

////////////////////////////////////
///
///
///
///
///
///
///
/// ANDREI MOSSO MENDOZA ///
///
/// TECNOLÓGICO DE MONTERREY, CAMPUS CUERNAVACA ///
///
////////////////////////////////////

En este artículo se tratan algunos de los aspectos históricos que hicieron de Unix lo que es ahora. Se menciona la forma en que Unix ha ido evolucionando, afectando la forma en de hacer las cosas. Veremos la forma en que gracias a Unix se han desarrollado estándares y otras herramientas que se han usado extensivamente en la historia de la computación. También se mencionan algunas de las versiones de Unix que más han afectado a la comunidad de la computación y como a su vez la comunidad ha afectado a la sociedad. Veremos algunas de las aplicaciones más comunes que se le han dado a Unix y las tendencias que los sistemas Unix tienen hoy en día.

I ORIGEN Y EVOLUCIÓN DE LOS SISTEMAS UNIX

Unix nació de una promesa que desafortunadamente Multics no pudo cumplir. Se trataba de un servicio de computación interactiva que permitiera a sus múltiples usuarios de manera muy conveniente escribir nuevos programas al vuelo, explorar nuevas posibilidades para los nuevos programas y hacer pruebas mientras se estaba escribiendo. Los integrantes del grupo inicial de investigadores (Thompson, Ritchie, McIlroy y Ossanna) en realidad fueron de los últimos investigadores que estuvieron trabajando sobre el proyecto Multics. De acuerdo con Dennis Ritchie: “Lo que queríamos preservar era no solo un buen ambiente de programación, sino un sistema alrededor del cual se pudiera formar una comunidad. Sabíamos por experiencia que la esencia de la computación colectiva, según provista por máquinas de acceso remoto y tiempo compartido, no es solo escribir programas desde una terminal en lugar de una perforadora, sino fomentar la comunicación de cerca”¹.

AT&T pretendió aprender la lección con el fracaso de Multics y rechazó todas las propuestas de desarrollo de un nuevo sistema operativo, sin embargo el grupo de Thompson continuó con el desarrollo de Unix bajo un proyecto de procesamiento de textos. El proyecto ameritó la compra de una máquina un poco más poderosa que la que tenían, sin embargo no se asemejaba a lo que inicialmente pedían para el desarrollo de un nuevo sistema operativo, pero algo es mejor que nada. Es aquí donde comienza uno de los más importantes elementos de la *filosofía Unix*: “*Keep It Short, Simple, Small, and Self-contained*” (*Mantenlo Corto, Simple, Pequeño y autocontenido*).

Así, el primer código fuente de Unix fue ideado por tres personas e implementado por Ken Thompson en dos días sobre una máquina que había sido diseñada como una terminal remota para una computadora de verdad².

Una de las primeras características que fueron diseñadas para Unix fue el sistema de archivos, que en esencia sigue funcionando igual, basado en un arreglo de *i-nodes* que contenía los permisos de acceso al archivo, el tipo de archivo, el tamaño y una lista de los sectores físicos en los que se contenía al archivo.

Tal vez el control de procesos fue lo que más modificaciones tuvo. En el principio el ciclo principal del sistema era el siguiente:

1. El intérprete de comandos cerraba todos sus archivos abiertos, después abría los archivos especiales de la terminal para entrada y salida estándar.
2. Leía una línea de comando de la terminal.
3. Hacía un enlace hacia el archivo que especificaba el comando, abría el archivo y eliminaba el enlace. Después copiaba un pequeño programa de arranque en la parte superior de la memoria y saltaba a ese punto; este programa de arranque escribía el archivo sobre el código del intérprete de comandos y por último saltaba a la primera línea del comando.

¹ Dennis M. Ritchie (1984). [The Evolution of the Unix Time-sharing System.](#)

² Eric Steven Raymond (2003). [The Art of Unix Programming.](#)

4. El comando hacía su trabajo y al terminar llamaba a *exit*. La llamada a *exit* causaba que el sistema leyera una nueva copia del intérprete sobre el código del comando terminado y entonces saltaba hacia el inicio del intérprete.
-

Tabla 1³

Este diseño probó ser muy problemático, ya que no permitía procesos en segundo plano ni archivos con comandos (*scripts*) para el intérprete, mucho menos *pipes*. Además no había forma de mantener el estado de la sesión, no era posible saber que había pasado antes de ejecutar cierto comando porque después de ejecutar dicho comando quedaríamos frente a una copia nueva del programa intérprete.

