

Unix no fue diseñado para evitar que sus usuarios hicieran cosas estúpidas, pues eso también prevendría que hicieran cosas inteligentes.
-Doug Gwyn, Ballistic Reseach Labs.

UNIX: Un ícono cultural, histórico... y de vanguardia.

Roberto Hoyos M.
Mayo de 2006



Cortesía de Phil McCrea, Australia.

Unix llegó para quedarse. Es un actor que sabe reinventarse a sí mismo; en eterna caravana puede desempeñarse en cualquier escenario, llamémosle hardware; proponer estándares y dar nacimiento a estilos de programación, como C; es un fenómeno que está más allá del tiempo (nació en 1969 y todo indica que durará muchos, muchos más años, a pesar de tantos y tantos vaticinios sobre su pronto reemplazo); está más allá de arquitecturas y propietarios, pues es un histrión que lo mismo aparece ante gurús que amateurs, gigantes (configuraciones) y pigmeos (dispositivos *handheld*). Y sin pretender la apología de

Unix, también es preciso decir que tiene aspectos no tan positivos. Es una *prima donna* caprichosa que exige que su audiencia sea exquisita, al menos, para poder apreciarlo en toda su grandeza y poder entender sus sutilezas; tiene muchos imitadores y no es fácil de manejar, pues es de juicios definitivos y nunca dice por favor, pero a pesar de sus detractores se mantiene muy saludable. Unix se ha convertido en una leyenda, en un ícono cultural.

Dice Steven Raymond, en su libro *The Art Of Unix Programming*, que parte de su sobrevivencia se debe a la filosofía misma de diseño de Unix¹, que define como flexible, simple, modular y económica –por mencionar algunas características, aun contra conceptos de simplicidad y/o el hecho de “ser correcto” (*correctness*). La otra parte que ha augurado su supervivencia es una inteligente estrategia de *marketing*. Hay, por supuesto, quienes ven en esta filosofía de diseño *La Falla* fundamental (la mayúscula es intencional), lo cual es perfectamente entendible, si consideramos que son usuarios que, como Richard Gabriel, provienen de ambientes de programación mucho más sofisticados, como LISP, donde es indispensable no dejar nada al azar. Pienso en la alegoría de Borges y Cervantes, dos grandes escritores de la lengua española cuyos estilos de escritura son tan diferentes como sus audiencias y su legado. Borges gustaba de escribir *la página perfecta*, aquella en la que palabras y oraciones tengan una cadencia tan armoniosa que retirar una de ellas equivaldría a derrumbar toda la composición (muy del estilo de la programación en LISP, digamos) contra un Cervantes cuya obra ha sobrevivido a tantas interpretaciones, cambios y lectores que, con todo, ha mantenido su mensaje (el estilo de la simplicidad, el estilo UNIX). Al final del día ¿cuál es mejor? Imposible decir.

Quienes sí pueden aportar debilidades en UNIX, destaco a Simson Garfinkel, editor del libro *The Unix Haters Book*, son implacables: los archivos en UNIX no tienen estructura más allá del nivel de byte, una vez borrado un archivo se borra para siempre, el modelo de seguridad es muy primitivo –hay que notar que UNIX nació bajo necesidades particulares de la era: máquinas de memoria limitada y periféricos no tan veloces como los actuales, y que el manejo de colas de trabajos es lento y torpe, pues deja algunos cambios de estado a nivel de aplicación. Unix es tan técnico, que prácticamente deja todas las decisiones de su comportamiento al usuario. Necesariamente estas fallas se llevan al nivel de administración. Cito, “Las herramientas de administración de Unix llevan más tiempo para

¹ *Worse Is Better*, también llamado *New Jersey Style*. Cabe notar el carácter antagónico de este estilo contra el *MIT style*, que pugna por programas cada vez más correctos a riesgo de no ser tan sencillos.

usar de lo que ahorran”² o “Muy pocas versiones de Unix permiten hacer nada más que trivialidades sin tener que ir a la interface de teletipos de los 70’s”³.

Es imposible no ser partícipe de la controversia, pues ambos bandos tienen razón: ni los diseñadores mismos pudieron prever el enorme éxito que tendría su sistema operativo. Nacido en los 70’s, es obvio que arrastrará cierta filosofía de la época (como el hecho de que los archivos son sólo bytes), por mencionar una, o las interfaces de texto, tan vilipendiadas hoy en día.

