En el principio… fue la línea de comandos

Neal Stephenson

1999

Índice

Presentación

Hace años que los fabricantes de sistemas operativos —como Microsoft o Apple— dedican ingentes recursos a ocultar cómo funcionan realmente los ordenadores, se supone que con la idea de simplificar su uso. Para ello, algunos de sus mejores ingenieros han inventado toda clase de metáforas visuales e interfaces gráficas, lo cual ha permitido que mucha gente se acerque a los ordenadores personales sin sentir pánico o sin provocar grandes gastos de formación de personal a sus empresas. Pero, lamentablemente, construir ese muro de metáforas en forma de interfaz gráfica entre el ordenador y el usuario (conocida como GUI*) ha tenido un coste social y cultural muy notable, al contribuir decisivamente a que la tecnología que subyace al ordenador se perciba como algo mágico, sin conexión alguna entre causas y efectos, recubriendo de un formidable manto de ignorancia todo lo que realmente sucede. Eso ha propiciado estrategias comerciales basadas en el engaño y la trampa,*[1] *cuando no abiertamente delictivas*[2] *y explica que productos muy deficientes, como el propio Windows, sean consumidos masivamente y tolerados por el gran público, que soporta resignadamente una mercancía plagada de errores y sin garantía* real *alguna, que acepta las pérdidas de da-*

[1] «Trampa en el ciberespacio», Roberto Di Cosmo, 1998, `http://sindominio.net/biblioweb/telematica/trampas.html`

[2] El fallo contra Microsoft del juez Jackson, emitido en abril de 2000, es absolutamente demoledor: califica su estrategia empresarial como «conducta depredadora», la compara con un «pulgar opresor» sobre sus competidores y la tilda de «violenta».

tos, los virus, las vulnerabilidades, el control sobre su intimidad y toda clase de errores inesperados como algo natural, inherente al propio ordenador, y no al sistema operativo que lo hace funcionar. El último —y gravísimo— atropello planificado por parte del principal constructor de interfaces amigables *tiene el nombre de TCPA/Palladium y pretende universalizar el software propietario con código malicioso incorporado. Hoy son las empresas las que «legislan» de facto mediante la tecnología y, de imponerse dicho sistema —una auténtica conspiración de Microsoft e Intel contra libertades básicas de las personas—, permitiría* realmente *la censura remota, la intrusión y el control de los ordenadores personales por parte de las corporaciones multimedia y de los gobiernos, a espaldas del usuario y sin su consentimiento.*

La «cultura de la interfaz» se ha impuesto, pero para llegar a ese punto ha hecho falta un largo recorrido salpicado de guerras no declaradas, una auténtica «lucha de clases en el escritorio» que nos ha llevado desde la línea de comandos hasta las vistosas interfaces gráficas actuales. Es precisamente esa historia la que nos narra, de forma amena y desenfadada, Neal Stephenson, autor por cierto de algunas de las mejores novelas de ciencia-ficción de la última década, tales como Snow Crash *y* Criptonomicón.

> Existe una comunidad, una cultura compartida, de programadores expertos y gurús de redes, cuya historia se puede rastrear décadas atrás, hasta las primeras minicomputadoras de tiempo compartido y los primigenios experimentos de Arpanet. Los miembros de esta cultura acuñaron el término *hacker.* Los hackers construyeron la Internet. Los hackers hicieron del sistema operativo Unix lo que es en la actualidad. Los hackers hacen andar Usenet. Los hackers hacen que funcione la WWW.[3]

El heredero de esa cultura es el movimiento del software libre, y su buque insignia: GNU/Linux. En ese ámbito sigue muy viva la interfaz de línea de comandos de la que nos habla Stephenson. Tal circunstancia no responde a ninguna clase de nostalgia o excentricidad, ni se debe solo a una decisión técnica, sino política, pues con ello se ha mantenido intactos el poder y la capacidad de

[3] «Cómo convertirse en hacker», Eric Raymond, 2001. La traducción castellana puede leerse en: `http://sindominio.net/biblioweb/telematica/hacker-como.html`

decisión del usuario sobre lo que hace su máquina. Es de esta historia, no muy conocida fuera del ámbito hacker, sobre la que nos ilustra EN EL PRINCIPIO... FUE LA LÍNEA DE COMANDOS. *La obra que presentamos constituye un ensayo sobre el pasado y el futuro de los ordenadores personales, un recorrido personal y subjetivo —pero no por ello menos preciso— a través de la evolución de los sistemas operativos que el autor ha conocido —Windows, MacOS, Linux, BeOS— y de la actitud que han representado a lo largo del tiempo cada uno de estos en el uso y el tipo de usuario a los que ha dado lugar. No es un libro que trate de evaluar o comparar técnicamente las prestaciones de los distintos sistemas operativos, ni que aborde la típica (y artificiosa) controversia entre usuarios de Mac y de Windows. De hecho, Stephenson sitúa correctamente en el mismo plano a Apple y a Microsoft, como dos caras de la misma moneda: tal y como no hay diferencia cualitativa entre un fabricante de ferraris y otro de ladas (por mucho que estética e incluso funcionalmente no haya comparación posible), tampoco la hay entre Redmond y Cupertino: ambos gigantes representan un modelo basado en el código cerrado, en la restricción y la apropiación de las fuentes del conocimiento y en la venta de licencias.*

La alternativa al software propietario no es otro software propietario que funcione mejor o sea más vistoso, o nos salga gratis, sino un modelo de desarrollo y uso del software que devuelva a los usuarios de ordenadores el poder y la libertad que han ido perdiendo a lo largo del tiempo o, aún más, que permita a los usuarios autoorganizarse para ello: ese, y no otro, es el valor del software libre, mucho más que sus excelencias técnicas, las cuales, siendo indiscutibles, no dejan de ser un hecho circunstancial. ¿Y qué es lo que caracteriza pues al software libre? el permiso de copiar, modificar y redistribuir el código (incluyendo su venta), con una única restricción que se puede sintetizar con el título del himno de Caetano Veloso y del Mayo francés: «prohibido prohibir», y que los hackers comprimen aún más llamándolo «copyleft». Esto no es una simple utopía de informáticos libertarios, sino la columna vertebral de Internet (más del 60 % de los servidores web se basan en un software libre llamado Apache), el modelo de negocio de numerosas empresas y el sistema que usan ya más de veinte millones de personas en sus ordenadores.

Esta obra sin duda supondrá un punto de vista novedoso para el usuario no especializado, pues le descubrirá de modo ameno un mundo que no es el que le han contado en las revistas de informática, ni en los rutilantes anuncios de las grandes compañías de software propietario, que prometen facilidad de uso a cambio de aceptar la entrega ciega e incondicional a sus productos. Neal Stephenson muestra que no es oro todo lo que reluce debajo de esa metáforas visuales y esos vistosos y (se supone) intuitivos escritorios, que se han impuesto a costa de un ejercicio tramposo de idealización equivalente a las películas de Walt Disney.

Hay que hacer una pequeña aclaración en cuanto a la excelente traducción de Asunción Álvarez. En el texto aparece a menudo «software gratis» como traducción castellana de free software. *En inglés, el término* free *es polisémico, y puede significar tanto* libre *como* gratis. *Sin embargo,* free software, *referido al movimiento que abandera GNU/Linux, se emplea* siempre *en el sentido de libertad, no de precio, y debe traducirse como «software libre». Pero Stephenson usa muchas veces a lo largo del texto* free *en un sentido inequívoco que indica gratuidad y por supuesto la traductora ha respetado dicho sentido. Cuando el autor quiere referirse a «software libre» opta por la denominación* open source *(«fuente abierta»). El software libre es libre incluso para ser vendido. Que el software se pueda copiar sin restricciones hace que tienda a llegar al usuario a coste cero, lo cual es distinto a que no haya costado nada producirlo o a que alguien no haya pagado por su desarrollo: la gratuidad, cuando se da, es una consecuencia del modelo de libre copia, no su razón de ser.*[4]

Para elaborar este libro se ha empleado únicamente software libre, en concreto el sistema de composición de textos LaTeX,[5] *el*

[4]De hecho existe software gratuito que en absoluto es software libre: el navegador Explorer de Microsoft es un buen ejemplo de cómo la gratuidad puede ser parte de una despiadada estrategia de *dumping*.

[5] TeX (pronúnciese «tej») fue creado en 1978 por Donald E. Knuth, figura sobresaliente en la ciencia de la computación moderna y máxima autoridad en el estudio de algoritmos matemáticos. TeX es sin duda uno de los programas libres más perfectos y de los que más orgullosos se sienten los amantes del software libre. LaTeX (y su sucesor LaTeX 2_ε) es un lenguaje estructurado construido a partir de TeX, usado por gran número de matemáticos, físicos, químicos e ingenieros, si bien se puede emplear en

editor GNU Emacs y el corrector Ispell,[6] *con los que se ha controlado todo el proceso hasta la salida final en un fichero «postcript» para la imprenta. Tenemos el empeño explícito por mostrar con hechos que el resultado de la maquetación con herramientas libres es incluso superior que el que se obtiene con los carísimos programas comerciales que se utilizan en la composición de libros en papel. Tampoco se ha usado interfaz gráfica: todo el proceso se ha realizado sin efectuar un solo click de ratón desde una terminal de línea de comandos (GNU bash). Una versión digital de este libro, libremente reproducible para uso personal, puede encontrarse en la Biblioweb de sinDominio.*[7]

Solo nos queda agradecer la cesión de la traducción a Asunción Álvarez y ciberpunk.org, en cuyo sitio se encuentra otra versión en línea de este ensayo.[8] *También deseamos que conste nuestro agradecimiento a Pedro Jorge Romero, por permitirnos reproducir la reseña que hizo para el Archivo de Nessus.*[9]

MIQUEL VIDAL
miquel@sindominio.net

cualquier tipo de documento. Aunque Word está haciendo estragos, aún muchas revistas de Física y Matemática o, por ejemplo, los libros de la editorial Addison-Wesley, se preprocesan utilizando TEX.

[6] Ispell es un programa antiquísimo de línea de comandos que también trabaja integrado en Emacs. Fue escrito originalmente para una máquina PDP-10 en 1971 por R.E. Gorin y reescrito en C por Pace Willisson, del MIT. Después de 30 años, sigue siendo el corrector ortográfico estándar de los sistemas Unix.

[7] `http://sindominio.net/biblioweb/telematica`

[8] `http://www.ciberpunk.com/basicos/neal_stephenson.html`

[9] `http://www.archivodenessus.com/rese/0186/`

Prólogo

Aparte de escribir buenas novelas de ciencia ficción (o cómo se llame lo que hace), Neal Stephenson tiene otra faceta más periodística. No está tan marcada como la de Bruce Sterling, quien ha dedicado muchos esfuerzos a informar desde cinco minutos en el futuro, pero es muy interesante, centrándose sobre todo en el mundo de la informática y las tecnologías avanzadas de comunicación. Y aquí es donde Neal Stephenson gana a muchos de los que tratan esos temas: él realmente entiende el fundamento técnico. No es que sus comentarios sean análisis secos de posibilidades tecnológicas, más bien todo lo contrario. Son piezas llenas de opiniones, subjetivas y claramente escritas por una persona en concreto, pero una persona cuya opinión merece tenerse en cuenta porque demuestra conocer bien el campo sobre el que escribe.

Un buen ejemplo es este libro dedicado a los sistemas operativos. EN EL PRINCIPIO... FUE LA LÍNEA DE COMANDOS es una combinación de historia del software, discusión sobre la progresiva ocultación de la realidad tras una «interfaz» cada vez más bonita, meditación sobre el sentido de la vida, diario de los problemas de enfrentarse a varios sistemas operativos diferentes, canto nostálgico a los días en que las cosas se hacían como debían hacerse y, un poco, defensa de los muy masculinos valores de la potencia y el control.

Todo empieza con una analogía: los sistemas operativos son como los coches. La compañía Microsoft empezó vendiendo bicicletas motorizadas (MS-DOS), luego pasó a producir una actualización (el Windows original) que permitía a la bicicleta ir más rápido. Y finalmente, produce un coche, no demasiado bonito, que pierde mucho aceite pero que la gente compra mucho. La otra compañía, Apple, vende unos coches muy cómodos, fáciles de usar, pero que vienen herméticamente cerrados de forma que es imposible saber qué hay en su interior. BeOS vende coches de alta tecnologías, hermosos, con gran estilo y capaces de volar, ir por el agua o hacer lo que uno quiera, y más baratos que la competencia. Y por último tenemos algo que no es ni siquiera una compañía, sino más bien un campamento de refugiados, lleno de voluntarios de gran talento, que produce tanques. Sí, tanques. Tan buenos, que nunca se rompen, fáciles de maniobrar, que consumen el mismo combustible que un coche, están fabricados con la última tecnología y, lo mejor de todo, son gratuitos. A medida que uno de esos tanques Linux, ¿no lo habían adivinado?, se termina, se deja en la calle y cualquiera puede llevárselo.

A partir de ahí, Neal Stephenson construye un discurso en el que explica el valor real de una compañía de sistemas operativos (ninguno; su valor sólo está en la cabeza de los clientes que, como Mulder, «quieren creer»), analiza la necesidad de la sociedad americana (y por extensión, el resto del mundo) de ocultar la complejidad tras unos bonitos botones, y discute los muchos problemas de instalar Linux. Y cuando uno sospechaba que está a punto de defender los valores de las herramientas para hombres (después de comparar a Linux con un, maravilloso en su experiencia, taladro industrial) se descuelga con una afirmación sorprendente para un hacker: el mejor sistema operativo sería aquel que combinase la potencia con una buena interfaz gráfica. Es decir, uno que te dejase la posibilidad de abrir una ventana a la línea de comando. Es decir, BeOS.

Porque la línea de comando es la mejor forma de relacionarse con el mundo. La línea de comando es lo que te permite acceder a la realidad fundamental. Seguro que dios cuando creo el universo lo hizo como un hacker delante de la pantalla de su ordenador tecleando crípticos comandos para crear universos.

¿Son 150 páginas de un discurso laberíntico? Muy posiblemente. ¿Tiene razón en lo que dice? En buena parte. ¿Se va por las ramas? Ciertamente. ¿Es apasionante de leer? Puedes apostarlo. Porque EN EL PRINCIPIO... FUE LA LÍNEA DE COMANDOS está escrito con pasión, y por un autor que sabe utilizar atrevidas metáforas y brillantes imágenes, que a cada página puede sorprender con una observación inteligente o un dato interesante. Cuando terminas, te quedas con el inexplicable deseo de instalar BeOS en tu ordenador. Lo que puede resumirse diciendo que es otro buen libro de Neal Stephenson.

PEDRO JORGE ROMERO[1]

[1]Pedro Jorge Romero (Arrecife, 1967) es licenciado en física, pero realmente se dedica a traducir, a la programación web y a escribir ocasionalmente. Ha traducido los tres volúmenes de la monumental novela de N. Stephenson *Criptonomicón* (Ediciones B, 2002), y está preparando la traducción de su esperada «secuela», *Azogue* (*Quicksilver*), cuya publicación está prevista para octubre de este año. Ediciones B ha publicado recientemente su primera novela, *El otoño de las estrellas*, escrita en colaboración con Miquel Barceló.

Introducción

HACE UNOS VEINTE AÑOS, a Jobs y Wozniak, los fundadores de Apple, se les ocurrió la muy extraña idea de vender máquinas de procesamiento de información para uso doméstico. El negocio despegó, sus fundadores hicieron un montón de dinero y recibieron el crédito que merecían como osados visionarios. Pero en esa misma época, a Bill Gates y Paul Allen se les ocurrió una idea todavía más extraña y fantasiosa: vender sistemas operativos de ordenador. Esto era mucho más extraño que la idea de Jobs y Wozniak. Un ordenador por lo menos tenía cierta realidad física. Venía en una caja, podía abrirse y enchufarse y se podía ver cómo parpadeaban las luces. Un sistema operativo no tenía ninguna encarnación tangible. Venía en un disco, claro, pero el disco no era, a todos los efectos, más que la caja que contenía el sistema operativo. El producto mismo era una serie muy larga de unos y ceros que, cuando se instalaba y se cuidaba bien, te daba la capacidad de manipular otras series muy largas de unos y ceros. Incluso los pocos que de hecho comprendían qué era un sistema operativo de ordenador posiblemente pensaban en ello como un prodigio increíblemente complicado de la ingeniería, como un reactor o un avión espía U-2, y no algo que pudiera llegar a ser (en la jerga de la alta tecnología) *productizado.*

Pero ahora la compañía que fundaron Gates y Allen vende sistemas operativos como Gillette vende hojas de afeitar. Se lanzan nuevas versiones de sistemas operativos como si fueran películas de Hollywood, con el respaldo de celebridades, apariciones en *talk shows* y giras mundiales. Su mercado es lo bastante vasto como para que la gente se preocupe de si ha sido monopolizado por una compañía. Incluso los menos inclinados a la técnica de nuestra sociedad tienen ahora al menos una idea nebulosa de lo que hacen los sistemas operativos; lo que es más, tienen opiniones sólidas sobre sus méritos relativos. Es ya un conocimiento compartido el que, si tienes un programa que funciona en tu Macintosh y lo pasas a una máquina Windows, no funciona. Esto sería, de hecho, un error risible e idiota, como clavar herraduras en las ruedas de un coche.

Una persona que entrara en coma antes de la fundación de Microsoft y despertara hoy, tomaría el *New York Times* de esta mañana y no entendería nada —o casi:

> Ítem: el hombre más rico del mundo hizo su fortuna a partir de ¿qué? ¿ferrocarriles? ¿buques? ¿petróleo? No, sistemas operativos.
> Ítem: el Departamento de Justicia está investigando el supuesto monopolio en sistemas operativos de Microsoft con herramientas legales que se inventaron para restringir el poder de los jefes de bandas de ladrones del siglo XIX.
> Ítem: una amiga mía me contó recientemente que había interrumpido un (hasta entonces) estimulante intercambio de e-mails con un joven. «Al principio parecía un tipo tan inteligente e interesante —dijo— pero luego empezó a ponerse en plan "PC-contra-Mac".»

¿Qué diablos está pasando aquí? Y ¿tiene futuro el negocio de los sistemas operativos, o sólo pasado? Lo que sigue es mi opinión, que es completamente subjetiva; pero, dado que me he pasado bastante tiempo, no sólo usando, sino programando en Macintosh, Windows, Linux y BeOS, tal vez no sea tan desinformada como para carecer por completo de valor. Este es un ensayo subjetivo, más crítica que artículo de investigación, y puede parecer injusto o sesgado comparado con lo que se puede encontrar en las revistas de PC. Pero, desde que salió el Mac, nuestros sistemas operativos están basados en metáforas, y, por lo que a mí respecta, es legítimo cuestionar cualquier cosa con metáforas dentro.

Descapotables, tanques y batmóviles

EN LA ÉPOCA EN QUE JOBS, Wozniak, Gates y Allen estaban soñando estos planes inverosímiles, yo era un adolescente que vivía en Ames, Iowa. El padre de uno de mis amigos tenía un viejo MGB descapotable[1] oxidándose en el garaje. A veces conseguía que arrancara y cuando lo hacía nos llevaba a dar una vuelta por el barrio, con una expresión memorable de salvaje entusiasmo juvenil en la cara; para sus preocupados pasajeros era un loco, tosiendo y renqueando por Ames, Iowa, y tragándose el polvo de oxidados Gremlins y Pintos, pero en su propia imaginación era Dustin Hoffman cruzando el Puente de la Bahía con el cabello al viento.

Mirando atrás, esto me reveló dos cosas acerca de la relación de las personas con la tecnología. Una fue que el romanticismo y la imagen influyen mucho sobre su opinión. Si lo dudan (y tienen un montón de tiempo libre), pregúntenle a cualquiera que tenga un Macintosh y que por ello imagina ser miembro de una minoría oprimida.

El otro punto, algo más sutil, fue que la interfaz es muy importante. Claro que aquel MGB era un coche malísimo en

[1]El MGB fue el coche deportivo británico más exitoso de todos los tiempos. Salió de la producción en Abingdon en 1962. Se fabricó también una versión coupé con la denominación MGB GT. La producción se suprimió en 1980, después de haber vendido medio millón de unidades. [*N. del E.*]

casi cualquier aspecto importante: pesado, poco fiable, poco potente. Pero era divertido conducirlo. Respondía. Cada guijarro de la carretera se sentía en los huesos, cada matiz en el asfalto se transmitía instantáneamente a las manos del conductor. Podía escuchar el motor y saber qué fallaba. El volante respondía inmediatamente a las órdenes de las manos. Para nosotros, los pasajeros, era un ejercicio fútil de no ir a ningún lado —más o menos tan interesante como mirar por encima del hombro de alguien que introduce números en una hoja de cálculo—. Pero para el conductor era una experiencia. Durante un breve tiempo, estaba expandiendo su cuerpo y sus sentidos en un ámbito más amplio, y haciendo cosas que no podía hacer sin ayuda.

La analogía entre coches y sistemas operativos es bastante buena, así que permítanme seguir con ella durante un rato como modo de dar un resumen sumario de nuestra situación hoy en día.

Imagínense un cruce de carreteras donde hay cuatro puntos de venta de coches. Uno de ellos (Microsoft) es mucho, mucho mayor que los demás. Comenzó hace años vendiendo bicicletas de tres velocidades (MS-DOS); no eran perfectas, pero funcionaban y, cuando se rompían, se arreglaban fácilmente.

Enfrente estaba la tienda de bicicletas rival (Apple), que un día empezó a vender vehículos motorizados: coches caros, pero de estilo atractivo, con los mecanismos herméticamente sellados, de tal modo que su funcionamiento era algo misterioso.

La tienda grande respondió apresurándose a sacar un kit de actualización (el Windows original) al mercado. Se trataba de un dispositivo que, cuando se atornillaba a una bicicleta de tres velocidades, le permitía seguir, a duras penas, el ritmo de los coches Apple. Los usuarios tenían que usar gafas de protección y siempre estaban sacándose bichos de los dientes,[2] mientras los usuarios de Apple corrían en su confort herméticamente sellado, burlándose por

[2]El autor juega en este y en otras partes del ensayo con la doble acepción de *bug:* «bicho, insecto» y «error, fallo informático». [*N. del E.*]

las ventanillas. Pero los Micro-motopedales eran baratos, y fáciles de reparar comparados con los coches Apple, y su cuota de mercado creció.

Al final la tienda grande acabó por sacar un coche en toda regla: un monovolumen colosal (Windows 95). Tenía el encanto estético de un bloque soviético de viviendas para obreros, perdía aceite y le estallaban las bujías, pero fue un éxito tremendo. Poco tiempo después, sacaron también un enorme vehículo para la circulación fuera de carretera destinado a usuarios industriales (Windows NT), que no era más bonito que el monovolumen, y sólo algo más fiable.

Desde entonces ha habido un montón de ruido y gritos, pero poco ha cambiado. La tienda pequeña sigue vendiendo elegantes sedanes de estilo europeo y gastándose mucho dinero en campañas publicitarias. Tienen carteles de «¡LIQUIDACIÓN!» puestos en el escaparate desde hace tanto tiempo que ya están amarillos y arrugados. La tienda grande sigue fabricando monovolúmenes y vehículos de circulación fuera de carretera cada vez más grandes.

Al otro lado de la carretera hay dos competidores que llegaron más recientemente. Uno de ellos, (Be, Inc.) vende batmóviles plenamente operativos (los BeOS). Son más bonitos y elegantes incluso que los eurosedanes, mejor diseñados, más avanzados tecnológicamente y al menos tan fiables como cualquier otra cosa en el mercado: y sin embargo son más baratos que los demás.

Con una excepción, claro: Linux, que está enfrente mismo, y que no es un negocio en absoluto. Es un conjunto de tiendas de campaña, yurtas, tipis y cúpulas geodésicas levantadas en un prado y organizadas por consenso. La gente que vive allí fabrica tanques. No son como los anticuados tanques soviéticos de hierro forjado; son más parecidos a los tanques M1 del ejército estadounidense, hechos de materiales de la era espacial y llenos de sofisticada tecnología de arriba abajo. Pero son mejores que los tanques del ejército. Han sido modificados de tal modo que nunca, nunca se averían, son lo bastante ligeros y maniobrables como para usarlos en la calle y no consumen más combustible que un coche compacto. Estos tanques se producen ahí mismo a un ritmo aterrador, y hay un número enorme de ellos alineados junto a la carretera con las llaves puestas. Cualquiera que quiera puede simplemente montarse en uno y marcharse con él gratis.

Los clientes llegan a este cruce en multitudes, día y noche. El noventa por ciento se van derechos a la tienda grande y compran monovolúmenes o vehículos para circulación fuera de carretera. Ni siquiera miran las otras tiendas.

Del diez por ciento restante, la mayoría va y compra un elegante eurosedán, deteniéndose sólo para mirar por encima del hombro a los filisteos que compran monovolúmenes y vehículos para circulación fuera de carretera. Si acaso llegan a fijarse siquiera en la gente al otro lado de la carretera, vendiendo los vehículos más baratos y técnicamente superiores, estos clientes los desprecian, considerándolos lunáticos y descerebrados.

La tienda de batmóviles vende unos cuantos vehículos al maniático de los coches de ocasión que quiere un segundo vehículo además de su monovolumen, pero parece aceptar, al menos de momento, que es un jugador marginal.

El grupo que regala los tanques sólo permanece vivo porque lo llevan voluntarios, que se alinean al borde de la calle con megáfonos, tratando de llamar la atención de los clientes sobre esta increíble situación. Una conversación típica es algo así:

HACKER CON MEGÁFONO: ¡Ahorra dinero! ¡Acepta uno de nuestros tanques gratis! ¡Es invulnerable, y puede atravesar roquedales y ciénagas a ciento cincuenta kilómetros por hora consumiendo dos litros a los cien!

FUTURO COMPRADOR DE MONOVOLUMEN: Ya sé que lo que dices es cierto… pero… eh… ¡yo no sé mantener un tanque!

MEGÁFONO: ¡Tampoco sabes mantener un monovolumen!

COMPRADOR: Pero esta tienda tiene mecánicos contratados. Si le pasa algo a mi monovolumen, puedo tomarme un día libre de trabajo, traerlo aquí y pagarles para que trabajen en él mientras yo me siento en la sala de espera durante horas, escuchando música de ascensor.

MEGÁFONO: ¡Pero si aceptas uno de nuestros tanques gratuitos te mandaremos voluntarios a tu casa para que lo arreglen gratis mientras duermes!
COMPRADOR: ¡Manténte alejado de mi casa, bicho raro!
MEGÁFONO: Pero. . .
COMPRADOR: ¿Pero es que no ves que todo el mundo está comprando monovolúmenes?

Lanzador de bits

LA CONEXIÓN ENTRE COCHES y modos de interactuar con los ordenadores no se me habría ocurrido en la época en que me llevaban de paseo en aquel descapotable. Me había apuntado a una clase de programación en el Instituto de Ames. Tras unas cuantas clases introductorias, nos dieron permiso a los estudiantes para entrar en una sala diminuta que contenía un teletipo, un teléfono y un módem anticuado consistente en una caja de metal con un par de cuencas de plástico encima (nota: muchos lectores, abriéndose camino a través de esta última oración, probablemente sintieron un retortijón inicial de temor de que este ensayo estuviera a punto de convertirse en una tediosa batallita sobre lo difícil que lo teníamos en los viejos tiempos; tranquilícense: lo que estoy haciendo, de hecho, es colocar mis piezas sobre el tablero de ajedrez, por así decirlo, preparándome para realizar una observación sobre temas realmente interesantes y actualizados como el software de fuente abierta. El teletipo era exactamente el mismo tipo de máquina que se había estado usando durante décadas para enviar y recibir telegramas. Se trataba básicamente de una máquina de escribir ruidosa que sólo podía generar LETRAS MAYÚSCULAS. Montada a un lado había una máquina más pequeña con un largo rollo de cinta de papel y una cesta de plástico transparente debajo.

Para conectar este dispositivo (que no era un ordenador en absoluto) con la Universidad Estatal de Iowa al otro lado de la ciudad, había que coger el teléfono, marcar el número del ordenador, esperar a que llegaran ruidos raros y entonces colocar el auricular en las cuencas de plástico. Si acertabas, una cuenca envolvía sus labios de neopreno en torno a la parte de la oreja y el otro en torno a la parte de la boca, consumando una especie de *sesenta y nueve* informacional. El teletipo se estremecía mientras era poseído por el espíritu del lejano ordenador, y empezaba a martillear mensajes crípticos.

Puesto que el tiempo de ordenador era un recurso escaso, usábamos una especie de técnica de procesamiento por lotes. Antes de marcar en el teléfono, conectábamos la perforadora de cinta (una máquina subsidiaria atornillada al costado del teletipo) y tecleábamos nuestros programas. Cada vez que pulsábamos una tecla, el teletipo imprimía una letra en el papel delante nuestro, de tal modo que pudiéramos leer lo que habíamos escrito; pero al mismo tiempo convertía la letra en un conjunto de ocho dígitos binarios, o bits, y perforaba un patrón correspondiente de agujeros a lo ancho de una cinta de papel. Los diminutos discos de papel salidos de la cinta caían en la cesta de plástico transparente, que lentamente se llenaba de lo que sólo puede describirse como bits reales. El último día del curso, el chico más listo de la clase (no yo) saltó desde detrás de su pupitre y lanzó varios kilos de estos bits por encima de la cabeza de nuestro profesor, como confetti, como una especie de broma semiafectuosa. La imagen de aquel hombre sentado allí, atenazado por las fases iniciales de una atávica reacción de lucha-o-huye, con millones de bits (megabytes) cayéndole por el pelo y metiéndosele por la nariz y la boca, el rostro poniéndosele morado a medida que se aproximaba a la explosión, es la escena más memorable de mi educación formal.

De cualquier modo, resultará obvio que mi interacción con el ordenador fue de una naturaleza extremadamente formal, que estaba dividida en diferentes fases, a saber: 1) sentado en casa con lápiz y papel, a kilómetros de distancia de cualquier ordenador, pensaba mucho acerca de lo que quería que hiciera el ordenador y traducía mis intenciones a un lenguaje informático —una serie de símbolos alfanuméricos sobre la página—; 2) llevaba esto a través de una especie de «cordón sanitario» informacional (cinco kilómetros a través de tormentas de nieve) hasta el colegio e introducía aquellas letras en una máquina —no un ordenador— que convertía los símbolos en números binarios y los registraba visiblemente en cinta; 3) entonces, mediante el módem de las cuencas de goma, enviaba aquellos números al ordenador de la universidad, que 4) hacía aritmética con ellos y devolvía números diferentes al teletipo; 5) el teletipo convertía estos números de nuevo en letras y los martilleaba en una página, y 6) yo, mirando, interpretaba las letras como símbolos significativos.