A pesar de todos estos problemas, la forma moderna de hacer el manejo de procesos se ideó e implementó unos días más tarde. Era increíble como todo iba encajando con una facilidad que años después caracterizaría a Unix como un sistema operativo muy fácil de modificar y ampliar.

La redirección de la entrada y la salida estándar junto con los *pipes*, aunque no fueron de las características tempranas de Unix, hoy son de las características más prácticas para construir comportamientos complejos a partir de la combinación de programas que hacen cosas muy específicas pero que las hacen bien. Fue McIlroy quien insistentemente propuso la adición de *pipes* a Unix. Los *pipes* permitieron que la salida de un programa se convirtiera en la entrada de otro. Aunque parece una idea muy simple, este paradigma hizo que muchos de los programas subsiguientes a los *pipes* se comportaran como filtros, añadiendo así una forma de programación muy poderosa y que redundaba en la creación de programas más pequeños, claros y simples.

Antes de Unix la mayor parte de la programación de sistemas se hacía en ensamblador para optimizar el uso de los escasos y caros recursos de cómputo, por lo que no había mucha gente que pensara que un sistema operativo escrito en un lenguaje de alto nivel (y por lo tanto más orientado hacia la portabilidad) fuera del todo posible. Como ya era la costumbre, al grupo Unix no le intimidó el hecho de que no tener un hardware muy potente, habrían de hacer lo mejor posible con los recursos que tuvieran disponibles. Un lenguaje de alto nivel facilitaría la tarea de escribir nuevo código para Unix, además de hacerlo mucho más legible y fácil de modificar. Por lo que Ken Thompson comenzó a escribir una versión de Fortran para la computadora PDP-7 con la que contaban; aunque el intento no duró lo suficiente como para completar la tarea, dio origen a un lenguaje nuevo que estaba influenciado por otros lenguajes de programación, pero que era lo suficientemente pequeño como para ser usado en la PDP-7. Estamos hablando del lenguaje B. Aunque no fue el lenguaje más popular, su importancia reside en que sentó las bases sobre las cuales el lenguaje C está fundado.

Es asombroso lo que el proyecto Unix ha influido en el área de la programación de sistemas, ya que el lenguaje C ha sido el más extensamente utilizado hasta nuestros días.

³ Dennis M. Ritchie (1984). [The Evolution of the Unix Time-sharing System](#).

2 LA FAMILIA UNIX

Después de que Unix fue expuesto públicamente en *Communications of the ACM* en 1974 todo mundo (que pudiera costear una computadora que llenara los requerimientos de Unix, como corporaciones, universidades y agencias del gobierno) quería probar el sistema. Gracias a que desde 1958 AT&T tenía prohibido entrar en el negocio de las computadoras por un caso de monopolio Unix no podía ser convertido en un producto, y por lo tanto debía conceder licencias a cuantas personas se lo pidieran. A lo cual Ken Thompson respondió con notas sobre las copias de Unix diciendo “love, ken”. Sin duda fue una pérdida monetaria muy grande en ese momento para AT&T, pero también fue uno de los factores que más propició el crecimiento de Unix, ya que se formó una comunidad en la que fluían ideas desde la industria hacia la academia y de regreso. La corriente de buenas ideas fue tan grande que nadie se preocupaba por patentarlas u ocultarlas y a nadie le importaba que alguien más pudiera usar libremente su trabajo. Esta posibilidad de contribuir a Unix de una forma abierta permitió que se generaran muchas versiones del sistema a continuación se muestran algunas de las vertientes más conocidas:

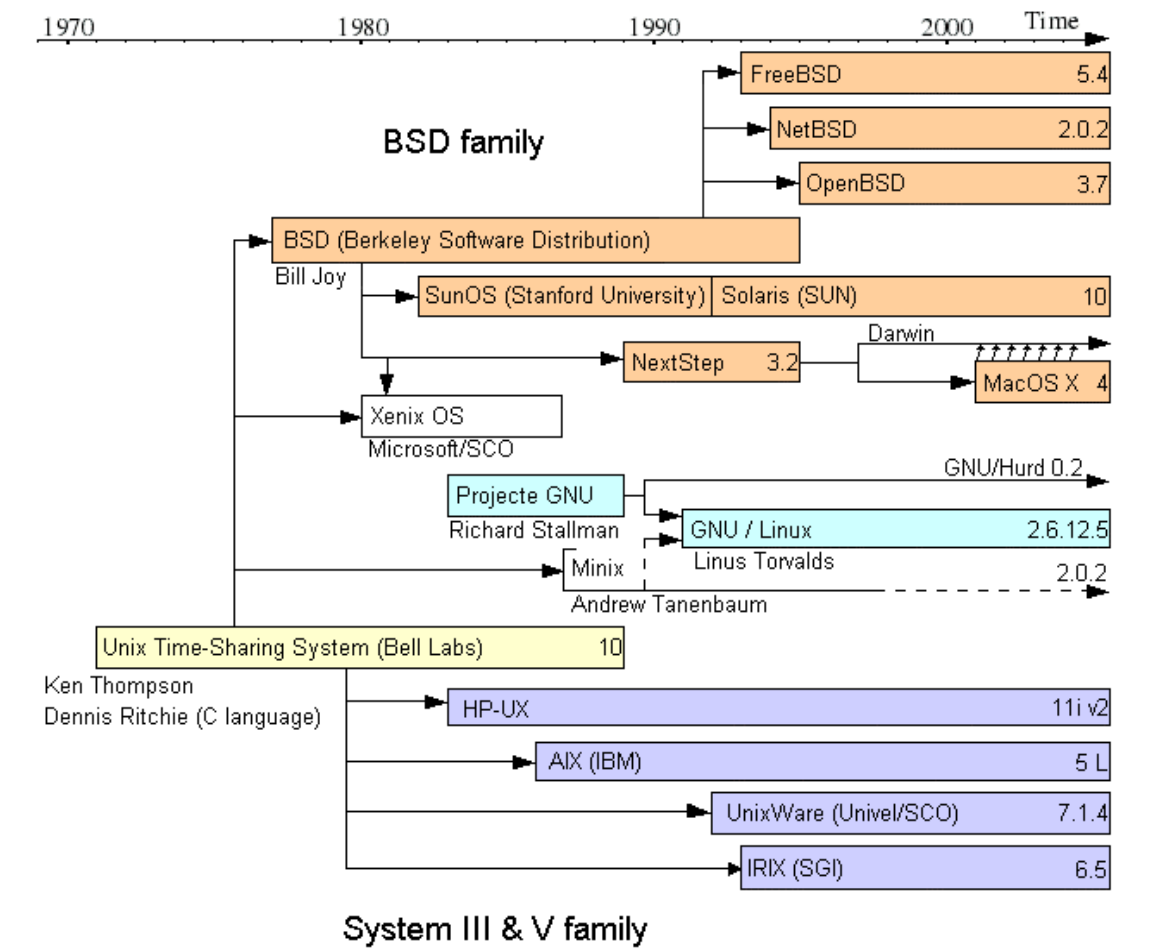


Figura 1⁴

⁴ Ver <http://www.levenez.com/unix/> para obtener un mayor detalle.

3 UNIX: GENERADOR DE ESTÁNDARES

Como ya hemos visto, el lenguaje C fue una necesidad que surgió a partir del desarrollo del sistema operativo y que al igual que Unix no tardó en hacerse muy popular, por lo que surgieron muchas variantes que los usuarios del lenguaje necesitaban. Aunque esto pudiera parecer bueno, en realidad no lo era tanto. Comenzaron a aparecer problemas de compatibilidad que limitaban las bondades de Unix al ser un sistema operativo portable. Esto causó que se formara un comité del *American National Standards Institute* para poner fin a las constantes mutaciones que surgían en el lenguaje. Esto no significaba un estancamiento en el progreso del lenguaje, sino una base común sobre la que todos pudieran desarrollar software compatible.

