

*Aquellos que no entienden Unix están  
condenados a reinventarlo, probablemente.  
-Henry Spencer.*

---

# POSIX: recluso diplomático.

**Roberto Hoyos M.**  
Mayo de 2006

Dos procesos existen en la guerra: el combate y la diplomacia. La doctrina indica que la diplomacia debe preceder al conflicto armado y que, por lo menos, pretende aminorarlo cuando se agotaron las pláticas. Pero, para Unix, como en el mundo moderno, el orden se invirtió, y la diplomacia juega después de que el daño está hecho. La metáfora viene a colación: Unix puede portarse muy fácilmente de una máquina a otra; de una arquitectura a otra; de un sistema a otro; puede también fácilmente ser extendido y reconfigurado. El árbol genealógico de versiones que se diversificaron a partir del *modelo AT&T* y el de *Berkeley* en la década de los '80 generaron una desgastante batalla que demeritó la aceptación de Unix en el mercado y provocó la llegada de Windows NT, que aprovechó el vacío provocado. POSIX es la diplomacia que busca restañar las heridas, agrupar todas las versiones para volver a hacer un frente multifrontal y colocar a Unix a la ventaja. Al menos, eso pretende. Lo cierto es que está lejos de la perfección y no ha logrado unificar al heterogéneo grupo de distribuidores de Unix.

La historia de POSIX nace de una no tan luminosa parte de la historia de Unix. Para 1981 existían dos variantes<sup>1</sup> principales de UNIX, BSD, de Berkeley, y Sistem III, de AT&T. Como sucede en un grupo de clanes, unos y otros buscan tener el control total, excluyendo a los demás. El clan Berkely había logrado integrar para 1980 el protocolo TCP/IP mientras que el clan AT&T tenía una versión que no era compatible con esas mejoras. No era un conflicto inusual si consideramos que hasta ese momento cada software estaba asociado a

---

<sup>1</sup> *The Art Of Unix Programming*. En el capítulo 2 Eric Steven Raymond profundiza en la naturaleza de las discrepancias entre las dos versiones, desde un punto de vista técnico y también viéndolo desde la óptica empresarial.

una máquina en particular y las compañías, sobre todo IBM, sabían que ello significaba dependencia de los usuarios. La guerra de los unix (*the unix wars*) habría de prolongarse durante toda la década, seriamente deteniendo el desarrollo conjunto del sistema. Como siempre, otro clan, en este caso particular, un influyente grupo de usuarios, *UniForum*, trató de conciliar las diferencias entre los dos sistemas. Ese fue el primer intento serio por generar estándares para el desarrollo de Unix.

¿Por qué es importante la uniformidad? ¿No acaso la evolución fomenta la diferenciación? Especies que tienen un ancestro común necesariamente divergen y eso propicia su supervivencia. El caso es que también repercute negativamente en la cooperación y comunicación entre ellos. El resultado fue que *Windows NT* ocupó el nicho de mercado que tanto peleaban y, en efecto, la portabilidad no sirve de mucho cuando no hay una plataforma común de comunicación. Dicho más coloquialmente como la tragedia de la torre de Babel, los hombres no pudieron construir una torre que llegara al cielo porque hablaban idiomas diferentes y nadie se entendía.

¿Por qué es importante, entonces, promover estándares? Concedemos, es importante la cooperación para llevar a buen puerto cualquier proyecto. Los estándares son ese pegamento que aglutine a los diferentes clanes. Las ventajas son tangibles: se incrementa la compatibilidad entre las (numerosas) distribuciones de Unix para que corran en cualquier sistema compatible y hacer el esfuerzo para reclutar a los creadores de software para portar y escribir productos para Unix/Linux. En realidad esa es la misión del LSB, o *Linux Standard Base*. ¿Dónde entra Posix, entonces? Esa es la parte negra de la historia, que ni siquiera hay consenso para crear el consenso. Pero trataremos esto después.

Resulta que el candidato ideal sobre el cual crear estándares es Unix. Es ampliamente utilizado, es robusto y es un sistema operativo del cual podemos decir que no existe un productor único, relativamente hablando. La IEEE venía rumiando estandarizar los comandos del API de C, propuesta que también era del interés de *The Open Group*. El resultado de tales esfuerzos se materializa en POSIX, que más o menos quiere decir *Portable Operating System Interface* (la X por la herencia de Unix). El problema es que la

IEEE fue muy restrictiva con quién y cómo accede la información. Hasta hace poco tiempo la obtención de esos documentos costaba y mucho. Todavía hoy publicar un documento bajo ese estándar cuesta dinero. ¿Cómo es que un estándar para un sistema operativo que es 'libre' y que es de interés general no pueda ser hecho público sin costo alguno? Las razones, por aberrantes que parezcan, dieron lugar a que en su lugar se crearan otros estándares, muy similares al propuesto por la IEEE, por cierto, que fueran gratuitos, tanto como para leer como para sugerir modificar. Tal estándar se conoce como *Single Unix Specification*. Hoy en día se pueden obtener ambas versiones simultáneamente, y aunque disponibles libremente, la IEEE exige el registro del usuario para poder ver la información, alegando que desea mantener información de quién accede a la información, cosa que sigue siendo molesta considerando que la información debe estar disponible para cualquiera en cualquier momento. El problema es que la disponibilidad de esta información no sucedió sino hasta recientemente.

Por ejemplo, se dice que Linus Torvalds diseñó Linux de tal forma que fuera apegado a POSIX lo más posible, pero que gran parte de su tarea fue adivinar, porque compró las especificaciones tiempo después de haber empezado, y que se basó en las *man pages* de otros sistemas.