El reparto de responsabilidades que todo esto conlleva es admirablemente limpio: los ordenadores hacen aritmética con bits de información. Los humanos interpretan los bits como símbolos significativos. Pero está distinción está desdibujándose, o al menos complicándose, con la llegada de los sistemas operativos modernos que usan, y frecuentemente abusan, del poder de la metáfora para hacer los ordenadores disponibles para un público más amplio. Por el camino —posiblemente debido a estas metáforas, que hacen de un sistema operativo una especie de obra de arte— la gente empieza a ponerse emotiva y le toma cariño a fragmentos de software del mismo modo que el padre de mi amigo le tenía cariño a su descapotable.

Puede que la gente que sólo ha interactuado con un ordenador a través de interfaces gráficas de usuario como MacOS o Windows —es decir, casi cualquiera que haya usado un ordenador— le haya sorprendido, o al menos llamado la atención, lo de la máquina de telégrafos que yo usaba para comunicarme con un ordenador en 1973. Pero había, y hay, una buena razón para usar este tipo particular de tecnología. Los seres humanos disponen de formas diversas de comunicarse *entre sí,* como la música, el arte, la danza y las expresiones faciales, pero algunas de ellas son más susceptibles que otras para expresarse como cadenas de símbolos. El lenguaje escrito es la más fácil porque, por supuesto, ya consiste en cadenas de símbolos *para empezar.*

Si resulta que los símbolos pertenecen a un alfabeto fonético (y no son, por ejemplo, ideogramas), convertirlos en bits es un procedimiento trivial que se fijó tecnológicamente en el siglo XIX, con la introducción del código morse y de otras formas de telegrafía.

Teníamos una interfaz humano/ordenador cien años antes de tener ordenadores. Cuando se crearon los ordenadores en la época de la Segunda Guerra Mundial, los humanos, de modo natural, se comunicaron con ellos, injertándolos en tecnologías ya existentes para traducir letras a bits y viceversa: teletipos y máquinas de tarjetas perforadas.

Estas encarnaban dos enfoques fundamentalmente diferentes de la computación. Cuando se usaban tarjetas, se perforaba todo un taco y se pasaban por el lector a la vez, lo cual se llamaba «procesamiento por lotes». También se podía hacer procesamiento por lotes con un teletipo, como ya he descrito, usando el lector de cinta de papel, y ciertamente se nos animaba a adoptar este enfoque cuando yo estaba en el instituto. Pero —aunque se hacían esfuerzos por mantenernos ignorantes de esto— el teletipo podía hacer algo que el lector de tarjetas no podía. En el teletipo, una vez se establecía el vínculo con el módem, se podía introducir sólo una línea y pulsar la tecla de retorno. El teletipo enviaría entonces esa línea al ordenador, que podía responder o no con líneas propias, que el teletipo martillearía —produciendo, con el tiempo, una transcripción del intercambio mantenido con la máquina—. Este modo de hacerlo ni siquiera tenía nombre entonces, pero cuando, mucho más tarde, apareció una alternativa, se denominó retroactivamente la «Interfaz de Línea de Comandos».

Cuando fui a la universidad, usaba los ordenadores en grandes salas abarrotadas donde manadas de estudiantes se sentaban frente a versiones ligeramente actualizadas de las mismas máquinas y escribían programas informáticos; estos ordenadores usaban mecanismos de impresión por matrices de puntos, pero eran (desde el punto de vista de la máquina) idénticas a los antiguos teletipos. En aquel momento, los ordenadores compartían mejor el tiempo —es decir, los *mainframes* seguían siendo los *mainframes*, pero se comunicaban mejor con un gran número de terminales a la vez—. En consecuencia, ya no era necesario usar procesamiento por lotes. Los lectores de tarjetas fueron desterrados a pasillos y sótanos, y el procesamiento por lotes se convirtió en una cosa exclusiva de *nerds*,[1] y en consecuencia adquirió un cierto tinte arcano incluso entre aquellos de nosotros que sabíamos siquiera que existía. Todos evitábamos ya los lotes, habiéndonos pasado a la línea de comandos: mi primer cambio de paradigma de sistema operativo, y yo sin enterarme.

Había una enorme pila de papel plegado en el suelo bajo cada uno de estos teletipos glorificados, y kilómetros de papel se estremecían mientras pasaban por sus rodillos. Casi todo este papel se tiraba o se reciclaba sin haber sido tocado jamás por la tinta, una atrocidad ecológica tan flagrante que aquellas máquinas pronto fueron reemplazadas por terminales de vídeo —los llamados «teletipos de vidrio», que eran más silenciosos y no desperdiciaban papel—. Sin embargo, desde el punto de vista del ordenador, estos también eran indistinguibles de las máquinas de teletipo de la Segunda Guerra Mundial. A todos los efectos, seguimos usando tecnología victoriana para comunicarnos con los ordenadores hasta cerca de 1984, cuando se introdujo el Macintosh con su Interfaz Gráfica de Usuario. Incluso después de eso, la línea de comandos siguió existiendo como estrato subyacente —una especie de reflejo medular— a muchos sistemas informáticos modernos durante la edad de oro de las Interfaces Gráficas de Usuario o GUI («Graphical User Interface»), como las llamaré de ahora en adelante.

---

[1] *Nerd:* El empollón de la clase, retratado tantas veces en las películas y las series de televisión norteamericanas, generalmente con dificultad para relacionarse socialmente y que en cambio suele destacar en materias tales como las matemáticas o la astronomía. En la jerga hacker se ha asumido de forma irónica («news for nerds» es el lema de slashdot, el foro web más importante dedicado a tecnología y software libre), perdiendo el matiz originalmente despectivo, y ha acabado usándose como sinónimo de alguien que se preocupa por las cosas importantes y no se entretiene en trivialidades. [*N. del E.*]

Las Interfaces Gráficas de Usuario

LO PRIMERO QUE TIENE QUE HACER cualquier programador al escribir un nuevo fragmento de software es decidir cómo tomar la información con que está trabajando (en un programa gráfico, una imagen; en una hoja de cálculo, una tabla de números) y convertirla en una serie lineal de bytes. Estas cadenas de bytes se suelen denominar archivos o (de modo algo más a la última) *flujos*. Son a los telegramas lo que los humanos actuales son al hombre de Cromañón, lo que quiere decir la misma cosa con distinto nombre. Todo lo que se ve en la pantalla del ordenador —Tomb Raider, los correos electrónicos de voz digitalizada, los faxes y los documentos de procesador de textos escritos en treinta y siete tipos diferentes— sigue siendo, desde el punto de vista del ordenador, igual que telegramas, sólo que son mucho más largos, y requieren más aritmética.

El modo más rápido de apreciarlo es abriendo el navegador, visitando un sitio web y seleccionando la opción «Ver Código Fuente» en el menú. Se mostrará un código informático parecido a este:

```
<HTML>
<HEAD>
  <TITLE>C R Y P T O N O M I C O N</TITLE>
</HEAD>
<BODY BGCOLOR="#000000" LINK="#996600" ALINK="#FFFFFF"
      VLINK="#663300">
```

```
<MAP NAME="navtext">
  <AREA SHAPE=RECT HREF="praise.html" COORDS="0,37,84,55">
  <AREA SHAPE=RECT HREF="author.html" COORDS="0,59,137,75">
  <AREA SHAPE=RECT HREF="text.html" COORDS="0,81,101,96">
  <AREA SHAPE=RECT HREF="tour.html" COORDS="0,100,121,117">
  <AREA SHAPE=RECT HREF="order.html"
      COORDS="0,122,143,138">
  <AREA SHAPE=RECT HREF="beginning.html"
      COORDS="0,140,213,157">
</MAP>

<CENTER>
<TABLE BORDER="0" CELLPADDING="0" CELLSPACING="0"
      WIDTH="520">
<TR>
<TD VALIGN=TOP ROWSPAN="5">
  <IMG SRC="images/spacer.gif" WIDTH="30" HEIGHT="1"
      BORDER="0">
</TD>
<TD VALIGN=TOP COLSPAN="2">
  <IMG SRC="images/main_banner.gif" ALT="Cryptonomicon by
      Neal Stephenson" WIDTH="479" HEIGHT="122" BORDER="0">
</TD>
</TR>
```

Esto se llama HTML (Lenguaje de Marcado de HiperTexto) y básicamente es un lenguaje de programación muy sencillo que le dice al navegador cómo dibujar una página en la pantalla. Cualquiera puede aprender HTML y mucha gente lo hace. Lo importante es que, por muchas espléndidas páginas multimedia que representen, los archivos de HTML son sólo telegramas.

Cuando Ronald Reagan era locutor de radio, solía informar de los partidos de béisbol leyendo las concisas descripciones que llegaban por el telégrafo y se imprimían en cinta de papel. Se sentaba solo en una habitación insonorizada con un micrófono y la cinta de papel salía de la máquina y le caía en la palma de la mano, cubierta de crípticas abreviaturas. Si el tanteo pasaba de tres a dos, Reagan describía la escena como se la imaginaba: «El fornido zurdo sale del puesto de bateo para secarse el sudor. El árbitro se adelanta para limpiar el polvo de la base», etc. Cuando el criptograma en la cinta de papel anunciaba un golpe en una base, Reagan golpeaba el borde de la mesa con un lápiz, creando un pequeño efecto sonoro y describía el arco de la pelota como si pudiera verlo de verdad. Sus oyentes, muchos de los cuales presumiblemente creían que Reagan estaba de hecho en el campo de juego viendo el partido, reconstruían la escena en su mente según sus descripciones.

Así es exactamente como funciona la WWW: los archivos HTML son la concisa descripción en la cinta de papel y el navegador es Ronald Reagan. Lo mismo vale para las interfaces gráficas en general.

De modo que un sistema operativo consiste en una pila de metáforas y abstracciones que media entre los telegramas y tú, encarnando diversos trucos que el programador usó para convertir la información con la que estás trabajando —ya sean imágenes, mensajes de correo electrónico, películas o documentos de procesador de textos— en las cadenas de bytes, que son lo único con lo que funcionan los ordenadores. Cuando usamos equipo telegráfico genuino (teletipos) o sus sustitutos de alta tecnología (teletipos de vidrio, o la línea de comandos de MS-DOS) para trabajar con nuestros ordenadores, estamos muy cerca de la base de esa pila. Cuando usamos la mayor parte de sistemas operativos modernos, sin embargo, nuestra interacción con la máquina se ve fuertemente mediada. Todo lo que hacemos es interpretado y traducido una y otra vez mientras se abre camino a través de todas las metáforas y abstracciones.

El sistema operativo de Macintosh fue una revolución en el buen y en el mal sentido de la palabra. Obviamente era cierto que las interfaces de línea de comandos (conocidas como CLI, *Command Line Interfaces*) no eran para todo el mundo, y que estaría bien hacer los ordenadores accesibles a un público menos técnico —si no por razones altruistas, siquiera porque este tipo de gente constituía un mercado incomparablemente mayor—. Está claro que los ingenieros de Mac vieron todo un país nuevo que se les abría; casi se les podía oír mascullar, «¡Caray! ¡Ya no tendremos que limitarnos más a los archivos como flujos lineales de bytes, *vive la révolution,* veamos lo lejos que llegamos con esto!» No había ninguna interfaz de línea de comandos disponible en el Macintosh; hablabas con la máquina a través del ratón, o no hablabas. Era una especie de declaración de principios, una credencial de pureza revolucionaria. Parecía que los diseñadores del Mac pretendían barrer las interfaces de línea de comandos a la papelera de la historia.

Mi propia historia de amor con el Macintosh comenzó en la primavera de 1984 en una tienda de ordenadores en Cedar Rapids, Iowa, cuando un amigo mío —por coincidencia, el hijo del dueño del descapotable— me mostró un Macintosh ejecutando MacPaint, el revolucionario programa de diseño. Terminó en julio de 1995, cuando traté de guardar un archivo grande e importante en mi Macintosh PowerBook y, en vez de eso, destruyó los datos de modo tan concienzudo que dos programas distintos de recuperación de datos fueron incapaces de hallar rastro alguno de que hubiera existido jamás. En aquellos diez años sentí una pasión por el MacOS que por entonces parecía virtuosa y razonable, pero que mirando atrás me parece el mismo tipo de enamoramiento engañoso que el padre de mi amigo tenía con su coche.

La introducción del Mac inició una especie de guerra santa en el mundo de la informática. ¿Eran las interfaces gráficas una brillante innovación tecnológica que convertía a los ordenadores en más accesibles para los humanos y por tanto para las masas, llevándonos a una revolución sin precedentes en la sociedad humana, o una insultante chorrada audiovisual diseñada por hackers zumbados de San Francisco, que despojaba a los ordenadores de su potencia y flexibilidad y convertía el serio y noble arte de la computación en un pueril videojuego?

De hecho, este debate me parece más interesante hoy en día que a mediados de los ochenta. Pero la gente más o menos dejó de debatir cuando Microsoft respaldó la idea de las interfaces gráficas al sacar el primer Windows. En aquel momento, los partidarios de la línea de comandos se vieron relegados al estatus de viejos carcamales, mientras se disparaba un nuevo conflicto entre usuarios de MacOS y de Windows.[1]

[1] De acuerdo con una rigurosa y algo anticuada definición de «sistema operativo», Windows 95 y 98 no lo son: serían un conjunto de operaciones que funcionan sobre MS-DOS, que sí es un sistema operativo. En la prác-

Había mucho sobre lo que discutir. Los primeros Macintosh parecían distintos de otros PC incluso estando apagados: consistían en una caja que contenía tanto la CPU (la parte del ordenador que hace aritmética con los bits) como la pantalla del monitor. Esto suponía, en aquel momento, una especie de afirmación filosófica: Apple quería convertir el ordenador personal en un electrodoméstico, como la tostadora. Pero también reflejaba las exigencias puramente técnicas de ejecutar una inferfaz gráfica de usuario. En una máquina con interfaz gráfica, los chips que dibujan las cosas en la pantalla tienen que ir integrados con la unidad de procesamiento central, o CPU, del ordenador, en un grado mucho mayor que en las interfaces de línea de comandos, que hasta hace poco ni siquiera sabían que no estaban hablando sólo con teletipos.

Esta distinción era de naturaleza técnica y abstracta, pero se hacía más clara cuando la máquina fallaba (como sucede frecuentemente con tecnologías cuyo funcionamiento se comprende mejor viéndolas fallar). Cuando todo se iba a la porra y la CPU empezaba a escupir bits aleatoriamente, el resultado, en una máquina de interfaz de línea de comandos, era líneas y líneas de caracteres perfectamente formados pero aleatorios en la pantalla —lo que los conocedores llamaban *ponerse cirílico*. Pero para el MacOS la pantalla no era un teletipo sino un lugar en el que poner gráficos; la imagen en pantalla era un mapa de bits, una representación literal de los contenidos de una parte dada de la memoria del ordenador. Cuando el ordenador fallaba y escribía tonterías en el mapa de bits, el resultado era algo que recordaba vagamente a la nieve en una televisión estropeada: un *snow crash*.[2]

tica, Windows 95 y 98 están comercializados como sistemas operativos, y trataré de referirme a ellos como tales. Esta nomenclatura es técnicamente cuestionable, políticamente difícil y ahora también legalmente gravosa, pero es la mejor para los propósitos de este ensayo, que trata principalmente aspectos estéticos y culturales.

[2]Literalmente: «cuelgue de nieve». *Snow Crash* es también el título de una novela de N. Stephenson, auténtica obra de culto entre los hackers, publicada en 1994, y editada en español por Gigamesh en 1999. [*N. del E.*]

E incluso, tras la introducción de Windows, las diferencias subyacentes persistieron: cuando una máquina Windows tenía problemas, la vieja interfaz de línea de comandos caía sobre la interfaz gráfica como un telón de amianto, sellando el escenario de una ópera incendiada. Cuando un Macintosh tenía problemas, te presentaba el dibujito de una bomba, que resultaba gracioso la primera vez que lo veías.

Y estas no eran en absoluto diferencias superficiales. El retorno de Windows a una interfaz de línea de comandos cuando tenía problemas les demostraba a los partidarios del Mac que Windows no era más que una fachada barata, como una chillona manta afgana tendida sobre un sofá putrefacto. Les perturbaba y molestaba la sensación de que bajo la ostensiblemente amistosa interfaz de usuario de Windows había —literalmente— un subtexto.

Por su parte, los fans de Windows podrían haber observado agriamente que todos los ordenadores, incluso los Macintosh, estaban construidos sobre ese mismo subtexto, y que la negativa de los dueños de Macs a admitir ese hecho parecía apuntar a una voluntad, incluso un deseo, de dejarse engañar.

En cualquier caso, un Macintosh tenía que mover bits individuales en los chips de memoria en la tarjeta de vídeo, y tenía que hacerlo muy rápido, y en patrones arbitrariamente complicados. Hoy en día esto resulta barato y fácil, pero en el régimen tecnológico vigente a principios de los ochenta, el único modo realista de hacerlo era integrar la placa base (que contenía la CPU) y el sistema de vídeo (que contenía la memoria proyectada sobre la pantalla) como un todo —de ahí el único contenedor, herméticamente sellado, que hacía al Macintosh tan distintivo.

Cuando apareció Windows llamaba la atención por su fealdad, y sus actuales sucesores, Windows 95 y Windows NT, no son cosas que la gente pagaría por ver. La absoluta falta de atención de Microsoft por la estética nos proporcionaba muchas oportunidades a todos los amantes de Mac para mirarles por encima del hombro. El que Windows se pareciera un montón a un calco directo de MacOS nos daba además una fuerte sensación de ultraje moral.[3] Entre las personas que realmente conocían y apreciaban los ordenadores (los *hackers,* en el sentido no peyorativo que Steven Levy le da a la palabra[4] y unos pocos otros ámbitos como los músicos profesionales, los artistas gráficos y los maestros), el Macintosh, durante un tiempo, era simplemente *el ordenador.* No sólo se consideraba una obra soberbia de ingeniería, sino la encarnación de ciertos ideales acerca del uso de la tecnología para beneficiar a la humanidad, mientras que Windows se consideraba una imitación patéticamente torpe y una siniestra combinación para dominar el mundo, todo en uno. Ya entonces se había establecido un patrón que persiste hasta nuestros días: a la gente no le gusta Microsoft, lo cual es comprensible; pero no les gusta por razones poco reflexionadas y, en último término, contradictorias.

[3]De hecho, Apple demandó a Microsoft por plagiarle la interfaz gráfica, juicio que perdió. Al parecer, Apple olvidó demasiado rápido que ellos mismos habían copiado diez años antes dicha interfaz a Xerox. [*N. del E.*]

[4]Se refiere a la ya obra clásica *Hackers* (1984), en la que S. Levy expuso una serie de principios que habían guiado a la ética hacker desde los años sesenta. Levy los resumió así: «El acceso a los ordenadores y a todo lo que te pueda enseñar algo sobre cómo funciona el mundo debe ser ilimitado. Toda la información debe ser libre. Desconfía de la autoridad, promueve la descentralización; los hackers deberían ser juzgados por su habilidad, no por su edad, nivel, raza o posición. Puedes crear arte y belleza con tu ordenador. Los ordenadores pueden cambiar tu vida a mejor.» [*N. del E.*]

Lucha de clases en el escritorio

AHORA QUE YA HEMOS DEJADO CLARO el trasfondo, merece la pena revisar algunos hechos básicos: como cualquier compañía de accionariado público y con fines de lucro, Microsoft ha tomado prestado un montón de dinero de algunas personas (sus accionistas) para estar en el negocio del bit. Como ejecutivo de esa compañía, Bill Gates sólo tiene una responsabilidad, que es maximizar el rendimiento de las inversiones. Lo ha hecho increíblemente bien. Cualquier acción emprendida en el mundo por Microsoft —cualquier software que publiquen, por ejemplo— es básicamente un epifenómeno que no puede comprenderse ni entenderse salvo en la medida en que reflejan el desempeño por parte de Bill Gates de su única responsabilidad.

De ello se sigue que si Microsoft vende mercancías que son estéticamente desagradables, o que no funcionan demasiado bien, no significa que sean (respectivamente) filisteos o medio tontos. Se debe a que la excelente dirección de Microsoft ha llegado a la conclusión de que pueden ganar más dinero para sus accionistas publicando productos con imperfecciones obvias y conocidas del que ganarían haciéndolos hermosos o libres de errores. Esto es irritante, pero (al final) no tan irritante como contemplar cómo Apple se autodestruye inexplicable e implacablemente.

No resulta difícil encontrar en la Red una hostilidad hacia Microsoft que mezcla dos elementos: resentidos que sienten que Microsoft es demasiado poderosa y desdeñosos que creen que es chapucera. Esto recuerda mucho al periodo culminante del comunismo y del socialismo, cuando se odiaba a la burguesía desde ambos lados: los proletarios, porque la burguesía tenía todo el dinero, y los intelectuales, por su tendencia a gastárselo en adornos de jardín. Microsoft es la encarnación misma de la moderna prosperidad de alta tecnología —en una palabra, es burguesa— y atrae los mismos odios de todos.

La pantalla inicial de Microsoft Word 6.0 lo resumía todo bastante bien: cuando arrancabas el programa, te mostraba la imagen de un bolígrafo caro encima de un par de folios de papel de escritura hecho a mano. Obviamente, era un intento por hacer que el software pareciera pijo, y puede que valiera para algunos, pero no para mí, porque era un bolígrafo, y yo soy hombre de pluma estilográfica. Si lo hubiera hecho Apple, habrían usado una pluma Mont Blanc, o quizás un pincel caligráfico chino. Dudo que esto fuera accidental. Hace poco estuve reinstalando Windows NT en uno de los ordenadores de mi casa, y tuve que hacer doble click en el icono del Panel de Control muchas veces. Por razones que resulta difícil comprender, este icono consiste en el dibujito de un martillo y un escoplo o un destornillador encima de una carpeta de archivos.

Estas meteduras de pata estéticas le dan a uno unas ganas casi incontrolables de reírse de Microsoft, pero, de nuevo, esa no es la cuestión: si Microsoft hubiese hecho pruebas con grupos-diana sobre posibles gráficos alternativos, probablemente habrían hallado que el oficinista medio asociaba las estilográficas con los amanerados ejecutivos de rango más alto, y estaba más cómodo con los bolígrafos. De igual forma, los tipos normales, los papás con entradas del mundo que posiblemente cargan con la responsabilidad de montar y configurar el ordenador en casa, probablemente prefieren el dibujito de un martillo (quizás al tiempo que albergan fantasías de usar un martillo de verdad con sus ordenadores).

Es el único modo en que consigo explicar cierto hechos curiosos acerca del actual mercado de sistemas operativos, tales como el que el noventa por ciento de todos los clientes sigan comprando monovolúmenes de la tienda de Microsoft mientras que uno se puede llevar los tanques gratuitos sin más, al otro lado de la calle.

A Bill Gates no le resultó difícil distribuir una sarta de unos y ceros, una vez se le ocurrió la idea. Lo duro era venderla: asegurarles a los clientes que de hecho estaban obteniendo algo a cambio de su dinero.

Cualquiera que haya comprado software en una tienda alguna vez habrá tenido curiosamente la desalentadora experiencia de llevarse la caja envuelta en plástico a casa, abrirla, encontrarse con que el 95 % es aire, tirar todas las tarjetitas, propaganda y basura y meter el disco en el ordenador. El resultado final (después de haber perdido el disco) no es nada más que algunas imágenes en la pantalla del ordenador y algunas posibilidades de las que antes se carecía. A veces, ni siquiera eso —en vez de ello, uno se encuentra con una serie de mensajes de error—. Pero el dinero se ha ido definitivamente. Ahora casi estamos acostumbrados a esto pero hace veinte años era una proposición muy sospechosa. De todas formas, Bill Gates consiguió que funcionara. No hizo que funcionara vendiendo el mejor software ni ofreciendo el precio más barato. Pero de algún modo consiguió que la gente creyera que estaban recibiendo algo a cambio de su dinero.

Las calles de todas las ciudades del mundo están llenas de esos pesados, ruidosos monovolúmenes. Cualquiera que no tenga uno se siente un poco raro, y se pregunta, pese a sí mismo, si no será hora de dejar de resistirse y comprar uno; cualquiera que tenga uno se siente seguro de que ha adquirido una posesión significativa, incluso los días en que el vehículo está en el taller de reparación.

Todo esto es perfectamente congruente con la pertenencia a la burguesía, que es un estado tanto mental como material. Y explica por qué Microsoft se ve constantemente atacado en la Red desde ambos lados. Los que se siente pobres y oprimidos interpretan todo lo que hace Microsoft como parte de algún siniestro complot orwelliano. A los que les gusta considerarse usuarios inteligentes e informados, les desquicia lo chapucero que es Windows.

No hay nada que moleste más a las personas sofisticadas que ver cómo alguien que es lo bastante rico como para evitarlo es hortera —a menos que se den cuenta, un momento después, de que probablemente sabe que es hortera y sencillamente no le importa y va a seguir siendo hortera, y rico, y feliz, para siempre; Microsoft tiene la misma relación con la elite de Silicon Valley que la que mantenían los paletos de Beverly con su banquero, el señor Drysdale— a quien no le irrita tanto el hecho de que los Clampetts se mudaran a su barrio como el saber que, cuando Jethro tenga setenta años, seguirá hablando como un palurdo y llevando petos, y seguirá siendo mucho más rico que el señor Drysdale.

Incluso el hardware que empleaba Windows, comparado con las máquinas que sacaba Apple, parecía cosa de palurdos, y en su mayor parte sigue pareciéndolo. La razón es que Apple era y es una compañía de hardware, mientras que Microsoft era y es una compañía de software. Apple tenía así el monopolio del hardware que ejecutaba MacOS, mientras que el hardware compatible con Windows venía del mercado libre. El mercado libre parece haber decidido que la gente no va a pagar por ordenadores elegantes; los fabricantes de hardware para PC que contratan a diseñadores para hacer que sus productos tengan un aire distintivo acaban vapuleados por fabricantes taiwaneses de clones metidos en cajas que parecen los ladrillos que uno se encontraría delante de una caravana. Pero Apple podía hacer su software todo lo bonito que quisiera y simplemente pasarle la factura a sus encantados consumidores, como yo. La semana pasada (escribo esta frase a principios de enero de 1999), las secciones de tecnología de todos los periódicos estaban llenas de reportajes aduladores sobre el lanzamiento por parte de Apple del iMac en varios colores nuevos, como arándano y mandarina.

Apple siempre ha insistido en tener el monopolio de su hardware, salvo durante un breve periodo a mediados de los noventa, cuando permitieron que los fabricantes de clones compitieran con ella, antes de acabar con su negocio. El hardware de Macintosh, en consecuencia, era caro. No lo abrías ni enredabas en él porque hacerlo anulaba la garantía. De hecho, el primer Mac estaba específicamente diseñado para resultar difícil de abrir: necesitabas un juego de herramientas exóticas, que podías comprar mediante pequeños anuncios que empezaron a aparecer en las páginas finales de las revistas unos pocos meses después de que saliera al mercado el Mac. Estos anuncios siempre tenían un cierto aire sórdido, como si anunciaran ganzúas en la contraportada de sensacionalistas revistas de detectives.

Esta política de monopolio puede explicarse al menos de tres maneras distintas.

La explicación caritativa es que la política de monopolio sobre el hardware reflejaba el deseo por parte de Apple de proporcionar una unión sin fallas de hardware, sistema operativo y software. Algo hay de esto. Ya resulta bastante difícil diseñar un sistema operativo que funcione bien en un hardware específico, diseñado y probado por ingenieros que trabajan al lado, en la misma compañía. Diseñar un sistema operativo que funcione en un hardware cualquiera, fabricado por hacedores de clones rabiosamente competitivos al otro lado de la Línea de Fecha Internacional, es muy difícil, y explica gran parte de los problemas que tiene la gente cuando usa Windows.

La explicación financiera es que Apple, a diferencia de Microsoft, es y siempre ha sido una compañía de hardware. Sencillamente depende de los ingresos de la venta de hardware, y no puede subsistir sin ellos.

La explicación no tan caritativa tiene que ver con la cultura corporativa de Apple, que tiene sus raíces en el *baby boom* del Área de la Bahía de San Francisco.

Dado que voy a hablar sobre cultura durante un rato, probablemente está bien que ponga las cartas sobre la mesa, para protegerme de las acusaciones de conflicto de intereses y falta de ética: 1) Geográficamente, soy de Seattle, de temperamento saturnino e inclinado a mirar con malos ojos la dionisíaca Área de la Bahía de San Francisco, igual que a ellos nosotros les molestamos y escandalizamos. 2) Cronológicamente, pertenezco a una generación posterior al *baby boom*. Al menos, así me siento, ya que nunca experimenté las partes divertidas y emocionantes del *baby boom* —sólo me pasé un montón de tiempo riéndome apropiadamente ante las irritantemente vacuas anécdotas de los pertenecientes al *baby boom* sobre lo puestos que iban en diversas ocasiones, y escuchando cortés sus aseveraciones de lo estupenda que era su música. Pero, incluso desde aquella distancia, resultaba posible extraer ciertos patrones, y uno que reaparecía tan regularmente como una leyenda urbana era el de alguien que se había mudado a una comuna de hippies con sandalias y signos de la paz para acabar descubriendo que, bajo aquella fachada, los tipos al mando eran de hecho obsesos del control; y que, dado que vivir en una comuna donde los ideales de la paz, el amor y la armonía se mantenían de boquilla les había privado de válvulas de escape normales y socialmente admitidas para su obsesión, tendía a salir de de otros modos, invariablemente más siniestros.

Dejaré aplicar esto al caso de Apple como ejercicio para el lector —un ejercicio no demasiado difícil.

Resulta un poco desconcertante, al principio, pensar en Apple como un obseso del control, porque contradice completamente su imagen corporativa. ¿No fueron estos los tipos que lanzaron los famosos anuncios durante la Super Bowl en los que ejecutivos trajeados, con los ojos vendados, saltaban como lemmings de un acantilado? ¿No es esta la compañía que ahora mismo saca anuncios con el Dalai Lama (salvo en Hong Kong) y Einstein y otros rebeldes alternativos?

