movimientos es todo lo que se necesita, ¡y esos son los mismos requerimientos que existieron en el desafío del 2004! Pero incluso desde el concurso del 2007, no ha existido más colaboración que en el concurso anterior.

Una vez que desarrollemos el sistema de visión, todo lo demás es técnicamente sencillo. Los juegos de vídeo contienen choferes controlados por computadoras que pueden correr mientras te disparan y ofenden. Su truco es que ellos ya poseen información detallada acerca de todos los objetos en su mundo simulado.

Luego de que hallamos construido el sistema de visión, existen aún muchos obstáculos interesantes a derribar: preparar audiencias del Congreso para discutir que estos automóviles deben tener un

límite de velocidad controlado por la computadora, o decirle a tu automóvil que no maneje de forma agresiva y derrame tu champaña, o probar y construir confiabilidad en ese sistema.[2]

Eventualmente, nuestras carreteras serán más inteligentes. Una vez que tengamos información del tráfico, tendremos computadoras que eficientemente nos digan una ruta para evitar la congestión. Un estudio encontró que los estancamientos del tráfico cuestan un billón de dólares al año en una ciudad grande promedio.

Ninguna organización hoy, incluyendo a Microsoft y Google, contiene cientos de expertos en visión computacional. ¿Usted piensa que GM (General Motors) será lo suficientemente valiente para crear un equipo de 100 expertos en visión aun si saben que podrían liderar este mercado?

Existen suficientes personas en todo el mundo trabajando en el problema de la visión en este momento. Si pudiéramos agrupar sus esfuerzos en una sola base de código, escrita en un lenguaje de programación moderno, podríamos tener automóviles-robot en cinco años. No es un asunto de invención, sino de ingeniería. Quizás el mundo solo necesita un Linus Torvalds de la visión computacional para hacer avanzar y liderar estos esfuerzos.

2 Existen varios asuntos acerca de la privacidad inherentes en los automóviles-robot. Cuando las computadoras sepan su localización, será muy fácil construir una “caja negra” que grabe toda la información e incluso la transmita al gobierno. Necesitamos estar seguros de que máquinas pertenecientes a un humano, permanezcan bajo su control, y no sean controladas por el gobierno sin una orden judicial y pruebas competentes.

Software y la Singularidad

Los futuristas hablan acerca de la “Singularidad”, como el momento cuando la capacidad computacional sobrepase la capacidad de inteligencia de los humanos. Ray Kurzweil predijo que sucederá en el 2045.[3] La falla con cualquier estimación de una fecha, otra diferente de su tendencia siempre a un error extremo, es que nuestro software de hoy no tiene capacidades de aprendizaje, debido a que la idea de un aprendizaje continuo no forma parte todavía de sus fundamentos.

Incluso las capacidades de aprendizaje de una hormiga serán útiles.

Yo creo que los beneficios inherentes a la singularidad ocurrirán tan pronto como nuestro software se convierta en “inteligente”. Yo no creo que debamos esperar por más progreso de la Ley de Moore para que esto ocurra. Las computadoras de hoy pueden realizar billones de operaciones por segundo, como adicionar 123,456,789 y 987,654,321. Incluso si usted puede realizar este cálculo en su cabeza en un segundo, le tomaría 30 años hacer el billón que su computadora realiza en un segundo.

---

3 Su predicción es que el número de computadoras, multiplicado por su capacidad computacional, sobrepasará el número de humanos, multiplicado por su capacidad computacional, en el 2045. Por lo que el mundo será fantástico cuando esto ocurra.

Estos cálculos son erróneos por varias razones:

1. Estaremos nadando en capacidad computacional mucho antes del 2040. Hoy en día, mi computadora típicamente corre al 2% de su CPU cuando la estoy usando, y además tiene 50 veces más capacidad computacional de la que yo necesito. Un agente inteligente con el doble de velocidad respecto a su predecesor no es necesariamente más útil.
2. Muchas de las neuronas en el cerebro no se gastan en el razonamiento, por lo que no necesitan estar en los cálculos.
3. Billones de humanos están meramente subsistiendo, y no están conectados a la red global, por lo no necesitan ser medidos.
4. No existe ninguna cantidad de aprendizaje continuo construido en el software de hoy.

Cada una de estas tiende a empujar la Singularidad hacia adelante y soportar el argumento que los beneficios de la singularidad no están esperando por el hardware. Los humanos hacen las computadoras más inteligentes, y este ciclo de retroalimentación hace que el 2045 sea un momento sin sentido.

Quién en el pasado no se ha preguntado: “¿Cuándo los hombres construirán un dispositivo que es mejor llevando las cosas que yo?” Las computadoras harán cualquier cosa, a cualquier hora, cuando se lo ordenemos. Una computadora juega ajedrez o reproduce música porque nosotros lo queremos. Un bombero robótico correrá a un edificio en llamas para salvar nuestras mascotas. Las computadoras no tienen propósito sin nosotros. Debemos preocuparnos acerca de robots matando a personas como nos preocupamos por alguien que se robe un helicóptero Apache y mate humanos hoy en día.

Incluso si usted no piensa que las computadoras tienen hardware suficientemente poderoso para ser inteligentes hoy, entienda que en muchos escenarios, el tamaño de la entrada es el factor que lleva al poder de procesamiento requerido. En reconocimiento de imágenes, por ejemplo, la cantidad de trabajo requerido para interpretar una imagen se encuentra mayormente en función al tamaño de esta. Cada paso en la tubería de reconocimiento de imágenes, y los procesos que toman parte en nuestro cerebro, reducen notablemente la cantidad de datos del paso anterior. Al comenzar un análisis podría haber una imagen de un millón de píxeles, requiriendo 3 millones de bytes de memoria. Al finalizar el análisis si los datos que tu estas mirando pertenecen a tu casa, el cuál es un concepto que requiere solo 10 bytes para representarse. El primero paso, trabajando en la imagen cruda, requiere el mayor poder de procesamiento, por lo que es la resolución de la imagen (y la taza de cuadros por segundo) son los que establecen los requerimientos, valores que son triviales de cambiar. ¡Nadie ha mostrado un software de reconocimiento de visión robusto corriendo a cualquier velocidad, para cualquier tamaño de imagen!

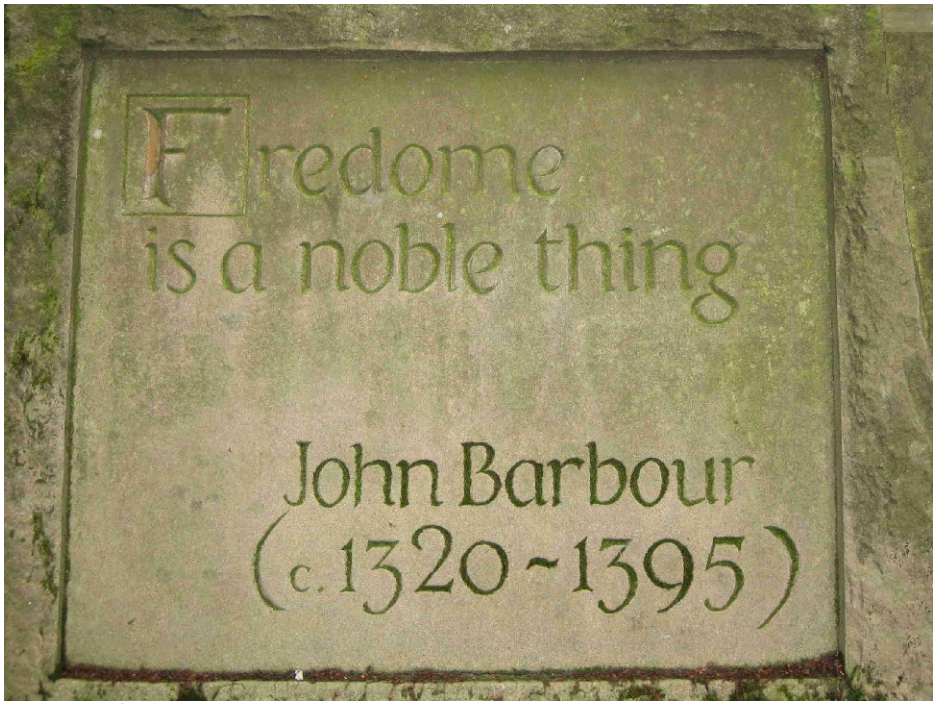
Mientras que el cerebro humano es diferente a una computadora en el hecho que realiza trabajo en paralelo, esta paralelización solo tiene sentido si ocurre rápidamente, y no cambia el resultado. Cualquier cosa lograda por nuestros cerebros paralelos puede ser también logrado por las computadoras de hoy, las cuales solo pueden hacer solo una cosa a la vez, pero al ritmo de billones por segundo. Un procesador de 1 GHz puede realizar 1,000 de diferentes operaciones en un millón de datos en un segundo. ¡Con esa velocidad, usted no necesita ni siquiera varios procesadores! Aún así más paralelismo viene llegando.[4] Una vez que construyamos software tan inteligente como una hormiga, construiremos software tan inteligente como los humanos el mismo día, debido a que es el mismo software.

4 La mayoría de las computadoras hoy tienen una CPU de doble núcleo y los fabricantes de procesadores prometen que 10 y más vienen llegando. Los procesadores de Intel tienen además un límite en su capacidad de procesamiento en paralelo de 4 vías conocidas como MMX y SSE. Intel podría adicionar mucho más soporte a este procesamiento en paralelo si las aplicaciones le dieran más uso a estas.

Software Libre

Si tu tienes una manzana y yo tengo una manzana y si intercambiamos esas manzanas entonces tu y yo seguiremos teniendo una manzana cada uno. Pero si tu tienes una idea y yo tengo una idea e intercambiamos esas ideas, entonces cada uno de nosotros tendrá dos ideas.

—George Bernard Shaw



Inscripción que se encuentra en una muralla en Edimburgo

Gran parte de la existencia del hombre hasta el tardío siglo 20 se concentraba en una economía enfocada principalmente en la manipulación de escasos y no maleables átomos. Las leyes del derecho de autor (Copyright) fueron creadas para proteger a los escritores de las editoriales. En un mundo digital, todos nosotros podemos ser escritores y editoriales a la vez, por lo que debemos revisar muchas preguntas fundamentales, desde los medios que utilizamos para proteger las ideas, hasta las vías que utilizamos para comprar estas.

Este capítulo y el siguiente discuten detalles del software libre, derechos de autor y leyes de patentes, pero comencemos recordando que el movimiento hacia asumir la existencia de información

digital libre es realmente una pregunta moral. El Internet hace que transmitir el conocimiento sea libre, así que como evangelista del software libre Eben Moglen preguntó: "¿Si tu pudieras alimentar el mundo de forma gratis, lo harías? Así mismo, ¿si tú pudieras proveer a cada niño de acceso a una biblioteca de conocimiento humano para eliminar sus límites, lo harías?" Es el Internet el que hace posible responder esta pregunta, y sea necesaria esta respuesta.

Sin el software indicado para decodificarla y manipularla, una idea digital es solo un grupo de bits en tu computadora. Con el Internet, podemos intercambiar bits, pero con el software libre, podemos intercambiar ideas. Mientras que el conocimiento libre y el software libre no son un objetivo directo del mercado libre, ellos proveen de tremendos beneficios al mercado libre ya que ellos permiten a *cualquiera* darle un valor adicional a este. Si el mayor objetivo es alentar tantos programadores como sea posible a escribir software, entonces la vía del software libre ya ha demostrado su superioridad, incluso cuando tiene solo el 1% de las computadoras de escritorio. De hecho, el mundo propietario de las PC siempre estuvo destinado a que fuera dominado por una sola compañía; un clon de Bill Gates que hubiera venido después no hubiera podido aprender o mejorar las innovaciones del primero.

El software libre trae consigo el beneficio libertario de permitir que la información se utilice de ilimitadas nuevas maneras, combinadas con el beneficio comunitario de asegurarse que nadie sea dejado atrás por el costo de acceso al conocimiento. Debido a que el "software libre" es mejor para el mercado libre que el software propietario, y un importante elemento de la sociedad caracterizada por el intercambio libre de ideas, yo creo que es un mejor nombre que "código abierto", incluso cuando ambos representan la misma idea básica. (Otra razón por la que llamarlo "software libre" es que existe una tradición académica que la persona que descubra o defina algo tiene el derecho de darle un nombre, y Richard Stallman definió "software libre" mucho antes de que otros lo llamaran "código abierto".)

Software como Ciencia

> En cualquier campo intelectual, uno puede alcanzar mayores alturas estando en los hombros de otros. Pero esto generalmente ya no se permite en el campo del software propietario—usted puede estar solo en los hombres de personas *de tu propia compañía*.
>
> El daño psicosocial asociado afecta el espíritu de la cooperación científica, la cual puede ser usada para que sea tan fuerte que los científicos cooperen aún incluso cuando sus países estén en guerra. En este espíritu, oceanógrafos japoneses abandonaron su laboratorio en una isla del Pacífico preservando cuidadosamente su trabajo para los invasores Marines estadounidenses, y dejaron una nota pidiéndoles que cuidaran bien de este.
>
> —Richard Stallman

El software es una ciencia: usted razona, usted crea una hipótesis en la forma de código fuente, y entonces tu pruebas tu hipótesis corriendo este en una computadora. Algunos piensan que el software es un arte ya que un software bien escrito le da cierta elegancia a este, pero la elegancia es simplemente la personificación del "Tan simple como sea posible, pero no simple" de Einstein.

Linus Torvalds resumió las similitudes entre el software libre y la ciencia en lo siguiente:

> La ciencia puede tomar algunos cientos de años en darse cuenta como funciona el mundo, pero esta llega a esto, exactamente debido a que personas pueden construir a partir del conocimiento de ellos mismos, y evoluciona a través del tiempo. En contraste, brujería/alquimia puede ser acerca de personas inteligentes, pero el cuerpo del conocimiento no se "acumuló" en ningún lado. Este puede ser transmitido a un aprendiz, pero el ocultamiento de la información significa básicamente que esta nunca podrá ser mejor que lo que una simple persona/compañía pueda entender.
>
> Y esto es exactamente el mismo problema con el código abierto (libre) vs productos propietarios. Las personas propietarias podrán diseñar algo que es inteligente, pero eventualmente se convertirá en algo muy complicado para que una sola entidad (incluso una gran compañía) pueda entender y manejar, y las políticas de la compañía y sus objetivos siempre limitarán a este.

Incluso la palabra "universidad", el lugar de los hombres para estudiar de forma compartida, se deriva del latín *universitas magistrorum et scholarium*, lo que significa "una comunidad de profesores y estudiantes." Las universidades fueron siempre comprendidas como lugares donde se agrupan personas para que aprendan unas de otras.