Conviene estudiar su historia, para entender cómo esto ha influenciado su desarrollo y evolución. Nació a partir de *Multics*, que en su tiempo fue un proyecto para tener un sistema robusto que manejara grandes cantidades de información. No es sorpresa para nadie que originalmente *Multics* fuera soportado por iniciativa de ARPA (agencia de proyectos de investigación avanzada, muy ligada al Departamento de Defensa de EE.UU.), pero cuando AT&T abandonó el proyecto porque se retrasó en la agenda, tres sujetos de su personal diseñaron un compilador, un sistema de archivos y un kernel pequeñísimo para una máquina PDP-7 con el mundanamente idealista propósito de permitirles jugar un programita llamado *Space Travel*. Luego fue necesario utilizar ese sistema en otras máquinas, y se vio que por obra misma de su diseño era fácil exportarlo. De igual manera fueron creciendo las necesidades por aplicaciones que pudieran ejecutarse en ese sistema operativo y fueron agregándose a Unix. Que hoy sea tan fácil pensar en programas portables es debido a Unix, pues antes los programas estaban ligados íntimamente a la arquitectura de las máquinas en las que trabajaban. Y sus usuarios eran programadores que muchas veces tenían que programar sus propias aplicaciones ellos mismos. Unix fue para ellos como maná caído del cielo. Claro, al ser programadores gustaban de usar su experticia como mejor les conviniera, y como sucede con las mentes brillantes, no pueden comprender por qué a los demás se les dificultan las operaciones que para ellos son tan triviales: por ello Unix fue diseñado con un alto nivel de decisiones finales sobre el usuario, no el programa. Conceptos como *Plug and Play* eran impensables. Es decir, Unix trabaja

² Simson Garfinkel, *The Unix Haters Book*, capítulo intitulado Deficient by Design.

³ Idem.

bajo la asunción de que el usuario era un experto y sabía lo que hacía. Pero ese mismo poder y flexibilidad llevaban a Unix a convertirse en un juguete de expertos, hackers y *geeks*. A razón de ese entusiasta uso de los expertos fue cambiando también, para ajustarse a nuevas necesidades. La lista de versiones de Unix es extensa hasta el punto de mezclarse con Linux. De hecho usaré el término indistintamente, ya que uno y otro hoy en día refieren el mismo concepto. ¿Cuál es mejor? Depende de tus necesidades. Hay Linux tan pequeños que caben en una memoria flash, como DSL (*Damn Small Linux*), *Puppy Linux* o *Vector Linux*. Hay otros orientados a *workstations*, como *Gentoo* o *Debian*, y otros más robustos que pueden servir para administración, como *FEDORA* o *AIX*(que no es Linux, propiamente), para servidores de red.

Pero ya que hemos tocado el tema, conviene hablar de la administración de Unix. En particular la administración en sistemas de red. A manera de nota: Unix no nació orientado a red, sino a una máquina particular, de un usuario generalmente (aunque se desempeña en multitasking formidablemente), con memoria limitada. Considerando esto, uno de los problemas que más plagan a las redes de Unix son la aparición, y eventual saturación de la red por *core dumps* de programas que fallaron; programas temporales que no lo son; archivos log; retransmisiones de red ilegítimas entre otras... Estos “desechos” navegan en la red y pueden amenazar con consumir el espacio disponible, volver el sistema más lento o simplemente provocar una falla. Se ha descrito el papel de un administrador de Unix (*sysadmin*) como una niñera bien pagada, que debe monitorear constantemente el estado de la red so pena de los efectos antes mencionados. Segunda asunción: los sistemas Unix fueron creados para máquinas que trabajaban durante horas, no durante días o semanas como otros sistemas. Es decir, máquinas que efectuaban un cálculo y punto. Las tareas de hoy exigen que el servidor trabaje *twenty-four-seven*, durante meses, pero en Unix se corre el riesgo de que se acumule basura, existan problemas de memoria y se corrompa el espacio de direcciones. En realidad la tarea de administración es costosa, eso sin mencionar la dificultad para asumir la cantidad de poder de decisión que tiene el administrador y traducirlo en una curva de aprendizaje razonable. Es decir, sí; Unix no es fácil de mantener, para administrarlo con propiedad es necesario un monitoreo constante y ‘maña’ para saber cómo, dónde y cuándo están sucediendo las fallas y por qué razones. Esto implica mucho

de arte y de intuición. Sí; Unix tiene un diseño apropiado para los años '70s y funciona en máquinas actuales. Haciendo humor de la tragedia también podemos decir que, también, debido a eso, Unix seguirá siendo una fuente de trabajo para los *sysadmins*. Hay quienes comparan a Unix como un virus, pues es pequeño, adaptable, tiene la habilidad de manipular los recursos del anfitrión, muta rápidamente, y puede migrar a otro anfitrión...