Ciertamente es la misma compañía, y el hecho de que hayan implantado esta imagen de sí mismos como librepensadores creativos y rebeldes en la mente de tantos escépticos inteligentes y encallecidos por los medios, realmente hace que uno se pare a pensar. Da fe del insidioso poder de las campañas publicitarias costosas y tal vez, en cierta medida, de la facilidad de la gente para creer lo que quiere creer. También suscita la pregunta de por qué a Microsoft se le da tan mal las relaciones públicas, cuando la historia de Apple demuestra que, pasándoles gordos cheques a buenas agencias publicitarias, se puede implantar una imagen corporativa en la mente de personas inteligentes que difiere completamente de la realidad. (La respuesta, para aquéllos a los que no les gustan las espadas de Damocles, es que, ya que Microsoft se ha hecho con las mentes y los corazones de la silenciosa mayoría —la burguesía—, les importa un bledo tener una imagen elegante, igual que Richard Nixon. «Quiero creer» —el mantra que Fox Mulder tiene puesto en la pared de su despacho en los *Expedientes X*— resulta aplicable de diferentes modos a estas dos compañías; los partidarios del Mac quieren creer en la imagen de Apple que transmiten estos anuncios, y en la noción de que los Macs son de algún modo fundamentalmente diferentes de otros ordenadores, mientras que los seguidores de Windows quieren creer que obtienen algo a cambio de su dinero, mediante una respetable transacción comercial).

En cualquier caso, en 1987 tanto MacOS como Windows ya estaban en el mercado, ejecutándose en plataformas de hardware que eran radicalmente diferentes entre sí, no sólo en el sentido de que MacOS usaba chips de CPU de Motorola, mientras que Windows usaba Intel, sino también en el sentido —entonces pasado por alto, pero a largo plazo mucho más significativo— de que el negocio de hardware de Apple era un monopolio rígido y Windows era un abierto-a-todos.

Pero todas las ramificaciones de esto no estuvieron claras hasta muy recientemente —de hecho, aún están desplegándose, de modos notablemente extraños, como explicaré cuando lleguemos a Linux—. El resultado es que millones de personas se acostumbraron a usar interfaces gráficas de una forma u otra. Con ello hicieron que Apple/Microsoft ganaran un montón de dinero. La fortuna de muchas personas ha acabado por ir ligada a la capacidad de estas compañías de seguir vendiendo productos cuyo carácter vendible resulta muy cuestionable.

Tarro de miel, pozo de brea, lo que sea

CUANDO GATES Y ALLEN INVENTARON la idea de vender software, se encontraron con la crítica tanto de los hackers como de los sobrios hombres de negocios. Los hackers entendían que el software sólo era información, y le ponían objeciones a la idea de venderla. Estas objeciones eran en parte morales. Los hackers salían del mundo científico y académico, donde resulta imperativo hacer que los resultados del propio trabajo queden disponibles para el público. También eran en parte objeciones prácticas: ¿cómo puedes vender algo que puede copiarse fácilmente? Los hombres de negocios, que son el polo opuesto de los hackers en tantos aspectos, tenían sus propias objeciones. Acostumbrados a vender tostadoras y seguros, era natural que les resultara difícil comprender cómo una larga sarta de unos y ceros podía constituir un producto vendible.

Obviamente, Microsoft remontó estas objeciones, así como Apple. Pero las objeciones siguen ahí. El hacker más hacker de todos, el Ur-hacker por así decirlo, era y es Richard Stallman, quien se irritó tanto con la malvada práctica de vender software que, en 1984 (el mismo año en que salió a la venta el Macintosh), fue y fundó la Fundación del Software Libre (Free Software Foundation), que comenzó a trabajar en algo llamando GNU. GNU son las siglas de Gnu's Not Unix («Gnu No es Unix»), pero se trata de una broma en más de un sentido, porque GNU ciertamente *es* Unix. Debido a cuestiones de copyright (Unix es una marca registrada de AT&T), sencillamente no podían afirmar que fuera Unix, y así, sólo para asegurarse, afirmaban que no lo era. Pese al incomparable talento y empuje del señor Stallman y otros seguidores de GNU, su proyecto no pudo construir un Unix libre para competir contra los sistemas operativos de Windows y Apple: era un poco como tratar de excavar un sistema de metro con una cucharilla. Esto es, hasta la llegada de Linux.[1]

Pero la idea básica de recrear un sistema operativo a partir de la nada era perfectamente consistente y completamente factible. Se ha hecho muchas veces. Es inherente a la naturaleza misma de los sistemas operativos.

Los sistemas operativos no son estrictamente necesarios. No hay razón por la que un escritor de código lo bastante dedicado no pueda partir de la nada en cada proyecto y escribir nuevo código para manejar operaciones tan básicas y de bajo nivel como controlar las cabezas lectoras/escritoras en los controladores de disco y activar píxeles en pantalla. Los primeros ordenadores tenían que programarse de este modo. Pero, dado que casi todos los programas tienen que desempeñar las mismas operaciones básicas, este enfoque llevaría a una tremenda duplicación del esfuerzo.

No hay nada más desagradable para el hacker que la duplicación del esfuerzo. El primer y más importante hábito mental que desarrolla la gente cuando aprende a escribir programas de ordenador es generalizar, generalizar, generalizar. Hacer su código lo más modular y flexible posible, descomponer los problemas grandes en pequeñas subrutinas que puedan usarse una y otra vez en diferentes contextos. En consecuencia, el desarrollo de los sistemas ope-

[1] Stallman insiste en que este sistema operativo debería ser siempre nombrado como GNU/Linux y tiene perfectas razones para hablar así, por ejemplo, para que el papel del proyecto GNU no sea ignorado. En la práctica, casi todo el mundo se refiere a este como Linux. Para los propósitos de este ensayo, enfatizo el papel de GNU describiéndolo explícitamente, más que usando la nomenclatura GNU/Linux.

rativos, pese a ser técnicamente innecesario, era inevitable. Porque en el fondo un sistema operativo no es más que una biblioteca que contiene el código más usado, escrito una vez (y con suerte, bien escrito), y puesto a disposición de cualquier escritor de código que lo necesite.

Así que un sistema operativo propietario, cerrado y secreto es una contradicción en los términos. Va contra la razón de ser de los sistemas operativos. Y de cualquier modo es imposible mantenerlos en secreto. El código fuente —las líneas originales de texto escritas por los programadores— pueden mantenerse en secreto. Pero el conjunto de un sistema operativo es una colección de pequeñas subrutinas que realizan tareas muy específicas y muy claramente definidas. Qué hacen exactamente esas subrutinas ha de ser público, de forma muy explícita y exacta, o de lo contrario el sistema operativo es completamente inservible para los programadores; no pueden usar esas subrutinas si no tienen perfecta y total comprensión de lo que hacen las subrutinas.

Lo único que no se hace público es exactamente cómo hacen las subrutinas lo que hacen. Pero una vez sabes lo que hace una subrutina, generalmente resulta bastante fácil (si eres un hacker) escribir tu propia rutina que haga exactamente lo mismo. Puedes tardar algo, y resulta tedioso y poco gratificante, pero en la mayoría de los casos no es demasiado difícil.

Lo que es difícil, para un hacker como para un escritor de ficción, no es escribir; es decidir qué escribir. Y los vendedores de sistemas operativos comerciales ya han decidido, y han hecho públicas sus decisiones.

Esto se sabe desde hace mucho. MS-DOS fue duplicado funcionalmente por un producto rival, escrito a partir de la nada, llamado ProDOS, que hacía las mismas cosas de modo muy parecido. En otras palabras, otra compañía pudo escribir código que hacía las mismas cosas que MS-DOS y lo vendió para obtener beneficios. Si usas el sistema operativo Linux, puedes obtener un programa libre llamando WINE que es un emulador de Windows; esto es, puedes abrir una ventana en tu escritorio que ejecute programas de Windows. Quiere decir que se ha recreado un sistema operativo Windows completamente funcional dentro de Unix, como un barquito en una botella. Y el propio Unix, que es un sistema operativo mucho más sofisticado que MS-DOS, ha sido reconstruido a partir de la nada una y otra vez. Sun, Hewlett-Packard, AT&T, Silicon Graphics, IBM y otros vendieron versiones de él.

En otras palabras, la gente lleva reescribiendo código básico de sistemas operativos tanto tiempo que toda la tecnología que constituía un sistema operativo en el sentido tradicional (pre-GUI) de esa expresión es ahora tan barata y común que es literalmente gratuita. No sólo no podrían Gates y Allen vender MS-DOS hoy, ni siquiera podrían regalarlo, porque ya se regalan sistemas operativos mucho más potentes. Incluso el Windows original (que era el único sistema de ventanas hasta 1995) ya no vale nada, dado que no tiene sentido poseer algo que puede emularse dentro de Linux, que es gratuito.[2]

De este modo, el negocio de los sistemas operativos es muy diferente de, pongamos, el negocio de la venta de coches. Incluso un viejo coche de segunda mano tiene algún valor. Puedes usarlo para ir al basurero, o vender sus partes. El destino de los bienes manufacturados es depreciarse lentamente a medida que envejecen y tienen que competir contra productos más modernos.

Pero el destino de los sistemas operativos es volverse gratuitos.

Microsoft es una gran compañía de aplicaciones de software. El de las aplicaciones —tales como Microsoft Word— es un área en el que la innovación lleva beneficios reales, directos y tangibles a los usuarios. Las innovaciones pueden consistir en nueva tecnología recién salida del departamento de investigación, o pueden estar en la categoría de

[2]El autor usa muchas veces a lo largo del texto *free* en un sentido inequívoco que indica gratuidad, como en este caso, y por supuesto hemos respetado dicho sentido en la traducción, a pesar de que el *free software* —incluido por supuesto GNU/Linux— es libre en el sentido de libertad, no de precio. Nos hemos extendido sobre esta cuestión en la «Presentación» de esta edición. [*N. del E.*]

los lacitos decorativos, pero en cualquier caso a menudo resultan útiles y parecen contentar a los usuarios. Y Microsoft está convirtiéndose en una gran compañía de investigación. Esto no se debe necesariamente a que sus sistemas operativos sean todos tan malos desde el punto de vista puramente tecnológico. Los sistemas operativos de Microsoft tienen sus problemas, claro, pero son mucho mejores de lo que solían ser, y son adecuados para la mayor parte de la gente.

¿Por qué digo entonces que Microsoft no es es una compañía de sistemas operativos tan grandes? Porque la naturaleza misma de los sistemas operativos es tal que no tiene sentido que una compañía específica los desarrolle y posea. Para empezar, es un trabajo muy desagradecido. Las aplicaciones crean posibilidades para millones de usuarios crédulos, mientras que los sistemas operativos imponen limitaciones a millones de cascarrabias escritores de código, y así los hacedores de sistemas operativos siempre estarán en la lista negra de cualquiera que cuente en el mundo de la alta tecnología. Las aplicaciones las usan personas cuyo gran problema es comprender todas sus características, mientras que los sistemas operativos se ven hackeados por escritores de código irritados con sus limitaciones. El negocio de los sistemas operativos ha sido bueno para Microsoft sólo en la medida en que les ha proporcionado el dinero necesario para lanzar un negocio de software de aplicaciones realmente bueno y contratar a un montón de investigadores inteligentes. Ahora debiera estar en posición de desembarazarse de su sistema operativo, como los cohetes se libran en algún momento de los tanques vacíos de combustible. La gran pregunta es si Microsoft es capaz de hacerlo. ¿O es adicta a la venta de sistemas operativos del mismo modo que Apple lo es a la venta de hardware?

Hay que tener en cuenta que los observadores expertos citaban en un tiempo la capacidad de Apple de monopolizar su propia provisión de hardware como su gran ventaja frente a Microsoft. En aquella época, parecía situarles en una posición mucho más fuerte. Al final, casi les mató, y todavía puede matarlos. El problema para Apple era que la mayor parte de los usuarios de ordenador del mundo acaba comprando hardware más barato. Pero un hardware barato no podía ejecutar MacOS, y esa gente se pasó a Windows.

Sustituyan hardware por sistemas operativos, y Apple por Microsoft y verán cómo lo mismo está a punto de suceder de nuevo. Microsoft domina el mercado de sistemas operativos, lo cual les reporta ingresos y parece una gran idea de momento. Pero hay sistemas operativos mejores y más baratos, y están haciéndose cada vez más populares en partes del mundo que no están tan saturadas de ordenadores como los EE.UU. Dentro de diez años, puede que la mayoría de los usuarios de ordenador del mundo acabe por tener estos sistemas operativos más baratos. Pero estos sistemas operativos, de momento, no ejecutan ninguna aplicación de Windows, y así esta gente acabará usando otra cosa.

Por expresarlo de forma más directa: cada vez que alguien decide usar un sistema operativo que no es de Microsoft, la división de sistemas operativos de Microsoft obviamente pierde un cliente. Pero, tal como están las cosas, la división de aplicaciones de Microsoft también pierde un cliente. No es para tanto, dado que casi todo el mundo usa sistemas operativos de Microsoft. Pero en cuanto la cuota de mercado de Windows empiece a disminuir, las matemáticas van a ponerse bastante torvas para los de Redmond.

Podría replicarse a este argumento diciendo que Microsoft sencillamente podría recompilar sus aplicaciones para que pudieran ejecutarse en otros sistemas operativos. Pero esta estrategia va contra los instintos corporativos normales. El caso de Apple resulta de nuevo instructivo. Cuando las cosas empezaron a ponerse feas para Apple, debieron haber llevado su sistema operativo a un hardware barato. Pero no lo hicieron. Por el contrario, trataron de hacer que su brillante hardware diera lo más posible de sí, añadiendo nuevas posibilidades y expandiendo la línea de productos. Pero esto sólo tuvo el efecto de hacer su sistema operativo más dependiente de esas características especiales del hardware, lo cual al final resulta peor para ellos.

Igualmente, cuando la posición de Microsoft en el mundo de los sistemas operativos se vea amenazada, sus instintos corporativos les dirán que apilen más posibilidades en sus sistemas operativos, y luego reconfiguren sus aplicaciones de software para explotar esas posibilidades especiales. Pero esto sólo tendrá el efecto de hacer que sus aplicaciones dependan de un sistema operativo con una cuota de mercado decreciente, y al final será peor para ellos.

El mercado de los sistemas operativos es una trampa letal, un pozo de brea, una ciénaga. Sólo hay dos motivos para invertir en Apple y en Microsoft. 1) Cada una de estas compañías está en lo que llamaríamos una relación de codependencia con sus clientes. Los clientes *quieren creer,* y Apple y Microsoft saben cómo darles lo que quieren. 2) Cada compañía trabaja muy duro para añadir nuevas posibilidades a sus sistemas operativos, lo cual tiene el efecto de asegurar la lealtad de sus clientes, al menos durante un tiempo.

En consecuencia, la mayor parte del resto de este ensayo tratará sobre estos dos temas.

La tecnosfera

UNIX ES EL ÚNICO SISTEMA OPERATIVO que queda cuya interfaz gráfica (un montón de código llamado X Window System[1]) está separado del sistema operativo en el antiguo sentido del término. Es decir, que puedes ejecutar Unix en puro modo de línea de comandos si quieres, sin ventanas, iconos, ratones, etc., y seguirá siendo Unix y capaz de hacer todo lo que se supone que hace Unix. Pero los demás sistemas operativos —MacOS, la familia Windows y BeOS— tienen sus GUI enmarañadas con las anticuadas funciones del sistema operativo en tal grado que han de ejecutarse en modo GUI o no se ejecutan verdaderamente. Así que ya no es posible pensar en las GUI como en algo distinto del sistema operativo; ahora forman una parte inalienable de los sistemas operativos a los que pertenecen —y son, con mucho, la parte mayor, más cara y difícil de crear.

Sólo hay dos modos de vender un producto: precio y funcionalidades. Cuando los sistemas operativos son gratuitos, las compañías de sistemas operativos no pueden competir mediante el precio, así que compiten mediante las

[1]A pesar de la semejanza con el nombre del producto estrella de Microsoft, el Sistema X Window de los Unices no tiene nada que ver con Windows, sino que se trata de un potente sistema de ventanas basado en una arquitectura cliente/servidor. Una de las ventajas de la arquitectura cliente/servidor es que puede ser implementada tanto de manera distribuida (es decir, aplicaciones y servidor gráfico ejecutándose en máquinas diferentes) como local (todo el subsistema gráfico ejecutándose en el mismo ordenador). [*N. del E.*]

funcionalidades. Esto significa que siempre tratan de superarse unos a otros escribiendo código que, hasta hace poco, no se consideraba parte de un sistema operativo en absoluto: cosas como las GUI. Esto explica en gran medida el comportamiento de estas compañías.

Explica por qué Microsoft añadió un navegador a su sistema operativo, por ejemplo. Resulta fácil obtener navegadores gratuitos, igual que sistemas operativos gratuitos. Si los navegadores son gratuitos y los sistemas operativos son gratuitos, pareciera que no hay modo de hacer dinero con los navegadores ni con los sistemas operativos. Pero si puedes integrar un navegador en un sistema operativo y así llenar ambos de nuevas funcionalidades, ya tienes un producto vendible.

Dejando a un lado, de momento, el hecho de que esto cabrea de verdad a los abogados antimonopolio del gobierno, esta estrategia tiene sentido. Al menos, tiene sentido si se asume (como parece hacer la dirección de Microsoft) que el sistema operativo ha de ser protegido a cualquier precio. La verdadera cuestión es si cada moda tecnológica nueva que aparezca ha de usarse como muleta para sostener la posición dominante del sistema operativo. Al enfrentarse al fenómeno de la Web, Microsoft tuvo que desarrollar un navegador web realmente bueno, y lo hicieron. Pero entonces tuvieron que elegir: podían hacer que ese navegador funcionara en múltiples sistemas operativos, lo cual daría a Microsoft una posición fuerte en el mundo de Internet con independencia de lo que le pasara a la cuota de mercado de su sistema operativo. O podían integrar el navegador con el sistema operativo, apostando a que esto haría que su sistema operativo pareciera tan moderno y atractivo que ayudaría a conservar su dominio en ese mercado. El problema es que cuando la posición del sistema operativo Windows empiece a venirse abajo (y dado que actualmente es de cerca del noventa por ciento, no puede sino descender) arrastrará todo tras de sí.

En la clase de geología del instituto probablemente les enseñaran que toda la vida sobre la Tierra existe en una delgada capa llamada biosfera, que existe entre miles de kilómetros de roca muerta por debajo, y frío espacio vacío, muerto y radiactivo, por encima. Las compañías que venden sistemas operativos existen en una especie de tecnosfera. Por debajo está la tecnología que ya es gratuita. Por encima está la tecnología que todavía ha de ser desarrollada, o que es demasiado disparatada y especulativa para ser explotada de momento. Como la biosfera de la Tierra, la tecnosfera es muy fina comparada con lo que tiene por encima y por debajo.

Pero se mueve mucho más rápido. En diversas partes del mundo, es posible visitar ricas capas fósiles en las que hay esqueletos apilados, los más recientes encima y los más antiguos debajo. En teoría, todos se remontan a los primeros organismos unicelulares. Y si se usa la imaginación un poco, uno se dará cuenta de que, si se queda ahí el tiempo suficiente, también quedará fosilizado, y con el tiempo algún organismo más avanzado quedará fosilizado encima tuyo.

El registro fósil —La Brea Tar Pits[2]— de la tecnología software es Internet. Cualquier cosa que aparezca allí se puede tomar de forma gratuita (posiblemente ilegal, pero gratuita). Los ejecutivos de compañías como Microsoft tienen que acostumbrarse a la experiencia —impensable en otras industrias— de invertir millones de dólares en el desarrollo de nuevas tecnologías, tales como navegadores web, y luego ver cómo aparece en Internet el mismo software, o un software equivalente, dos años, un año, o incluso pocos meses después.

Al seguir desarrollando nuevas tecnologías y añadiendo posibilidades a sus productos, pueden mantenerse un paso por delante del proceso de fosilización, pero algunos días deben de sentirse como mamuts atrapados en La Brea, usando todas sus energías para salir adelante, una y otra

---

[2]Las canteras de Rancho La Brea Tar Pits es un yacimiento de fósiles situado en el Condado de Los Angeles (EE.UU.). Durante casi cuarenta mil años, la mina (*pits*) ha emitido una gran cantidad de brea, chapapote pegajoso y espeso que ha dejado atrapados a lo largo del tiempo a muchos especímenes de plantas y animales prehistóricos. [*N. del E.*]

vez, escapando de la pegajosa brea caliente que quiere cubrirles y engullirles.

La supervivencia en esta biosfera requiere colmillos fuertes y pies que puedan pisotear en un extremo de la organización, y Microsoft es famosa por tenerlos. Pero pisotear a los otros mamuts en la brea sólo puede mantenerte vivo cierto tiempo. El peligro es que, con su obsesión por mantenerse fuera de las capas fósiles, estas compañías olviden lo que hay por encima de la biosfera: el ámbito de la nueva tecnología. En otras palabras, deben seguir con sus armas primitivas y bastos instintos competitivos, pero también han de desarrollar cerebros potentes. Parece ser que esto es lo que está haciendo Microsoft con su departamento de investigación, que contrata a personas inteligentes por doquier. (Y aquí debo mencionar que, aunque conozco y me relaciono con varias personas del departamento de investigación de esa compañía, nunca hablamos de negocios, y no tengo ni idea de qué demonios están haciendo. He aprendido mucho más sobre Microsoft usando el sistema operativo Linux de lo que habría aprendido usando Windows).

Da igual cómo hiciera antes dinero Microsoft; hoy en día, hace dinero gracias a una especie de arbitraje temporal. *Arbitraje*, en el sentido habitual, significa hacer dinero aprovechándose de las diferencias en los precios de algo en diferentes mercados. En otras palabras, es espacial y se basa sobre el hecho de que el árbitro sabe por qué tecnologías pagará dinero la gente el año que viene, y cuánto tardarán esas tecnologías en volverse gratuitas. Lo que el arbitraje espacial y temporal tienen en común es que ambos pivotan sobre la información extremadamente buena del árbitro; información sobre los gradientes de precios en un momento dado en un caso, sobre los gradientes de precios a lo largo del tiempo en un lugar dado en el otro.

Así que Apple/Microsoft ofrecen nuevas posibilidades a sus usuarios casi a diario, con la esperanza de que un flujo constante de genuinas innovaciones técnicas, combinadas con el fenómeno del «quiero creer» impedirá que sus clientes miren al otro lado de la carretera, hacia los sistemas operativos, mejores y más baratos, que tienen disponibles. La cuestión es si esto tiene sentido a largo plazo. Si Microsoft es adicta a los sistemas operativos como Apple lo es al hardware, entonces se apostarán la camisa por sus sistemas operativos, y vincularán todas sus nuevas aplicaciones y sistemas operativos a ellos. Su supervivencia dependerá entonces de estas dos cosas: añadir más posibilidades a sus sistemas operativos, de tal modo que sus clientes no se pasen a las alternativas más baratas, y mantener la imagen que, de algún modo misterioso, les da a estos clientes la sensación de que obtienen algo a cambio de su dinero.

Este último es un fenómeno cultural verdaderamente extraño e interesante.

La cultura de la interfaz

HACE UNOS AÑOS[1] entré en una tienda cualquiera y me encontré con la siguiente escena: cerca de la entrada había una pareja joven frente a un gran mostrador de cosméticos. El hombre sostenía estólidamente una cesta de la compra en las manos mientras su compañera arramblaba con productos de maquillaje del mostrador y los apilaba en la cesta. Desde entonces siempre he pensado en ese hombre como la personificación de una interesante tendencia humana: no sólo no nos ofenden las imágenes manufacturadas sino que nos gustan. Prácticamente insistimos en ello. Estamos ansiosos por ser cómplices de nuestro propio engaño: por pagar dinero por el pase a un parque temático, votar a un tipo que obviamente nos está mintiendo o permanecer de pie sosteniendo la cesta que se llena de cosméticos.

Hace poco estuve en Disney World, concretamente en la parte llamada el Reino Mágico, caminando por Main Street USA. Esta es la perfecta pequeña ciudad victoriana y cuca que lleva al castillo Disney. Había mucha gente; nos abríamos camino más que caminábamos. Justo delante mío había un hombre con una videocámara. Era una de esas nuevas videocámaras en las que, en vez de mirar por un visor, contemplas una pantalla plana en color del tamaño de un

[1] Excusas por el título de este capítulo a Steve Johnson, autor de *Interfaz Culture: How New Technology Transforms the Way We Create and Comunicate,* [«La cultura de la interfaz: cómo las nuevas tecnologías transforman el modo en que creamos y comunicamos»], San Francisco, Harper, 1997.

naipe, que televisa en directo lo que quiera que la cámara esté grabando. Sostenía el aparato cerca de la cara, de tal modo que le tapaba la vista. En vez de ir a ver una pequeña ciudad de verdad gratis, había pagado dinero por ver una falsa, y en vez de verla a simple vista estaba contemplándola por televisión.

Y en vez de quedarme en casa y leer un libro, yo le estaba mirando a él.

La preferencia de los estadounidenses por las experiencias mediadas resulta bastante obvia, y no voy a dar la murga con ello. Ni siquiera voy a hacer comentarios desdeñosos acerca de ello —después de todo, yo estaba en Disney World como cliente de pago—. Pero claramente está relacionado con el colosal éxito de las GUI, así que tengo que hablar algo acerca de ello. A los de Disney se le dan mejor que a nadie las experiencias mediadas. Si entendieran qué son los sistemas operativos, y por qué los usa la gente, aplastarían a Microsoft en uno o dos años.

En la sección de Disney World llamada el Reino Animal hay una nueva atracción, que se supone abrirá en marzo de 1999, llamada el Viaje por la Jungla del Maharajá. Lo habían abierto como anticipo cuando yo estuve allí. Es una reproducción completa, piedra a piedra, de una hipotética ruina en las junglas de la India. Según decían, fue construida por un rajá local en el siglo XVI como reserva de caza. El rajá iba allí con sus principescos huéspedes a cazar tigres de Bengala. Con el paso del tiempo, quedó abandonada y la ocuparon los tigres y los monos; finalmente, en torno a la época de la independencia de la India, se convirtió en una reserva natural del gobierno, ahora abierto a los visitantes.

El lugar se parece más a lo que he descrito que ningún edificio real que se pueda encontrar en la India. Todas las piedras en los muros derrumbados tenían el aspecto de haber sido desgastados por las lluvias monzónicas durante siglos, la pintura de las paredes está descascarillada y apagada y los tigres de Bengala se mueven entre las columnas rotas. Allí donde se podrían realizar reparaciones modernas en la antigua estructura, se han hecho, pero no como las llevarían a cabo los ingenieros de la Disney, sino ahorrativos encargados indios, con bambú y barras herrumbrosas. La herrumbre está pintada, claro, y protegida de la herrumbre auténtica por una capa de plástico transparente, pero no se nota a menos que uno se agache.

En cierto punto se puede caminar junto a un muro de piedra con una serie de desgastados frisos antiguos esculpidos. Un extremo del muro se ha derrumbado y caído a tierra, quizás debido a algún terremoto largo tiempo olvidado, y uno o dos paneles tienen anchas fisuras, pero la historia sigue siendo legible: primero, el caos primordial lleva a la creación de muchas especies animales. Luego, vemos el Árbol de la Vida rodeado de diversos animales. Esta es una alusión obvia al enorme Árbol de la Vida que domina el centro del Reino Animal de Disney, igual que el Castillo domina el Reino Mágico o la Esfera domina Epcot. Pero está hecho en un estilo históricamente correcto, y probablemente engañaría a cualquiera que no tuviera un doctorado en historia del arte indio.

El siguiente panel muestra a un *homo sapiens* bigotudo derribando el Árbol de la Vida con una cimitarra y a animales huyendo en todas direcciones. El panel que va después muestra al errado humano golpeado por un tsunami, parte de un Diluvio presumiblemente provocado por su estupidez.

El panel final muestra al Brote de la Vida que vuelve a crecer, pero ahora el Hombre ha abandonado su afilada arma y se ha unido a los demás animales, que lo rodean para ensalzarlo y adorarlo.

Es, en otras palabras, una profecía del *cuello de botella:* la situación, planteada habitualmente por los modernos ecologistas, de que el mundo se enfrentará pronto a un periodo de graves tribulaciones ecológicas que durarán unas pocas décadas o siglos y acabarán cuando encontremos un nuevo y armonioso *modus vivendi* con la Naturaleza.

En conjunto, el friso es una obra bastante brillante. Obviamente no es una antigua ruina india, y alguna persona o personas vivas merecen ser elogiadas. Pero no hay firmas en la reserva de caza de Maharajá en Disney World. No hay firmas en nada, porque arruinaría el efecto si largos créditos colgaran de cada ladrillo desgastado a medida, como en las películas de Hollywood.

Entre los guionistas de Hollywood, Disney tiene la reputación de ser una madrastra verdaderamente malvada. No resulta difícil ver por qué. Disney está en el negocio de los productos de ilusión sin fisuras —un espejo mágico que refleja el mundo mejor de lo que realmente es—. Pero un escritor está hablando literalmente a sus lectores, no sólo creando un ambiente o presentándoles algo donde mirar; y así como la interfaz de línea de comandos abre un canal mucho más directo y explícito entre usuario y máquina que la GUI, lo mismo sucede con palabras, escritor y lector.

La palabra, al final, es el único sistema para codificar los pensamientos —el único medio— que no es fungible, que se niega a disolverse en el torrente devorador de los medios electrónicos (los turistas más ricos en Disney World llevan camisetas con los nombres de diseñadores famosos impresos, porque los propios diseños pueden copiarse fácilmente y con impunidad. El único modo de fabricar ropa que no puede copiarse legalmente es imprimir palabras con copyright y marca registrada; una vez se ha dado ese paso, la ropa misma ya no importa realmente, y así una camiseta es tan buena como cualquier otra cosa. Las camisetas con palabras caras son ahora la insignia de la clase alta. Las camisetas con palabras baratas, o sin palabras, son para el común de los mortales).

Pero esta cualidad especial de las palabras y de la comunicación escrita tendría el mismo efecto sobre el producto de la Disney que un graffiti de spray sobre un espejo mágico. Así que la Disney lleva a cabo la mayor parte de su comunicación sin recurrir a las palabras, y en su mayor parte, no se echa de menos las palabras. Algunas de las propiedades más antiguas de la Disney, como Peter Pan, Winnie Pooh, y Alicia en el País de las Maravillas, salieron de libros. Pero los nombres de sus autores se mencionan raramente, si es que se mencionan, y no se pueden comprar los libros originales en la tienda Disney. Si se pudiera, parecerían viejos y extraños, como versiones muy raras de los originales más puros y auténticos de la Disney. Comparados con producciones más recientes como *La Bella y la Bestia* y *Mulan,* las películas de la Disney basadas en estos libros (en particular *Alicia en el País de las Maravillas* y *Peter Pan*) parecen profundamente extrañas, y no del todo apropiadas para niños. Lo cual es razonable, porque Lewis Carroll y J.M. Barrie eran hombres muy raros, y la naturaleza de la palabra escrita es tal que su rareza personal se filtra a través de todas las capas de *disneyficación* como rayos X a través de una pared. Probablemente, por esta misma razón, la Disney parece haber dejado de comprar libros y ahora encuentra sus temas y caracteres en los relatos tradicionales, que tienen la cualidad lapidaria y gastada por el tiempo de los antiguos bloques de piedra de las ruinas del Maharajá.