Desafortunadamente, hoy, el software propietario se ha esparcido desde el mundo corporativo hasta las universidades y otras instituciones públicas. Si las corporaciones quieren salvaguardar sus avances científicos, está bien, aunque hacen notar la vista corta de estas, pero otras instituciones públicas ¡no deberían seguir este ejemplo! No solo el automóvil-robot de Stanford, Stanley, otra *gran cantidad* de software propietario es escrito por instituciones públicas hoy en día. Solo desencadenando nuestras instituciones públicas hacia el software libre habrá un gran aumento del progreso, sin ni siquiera tener en cuenta el software hecho por corporaciones.

Algunos piensan en el software libre como una idea Marxista, pero la ciencia siempre ha sido pública ya que se ha entendido como ciencia, que cualquier hecho que descubramos debe ser reproducible por otros bajo las mismas condiciones. Dar el conocimiento a otros generará aun mas descubrimientos, y esto es debido a que los científicos necesitan algunos hombros en los que descansar en primer lugar. Las corporaciones no fueron creadas para acaparar conocimiento sino para tomar los avances de la ciencia y aplicarlos a usos prácticos. Aún existe una gran cantidad de oportunidades de competencia y libre mercado, incluso cuando los avances de la ciencia estén disponibles libremente.

El trabajo científico es revisado por otros científicos, publicados en revistas, y discutido en conferencias; el intercambio de ideas entre los participantes en fundamental para el avance de la ciencia. Esto incluso no entra en conflicto con la competitiva naturaleza de los humanos: en ser el primero en descubrir algo, o poner los avances a usos comerciales.

Con el software propietario, hemos creado algo fuera del modelo clásico de investigación científica, y aun peor, se ha convertido en un modelo muy dominante en instituciones privadas y públicas. Dejar que una compañía se apropie de una innovación científica quizás le de a esta un incentivo a elaborar mejores productos, pero limita a otras muchas personas que pudieran usar esa innovación para otros fines.

La ciencia no es todo el conocimiento. La ciencia no es un negocio, un servicio, o un producto. Y para el producto que utiliza ciencia para su realización, fueron creadas las patentes para proteger ideas que pueden ser fácilmente copiadas pero no fácilmente inventadas o descubiertas.

Uno podría incluso argumentar que las patentes no son necesarias para estimular el progreso ya que el desafío para alguien que va

en contra de la idea del bombillo de GE es la infraestructura de producción y distribución de este producto, y el conocimiento que se obtiene de todo esto. Una vieja frase en los negocios dice que aquel que gana es “el primero con todo.” Grandes empresas tienen economías de gran escala y clientes satisfechos, hechos al que cualquiera que empiece en el negocio tendrá que sobreponerse, incluso si estos tienen un mejor producto.

Economías de gran escala son buenas para el libre mercado ya que ellos son un poderoso controlador de eficiencia incrementada y calidad, pero eso también significa que para vencer a alguien tendrás que ser un 10% mejor que este. La necesidad de avances para vencer competidores es bueno para el libre mercado ya que fuerza a los que entran en este a pensar en grande. Afortunadamente, las tecnologías transformativas se descubren tan a menudo que ningún competidor está siempre completamente seguro.

Debido a que el software es una ciencia, hacer este disponible de forma libre puede dañar a las compañías de software propietario, pero ayudará a cualquier otro tipo de compañía.

Definición del Software Libre

> Nada hoy, como nada desde que dominamos el fuego, es genuinamente nuevo: La Cultura, como la ciencia y la tecnología, crece por adición, cada nuevo creador se desarrolla en base al trabajo de los que le precedieron.
>
> —Juez Alex Kozinski, Corte de Apelaciones de los EEUU

> Hacer que Linux usara la licencia GPL fue definitivamente la mejor cosa que hice.
>
> —Linus Torvalds

Debido a que el software es una ciencia, como sociedad necesitamos crear licencias que permitan e incluso fomenten la cooperación entre los programadores. Los científicos de la computación necesitan de software que se encuentre disponible libremente para que puedan hacer su trabajo.

Richard Stallman definió las cuatro libertades de un software:

1. La libertad de ejecutar el programa, para cualquier objetivo. (Usted, no tu software, está en control de lo que está ocurriendo)
2. La libertad de estudiar como funciona el programa y adaptarlo a tus necesidades.
3. La libertad de darle una copia del programa a tu vecino. Compartir ideas es imposible sin compartir el programa que las crea y muestra.
4. La libertad de mejorar el programa, y liberar tus mejoras al público, de forma que se beneficie toda la comunidad.

La Licencia Pública General (GPL) de GNU es el mecanismo de derechos de autor que el planteó para proteger estas libertades, la cual permite un reuso máximo de los avances. El objetivo es proteger la libertad del usuario de software libre. Sin la posibilidad de estudiar y manipular el software, usted tendrá una gran limitación a la hora de utilizar y mejorar a este.

El Copyright fue creado para proteger a los creadores de los publicadores (aquellos que tienen los medios para hacer copias), garantizándole al creador derechos exclusivos. La GPL de GNU es llamada algunas veces “copyleft”, debido a que garantiza los mismos derechos expansivos a cualquiera, creador, y usuario.

Proteger a los usuarios y no al creador suena al revés, pero protegiendo a los usuarios también ayuda a los creadores. Todos los programadores de Linux excepto Linus comenzaron como usuarios. Linus fue el primer usuario de Linux como fue también el primer contribuidor de Linux. A el pertenecía todo su código y podía arre-

glar cualquier problema que encontrara. Copyleft se asegura que el código protegido por esta licencia provee de las mismas garantías a los futuros usuarios y creadores.

El Copyleft ayudó a Linus debido a que este requería programadores que mejoraran el Linux y devolvieran estas mejoras a este. Linux solo corría en una CPU 80386 cuando se lanza por primera vez ya que era esta la que Linux poseía. Todas las mejoras que tomaron para que este corriera en otros procesadores fueron puestas en Linux, para el beneficio de Linus, y el resto del mundo.

Cualquiera que sea tu pensamiento acerca del software libre hoy, usar este es una elección. De hecho, crear este es un acto caritativo, y todos debemos estar agradecidos a Linus Torvalds por liberar su trabajo, como mismo debemos estar agradecidos a Einstein por hacer su $E = mc^2$ públicamente disponible y no quedara solamente en una pizarra.

Microsoft, Apple, Google, y muchas de las otras compañías de computadoras de chip-azul no aceptan todavía la idea de que el software debe ser libre por las vías definidas por Stallman anteriormente. Según ellos, usted puede generalmente ejecutar el código con cualquier objetivo, pero no copiar, estudiar, o mejorar este.

Copyleft y el Capitalismo

Con las libertades de la GPL viene una importante responsabilidad: asegurarse que las mejoras futuras al código libre queden tan libres como el código original. Esto alienta a la comunidad científica a quedarse en esta vía, pero es una responsabilidad solo para el grupo de usuarios que después eligen en convertirse en creadores.

El software protegido por el copyleft es a menudo llamado software libre, pero este no es realmente libre debido a que el sistema de retroalimentación que se asegura que futuras mejoras a este también sean libres. No existe un almuerzo gratis en este mundo, y el copyleft se asegura que las personas están devolviendo las mejoras que realizan. La obligación del copyleft es necesaria pero no cara.

Necesario

La razón por la que es necesario tener un copyleft es que solo el llamado 100% software libre es aquel que otra persona puede mejorar. Mejorar un pedazo de software libre, y hacer estas mejoras propietarias, efectivamente hacen del pedazo completo del software propietario. Usted necesita acceso a la base de código completa

para poder realizar cambios en cualquier lugar de esta. Negar la libre disponibilidad de las mejoras realizadas al código crea nuevas fronteras entre la ciencia y la alquimia.

No Caro

> Un efectivo y equitativo sistema de propiedad intelectual debe tener en cuenta tanto los que crean al principio como los que vendrán después a construir a partir del trabajo de estos.
>
> —Comité para el Desarrollo Económico

El software libre no es caro, debido a que en términos prácticos, los avances en el software generalmente son basados en conocimiento existente y tienen un objetivo bastante pequeño. Lo que usted debe dar es mucho más pequeño de lo que ha recibido libremente.

Cuando Linus liberó Linux, este tan solo tenía 10,000 líneas de código. Desde esto el ha recibido 8.2 millones de líneas en respuesta, que significan 4,000 años-hombre de trabajo, o 400 millones de dólares. El software libre, como ciencia, es un esquema de pirámide que funciona. La idea de que alguien te de 400 millones de dólares en tecnología de forma gratis significa que es muy probable que no tengas que hacer mucho para que este trabaje para ti, y las obligaciones de tu copyleft serán solo una pequeña fracción de lo que has recibido de forma totalmente gratis.

Un cambio en el software es generalmente de 10 a 100 líneas de código, lo que significa el 0.001% al 0.01% de una base de código de un millón de líneas. La tecnología avanza de forma estable, paso a paso. Si embargo, es importante que una vez que una pieza de software es libre, cada pequeño avance a esta también sea libre. Tanto la Wikipedia como Linux requieren de la obligación del copyleft, y yo creo que es una parte importante de su éxito. Ellos también demuestran que la obligación del copyleft es suficiente para el desarrollo sostenible del software libre, sin involucrar tarifas de licencia.

El software libre pone a otras personas como beneficiarios de tus soluciones, así estos pueden enfocarse en otros problemas, para tu beneficio. El software libre permite una división óptima del trabajo, realizado por personas que no necesariamente necesitan entender todo el cuadro, pero que están tratando que el software funcione mejor para ellos, o para un cliente de ellos.

Una licencia libre no te quita tu habilidad de beneficiarte con tu trabajo. Y mientras esta no te quita la habilidad de beneficiarte, siempre existirá motivación para escribir software libre. Como mismo la disponibilidad pública de los avances científicos no han

disminuido la motivación para hacer ciencia, el flujo libre de innovaciones de software no amenaza con minar el mercado de software. Si las personas solo hacen cambios al software cuando ellos lo desean, y estos cambios son capturados por otros para su uso, nadie es dejado atrás. Esto ayuda a otros, y muy probablemente ayude al creador ya que escribir software es un trabajo duro.

Es el Copyleft un Requerimiento para el Software Libre?

El copyleft de Richard Stallman, la idea de asegurarse que la ciencia libre permanezca libre, es uno de los conceptos más únicos en la era digital. Es una idea tan radical que no es aceptada universalmente, incluso en la comunidad del software libre.

De hecho, el código fuente es considerado libre por la comunidad del software hoy si soporta las tres primeras libertades (ejecución, estudio, copiado), pero no el copyleft (hacer que las mejoras sen libremente disponibles para todos.)

Dos licencias libres muy populares, las licencias MIT y BSD son consideradas libres pero estas simplemente dicen: “Por favor incluya este mensaje de copyright al principio de su código fuente.” Usted puede usar este código, copiarlo, estudiarlo, pero también puedes que este sea propietario otra vez. Este tipo de software libre no requiere que las mejoras se retornen a este de forma libre.

Stallman considera a estas licencias como flojas; mientras que estas suenan razonables y hablan estrictamente, “mas libre” que el copyleft, el problema es que este software pobremente protegido frecuentemente se vuelve propietario otra vez. Keith Packard ha contado como el sistema de ventanas de Unix fue creado inicialmente con una licencia floja, pero fue reescrito en múltiples ocasiones debido a que fue pirateado y hecho propietario en varias ocasiones. Una enorme cantidad de trabajo de programación fue gastado debido a que la base de código no fue GPL desde el mismo comienzo.

Una de las razones debido a por qué Unix nunca fue mucha competencia para Windows es que muchas de las compañías que desarrollaban para este no trabajaban en conjunto. La licencia GPL de Linux une a las personas a que trabajen en conjunto para ahorrar dinero en costos de desarrollo total y velocidad de progreso.

Algunos argumentan que las licencias flojas “dan menos miedo” a las organizaciones que no entienden o aprecian realmente el soft-

ware libre. Este problema puede ser resuelto por una mejor educación para la comunidad computacional, no alentando a acuerdos de licencia con gente ignorante. Como Eben Moglen apuntó, las cosas en dominio público pueden ser apropiadas desde maneras libre-discrepantes. En general, una vez que las personas entiendan al software como ciencia, la idea de habilitar la ciencia de forma propietaria no va a ser interesante. La consecuente adopción del copyleft incrementará la eficiencia de la comunidad del software libre y ayudará a alcanzar la dominación global más rápido.

Estándares y la Web

De: Bill Gates
Enviado: Sábado, 5 de Diciembre, 1998
Para: Bob Muglia, Jon DeVann, Steven Sinofsky
Asunto: Interpretación de Office

Una cosa que tenemos que cambiar en nuestra estrategia - permitiendo que los documentos de Office sean interpretados muy bien por los navegadores de otras personas es una de las cosas más destructivas que podemos hacer a la empresa.

Tenemos que dejar de poner cualquier esfuerzo en esto y asegurarse de que los documentos de Office dependan muy bien de las capacidades de PROPIEDAD IE.

Todo lo demás es un suicidio para nuestra plataforma. Este es un caso en que Office tiene que evitar hacer algo para destruir a Windows.

Yo estaría encantado de explicar en una mayor profundidad.

Asimismo este amor al estándar DAV en Office / Exchange es un gran problema. También me gustaría de asegurarme de que las personas entendieran esto.

Para utilizar la Internet, usted necesita un software que soporte dos grandes estándares: TCP / HTTP y HTML. No hay estándar “HTML“ compitiendo con un estándar “HTMM“, ya que la idea sería tonta, pero estas redundancias existen en muchas otras áreas en el mundo de los bits en la actualidad. Cuando usted no puede ponerse de acuerdo sobre un formato de archivo, su capacidad de intercambio de información va de 1 a 0.

El software libre ha sido una parte de la Internet desde el principio. De hecho, un sitio web necesita enviarle un software, en forma de HTML y JavaScript, para que usted pueda tener algo que ver e interactuar con él. Es fácil de aprender como un sitio web hace su magia, como las etiquetas de un documento HTML se describen a sí mismas y, por encima de eso, existe una organización llamada W3C cuyo trabajo es plenamente describirlos.

En contraste, para mostrar un documento de Word o WordPerfect, usted tiene que hacerle ingeniería inversa a un complicado formato de archivo binario! Esto impidió su uso generalizado como el formato de documento estándar de Internet.

Imágenes Digitales

A diferencia del audio y el vídeo, en el reino de las imágenes fijas las cosas están en buena forma. JPEG es un eficiente, libre y ampliamente apoyado estándar para la compresión de imágenes.[1] Podría haber un par de mejores estándares que el JPEG por ahí, pero el final está cerca. (Hay un estándar JPEG 2000 sobre la base de wavelets[2] que es 20% mejor que el JPEG, pero tiene demandas de memoria y de procesamiento superiores. A veces para conseguir un poco más de compresión, se tiene que hacer mucho más trabajo, y así llegar a un punto de rendimientos decrecientes.)