La situación de gran cooperativismo entre las universidades y la industria era demasiado buena como para no sacar ganancias a costa de Unix, por lo que AT&T después de muchos años de pelea legal logró hacer que se acabara la imposibilidad de comercializar productos relacionados con la computación. Quizás esto fue lo que más ha frenado a Unix en su desarrollo, ya que el ánimo de lucro limitó mucho las posibilidades de seguir aportando a Unix libremente.

Por otro lado, la Universidad de Berkeley no permitiría que todo su trabajo fuera absorbido por AT&T y se desprendió una de las más importantes versiones de Unix: *Berkeley Software Distribution*. Como se muestra en la figura 1, BSD no fue la única vertiente de Unix, lo que propició a que las diferentes versiones tendieran a generar características de incompatibilidad. Algunos esfuerzos por estandarizar Unix fueron hechos con poco éxito por no crear una especificación lo suficientemente general como para abarcar las mejores características de las diferentes versiones.

Fue hasta que AT&T publicó la definición de la interfaz de su *System V* junto con algunas adiciones por parte de BSD que lograron sentar las bases sobre las cuales POSIX está actualmente fundado.

4 SISTEMAS ABIERTOS

En el principio, Unix se caracterizó por distribuirse con el código fuente. A diferencia de muchos otros productos de hardware y de software que se vendían sin la posibilidad de estudiar sus entrañas, y mucho menos la posibilidad de modificarlos a favor de las necesidades de sus usuarios.

4.1 BERKELEY SOFTWARE DISTRIBUTION

Una vez que se dio aquella etapa en la vida de Unix conocida como las guerras Unix (por el problema de las licencias), BSD optó por alinearse hacia preparar un sistema completamente abierto en el que la libertad favoreciera un ambiente de cooperación y en el que cualquiera tuviera la posibilidad de construir sobre BSD algo más elaborado sin tener que partir de cero.

Con el advenimiento de las guerras Unix, muchas de las vertientes del sistema operativo dejaron de ser completamente abiertas. Incluso mucha de la funcionalidad agregada comenzó a ser implantada sobre tecnologías propietarias. Esta situación afectó mucho a los estándares ya establecidos.

Sin embargo, la comunidad Unix, y en particular la IETF (Internet Engineering task Force) asimiló que los estándares existentes y los nuevos no podrían basarse sobre tecnologías propietarias, porque no hay garantía de que éstas se mantengan compatibles hacia atrás, perdiendo así mucho trabajo solo por los intereses económicos de una corporación.

4.2 LINUX

En los primeros años de la década de 1990 la comunidad del software abierto (que estaba principalmente formada por hippies y/o estudiantes universitarios) se estaba preparando para liberarse de una vez de las restricciones (y los precios exagerados) que imponían las grandes corporaciones sobre Unix. Básicamente la propuesta era reescribir el sistema operativo completo para ya no tener nada que ver con el código fuente ni los dueños de Unix.

El proyecto GNU era en realidad muy ambicioso, inclusive no más que un sueño para muchos, y en verdad que lo aparentó durante varios años en los cuales se implementaron muchas de las herramientas existentes en Unix y otras nuevas con la diferencia que éstas eran donadas a la comunidad para su uso libre. El proyecto (aún hoy) incluye la implementación de un kernel completamente independiente, pero compatible. Para lograrlo Richard Stallman y sus seguidores se basaron en las especificaciones establecidas en estándares como POSIX.

Sin duda alguna se trata de un proyecto sublime en cuanto a su objetivo. Sin embargo no todo en la vida se logra a base de buenos deseos. Por algunos años, el proyecto GNU, parecía no tener la capacidad para lograr la independencia de Unix, hasta que en 1991 Linus Torvalds anunció el proyecto Linux, que finalmente se convirtió en la pieza faltante para echar a andar un GNU con muchas aplicaciones, pero sin una base propia para ser utilizadas.

Aún con el kernel provisto por Torvalds (que por cierto todavía no era un reemplazo digno para el núcleo de Unix) tomó algunos años más (hasta la explosión del Internet) para lograr consolidarse como una alternativa real.