Si siguiéramos a esos turistas a sus casas, podríamos encontrar arte, pero sería el tipo de arte folclórico no firmado que venden en las tiendas de la Disney de tema africano y asiático. En general, sólo parecen estar cómodos con medios que han sido ratificados por su antigüedad, por su aceptación popular masiva o por ambas cosas.

En este mundo, los artistas son como los obreros anónimos y analfabetos que construyeron las grandes catedrales en Europa y luego desaparecieron en tumbas anónimas del cementerio. La catedral en conjunto es apabullante y conmovedora a pesar de, y posiblemente debido a, el hecho de que no tenemos ni idea de quién la construyó. Cuando caminamos por ella comulgamos no con obreros individuales sino con toda una cultura.

Disney World funciona del mismo modo. Si se es un intelectual, un lector o un escritor de libros, lo más amable que se puede decir al respecto es que la ejecución es soberbia. Pero resulta fácil encontrarlo todo un poco siniestro, porque falta algo: la traducción de todo su contenido a palabras escritas, claras y explícitas, las atribución de las ideas a personas específicas. No se puede discutir con ello. Parece como si se estuviera pasando por alto un montón de cosas, como si Disney World nos estuviera engañando, y posiblemente colándonos todo tipo de asunciones ocultas y pensamiento débil.

Pero esto es exactamente lo mismo que se pierde en la transición de la interfaz de línea de comandos a la GUI.

La Disney y Apple/Microsoft están en el mismo negocio: cortocircuitar la laboriosa y explícita comunicación verbal con interfaces de diseño caro. La Disney es una especie de interfaz de usuario en sí misma —y más que meramente gráfica—. Llamémosla *interfaz sensorial.* Puede aplicarse a cualquier cosa en el mundo, real o imaginada, aunque a un precio apabullante.

¿Por qué rechazamos las interfaces basadas en la palabra, y preferimos las gráficas o sensoriales —una tendencia que explica el éxito tanto de Microsoft como de la Disney?

Parte de ello es simplemente que el mundo es ahora muy complicado —mucho más complicado que el mundo de los cazadores-recolectores con el cual evolucionaron nuestros cerebros— y sencillamente no podemos manejar todos los detalles. Tenemos que delegar. No tenemos más opción que confiar en algún artista anónimo de la Disney o en algún programador de Apple o Microsoft para que elijan por nosotros, nos libren de algunas opciones y nos den un resumen convenientemente empaquetado.

Pero más importante es el hecho de que durante este siglo el intelectualismo falló, y todo el mundo lo sabe. En lugares como Rusia y Alemania, la gente común renunció a su control sobre los modos de vida tradicionales, costumbres y religión, y permitió que los intelectuales llevaran el cotarro, y los intelectuales lo estropearon todo y convirtieron el siglo en un matadero. Aquellos intelectuales de tanta palabrería solían percibirse como algo meramente tedioso; ahora también parecen algo peligrosos.

Los estadounidenses somos los únicos que no salimos malparados en ningún momento de todo esto. Somos libres y prósperos porque heredamos sistemas políticos y de valores fabricados por un conjunto dado de intelectuales del siglo XVIII que por casualidad acertaron. Pero hemos perdido contacto con esos intelectuales, y con cualquier cosa parecida al intelectualismo, hasta el punto de no leer libros ya, aunque sabemos leer. Estamos mucho más cómodos transmitiéndoles esos valores a las generaciones futuras de forma no-verbal, mediante el proceso de inmersión mediática. Parece que esto funciona hasta cierto punto, porque la policía en muchos países ahora se queja de que los arrestados insisten en que les lean sus derechos, como en las películas de policías estadounidenses. Cuando se les explica que están en un país diferente, se indignan. Puede que las reposiciones de Starsky y Hutch, dobladas a diversas lenguas, resulten ser, a largo plazo, una fuerza más potente en favor de los derechos humanos que la Declaración de Independencia.

Una cultura enorme, rica y nuclear que propaga sus valores nucleares mediante la inmersión mediática parece una mala idea. Está el riesgo obvio de errar. Las palabras son el único medio inmutable que tenemos, que es el motivo por el cual son el vehículo preferido para conceptos extremadamente importantes como los Diez Mandamientos, el Corán y la Declaración de Derechos. A menos que los mensajes transmitidos por nuestros medios vayan ligados a algún conjunto fijo de preceptos, pueden desperdigarse por doquier y posiblemente llenar la mente de la gente de estupideces.

Orlando tenía una base militar llamada McCoy Air Force Base, con largas pistas desde las que podían despegar los B-52 para llegar a Cuba o a cualquier otro lugar, cargados de bombas nucleares. Pero ahora McCoy ha sido desmantelada y sus instalaciones se han destinado a otros fines. El aeropuerto civil de Orlando las ha absorbido. Las largas pistas se usan ahora para descargar turistas llegados en vuelos 747 desde Brasil, Italia, Rusia y Japón, a fin de que vengan a Disney World y empaparse de nuestros medios durante un tiempo.

Para las culturas tradicionales, especialmente las basadas en la palabra como el Islam, esto resulta infinitamente más amenazante de lo que lo fueron jamás los B-52. Resulta obvio para cualquiera fuera de los Estados Unidos que nuestras archimuletillas, multiculturalismo y diversidad, son fachadas que encubren (en muchos casos involuntariamente) una tendencia global a erradicar las diferencias culturales. El pilar básico del multiculturalismo (o de «honrar la diversidad», o como se quiera llamarlo) es que las personas tienen que dejar de juzgarse unas a otras —dejar de aseverar (y, gradualmente, dejar de creer) que esto está bien y esto está mal, que una cosa es fea y otra hermosa, que Dios existe y tiene estas o aquellas cualidades.

La lección que la mayor parte de la gente ha extraído del siglo XX es que, para que un gran número de diferentes culturas coexistan pacíficamente en el globo (o incluso en el barrio), es necesario que la gente suspenda el juicio de este modo. De ahí (argumento) nuestra sospecha, u hostilidad, respecto de todas las figuras de autoridad en la cultura moderna. Como explicó David Foster Wallace en su ensayo *E Unibus Pluram,* este es el mensaje fundamental de la televisión; es el mensaje que la gente se lleva a casa, de cualquier modo, tras llevar inmersos en los medios el tiempo suficiente. No está expresado en esos términos altisonantes, claro. Se transmite a través de la presunción de que todas las figuras de autoridad —maestros, generales, policías, sacerdotes, políticos— son bufones hipócritas, y que el cinismo descreído es el único modo de ser.

El problema es que una vez que nos hemos librado de la capacidad de juzgar lo bueno y lo malo, lo verdadero y lo falso, etc., ya no queda cultura. Todo lo que queda son los bailes folclóricos y el macramé. La capacidad de juicio, de creencia, es el fin mismo de tener una cultura. Creo que por eso aparecen a veces tipos con metralletas en lugares como Luxor, y empiezan a disparar a los occidentales. Entienden perfectamente la lección de la base aérea McCoy. Cuando los hijos llegan con gorras ladeadas de los Chicago Bulls, los padres enloquecen.

La anticultura global transmitida a todos los rincones del mundo por la televisión es una cultura en sí misma, y según los estándares de grandes y antiguas culturas como el Islam o Francia, parece inmensamente inferior, al menos al principio. Los único bueno que se puede decir de ella es que hace que guerras mundiales y holocaustos parezcan menos probables —¡y de hecho eso es algo bastante bueno!

El único problema real es que cualquiera que no tenga más cultura que esta monocultura global está completamente jodido. Cualquiera que crezca viendo la televisión, que nunca vea nada de religión o filosofía, se críe en una atmósfera de relativismo moral, aprenda ética viendo escándalos sexuales en el telediario, y vaya a una universidad donde los posmodernos se desviven por demoler las nociones tradicionales de verdad y cualidad, va a salir al mundo como un ser humano bastante incapaz. Y —de nuevo— tal vez el fin de todo esto es hacernos incapaces, de modo que no nos bombardeemos mutuamente con armas nucleares.

Por otro lado, si te crías en el ámbito de una cultura dada, acabas con un conjunto básico de herramientas que se pueden usar para pensar y comprender el mundo. Puedes usar esas herramientas para rechazar la cultura en que te criaste, pero al menos tienes algunas herramientas.

En este país, la gente que lleva el cotarro —los que llenan los bufetes y las juntas directivas— comprende todo esto a cierto nivel. Apoyan el multiculturalismo y la diversidad y la suspensión del juicio de boquilla, pero no educan a sus propios hijos así. Tengo amigos altamente educados y técnicamente sofisticados que se han mudado a pequeñas ciudades de Iowa para vivir y criar a sus hijos, y hay enclaves judíos hasidim en Nueva York donde muchos niños se crían según creencias tradicionales. Cualquier comunidad suburbana puede considerarse un lugar donde personas que tienen ciertas creencias (básicamente implícitas) van a vivir entre otros que piensan de igual manera.

Y esta gente no sólo se siente responsable respecto a sus propios hijos, sino con el país en general. Algunos miembros de la clase alta son viles y cínicos, por supuesto, pero muchos pasan al menos parte de su tiempo preocupándose por la dirección en que va el país, y sus propias responsabilidades. Y así, cuestiones que son importantes para los intelectuales lectores de libros, como el colapso ambiental global, acaban por filtrarse a través de la cultura de masas y aparecen como antiguas ruinas hindúes en Orlando.

Puede que se estén preguntando: ¿qué narices tiene que ver todo esto con los sistemas operativos? Como ya he dicho, no hay modo de explicar la dominación del mercado de sistemas operativos por Apple/Microsoft sin explicaciones culturales, así que no puedo llegar a ninguna parte en este ensayo sin hacerles saber antes de dónde vengo en lo que concierne a la cultura contemporánea.

La cultura contemporánea es un sistema de dos niveles, como los morlocks y los eloi de *La máquina del tiempo,* de H.G. Wells, salvo que está del revés. En *La máquina del tiempo,* los eloi eran la amanerada clase alta, mantenida por montones de morlocks subterráneos que hacían que los engranajes tecnológicos se movieran. Pero en nuestro mundo es al revés. Los morlocks son minoría, y hacen que las cosas se muevan porque comprenden cómo funciona todo. Los mucho más numerosos eloi aprenden todo lo que saben por verse inmersos desde su nacimiento en medios electrónicos dirigidos y controlados por los morlocks lectores de libros. Así que muchas personas ignorantes serían peligrosas si se las apuntara en la dirección equivocada, con lo cual hemos desarrollado una cultura popular que a) es increíblemente infecciosa y b) neutraliza a toda persona que se ve infectada, haciéndolos reticentes a emitir juicios e incapaces de tomar posiciones.

Los morlocks, que tienen la energía e inteligencia como para aprehender los detalles, van y dominan temas complejos y producen *interfaces sensoriales* tipo Disney, de tal modo que los eloi puedan entender el meollo sin tener que forzar la mente o soportar el aburrimiento. Esos morlocks van a la India y tediosamente exploran cientos de ruinas, luego vuelven a casa y construyen versiones higiénicas y sin bichos: el *Selecciones del Reader's Digest,* por así decir. Esto cuesta un montón, porque los morlocks insisten en que les den buen café y billetes de avión en primera, pero no es problema porque a los eloi les gusta que los deslumbren y pagarán gustosos.

Me doy cuenta de que la mayor parte de esto probablemente suena desdeñoso y amargado hasta el absurdo: el típico intelectual pijo con un berrinche por culpa de esos filisteos analfabetos. Como si yo fuera una especie de Moisés bajando solo de la montaña, con las tablas de los Diez Mandamientos grabadas en piedra inmutable —la interfaz de línea de comandos original— y cabreándose con los débiles hebreos no iluminados que adoran imágenes. No sólo eso, sino que parece que creo que hay una especie de teoría de la conspiración.

Pero eso no es lo que quiero decir con todo esto. La situación que describo aquí podría ser mala, pero no tiene por qué ser mala, y no es necesariamente mala ahora.

La cuestión es que, sencillamente, estamos demasiado ocupados hoy en día como para comprenderlo todo con detalle. Y es mejor comprenderlo por una interfaz, oscuramente, que no comprenderlo en absoluto. Mejor que diez millones de eloi vayan al Safari por el Kilimanjaro en Disney World que no que mil cirujanos cardiovasculares y directivos de aseguradoras vayan de safari auténtico por Kenia. La frontera entre ambas clases es más porosa de lo que he dado a entender. Constantemente me encuentro con tipos normales —albañiles, mecánicos, taxistas, gente de a pie en general— que básicamente carecían de cultura hasta que algo hizo necesario que se convirtieran en lectores y empezaran a pensar en serio acerca de las cosas. Tal vez tuvieron que vérselas con el alcoholismo, tal vez fueron a la cárcel, o enfermaron, o sufrieron una crisis de fe, o simplemente se aburrieron. Tales personas pueden aprender sobre temas particulares a toda prisa. A veces su falta de una educación amplia les lleva a acometer empresas intelectuales desquiciadas pero bueno, al menos la empresa intelectual desquiciada es un buen ejercicio. El fantasma de una política controlada por los caprichos y veleidades de los votantes que creen realmente que hay diferencias significativas entre las cerveza Bud Lite y Miller Lite, y que creen que la lucha libre es real, es naturalmente alarmante para aquellos que no lo creen. Pero los países controlados mediante la interfaz de la línea de comandos, por así decirlo, por sesudos intelectuales, ya sean religiosos o seculares, son por lo general tristes lugares donde vivir. La gente sofisticada se burla de los entretenimientos disneyescos por facilones y asacarinados, pero si el resultado es provocar reflejos básicamente cálidos y simpáticos a nivel preverbal en cientos de millones de iletrados inmersos en los medios, no pueden ser tan malos. Anoche matamos una langosta en nuestra cocina y mi hija lloró durante una hora. Los japoneses, que solían ser el pueblo más feroz del mundo, están obsesionados con adorables personajes de dibujos animados. Mi propia familia —la gente que mejor conozco— está dividida de modo más o menos equitativo entre personas que probablemente lean este ensayo y personas que casi con toda certeza no lo hará, y no puedo decir a ciencia cierta que un grupo sea necesariamente más cálido, feliz o mejor adaptado que el otro.

Morlocks y Eloi al teclado

EN LOS TIEMPOS de la interfaz de línea de comandos, los usuarios eran todos morlocks que tenían que convertir sus pensamientos en símbolos alfanuméricos e introducirlos a mano, un proceso insufriblemente tedioso que eliminaba toda ambigüedad, revelaba todas las asunciones ocultas y castigaba cruelmente la pereza y la imprecisión. Entonces los hacedores de interfaces se pusieron a trabajar en sus GUI, e introdujeron una nueva capa semiótica entre la gente y las máquinas. Las personas que usan tales sistemas han renunciado a la responsabilidad, y al poder, de enviar bits directamente al chip que lleva a cabo la aritmética, y le han pasado esa responsabilidad y poder al sistema operativo. Esto resulta tentador porque dar instrucciones claras, a alguien o a algo, es difícil. No podemos hacerlo sin pensar y, dependiendo de la complejidad de la situación, debemos pensar intensamente en cosas abstractas y considerar cualquier número de ramificaciones para hacerlo bien. Para la mayoría de nosotros, esto es una ardua tarea. Queremos que las cosas sean más fáciles. La medida de cuánto lo queremos viene dada por el grueso de la fortuna de Bill Gates.

El sistema operativo (por tanto) se ha convertido en una especie de instrumento para ahorrarse trabajo intelectual, que traduce las intenciones vagamente expresadas de los humanos a bits. De hecho, les pedimos a nuestros ordenadores que tomen responsabilidades que siempre se han considerado propias de seres humanos: queremos que comprendan nuestros deseos, que prevean nuestras necesidades, que establezcan conexiones, que desempeñen tareas rutinarias sin necesidad de pedírselo, que nos recuerden lo que tendría que recordársenos a la vez que filtran el ruido. En los niveles más elevados (es decir, más próximos al usuario) esto tiene lugar mediante una serie de convenciones —menús, botones, etc—. Estas funcionan en el sentido en que funcionan las analogías: ayudan a los eloi a comprender conceptos abstractos o poco familiares comparándolos con algo conocido. Pero se usa el término más pretencioso de *metáfora.*

El concepto que lo englobaba todo en MacOS era la «metáfora del escritorio», que subsumía cierto número de metáforas menores (y a menudo contradictorias, o al menos mezcladas). Con una GUI, un archivo (frecuentemente llamado «documento») se metafrasea como una ventana en pantalla (al que se denomina «escritorio»). La ventana siempre es demasiado pequeña para contener el documento, así que uno «se mueve» o, más pretenciosamente, «navega» por el documento «pinchando y arrastrando» el «dedo» en la «barra de desplazamiento». Cuando se «teclea» (usando un teclado) o «dibuja» (usando un «ratón») en la «ventana» o se usan «menús» desplegables y «cuadros de diálogo» para manipular sus contenidos, los resultados del trabajo se almacenan (al menos en teoría) en un «archivo», y luego la misma información se recupera en otra «ventana». Cuando ya no se necesita, se «arrastra» a la «papelera».

Hay una mezcla masiva y promiscua de metáforas aquí y podría deconstruirla hasta que las ranas criaran pelo, pero no lo haré. Considérese sólo una palabra: «documento». Cuando documentamos algo en el mundo real, creamos registros fijos, permanentes e inmutables de ello. Pero los documentos de un ordenador son volátiles, efímeras constelaciones de datos. A veces (como cuando se abren o guardan), el documento que aparece en la ventana es idéntico al que está almacenado, bajo el mismo nombre, en un archivo de disco, pero otras veces (como cuando se hacen cambios sin guardarlos), es completamente diferente. En cualquier caso, cada vez que se pulsa «Guardar», se aniquila la versión previa del documento, reemplazándola por lo que quiera que aparezca en la ventana en ese momento. Así que, incluso la palabra *guardar,* se usa en un sentido que es grotescamente engañoso: «destruir una versión, guardar otra» sería más exacto.

Cualquiera que use un procesador de textos durante mucho tiempo inevitablemente sufrirá la experiencia de emplear horas de trabajo en un documento largo y luego perderlo porque el ordenador falla o se corta la luz. Hasta el momento en que desaparece de pantalla, el documento parece tan sólido y real como si estuviera impreso en papel y tinta. Pero un momento después, sin avisar, se ha esfumado, completa e irremediablemente, como si nunca hubiera existido. El usuario queda con una sensación de desorientación (por no hablar del cabreo) proveniente de un trasquilón metafórico: uno se da cuenta de que ha estado viviendo y pensando dentro de una metáfora que es esencialmente falsa.

Así que las interfaces gráficas usan metáforas para hacer que la informática resulte más fácil, pero son malas metáforas. Aprender a usarlas es esencialmente un juego de palabras, el proceso de aprender nuevas definiciones de palabras como «ventana» y «documento» y «guardar», que son diferentes, y en muchos casos diametralmente opuestas a las antiguas. Por muy improbable que parezca, esto ha salido muy bien, al menos desde el punto de vista comercial, lo cual significa que Apple/Microsoft han hecho mucho dinero con ello. Todos los otros sistemas operativos modernos han aprendido que, para ser aceptados por los usuarios, han de ocultar sus entrañas bajo el mismo tipo de adornos. Esto tiene ciertas ventajas: si se sabe usar un sistema operativo de GUI, probablemente se puede deducir cómo usar cualquier otro en pocos minutos. Todo funciona de modo algo distinto, como las cañerías europeas pero, enredando un poco, se puede escribir una nota y navegar por la red.

La mayor parte de la gente que compra sistemas operativos (si es que se molestan en comprarlo) no comparan las funciones subyacentes, sino el aspecto y sensación superficiales. El comprador medio de un sistema operativo no paga realmente, y no le interesa especialmente, el código de bajo nivel que asigna memoria y escribe bytes en el disco. Lo que compramos realmente es un sistema de metáforas. Y —mucho más importante— a lo que nos vendemos es al presupuesto implícito de que las metáforas son un buen modo de tratar con el mundo.

Desde hace poco se ha vuelto disponible un montón de nuevo hardware que les proporciona a los ordenadores numerosos modos interesantes de afectar al mundo real: hacer que las impresoras escupan papel, dirigir haces radiactivos hacia enfermos de cáncer, crear películas realistas sobre el Titanic. Windows se usa ahora como sistema operativo para cajas registradoras y cajeros automáticos. El sistema de mi televisión por satélite emplea una especie de GUI (interfaz gráfica) para cambiar de canal y mostrar guías de programas. Los modernos teléfonos móviles llevan una cruda GUI metido en una diminuta pantalla. Incluso Lego tiene una GUI: se puede comprar un juego de Lego llamado Mindstorms que permite construir pequeños robots Lego y programarlos mediante una GUI en el ordenador.

Así que ahora le pedimos a la GUI que haga mucho más que servir de máquina de escribir glorificada. Ahora queremos que se convierta en una herramienta generalizada para tratar con la realidad. Esto ha hecho que las compañías que viven de sacar nueva tecnología al mercado de masas vivan una bonanza económica.

Obviamente, no se puede vender un complicado sistema tecnológico a la gente sin algún tipo de interfaz que les permita usarlo. La dinamo de combustión interna fue una maravilla tecnológica en su época, pero era inútil como bien de consumo hasta que le conectaron una palanca de cambios, transmisión, volante y frenos. Esa extraña colección de cacharros, que sobrevive hasta nuestros días en cada coche que surca las carreteras, constituye lo que hoy llamaríamos una interfaz de usuario. Pero si los coches se hubieran inventado después que los Macintosh, los fabricantes de coches no se habrían molestado en diseñar todos esos complicados dispositivos. Tendríamos una pantalla de ordenador por salpicadero, y un ratón (o como mucho un joystick) por volante, y cambiaríamos de marchas desplegando un menú:

```
APARCAR

MARCHA ATRÁS

PUNTO MUERTO

3
2
1

Ayuda...
```

Así, unas pocas líneas de código pueden sustituir cualquier interfaz mecánica imaginable. El problema es que en muchos casos el sustituto es defectuoso. Conducir un coche mediante una GUI sería una experiencia horrible. Incluso si la GUI estuviera totalmente libre de fallos, sería increíblemente peligroso, porque los menús y botones sencillamente no pueden responder tan bien como los controles mecánicos directos. El padre de mi amigo, el señor que restauraba el descapotable, nunca se habría tomado la molestia si hubiera ido equipado con una GUI. No habría sido divertido.

El volante y la palanca de cambios se inventaron en una era en la que la tecnología más complicada en la mayor parte de las casas era la batidora de mantequilla. Aquellos primeros fabricantes de coches tenían mucha suerte, ya que podían diseñar la interfaz que resultara más adecuada para la tarea de conducir un automóvil, y la gente la aprendía. Lo mismo sucedió con el teléfono de marcado y la radio AM. Ya en la Segunda Guerra Mundial, la mayor parte de la gente conocía varias interfaces: no sólo podían batir mantequillas, sino también conducir un coche, marcar en el teléfono, conectar la radio, encender un mechero y cambiar una bombilla.

Pero ahora cualquier cosita —relojes de pulsera, vídeos, hornillos— está lleno de funcionalidades, y cada funcionalidad es inútil sin interfaz. Si usted es como yo y como la mayoría de consumidores, nunca ha usado el noventa por ciento de las funcionalidades de su microondas, vídeo o teléfono móvil. Ni siquiera sabe que estas funcionalidades existen. El pequeño beneficio que podrían aportarle queda anulado por la pura molestia de tener que aprenderlas. Esto debe de ser un gran problema para los fabricantes de bienes de consumo, porque no pueden competir sin ofrecer características.

Ya no es aceptable que los ingenieros inventen toda una nueva interfaz de usuario para cada nuevo producto, como hicieron en el caso del automóvil, en parte porque resulta demasiado caro y en parte porque hay un límite en lo que puede aprender la gente normal. Si el vídeo se hubiera inventado hace cien años, tendría una ruedecita para la sintonización y una palanca para avanzar y rebobinar, y una gran asa de hierro forjado para cargar o expulsar los cassettes. Llevaría un gran reloj analógico delante, y habría que ajustar la hora moviendo las manillas en la esfera. Pero debido a que el vídeo se inventó cuando se inventó —durante una especie de incómodo periodo de transición entre la era de las interfaces mecánicas y las GUI— tiene sólo unos cuantos botones delante y, para fijar la hora, hay que pulsar los botones de modo correcto. Esto le debe de haber parecido bastante razonable a los ingenieros responsables, pero para muchos usuarios es sencillamente imposible. De ahí el famoso 12:00 que parpadea en tantos vídeos. Los informáticos lo llaman el *problema del doce parpadeante.* Cuando hablan de ello, empero, no suelen estar hablando de vídeos.

Los vídeos modernos habitualmente tienen algún tipo de programación en pantalla, lo cual significa que se puede fijar la hora y controlar las demás funcionalidades mediante una especie de GUI primitivo. Los GUI también tienen botones virtuales, claro, pero también tienen otros tipos de controles virtuales, como botones de radio, casillas que tachar, espacios para introducir textos, esferas y barras. Las interfaces compuestas de estos elementos parecen ser mucho más fáciles para muchas personas que pulsar esos botoncitos en la máquina, y así el propio 12:00 parpadeante está desapareciendo lentamente de los salones de Estados Unidos. El *problema* del doce parpadeante ha pasado a otras tecnologías.

Así que la GUI ha pasado de ser una interfaz para ordenadores personales a convertirse en una especie de metainterfaz que se emplea en cualquier nueva tecnología de consumo. Raramente es ideal, pero tener una interfaz ideal o incluso buena ya no es la prioridad; lo importante ahora es tener algún tipo de interfaz que los clientes usen realmente, de tal modo que los fabricantes puedan afirmar con toda seriedad que ofrecen nuevas posibilidades.

Queremos GUI básicamente porque son convenientes y porque son fáciles —o al menos la GUI hace que así parezca—. Por supuesto, nada es realmente fácil y simple, y poner una bonita interfaz no cambia ese hecho. Un coche controlado a través de una GUI sería más fácil de conducir que uno controlado por los pedales y el volante, pero sería increíblemente peligroso.

Al usar GUI todo el tiempo, hemos aceptado sin darnos cuenta una premisa que pocas personas aceptarían si se les planteara directamente, a saber: que las cosas difíciles pueden hacerse fáciles, y las complicadas pueden volverse simples, acoplándoles la interfaz adecuada. Para comprender lo raro que es todo esto, imagínense que las críticas de libros se escribieran según el mismo sistema de valores que aplicamos a las interfaces de usuario: la escritura de este libro es maravillosamente simple; el autor pasa por encima de temas complicados y emplea generalizaciones ramplonas casi en cada oración. Los lectores rara vez tendrán que pensar, y se les ahorrará toda la dificultad y el tedio generalmente asociados con la lectura de libros anticuados. Mientras nos limitemos a operaciones sencillas como fijar la hora en nuestro vídeo, no es para tanto. Pero cuando tratamos de hacer cosas más ambiciosas con nuestra tecnología, inevitablemente nos topamos con el problema de «el trasquilón metafórico».

El trasquilón metafórico

EMPECÉ A USAR MICROSOFT WORD en cuanto sacaron la primera versión en torno a 1985. Tras algunos problemas iniciales descubrí que era mejor herramienta que MacWrite, que era su único competidor en aquel momento. Escribí un montón de cosas en versiones tempranas de Word, guardándolo todo en disquetes, y transferí los contenidos de todos mis disquetes a mi primer disco duro, que adquirí en torno a 1987. A medida que salían nuevas versiones de Word yo actualizaba fielmente, razonando que como escritor tenía sentido que me gastara una cierta cantidad de dinero en herramientas.

En algún momento, a mediados de los ochenta, traté de abrir uno de mis antiguos documentos Word que databa más o menos de 1985 usando la versión entonces vigente de Word: 6.0. No funcionó. Word 6.0 no reconocía un documento creado por una versión anterior de sí mismo. Abriéndolo como archivo de texto, pude recuperar las secuencias de letras que constituían el texto del documento. Mis palabras seguían allí. Pero el formato parecía pasado por un colador —las palabras que yo había escrito iban interrumpidas por cuadros rectangulares vacíos y basura.

Ahora bien, en el contexto de una empresa (el principal mercado de Word) este tipo de cosa sólo es una molestia —uno de los problemas rutinarios que comporta usar ordenadores—. Es fácil comprar programitas de conversión de archivos que se ocupan de este problemas. Pero si eres un escritor, cuyo oficio son las palabras, cuya identidad profesional es un corpus de documentos escritos, este tipo de cosa resulta extremadamente desasosegante. En mi tipo de trabajo hay muy pocos presupuestos establecidos, pero uno de ellos es que una vez escribes una palabra, queda escrita y no puede *desescribirse.* La tinta mancha el papel, el escoplo corta la piedra, el estilo marca la arcilla y algo ha sucedido irrevocablemente (mi cuñado es un teólogo que lee tablillas en cuneiforme de hace 3250 años —puede reconocer la escritura de algunos escribas individuales, e identificarlos por su nombre—). Pero el software de procesamiento de textos —particularmente el tipo que emplea formatos de archivo especiales y complejos— tiene el sobrenatural poder de *desescribir* las cosas. Un pequeño cambio en los formatos de archivo, o unos pocos bits revueltos, y la producción literaria de meses o años puede dejar de existir.

Esto era técnicamente un fallo de la aplicación (Word 6.0 para Macintosh), no del sistema operativo (MacOS 7 punto algo), así que el blanco inicial de mi enfado fueron los responsables de Word. Por otro lado, yo podía haber elegido la opción *guardar como texto* en Word y haber guardado todos mis documentos como simples telegramas, y este problema no habría surgido. Por el contrario, me había dejado seducir por todas esas vistosas opciones de formateo que ni siquiera existían hasta que las GUIs aparecieron y las hicieron practicables. Había caído en el hábito de usarlas para que mis documentos tuvieran un bonito aspecto (tal vez más bonito del que merecían; todos esos viejos documentos en los disquetes resultaron ser más o menos una porquería). Ahora estaba pagando el precio de mi autoindulgencia. La tecnología había avanzado y hallado maneras de que mis documentos parecieran aún más bonitos, y la consecuencia de ello era que todos los viejos y feos documentos habían dejado de existir.