Microsoft anunció a principios de 2007 un nuevo formato Windows Media Photo, que también es 20% mejor que el JPEG, pero sin demandas de memoria y de procesamiento tan altas, como los exigidos por JPEG 2000. El nuevo formato está basado en JPEG, pero con nueve pequeños retoques. La especificación es pública, y hay incluso código fuente libre público, pero la licencia excluye expresamente que se utilice en combinación con licencias copyleft:

> 2. c. Restricciones de distribución. Usted no puede ... modificar o distribuir el código fuente de cualquier Código Distribuible de modo que una parte del mismo quede sujeta a una Licencia de Exclusión. Una Licencia de Exclusión es una que requiere, como condición de uso, modificación o distribución, que el código sea divulgado o distribuido en forma de código fuente o que otros tengan el derecho a modificarlo.

Afortunadamente, la ingeniería inversa es fácilmente factible debido a que la especificación es pública y si se comienza con el código JPEG existente ya usted está bastante cerca - un pequeño y oculto freno que Microsoft ha creado para la comunidad de software libre.

---

1 También hay PNG y GIF para la compresión sin pérdidas, pero no son adecuados para imágenes del mundo real con gradientes continuos, como las nubes, etc. Tomé una JPEG de 1,9 MB de alta calidad y la convertí a PNG y llegó a ser 2,9 veces mayor. Curiosamente, estos formatos sin pérdida pueden hacer un mejor trabajo que JPEG para imágenes determinadas, como capturas de pantalla ya que JPEG no soporta las transiciones de agudos de negro a blanco, etc. que se encuentran en una pantalla de ordenador. Un JPEG de una captura de pantalla es 2,2 veces más grande que un PNG equivalente. Con un JPEG del mismo tamaño, encontraras que ha añadido artefactos de pantalla grises en las transiciones negro/blanco. PNG fue creado sólo porque después de que se hizo popular GIF, Compuserve comenzó a demandar.

2 Wikipedia: “Un wavelet es una especie de función matemática utilizada para dividir una determinada función o señales de tiempo continuo en componentes de frecuencia diferentes y estudiar cada componente, con una resolución que coincida con su escala.“

Audio Digital

Es el desastre de los estándares propietarios y las restricciones de patentes que impiden el progreso del audio digital. Las compañías de software propietario han estado introduciendo sus estándares por nuestra garganta. Si inserta un CD de música en un equipo Windows, este quiere extraer el audio en WMA, un formato propietario, que Mac OS no soporta de ninguna manera. Si inserta ese CD en un Mac, este extrae la música en AAC, un formato que Windows no soporta por defecto. No estoy seguro de lo que debería hacerse acerca de este desastre colosal. Debemos seguir tratando de elegir un formato estándar para audio, como se ha convertido para las imágenes fijas JPEG. Los mejores candidatos son OGG y MP3. MP3 es un formato antiguo y, si bien no se considera estado del arte, es lo suficientemente eficaz. El problema principal de MP3 es que hay un número de empresas con reclamaciones de patente en contra de este. Si la industria no puede ponerse de acuerdo para poner fin a los pleitos de las patentes, debe adoptar OGG o algún otro formato totalmente libre y decir a esos propietarios de licencias de MP3 que se tomen una larga caminata a algún muelle.

Si finalmente pudiéramos ponernos de acuerdo sobre un formato libre de música digital, finalmente podríamos tener música digital - lo que también abriría muchas posibilidades de una rica relación entre el artista y el consumidor. Independientemente del formato elegido, es necesario definir el número de bits por segundo necesarios para lograr la transparencia.[3] También tenemos que elaborar un protocolo de transmisión estándar y el formato de vídeo, pero ni siquiera voy a entrar en eso aquí, tenemos que gatear antes de poder caminar.

El Desastre de la Próxima Generación del DVD

Al igual que muchas otras cosas en la industria informática, el formato de próxima generación para DVDs de alta definición fue un desastre por varios años debido a que dos estándares fueron crea-

3 La transparencia se define como audio que es de una calidad tan alta que no se puede distinguir de un CD. La transparencia *debe* significar pasar el test de audición antes mencionados, además de que al convertirlo a otros códecs transparentes y de nuevo al anterior, incluso 100 veces, la calidad no disminuya. Para lograr que no sea necesario ir a la compresión sin pérdida, que es de cinco a siete veces más grande.

dos: HD-DVD y Blu-Ray. Ambos formatos son de alta calidad y sus discos tienen el mismo aspecto, pero el hardware de reproducción es inexplicablemente incompatible.

Afortunadamente, a principios de 2008, HD-DVD fue abandonado, aunque miles de millones de dólares fueron malgastados, porque los dos bandos peleaban como hermanas gemelas, y no pudieron ponerse de acuerdo sobre algunos detalles técnicos minúsculos antes de salir al mercado. La mala noticia es que la adopción de Blu-Ray es lento. A mediados de 2009, se informó de que los reproductores HD-DVD son 60% más popular que los Blu-Ray, a pesar de que había sido abandonado por la industria. Además, ¿por qué pagar $ 300 por un nuevo reproductor cuando todas las películas que ya poseemos en nuestro poder no están en dicho formato? ¿Por qué debemos pagar el precio completo sólo para obtener una copia de mayor calidad de algo que ya tienes? ¿No queríamos decir en realidad comprar una copia de alta calidad en primer lugar? La mejor manera para una adopción más rápida sería la de crear un servicio de correo en donde su viejo VHS o DVD es enviado, y te enviaran una copia de las versiones HD de ellos por unos pocos dólares cada uno. Eso podría ser un enorme negocio , eso sí, de bajo margen de beneficios.

Los consumidores se adaptarían más rápido a los nuevos estándares si inmediatamente pudiesen disfrutar de todo lo que ya poseen en ese nuevo formato en algo que se acerca al coste real de producir un disco, en lugar del precio de venta total. Esta es otra área donde la reducción de la expiración del copyright será de ayuda. Hará casi gratis todas esas cosas por las que pagamos una cuota de licencia, pero de las que ahora sólo tenemos reproducciones de baja calidad.

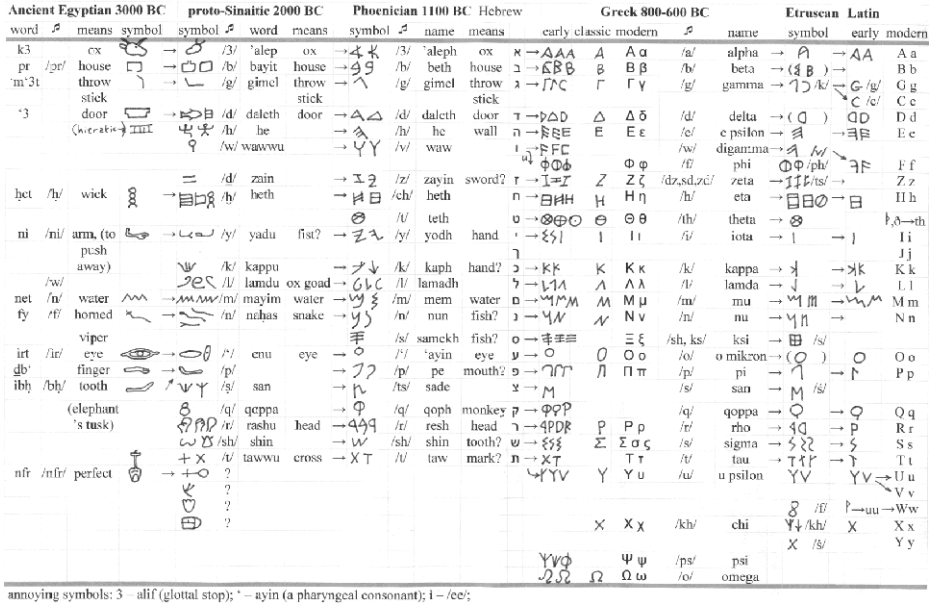
En algún momento, la industria deberá eliminar las advertencias de la Interpol y del DHS acerca de por qué no deberíamos haber robado lo que estamos a punto de ver. Cuando usted inserte un disco, debería haber dos botones que aparecieran en 5 segundos: “Reproducir“ y “Menú“. Tengo un estuche de DVDs que muestran cuatro minutos de las advertencias introductorias y de auto-promoción que incluso no puedo saltar, antes de reproducir el contenido real. Una vez puse el disco equivocado y tuve que repetir las molestias. Esto puede parecer pequeño, pero estas invasiones empeoran. Tengo que aceptar el acuerdo de licencia de mi sistema de navegación cada vez que enciendo mi coche![4] Cuando la gente se

4 Suponte que no active mi sistema de navegación un día porque no puedo ser molestado. Entonces, me accidento y muero porque el sistema no me estaba ayudando. Por lo tanto, podría argumentarse que la necesidad de aceptar el acuerdo

siente estafado y tratado como un tonto, se ha creado una cultura donde la gente decide no pagar por lo que se vende. El respeto es un camino de dos vías.

de licencia juega un factor en la causa de mi muerte. La buena noticia es que se crearía derecho para demandar! Ahora, si tan sólo pudiéramos encontrar una forma para que alguien muera porque tenían que sentarse a ver al principio de un vídeo todas estas cosas, entonces pudiéramos esperar por eso!

Soporte de Estándares de MS



Las formas del alfabeto latín son en su mayoría casualidad. Muchos detalles en este mundo no importan, es sólo que estamos de acuerdo con ellos..

Microsoft tuvo una actitud mixta hacia los estándares. Desempeñó un papel clave en la creación de muchos, tales como: HTTP, HTML y XML, USB, ACPI, Unicode y TrueType, y también apoyó a los estándares en los que no era parte en la creación, como TCP, SSL, SQL y SMTP.

Sin embargo, en muchos casos, Microsoft tiene una estrategia para crear un estándar competidor, y no estaba particularmente interesado en la documentación de sus tecnologías en estándares abiertos. Antes del despegue de HTML y HTTP, Microsoft tuvo un producto, cuyo nombre en código era Blackbird, que contenía muchos rasgos parecidos a los usados en la web, pero todos los elementos fueron especificados en un único mundo de Microsoft. Microsoft finalmente archivó Blackbird, pero por un tiempo, era mucho más importante dentro de la compañía que Internet Explorer, que empezó como un proyecto por amor, basado en la adquisición de código de terceros. En la mayoría de los casos, Microsoft apoyó los estándares sólo cuando sintió la presión comercial para hacerlo, uno de los lemas en los primeros días fue que Microsoft: “establece el estándar“.

Muchos de los ingenieros de la compañía escribieron código días enteros en productos líderes del mercado que trabajan con los competidores para estandarizar los detalles arbitrarios que parecían inútiles. El protocolo para compartir archivos e impresoras entre ordenadores fue indocumentado por muchos años, cuál sería el punto cuando nunca fue una prioridad hacerlo funcionar en Mac y Unix? Los protocolos propietarios son un tipo especial de estándar malo porque te atan a dos piezas de la tecnología de alguien. Outlook de Microsoft utiliza un protocolo propietario para buscar y retirar el correo electrónico de Exchange; este protocolo te ata al cliente y el servidor.

Los chicos del software libre no tienen , y no deberían tener ningún reparo acerca de la implementación de estándares cerrados, mal documentados o indocumentados, siempre y cuando el estándar sea muy popular. Un estándar abierto es siempre mejor, pero es el código fuente el verdadero valor de la propiedad intelectual, no si los detalles son sancionados por una organización en particular.

El software libre soporta más estándares que el software propietario. Si un estándar es popular, este conseguirá ser implementado. Linux por defecto extrae la música en el libre OGG, pero es fácil cambiarlo. El reproductor multimedia de Linux soporta los formatos de medios de Microsoft y Apple, además de estándares libres. El applet de mensajería instantánea de Linux soporta MSN, Yahoo, AIM, ICQ, GroupWise, Jabber, y otros. De hecho, el software propietario está deteniendo el desarrollo y la adopción de nuevos estándares. La BBC ha creado un códec de vídeo libre basado en wavelet conocido como Dirac, pero sólo Linux lo soporta fuera de la caja actualmente. Dado que es construido por los usuarios quienes añaden características según les sea necesario, el software libre te da más opciones que las que cualquier otro proveedor propietario puedan motivarse ellos mismos a prestar. No sólo hay mejor apoyo para los estándares, sino también que nunca te quedarás encerrado en una esquina porque el apoyo a un estándar desapareció por razones “estratégicas“. Con los formatos abiertos y software libre, estas construyendo una plataforma resistente a pruebas futuras.

La web es el estándar más importante para el intercambio de información, pero lo que es casi tan importante es el estándar para documentos de oficina.

OpenDocument Format (ODF)

De: Bill Gates
Enviado: Jueves, 5 de Agosto, 1999
Para: Bob Muglia

¿Por qué el Grupo de Office debería estar dando el formato de Office 2000 a los competidores? Para mí, esto suena como una locura.

Captura de pantalla promocional de Microsoft Office 2007 (¿Tienes azul?) Office incluye Word, Excel, PowerPoint, Outlook, Access, FrontPage, Visio, Project, OneNote, servicios web, y herramientas para construir extensiones de Office, un paquete increíble de interconexión de tecnologías propietarias.

La relación de Microsoft con los creadores de documentos es más antiguo y más arraigadas que su relación con los usuarios de Windows, he estado usando Windows desde 1990, pero he estado usando Word y Excel desde 1986. Para cientos de millones de trabajadores de la información, estudiantes, escritores, y los empleados públicos en todo el mundo, las principales herramientas para la producción de su propiedad intelectual está dentro de Microsoft Office, y hay miles de millones de documentos allá afuera. He oído decir que cada compra corporativa en el Reino Unido supone la creación

de una hoja de cálculo Excel. Un amigo en el ejército me dijo que la creación de documentos de PowerPoint es una habilidad necesaria para las altas esferas.

Microsoft Office es una gran parte de las ganancias de Microsoft, y la mejor razón para instalar Windows. Hoy en día, alrededor de 50 millones de personas utilizan el libre OpenOffice, mientras que el resto del mundo ha pagado $ 200 o $ 0 por una copia de Office. (El nombre de “OpenOffice“ es una marca registrada por otra persona, por lo que su “nombre oficial“ es en realidad “OpenOffice.org“.) Cuando te das cuenta del gran esfuerzo intelectual que es dedicado dentro de las herramientas de productividad, te das cuenta de lo importante que es que los formatos de archivo sean documentados. A una empresa o un gobierno le gustaría saber que pueden mantener estos archivos durante décadas, como lo pueden hacer con el papel. Esto es, por supuesto, un reto mucho más difícil porque en un pedazo de papel, todas las palabras digitalizadas, las reglas de diseño de texto y otra información de formato se pierden - imagina si alguien sustituyera todos sus documentos de Word por capturas de pantalla de estos documentos, la información sería legible, pero no editable.