La pregunta obligada, el corolario incisivo, es, “bueno, si realmente el sistema tiene muchos defectos” –que incluso los gurús reconocen como tales y graves, ¿Cómo es que se ha mantenido? ¿De dónde proviene su éxito? La respuesta no elude, más bien, confronta. Es que ese diseño, ese mismo diseño tan defectuoso, simplista, minimalista y flexible también ha traído no pocas ventajas. Entre ellas la principal es que es un software *abierto*, es decir, pronto a la sugerencia y a la activa colaboración de una comunidad global, en particular a la comunidad hacker y al movimiento del mismo nombre que nació en los '60s y se convirtió en una tendencia mundial; es un sistema muy portable, compatible con TCP/IP (sobre la cual Internet trabaja); y ha permitido el desarrollo de estándares. Por virtud de su popularidad entre la comunidad hacker, y la subsecuente versatilidad a la que ascendió, la IEEE utilizó a Unix para hacer los estándares de *POSIX*, por ejemplo. Lo mismo han sido su jerarquía de archivos, y formatos para archivos binarios. Quizá el más importante concepto en cuestión de estándares es que fue el precursor de C. De hecho el nacimiento de C sucedió a partir de Unix y con ello ha influenciado otros lenguajes de programación como JAVA. También la tendencia es que Unix y sus variantes vayan hacia UTF-8, otra forma de codificación de datos, como ASCII en su tiempo. Unix tiene una importancia histórica, por eso insisto en que se ha convertido en un icono cultural. Quizá los Beatles no componían canciones musicalmente tan sofisticadas, pero no dejó de ser irreverentemente brillante, asombrosamente versátil, y su popularidad se ha mantenido al punto de convertirlos en leyendas, influenciando a generaciones y a la música en general desde entonces.

La cultura hacker, y todo el movimiento “abierto” a la generación, supervisión, mantenimiento y prueba de código se basa en la cooperación. Fueron estos mismos hackers los que fundaron los primeros grupos de usuarios de Unix, para “esparcir la palabra”

(*spread the word*), como evangelizadores de la nueva fe. Richard Stallman, siendo uno de ellos, en broma dice pertenecer a la iglesia de San IGNUcius, de la religión *emaciana*. La mecánica es simple: reunirse para hablar de Unix, cómo usarlo, cómo instalarlo, distribuirlo gratuitamente, generar código, hacer concursos, etc. Ir del FAQ (*frequently asked questions*) al RTFM (*read the fucking manual*). Misioneros de una nueva fe. Finalmente son los usuarios los que definirán hacia dónde va el software, y en ese sentido, cómo ponerlo al día.

¿Y hacia dónde va Unix? La respuesta se llama *Plan 9*. En realidad fue un proyecto de Bell Labs en la década de los '90, que no tuvo el éxito esperado. Es una mejora sustancial de Unix, pero ni aún ese sistema logró desplazar a *la leyenda*. Sin embargo fue un buen ejercicio de mejora, donde se dejaron al descubierto fallas del diseño anterior. Para empezar este “Unix” fue diseñado completamente para trabajo en red, y a juzgar por la documentación de *Plan 9*, fue completamente re-hecho. Aporta conceptos novedosos como el hecho de que cada proceso tenga su *namespace* mutable (es decir, cada proceso puede tener su propia vista de servicios de sistema), un nuevo protocolo de archivos, mucho más simple, la habilidad de realizar conexiones más fácilmente, puede ver archivos eliminados (*dump filesystem*), y soberbiamente puede manejar recursos de otras terminales o de la red misma (*network stack*). En cuestiones de seguridad representa cualquier recurso como archivo, por lo que su acceso se maneja de idéntica manera a las formas tradicionales de archivos. También maneja la eliminación del *superusuario*. Pero cabe notar que modernas versiones de Linux, de hecho, están proponiendo estos cambios en sus sistemas. Hoy por hoy, FreeBSD ya incluye nuevas formas de control de procesos, entre ellos *clone()* en contraposición a *fork()*.