Era —si me disculpan una pequeña y extraña fantasía durante un momento— como si hubiera ido a alojarme en un hotel exquisitamente diseñado, poniéndome en manos de los antiguos maestros de la *interfaz sensorial*, me hubiera sentado en mi habitación y hubiese escrito una historia con un bolígrafo en papel amarillo y, al volver de la cena, me hubiese encontrado con que la doncella se había llevado mi trabajo y en su lugar había dejado una pluma y una resma de pergamino —explicando que la habitación tenía mucho mejor aspecto así y era todo parte de una actualización rutinaria—. Pero escritas en aquellas hojas de papel, en impecable ortografía, habría largas secuencias de palabras escogidas al azar del diccionario. Espantoso, cierto, pero legalmente no podría demandar a la dirección, porque al alojarme en ese hotel había dado mi consentimiento para ello. Había entregado mis credenciales de morlock y me había convertido en un eloi.

Linux

A FINALES DE LOS AÑOS OCHENTA y principios de los noventa me pasé un montón de tiempo programando para Macintosh, y al final decidí pagar varios cientos de dólares por un producto de Apple llamado el Macintosh Programmer's Workshop, o MPW. MPW tenía competidores, pero era incuestionablemente el mejor sistema de desarrollo de software para el Mac. Los propios ingenieros de Apple solían escribir código Macintosh con él. Puesto que MacOS era con mucho el sistema operativo más desarrollado tecnológicamente en aquel momento, y puesto que Linux ni siquiera existía todavía, y puesto que este era el programa que usaba de hecho el equipo de ingenieros creativos de elite de Apple, tenía grandes expectativas. Venía en una pila de disquetes de un pie de alto, así que tuve tiempo para que mi emoción creciera durante el interminable proceso de instalación. La primera vez que inicié MPW, probablemente me esperaba algún tipo de quisquilloso muestrario multimedia. Por el contrario, era austero, casi hasta el punto de resultar intimidatorio. Era una ventana desplazable en la que se podía escribir texto simple, sin formato. El sistema interpretaba entonces esas líneas de texto como comandos, y trataba de ejecutarlos.

Era, en otras palabras, un teletipo de vidrio ejecutando una interfaz de línea de comandos. Venía con todo tipo de comandos crípticos pero potentes, que podían invocarse tecleando sus nombres, y que sólo gradualmente aprendí a usar. Sólo algunos años después, cuando empecé a enredar con Unix, comprendí que la interfaz de línea de comandos encarnada en MPW era una recreación de Unix.

En otras palabras, lo primero que habían hecho los hackers de Apple cuando consiguieron que MacOS fuese funcional —posiblemente *antes* de que lo fuera— había sido recrear la interfaz de Unix, para poder hacer algún trabajo útil. En aquel momento, mi mente no daba para entender esto pero, en lo que concernía a los hackers de Apple, la muy pregonada Interfaz Gráfica de Usuario del Mac era un impedimento, algo a evitar incluso antes de que el aparatito saliera siquiera al mercado.

Incluso antes de que mi PowerBook fallara y destruyera mi gran archivo en julio de 1995, había habido señales de peligro. Un viejo amigo mío, que crea y lleva compañías de alta tecnología en Boston, había desarrollado un producto comercial usando el Macintosh. Básicamente el Mac funcionaba como terminal gráfico de alto rendimiento, escogido por su bonita interfaz de usuario, que daba al usuario acceso a una gran base de datos de información gráfica almacenada en una red de ordenadores mucho más potentes, pero de uso menos orientado al usuario. Este tipo era la segunda persona que llamó mi atención sobre el Macintosh, por cierto, y a mediados de los ochenta compartíamos la emoción de ser expertos en alta tecnología y de usar la tecnología Apple en un mundo de tontainas usuarios de DOS. Las primeras versiones del sistema de mi amigo funcionaron bien pero, cuando se unieron varias máquinas a la red, empezaron a producirse misteriosos fallos; a veces todo el sistema sencillamente se detenía. Era uno de esos fallos que no podían reproducirse fácilmente. Finalmente se dieron cuenta de que estos errores del sistema se producían cada vez que un usuario, buscando algo en los menús, mantenía el botón del ratón pulsado durante más de dos segundos.

Básicamente, el MacOS sólo podía hacer una cosa por vez. Desplegar un menú en la pantalla es una cosa. Así que cuando de desplegaba un menú, el Macintosh no era capaz de hacer nada más hasta que el usuario indeciso soltaba el botón.

Esto no es algo tan terrible en una máquina de un solo usuario y un solo proceso (aunque es una cosa bastante mala), pero es un desastre en una máquina que forma parte de una red, porque formar parte de una red conlleva algún tipo de interacción continua de bajo nivel con otras máquinas. Al no responder a la red, el Mac provocó un fallo en todo el sistema de red.

Para trabajar con otros ordenadores, y con diferentes tipos de hardware, un sistema operativo ha de ser incomparablemente más potente que MS-DOS y que el MacOS original. El único modo de conectarse con Internet que merece la pena tomarse en serio es PPP, el Protocolo Punto-a-Punto, que (no importan los detalles) convierte a su ordenador —temporalmente— en un miembro de pleno derecho de la Internet Global, con su propia dirección única, y diversos privilegios, poderes y responsabilidades. Técnicamente, significa que su máquina ejecuta el protocolo TCP/IP, que, brevemente, se basa en el envío de paquetes de datos, en ningún orden en particular, y en momentos impredecibles, siguiendo un inteligente y elegante conjunto de reglas. Pero enviar un paquete de datos es una cosa, así que un sistema operativo que sólo pueda hacer una cosa por vez no puede formar parte de Internet y hacer otra cosa simultáneamente. Cuando se inventó TCP/IP, ejecutarlo era un honor reservado a los Ordenadores Serios —*mainframes* y miniordenadores de alta potencia usados en contextos técnicos y comerciales—, así que el protocolo está diseñado con el presupuesto de que cada ordenador que lo usa es una máquina seria, capaz de hacer muchas cosas a la vez. Hablando pronto y mal, una máquina Unix. Ni MacOS ni MS-DOS se construyeron originalmente pensando en eso, así que cuando Internet se puso caliente, hubo que llevar a cabo cambios radicales.

Cuando mi PowerBook me partió el corazón y cuando Word dejó de reconocer mis antiguos archivos, me pasé a Unix. La alternativa obvia a MacOS habría sido Windows. En realidad yo no tenía nada *contra* Microsoft, ni contra Windows. Pero ya resultaba bastante obvio que los antiguos sistemas operativos de PC estaban funcionando más allá de sus posibilidades y lo mostraban, así que tal vez era mejor evitarlos hasta que hubieran aprendido a caminar y mascar chicle al mismo tiempo.

El cambio tuvo lugar un día particular en el verano de 1995. Llevaba un par de semanas en San Francisco, usando mi PowerBook para trabajar en un documento. El documento era demasiado grande para caber en un solo disquete, así que no había realizado ninguna copia desde que salí de casa. El PowerBook se colgó y borró todo el archivo.

Sucedió justo cuando salía a visitar una compañía llamada Electric Communities, que en aquella época estaba en Los Altos. Me llevé mi PowerBook conmigo. Mis amigos en Electric Communities eran usuarios de Mac que tenían todo tipo de software para recuperar archivos y datos perdidos por fallos de disco, y estaba seguro de que podría recobrar la mayor parte del archivo.

Resultó que dos utilidades diferentes para la recuperación de datos por fallo del Mac fueron incapaces de hallar rastro alguno de que mi archivo había existido alguna vez. Estaba completa y sistemáticamente borrado. Peinamos el disco duro bloque a bloque, y encontramos fragmentos disjuntos de incontables archivos antiguos, descartados y olvidados, pero nada de lo que yo quería. El trasquilón metafórico fue especialmente brutal ese día. Fue algo así como ver cómo la chica de la que llevas diez años enamorado se mata en un accidente de tráfico, y luego estar presente en su autopsia, para darte cuenta de que bajo la ropa y el maquillaje era sólo carne y hueso.

Debí de vagar por los pasillos de la Electric Communities en una especie de fuga jungiana primaria, porque en aquel momento sucedieron tres cosas extrañamente sincrónicas.

1. Randy Farmer, cofundador de la compañía, llegó en una visita rápida con su familia (estaba recuperándose de una operación en la espalda en aquel momento). Traía noticias candentes: «Hoy han masterizado Windows 95.» Quería decir que el nuevo sistema operativo de Microsoft había sido colocado ese mismo día en

un disco compacto especial conocido como el «master dorado», que se usaría para sacar trillones de copias, preparando su estruendoso lanzamiento unas pocas semanas después. Esta noticia fue recibida con fastidio por los empleados de Electric Communities, incluyendo uno que tenía la puerta del despacho llena de las viñetas y novedades habituales, por ejemplo.

2. Un cómic de *Dilbert* en el que Dilbert, el sufridor ingeniero de software de una empresa, se encuentra con un hombre barbudo y peludo de cierta edad, algo parecido a Santa Claus, pero más siniestro, y con cierta sorna. Dilbert reconoce a este hombre, por su apariencia y efecto, como un hacker de Unix, y reacciona con una cierta mezcla de nerviosismo, respeto y hostilidad. Dilbert realiza endebles intentos por meterse con el perturbador extraño durante un par de viñetas; el hacker de Unix le escucha con una especie de irritante calma beatífica y luego, en la última viñeta, mete la mano en el bolsillo. «Ten una moneda, chico», dice, «y ve a comprarte un ordenador de verdad».
3. El dueño de la puerta y del cómic era un tal Doug Barnes. Era sabido que Barnes tenía ciertas opiniones heréticas sobre el tema de los sistemas operativos. A diferencia de la mayoría de los *techies* del Área de la Bahía, que adoraban el Macintosh, considerando que era la máquina del verdadero hacker, a Barnes le gustaba señalar que el Mac, con su arquitectura herméticamente sellada, era de hecho hostil a los hackers, a quienes les gusta enredar y para los que la apertura es un dogma. En cambio, las máquinas compatibles con IBM, que pueden montarse y desmontarse fácilmente, eran mucho más hackeables.

Así que cuando volví a casa empecé a enredar con Linux, que es una de las muchísimas distintas implementaciones concretas del ideal abstracto y platónico llamado Unix. No me apetecía cambiarme a un nuevo sistema operativo, porque mis tarjetas de crédito todavía echaban humo después de todo el dinero que me había gastado en hardware para el Mac en el curso de los años. Pero la gran virtud de Linux era, y es, que podía ejecutarse en exactamente el mismo tipo de hardware que el sistema operativo de Microsoft —es decir, el hardware más barato que existe—. Como para demostrar que esto era una gran idea, una o dos semanas después de volver a casa pude hacerme con un ordenador entonces bastante bueno (un 486 a 33Mhz) gratis, porque conocía a un tipo que trabajaba en una oficina en la que estaban tirándolos. Una vez llegué a casa, le quité la funda, metí las manos y empecé a cambiar las tarjetas. Si algo no funcionaba, iba a una tienda de ordenadores de segunda mano, buscaba en una cesta llena de componentes y compraba una nueva tarjeta por unos cuantos dólares.

La disponibilidad de todo este hardware barato pero efectivo fue una consecuencia involuntaria de decisiones que se habían tomado hacía más de una década en IBM y Microsoft. Cuando salió Windows y llevó la GUI a un mercado mucho más amplio, el régimen del hardware cambió: el precio de las tarjetas de vídeo en color y los monitores de alta resolución empezó a caer, y sigue cayendo. Este enfoque del hardware gratis-para-todos significó que Windows era inevitablemente torparrón comparado con MacOS. Pero la GUI llevó la informática a un público tan vasto que el volumen aumentó muchísimo y los precios se vinieron abajo. Mientras tanto Apple, que tanto deseaba un sistema operativo limpio e integrado, con el vídeo totalmente integrado en el hardware de procesamiento, había quedado muy por detrás en la cuota de mercado, en parte al menos porque su precioso hardware costaba tanto.

Pero el precio que tuvimos que pagar los dueños de un Mac por una estética y un diseño superiores no fue meramente financiero. Había un precio cultural también, debido al hecho de que no podíamos abrir el ordenador y enredar con él. Doug Barnes tenía razón. Apple, pese a su reputación de ser la opción de los hackers creativos y contestatarios, había creado de hecho una máquina que desalentaba el hackeo, mientras que Microsoft, considerada una perezosa tecnológica y una plagiaria, había creado un vasto bazar de componentes sin orden ni concierto: una sopa primordial que había acabado autoorganizándose en Linux.

El «hole hawg» de los sistemas operativos

UNIX SIEMPRE HA ESTADO PULULANDO provocativamente en el trasfondo de las guerras de los sistemas operativos, como el Ejército ruso. La mayor parte de la gente sólo conoce su reputación, y su reputación, como sugiere el cómic de *Dilbert,* es mixta. Pero todo el mundo parece estar de acuerdo en que sólo con que se planteara su actuación en serio y dejara de cederle enormes extensiones de ricos terrenos agrícolas y cientos de miles de prisioneros de guerra a los invasores, los aplastaría, a ellos y a cualquier otra oposición.

Resulta difícil explicar cómo se ha ganado Unix este respeto sin meterse en horrorosos detalles técnicos. Tal vez el meollo pueda explicarse contando una historia sobre taladradoras.

«Hole Hawg» es una gama de máquinas de taladrar fabricadas por la Compañía de Herramientas Milwaukee. Si observan el escaparate de una típica ferretería, pueden encontrar taladros de Milwaukee más pequeños, pero no el «hole hawg», que es demasiado potente y caro para usuarios domésticos. El «hole hawg» no tiene el diseño en forma de pistola del barato taladro doméstico. Es un cubo de metal sólido con un mango que sale por un lado y una protuberancia en otro. El cubo contiene un motor eléctrico desconcertantemente potente. Se puede sostener el mango y apretar el gatillo con el índice pero, a menos que se sea excepcionalmente fuerte, no se puede controlar el peso del «hole hawg» con una mano: hay que sujetarlo con ambas manos. Para compensar el contratorque del «hole hawg», se usa un mango adicional (viene incluido), que se atornilla en uno u otro lado del cubo de hierro, dependiendo de si se usa la mano izquierda o la derecha para apretar el gatillo. Este mango no es esbelto y ergonómico como lo sería en un taladro doméstico. Es simplemente un pedazo de tubería galvanizada normal de un pie de largo, con un agujero en un extremo, con un mango de goma negra en el otro. Si lo pierdes, simplemente vas a la tienda de fontanería local y compras otro pedazo de tubería.

Durante los ochenta hice algo de albañilería. Un día, otro obrero apoyó una escalera contra la fachada del edificio que estábamos construyendo, subió al segundo piso y uso el «hole hawg» para hacer un agujero en el muro exterior. En algún momento, la broca se atascó en el muro. El «hole hawg», siguiendo su único imperativo, siguió funcionando. Giró el cuerpo del obrero como una muñeca de trapo, haciendo que tirara la escalera. Por suerte, se mantuvo agarrado al «hole hawg», que permaneció encajado en el muro, y simplemente colgó de él y pidió ayuda hasta que vino alguien y puso de nuevo la escalera.

Yo mismo usé un «hole hawg» para hacer muchos agujeros a través de remaches, lo cual hice como una picadora pica coliflor. También la usé para hacer unos cuantos agujeros de seis pulgadas de diámetro en un viejo techo de escayola. Introduje una nueva sierra, subí al segundo piso, metí la mano por entre las recientes juntas del suelo y empecé a cortar el techo del primer piso. Allí donde mi broca doméstica las había pasado canutas para hacer girar el enorme hierro, y se había detenido a la menor obstrucción, la «hole hawg» rotaba con la estúpida consistencia de un planeta giratorio. Cuando la sierra ganó velocidad, el «hole hawg» giró sobre sí mismo y me hizo girar a mí también, aplastando una de mis manos entre el mango de acero y una junta, produciéndome algunas laceraciones, cada una rodeada por una amplia corona de carne magullada. También dobló la propia sierra, aunque no tanto como para que no pudiera volver a usarla. Tras unos pocos encontronazos parecidos, cada vez que tenía que usar el «hole hawg» mi corazón empezaba a latir con terror atávico.

Pero nunca le eché la culpa al «hole hawg»: me eché la culpa a mí mismo. El «hole hawg» es peligroso porque hace exactamente lo que se le pide que haga. No se ve constreñido por las limitaciones físicas inherentes a un taladro barato, ni por los cierres de seguridad que puede incluir un fabricante temeroso de las responsabilidades penales en un producto doméstico. El peligro no está en la máquina misma, sino en la incapacidad del usuario de contemplar todas las consecuencias de las instrucciones que le da.

Una herramienta más pequeña también es peligrosa, pero por razones completamente distintas: trata de dar lo que se le pide, y falla de un modo que resulta impredecible y casi siempre indeseable. Pero el «hole hawg» es como el genio de las antiguos cuentos de hadas, que lleva a cabo las instrucciones de su amo literalmente, con precisión y un poder ilimitado, a menudo con desastrosas consecuencias imprevistas.

Antes del «hole hawg», solía examinar el surtido de taladros en las ferreterías de un modo que consideraba sensato, desechando los modelos más pequeños y levantando los grandes y caros apreciativamente, deseando poder permitirme una de aquellas bellezas. Ahora las miro a todas con tal desdén que ni siquiera considero que sean taladros de verdad —son simplemente juguetes diseñados para explotar las tendencias delirantes de urbanitas que quieren creer que han comprado una herramienta de verdad—. Sus estuches de plástico, cuidadosamente diseñados y verificados con grupos-diana para transmitir una sensación de solidez y potencia, me parecen asquerosamente frágiles y baratos, y me avergüenzo de haber picado alguno vez y comprado tales menudencias.

No resulta difícil imaginar qué aspecto tendría el mundo para alguien que hubiese sido criado por constructores y que nunca hubiese usado más taladro que el «hole hawg». Tal persona, al ver el mejor y más caro taladro de una ferretería, ni siquiera lo reconocería como tal. Por el contrario, puede que lo confundiera con un juguete de niños, o con una especie de destornillador motorizado. Si el vendedor o confuso urbanita se refiriera a ello como un taladro, se reiría y les diría que estaban equivocados —sencillamente, se habían confundido con la terminología—. Su interlocutor se marcharía irritado, y probablemente bastante a la defensiva en lo tocante a su sótano lleno de vistosas herramientas baratas, peligrosas y coloridas.

Unix es el «hole hawg» de los sistemas operativos,[1] y los hackers de Unix, como Doug Barnes y el tipo del cómic de *Dilbert* y muchas otras personas que pueblan Silicon Valley, son como hijos de constructores que se criaron usando sólo taladros industriales «hole hawg». Podrían usar los sistemas operativos de Apple/Microsoft para escribir cartas, jugar a videojuegos o llevar las cuentas, pero no consiguen tomarse esos sistemas operativos en serio.

---

[1]Nathan Myhrvold, [director técnico] de Microsoft, ha establecido su pleistocénica elección, ha tomado el reto y ha contraatacado con una mordaz analogía de taladradoras, de propia cosecha, que giran en sentido contrario al que lo hacía la nuestra. Su analogía de la taladradora es probablemente, al final, mejor que la mía. No la presentaré aquí porque un duelo público sobre analogías de taladradoras presentaría un espectáculo ridículo e indigno. He aquí algunos extractos:

> «Existe un estúpido romanticismo de que, cuanto más primitivo es el instrumento y más habilidades requiere para el operador, debe de alguna manera ser más poderoso. Esto normalmente es una cagada…»

> «Una razón fundamental por la que Linux se ha convertido en algo interesante es porque Internet ha causado temporalmente una fase de retroceso en la que de repente los programas interesantes son muy poco sofisticados. Apache, o un servidor NNTP, es un software muy simple que no le exige demasiado a un sistema operativo. Lo mismo ocurre con muchas tareas orientadas a la Web. Linux está bien para esto.»

La tradición oral

UNIX ES DIFÍCIL DE APRENDER. El proceso de aprenderlo tiene múltiples pequeñas epifanías. Lo típico es estar a punto de inventar una herramienta o utilidad necesaria cuando te das cuenta de que alguien ya la inventó, y la incorporó, y eso explica algún extraño archivo o directorio que viste pero que nunca comprendiste realmente antes.

Por ejemplo, hay un comando (un pequeño programa, parte del sistema operativo) llamado `whoami`, que permite preguntarle al ordenador quién cree que eres —en una máquina Unix, siempre entras bajo un nombre, ¡posiblemente, incluso el tuyo!—: con qué archivos puedes trabajar o qué software puedes usar, depende de tu identidad. Cuando empecé a usar Linux, tenía una máquina sin conectar a la red en mi sótano, con sólo una cuenta de usuario, así que cuando descubrí el comando `whoami` me pareció ridículo. Pero cuando entras como una persona, puedes usar temporalmente un pseudónimo para acceder a diferentes archivos. Si tu ordenador está conectado a Internet, puedes entrar en otros ordenadores siempre que tengas un nombre de usuario y una contraseña. En ese momento la máquina distante no difiere en nada de la que tienes justo delante de ti. Estos cambios de identidad y localización pueden anidarse unos dentro de otros, con muchas capas, incluso si no se está haciendo nada criminal. Cuando te olvidas de quién eres y dónde estás, el comando `whoami` es indispensable. Yo lo uso todo el tiempo.

Los sistemas de archivos de las máquinas Unix tienen todos la misma estructura general. En los sistemas operativos endebles, se pueden crear directorios (carpetas) y ponerles nombres como «Frodo» o «Mis Cosas» y ponerlos más o menos donde a uno le dé la gana. Pero en Unix el nivel más alto —la raíz— del sistema de archivos siempre es designado por el carácter único «/» y siempre contiene el mismo conjunto de directorios de nivel superior:

```
/usr /etc /var /bin /proc /boot /home /root
/sbin /dev /lib /tmp
```

y cada uno de estos directorios típicamente tiene su propia estructura distintiva de subdirectorios. Fíjense en el uso obsesivo de abreviaturas y en cómo se evitan las mayúsculas; se trata de un sistema inventado por gente a la que el desorden repetitivo por estrés es lo que la silicosis a los mineros. Los nombres largos se desgastan hasta convertirse en colillas de tres letras, como guijarros pulidos por el río.

Este no es el lugar para tratar de explicar por qué existe cada uno de los anteriores directorios, y qué contiene. Al principio todo parece oscuro; peor, parece deliberadamente oscuro. Cuando empecé a usar Linux, estaba acostumbrado a poder crear directorios donde quisiera y a darles los nombres que me apeteciera. Con Unix se puede hacer eso, por supuesto (eres libre de hacer lo que quieras), pero a medida que se adquiere experiencia con el sistema se llega a comprender que los directorios listados antes se crearon por las mejores razones y que la vida de uno será mucho más fácil si se sigue el juego (dentro de `/home`, por cierto, uno tiene libertad ilimitada).

Cuando este tipo de cosa ha sucedido varios cientos o miles de veces, el hacker comprende por qué Unix es como es, y está de acuerdo en que no podría ser lo mismo de ningún otro modo. Es este tipo de aculturación lo que les da a los hackers de Unix su confianza en el sistema, y la actitud de reposada, inamovible, irritante superioridad que reflejaba el cómic de *Dilbert.* Tanto Windows 95 como MacOS son productos diseñados por ingenieros al servicio de compañías específicas. Unix, en cambio, no es tanto un producto como una historia oral escrupulosamente compilada de la subcultura hacker. Es nuestra épica de Gilgamesh.

Lo que hacía que las antiguas épicas como la de Gilgamesh resultaran tan poderosas y tan longevas se debía a que eran cuerpos vivientes de narrativa que mucha gente se sabía de memoria, y contaban una y otra vez, añadiendo sus propios adornos cuando les apetecía. Los malos adornos no gustaban, los buenos eran retomados por otras personas, pulidos, mejorados, y con el tiempo se incorporaban a la historia. De igual modo, Unix es conocido, amado y comprendido por tantos hackers, que puede recrearse a partir de cero cuando alguien lo necesita. Esto resulta muy difícil de entender para la gente acostumbrada a pensar en los sistemas operativos como cosas que tienen que ser compradas.

Muchos hackers han lanzado reimplementaciones más o menos exitosas del ideal de Unix. Cada una lleva nuevos adornos. Algunos mueren rápidamente, otros se funden con innovaciones semejantes y paralelas creadas por diferentes hackers que atacaban el mismo problema, otros se adoptan e incorporan a la épica. Así, Unix ha crecido lentamente alrededor de un núcleo simple y ha adquirido una complejidad y asimetría a su alrededor que es orgánica, como las raíces de un árbol, o las ramificaciones de una arteria coronaria. Comprenderlo se parece más a la anatomía que a la física.

Durante al menos un año, antes de mi adopción de Linux, había oído hablar de él. Personas creíbles y bien informadas me decían que unos cuantos hackers habían construido una implementación de Unix que podía descargarse gratuitamente de Internet. Durante mucho tiempo no pude tomarme la idea en serio. Era como oír rumores de que un grupo de entusiastas de las maquetas de cohetes habían creado un Saturno V completamente funcional intercambiando planos por la Red y enviándose mutuamente válvulas y alerones.

Pero es cierto. Normalmente el mérito de Linux se atribuye a su tocayo humano, un tal Linus Torvalds, un finlandés que inició el asunto en 1991, cuando usó algunas de las herramientas de GNU para escribir el principio de un núcleo Unix que pudiera ejecutarse en hardware compatible con PC. Y ciertamente Torvalds merece todo el crédito que se le ha dado, y mucho más. Pero no podría haberlo conseguido él solo, como tampoco habría podido Richard Stallman. Para escribir el código, Torvalds necesitó tener herramientas de desarrollo baratas pero potentes, y obtuvo éstas del proyecto GNU de Stallman.

Y tenía un hardware barato en que escribir ese código. El hardware barato es algo mucho más difícil de lograr que el software barato: una sola persona (Stallman) puede escribir software y colgarlo en la Red de modo gratuito, pero para fabricar hardware hay que tener toda una infraestructura industrial, lo cual no es barato ni de lejos. Realmente, el único modo de hacer que el hardware resulte barato es sacar un número increíble de copias, de tal modo que el precio por unidad acabe cayendo. Por las razones ya explicadas, Apple no tiene ninguna gana de ver cómo cae el precio del hardware. La única razón por la que Torvalds tenía hardware barato era Microsoft.

Microsoft se negó a entrar en el negocio del hardware, insistiendo en hacer que su software pudiera ejecutarse en hardware que cualquiera podía fabricar, y creó así las condiciones de mercado que permitieron que los precios del hardware cayeran en picado. Al tratar de comprender el fenómeno Linux, pues, tenemos que contemplar no a un único innovador, sino una especie de extraña Trinidad: Linus Torvalds, Richard Stallman y Bill Gates. Elimínese cualquiera de estos tres y Linux no existiría.

Shock de sistema operativo

Los jóvenes estadounidenses que dejan su gran país homogéneo y visitan otra parte del mundo típicamente sufren varios grados de shock cultural: primero, inmenso asombro. Luego, un acercamiento tentativo a las costumbres, cocina, sistemas públicos de circulación y retretes del nuevo país, lo cual lleva a un breve periodo de confianza fatua en que son expertos instantáneos en el nuevo país. A medida que continúa la visita, empieza la morriña y el viajero empieza a apreciar, por primera vez, cuánto daba por sentado en casa. Al mismo tiempo, empieza a resultar obvio que las propias culturas y tradiciones son esencialmente arbitrarias: conducir por la derecha, por ejemplo. Cuando el viajero vuelve a casa y hace balance de la experiencia, puede haber aprendido bastante más sobre los Estados Unidos que sobre el país que fueron a visitar.

Por los mismos motivos, merece la pena probar Linux. Ciertamente, es un país extraño, pero no hay por qué vivir ahí; una breve estancia basta para experimentar el gusto del lugar y —lo que es más importante— revelar todo lo que se da por sentado, y todo lo que se podría haber hecho de modo distinto, en Windows o MacOS.

No se puede probar sin instalarlo. Con cualquier otro sistema operativo, instalarlo sería una transacción sencilla: a cambio de dinero, una compañía te daría un CD-ROM, y ya está. Pero hay un montón de cosas subsumidas bajo ese tipo de transacción, y hay que verlas y diferenciarlas.

En Estados Unidos nos gustan los tratos simples y las transacciones sin complicaciones. Si vas a Egipto y, pongamos, tomas un taxi en algún sitio, te conviertes en parte de la vida del taxista; se niega a aceptar tu dinero porque rebajaría vuestra amistad, te sigue por la ciudad y llora como un crío cuando te metes en el taxi de otro. Acabas por conocer a sus hijos en algún momento y tienes que ingeniártelas para hallar algún modo de compensarle sin insultar su honor. Es agotador. A veces simplemente quieres tomar un taxi como en Manhattan.

Pero para tener un sistema de estilo estadounidense, en el que puedes salir, parar un taxi y ya está, tiene que haber todo un aparato de licencias, inspectores, comisiones, etc., lo cual está muy bien siempre que los taxis sean baratos y siempre que puedas llamar a uno. Cuando el sistema no funciona de alguna manera, resulta misterioso y enervante y convierte a personas habitualmente razonables en teóricos de la conspiración. Pero cuando el sistema egipcio se viene abajo, se viene abajo de forma transparente. No puedes tomar un taxi, pero aparecerá el sobrino del taxista, a pie, para explicarte el problema y disculparse.

Microsoft y Apple hacen las cosas al estilo de Manhattan, con una vasta complejidad oculta tras el muro de la interfaz. Linux hace las cosas al estilo de Egipto, con una vasta complejidad desperdigada por todo el paisaje. Si acabas de llegar de Manhattan, tu primer impulso será llevarte las manos a la cabeza diciendo: «¡Esto es de locos! ¿Por qué narices no os comportáis como es debido?» Pero esto no te granjearía más amigos en Linuxlandia que en Egipto.

Se puede extraer Linux del aire mismo, por así decir, descargando los archivos adecuados y poniéndolos en los lugares adecuados, pero posiblemente no más de unos pocos cientos de personas en el mundo podrían crear un sistema Linux funcional de ese modo. Lo que realmente se necesita es una distribución de Linux, lo cual quiere decir un conjunto preempaquetado de archivos. Pero las distribuciones son una cosa distinta de Linux *per se.*

Linux *per se* no es un conjunto específico de unos y ceros, sino una subcultura autoorganizada de la Red. El resultado final de sus elucubraciones colectivas es un vasto cuerpo de código fuente, casi todo escrito en C (el lenguaje de programación dominante). El código fuente es sencillamente un programa de ordenador escrito y editado por algún hacker. Si está en C, el nombre del archivo probablemente llevará `.c` o `.cpp` al final, dependiendo del dialecto empleado; si está en otro lenguaje llevará otro sufijo. A menudo, este tipo de archivos pueden encontrarse en un directorio con el nombre `/src`, que es la abreviatura hebraica del hacker para *source*, «fuente».