El equipo de Microsoft Word no trató de construir un formato de archivo con el que estarían contentos por 20 años, porque sabían que la ingeniería de Word en 1993 para que leyera los archivos de Word 2013 era una tarea imposible. Durante los primeros 10 años, Word *n* ni siquiera podía leer los archivos de Word *n + 1*. Esto es porque durante muchos años Office utilizó formatos binarios, y se habría colapsado si lo hubiese intentado. Dentro de un archivo binario de Word, puedes encontrar los siguientes datos:

```
05     01
bold   on
```

'0501' significa encender negrita

```
05     00
bold   off
```

'0500' significa apagar negrita

```
06      ??
?!      ?!
```

¿Qué pasa si un Word viejo no entiende '06 '?¿Qué es lo siguiente?

Si usted no sabe lo que quiere decir `'06'`, usted no sabe lo próximo que viene, y así usted no puede continuar y debe abortar. Incluso si

quisiera saltar encima de él, no puedes porque no sabes hasta qué punto avanzar. Microsoft no creo los formatos binarios para bloquear otros proveedores. Los formatos de todos los procesadores de texto fueron binario durante muchos años debido a su eficiencia y porque una mejor solución no se había inventado.

La respuesta a este enigma, que ha plagado a la computación desde el principio, es que los documentos se auto-describan, y de todo esto es de lo que eXtensible Markup Language (XML) se trata. XML define cómo crear un documento en el que cualquier software puede leer y escribir sin que se colapse, aunque no lo entienda completamente. Puedo garantizar que el siguiente XML (fragmento, adornado para mayor claridad) será legible por un procesador de textos en 20 años:[5]

```
<?xml version="1.0" encoding="UTF-8"?>
<office:document-content office:version="1.0">
  <office:body>
    <office:text>
      <text:p text:style-name="Standard">Hello, 2028!</text:p>
    </office:text>
  </office:body>
</office:document-content>
```

"Hola, 2028!" en Open Document Format (ODF)

XML se basa en Unicode, la estandarización de los caracteres de todo el mundo, y añade a este <brackets> que le permiten encontrar el final de cada elemento, incluso los que no lo entiendo. </brackets> Tan pequeño como suena, la clave para ser capaz de leer archivos antiguos años más tardes, o los archivos nuevos con el código viejo, es simplemente hacer los datos auto-descriptivos, lo que le permite descubrir un pedazo del nombre y la longitud de los datos . Los brackets, y el formato XML exacto, no es importante, lo importante es el hecho de que todos los tipos de información que le gustaría representar son factibles de una manera que es posible para los seres humanos de leer y para los ordenadores de manipular. Cuando cada ordenador utiliza un formato estandarizado, auto-descriptivo, habremos dado un primer gran paso en ser capaces de intercambiar documentos sin causar un colapso.[6]

5 Incluso si el esquema no cambia de una manera incompatible, deberá ser posible escribir un pequeño programa para actualizarse al nuevo esquema - intente hacerlo con un formato binario. XML también le permite mantener, pero ignorar la información que usted no entiende, mientras que el código que admite un formato binario por lo general tira a la basura las cosas que no entiende.

6 Podría haber otros formatos como XML que particularmente no serían eficaces para analizar por los ordenadores. La eficiencia nunca fue una parte del diseño, ya que el código fue creado después de la especificación. Las personas de XML

XML consiste en tener un estándar, un formato de archivo auto-descriptivo y es una de los estándares más importantes en la historia de la computación, y *sólo uno* de sus usos será el esquema estándar para representar a los miles de millones de documentos de oficina.

Los formatos binarios de Office no se documentaron durante muchos años, y el contrato de licencia para la documentación de hoy en día dice que sólo se puede utilizar la información para productos que "complementan Microsoft Office." Es *suplantar* lo mismo que *complementar*?!

Microsoft no está particularmente interesado en la creación de un estándar abierto, porque nunca representaría perfectamente sus características, y debido a que un estándar abierto hace que sea fácil cambiar de herramientas. Ahora todo el mundo compra Office, porque eso es lo que usted necesita para leer los documentos que recibe hoy en día. La adopción de un formato abierto para las herramientas de productividad es una amenaza mortal para el nivel de ganancias de Microsoft Office.

Lamentablemente, hay una batalla en curso en el espacio de documentos de oficina XML. Microsoft ha ignorado durante muchos años y luego resistido el estándar ISO llamado OpenDocument Format (ODF), y ahora ellos han creado su propio estándar competitivo llamado Office OpenXML (OOXML). Sin embargo, el objetivo de un estándar es no tener dos de ellos.

XML proporciona la estructura para sus archivos y garantiza que las aplicaciones deben ser capaces de analizarlo todo, incluso partes que no comprenden, sin que se colapsen. Dada esa línea base, debería ser posible crear un formato que pueda representar las características de las herramientas de productividad de oficina. La especificación de OOXML de Microsoft, la cual proporciona 100% de compatibilidad con Microsoft Office, es de 6.000 páginas, mientras que la especificación ODF es sólo de 1.000 páginas, ya que no vuelve a utilizar muchos estándares existentes, como SVG, SMIL, MathML y Xforms.[7]

---

dicen que pueden aparecerse con formatos binarios más eficientes y sugiero que le tomemos la palabra. Tal vez el software propietario está llevando a cabo esto porque un XML binario requeriría cambiar mucho código base XML.

7 Stéphane Rodriguez documentó un número de defectos en el estándar OpenXML: http://ooxmlisdefectivebydesign.blogspot.com/.

OpenXML también está lleno de inflación de legado. En la parte superior de 600 páginas de la especificación VML está el texto siguiente:

> Nota: El formato VML es un formato heredado introducido originalmente con Office 2000 y está incluido y completamente definido en el presente estándar por razones de compatibilidad hacia atrás. El formato DrawingML es un formato más nuevo y más rico creado con el objetivo de la eventual sustitución de los usos de VML en los formatos Office Open XML. VML debe considerarse como un formato obsoleto incluido en Office Open XML por razones únicas de legado y las nuevas aplicaciones que necesitan un formato de archivo para los dibujos se recomienda encarecidamente usar preferentemente DrawingML.

Microsoft se ha movido desde entonces a despreciar la especificación VML en favor del formato DrawingML que es igualmente propietario, pero Word 2007 genera VML, por lo que la desaprobación de la especificación no hace que el trabajo desaparezca para las aplicaciones que quieren interoperabilidad con Microsoft. Google escribió en su análisis de OpenXML:

> Aunque formalmente OOXML podría cumplir con Ecma, era evidente que no fue diseñado con un espíritu “abierto“. Comparando la situación actual con la futura, la interoperabilidad es probable que sea más difícil, en lugar de más fácil. La implementación de un importador ODF totalmente compatible (los actuales esfuerzos en relación con .doc y .xls) no es una tarea fácil, pero queda eclipsada por la implementación de un importador de OOXML totalmente compatible, que nosotros estimamos tomará entre 50 a 500 personas años, o incluso más. Por lo tanto, aunque teóricamente es posible generar un documento de OOXML, este documento es probable que utilice sólo un subconjunto muy pequeño del estándar.
>
> En resumen, OOXML se puede comparar con que Microsoft dé acceso a un laberinto pero del que sólo ella posee un mapa y, además, algunos túneles dentro de este laberinto no son accesibles sin una clave que sólo Microsoft tiene, y que los terceros tendrían que replicar primero. (Y, al hacerlo, estos terceros no sabrían si violarían cualquiera de los derechos que los expone a los litigios).

Todas las cosas son iguales, aprovechando los estándares existentes es mejor que volverlos a inventar, y en la elección entre dos estándares, uno que es más pequeño, ya que se reutilizan los otros estándares, va a ser una elección mucho mejor para la industria. Un estándar que es difícil de darle soporte será adoptado lentamente, o tendrá implementaciones con errores. Hoy en día, OpenXML es mayoritariamente apoyada sólo por Microsoft, a diferencia de ODF,

que cuenta con un amplio apoyo en la industria de compañías como Red Hat, Adobe, Computer Associates, Corel, Nokia, Intel, Oracle, Novell, Google, IBM y Sun.

OpenXML es un formato propietario envuelto en XML. Este no es un estándar adecuado para el uso de muchos tipos diferentes de herramientas durante muchos años. Si usted comienza el Día 1 con un montón de equipaje, estás condenado. Microsoft ha presionado agresivamente para apoyar OpenXML, reconociendo que la adopción de ODF podría hacer más fácil migrar de Office.

Es importante reconocer que Microsoft Office es la suite de herramientas de productividad más completa y popular en el planeta, y por lo tanto cualquier formato abierto debe soportar importantes funciones de Office. Mi impresión de la lectura a través de la especificación es que los chicos OpenDocument han hecho lo imposible para garantizar una buena compatibilidad con Microsoft Office, y deben ser felicitados por su apertura mental. Por ejemplo, me sorprendió, y un poco consternado, para encontrar las referencias dentro de la especificación de DDE, una oscura y ahora casi muerta tecnología exclusiva Microsoft. Sin embargo, esta tecnología se convirtió en una parte del formato "monikor" Microsoft OLE, que especifica cómo los documentos deben integrar porciones de hojas de cálculo, y se convirtió en una parte importante de los documentos de Microsoft, por lo tanto ODF los soporta.

Un formato de archivo robusto, estándar, auto-descriptivo para las herramientas de productividad permitirá a las personas archivar sus documentos, confiados de que el formato se podrá leer en muchos años en el futuro.[8] Además, como todo lo que construimos en los ordenadores, los estándares pueden convertirse en platafor-

8 La cosa difícil sobre la construcción de un estándar es que las nuevas exigencias pueden causar ondulaciones a través del diseño de un sistema. Imaginemos dos personas sentadas en distintos países colaborando en un solo archivo. El desafío es que el formato de archivo ODF no fue hecho para ser incremental, sino para representar a un documento completo. ¿Envías una nueva copia del documento cada vez que el usuario realiza un cambio? Esto es muy ineficiente y sin embargo no le dice al usuario lo que ha cambiado. Otra solución es utilizar la pila de deshacer, pero parece que OpenDocument no almacena una pila de deshacer con el documento. Ellos solo podrían enviar diferenciaciones de XML, pero el XML no es normalmente la representación en memoria de un documento, en cuyo caso el XML no es fácilmente utilizable! Una solución es tener un modelo de objetos (con funciones como CreateTable) en la parte superior del formato del archivo, y pueden ser enviados entre ordenadores, pero el comité de OpenDocument no ha atacado este todavía.

Espero con interés ver cómo ellos resuelven este problema, o si deciden eso mientras es una característica factible, es muy difícil y fuera del ámbito de aplicación del estándar OpenDocument. El software es infinitamente maleable, pero eso no quiere decir que le gustarán los requisitos que le han impuesto!

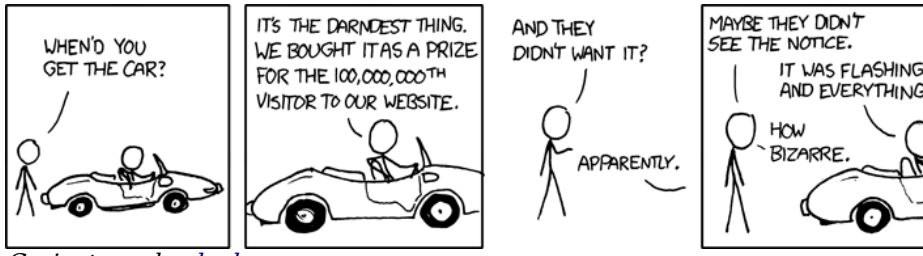
mas para otros estándares. Cuando ODF incorpora escenarios para cifrar y firmar digitalmente documentos, para certificar y transmitir documentos legales, y soportar el flujo de trabajo entre empresas, e-formularios y e-gobierno, podría convertirse en una lengua franca, de una manera que el DOC de Microsoft Word y el PDF *combinados* nunca han logrado.

“Los estándares competidores“ es un nombre inapropiado en mi opinión, lo que tal vez sería mejor si todos fueran a adoptar ODF. Sun está construyendo extensiones de Office para soportar a este formato, aunque si la gente comienza a utilizar el libre OpenOffice, que utiliza este formato como su formato nativo, no hay necesidad real de Office. Este libro fue escrito usando OpenOffice, y mientras la aplicación está lejos de ser perfecta, está mucho más allá de lo suficientemente bueno para la mayoría de los usuarios.

El estado de Massachusetts estaba con visión futurista hacia la *casi* adopción de ODF, pero después de toneladas de presión por parte de Microsoft, este cambió de rumbo y ahora ni respalda OpenXML, ni ODF. Ellos también cedieron a la idea de crear un estándar, o ellos realmente no entendieron el asunto. Microsoft ha intentado confundir a muchos sobre la importancia de los estándares como si dijera: “Hay muchos estándares por ahí, y el código debe ser capaz de trabajar con todos ellos.“ Es cierto que hay muchos estándares, pero cada uno debe servir a un propósito diferente!

La Web

> El primer mensaje que se envía a través de ARPANET se produce el 29 de Octubre de 1969, a las 10:30 PM. El mensaje en si era simplemente la palabra “login”. La “l” y la “o” se transmitieron sin problemas pero entonces el sistema colapso. Por lo tanto, el primer mensaje en la ARPANET fue “lo”. Ellos fueron capaces de hacer el login completo cerca de una hora más tarde.
>
> —Artículo de Wikipedia sobre el precursor de Internet ARPANET



Caricatura de xkcd.com

Se podría escribir un libro completo sobre la web, pero yo quería incluir algunas ideas aquí.

Desde una perspectiva técnica, HTML ha sido deficiente siempre como un estándar de formato de texto. Es importante por dos razones:

Es ampliamente usado: es famoso porque es famoso.

Tiene un modelo de fácil implementación.

HTML no se diseñó de forma rigurosa por los expertos de procesamiento de texto y que nunca tuvo ningún tipo de respeto imponente que el sistema de composición TeX de Donald Knuth ha tenido. HTML era maduro antes de que incluso añadiera soporte para el concepto de una página, motivo por el cual la impresión no funciona bien todavía. El procesamiento de texto es un problema difícil, pero no incorporar las características básicas como los estilos por siete años demuestra que los chicos estaban fuera de sus cabezas y que no deberían haber reinventado la rueda.

Además, la incorporación del lenguaje JavaScript fuera de la web ha sido mínima. No hay nada específico de la web acerca del lenguaje, por lo que ni siquiera necesitan ser creados. Esto solo hace la situación torre de Babel del lenguaje de programación peor.

A pesar de sus limitaciones, hoy HTML es el mejor sistema widget multiplataforma y está apoyado por una amplia gama de herramientas, por lo que las empresas deberían estar usándolo para el mayor

número de aplicaciones corporativas como sea posible. Un avión Boeing podría incluso tener que toda la interfaz de usuario de la cabina sea un sitio web. Usted puede construir algo simple, fiable y lo suficientemente bonito si ha seleccionado un subconjunto correcto de HTML. En este momento un avión tiene un reguero de botones y perillas, ya que cada subsistema del avión tiene su propio conjunto, los chicos que construyen los flaps no quieren compartir ningún botón con los que controlan las luces del techo.