5 LAS MEJORES VERSIONES DE UNIX

Existen tres vertientes de Unix que son diferentes no sólo por sus dueños/creadores y sus implementaciones, sino que llevan consigo una ideología diferente y representan a diferentes culturas que han estado entremezcladas alrededor de Unix desde sus comienzos. No me atrevería a decir que alguna de estas culturas sea superior a otra, pues todas han contribuido (de una forma u otra) al desarrollo de lo que los sistemas *Unix-like* representan.

5.1 SYSTEM V

De esta versión descienden muchas de las plataformas Unix propietarias. Las licencias son muy caras y en muchos casos, aunque se tuviera una copia del software se necesita una plataforma de hardware también propietaria o no muy común para los usuarios normales.

Algunas de estas versiones aunque no son muy populares entre la comunidad sin grandes presupuestos, pero no hay que tomarlas a la ligera, pues han sido alimentadas por cientos de millones de dólares en investigación y desarrollo.

- AIX de IBM.
- IRIX de Silicon Graphics Inc.
- HP-UX de Hewlett packard.
- Solaris de Sun Microsystems.

5.2 BSD UNIX

Sus descendientes directos, aunque son pocos, no son menos importantes. BSD ha estado casi desde el principio y no existe una sola distribución que no haya sido afectada por los desarrollos aportados por la comunidad BSD.

La cultura que engloba BSD no puede ser descrita de forma sencilla, pero parece ser una comunidad que no solo se ha preocupado por los desarrollos tecnológicos (que por cierto son excelentes), sino que también ha defendido el estado abierto de Unix.

- Free BSD.
- Net BSD.
- Open BSD.
- Darwin.

5.3 LINUX

Aunque se caracteriza por ser el movimiento más reciente, no ha dejado de asombrar al mundo por su eficacia a la hora de desarrollar software. Aunque tampoco son los pioneros de los sistemas abiertos, ninguna otra comunidad ha impulsado al software libre como lo ha hecho la de Linux.

Su éxito no solo se basa en que han logrado desarrollar un sistema operativo completo y confiable, sino que introdujo una forma de hacer negocios que algunos corporativos consideran una amenaza y otros han sabido adoptarlo y convivir con Linux.

- Debian.
- Red Hat.
- Slackware.
- SUSE.

6 ADMINISTRACIÓN DE LOS SISTEMAS UNIX

Los sistemas Unix-like son los más flexibles que existen. Pueden funcionar en casi cualquier dispositivo de cómputo, desde un sistema mínimo construido en casa hasta las supercomputadoras más potentes que involucran miles de CPUs.

Por lo tanto no se puede decir que la administración de un sistema Unix sea asunto sencillo o complicado, realmente depende de los recursos de cómputo con los que se cuenten. Pero lo que si podemos hacer es tomar hablar de una de las aplicaciones más comunes que se les da a los sistemas Unix: Servidores de aplicaciones. Comparativamente Unix es más difícil de manejar que otros sistemas operativos, ya que hay una gran libertad para operarlos, lo que implica un mayor margen de error.

Actualmente son pocas las universidades que usa Unix como una plataforma de aprendizaje para los futuros profesionales de la computación, por lo que resulta un tanto difícil hacerse de un buen administrador de sistemas Unix.

Dependiendo del tamaño y la complejidad del sistema en cuestión se puede hacer una clasificación de los administradores de sistemas:

-
- Juniors (1-3 años):
 - Administran un sitio pequeño solos.
 - Pueden asistir a administradores de sistemas más grandes.
 - Intermedios/avanzados (3-5 años):
 - Administran solos un sitio mediano.
 - Pueden asistir a administradores de sistemas más grandes.
 - Ayudan a la planeación del futuro del sitio.
 - Evalúan y recomiendan compras.
 - Intermedios/avanzados (5+ años):
 - Diseñan e implementan LAN's y WAN's complejas.
 - Administran sitios o redes grandes.
 - Establece y recomienda políticas de uso del sistema y de los servicios de que éste ofrece.
 - Supervisa a todos los demás administradores y programadores de sistemas.
 - Tiene autoridad y la responsabilidad de comprar nuevos equipos necesarios para el buen funcionamiento del sistema.
-