Los archivos fuente son inútiles para el ordenador, y de poco interés para la mayoría de usuarios, pero tienen una enorme significación cultural y política, porque Microsoft y Apple los mantienen en secreto, mientras que Linux los hace públicos. Son las joyas de la familia. Son el tipo de cosa que en los thrillers de Hollywood se usa como McGuffin: el núcleo de la bomba de plutonio, los planos de alto secreto, el maletín lleno de documentos financieros, el microfilm. Si los archivos fuente de Windows o MacOS se hicieran públicos en la Red, esos sistemas operativos se volverían gratuitos, como Linux —sólo que no tan buenos, porque no habría nadie para arreglar los fallos y responder a las preguntas—. Linux es software de fuente abierta,[1] lo cual sencillamente quiere decir que cualquiera puede obtener copias de sus archivos de código fuente.

Un ordenador no necesita código fuente más de lo que lo necesita usted: necesita «código objeto». Los archivos de código objeto típicamente llevan el sufijo `.o` y son ilegibles para todo el mundo salvo unos pocos humanos altamente extraños, porque consisten en unos y ceros. En consecuencia, este tipo de archivo normalmente aparece en un directorio con el nombre `/bin`, por binario.

Los archivos fuente son sencillamente archivos de texto ASCII. ASCII denota un modo particular de codificar las letras en patrones de bits. En un archivo ASCII, cada carácter

[1] *Open Source software* es otro modo de denominar al software libre: esto es, aquel que puede ser usado, copiado, modificado y redistribuido sin restricciones. [*N. del E.*]

tiene ocho bits para él solito. Esto crea un alfabeto potencial de 256 caracteres distintos, dado que ocho dígitos binarios pueden formar ese número de patrones únicos. En la práctica, por supuesto, nos limitamos a las letras y dígitos familiares. Los patrones de bits empleados para representar esas letras y dígitos son los mismos que se introducían físicamente agujereando la cinta de papel de mi teletipo del instituto, que a su vez eran los mismos que había usado antes la industria telegráfica durante décadas. Los archivos de texto ASCII, en otras palabras, son telegramas, y como tales no tienen adornos tipográficos. Pero por eso mismo son eternos, porque el código nunca cambia, y universales, porque todo el software de edición y procesamiento de textos existente conoce este código.

Por tanto, se puede usar cualquier software para crear, editar o leer archivos de código fuente. Los archivos de código objeto, entonces, son creados a partir de estos archivos fuente por un software llamado *compilador,* y son convertidos en una aplicación funcional por otro software llamado *enlazador.*

La tríada de editor, compilador y enlazador, tomados juntos, constituye el núcleo de un sistema de desarrollo de software. Ahora es posible gastarse un montón de dinero en sistemas de desarrollo envueltos en plástico, con preciosas interfaces gráficas de usuario y diversas mejoras ergonómicas. En algunos casos puede que hasta resulte un modo bueno y razonable de gastar el dinero. Pero en este lado de la carretera, por así decir, el mejor software es a menudo el gratuito. Editor, compilador y enlazador son a los hackers lo que ponies, estribos y arcos y flechas eran a los mongoles. Los hackers viven a caballo, y hackean sus propias herramientas incluso mientras las usan para crear nuevas aplicaciones. Resulta bastante inconcebible que herramientas superiores de hacking pudieran haber sido creadas en una hoja en blanco por ingenieros informáticos. Incluso aunque fueran los ingenieros más inteligentes del mundo, se verían sencillamente superados.

En el mundo de GNU/Linux hay dos grandes programas de edición de textos: el minimalista `vi` (conocido en algunas implementaciones como `elvis`) y el maximalista `emacs`. Yo uso `emacs`, que puede considerarse un procesador de textos termonuclear. Fue creado por Richard Stallman, y con esto ya está todo dicho. Está escrito en LISP, que es el único lenguaje de ordenador que es hermoso. Es colosal, y sin embargo sólo edita archivos de texto ASCII, lo cual significa: nada de fuentes, nada de negrita, nada de subrayado. En otras palabras, las horas que dedicaron los ingenieros, en el caso de Windows, a cosas como la fusión de correo y la capacidad de incrustar películas de dos horas en memorandos de empresa, se dedicaron, en el caso de `emacs`, con intensidad maníaca al engañosamente simple problema de editar texto. Si eres un escritor profesional —esto es, si otra persona está siendo pagada para preocuparse de cómo se formatean e imprimen tus palabras— `emacs` hace sombra a cualquier otro software de edición más o menos del mismo modo que el sol de mediodía hace sombra a las estrellas. No sólo es mayor y más luminoso: sencillamente hace que todo lo demás se desvanezca. Para el formateo y la impresión de la página se puede usar TeX: un vasto corpus de ciencia tipográfica escrito en C y también disponible en la Red gratuitamente.[2]

Podría decir un montón de cosas sobre `emacs` y TeX, pero ahora mismo trato de contar una historia acerca de cómo instalar de hecho Linux en el ordenador. El enfoque de pura supervivencia sería descargarse un editor como `emacs` y las herramientas GNU —el compilador y el enlazador– que son tan pulidas y elegantes como `emacs`. Equipado con esto, uno ya puede empezar a descargar archivos de código fuente ASCII (`/src`) y a compilarlos en archivos de código objeto binario (`/bin`) ejecutables por el ordenador. Pero para llegar siquiera a este punto —para ejecutar `emacs`, por ejemplo— hay que tener Linux instalado y funcionando en el ordenador. E incluso un sistema operativo mínimo

[2]Esta versión castellana de la obra que tiene el lector en sus manos ha sido maquetada y compuesta íntegramente con LaTeX —un lenguaje estructurado construido a partir de TeX— y con el editor GNU `Emacs`. [*N. del E.*]

de Linux requiere miles de archivos binarios actuando en concierto, dispuestos y vinculados para que lo hagan.

Por tanto, diversas entidades se han ocupado de crear distribuciones de Linux. Por extender algo más la analogía con Egipto, estas entidades se parecen algo a los guías turísticos que te reciben en el aeropuerto, hablan tu idioma y te ayudan con el shock cultural inicial. Si uno es egipcio, claro, se puede ver del otro modo; los guías turísticos existen para evitar que los brutos extranjeros se metan en las mezquitas haciendo las mismas preguntas una y otra y otra vez.[3]

Algunos de estos guías turísticos son organizaciones comerciales, como Red Hat Software, fabricante de una distribución llamada Red Hat, que tiene un cierto aire comercial. En la mayoría de casos metes un CD-ROM de Red Hat en el PC, lo inicias y él solito maneja todo lo demás. Así como el guía turístico egipcio esperará algún tipo de compensación por sus servicios, hay que pagar por las distribuciones comerciales. En la mayoría de los casos no cuestan casi nada y merece la pena.

Yo uso una distribución llamada Debian[4] (la palabra es una contracción de «Deborah» e «Ian»), que es nocomercial. Está organizada (o más bien debiera decir «se ha organizado») siguiendo las mismas líneas que Linux en general, esto es, consiste en voluntarios que colaboran en la Red, cada uno responsable de cuidar de un pedazo distinto

---

[3]En un país exótico, el mejor guía es un nativo que tenga buen inglés. Eric S. Raymond es un eminente hacker del software de fuente abierta, que se ha convertido en el principal antropólogo de la tribu del software de fuente abierta. Tiene series continuas de artículos disponibles en la web. El primero y mejor conocido es «La catedral y el bazar». El segundo es «Cultivando la noosfera». Otros están planeados. Probablemente el medio más seguro para encontrar estos artículos es visitar la web de Raymond, en `http://www.tuxedo.org/~esr` [ambos artículos se encuentran disponibles en castellano en la BiblioWeb del Proyecto sinDominio: `http://sindominio.net/biblioweb` (*N. del E.*)]

[4]De nuevo, el vocablo adecuado de acuerdo a la terminología propuesta por Stallman sería «Debian GNU/Linux». Esta nomenclatura es un modo implícito de recordarnos algo que he intentado hacer explícito en este ensayo: que nada de esto existiría sin GNU.

del sistema. Estas personas han dividido Linux en diversos paquetes, que son archivos comprimidos que pueden descargarse a un sistema Linux de Debian ya en funcionamiento, luego se abren y descomprimen usando una aplicación de instalación libre. Por supuesto, como tal, Debian no tiene rama comercial —no tiene mecanismo de distribución—. Se pueden descargar todos los paquetes de Debian por Internet, pero la mayoría de la gente prefiere tenerlos en CD-ROM. Diversas compañías se han ocupado de meter todos los actuales paquetes de Debian en CD-ROM y venderlos. Yo compré el mío de Linux Systems Labs. Un conjunto de tres discos, que contenía Debian completo, me costó menos de tres dolares. Pero (y esta es una distinción importante) ni un centavo de esos tres dólares va a parar a ninguno de los programadores que codificaron Linux, ni a los empaquetadores de Debian. Va a parar a Linux Systems Labs y no paga el software ni los paquetes, sino el coste de imprimir los CD-ROM.

Toda distribución de Linux encarna algún truco más o menos astuto para evitar el proceso normal de encendido y hacer que cuando el ordenador arranque se organice no como un PC ejecutando Windows, sino como un *host*[5] que ejecuta Unix. Esto resulta algo alarmante la primera vez que se ve, pero es completamente inofensivo. Cuando se inicia un PC, lleva a cabo una pequeña autocomprobación de rutina, realizando un inventario de los discos y memoria disponibles, y luego empieza a buscar un disco desde el que arrancar. En cualquier ordenador Windows normal, ese disco será el disco duro. Pero si el sistema está bien configurado, primero buscará un disquete o un disco de CD-ROM, y arrancará a partir de uno de estos si están disponibles.

Linux explota esta rendija en las defensas. El ordenador percibe un disco de inicio en la disquetera o en la unidad de CD-ROM, carga el código objeto de ese disco y ciegamente empieza a ejecutarlo. Pero no es código de Microsoft o Ap-

[5]En el mundo Unix, *host* es sinónimo de máquina capaz de conectarse a una red. [*N. del E.*]

ple, es código Linux, así que en este punto el ordenador se empieza a comportar de un modo muy distinto al acostumbrado. Empiezan a aparecer mensajes crípticos en pantalla. Si se hubiera iniciado desde un sistema operativo comercial, en este momento se vería un dibujito de «Bienvenido a MacOS», o una pantalla llena de nubes en el cielo azul y el logo de Windows. Pero con Linux aparece un largo telegrama impreso en crudas letras blancas en una pantalla negra. No hay ningún mensaje de bienvenida. La mayor parte del telegrama tiene el semiescrutable aire amenazante de los graffitis:

```
Dec 14 15:04:15 theRev syslogd 1.3-3#17: restart.
Dec 14 15:04:15 theRev kernel: klogd 1.3-3, log source =
  /proc/kmsg started.
Dec 14 15:04:15 theRev kernel: Loaded 3535 symbols from
  /System.map.
Dec 14 15:04:15 theRev kernel: Symbols match kernel version
  2.0.30
Dec 14 15:04:15 theRev kernel: No module symbols loaded.
Dec 14 15:04:15 theRev kernel: Intel MultiProcessor
  Specification v1.4
Dec 14 15:04:15 theRev kernel: Virtual Wire compatibility
  mode.
Dec 14 15:04:15 theRev kernel: OEM ID: INTEL Product ID:
  440FX APIC at: 0xFEE00000
Dec 14 15:04:15 theRev kernel: Processor #0 Pentium(tm) Pro
  APIC version 17
Dec 14 15:04:15 theRev kernel: Processor #1 Pentium(tm) Pro
  APIC version 17
Dec 14 15:04:15 theRev kernel: I/O APIC #2 Version 17 at
  0xFEC00000.
Dec 14 15:04:15 theRev kernel: Processors: 2
Dec 14 15:04:15 theRev kernel: Console: 16 point font, 400
  scans
Dec 14 15:04:15 theRev kernel: Console: colour VGA+ 80x25,
  1 virtual console (max 63)
Dec 14 15:04:15 theRev kernel: pcibios_init : BIOS32
  Service Directory structure at 0x000fdb70
Dec 14 15:04:15 theRev kernel: pcibios_init : BIOS32
  Service Directory entry at 0xfdb80
Dec 14 15:04:15 theRev kernel: pcibios_init : PCI BIOS
  revision 2.10 entry at 0xfdba1
Dec 14 15:04:15 theRev kernel: Probing PCI hardware.
Dec 14 15:04:15 theRev kernel: Warning : Unknown PCI device
  (10b7:9001). Please read include/linux/pci.h
Dec 14 15:04:15 theRev kernel: Calibrating delay loop..
  ok - 179.40 BogoMIPS
Dec 14 15:04:15 theRev kernel: Memory: 64268k/66556k
  available (700k kernel code, 384k reserved, 1204k data)
Dec 14 15:04:15 theRev kernel: Swansea University Computer
  Society NET3.035 for Linux 2.0
Dec 14 15:04:15 theRev kernel: NET3: Unix domain sockets
```

```
  0.13 for Linux NET3.035.
Dec 14 15:04:15 theRev kernel: Swansea University Computer
  Society TCP/IP for NET3.034 Dec 14 15:04:15 theRev
  kernel: IP Protocols: ICMP, UDP, TCP  Dec 14 15:04:15
  theRev kernel: Checking 386/387 coupling... Ok, fpu using
  exception 16 error reporting.
Dec 14 15:04:15 theRev kernel: Checking 'hlt'
  instruction... Ok.
Dec 14 15:04:15 theRev kernel: Linux version 2.0.30
  (root@theRev) (gcc version 2.7.2.1) #15 Fri Mar 27
  16:37:24 PST 1998
Dec 14 15:04:15 theRev kernel: Booting processor 1 stack
  00002000:Calibrating delay loop.. ok - 179.40 BogoMIPS
Dec 14 15:04:15 theRev kernel: Total of 2 processors
  activated (358.81 BogoMIPS).
Dec 14 15:04:15 theRev kernel: Serial driver version 4.13
  with no serial options enabled
Dec 14 15:04:15 theRev kernel: tty00 at 0x03f8 (irq = 4) is
  a 16550A
Dec 14 15:04:15 theRev kernel: tty01 at 0x02f8 (irq = 3) is
  a 16550A
Dec 14 15:04:15 theRev kernel: lp1 at 0x0378, (polling)
Dec 14 15:04:15 theRev kernel: PS/2 auxiliary pointing
  device detected -- driver installed.
Dec 14 15:04:15 theRev kernel: Real Time Clock Driver v1.07
Dec 14 15:04:15 theRev kernel: loop: registered device at
  major 7
Dec 14 15:04:15 theRev kernel: ide: i82371 PIIX (Triton) on
  PCI bus 0 function 57
Dec 14 15:04:15 theRev kernel: ide0: BM-DMA at
  0xffa0-0xffa7
Dec 14 15:04:15 theRev kernel: ide1: BM-DMA at
  0xffa8-0xffaf
Dec 14 15:04:15 theRev kernel: hda: Conner Peripherals
  1275MB - CFS1275A, 1219MB w/64kB Cache, LBA,
  CHS=619/64/63
Dec 14 15:04:15 theRev kernel: hdb: Maxtor 84320A5, 4119MB
  w/256kB Cache, LBA, CHS=8928/15/63, DMA
Dec 14 15:04:15 theRev kernel: hdc: , ATAPI CDROM drive
Dec 15 11:58:06 theRev kernel: ide0 at 0x1f0-0x1f7,0x3f6 on
  irq 14
Dec 15 11:58:06 theRev kernel: ide1 at 0x170-0x177,0x376 on
  irq 15
Dec 15 11:58:06 theRev kernel: Floppy drive(s): fd0 is
  1.44M
Dec 15 11:58:06 theRev kernel: Started kswapd v 1.4.2.2
Dec 15 11:58:06 theRev kernel: FDC 0 is a National
   Semiconductor PC87306
Dec 15 11:58:06 theRev kernel: md driver 0.35 MAX_MD_DEV=4,
  MAX_REAL=8
Dec 15 11:58:06 theRev kernel: PPP: version 2.2.0 (dynamic
 channel allocation)
Dec 15 11:58:06 theRev kernel: TCP compression code
  copyright 1989 Regents of the University of California
Dec 15 11:58:06 theRev kernel: PPP Dynamic channel
  allocation code copyright 1995 Caldera, Inc.
Dec 15 11:58:06 theRev kernel: PPP line discipline
  registered.
```

```
Dec 15 11:58:06 theRev kernel: SLIP: version
  0.8.4-NET3.019-NEWTTY (dynamic channels, max=256).
Dec 15 11:58:06 theRev kernel: eth0: 3Com 3c900 Boomerang
  10Mbps/Combo at 0xef00, 00:60:08:a4:3c:db, IRQ 10
Dec 15 11:58:06 theRev kernel: 8K word-wide RAM 3:5 Rx:Tx
  split, 10base2 interface.
Dec 15 11:58:06 theRev kernel: Enabling bus-master
  transmits and whole-frame receives.
Dec 15 11:58:06 theRev kernel: 3c59x.c:v0.49 1/2/98 Donald
  Becker http://cesdis.gsfc.nasa.gov/linux/drivers
  /vortex.html
Dec 15 11:58:06 theRev kernel: Partition check:
Dec 15 11:58:06 theRev kernel: hda: hda1 hda2 hda3
Dec 15 11:58:06 theRev kernel: hdb: hdb1 hdb2
Dec 15 11:58:06 theRev kernel: VFS: Mounted root (ext2
  filesystem) readonly.
Dec 15 11:58:06 theRev kernel: Adding Swap: 16124k
  swap-space (priority -1)
Dec 15 11:58:06 theRev kernel: EXT2-fs warning: maximal
  mount count reached, running e2fsck is recommended
Dec 15 11:58:06 theRev kernel: hdc: media changed
Dec 15 11:58:06 theRev kernel: ISO9660 Extensions:
  RRIP_1991A
Dec 15  11:58:07 theRev syslogd 1.3-3#17: restart.
Dec 15 11:58:09 theRev diald[87]: Unable to open options
  file /etc/diald/diald.options: No such file or directory
Dec 15 11:58:09 theRev diald[87]: No device specified. You
  must have at least one device!
Dec 15 11:58:09 theRev diald[87]: You must define a
  connector script (option 'connect').
Dec 15 11:58:09 theRev diald[87]: You must define the
  remote ip address.
Dec 15 11:58:09 theRev diald[87]: You must define the local
  ip address.
Dec 15 11:58:09 theRev diald[87]: Terminating due to
  damaged reconfigure.
```

Las únicas partes de esto que resultan legibles para las personas normales son los mensajes de error y las advertencias. Y sin embargo, es notable que Linux no se detiene, o se viene abajo, cuando encuentra un error; escupe un gemido quejumbroso, abandona los procesos dañados, y sigue adelante. Decididamente, esto no era así en las primeras versiones de los sistemas operativos de Apple y Microsoft, por el sencillo motivo de que un sistema operativo que no es capaz de andar y mascar chicle a la vez no puede recobrarse de los errores. Buscar y solucionar errores requiere un proceso aparte que corra en paralelo al que ha fallado. Una especie de superego, si lo prefieren, que mantiene vigilados a los demás y entra en acción cuando uno se desvía. Ahora que MacOS y Windows pueden hacer más de una cosa a la vez se les da mucho mejor tratar con los errores que antes, pero no se aproximan siquiera a Linux o los demás sistemas Unix en este aspecto; y su mayor complejidad les ha hecho vulnerables a nuevos tipos de error.

Falibilidad, enmienda, redención, confianza y otros arcanos conceptos técnicos

LINUX NO ES CAPAZ DE TENER POLÍTICAS centralmente organizadas que dicten cómo escribir mensajes de error y documentación, así que cada programador escribe los suyos propios. Habitualmente están en inglés, aunque montones de programadores de Linux son europeos. Frecuentemente son graciosos. Siempre son honestos. Si ha ocurrido algo malo porque el software sencillamente todavía no está acabado, o porque el usuario fastidió algo, lo dirán con todas las letras. La interfaz de línea de comandos facilita que los programas escupan pequeños comentarios, advertencias y mensajes aquí y allí. Incluso si una aplicación está implosionando como un submarino dañado, habitualmente puede seguir lanzando un pequeño mensaje de SOS. A veces, cuando se deja de trabajar con un programa y se cierra, uno se encuentra con que ha dejado detrás una serie de advertencias y mensajes de error no muy graves en la ventanas de la interfaz de línea de comandos desde la que se ejecutó. Como si el software te contara cómo le iba mientras trabajabas con él.

La documentación, en Linux, viene en forma de páginas `man` (abreviatura de *manual*. Se puede acceder a ellas bien mediante una GUI (`xman`) o desde la línea de comandos (`man`). Esta es una muestra de la página `man` de un programa llamado `rsh`:

> Detener señales detener sólo el proceso rsh local; esto es posiblemente erróneo, pero actualmente bastante difícil de solucionar por razones demasiado complicadas para explicarlas aquí.

Las páginas `man` contienen un montón de material parecido, que suena como las murmuraciones de pilotos pugnando con los mandos de aviones averiados. La sensación general es la de miles de monumentales pero oscuras pugnas vistas a la luz paralizante de un estroboscopio. Cada programador está tratando con sus propios obstáculos y fallos; está demasiado ocupado solucionándolos, y mejorando el software, para explicar las cosas en detalle o tener elaboradas pretensiones.

En la práctica casi nunca se encuentra un fallo serio en Linux. Cuando se encuentra, es casi siempre en el software comercial (varios vendedores comercializan software que funciona en Linux). El sistema operativo y sus programas fundamentales de utilidad son demasiado importantes para contener fallos serios. Llevo ejecutando Linux cada día desde finales de 1995 y he visto cómo muchos programas de aplicaciones caían pasto de las llamas, pero nunca he visto que el sistema operativo se venga abajo. Nunca. Ni una sola vez. Hay unos cuanto sistemas Linux que llevan meses o años funcionando continuamente y trabajando duro sin necesidad de reiniciarlos.

Los sistemas operativos comerciales tienen que adoptar la misma postura oficial hacia los errores que tenían los países comunistas frente a la pobreza. Por razones de doctrina, no resultaba posible admitir que la pobreza era un serio problema en los países comunistas, porque la idea misma del comunismo era erradicar la pobreza. Igualmente, las compañías de sistemas operativos comerciales como Apple o Microsoft no pueden ir por ahí admitiendo que su software tiene errores y se cae todo el rato, no más de lo que Disney puede emitir comunicados de prensa firmando que el ratón Mickey es un actor disfrazado.

Esto es un problema, porque los errores existen y suceden. Cada pocos meses Bill Gates trata de hacer una demostración de un nuevo producto de Microsoft ante un gran público sólo para que le reviente en las narices. Los distribuidores de sistemas operativos comerciales, como consecuencia directa de ser comerciales, se ven forzados a adoptar la posición groseramente tosca de que los errores son raras aberraciones, habitualmente la culpa de otro, y por tanto no merece la pena hablar de ello en detalle. Esta postura, que todo el mundo sabe que es absurda, no se limita a comunicados de prensa y campañas publicitarias: constituye el modo mismo en que estas compañías hacen negocios y se relacionan con sus clientes. Si la documentación estuviera bien escrita, mencionaría fallos, errores y caídas del sistema en cada página. Si los sistemas de ayuda en línea que vienen con estos sistemas operativos reflejaran la experiencia y preocupaciones de sus usuarios, estarían dedicados básicamente a instrucciones acerca de cómo tratar con los fallos y errores del sistema.

Pero esto no sucede. Las compañías de accionistas son maravillosos inventos que nos han dado muchos excelentes bienes y servicios. Se les dan bien muchas cosas. Admitir el fracaso no es una de ellas. Diablos, ni siquiera admiten fallos menores.

Por supuesto, este comportamiento no es tan patológico en una compañía como lo sería en un ser humano. La mayoría de la gente hoy en día entiende que los comunicados de prensa de las empresas se lanzan para quedar bien con los accionistas de la compañía, no para ilustrar al público. A veces los resultados de esta deshonestidad institucional pueden ser espantosos, como en el caso del tabaco y del amianto. En el caso de los distribuidores de sistemas operativos comerciales no es nada así, por supuesto; solamente es irritante.

Algunos podrían argüir que la irritación de los consumidores, con el tiempo, se convierte en una especie de placa endurecida que puede ocultar un serio deterioro, y que la honestidad podría ser así la mejor política a largo plazo; el jurado aún tiene que decidir acerca de esto en el mercado de los sistemas operativos. El negocio se está expandiendo lo bastante rápido como para que siga siendo mucho mejor tener miles de millones de clientes crónicamente irritados que millones de clientes contentos.

La mayoría de administradores de sistemas que conozco que trabajan siempre con Windows NT están de acuerdo en que cuando tiene un fallo hay que reiniciarlo, y cuando se fastidia en serio el único modo de arreglarlo es reinstalar el sistema operativo desde el principio. O al menos éste es el único modo que conocen de arreglarlo, lo cual viene a ser lo mismo. Es muy posible que los ingenieros de Microsoft tengan un montón de información privilegiada sobre cómo arreglar el sistema cuando va mal, pero si la tienen, no parecen estar transmitiendo el mensaje a ninguno de los administradores de sistema que yo conozca.

Debido a que Linux no es comercial —porque es, de hecho, gratuito, así como bastante difícil de obtener, instalar, y operar[1]— no tiene que mantener ninguna pretensión acerca de su fiabilidad. En consecuencia, es mucho más fiable. Cuando algo falla en Linux, el error es detectado y discutido vivamente de inmediato. Cualquiera con los conocimientos técnicos necesarios puede ir derecho al código fuente y señalar el origen del error, que es rápidamente solucionado por el hacker que fuera responsable de ese programa en particular.

Por lo que yo sé, Debian es la única distribución de Linux que tiene su propia constitución,[2] pero lo que realmente me convenció fue su impresionante base de datos de errores[3], que es una especie de Archivo de Indias interactivo del error, la falibilidad y la redención. Es la simplicidad misma. Cuando tuve un problema con Debian a prin-

[1] Recordemos que este ensayo se escribió a principios de 1999: ha pasado casi un lustro, y desde entonces se han dedicado grandes esfuerzos a distribuir y facilitar la instalación de cualquier sistema GNU/Linux, por lo que hoy día es fácil hacerse con uno y su dificultad de instalación y de uso no es mayor a la de cualquier sistema operativo comercial. [*N. del E.*]

[2] `http://www.debian.org/devel/constitution`

[3] Se la conoce como BTS (*Bug Tracking System*), «Sistema de seguimiento de fallos»: `http://www.debian.org/Bugs` [*N. del E.*]

cipios de enero de 1997, mandé un mensaje describiendo el problema a `submit@bugs.debian.org`. De inmediato, a mi problema se le asignó un número de informe de fallo (#6518) y un nivel de gravedad (las opciones disponibles eran: crítico, grave, serio, importante, normal, menor, arreglado y petición de características [*wishlist*]) y se reenvió a las listas de correo por las que merodea la gente de Debian. En veinticuatro horas había recibido cinco emails que me decían cómo solucionar el problema: dos de Norteamérica, dos de Europa y uno de Australia. Todos estos emails me daban la misma sugerencia, que funcionó, e hizo que mi problema se desvaneciera. Pero al mismo tiempo se envió una transcripción de este intercambio a la base de datos de fallos de Debian, de tal modo que si otros usuarios tenían el mismo problema más adelante, podrían buscar y hallar la solución sin tener que realizar un nuevo y redundante informe de fallo.

Compárese esto con la experiencia que tuve cuando traté de instalar Windows NT 4.0 en el mismo ordenador cerca de diez meses después, a finales de 1997. El programa de instalación sencillamente se detuvo a mitad del proceso sin emitir ningún mensaje de error. Fui al sitio web de Microsoft y traté de buscar documentos de ayuda que abordasen mi problema. El motor de búsqueda no funcionaba en absoluto; no hizo nada. Ni siquiera me dio un mensaje que me dijera que no funcionaba.

Al final decidí que mi placa base debía de ser defectuosa; era una marca y modelo ligeramente inusuales y NT no soportaba tantas placas base como Linux. Siempre ando buscando excusas por muy endebles que sean para comprar nuevo hardware, así que compré una nueva placa base compatible con Windows NT, lo cual quería decir que llevaba el logotipo de Windows NT impreso en la caja. La instalé en mi ordenador, arranqué Linux y traté de instalar Windows NT de nuevo. De nuevo la instalación falló sin ningún mensaje de error y ninguna explicación. Para entonces ya habían transcurrido un par de semanas y pensé que tal vez el motor de búsqueda del sitio web de Microsoft estaría funcionando. Lo intenté, pero seguía sin funcionar.

Así que creé una nueva cuenta de ayuda Microsoft, me registré e informé del incidente. Di el número de ID de mi producto cuando me lo pidieron y empecé a seguir las instrucciones en una serie de pantallas de ayuda. En otras palabras, estaba enviando un informe de fallo igual que en el sistema Debian. Solamente que la interfaz era más elegante —yo escribía mi queja en pequeños cuadros de edición de texto en formularios web, haciéndolo todo a través del GUI, mientras que con Debian se envía un telegrama en forma de email—. Sabía que cuando terminara de enviar el informe de fallo, se convertiría en propiedad intelectual de Microsoft, y otros usuarios no podrían verlo. Muchos usuarios de Linux se negarían a participar en tal proceso por motivos éticos, pero yo quise probar como experimento. Sin embargo, finalmente nunca pude enviar mi informe de fallo, porque la serie de páginas web enlazadas que estaba rellenando acabó por llevarme a una página completamente en blanco: un callejón sin salida.

Así que volví atrás, hice click en los botones de «ayuda telefónica» y acabaron por darme un número de teléfono de Microsoft. Cuando marqué este número, me respondió una serie de pitidos punzantes y un mensaje grabado de la compañía de teléfonos que decía «Lo sentimos, el número que ha marcado no existe».

Probé de nuevo con la página de búsqueda: seguía sin funcionar. Luego probé PPI (Pago Por Incidencia) de nuevo. Esto me llevó a otra serie de páginas web hasta que acabé en una que decía: «Atención: no hay ninguna página web que corresponda a su petición.»

Probé de nuevo, y acabé llegando una pantalla de Pago Por Incidencia que decía: «NO HAY INCIDENCIAS. No hay ninguna incidencia sin usar en su cuenta. Si desea adquirir una incidencia de ayuda, haga click en OK: entonces podrá pagar por anticipado por una incidencia…» El precio por incidencia era de 95 dólares.