La web hoy en día todavía está lejos de ser un instrumento adecuado para la creación de aplicaciones ricas, y casi todo lo relacionado con la web es más difícil que la construcción de una funcionalidad equivalente en una aplicación de cliente enriquecido. Google ha anunciado recientemente el sistema operativo Chrome con una visión donde todas las aplicaciones se ejecutan en la web, pero incluso ellos han creado una serie de aplicaciones que, presumiblemente, no pudieron ser construidas usando HTML, tales como Google Earth, Picasa y Google Desktop. Google Docs es un esfuerzo impresionante de ingeniería basada en la web, pero es lento, torpe, de características limitadas, no funciona desconectado, tiene su propio mecanismo de autenticación, se eliminan las capacidades limitadas de impresión de la red y no plantea ninguna amenaza al negocio de Microsoft Office en un corto plazo. La capacidad más interesante de Google Docs es su soporte para colaboración en tiempo real, pero no es necesario volver a construir una aplicación completa en HTML y Javascript para agregar este elemento relativamente pequeño.[9]

Las características de HTML que faltan son las razones por las que Flash de Adobe se ha vuelto tan popular. Flash comenzó como un entorno de programación fallido basado en el cliente, pero tuvo un renacimiento como un plugin web como una forma de solucionar las limitaciones de HTML.

Adobe Flash

Flash es un entorno de ejecución GC primitivo cuya principal ventaja es que es multiplataforma como la web. Su lenguaje de programación se conoce como ActionScript, que se basa en el estándar Ecmascript, y que es similar pero incompatible con JavaScript y Jscript. (Es un desastre.)

Flash se interpreta, algo propenso a errores y no es un estándar, ni se encuentra como código libre. Debemos minimizar el uso de Flash, ya que es una gran caja negra para el servidor web, navegador web,

9 De hecho, esto podría ser una característica del sistema operativo que te permitiría compartir una aplicación con cualquier número de otras personas.

motores de búsqueda, y todos los instrumentos existentes de HTML para crear y gestionar sitios web. Flash tampoco hace cumplir los estándares de interfaz de usuario. Todos los sitios flash funcionan y se ven diferente y a veces ni siquiera puedo decir lo que es un botón que se pueda pulsar! Los creadores del sitio web, quienes ciegamente viven en su aplicación, no lo notan, pero los usuarios que visitan el sitio web por primera vez lo hacen.

Muchos sitios web construidos por personas no técnicas contratar programadores para escribir un sitio web Flash debido a la interfaz de usuario agradable que permite, pero luego no actualizan su sitio desde hace años porque requeriría la contratación de un programador de nuevo. Es posible construir páginas web bonitas e interactivas usando sólo HTML y Javascript, como el mapeo web moderno ha demostrado. Si desea un sitio web bonito, usa imágenes bonitas!

Limitando el Flash ha porciones específicas de determinadas páginas, como YouTube hace con su reproductor de vídeo, es bastante razonable teniendo en cuenta ciertas limitaciones de la web y el desastre de los estándares de vídeo, pero la construcción de sitios web en Flash es todo un error y una amenaza a la web.

Fusionando los clientes enriquecidos y la Web

Además de sus limitaciones, HTML se ha estancado desde el lanzamiento de 4.01 en 1999. Tenemos que seguir evolucionando el HTML, y poner las herramientas al día con el último estándar. El software libre disminuye la fricción de la distribución de código y lo convierte en más fácil de hacer. Me preocupa que Internet Explorer mantenga el avance en HTML, ya que es tan popular, sin embargo, todavía esta años atrás en algunos estándares, y Microsoft ha disuelto el equipo varias veces.

XHTML y HTML 5 son esfuerzos incipientes para mejorar la web, aunque siguen siendo fundamentalmente limitados debido a que como estándar definen los límites de lo que una aplicación web puede ser. Mi ordenador tiene miles de componentes de software que las aplicaciones web no pueden incorporar porque no son parte del estándar HTML.[10] La Internet es una herramienta de transformación que permite todo tipo de colaboración, pero utilizar el formato de texto HTML no es necesario.

10 Incluso si pudieran incorporarlos en ellos, no podrían instalarlos si no estuvieran en tu ordenador.

El santo grial de la informática es encontrar una manera de combinar lo mejor del cliente enriquecido y la web. Java intentó sin éxito construir un sólido conjunto de widgets multiplataforma en un entono de ejecución extensible. Tal vez el lenguaje de programación que sustituya a C / C + + en el escritorio de Linux podrá revivir esta posibilidad. Este es uno de los interesantes desafíos que resta en la computación.

La Web Etc.

La web necesita un contenido más personalizado. Yo vivo en Seattle, y yo soy un gran fan del equipo de fútbol Seahawks, pero este es el artículo de noticias que ESPN me está ofreciendo a mí en un sitio web hoy: “Bengals rechazan grandemente la oferta de los Redskins por WR Johnson.“ Si quieren que haga clic , van a tener que hacerlo mejor que eso.

Amazon.com tiene muestras de música que puedo escuchar, pero esto toma 15 segundos para cargarlos antes de que pueda oírlos. Los archivos de música en la red se encuentran a menudo grabados en MP3 por debajo de 192 kbps, lo que significa que suenan peor que un CD, y con ello vamos de regreso a la tecnología antes de 1982.

Las imágenes deben ser grandes, y ajustadas en pantallas pequeñas. Drugstore.com tiene imágenes que son a lo mejor de 300 x 300 píxeles:

Puedes leer el texto en esta caja, como se pudiera hacer en una tienda real?

Esto es una “foto“ existente del monte Saint Helens encontrada en un sitio web:

Esta foto no es tan buena como estar allí.

Una experiencia de lectura inmersiva está olvidando una cosa: monitores de 200 dpi. Vi tal monitor en el año 2002, y era tan bonito que no podía sacar mis ojos, pero han sido inexplicablemente retirados del mercado.[11] La lectura es algo agotador, porque se necesita más trabajo para los ojos para reconocer las letras cuando son dentadas o borrosas.

La web necesita seguir integrándose con otros medios de comunicación como la televisión. La mejor manera de implementar la televisión interactiva es superponer visualmente HTML en la parte superior de la señal de TV. Lo bueno de esto es que usted puede eliminar la distracción arrastrando el texto a la parte inferior de la pantalla si usted quiere. Si la industria de la televisión quisiera algunas características para hacer que parezca más bonito que una página web, eso sería muy fácil de hacer. Una empresa francesa llamada Free es pionera en esto, utilizando el software libre como parte de este esfuerzo. (Un proveedor de cable es un buen ejemplo de una empresa que no quiere mantener un grupo de software propietario.)

Todo aquel que produce programas de televisión también debería crear un esquema XML que contuviera información como cuales son los invitados, etc. Esta información adicional permite una experiencia más personalizada similar a lo que recibo con la web: Yo podría poner mi caja de cable a grabar cada vez que el comediante Dennis Miller se encontrara en cualquier canal. Si un programa se pasase de tiempo, que a menudo ocurre con los deportes, la caja de cable sería lo suficientemente inteligente para no detener la grabación, sino que sólo necesitaría un poquito de información adicional para hacer esto. La televisión interactiva está simplemente esperando a alguien para crear, y que resto del mundo se quede detrás, dos estándares sencillos, cada uno de los cuales sería inferior a 50 páginas si vuelven a utilizar HTML y XML. (También sería bueno poder ver a los Seahawks no importa en qué ciudad viviéramos. Comcast sólo me ofrece 4 partidos de fútbol por semana. La gente se queja: “Hay 500 canales con nada.“ En verdad, todavía no estamos a ese punto!!)

11 Me di cuenta de que su alta resolución rompió sitios web que funcionan en píxeles, y lo hechaba a perder tanto cuando los píxeles eran 4 veces más pequeños, haciendo cosas como mostrar sólo 3 palabras por línea.
Un avance mucho menor en la tecnología de los monitores es utilizar los LEDs como luz de fondo para los monitores LCD en lugar de bombillas fluorescentes. Los LEDs son más durables y más eficientes que las bombillas fluorescentes de hoy. Los LEDs sustituirán eventualmente muchos usos de la luz incandescente y fluorescente, ya que son 45 veces más eficientes que la incandescente, y 7 veces más que el fluorescente.

Hardware

Mucha gente se preocupan acerca de quedarse sin ancho de banda de internet, pero no son más que nabobs hablantines del negativismo. Nippon Telephone and Telegraph de Japón lo demostró enviando 14 trillones de bits por segundo por una sola hebra de fibra – o 2660 CDs de música en un segundo. No estamos hiendo en contra de los límites de las leyes de la física todavía!

Medir los datos suena lógico, aunque pudiera parecer que la distancia sería una mejor medida. Sin embargo, yo conté el número de saltos de enrutador para obtener datos de mi hogar en Seattle para diversos destinos:

De Seattle a:	Saltos de Router
google.com (Mountain View, CA)	23
msn.co.jp (Japón)	23
www.tmobile.de (Alemania)	30
www.latviatourism.lv (Letonia)	17
www.google.co.uk (Reino Unido)	15
www.gws.com.tw (Taiwan)	23

En otras palabras, la distancia hasta el destino no se correlacionó con la cantidad de trabajo necesario para enrutar el paquete hacia el destino. Por lo tanto, el enrutamiento basado en la distancia podría no tener sentido. Sin embargo, si vamos a medir sobre la base de ancho de banda utilizado, ¿cómo suena diez centavos de dólar por gigabyte? Por 20 dólares al mes, mi proveedor de servicios de Internet me da 200 gigabytes de transferencia de datos, una dirección IP, una instancia de virtualización, diez gigabytes de almacenamiento de disco redundantes, y 360 MB de RAM. Tengo diez centavos de dólar por gigabyte, y todas las cosas de forma gratuita. 1 gigabyte por diez centavos te permite enviar dos copias del texto de la Enciclopedia Británica, que sería todo el tráfico web que muchos necesitan para todo un mes.

Tenga en cuenta que el coste de transmisión de paquetes sigue la ley de Moore porque un router es un ordenador especializado. Por lo tanto, el costo de envío de un paquete está disminuyendo de manera exponencial. En 18 meses, deberíamos preguntar por cinco centavos de dólar por gigabyte. Lo bueno de mudarse a este modelo es que

esto crearía un incentivo para las personas para ofrecer vías para usar más ancho de banda. Las compañías de cable ofrecerían más contenido de alta definición si se les estuvieran pagando más por el tráfico.

Da Future

Fase II de la Carrera de Bill Gates

> Las regalías que nos pagan a nosotros, menos nuestros gastos del manual, la cinta y la sobrecarga hacen a Microsoft una operación en equilibrio en la actualidad.
>
> —Bill Gates, Carta Abierta a los Aficionados (Open Letter to Hobbyists), 1976

Bill a Steve:"Fue divertido mientras duró."

En Junio del 2006, nos enteramos de que Bill Gates dejará el cargo de Microsoft en junio de 2008. Esto le dio un montón de advertencias a los mercados, pero también significa que Steve Ballmer, permanecerá como CEO por una década o más, los rumores de su dimisión han sido muy exagerados.

Uno puede presumir que Bill Gates no cree que su legado es impulsado por el gasto de más tiempo en Microsoft. Teniendo en cuenta la doble amenaza del software libre y Google, la historia puede juzgar a Bill Gates, más como un Andrew Carnegie que de un Michelangelo.

Microsoft tuvo éxito porque fue la empresa que explotó la ley de Metcalfe para su mayor ventaja. Microsoft tuvo a todo el mundo usando MS-DOS, que se usó para aspirar a los clientes hacia Windows, Office, y todo lo demás. Muy pocas compañías tienen esta estrategia, o los recursos.

Bill Gates ha dado un montón de liderazgo a la industria de la computación en las últimas décadas. En áreas desde interfaces gráficas de usuario y herramientas de productividad integradas, hasta el software como un servicio, al estilo de vida web, por lo que su dimisión podría ser una pérdida mayor para la industria que incluso Microsoft desprecia. ¿Quién más puede ofrecer esa visión, algo que agrupe a toda la industria?

Dicho esto, Bill no ha proporcionado tanto liderazgo en los últimos años. La última vez que estuvo en la portada de Time Magazine, él estaba hablando de la XBox 360.

¿Por qué Bill no está sonriendo? Tal vez porque la Xbox 360 es una PC sin un teclado, un navegador web y un montón de otros programas.

Puede que haya varias razones por las que Bill no se ha mantenido como un símbolo de la industria en los últimos años. Parte de ello es que Microsoft no parece tener nada nuevo que decir. HTML ha cambiado muy poco en los últimos seis años, por lo que Microsoft no tiene nada de qué hablar con respecto a la web.

En segundo lugar, lo que ocurre con Microsoft no importa tanto a la industria de la computación. Mucha gente que construyen sitios web lo hacen en PHP, un lenguaje de programación libre. MySQL es la

segunda base de datos más popular en América del Norte, así que todos estos usuarios no se preocupan por lo que está incluido en el más reciente Microsoft SQL Server, o cómo se integra mejor que nunca con el más reciente Windows. Vista fue un lanzamiento importante, pero no tiene funciones imprescindibles, por lo que la emoción fue silenciada. (La innovación más importante de Windows 95 fue la computación de 32 bits.) El formato OpenDocument es una especificación casi tan importante como HTML, pero Microsoft no lo admite, por lo que Bill no puede hablar de ello.

Bill Gates, firmó el último pacto con el Diablo en la historia de los negocios: el software propietario hizo a Microsoft la compañía más valiosa que jamás se haya creado, pero estaba destinada al fracaso porque no adoptó un acuerdo de licencia expansiva que deja a sus usuarios contribuir con el sistema.

Si Windows NT hubiese adoptado GPL, no habría habido ninguna razón para inventar Linux. Unix es anterior a Microsoft DOS, y tal vez DOS y Windows no se hubieran inventado, si los diferentes sistemas Unix hubieran sido desde el principio GPL.

Una gran cantidad de software ha sido propietario desde que las computadoras se inventaron, pero es interesante preguntarse dónde estaríamos hoy si GPL hubiese sido el acuerdo de licencia estándar desde el principio. La industria sería muy diferente, y desde luego mucho más adelantada.

A pesar de que el software libre elimina la necesidad de la existencia de Microsoft, la historia puede recordar a Bill Gates por muchas otras cosas aparte de su papel en la empresa. La Fundación Gates tiene 80 mil millones para gastar lo que es suficiente para contratar a 20.000 trabajadores calificados por 40 años. Gastar 80 mil millones dólares es mucho más difícil de lo que parece, sobre todo si se invierte en proyectos que crean valor, y a su vez generen ingresos! Bill tiene los recursos para involucrarse en esfuerzos *muy grandes*, incluida la exploración espacial. (El comediante Dennis Miller dijo que Bill Gates es sólo un monóculo y un gato persa lejos de ser un villano de James Bond.)