Tabla 2⁵

7 SISTEMAS UNIX EN ARQUITECTURAS DISTRIBUIDAS

Unix se caracteriza por dominar el mercado corporativo de los sistemas computacionales. Muchas veces los requerimientos en cuanto a poder de cómputo de las empresas grandes o de equipos de investigadores sobrepasan cualquier cosa que se pueda

⁵ Bozidar Levi (2002). UNIX Administration. pp. 13-14.

ofrecer con una computadora personal. Por lo que en estos casos es necesario construir sistemas distribuidos en los que se combinen numerosos recursos de cómputo similares (para dar escalar la aplicación) o disponer de recursos especializados que contribuyan a la realización de la tarea.

Los sistemas distribuidos se caracterizan por tener las siguientes características:

-
- Datos compartidos: Permiten que varios usuarios tengan acceso a una base de datos común.
 - Dispositivos compartidos: Permiten que varios usuarios compartan periféricos caros, como las impresoras a color.
 - Comunicación: Facilita la comunicación de persona a persona; por ejemplo, mediante el correo electrónico.
 - Flexibilidad: Difunde la carga de trabajo entre las máquinas disponibles en la forma más eficaz en cuanto a los costos.
-

Tabla 3⁶

8 TENDENCIAS DE LOS SISTEMAS UNIX

Unix se ha caracterizado por atender a un mercado muy específico: los grandes corporativos, empresas con servicios de aplicaciones, usuarios profesionales en el área de la computación. Sin embargo esto está cambiando. Existen varios proyectos específicamente centrados para hacer la interacción con el usuario inexperto mucho más fácil y llamativa, esto con la finalidad última de introducirse en el mercado de las PCs que ha sido dominado por Microsoft desde los principios de este nicho.

Los sistemas Unix son muy estables y flexibles, pero la única desventaja es que hay que ser un poco más que un usuario inexperto para poder aprovecharlos al máximo. Actualmente Linux cuenta con el apoyo de la comunidad del software libre, está siendo adoptado por muchas universidades del mundo y algo que no se puede dejar de lado es que tiene un creciente apoyo por parte de la industria.

9 CONCLUSIONES

Nunca pensé que un tema tan técnico como lo es el de los sistemas operativos tuviera tal carga cultural. La historia de Unix es fascinante. No solo ha contribuido al desarrollo tecnológico del mundo, sino que también ha revolucionado en más de un sentido la forma en que se hacen las cosas.

Creo que Unix es una herramienta muy útil, ningún sistema operativo ha sido más diversamente utilizado. Provee una base tecnológica de gran calidad sobre la cual la imposibilidad es invisible. No existe otro sistema operativo que haya sido más extensamente utilizado. ¿A caso se conocen muchas supercomputadoras corriendo Windows?

⁶ Andrew S. Tanenbaum (1996). Sistemas Operativos Distribuidos. pp. 7.

REFERENCIAS

- Eric Steven Raymond (2003). The Art of Unix Programming. [Fecha de consulta: 3 de marzo del 2006]. Disponible en <http://www.catb.org/~esr/writings/taoup/html/>.
- Bozidar Levi (2002). UNIX Administration: A Comprehensive Sourcebook for Effective Systems and Network Management. Boca Ratón, FL: CRC Press.
- Andrew S. Tanenbaum (1996). Sistemas Operativos Distribuidos. México: Prentice Hall Hispanoamericana.
- Dennis M. Ritchie (1984). The Evolution of the Unix Time-sharing System. AT&T Bell Laboratories Technical Journal 63 No. 6 Part 2, October 1984, pp. 1577-93.
- Dennis M. Ritchie & Ken Thompson (1974). The Unix Time-Sharing System. CACM 17 No. 7.
- Wikipedia contributors (2006). Unix philosophy. Wikipedia, The Free Encyclopedia. [Fecha de consulta: 3 de marzo de 2006]. Disponible en http://en.wikipedia.org/w/index.php?title=Unix_philosophy&oldid=41346724.