El experimento empezaba a resultar bastante caro, así que renuncié a abordarlo desde el PPI y decidí intentarlo con las «Preguntas Frecuentes» (FAQ, *Frecuently Asked Questions*) en el sitio web de Microsoft. Ninguna de las preguntas habituales disponibles tenía nada que ver con mi problema, salvo una titulada «Tengo problemas al instalar NT», que parecía escrita por publicistas, no por ingenieros.

Así que me rendí, y hasta el día de hoy no he instalado Windows NT en ese ordenador. Para mí, el camino de menor resistencia era simplemente usar Debian Linux.

En el mundo del software de fuente abierta, los informes de fallo son una información útil. Hacerlos públicos es un servicio para los demás usuarios y mejora el sistema operativo. Hacerlos públicos sistemáticamente es tan importante que personas altamente inteligentes invierten tiempo y dinero en mantener bases de datos de fallos. En el mundo de los sistemas operativos comerciales, sin embargo, informar de un fallo es un privilegio por el que hay que pagar mucho dinero. Pero si lo pagas, resulta que el informe de fallo debe ser confidencial… ¡de otro modo, cualquiera puede beneficiarse de tus noventa y cinco pavos! Y sin embargo, nada impide a los usuarios de NT el montar su propia base de datos de fallos pública.

Este es, en otras palabras, otro rasgo del mercado de sistemas operativos que sencillamente carece de sentido a menos que se examine en su contexto cultural. Lo que Microsoft está vendiendo a través del Pago Por Incidente no es tanto un apoyo técnico como la ilusión continuada de que sus clientes están llevando a cabo una especie de transacción racional de negocios. Es una especie de tasa rutinaria de mantenimiento para sostener la fantasía. Si la gente quisiera realmente un sistema operativo sólido, usarían Linux, y si realmente quisieran apoyo técnico encontrarían un modo de obtenerlo; los clientes de Microsoft quieren otra cosa.

En el momento en que escribo esto (enero de 1999), la base de datos de Debian Linux contiene cerca de 32.000 fallos. Casi todos fueron solucionados hace mucho tiempo. Hay doce fallos «críticos» todavía en pie, el más antiguo de los cuales fue enviado hace 79 días. Hay 20 fallos «graves» en pie, el más antiguo de los cuales tiene 1166 días. Hay 48 fallos «importantes» y cientos de fallos «normales» y menos importantes.

Igualmente, BeOS (al que llegaré en un momento) tiene su propia base de datos de errores[4] con su propio sistema de clasificación, incluyendo tales categorías como «No es un fallo», «Característica reconocida» y «No se va a arreglar». Algunos de estos fallos no son nada más que hackers de Be desfogándose, y se clasifican como «Input reconocido». Por ejemplo, encontré uno que se envío el 30 de diciembre de 1998. Está en mitad de una larga lista de fallos, entre uno llamado «El ratón funciona de modo muy raro» y otro llamado «El cambio de marco BView no afecta si BView no va unida a una BWindow». Este se titula:

> R4: A BeOS le falta un cabeza de turco megalómano para centrar y mantener bajo control la furia del programador

y dice lo siguiente:

> Be Status: Input Reconocido
> BeOS Versión: R3.2
> Componente: desconocido
>
> Descripción Completa:
>
> El BeOS necesita un megalómano egomaníaco sentado en su trono para darle un personaje humano que a todo el mundo le encante odiar. Sin esto, el BeOS languidecerá en el ámbito impersonificable de los sistemas operativos que la gente nunca consigue manejar. Se puede juzgar el éxito de un sistema operativo no por la calidad de sus características, sino por lo infames y detestados que son sus líderes.
>
> Creo que esto es un efecto colateral de la camaradería entre programadores en condiciones penosas. Después de todo, a la desdicha le encanta la compañía. Creo que hacer que el BeOS sea menos accesible conceptualmente y mucho menos fiable requerirá que los programadores se unan, desarrollando el tipo de comunidad en la que los extraños se hablan, algo así como en un supermercado antes de una enorme tormenta de nieve.
>
> Siguiendo el mismo programa, probablemente resulte necesario desplazar el cuartel general del BeOS a un clima mucho menos agradable. El incómodo ambiente general generará esta actitud, y realmente no hay mejor receta para el éxito. Yo sugeriría Seattle, pero creo que ya está ocupado.

[4] `http://www.be.com/developers/bugs/index.html`

> Podría intentarse Washington DC, pero definitivamente no un sitio como San Diego o Tucson.

Por desgracia, el sistema de informes de fallo de Be elimina los nombres de las personas que informan de los fallos (¿para protegerles de la venganza?), así que no sé quién escribió esto.

Así que pareciera que estoy pregonando la superioridad técnica y moral de Debian Linux. Pero como casi siempre sucede en el mundo de los sistemas operativos, es más complicado. Tengo Windows NT instalado en otro ordenador y el otro día (enero de 1999), cuando tuve un problema con él, decidí probar con la ayuda técnica de Microsoft otra vez. Esta vez el motor de búsqueda sí que funcionaba (aunque para llegar a él tuve que identificarme como «avanzado»). Y en vez de hacerme las inútiles preguntas habituales, localizó cerca de doscientos documentos (yo estaba usando unos criterios de búsqueda muy vagos) que eran obviamente informes de fallos —aunque se llamaban de otro modo—. Microsoft, en otras palabras, tiene montado un sistema que es funcionalmente equivalente a la base de datos de fallos de Debian. Tiene un aspecto diferente, claro, pero contiene datos técnicos y no disimula la existencia de errores.

Como he explicado, vender sistemas operativos por dinero es una posición bastante insostenible, y el único modo en que Apple y Microsoft lo consiguen es llevando los avances tecnológicos adelante lo más agresivamente que pueden, y haciendo que la gente crea en, y pague por, una imagen particular: en el caso de Apple, la de un librepensador creativo y, en el caso de Microsoft, la del respetable tecnoburgués. Igual que la Disney, están haciendo dinero vendiendo una interfaz, un espejo mágico. Tiene que estar pulido y perfecto o toda la ilusión se arruinará y el plan de negocios se desvanecerá como un espejismo.

En consecuencia, hasta hace poco la gente que escribía manuales y creaba sitios web de apoyo técnico al cliente para sistemas operativos comerciales se veía impedida, por los departamentos legales o de Relaciones Públicas de sus empresas, en admitir, aunque fuera indirectamente, que el software podría contener fallos o que la interfaz podría sufrir el problema del doce parpadeante. No podían tratar las dificultades reales de los usuarios. Los manuales y sitios web eran por tanto inútiles, y hacían que incluso los usuarios seguros de sí mismos en el terreno técnico se preguntaran si se estaban volviendo sutilmente locos.

Cuando Apple tiene este tipo de comportamiento corporativo, uno quiere creer que realmente lo hacen lo mejor que pueden. Todos queremos darle a Apple el beneficio de la duda, porque el malvado Bill Gates les hizo morder el polvo, y porque tienen unas buenas Relaciones Públicas. Pero cuando lo hace Microsoft, uno casi no puede evitar convertirse en un paranoico de las conspiraciones. ¡Obviamente nos están ocultando algo! ¡Y además son tan poderosos! ¡Están tratando de volvernos locos!

Este modo de tratar con los clientes está tomado directamente del totalitarismo centroeuropeo de mediados del siglo XX. A uno le vienen los adjetivos *kafkiano* y *orwelliano* a la mente. No podía durar, no más que el Muro de Berlín, así que ahora Microsoft tiene un base de datos de fallos públicamente disponible. Se llama de otro modo, y lleva un rato encontrarla, pero está ahí.

En otras palabras, se han adaptado a la estructura de dos niveles eloi/morlock de la sociedad tecnológica. Si eres un eloi, instalas Windows, sigues las instrucciones, esperas que todo vaya bien y sufres mudamente cuando se rompe. Si eres un morlock, vas al sitio web, le dices que eres «avanzado», encuentras la base datos de fallos y obtienes la verdad directamente de algún anónimo ingeniero de Microsoft.

Pero una vez que Microsoft ha dado este paso, surge la cuestión, de nuevo, de si tiene algún sentido estar en el negocio de los sistemas operativos en absoluto. Los clientes pueden estar dispuestos a pagar 95 dólares por informar a Microsoft de un problema si, a cambio, les dan un consejo que ningún otro usuario va a obtener. Esto tiene el útil efecto secundario de mantener a los usuarios mutuamente alienados, lo cual contribuye a mantener la ilusión de que los fallos son raras aberraciones. Pero una vez que los resultados de esos informes de fallo están abiertamente disponibles en el sitio web de Microsoft, todo cambia. Nadie va a soltar 95 dólares por informar de un problema cuando lo más probable es que algún otro tipo ya lo haya hecho, y las instrucciones para solucionar el fallo aparezcan de forma gratuita en un sitio web público. Y a medida que crece el tamaño de la base de datos de fallos, acaba convirtiéndose en una clara admisión, por parte de Microsoft, de que sus sistemas operativos tienen tantos fallos como los de sus competidores. Eso no es ninguna vergüenza; como mencioné, la base de datos de fallos de Debian contiene 32.000 informes hasta ahora. Pero pone a Microsoft al mismo nivel que los demás y hace mucho más difícil que sus clientes —que *quieren creer*— crean.

Memento Mori

UNA VEZ QUE LA MÁQUINA LINUX ha terminado de escupir su telegrama de inicio en jerga, me insta a que introduzca un nombre de usuario y una contraseña. En este momento la máquina todavía está ejecutando la interfaz de línea de comandos, con letras blancas sobre fondo negro. No hay ventanas, menús ni botones. No responde al ratón; ni siquiera sabe que el ratón está ahí. En este punto, sin embargo, ya es posible ejecutar un montón de software. Emacs, por ejemplo, existe tanto en versión linea de comandos como en versión gráfica (de hecho hay dos versiones GUI, que reflejan una especie de cisma doctrinal entre Richard Stallman y algunos hackers que se hartaron de él). Lo mismo puede decirse de muchos otros programas Unix. Muchos no tienen siquiera una GUI, y muchos de los que la tienen pueden ejecutarse desde la línea de comandos.

Por supuesto, dado que mi ordenador sólo tiene una pantalla, sólo puedo ver una línea de comandos, así que puede que crean que sólo puedo interactuar con un programa cada vez. Pero si mantengo apretada la tecla Alt y luego pulso el botón de función F2 en lo alto de mi teclado, aparece otra pantalla negra vacía que me pide que dé mi nombre de usuario y contraseña. Puedo entrar e iniciar otro programa, luego pulsar Alt-F1 y regresar a la primera pantalla, que sigue haciendo lo que quiera que estuviera haciendo cuando la dejé. O puedo pulsar Alt-F3 y entrar en otra pantalla, y una cuarta, y una quinta. En una de estas pantallas puedo entrar como yo mismo, en otra como *root* (el administrador del sistema), y en otra puedo entrar en un ordenador distinto a través de la Red.

Cada una de estas pantallas se llama, en jerga Unix, un `tty`, que es la abreviatura de *teletipo*. Así que cuando uso mi sistema Unix de este modo regreso a esa pequeña habitación en el Instituto de Ames donde escribí mi primer código hace veinticinco años, excepto que el `tty` es más silencioso y rápido que un teletipo, y es capaz de ejecutar un software incomparablemente superior, tal como `emacs` o las herramientas de desarrollo de GNU.

Resulta fácil (fácil para el estándar de Unix, no el de Apple/Microsoft) configurar un sistema Unix de modo que vaya directamente a un GUI cuando lo inicies. De este modo nunca se ve una pantalla `tty`. Yo todavía hago que el mío se inicie con esta pantalla de teletipo, blanco sobre negro, como un *memento mori* computacional. Solía estar de moda que los escritores tuvieran un cráneo humano sobre su escritorio como recordatorio de su mortalidad, de que todo era vanidad. La pantalla `tty` me recuerda que lo mismo sucede con las elegantes GUI.

El X Window System, que es la GUI de Unix, ha de ser capaz de ejecutarse en cientos de tarjetas de vídeo diferentes con diferentes chips, memoria y buses de placa base. Igualmente, hay cientos de tipos distintos de monitores en el mercado nuevo y usado, cada uno con diferentes especificaciones, así que probablemente haya más de un millón de combinaciones posibles de tarjeta y monitor. Lo único que todas tienen en común es que funcionan en modo VGA, que es la vieja pantalla de línea de comandos que se ve durante unos pocos segundos al iniciar Windows. Así que Linux siempre arranca en VGA, con una interfaz de teletipo, porque al principio no tiene ni idea de qué clase de hardware está conectado al ordenador. Para ir desde el teletipo hasta el GUI, hay que decirle a Linux exactamente qué tipo de hardware hay. Si te equivocas, obtendrás una pantalla en blanco en el mejor de los casos y, en el peor, podrías destruir físicamente el monitor, al enviarle señales que no puede manejar.

Cuando empecé a usar Linux, todo esto había que hacerlo a mano. Una vez me pasé casi un mes tratando de hacer que un monitor rebelde funcionara, y llené la mayor parte de un cuaderno con notas garabateadas cada vez más desesperadas. Hoy en día, la mayor parte de las distribuciones Linux incluyen un programa que automáticamente examina y configura el sistema, así que instalar X Window es casi tan fácil como instalar una GUI de Apple/Microsoft. La información crucial va a un archivo (un archivo de texto ASCII, naturalmente) llamado XF86Config, al que merece la pena echar un vistazo incluso aunque la distribución lo cree automáticamente. Para la mayor parte de la gente parece una serie de ensalmos crípticos sin sentido —y esa era la idea de mirarlo—. Un sistema Apple/Microsoft debe de tener la misma información para lanzar su GUI, pero posiblemente esté escondida en las profundidades, o probablemente esté en un archivo que ni siquiera puede abrir y leer un editor de textos. Todos los archivos importantes que hacen que los sistemas Linux funcionen están a la vista. Siempre son archivos de texto ASCII, así que no hacen falta herramientas especiales para leerlos. Se pueden mirar siempre que se quiera, lo cual es bueno, y se puede enredar con ellos y volver el sistema completamente disfuncional, lo cual ya no es tan bueno.

En cualquier caso, asumiendo que mi archivo `XF86Config` esté tal cual, introduzco el comando `startx` para iniciar el sistema X Window. La pantalla queda en blanco durante un minuto, el monitor emite extraños ruidos chirriantes, luego se reconstituye como un escritorio gris en blanco con un cursor de ratón en el medio. Al mismo tiempo inicia el gestor de ventanas. X Window es software de bastante bajo nivel: proporciona la infraestructura para una interfaz gráfica de usuario, y es una infraestructura pesada e industrial. Pero no trabaja con ventanas. Eso lo maneja otra categoría de la aplicación colocada encima de X Window, llamada «gestor de ventanas». Hay varios disponibles, todos gratuitos, por supuesto. El clásico es Tom's Window Manager (*twm*, el «Gestor de Ventanas de Tom») pero hay una variante más pequeña y supuestamente más eficiente llamada *fvwm*, que es la que yo uso. Le tengo el ojo echado a un gestor de ventanas completamente diferente llamado *Enlightenment,* que puede ser el producto tecnológico más elegante que haya visto nunca, puesto que a) es para Linux, b) es libre, c) está siendo desarrollado por un número muy pequeño de hackers obsesos, y d) tiene un aspecto asombrosamente estiloso; es el tipo de gestor de ventanas que podría aparecer en el trasfondo de una película de Alien.

En cualquier caso, el gestor de ventanas funciona como un intermediario entre X Window y el software que se esté usando. Dibuja los bordes de las ventanas, los menús, y demás, mientras las aplicaciones dibujan el contenido de las ventanas. Las aplicaciones pueden ser de cualquier tipo: editores de texto, navegadores web, paquetes gráficos o utilidades, como un reloj o una calculadora. En otras palabras, a partir de este punto, da la sensación de haber pasado a un universo paralelo bastante parecido al familiar universo de Apple o Microsoft, pero ligera y ubicuamente diferente. El principal programa gráfico en Apple/Microsoft es Adobe Photoshop, pero en Linux es algo llamado Gimp. En vez de Microsoft Office, se puede comprar algo llamado ApplixWare.[1] Hay muchos paquetes de software comercial, tales como Mathematica, Netscape Communicator y Adobe Acrobat, disponibles en versión Linux y, según cómo se configure el gestor de ventanas, se puede hacer que tengan el mismo aspecto y se comporten igual que lo harían en MacOS o Windows.

Pero hay un tipo de ventana que se verá en la interfaz gráfica de Linux y que es raro o inexistente en otros sistemas operativos. Estas ventanas se llaman `xterm` y no contienen nada más que líneas de texto —esta vez texto negro sobre fondo blanco, aunque se pueden cambiar los colores: cada ventana `xterm` es una interfaz de línea de comandos en sí misma—, un `tty` en una ventana. Así que incluso

[1]Como sustituto de Microsoft Office, hoy día hay disponible una magnífica suite ofimática libre y multiplataforma llamada OpenOffice: `http://www.openoffice.org` [*N. del E.*]

cuando se está en pleno modo gráfico, se puede seguir hablando con el ordenador Linux a través de una interfaz de línea de comandos.

Hay mucho buen software de Unix que no tiene interfaz gráfica en absoluto. Esto puede deberse al hecho de que se desarrolló antes de que X Window estuviera disponible, o porque las personas que lo escribieron no querían sufrir todo el agobio de crear una GUI, o sencillamente porque no lo necesitaban. En cualquier caso, esos programas pueden invocarse introduciendo sus nombres en la línea de comandos de una ventana `xterm`. El comando `whoami`, mencionado antes, es un buen ejemplo. Hay otro llamado `wc` (*word count,* recuento de palabras) que sencillamente devuelve el número de líneas, palabras y caracteres en un archivo de texto.

La capacidad de ejecutar este programitas de utilidades en la línea de comandos es una gran virtud de Unix, y una que es improbable que dupliquen los sistemas operativos de interfaz gráfica pura. El comando `wc`, por ejemplo, es el tipo de cosa que resulta fácil de escribir con una interfaz de línea de comandos. Probablemente no consiste más que de una pocas líneas de código, y un programador listo quizá podría escribirlo en una sola línea. En forma compilada sólo ocupa unos pocos bytes de espacio de disco. Pero el código requerido para darle una interfaz gráfica de usuario a ese programa probablemente tendría cientos o incluso miles de líneas, dependiendo del capricho del programador. Compilado en un software ejecutable, tendría un montón de código GUI. Sería lento de iniciar y ocuparía un montón de memoria. Este esfuerzo sencillamente no valdría la pena, así que `wc` nunca se escribiría como un programa independiente. Los usuarios tendrían que esperar a que el recuento de palabras viniera incluido en un paquete de software comercial.

Las interfaces gráficas tienden a imponer un montón de código superfluo al software, incluso al más pequeño, y este plus cambia completamente el entorno de programación. Las pequeñas utilidades ya no merecen la pena escribirse. Esta funciones tienden a ser aglutinadas en paquetes más amplios de software. A medida que las interfaces gráficas se vuelven más complejas, e imponen cada vez más código superfluo, esta tendencia se vuelve omnipresente, y los paquetes de software se hacen cada vez más colosales; a partir de cierto punto empiezan a fusionarse, como Word, Excel y PowerPoint se fundieron en Microsoft Office: un enorme Corte Inglés de software al borde de una ciudad llenas de tiendecitas en quiebra.

Es una analogía injusta, porque cuando una tiendecita quiebra significa que un tendero ha cerrado el negocio. Por supuesto, nada de eso ocurre cuando `wc` queda subsumido en uno de los incontables elementos del menú de Microsoft Word. El único inconveniente real es la pérdida de flexibilidad para el usuario, pero es una pérdida que la mayoría de clientes obviamente no nota o no les importa. El inconveniente más serio del «enfoque Corte Inglés» es que la mayoría de usuarios sólo quieren o necesitan una pequeña parte de lo que contienen estos gigantescos paquetes de software. El resto es basura, peso muerto. Y sin embargo el usuario en el cubículo de al lado tendrá opiniones completamente distintas acerca de qué es útil y qué no lo es.

La otra cosa importante que hay que mencionar aquí es que Microsoft ha incluido una característica verdaderamente elegante en Office: un paquete de programación en Basic. Basic es el primer lenguaje de ordenador que aprendí, allá cuando usaba la cinta de papel y el teletipo. Usando la versión de Basic que viene incluida en Office uno puede escribir sus propias utilidades que saben cómo interactuar con todos los enredos, pijaditas, lacitos y pompones de Office. Basic es más fácil de usar que los lenguajes utilizados habitualmente en la programación Unix de línea de comandos, y Office ha llegado a muchas más personas que las herramientas GNU. Así que es bastante posible que esta característica de Office acabe por generar mucho más hacking que GNU.

Pero ahora estoy hablando del software de aplicaciones, no de sistemas operativos. Y como he dicho, el software de aplicaciones de Microsoft tiende a ser muy bueno. Yo no lo uso mucho, porque no entro dentro de su mercado diana.

Si Microsoft saca alguna vez un paquete de software que yo use y me guste, entonces será el momento de que se deshagan del stock, porque yo soy un segmento de mercado de una persona.

La fatiga del «geek»

En los años que llevo trabajando con Linux[1] he llenado tres cuadernos y medio registrando mis experiencias. Sólo empiezo a escribir cosas cuando estoy haciendo algo complicado, como instalar X Window o enredar con mi conexión de Internet, así que estos cuadernos sólo contienen el registro de mis luchas y frustraciones. Cuando las cosas me salen bien, trabajo feliz y contento durante muchos meses sin anotar nada. Así que estos cuadernos son una lectura bastante lúgubre. Cambiar nada en Linux es cuestión de abrir varios de esos pequeños archivos ASCII y cambiar una palabra aquí y un carácter allí, de modos que resultan extremadamente significativos para el funcionamiento del sistema.

Muchos de los archivos que controlan el funcionamiento de Linux no son nada más que líneas de comando que se volvieron tan largas y complicadas que ni siquiera los

[1] Los *geeks* son primos hermanos de los *nerds* y de los hackers. De hecho muchos de ellos son las tres cosas. El hacker se autodenomina a menudo *geek* en lugar de *hacker,* término reverencial y con demasiado peso que rara vez un hacker usará para referirse a sí mismo. Los *geeks* suelen ser gente con buenas aptitudes tecnológicas, que adoran los *gadgets*, van cargados a todas partes con diferentes cacharros electrónicos y llevan siempre camisetas de congresos de tecnología, de hacklabs o de series de ciencia ficción. No son necesariamente adolescentes: un *geek* puede ser uno de los altos y serios directivos de una empresa tecnológica, el joven estudiante universitario que insiste en que haya conexión por cable o ADSL en la residencia de estudiantes o un abuelo que acaba de descubrir Internet. [*N. del E.*]

hackers de Linux podrían escribirlas correctamente. Cuando se trabaja con algo tan potente como Linux, fácilmente se puede dedicar toda una media hora a escribir una sola línea de comando. Por ejemplo, el comando `find`, que busca en todo el sistema de archivos aquellos ficheros que cumplan ciertos criterios, es fantásticamente potente y general. Su `man` tiene once páginas, y son páginas concisas; podrían expandirse a todo un libro. Además, como si eso no fuera lo bastante complicado por sí mismo, siempre se puede dirigir la salida de un comando Unix a la entrada de otro igualmente complicado. El comando `pon`, que se usa para activar una conexión PPP con Internet, requiere tanta información detallada que básicamente resulta imposible lanzarlo todo desde la línea de comandos. En lugar de eso, se abstraen grandes pedazos de su entrada a tres o cuatro archivos distintos. Hace falta un *script*[2] de marcación, que de hecho es un programita que le dice cómo marcar el teléfono y responder a diversos sucesos; un archivo llamado `options`, que lista cerca de sesenta opciones diferentes sobre cómo instalar la conexión PPP; y un archivo llamado `secrets`, que incluye información sobre las contraseñas.

Presumiblemente hay hackers cuasidivinos de Unix en algún lugar del mundo que no tienen por qué usar estos pequeños scripts y archivos de opciones como muleta, y que sencillamente pueden sacar líneas de comando fantásticamente complejas sin cometer errores tipográficos y sin tener que pasarse horas hojeando la documentación. Pero yo no soy uno de ellos. Como casi todos los usuarios de Linux, dependo de miles de pequeños archivos de texto ASCII que ocultan todos esos detalles y que a su vez están metidos en recovecos del sistema de archivos de Unix. Cuando quiero cambiar algo acerca del modo en que funciona mi sistema, edito esos archivos. Sé que si no sigo la pista de cada pequeño cambio que he realizado, no podré hacer que el sistema

[2]Un *script* o «guión» es un fichero de texto que contiene una serie de instrucciones que se pueden invocar en la línea de comandos, y que se ejecutarán de forma secuencial. En ese sentido es semejante a un fichero con extensión BAT de MS-DOS, si bien es muchísimo más potente y puede ser programado de modo mucho más complejo. [*N. del E.*]

funcione tras haber enredado con él. Mantener registros escritos a mano es tedioso, por no decir algo anacrónico. Pero es necesario.

Probablemente me habría ahorrado un montón de dolores de cabeza trabajando con una compañía llamada Cygnus Support, que existe para proporcionar ayuda a los usuarios de software libre. Pero no lo hice, porque quería ver si podía hacerlo yo solo. La respuesta resultó ser que sí, pero por los pelos. Y hay muchos retoques y optimizaciones que probablemente podría hacer a mi sistema que nunca he llegado a probar, en parte porque algunos días me canso de ser un morlock, y en parte porque me da miedo estropear un sistema que en general me funciona bien.

Aunque Linux me vale a mí y a muchos otros usuarios, su potencia y generalidad son su talón de Aquiles. Si uno sabe lo que está haciendo, puede comprar un PC barato de cualquier tienda de ordenadores, tirar los discos de Windows que lleva incluidos y convertirlo en un sistema Linux de desconcertante complejidad y potencia. Puede enchufarlo a otros doce ordenadores Linux y convertirlo en parte de un ordenador paralelo. Puede configurarlo de tal modo que cien personas diferentes puedan entrar en él a través de Internet, por vía de otras tantas líneas de módem, tarjetas Ethernet, *sockets* TCP/IP, y enlaces de *packet radio*.[3] Puede unirlo a media docena de monitores diferentes y jugar a DOOM con alguien en Australia mientras sigue a satélites de comunicaciones en órbita y controla las luces y termostatos de casa y la grabación en directo de su webcam y navegar en Internet y diseñar circuitos en las demás pantallas. Pero la potencia y complejidad del sistema —las cualidades que lo hacen tan enormemente superior en el aspecto técnico a los demás sistemas operativos— a veces hacen que parezca demasiado formidable para el uso cotidiano.

[3]El *packet radio* es un sistema de comunicación digital basado en las emisoras de radioaficionados. Consiste en la transmisión-recepción, a través de la radio, de señales digitales empaquetadas con reconocimiento de errores en recepción. Su nombre es debido a que envía los datos digitales agrupándolos en pequeños paquetes. El kernel Linux soporta perfectamente este protocolo. [*N. del E.*]

A veces, en otras palabras, sólo quiero ir a Disneylandia.

Mi sistema operativo ideal sería uno que tuviera una interfaz gráfica bien diseñada, que resultase fácil de instalar y usar, pero que incluyera ventanas de terminal desde las que pudiera regresar a la interfaz de línea de comandos, y ejecutar software GNU, cuando tuviera que hacerlo. Hace unos cuantos años, Be Inc. inventó exactamente ese sistema operativo. Se llama BeOS.

Etre

MUCHAS PERSONAS EN EL NEGOCIO de los ordenadores lo han pasado mal para vérselas con Be, Incorporated, por el simple motivo de que no parece tener ningún sentido. Se fundó a finales de 1990, lo cual lo hace más o menos contemporáneo de Linux. Desde el principio se ha dedicado a crear un nuevo sistema operativo que es, por su diseño, incompatible con todos los demás (aunque, como veremos, es compatible con Unix en algunos aspectos muy importantes). Si una definición de *celebridad* es la de alguien que es famoso por ser famoso, entonces Be es una anticelebridad. Es famoso por no ser famoso; es famoso por estar condenado. Pero lleva condenado muchísimo tiempo.

La misión de Be podría tener más sentido para los hackers que para otra gente. Para explicar la razón tengo que exponer el concepto de *cruft*,[1] que para los que escriben código es casi tan aberrante como una repetición innecesaria.

Si han estado en San Francisco habrán visto viejos edificios que han sido sometidos a actualizaciones sísmicas, lo cual frecuentemente significa que se han erigido grotescas superestructuras de acero moderno alrededor de edificios construidos, por ejemplo, en un estilo clásico. Cuando lleguen nuevas amenazas —si tenemos otra Era Glacial, por

[1] *Cruft* no suele traducirse. Tampoco aparece en ningún diccionario de inglés, aunque sí en el *Jargon File*, que es el archivo oficioso de la jerga hacker: significa «excesivo», «superfluo», «basura», los hackers lo emplean para referirse en particular al código redundante o sobrante. [*N. del E.*]

ejemplo —podrán construirse capas adicionales de tecnología todavía más alta, a su vez, alrededor de estas, hasta que el edificio original sea como una reliquia en una catedral —un pedazo de hueso amarillento incrustado en media tonelada de un bonito amasijo decorativo.

Se pueden tomar medidas análogas para hacer que viejos sistemas operativos renqueantes sigan funcionando. Se hace todo el tiempo. Remendar un viejo sistema operativo desgastado debiera verse simplificado por el hecho de que, a diferencia de los viejos edificios, los sistemas operativos no tienen ningún mérito estético o cultural que les haga intrínsecamente dignos de salvarse. Pero en la práctica no funciona así. Si trabajan con un ordenador, probablemente hayan personalizado su escritorio, el entorno en el que se sientan a trabajar cada día, y se han gastado mucho dinero en software que funciona en ese entorno, y han dedicado mucho tiempo a familiarizarse con el modo en que todo funciona. Esto lleva mucho tiempo, y el tiempo es dinero. Como ya mencioné, el deseo de simplificar las interacciones con las tecnologías complejas a través de la interfaz, y de rodearse de enanitos de jardín y figuritas de Lladró virtuales, es natural y omnipresente —presumiblemente una reacción contra la complejidad y formidable abstracción del mundo informático—. Los ordenadores nos dan más opciones de las que realmente queremos. Preferimos elegir una sola vez, o aceptar la configuración por defecto que nos dan las compañías de software, y dejar las cosas tranquilas. Pero cuando un sistema operativo se cambia, todo se desmadra.