El espacio, o cómo el hombre hizo su sueño realidad

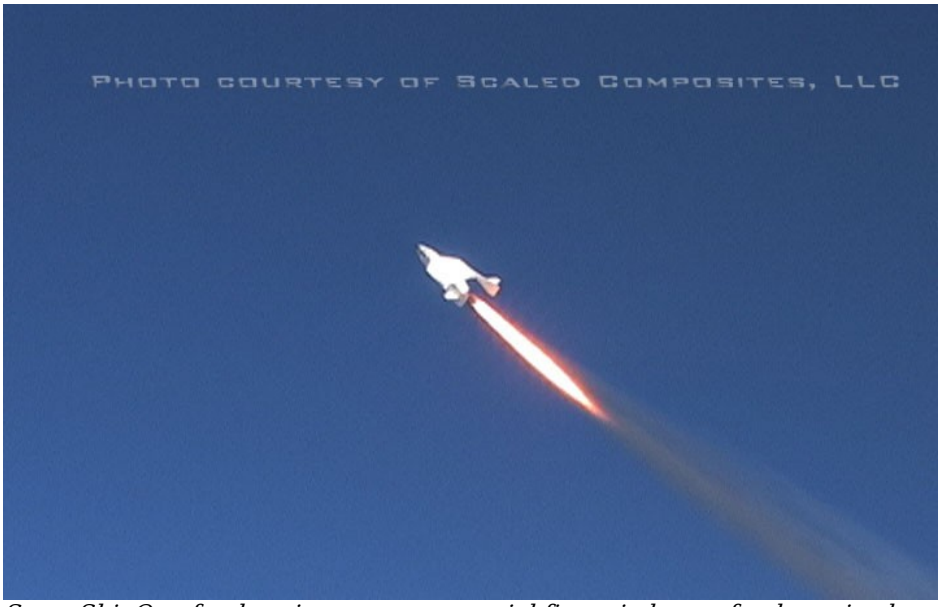
Es medianoche, 20 de julio de 1969, un claroscuro de contrastes chillones aparece en la pantalla del televisor. Una de las sombras se mueve, es unas de las piernas del astronauta Edwin Aldrin fotografiado por Neil Armstrong. Los hombres caminan sobre la luna. Miramos hechizados. La Tierra observa.

Setecientos millones de personas estaban atentos a sus radios y pantallas de televisores en aquella noche de julio de 1969. ¿Qué puedes hacer con la luna? Nadie sabía. Aún así, una sensación en el estómago nos decía que este era un gran momento en la historia de la vida. Estábamos dejando La Tierra, nuestros pies estaban moviendo el polvo de un mundo extraterrestre.

—Robert Jastrow, *Viaje a las Estrellas*

!La Administración esta haciendo bien las cosas , la Jefatura esta haciendo las cosas correctas!

—Peter Drucker



SpaceShipOne fue la primera nave espacial financiada con fondos privados en ir al espacio, y constituyó una serie de importantes "números uno", incluyendo que fue la primera astronave privada en superar el Mach 2[1] y el Mach 3[2], la primera aeronave espacial privada en exceder la altitud de 100 km y la primera nave reusable. El proyecto se estima que ha costado $25

1 Es un número adimensional típicamente usado para describir la velocidad de los aviones. Mach 1 equivale a la velocidad del sonido.
2 Mach 2 es dos veces la velocidad del sonido.

millones de dólares y fue construida por 25 personas. Ahora se encuentra en el Smithsonian ya que no sirve para ningún propósito comercial, y porque el desafío no ha sido llegar al espacio, siempre ha sido el costo.

En el siglo 21, más cooperación, mejores softwares y la nanotecnología brindarán profundos beneficios para nuestro mundo; y avergonzaremos a los Baby Boomers. En este libro me centro sólo en la tecnología de la información, pero las ciencias materiales serán unas de las tareas más grandes que ocuparán nuestras mentes en el siglo 21; y muchos futuristas dicen que la nanotecnología es el próximo(¿y el último?) gran reto después de la infotecnología.

Me gustaría terminar este libro con una idea más grande: cómo ponemos en marcha la revolución de la nanotecnología y la usamos para colonizar el espacio. El espacio, más que cualquier otro esfuerzo, tiene la habilidad para poner a trabajar nuestra imaginación y darnos esperanzas para el futuro. Cuando un hombre esta explorando nuevos horizontes, denota cierta arrogancia en sus pasos.

Colonizar el espacio cambiará la perspectiva del hombre. El acaparamiento es un instinto muy natural. Si das un hueso a un perro bien alimentado, él lo enterrará para guardarlo hasta días más difíciles. Todo animal acapara. Los humanos acumulan dinero, joyas, ropas, amigos, arte, créditos, libros, música, películas, sellos, botellas de cerveza, estadísticas de béisbol, etc. Nos ligamos mucho a estos “tesoros”. Así peleemos por $5,000 ó $5,000,000 las emociones tienen exactamente la misma intencidad.

Cuando nos sentimos atiborrados en este punto azul pálido, nos olvidamos que fuera de él hay cualquier recurso que podamos querer en grandes cantidades. Si distribuimos solo los recursos de nuestro sistema solar a los 6 billones de personas por igual, entonces cada uno de nosotros recibiría:

Recurso	**Cantidad**
Hidrógeno	34,000 billones de Toneladas
Hierro	834 billones de Toneladas
Silicatos (arena, vidrio)	834 billones de Toneladas
Oxígeno	34 billones de Toneladas
Carbono	34 billones de Toneladas
Producción de energía	64 trillones de Kilowatts por hora

Incluso si nos limitamos solo a los recursos de este planeta, tenemos mucho más de lo que podríamos necesitar. Este simple entendimiento es un requisito previo para una sociedad más optimista y generosa, la cual se ha caracterizado por tener eras de gran progreso. Desafortunadamente, los actuales planes de la NASA estan lejos de querer esto.

Si la NASA sigue adelante con su visión del 2004 de retirar el Transbordador Espacial, volver a los cohetes espaciales, e ir a la Luna otra vez, esta sería su propia visión de lo que estaríamos viendo en DrudgeReport.com en el 2020.

Foto eliminada

Nuestros astronautas aún estarán orinando en sus trajes espaciales en el 2020.

De acuerdo a la NASA, la foto de abajo es lo que veremos en el 2020, pero si le echamos una mirada de reojo, luce igual que 1969:

Foto eliminada

Todo esto fue hecho sin esas cosas que llamaríamos computadoras.

Solo la burocracia gubernamental puede hacer tan poco progreso en 50 años y aún así considerarlo negocio, como es de constumbre. Hay muchos casos documentados de grandes organizaciones gubernamentales que están plagados de fracasos de la imaginación, todavía nadie considera que los científicos burócratas de la NASA puedan también estar acosados por esta aflicción. Esto es especialmente irónico porque el actual administrador de la NASA, Michael Griffin, ha admitido que muchos de sus esfuerzos en el pasado fueron un fracaso:

El transbordador espacial, diseñado en los 70s, esta considerado un fracaso por ser poco fiable, caro y pequeño. Cuesta $20,000 la libra de carga útil ponerlo en órbita terrestre baja(OTB), a apenas unas pocas millas de altura.

La estación espacial es pequeña y está a solo 200 millas de distancia donde la gravedad es equivalente al 88% de la existente al nivel del mar. No es auto-sostenible y no nos acercará a poner a la gente en la Luna o Marte.(Moviendose a 17,000 millas por hora, cae lo suficientemente rápido para permanecer en la misma óbita) Los Estados Unidos solamente gastó $100 billones en este despilfarro.

La clave para el éxito definitivo de cualquier organización, desde la NASA hasta cualquier otra empresa privada, es que haya en la cima un líder con visión. Los errores que se cometieron en la NASA no fueron porque fue construida por el Gobierno, sino que los líderes

tomaron decisiones incorrectas. Microsoft, en contraste, triunfó porque Bill Gates tomó muchas decisiones muy inteligentes. El actual objetivo de la NASA es “plantar banderas y huellas”, cuando debería ser, cómo hacer más fácil el hacer estas cosas, que es un objetivo completamente diferente.[1]

No es que apoye el rediseño del transbordador espacial, pero tampoco creo que nadie en la NASA haya considerado seriamente construir una nave espacial reusable de nueva generación. La NASA está basando su decisión de volver a los cohetes espaciales principalmente en las fallas del primer transbordador espacial, una idea similar sería mirar el primer carro construido y concluir por esto que los carros nunca funcionarán.

Desafortunadamente, la NASA ahora está regresando a tecnología mucho más primitiva que la del transbordador. El “concenso” en la industria aeroespacial hoy, es que los cohetes son el futuro. Los cohetes pueden ser nuestro futuro, pero también están en el pasado. El estado del arte de las investigaciones espaciales es hacerlas el 15% más eficientes. Hoy día, las investigaciones espaciales aumentan más porque la física y la química básica no han cambiado desde que fueron lanzadas a mitad del siglo 20.

Los cohetes químicos son un error porque el combustible que los impulsa para subir es ineficiente. Tienen un bajo “impulso específico”, lo que significa que necesita mucho combustible para acelerar la carga útil, y consume mucho más para acelerar dicho combustible. Como puede ver en las impresionantes escenas de los lanzamientos de los transbordadores; la tecnología actual no es del todo eficiente; los cohetes suelen contener el 6% de carga útil y el 94% de gastos generales.(Los motores de reacción no trabajan sin oxígeno pero son 15 veces más eficientes que los cohetes)

Si quiere saber por qué no hemos vuelto a la luna por décadas, aquí hay una analogía:

¿Cuánto costaría la entrega de este carro?

1 Los europeos no han mostrado gran liderazgo tampoco. Una de las grandes inversiones de sus agencias espaciales, además de la EEI(ISS, por sus siglas en inglés), es construir un duplicado de la constelación de satélites GPS, el cual lo están haciendo principalmente debido al anti-americanismo! Muy malo, ellos no se dan cuenta de que sus emociones les está llevando a reimplementar tecnología de 35 años, en vez de gastar esos $5 billones en un verdadero y nuevo avance. Clonando el GPS en 2013: !Vaya logro, Europa!

Un californiano compra un carro hecho en Japón.
El carro es enviado en su propio vehículo transportador.
El carro es descargado en el puerto de Los Angeles.
El carguero luego se hunde.

Lo último en tecnología de propulsión es el accionamiento eléctricos de iónes el cual acelera los átomos 20 veces más rápido que los cohetes químicos, lo cual significa que se necesita mucho menos combustible. La ineficiencia de nuestros actuales cohetes químicos es lo que esta impidiendo al hombre de colonizar el espacio. Nuestros simples cohetes modernos puede que sean más baratos que nuestra vieja y complicada Lanzadera Espacial pero aún así va a costar miles de dólares por libra para llegar a órbita terrestre baja, un acrónimo de lujo para 200 millas de distancia. Trabajar hoy en los cohetes químicos es el equivalente tecnológico de pulir un mojón polvoriento; aún así, esto es lo que nuestra estimada NASA está haciendo.

El ascensor espacial.

> Cuando un científico distinguido y anciano indica que algo es posible, él está casi seguro de que es cierto. Cuando él indica de que algo es imposible es muy probable que este equivocado.
>
> —Arthur C. Clarke RIP, 1962

> La mejor forma de predecir el futuro es inventarlo. El futuro no está dispuesto, ordenado en un carril. Es algo que decidimos nosotros, y en la medida en que no rompamos alguna ley conocida del universo podemos probablemente hacer que trabajen de la forma en que queramos.
>
> —Alan Kay

Picture removed to save space.

Una descripción de la NASA del ascensor espacial. Un ascensor espacial hará cientos de veces más barato poner una libra en el espacio. Se trata de una diferencia de eficacia comparable a la existente entre el caballo y la locomotora.

Una de las mejores vías de volver al espacio de una forma barata está rondando los laboratorios de investigación de la NASA:

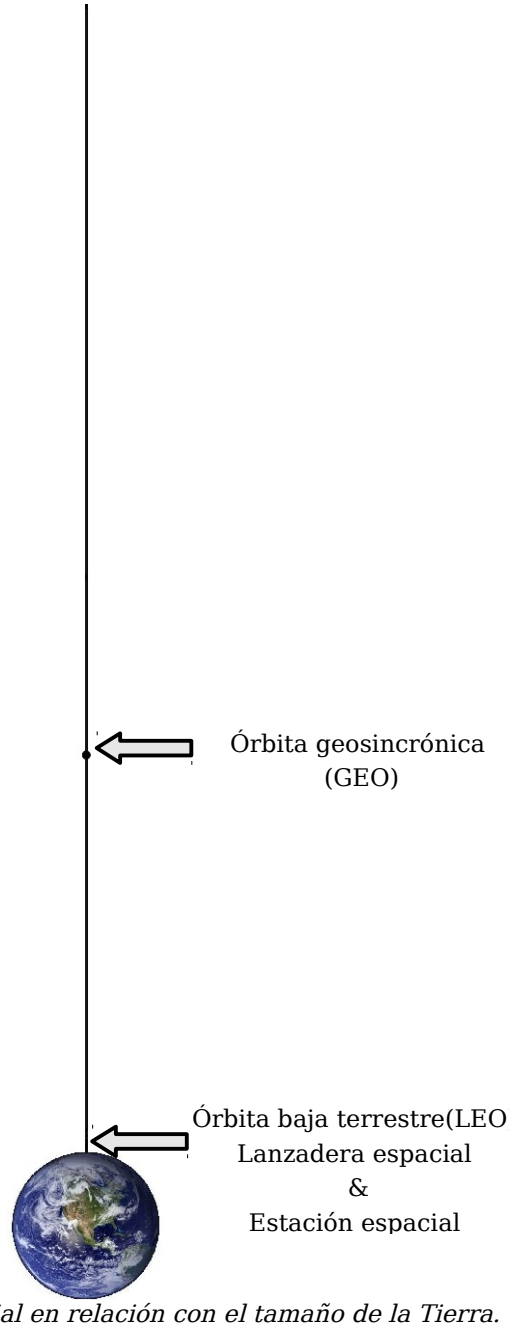


Foto escala del elevador espacial en relación con el tamaño de la Tierra. La Luna está a 30 diámetros de la Tierra de distancia, pero una vez y estás en órbita geoestacionaria se requiere relativamente poca energía para llegar a la Luna o cualquier otro lugar.

Un ascensor espacial es una cuerda de sujeción de 36,000 kilómetros que puede lanzar cosas al espacio de forma lenta, segura y barata.

Y estos escaladores ni siquiera tienen la necesidad de llevar su energía así como usted puede utilizar paneles solares para proporcionar la energía para los escaladores. Todo esto significa que usted necesita mucho menos combustible. Todo es completamente reusable así que cuando se ha construido tal sistema es fácil tener lanzamientos diarios.

Los primeros escaladores de ascensor viajarán al espacio a unos pocos miles de millas por hora, una velocidad muy segura. Construir un dispositivo que pueda sobrevivir a la aceleración y al empuje es en gran parte el costo de poner cosas en el espacio hoy día. Esta tecnología hará cientos y luego miles de veces más barato poner cosas y finalmente personas en el espacio.