Pero la filosofía Unix y sus tecnicismos corresponden a sólo la mitad de su éxito. La otra mitad está en la forma en que llega a las masas. La FSF (*Free Software Foundation*) hace la distinción en el vocablo: libre como en libertad, no en gratuito. Es decir, Unix/Linux se ha podido comercializar con éxito en el mundo de un tal William Gates. La campaña de marketing establece que cualquiera tiene derecho a la distribución del sistema y puede cobrar por ese servicio. Y aunque el movimiento por la “apertura” del software ha tenido

sus altas y bajas, la tendencia es que Unix siga siendo utilizado para aplicaciones con necesidades particulares y se comercialice. Después de todo, que un software sea gratuito no quiere decir que las necesidades sean genéricas. Dispositivos handheld, empotrados, aplicaciones de red, sistemas dedicados, etc., todos ellos requieren un sistema “a la medida”. Unix tiene un público diverso. Nadie creyó al principio que algo que fuera “abierto” pudiera comercializarse, pero también olvidan que en un principio la limitante era el hardware.

Mal que bien, con sus pros y contras, contra todo pronóstico, Unix se ha mantenido. Representa más que un programa, una filosofía, que no pertenece a nadie porque pertenece a todos. La obra maestra casi nunca pertenece al autor. Mozart ganaba más dirigiendo a la sinfónica que interpretaba sus partituras que por la composición misma. Unix y Linux han tomado el slogan de un pingüinito. Nuestro actor tiene rostro. Paris Hilton usa camisas con el logotipo, *chicas Linux* ofrecen generosos escotes en camisas con pingüinos deformes por lo entallado de sus atributos, *ser nerd* se vuelve *chic*, Linus Torvalds recibe trato de estrella de rock donde quiera que va; los gurús reemplazan a nuestros consejeros espirituales. Sí; Unix/Linux se ha convertido en un fenómeno.

Pleonasmo dialéctico, volvemos a donde todo empezó: en el principio. Para reinventar a Unix se tuvo que volver a Unix. Los imperativos de la era moderna son trabajo en red, mejor manejo de procesos en ambiente multiusuario y más seguridad, consolidando el tramo ya recorrido –no olvidar. Hacia ese camino se orienta el futuro de Unix, y de la computación en general. Los íconos nunca dejan la primera plana, y este actor requiere más que nuevo maquillaje: está listo para seguir cosechando éxitos en este mundo de nuevos gustos y públicos más exigentes y volubles. Unix, una leyenda histórica, una leyenda actual.

Bibliografía Selecta:

- * Eric Steven Raymond, *The Art of Unix Programming*, 2003, [online version]
<http://www.catb.org/~esr/writings/taoup/html/>
- * Simson Garfinkel, *The Unix Haters Book*, 1994, [online version]
<http://research.microsoft.com/~daniel/uhh-download.html>
- * Richard P. Gabriel, Lisp: Good News, Bad News. How to Win Big, publicación del Laboratorio de Inteligencia artificial del MIT,
<http://www.ai.mit.edu/docs/articles/good-news/subsection3.2.1.html>
- * Brian W. Kernighan y Rob Pike, *The Unix Programming Environment*, Bell Labs
<http://cm.bell-labs.com/cm/cs/upe/>
- * *Historia de Unix*, por Bell Labs, Lucent Technologies, serie de artículos sobre el nacimiento, la historia y desarrollo de Unix y biografías de sus creadores.
<http://www.bell-labs.com/history/unix/>
- * Frank G. Fiamingo, *Unix Administration*, University Technology Services, Universidad de Ohio, 1996,
http://wks.uts.ohio-state.edu/sysadm_course/html/sysadm-1.html
- * Plan 9, *Introductory Papers*, Bell Labs,
<http://www.cs.bell-labs.com/sys/doc/>
- * *Man Pages* (páginas de manual), Vector LINUX 5.
- * *Unix Guide*, Guía para elegir Unix a tu medida,
<http://www.unixguide.net/>