El usuario medio de ordenador es un anticuario tecnológico al que realmente no le gusta que las cosas cambien. Es un profesional urbano que acaba de comprarse un precioso chalet adosado y está poniendo los muebles y la decoración, y reorganizando las alacenas, de tal modo que todo esté bien. Si es necesario que una banda de ingenieros hurguen en el sótano reforzando los cimientos para que puedan soportar la nueva bañera de hierro con patas, metiendo nuevos cables y tuberías en las paredes para instalar electrodomésticos modernos, bueno, que así sea —los ingenieros son baratos, al menos cuando millones de usuarios de sistemas operativos se reparten el coste de sus servicios.

Igualmente, a los usuarios de ordenador les gusta tener el último Pentium, y poder navegar por la red, sin alterar las cosas que les hacen sentir como si supieran qué demonios está pasando. A veces esto resulta posible, de hecho. Añadir más RAM al sistema es un buen ejemplo de una actualización que probablemente no estropee nada.

Por desgracia, muy pocas actualizaciones son así de pulcras y sencillas. Lawrence Lessig, que fue durante un tiempo Maestro Especial en el pleito antimonopolio del Ministerio de Justicia contra Microsoft, se quejaba de que había instalado Internet Explorer en su ordenador, y al hacerlo había perdido toda su lista de páginas favoritas (su lista personal de señales que usaba para navegar por el laberinto de Internet). Era como si hubiera comprado un nuevo juego de llantas para su coche y luego, al marcharse del taller, descubriera que, debido a algún inescrutable efecto colateral, todas las señales y mapas de carreteras del mundo hubieran sido destruidos. Si es como la mayoría de nosotros, habría gastado un montón de esfuerzo en compilar esa lista de favoritos. Este es sólo un pequeño ejemplo del tipo de problema que pueden provocar las actualizaciones. Los sistemas operativos viejos y desvencijados tienen valor en el sentido básicamente negativo de que los nuevos nos hacen desear no haber nacido.

Todos los apaños y remiendos que tienen que hacer los ingenieros para proporcionarnos los beneficios de la nueva tecnología sin forzarnos a pensar en ello, o a cambiar nuestras costumbres, producen un montón de código que, con el tiempo, se convierte en un gigantesco pegote de chicle, engrudo, hilo de embalaje y cinta aislante que rodea a todo sistema operativo. En la jerga de los hackers, se llama *cruft.* Un sistema que tiene muchas, muchas capas se describe como *crufty,* «cruftoso». Los hackers detestan hacer las cosas dos veces, pero cuando ven algo cruftoso, su primer impulso es arrancarlo, tirarlo y empezar de nuevo.

Si Mark Twain volviera a San Francisco hoy y estuviera en uno de estos viejos edificios sísmicamente restaurados, le parecería igual, con todas las puertas y ventanas en el mismo sitio: pero si saliera a la calle, no lo reconocería. Y —si hubiera vuelto con su ingenio intacto— podría cuestionar si había merecido tomarse tanta molestia para salvar ese edificio. En algún momento, hay que hacerse la pregunta: ¿merece la pena, o deberíamos derribarlo y levantar uno bueno? ¿Deberíamos poner otra ola humana de ingenieros a estabilizar la Torre Inclinada de Pisa, o deberíamos sencillamente dejar que la dichosa torre se caiga y construir una que no esté mal hecha?

Como la restauración de un viejo edificio, el *cruft* siempre parece una buena idea cuando se ponen las primera capas —sólo es mantenimiento rutinario, una gestión sólida y prudente—. Este resulta especialmente cierto cuando (por así decir) nunca se baja al sótano, ni se mira detrás del encofrado. Pero cuando eres un hacker que se pasa todo el tiempo mirando las cosas desde ese punto de vista, el *cruft* es fundamentalmente asqueroso, y no puedes evitar querer sacarlo a golpe de escoplo. O, mejor aún, sencillamente salir del edificio —dejar que la Torre Inclinada de Pisa se caiga— y ponerse a construir una nueva *que no se incline*.

Durante mucho tiempo, resultaba obvio a Apple, a Microsoft y a sus clientes que la primera generación de sistemas operativos con interfaz gráfica estaba condenada, y que acabarían por ser desechada en favor de sistemas completamente nuevos. A finales de los ochenta y principios de los noventa, Apple realizó unos cuantos esfuerzos estériles para crear nuevos sistemas operativos posteriores a MacOS, tales como Pink y Taligent. Cuando estos esfuerzos fallaron, realizaron un nuevo proyecto llamado Copland, que también falló. En 1997 coquetearon con la idea de adquirir Be, pero en vez de eso adquirieron NeXT, que tiene un sistema operativo llamado NextStep que es, de hecho, una variante de Unix. A medida que estos esfuerzos se sucedían y fracasaban, uno detrás de otro, los ingenieros de Apple, que eran de los mejores en la profesión, no dejaban de añadir capas de *cruft.* Estaban tratando de convertir la pequeña tostadora en una máquina multitarea y apta para Internet, y les salió sorprendentemente bien durante cierto tiempo: algo así como el héroe de una película que cruza un río en la selva saltando sobre los lomos de los cocodrilos. Pero en el mundo real los cocodrilos terminan por acabarse, o bien pisas a uno realmente listo.

Hablando de ello, Microsoft abordó el mismo problema de un modo considerablemente más ordenado, creando un nuevo sistema operativo llamado Windows NT, que está explícitamente pensado para ser un competidor directo de Unix. NT quiere decir *New Technology*, «Nueva Tecnología», lo cual podría leerse como un rechazo del *cruft*. Y de hecho NT tiene la reputación de ser mucho menos cruftoso de lo que acabó siendo MacOS; en un momento dado, la documentación necesaria para escribir código en el Mac llenaba algo así como 24 carpetas. Windows 95 era, y Windows 98 es, cruftoso porque tienen que ser retroactivamente compatibles con los anteriores sistemas operativos de Microsoft. Linux trata con el problema del cruft del mismo modo en que los esquimales trataban con sus jubilados: si insistes en usar viejas versiones de software Linux, antes o después acabarás por encontrarte flotando por el Estrecho de Bering en un iceberg cada vez más pequeño. Pueden permitírselo porque la mayor parte del software es gratuito, así que no cuesta nada descargarse versiones actualizadas, y la mayor parte de los usuarios de Linux son morlocks.

La gran idea detrás de BeOS fue partir de una hoja de papel en blanco y diseñar un sistema operativo del modo correcto. Y eso es exactamente lo que hicieron. Esto era obviamente una buena idea desde el punto de vista estético, pero no es un buen plan de negocios. Algunas personas que conozco en el mundo GNU/Linux están molestos con Be por haber emprendido esta aventura quijotesca cuando sus formidables capacidades podían haber contribuido a extender Linux.

De hecho, no tiene ningún sentido hasta que uno recuerda que el fundador de la compañía, Jean-Louis Gassée, es de Francia —un país que durante muchos años mantuvo su propia versión separada e independiente de la monarquía inglesa en la corte de St. Germain, con cortesanos, ceremonias de coronación, religión estatal, y política exterior—.

Ahora, la misma fastidiosa pero admirable testarudez que nos dio a los jacobinos, la *force de frappe,* el Airbus y las señales de *Arrêt* en Quebec, nos ha dado un sistema operativo realmente chulo. ¡Me cisco en vosotros, perros anglosajones!

Crear completamente un sistema operativo a partir de la nada, sencillamente porque ninguno de los existentes era exactamente adecuado, me pareció un acto de tal chulería que me vi compelido a apoyarlo. Me compré un BeBox en cuanto pude. El BeBox era un ordenador de procesador dual, con chips de Motorola fabricados específicamente para ejecutar el BeOS; no podía ejecutar ningún otro sistema operativo. Por eso lo compré. Sentí que era un modo de quemar las naves. Su característica más distintiva son dos pilotos en el panel frontal que suben y bajan como tacómetros para dar la sensación de lo duro que está trabajando cada procesador. Me pareció elegante, y además, calculé que en cuanto la compañía quebrara en unos poco meses, mi BeBox sería un valioso objeto de coleccionista.

Han pasado dos años y estoy escribiendo esto en mi BeBox. Los pilotos (*Das Blinkenlights*, como los llaman en la comunidad Be) parpadean alegremente junto a mi codo derecho mientras pulso las teclas. Be, Inc. sigue en activo, aunque dejaron de fabricar BeBoxes casi inmediatamente después de que yo comprara el mío. Tomaron la triste pero probablemente bastante acertada decisión de que el hardware era mal negocio, y se llevaron el BeOS a Macintosh y a clones del Mac. Puesto que estos usan el mismo tipo de chips Motorola que usaba el BeBox, no resultó especialmente difícil.

Muy poco tiempo después, Apple estranguló a los fabricantes de clones del Mac y restauró su monopolio del hardware. Así que durante un tiempo Apple fabricó los únicos nuevos ordenadores que podían ejecutar BeOS.

A estas alturas Be, como Spiderman con su sentido arácnido, había desarrollado un agudo sentido de cuándo iban a aplastarlo como a un bicho. Incluso aunque no lo hubieran tenido, la idea de depender de Apple —tan frágil y sin embargo tan letal— para seguir existiendo hubiera espantado a cualquiera. Emprendiendo su propia aventura de salto de cocodrilos, trasladaron el BeOS a chips de Intel (los mismos chips que usan los ordenadores de Windows). Y justo en el momento adecuado, cuando Apple lanzó su nuevo hardware, basado en el chip G3 de Motorola, mantuvieron en secreto los datos técnicos que los ingenieros de Be habrían necesitado para ejecutar el BeOS en aquellos ordenadores. Esto habría matado a Be como una bala entre ceja y ceja, de no haber dado ya el salto a Intel.

Así que ahora el BeOS se puede ejecutar en una gama increíblemente variada de hardware: BeBoxes, viejos Macs y huérfanos clones del Mac y ordenadores Intel para uso con Windows. Por supuesto estos últimos son ubicuos y sorprendentemente baratos hoy en día, así que pareciera que los problemas de hardware de Be han llegado a su fin. Algunos hackers alemanes incluso han creado un sustituto de *Das Blinkenlights:* es un circuito que se puede enchufar a máquinas compatibles con PC que ejecuten BeOS. Lleva los pilotos en forma de tacómetro que habían sido una característica tan popular del BeBox.

Mi BeBox ya empieza a estar viejo, como les pasa a todos los ordenadores cada dos años o así y, antes o después, tendré que sustituirlo por un ordenador Intel. Incluso después de eso, sin embargo, podré seguir usándolo. Porque, inevitablemente, alguien ya ha llevado Linux al BeBox.

En cualquier caso, BeOS tiene una interfaz gráfica extremadamente bien pensada, construida sobre un marco tecnológico sólido. Se basa desde el principio en modernos principios del software orientado a objetos. El software del BeOS consiste en entidades cuasiindependientes de software llamadas objetos, que se comunican enviándose mensajes unas a otras. El sistema operativo mismo está compuesto de tales objetos, y funciona como una especie de oficina de correos o Internet a través de la cual se mandan mensajes de objeto a objeto. El sistema operativo tiene múltiples hilos, lo cual quiere decir que como todos los demás sistemas operativos modernos puede caminar y mascar chicle a la vez; pero les da a los programadores un montón de poder sobre la generación y eliminación de hilos, o subprocesos independientes. También es un sistema operativo multiprocesador, lo cual significa que se le da inherentemente bien ejecutarse en ordenadores con más de una CPU (Linux y Windows NT también hacen esto con eficacia).

Para este usuario, un punto fuerte de BeOS es su aplicación incrustada «Terminal», que permite abrir ventanas equivalente a las ventanas `xterm` de Linux. En otras palabras, la interfaz de línea de comandos está disponible si la quieres. Y debido a que BeOS sigue cierto estándar llamado POSIX, puede ejecutar la mayor parte del software GNU. Es decir, que la inmensa cantidad de software de línea de comandos desarrollado por los de GNU funciona en una ventana terminal de BeOS sin problemas. Esto incluye las herramientas de desarrollo de GNU —el compilador y el enlazador—. E incluye todos los programitas de utilidades. Estoy escribiendo esto usando una especie de moderno editor de texto llamado Pe, escrito por un holandés llamado Maarten Hekkelman, pero cuando quiero averiguar cuánto he escrito, paso a una ventana terminal y ejecuto wc.

Como sugiere el informe de fallo que cité antes, la gente que trabaja para Be, y los programadores que escriben el código de BeOS, parecen divertirse más que sus homólogos en otros sistemas operativos. También parecen ser más diversos en general. Hace un par de años fui a una universidad local para asistir a la conferencia de unos representante de Be. Fui porque asumí que el auditorio estaría desierto, y me pareció que merecían un público de al menos una persona. De hecho, acabé de pie en el pasillo, pues había cientos de estudiantes llenando la sala. Era como un concierto de rock. Uno de los dos ingenieros de Be en el escenario era negro, lo cual desgraciadamente es algo muy raro en el mundo de la alta tecnología. El otro denunció animadamente el *cruft*, y cantó las loas de BeOS por sus cualidades libres de *cruft*, y de hecho acabó diciendo que en diez o quince años, cuando BeOS se volviese tan cruftoso como MacOS y Windows95, sería hora de tirarlo y crear un nuevo sistema operativo a partir de la nada. ¡Dudo que esto fuera política oficial de Be, pero impresionó a todo el mundo en la sala! A finales de los ochenta, el MacOS fue, durante un tiempo, el sistema operativo de los artistas en la onda y los hackers —y BeOS parece tener el potencial para atraer a la misma gente hoy—. Las listas de correo de Be están llenas de hackers con nombres como Vladimir y Olaf y Pierre, poniéndose a parir unos a otros en quebrado tecnoinglés.

La única pregunta real acerca de BeOS es si está condenado o no.

Últimamente, Be ha respondido a la cansina acusación de que están condenados con la aseveración de que BeOS es un «sistema operativo multimedia» fabricado para los creadores de contenidos multimedia, y por tanto no entra en competición con Windows. Esto es un poco ingenuo. Por volver a la analogía de los concesionarios de coches, es como si el dueño de la tienda de batmóviles afirmara que en realidad no compite con los demás porque su coche puede ir tres veces más rápido y además puede volar.

Be tiene una oficina en París y, como mencioné, la conversación en las listas de correos sobre Be tiene un sabor fuertemente europeo. Al mismo tiempo se han esforzado mucho por hallar un nicho en Japón, e Hitachi acaba de empezar a meter BeOS en sus PC. Así que, si tuviera que lanzar una predicción, yo diría que están jugando al go mientras Microsoft juega al ajedrez. Por el momento, se mantienen lejos de la posición abrumadoramente fuerte de Microsoft en Norteamérica. Están tratando de asentarse en los bordes del tablero, por así decir, en Europa y Japón, donde la gente puede estar más abierta a sistemas operativos alternativos, o al menos puede ser más hostil a Microsoft, que en los Estados Unidos.

Lo que mantiene a Be trabado en este país es el hecho de que a la gente inteligente le da miedo parecer imbécil. Corres el riesgo de parecer ingenuo cuando dices: «He probado BeOS, y esto es lo que opino.» Parece mucho más sofisticado decir: «Las probabilidades de que Be encuentre un nicho en el mercado altamente competitivo de los sistemas operativos se aproximan a cero.» Es, en jerga técnica, un problema de mente compartida. Y en el negocio de los sistemas operativos, la mente compartida es algo más que una mera cuestión de RP; tiene efectos directos sobre la tecnología misma. Todos los enredos periféricos que pueden enchufarse a un ordenador personal —las impresoras, escáneres, interfaces de PalmPilot y Lego Mindstorms— precisan de unos elementos de software llamados controladores o *drivers*. Igualmente, las tarjetas de vídeo y (en menor medida) los monitores necesitan *drivers.* Incluso los diferentes tipos de placas madre en el mercado se relacionan con el sistema operativo de diferentes maneras, y se precisa un código distinto para cada una. Todo este código específico para el hardware no sólo ha de escribirse, sino también probarse, mejorarse, actualizarse, mantenerse, y repararse. Debido al hecho de que el mercado del hardware se ha vuelto tan enorme y complicado, lo que realmente determina el destino de un sistema operativo no es lo bueno que sea técnicamente, ni cuánto cueste, sino la disponibilidad del código específico del hardware. Los hackers de Linux tienen que escribir ese código ellos mismos, y han mantenido una rapidez asombrosa. Be, Inc. tiene que escribir todos sus propios *drivers,* aunque a medida que BeOS ha ido ganando impulso programadores independientes han empezado a contribuir con *drivers,* que están disponibles en el sitio web de Be.

Pero Microsoft lleva ventaja, de momento, porque no tiene que escribir sus propios *drivers.* Cualquier fabricante de hardware que lance hoy día una nueva tarjeta de vídeo o un nuevo periférico al mercado sabe que será invendible a menos que incluya el código específico del hardware que haga que funciones con Windows, y así todos los fabricantes de hardware han aceptado la carga de crear y mantener su propia biblioteca de *drivers.*[2]

---

[2] A finales de 2001, Be Inc. cerró sus puertas y vendió su propiedad intelectual a Palm, incluido BeOS. Unos cuantos días antes de anunciarse la venta, un grupo de hackers iniciaron el «OpenBeOS Project» (`http://open-beos.sourceforge.net`), un proyecto dedicado a re-crear, y luego extender, un clon libre de BeOS. [*N. del E.*]

Mente compartida

La afirmación del Gobierno de los EE.UU. de que Microsoft tiene el monopolio del mercado de sistemas operativos puede ser la aseveración más obviamente absurda jamás presentada por la mente legal. Linux, un sistema operativo técnicamente superior, se regala, y BeOS está disponible por un precio nominal. Esto es sencillamente un hecho, que hay que aceptar te guste o no Microsoft.

Microsoft es realmente grande y rica, y si hay que creer a algunos de los testigos del Gobierno, no son muy agradables. Pero la acusación de monopolio sencillamente carece de sentido.

Lo que realmente está pasando es que Microsoft se ha hecho, de momento, con cierta ventaja: dominan la competición por la mente compartida, así que cualquier fabricante de hardware o software que quiera ser tomado en serio se siente obligado a fabricar un producto que sea compatible con sus sistemas operativos. Dado que los fabricantes de hardware escriben *drivers* compatibles con Windows, Microsoft no tiene por qué escribirlos; a todos los efectos, los fabricantes de hardware están añadiendo nuevos componentes a Windows, convirtiéndolo en un sistema operativo más capaz, sin cobrar a Microsoft por sus servicios. Es una buena posición en la que estar. El único modo de combatir a tal adversario es tener un ejército de programadores altamente competentes que escriban drivers equivalentes de forma gratuita, que es lo que hace Linux.

Pero la posesión de esta ventaja tecnológica es diferente de un monopolio en cualquier sentido normal de la palabra, porque aquí el dominio no tiene nada que ver con los resultados técnicos o el precio. Los antiguos monopolios de barones ladrones eran monopolios porque controlaban físicamente los medios de producción y/o distribución. Pero en el negocio del software, los medios de producción son los hackers que escriben código, e Internet es el equivalente a los medios de distribución, y nadie afirma que Microsoft controle eso.

Aquí, por el contrario, el dominio se encuentra en las mentes de la gente que compra software. Microsoft tiene poder porque la gente cree que lo tiene. Hace mucho dinero. A juzgar por los recientes procedimientos judiciales en ambos Washingtons, pareciera que este poder y este dinero impelieron a algunos ejecutivos muy peculiares a trabajar para Microsoft, y que Bill Gates debiera haber realizado tests de saliva antes de darles tarjetas de identidad de Microsoft.

Pero este no es el tipo de poder que encaja con cualquier definición normal de la palabra *monopolio,* y no es regulable legalmente. Puede que los tribunales ordenen a Microsoft que haga las cosas de otro modo. Incluso puede que partan la compañía.[1] Pero en realidad no pueden hacer nada respecto del monopolio de la mente compartida, a menos que agarren a cada hombre, mujer y niño en el mundo de-

---

[1]En 1999, el juez federal Thomas Penfield Jackson dictaminó que Microsoft había incurrido en las prácticas monopolistas ilegales de las que se le acusaba, y ordenó una división de la empresa en dos firmas, una que produciría el sistema operativo Windows y otra dedicada a programas de aplicaciones. En 2000, en respuesta a una apelación de Microsoft, el Tribunal Supremo de Estados Unidos anuló el fallo de ese juez federal y remitió el caso al tribunal de la juez Kollar-Kotelly. Después de tres años de asedio judicial, el nuevo gobierno del presidente George W. Bush, a través de su Departamento de Justicia, ofreció una salida fácil a la compañía de Redmond, renunciando a dividirla en dos y liberándoles de la obligación de publicar las especificaciones técnicas del sistema operativo, que hubiera permitido a terceros desarrollar aplicaciones en igualdad de condiciones. Muchos atribuyeron este cambio de actitud y la magnífica sintonía entre Microsoft y la Casa Blanca a las suculentas contribuciones de campaña realizadas por Microsoft a los republicanos. [*N. del E.*]

sarrollado y los sometan a un largo proceso de lavado de cerebro.

El dominio de la mente compartida es, en otras palabras, una cosa muy rara, algo que los creadores de las leyes antimonopolio nunca podrían haberse imaginado. Se parece a uno de esos desquiciados fenómenos modernos de teoría del caos, algo relacionado con la complejidad, en la que un montón de entidades independientes pero conectadas (los usuarios de ordenadores del mundo), tomando sus propias decisiones, según una pocas reglas elementales, generan un enorme fenómeno (el dominio total del mercado por una sola compañía) que no tiene sentido por ningún análisis racional. Tales fenómenos están llenos de puntos pivotales ocultos y enmarañados con extraños bucles de retroalimentación, y no pueden entenderse: los que lo intentan acaban

1. Volviéndose locos
2. Rindiéndose
3. Desarrollando teorías desquiciadas, o
4. Convirtiéndose en consultores sobre teoría del caos muy bien pagados.

Puede que haya una o dos personas en Microsoft lo bastante tontas para creer que el dominio de la mente compartida es una posición estable y duradera. Tal vez eso explica alguno de los chiflados que han contratado en el sector de negocios, los fanáticos que jueces enfurecidos constantemente llevan a los tribunales. Pero la mayoría de ellos deben de tener la inteligencia para comprender que fenómenos como estos son desquiciantemente inestables, y que no se puede decir qué suceso extraño y aparentemente irrelevante podría hacer que el sistema pasara a una configuración radicalmente diferente.

Por expresarlo de otro modo, Microsoft puede estar segura de que el juez Thomas Penfield Jackson no emitirá una orden para que se reprogramen sumariamente los cerebros de todos los habitantes del mundo desarrollado. Pero no hay modo de predecir cuándo la gente decidirá, en masa, reprogramar sus propios cerebros. Esto podría explicar parte del comportamiento de Microsoft, como su política de tener reservas extrañamente grandes de dinero, y la angustia extrema que les entra cuando aparece algo como Java.

Nunca he visto el interior del edificio de Microsoft donde están todos los altos ejecutivos, pero tengo la fantasía de que en los pasillos, a intervalos regulares, hay grandes cajas rojas de alarma atornilladas a las paredes. Cada una contiene un gran botón rojo protegido por un cristal. Un martillo de metal cuelga por una cadena junto a él. Encima hay un gran cartel que dice:

ROMPER EL CRISTAL EN CASO DE DESPLOME
DE LA CUOTA DE MERCADO

No sé qué sucede cuando alguien rompe el cristal y aprieta el botón, pero seguro que sería interesante averiguarlo. Me imagino bancos arruinándose en todo el mundo mientras Microsoft retira sus reservas, y paquetes de billetes de cien envueltos en plástico cayendo del cielo. Sin duda, Microsoft tiene un plan. Pero lo que realmente me gustaría saber es si, a cierto nivel, sus programadores respirarían aliviados si la carga de escribir la Única Interfaz Universal para Todo fuera súbitamente retirada de sus hombros.

El meñique derecho de Dios

EN SU LIBRO *La vida del cosmos*, que todo el mundo debería leer, Lee Smolin da la mejor descripción que he leído nunca de cómo nuestro universo emergió de un equilibrio sorprendentemente preciso de diferentes constantes fundamentales. La masa del protón, la fuerza de la gravedad, el ámbito de la fuerza nuclear débil y unas pocas docenas más de constantes fundamentales determinan por completo qué tipo de universo surgirá de un Big Bang. Si estos valores hubieran sido incluso ligeramente diferentes, el universo habría sido un enorme océano de gas tibio o un nudo caliente de plasma o alguna otra cosa básicamente poco interesante —pura filfa, en otras palabras—. El único modo de obtener un universo que no sea filfa —que tenga estrellas, elementos pesados, planetas y vida— es calcular bien los números básicos. Si hubiera algún ordenador, en algún lugar, que pudiera escupir universos con valores aleatoriamente escogidos para sus constantes fundamentales, por cada universo como el nuestro produciría 10^{229} universos fallidos.

Aunque no me he sentado a hacer el cálculo, a mí esto me parece comparable a la probabilidad de hacer que un ordenador Unix haga algo útil entrando en un `tty` e introduciendo líneas de comando cuando te has olvidado de todas las opciones y palabras clave. Cada vez que tu meñique pulsa la tecla `ENTER`, lo estás intentando. En algunos casos el sistema operativo no hace nada. En otros casos borra todos tus archivos. En la mayoría de los casos simplemente te da un mensaje de error. En otras palabras, obtienes muchas filfas. Pero a veces, si lo haces todo bien, el ordenador rumia durante un rato y luego produce algo como `emacs`. De hecho, genera complejidad, que es el criterio de Smolin para la propiedad de resultar *interesante*.

No sólo eso, sino que además parece que, una vez que vas por debajo de cierto tamaño —mucho más abajo del nivel de los quarks, al ámbito de la teoría de supercuerdas— el universo no puede describirse con la física que se practica desde tiempos de Newton. Si se mira a una escala lo bastante pequeña, se ven procesos que parecen de naturaleza casi computacional.

Creo que el mensaje está muy claro: en algún lugar fuera y más allá de nuestro universo hay un sistema operativo, codificado a lo largo de incalculables periodos de tiempo por algún tipo de demiurgo-hacker. El sistema operativo cósmico usa una interfaz de línea de comandos. Se ejecuta en algo parecido a un teletipo, con montones de ruido y calor; los bits introducidos revolotean a la papelera como estrellas fugaces. El demiurgo está sentado frente a su teletipo, introduciendo una línea de comando tras otra, especificando los valores de las constantes fundamentales de la física:

```
root@god:~# universe -G 6.672e-11 -e 1.602e-19 \
-h 6.626e-34 --protonmass 1.673e-27....
```

y cuando acaba de escribir la línea de comandos, su meñique derecho titubea sobre la tecla `enter` durante uno o dos eones, preguntándose qué va a pasar; luego cae —y el *boom* que se oye es otro Big Bang.

Ese sí que es un sistema operativo chulo, y si estuviera disponible en Internet (libre, por supuesto) todos los hackers del mundo se lo descargarían enseguida y se pasarían toda la noche enredando, escupiendo universos a diestro y siniestro. La mayoría serían universos bastante sosos pero algunos serían simplemente asombrosos. Porque los que esos hackers estarían tratando de conseguir sería algo mucho más ambicioso que un universo con unas pocas estrellas y galaxias. Cualquier hacker corrientucho podría hacer eso. No, el modo de labrarse una gran reputación en Internet sería ser tan bueno con la línea de comandos que los universos desarrollaran vida espontáneamente. Y una vez que el modo de conseguir eso se convirtiera en un conocimiento común, esos hackers irían más allá, tratando de hacer que sus universos desarrollaran el tipo adecuado de vida, tratando de hallar el único cambio en el n-ésimo lugar decimal de una constante física que nos daría una Tierra en la que, pongamos, aceptaran a Hitler en la Escuela de Bellas Artes después de todo, y acabara como artista callejero con curiosas opiniones políticas.

Incluso si esa fantasía se volviera realidad, sin embargo, la mayoría de los usuarios (incluyéndome a mí mismo, algunos días) no querrían molestarse en aprender todos esos arcanos comandos, y pugnar con todos los fracasos; unos pocos universos fallidos realmente pueden atiborrarte el trastero. Tras pasar un rato introduciendo líneas de comando y pulsando la tecla `enter` y engendrando aburridos universos fallidos, empezaríamos a desear que hubiera un sistema operativo que fuera todo lo contrario: un sistema operativo que tuviera la potencia para hacerlo todo: para vivir nuestra vida por nosotros. En este sistema operativo, todas las decisiones posibles que tuviéramos que tomar habrían sido predeterminadas por astutos programadores, y condensadas en una serie de cuadros de diálogo. Pulsando en botones de radio podríamos escoger de entre opciones mutuamente excluyentes (HETEROSEXUAL/HOMOSEXUAL). Las columnas de cuadritos a tachar nos permitirían seleccionar las cosas que quisiéramos en nuestra vida ((CASARSE/ESCRIBIR LA GRAN NOVELA AMERICANA) y para las opciones más complicadas podríamos rellenar cuadritos de texto (NÚMERO DE HIJAS: NÚMERO DE HIJOS).

Incluso esta interfaz de usuario empezaría a parecer tremendamente complicada pasado un tiempo, con tantas opciones y tantas interacciones ocultas entre opciones. Se volvería casi inmanejable —el problema del doce parpadeante de nuevo—. La gente que nos la proporcionó tendría que proporcionar también asistentes y plantillas, dándonos unas pocas vidas por defecto que pudiéramos usar como base para diseñar la nuestra. Lo más probable es que estas vidas por defecto le parecieran bastante buenas a la mayoría de la gente, de todas formas, así que les fastidiaría enredar con ellas por miedo a empeorarlas. Así que, tras unas pocas versiones, el software sería aún más simple: lo iniciarías y te presentaría un cuadro de diálogo con un único botón grande en medio etiquetado: `vivir`. Una vez pulsaras ese botón, empezaría tu vida. Si algo fuese mal, o no respondiese a tus expectativas, podrías quejarte al Departamento de Atención al Cliente de Microsoft. Si te atendiese un empleado de atención al público, te diría que tu vida iba bien, que no le pasaba nada y que en cualquier caso irá mucho mejor con la próxima actualización. Pero si insistieras, y te identificaras como *avanzado*, podrías hablar con un ingeniero de verdad.

¿Qué diría el ingeniero, una vez hubieras explicado tu problema y enumerado todas las insatisfacciones de tu vida? Probablemente te diría que la vida es una cosa muy difícil y complicada; que ninguna interfaz puede cambiar eso; que cualquiera que crea lo contrario es un imbécil; y que si no te gusta que escojan por ti, deberías empezar a elegir por ti mismo.