Un elevador espacial puede sonar a ciencia-ficción, pero como otras tantas ideas de ciencia-ficción, es una fantasía que tiene sentido económico. Mientras que usted no tiene que confiar en mi opinión de si un elevador espacial es factible, la NASA nunca ha evaluado oficialmente este tema, ellos no lo han considerado seriamente.

Todo esto puede sonar a ciencia-ficción pero comparado con la tecnología de los 60s, cuando la humanidad por primera vez se embarcó en un viaje hacia la Luna, un elevador espacial es simple de construir para nuestro mundo moderno. De hecho, si transportas el celular a los científicos del Apollo, ellos lo tratarían como si fuera una supercomputadora y tendría grupos de ingenieros inclinados sobre él 24 horas al día. Sólo con la adición de la tecnología de computación de un móvil, podríamos haber adelantado un año la fecha del primer alunizaje.

Nanotubos de carbono

Picture removed to save space.

Los nanotubos son átomos en forma de hexágono. Gráfico creado por Michael Ströck.

Tenemos todas las capacidades tecnológicas necesarias para construir un elevador espacial con excepción de: los nanotubos de carbono. Para adaptar una frase de Thomas Edison, un ascensor espacial es 1% inspiración y 99% transpiración.

Los nanotubos son extremadamente fuertes y ligeros, con una fuerza teórica de tres millones de kilogramos por centímetros cuadrados; un haz del tamaño de unos pocos pelos puede levantar un carro. La fuerza teórica de los nanotubos es mucho más de lo que necesitaríamos para nuestro elevador espacial; los diseños preliminares actuales especifican un papel delgado, una cinta de 3 pies de ancho. Estas dimensiones aparentemente frágiles pueden ser lo suficientemente fuertes para resistir su propio peso, y las 10 toneladas de los escaladores utilizando el escalador.

Los nanotubos que necesitamos para nuestro elevador espacial son el momento perfecto para empezar la revolución de la nanotecnología porque, a diferencia de la investigación en la nanotecnología biológica que utiliza cientos de átomos con estructuras extremadamente complicadas, los nanotubos tienen un diseño trivial.

La mejor forma de atacar un gran problema como la nanotecnología es, primero, atacar una pequeña parte de él, como los nanotubos. Un “Proyecto Manhattan” sobre nanotecnología general no tiene sentido porque el problema está muy desenfocado, pero tal esfuerzo podría tener sentido para los nanotubos. O simplemente puede requerir la experiencia industrial de una compañía como Intel. Intel ya está experimentando con los nanotubos dentro de los chips de computadora porque el metal pierde la habilidad de conducir la electricidad en diámetros muy pequeños. Pero nadie les ha preguntado si pueden construir cuerdas de millas de largo.

El gobierno de los Estados Unidos recientemente ha incrementado las inversiones en nanotecnología, pero no hemos visto muchos resultados. Del experto en elevadores espaciales Brad Edwards:

> Existe lo que se llama la Iniciativa Nacional de Nanotecnología. Cuando investigué sobre él, el presupuesto fue de miles de millones de dólares pero cuando lo miras más de cerca, está dividido en una docena de agencias diferentes, y dentro de cada agencia esta dividida en otra docena de áreas diferentes, gran parte de ellos terminan como subsidios de $100,000. Investigamos sobre lo que respecta a los compuestos de nanotubos de carbono y aparecío que cerca de treinta millones de dólares se van en materiales de alta resistencia y mucho de eso ha sido gastado internamente en muchas de las agencias; al final hay solo un par de millones de esos miles de millones de dólares de presupuesto que están destinados a algo que pueda ser útil para nosotros.
>
> El dinero no tiene un centro y se extiende para abarcarlo todo. Se obtiene un poco de esfuerzo en miles de lugares diferentes. Gran parte del presupuesto es gastado en una entidad que trata de alcanzar a quienquiera que esté a la delantera. En vez de financiar al líder, ellos financian internamente a alguien más para ponerse al día.

Una vez más, hay aquí un problema similar al que encontramos en el software de hoy: la gente jugando a alcanzarse en vez de trabajar en conjunto. No se lo que hacen los científicos de nanotecnología cada día pero parece que harían bien en seguir los pasos de los pioneros del software libre y empezar a cooperar.

La producción ampliada de nanotubos podría el comienzo de la revolución de la nanotecnología. Y el elevador espacial, la aplicación estrella de los nanotubos, permitirá la colonización del espacio.

Por qué?

William Bradford, hablando en 1630 de la fundación de la colonia de Plymouth Bay, dijo que toda honorable y gran acción está acompañada de grandes dificultades, y ambas deben ser emprendidas y superadas con corage.

Hasta el momento no hay ninguna lucha, prejuicio o conflicto nacional en el espacio exterior. Sus peligros son hostiles a todos nosotros. Su conquista merece lo mejor de toda humanidad, y la oportunidad de una cooperación pacífica puede no volver nunca más. Pero ¿por qué algunos dicen, la Luna? ¿Por qué escoger esto como nuestra meta? Y se puede preguntar, ¿por qué escalar la montaña más alta? ¿Por qué hace 35 años atrás, sobrevolar el Atlántico? ¿Por qué juega Rice con Texas?

Escogemos ir a la Luna. Escogimos ir a la Luna en esta década y hacer las demás cosas, no porque son fáciles sino porque son difíciles, porque esa meta servirá para organizar y calcular lo mejor de nuestras energías y habilidades, porque ese desafío es uno que estamos dispuestos a aceptar, uno que no estamos discpuestos a posponer, y el cual tenemos la intensión de ganar, y los demás, también.

Por estas razones considero que la decisión del año pasado de pasar de pocos a grandes nuestros empeños en el espacio, entre las mejores decisiones que serán hechas bajo mi mandato en la oficina de la Presidencia.

En las últimas 24 horas hemos visto las facilidades ahora creadas por la más grande y compleja exploración en la historia del hombre. Hemos sentido el temblor de tierra y el aire quebrado por la prueba de un cohete acelerador Saturn C-1 , muchas veces tan poderoso como el Atlas, el cual lanzó John Glenn, generando un poder equivalente a 10,000 automóviles con sus aceleradores en el piso. Hemos visto el sitio donde los cinco motores F-1, cada uno tan poderoso como los ocho motores del Saturn combinados, serán agrupadas todas para hacer el avanzado misíl Saturn, ensamblado en una nueva edificación para ser construido en Cabo Canaveral tan alto como un edificio de 48 plantas, tan amplio como una manzana, y tan largo como dos veces la longitud de esta área.

El crecimiento de nuestra ciencia y educación será enriquecida con el conocimiento de nuestro universo y el medio ambiente, con nuevas técnicas de aprendizaje, mapeo y de observación, con nuevas herramientas y computadoras para la industria, la medicina, el hogar así como para las escuelas.

No digo que debemos ir o que irémos desprotegidos contra el hostil mal uso del espacio algo más desprotegidos de lo que iríamos contra el uso hostil de la tierra o el mar, pero si digo que el espacio puede ser explorado y dominado sin alimentar los fuegos de la guerra, sin repetir los errores que el hombre ha cometido para extender su mandato alrededor del mundo.

> Le hemos dado a este programa una alta prioridad nacional – si bien me doy cuenta que esto es en cierta medida un acto de fé y vision, porque ahora no sabemos cuales beneficios nos aguardan. Pero si fuera a decir, mis conciudadanos, que enviaremos a la Luna, a una distancia de 240,000 millas de la estación de control en Houston, un cohete gigante de más de 300 pies de alto, del largo de este campo de fútbol, hecha de nuevas aleaciones de metal, algunas de las cuales no se han inventado todavía, capaz de soportar el calor y acentuado muchas veces más del que se ha experimentado, ajustado con una precisión mejor que la del reloj más refinado, cargar con todo el equipamiento necesario para la propulsión, la orientación, el control, las comunicaciones , comida y otros medios de subsistencia, en una misión nunca probada, a un cuerpo celeste, y luego retornar a salvo a Tierra, volver a entrar a la atmósfera a una velocidad de más de 25,000 millas por horas, causando una temperatura de casi la mitad de la temperatura del Sol-casi tan caliente como está aquí hoy- y hacer esto y hacerlo bien, y hacerlo antes de que esta década pase- entonces seríamos audaces.
>
> —John F. Kennedy, 12 de septiembre de 1962

Picture removed to save space.

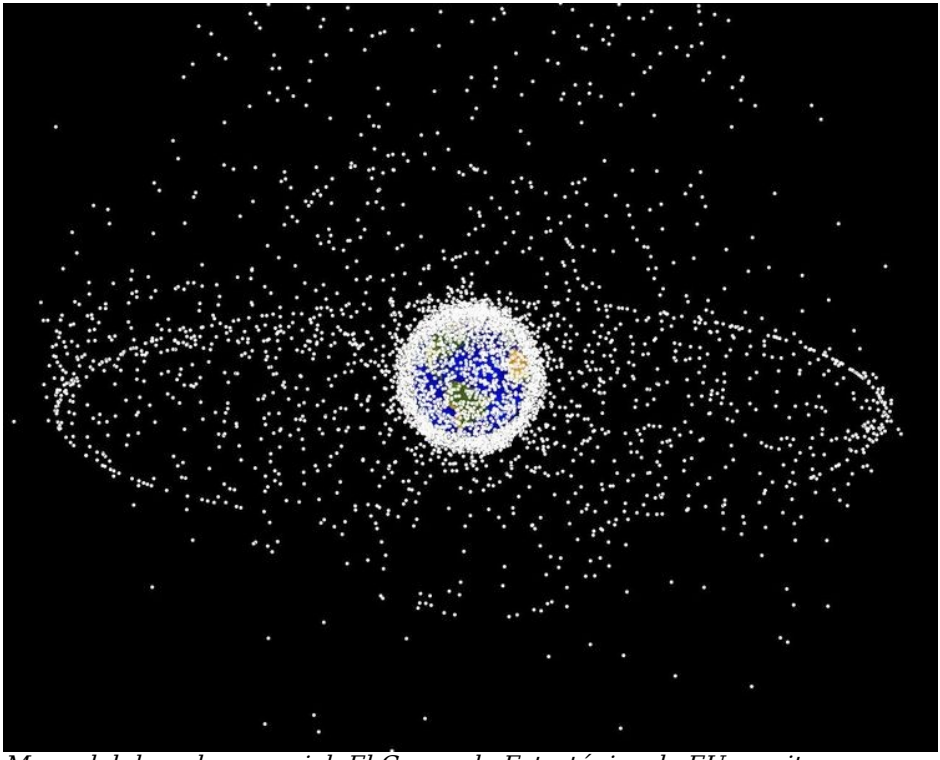
El aterrizador lunar en la cima de un cohete. Los cohetes son caros e imponen significativas limitaciones de diseño de la carga flotante en el espacio.

La NASA tiene 18,000 empleados y un presupuesto de 17 billones de dólares. Incluso con una fracción de esos recursos, su habilidad para supervisar el diseño, controlar la misión, y trabajar con muchos socios está más que a la altura de esta tarea.

Si la NASA no construye el elevador especial, alguien más puede que lo haga, y cambiaría casi todo acerca de cómo la NASA hace las cosas hoy. La nueva y diminuta nave especial Orión de la NASA (15 pies de ancho), la cual fue contruida para regresarnos a la Luna, fue diseñada para ajustarse encima de un cohete y retornar a los astronautas a La Tierra de un golpe a 25,000 millas por horas, igual que en los días del Apollo. Sin las restricciones que un cohete impone, la nave especial de la NASA para regresarnos a la Luna podría tener un diseño muy diferente. La NASA necesitaría tirar mucho del R&D que están haciendo ahora si el elevador espacial fuera construido.

Otra de las razones por la cual el elevador especial tiene sentido es que este haría a varios científicos en la NASA trabajar juntos en una gran meta compartida. La NASA recientemente ha mandado roboes a Marte a cavar hoyos de dos pulgadas en la suciedad. Ese tipo de experiencia es similar a la habilidad necesaria para construir los escaladores robóticos que escalarían el elevador, aprovechando a los científicos en un propósito más grande.

Los desechos del espacio es un peligro al acecho, y una amenaza para la cinta:



Mapa del desecho especial. El Comando Estratégico de EU monitorea 10,000 cuerpos grandes para prevenir que sean malinterpretados como misiles hostiles.

El elevador especial provee un motivo y un medio para lanzar cosas al espacio para eliminar los desechos. (El primer elevador necesitará ser diseñado con la habilidad de trasladarse para evitar los desechos)

Una vez usted haya construido su primer elevador especial, el costo de construcción del segundo cae dramáticamente. Un elevador espacial finalmente hará a $10 por libra poner algo en el espacio. Esto abrirá muchas puertas para los científicos e ingenieros alrededor del globo: más grandes y mejores observatorios, un puerto espacial en la GEO, y así sucesivamente.

Sorprendentemente uno de los más grandes incentives para la exploración especial es probable que sea el turismo. Desde Hawai a África a Las Vegas, el principal ingreso en muchos lugares exóticos es el turismo. Irémos a las estrellas porque el hombre está impulsado a explorar y ver cosas nuevas.

El espacio es un lugar extremadamente escabroso, es por lo cual es todo un milagro para empezar, que exista vida en la Tierra. La Luna es muy pequeña para tener un satélite, pero nosotros podemos transformer Marte para crear una, y hacerla segura de las radiaciones y agradable a las visitas. Esto también nos enseñará mucho acerca de los cambios climáticos, y de hecho, hasta que no hayamos transformado Marte , voy a asumir que los alarmistas del calentamiento global realmente no saben todavía de lo que están hablando.[2] Una de las clases en ingeniería es que uno no sabe cómo funcionan las cosas hasta que las hace una vez.

Transformar Marte como la Tierra puede sonar hoy como una idea tonta, pero es simplemente otra tarea de ingeniería.[3] Trabajé en muchos grupos diferentes en Microsoft, aun cuando los conjuntos de algoritmos relacionados a las bases de datos son completamente diferentes de los de los motores de textos, todos son problemas de ingeniería y el enfoque es el mismo: desglosar el problema y analizar cada pedazo. (Una de las interesantes lecciones que aprendí en Microsoft fue la diferencia entre la vida real y las pruebas estandarizadas En una prueba estandarizada, si una pregunta parece difícil, puedes saltártela y continuar avanzando de manera que no pierdas un tiempo preciado. En Microsoft, podíamos pasar por alto los problemas fáciles y concentrarnos en las dificiles).

La ingeniería enseña que hay un número infinito de formas de resolver un problema, cada uno con diversos cambios; puede tomar 1,000 años transformar Marte si fueramos a mandar una tonelada de material, pero solo 20 años si mandaramos 1,000 toneladas de material. Lo que sea que al final terminemos haciendo, los primeros humanos en visitar Marte estarán felices de que nosotros lo tornamos verde por ellos. Esta es otra forma de la que nuestra generación puede de dejar su marca.