Como vemos, POSIX se convierte en una especie de diplomático que asiste a banquetes donde existe mucho dinero, pero que prácticamente está aislado. Pero esa no es la única ni la mayor queja contra POSIX. Para empezar, POSIX es una certificación, y los sistemas que deseen tener la etiqueta *POSIX compliant* deben de certificarse, proceso que es costoso y largo y que, de aplicarse a la velocidad en que el software cambia, debería de realizarse muchísimas veces, cosa impráctica. Además, una vez que se llega a la certificación, en realidad no significa mucho: pues los estándares son extensos y ningún programa por complejo que sea requerirá tan alto nivel de requerimientos, tan sólo cumplirá, por necesidad, con uno o dos, pero no todo el estándar Posix es pertinente. No sólo eso, sino que, hablando más técnicamente POSIX tampoco habla mucho sobre sockets, que son una parte importante en los sistemas Unix que trabajan en red, puesto que cada aplicación debe saber qué tipo de socket y eso es difícil definir en una especificación de estándar. Como se

ve, POSIX representa un intento serio para crear estándares, pero su efectividad no es la que se esperaba, pues, para empezar, parece bizarro que su costo limitara su distribución cuando era imprescindible que fuera fácilmente adquirible. Parece una forma “cerrada” de hacer estándares para un sistema “abierto”. El diplomático ausente y aislado. La segunda crítica es que en realidad pocos sistemas seriamente se certifican en POSIX o buscan ser 100% congruentes con el estándar. En realidad es impráctico, como se vio. Se cuenta la historia de un sujeto cuya primordial preocupación era hacer sus sistema congruente con los estándares, y al final, quebró, porque a sus usuarios no les importaba; no debió haber destinado tantos recursos a un objetivo esquivo. Otra desventaja que bien nota Claude Kaiser del Centro de Estudios para la Investigación en Informática<sup>2</sup>, del CNAM en París, dice que los estándares ofrecen soporte a llamadas de bajo nivel para programas concurrente, cuando hay aplicaciones, muchas, en el mercado, que son de alto nivel para el mismo efecto.

Hay que reconocer, la divergencia entre las versiones de Unix era un proceso natural, casi predecible. Igualmente válido era el intento por satisfacer la necesidad para crear compatibilidad, lo que en palabras llanas es hablar el mismo idioma. Pero lo realmente triste es que *los clanes* seguían yendo en direcciones opuestas, provocando que Unix, en general, no fuera a ninguno en particular.

Es cierto, Unix ha sobrevivido. Hoy por hoy las aplicaciones que cumplen con estándares son más fáciles de portar y modificar para seguir siendo compatibles. La industria del software siempre se ha orientado por paradigmas y consensos para ser productivos, pero hubo un daño irreparable, que fue la pérdida de nichos de mercado, la emergencia de sistemas competidores que lejos de representar la mejor opción representaron otra, menos ideologizada y más pronta para jugar *el juego de mercado*. Claro, ha permitido que el daño no se haya extendido más allá de la redención. Quizá el punto más notable no es la creación de POSIX, sino su *liberación*. Que *The Open Group* y la IEEE hay permitido el libre acceso a la información (aunque todavía requiriendo registro de por medio) es un paso

---

<sup>2</sup> Cédric, Centre d'Étude et de Recherche en Informatique du CNAM. Establece que JAVA y ADA son aplicaciones de alto nivel que permiten programación concurrente, pero el soporte de POSIX se limita a muy bajo nivel, por lo que no ofrece el nivel de ayuda necesario para estas aplicaciones.

adelante en la creación de software compatible y portable, bajo la filosofía y para la familia de Unix. Pero todo esfuerzo encaminado en la continuidad de esta cultura de creación comunitaria –comunidad mundial– debe buscar el consenso de todos. Ulrich Dreppin ha culpado a *Linux Standard Base* de no permitir la incorporación de las ideas de otros productores, como *Debian*. Como vemos, Unix está todavía dividido en pequeñas batallas internas, secuelas de otro mayor conflicto.

La industria de software misma está seriamente amenazada por las patentes de software, que son tantas y tan diversas que, por ejemplo, es muy difícil salir con nuevos algoritmos de, digamos, compresión de datos que no infrinjan alguna patente.

El beneficio, pertinente. Pero los resultados, podrían ser mejores. El diplomático aislado debe recorrer mucho camino antes de poder decir que aglutina a todas las facciones, a todos los *clanes* que son responsables de Unix. Como en las democracias, el trabajo puede no ser rápido, pero está respaldado por una mayoría dispuesta a hacer que funcione mejor cada día. Aún hay tiempo, dice Steven Raymond, para rectificar. Después de todo, la diplomacia es el último recurso que se debe agotar.

## Bibliografía Selecta:

- \* Eric Steven Raymond, *The Art of Unix Programming*, 2003, [online version]  
<http://www.catb.org/~esr/writings/taoup/html/>
- \* Unix Estándar 1003.1 2004 Edition (POSIX), The Open Group e IEEE, [online version]  
[http://www.unix.org/version3/ieee\\_std.html](http://www.unix.org/version3/ieee_std.html)
- \* *La Industria en Peligro*, Liga para la Libertad en la Programación,  
<http://lpf.ai.mit.edu/Patents/industry-at-risk.html>
- \* *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/>
- \* Claude Kaiser, Jean-François Pradat-Peyre, Chameneos, a Concurrency Game for Java, Ada and Others, CEDRIC - CNAM Paris  
<http://cedric.cnam.fr/PUBLIS/RC474.pdf>
- \* The IEEE Standards Association,  
<http://standards.ieee.org/regauth/posix/>
- \* Ulrich Drepper, *POSIX Option Groups*, RedHat Linux,  
<http://people.redhat.com/~drepper/posix-option-groups.html>