Un elevador especial es un mega-proyecto factible, pero no hay ningún progreso más allá de unos pocos libros y conferencias porque las pocas personas en este planeta que son capaces de iniciar este proyecto no está conciente de la factibilidad de la tecnología.

---

2 El carbono no es un contaminante y es valioso. Es el 18% de la masa del cuerpo humano, pero solo .03% de la masa de la Tierra. Si el carbono estuviera más extendido, los diamantes serían más baratos. Manejando carros muy rápidos es la mejor forma de desencerrar el carbono que necesitamos. Cualquiera que piense que piense que estamos agotando la energía no entiende el álgebra en $E = mc^2$.

3 La luna de Marte, Phobos, está a solo 3,700 millas por encima de Marte, y si creamos una atmósfera, disminuirá la velocidad y se estrellará. Necesitaremos encontrar un lugar para chocar los fragmentos, yo sugiero en uno de los más grandes cañones que podamos hallar; los pondríamos we could put them next to a cross dipped in urine and call it the largest man-made art.

Brad Edwards, uno de los expertos mundiales sobre elevadores espaciales, tiene un PhD y una década de experiencia diseñando satellites en los Laboratorios Nacionales Los Alamos, y me ha dicho que es incapaz de entrar por las puertas de la dirección de la NASA, o la Fundación Gates, etc. Apartando la nanotecnología, podríamos necesitar solo 5,000 hombres-años de trabajo para cumplir esta tarea, pero nadie que tenga la autoridad para organizar esto entiende que que el elevador espacial es factible.

Glenn Reynolds ha blogueado sobre los elevadores espaciales en su muy influyente Instapundit.com, sobre este tema todavía no se ha hecho un diálogo nacional, y la NASA continúa adelante con sus caras y tontas ideas. Mi libro es una súplica adicional: una vez más, y con sentimientos!

Cómo y Cuándo

> No se deduce que de la separación, de la planificación y la ejecución de una obra, en el análisis del trabajo, el planificador y el ejecutor deberían ser dos personas diferentes. No se deduce que el mundo industrial debería estar dividido en dos clases de personas: unos pocos que decide que debe ser hecho, diseña el trabajo, fija el paso, el ritmo y la marcha, y le encarga a otros que hagan; y los muchos otros que hacen lo que ellos dicen y como lo dicen.
>
> —Peter Drucker

Hay muchos detalles interesantes alrededor del elevador especial, para aquellos que están interesados en detalles adicionales, les recomiendo El Elevador Espacial, escrito en conjunto con Brad Edwards.

El tamaño del primer elevador es una de las grandes preguntas a resolver. Si usted fuera a extender fibra óptica a través del océano Atlántico, repartiría una tonelada de capacidad de ancho de banda. De igual modo, la métrica más importante para nuestro elevador especial es su tamaño.

La otra limitación con los diseños actules es que ellos asumen que los escaladores viajan cientos de millas por hora. Esta es una buena velocidad por carga, pero significa que tomará días en entrar en órbita. Si queremos mandar humanos al espacio en un elevador, necesitamos construir escaladores que puedan viajar a 10,000 millas por hora. Mientras esto parece ridículamente rápido, si usted acelera a esta velocidad por unos minutos, no será irritante. Quizás este sería el desafío para la versión dos si no lo pueden terminar la primera vez.

La sabiduría convencional entre aquellos que piensan que es aun posible es que nos tomará entre 20 y 50 años para construir el elevador especial. Sin embargo, alguien que hace tales predicciones no entiende que la ingeniería es un activo fungible. Dos personas, en general, lograrán algo dos veces tan rápido como una persona.[4] ¿Cómo puedes decir que algo, inequívocamente, tomará cierto tiempo cuando no especificas cuántos recursos se requerirán o cuántas personas planeas asignar para la tarea?

Además, las predicciones por lo general son lejanas. Si le preguntabas a alguien cuánto tiempo le tomaría a voluntarios hacer la Wikipedia tan grande como la Encyclopedia Britannica, nadie hubiera adivinado la respuesta correcta que es dos años y medio. Desde crear un elevador espacial hasta la dominación mundial de Linux, cualquier cosa puede pasar en mucho menos tiempo del que pensamos que es posible si cada cual simplemente juega su papel. La forma de ser parte del futuro es inventarlo, dando riendas sueltas a nuestros científicos y a la energía creativa para con las grandes metas compartidas. Wikipedia, así como nuestra enciplopedia, fue una inspiración para millones de personas, y por eso los recursos se han venido acumulando. La forma de conseguir ayuda es creando una vision que inspire a la gente.

En un período de 75 años, el hombre pasó de usar caballos y vagones a aterrizar en la Luna. ¿Por qué tomaría 30 años construir algo que hoy es 99% factible?

Muchos de los componentes de un elevador espacial son tan simples que los chicos de la Universidad están construyendo prototipos de elevadores en su tiempo libre. El concurso El Elevador:2010 es patrocinado por la NASA, mientras que estos concursos han generado emoción e interés en la prensa, ellos están construyendo juguetes, muy similares a un avión controlado por radio que es un juguete comparado a un avión Boeing.

Creo que podríamos tener un elevador espacial construido en 7 años. Si se divide de tres a cuatro años por personas, y le adicionas algún tiempo de mejoramiento y prueba, usted puede ver como siete años es bastante razonable. El hombre aterrizó en la Luna siete años después del discurso de Kennedy, exactamente como él ordenó,

4 El *The Mythical Man-Month* de Fred Brooks argumenta que adicionar tarde ingenieros a un proyecto lo retrasa, y el tiempo de mejoramiento es solo ruido en un proyecto de ingeniería. También, las wikis, los motores de búsqueda y otras tecnologías inventados desde su libro han disminuido los costos de la colaboración.

porque las fechas pueden ser profecías autocumplidas. Permite a cada uno medirse según sus metas, y determinar si necesitan recursos adicionales.

Si el diseño del hardware y el software fuera hecho de manera pública, otros pudieran tomar los logros intermedios, probarlos y mejorarlos, por tanto ahorrar tiempo de ingeniería. Quizás la NASA podría ocurrírsele cientos de verdaderos y útiles proyectos de investigación para los universitarios para ayudar en vez de alentarlos a contruir juguetes.

La incógnita desconocida son los nanotubos, pero casi todas las otras piezas pueden ser construidas sin tener acceso a a ellos. Solo los necesitaremos enrollados a una gran bobina el día del lanzamiento.

Me puedo imaginar que cualquier esfuerzo como este podría involucrar una gran cantidad de disputas políticas que fácilmente le añadiría al proyecto años. No dejaríamos que esto pasara, y deberíamos recordarnos unos a otros que el elevador es solo el vagón hacia el espacio- la cosa emocionante es el cargamento que está dentro y las posibilidades allá afuera. Un elevador espacial no es un esfuerzo en vano: permitiría muchos otros grandes proyectos que son totalmente irrealizables actualmente. Un elevador espacial permitiría a varias agencias espaciales internacionales que tienen el dinero , pero no un gran propósito, trabajar juntos en una gran y conjunta meta. Y como efecto secundario fortalecerían las relaciones internacionales.[5]

5 Quizás los europeos podrían construir una estación en la GEO. Rusia podría construir la nave para mover el cargamento entre el elevador especial y la Luna. El Medio Oriente podría proveer una red eléctrica para la Luna. China podría encargarse del problema de limpiar el desecho orbital espacial y construir la primera base lunar. África podría atacar el problema de transformar Marte, etc.

Renacimiento del Siglo 21

> Los grandes logros del capitalismo occidental han rebotado principalmente hacia el beneficio de las personas comunes.
>
> —Milton Friedman

> Si hemos aprendido una cosa de la historia de la invención y el descubrimiento, es que, a largo plazo, y a menudo a corto plazo, las profecías mas atrevidas parecen ridículamente conservadoras.
>
> —Arthur C. Clarke, 1951

Como veo, aun vivo en el siglo 20. La historia recordará el siglo 21 como la época en que el hombre entró en el nuevo Renacimiento, y esto no fue solo una foto.

Si pudiéramos remplazar los periodistas por roboes, los prejuicios de los medios de comunicación desaparecerían, y podrían lucir más apuestos.

Incluso en nuestro aun primitivo mundo de hoy, preferiría estar haciendo $30,000 al año que haber prometido $100 millones de dólares si volviera al 1986,y no lo digo por el gran peinado. La internet estaba todavia a siete años de su primera página web en 1986. Es posible que no podamos volver atrás en el tiempo, pero en general no querríamos.

Mucha gente no aprecia cuán rápido el mundo se mueve cada día con la creación de la Internet y otras tecnologías modernas. Algún día: “la única constante es el cambio”, pero esto está errado: la única constant es la aceleración, un aumento del ritmo del cambio. La gente no siente la aceleración todavía porque el mundo se esta moviendo muy lentamente. Los precios de la energía son altos porque no hemos construido una planta de energía nuclear en 30 años (culpe a los Demócratas), y construir plantas de energía nuclear supuestamente toma 15 años (culpe a los burácratas). Tomó décadas para extender la televisión de alta definición (HDTV, siglas en inglés para *high definition television*). Las comidas genéticamente modificadas son tratadas como Frankensteins de la naturaleza. Las compañias farmacéuticas contratan tantos abogados como investigadores, y luego nosotros nos preguntamos por qué las medicinas cuestan tanto, y por qué toma tantos años liberar nuevas al mercado. El aparato está desintegrado y es estúpido. Esperar en la cola está considerado parte normal de la vida.

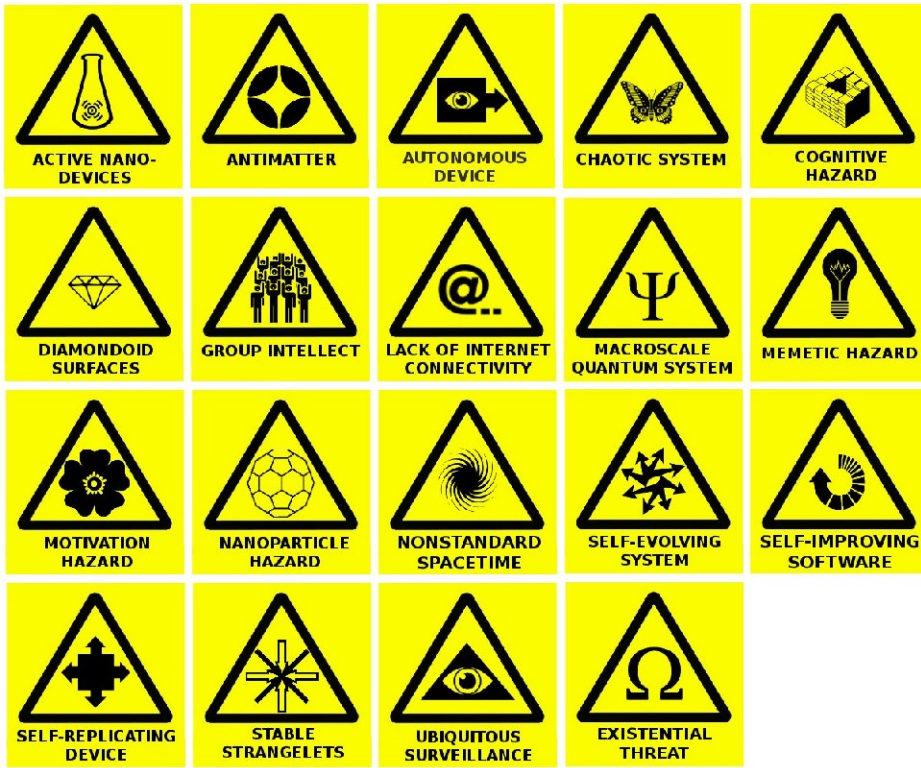
La diferencia entre el rico y el pobre no causa inestabilidad social porque Bill Gates no vive materialmente mejor que ninguno. Sus lentes de contactos no son mejores que los mios, y ninguno de nosotros posee un carro manejado por un robot.

Con mejor cooperación y mejores herramientas, el Renacimiento del siglo 21 está esperando por nosotros. El mundo será caracterizado por ser la economía regalo, así que resista la tentación de acumular. Una vez y cruzamos esa línea; dejamos atrás nuestro juicio y nuestra felicidad. ¡Apurémonos!

No temer al futuro; el hombre fue creado para resolver problemas y divertirse en el proceso. La pregunta de si los Estados Unidos será pertinente en el siglo 21 depende de las respuestas a estas simples preguntas: ¿tendrá los mejores científicos, están ellos aprendiendo unos de otros, y por tanto trabajando juntos?

Fin

Señales de Advertencias Del Futuro



Este es nuestro espeluznante futuro, y desafortunadamente la humanidad está avanzando lentamente hacia él.[6]

6 De http://lifeboat.com/ex/warning.signs.for.tomorrow, creado por Anders Sandberg. Envié múltiples correos electrónicos a direcciones electrónicas del the Lifeboat Foundation para intentar obtener permiso para usar estas imágenes, pero no reciví respuesta. Quizás ellos están muy ocupados con su misión de " alentar a los progresos científicos para ayudar a que la humanidad sobreviva a los riesgos existenciales" como para responder. Por eso doné $200 por sus usos aquí :-)

Vamos hacia adelante con completa confianza en el eventual triunfo de la libertad. No porque la historia vaya en ruedas de la inevitabilidad; son las decisiones humanas las que mueven los eventos. No porque nos consideremos una nación escogida; Dios mueve y escoge a voluntad. Tenemos confianza porque la libertad es la permanente esperanza de la humanidad.

Cuando nuestros fundadores declararon un nuevo orden de las eras; cuando los soldados murieron la oleada de una unión basada en la libertad; cuando los ciudadanos marcharon con indignación pacífica bajo el lema "Libertad ahora" - estaban actuando bajo una antigua esperanza que tienen la intención de que se cumpla. La historia tiene un flujo y reflujo de la justicia, pero la historia también tiene una dirección visible, fijada por la libertad y el Autor de la Libertad.

—Discurso Inaugural Presidencial de los EE.UU., 2005

El genio es la paciencia eterna.

El mayor peligro para la mayoría de nosotros no se encuentra en el establecimiento de nuestro objetivo demasiado alto y quedarnos cortos, sino en establecerlo demasiado bajo, y lograr nuestro objetivo.

La verdadera obra de arte no es más que una sombra de la perfección divina.

—Michelangelo

La mayoría de las cosas que vale la pena hacer en el mundo se han declarado imposibles antes de que se realizaran.

—Louis Brandeis, Suprema Corte de Justicia de los Estados Unidos

Tenemos una sola alternativa: o bien construir una sociedad industrial funcional o ver la libertad misma desaparecer en la anarquía y la tiranía.

—Peter Drucker

La mente no es un vaso por llenar sino un fuego a ser encendido.

—Plutarco