

# NOTICIAS DEL KERNEL

## ESTADO DE GPL3

Linus Torvalds ha manifestado que no tiene intención de colocar ninguna de sus contribuciones al kernel bajo ninguna licencia que no sea la versión 2 de la GPL. Sin embargo, eso no impide que otros contribuyan código bajo la licencia que les venga en gana, con tal de que también se someta a la licencia GPL2 o (tal y como ocurre con código BSD) pueda relicenciarse bajo la GPL versión 2.

Incluso si Linus estuviera dispuesto, sería difícil cambiar la licencia del kernel. Todo aquel que contribuye código al kernel conserva su copyright y simplemente cede una licencia para el uso del código en el kernel.

A menos que la misma licencia defina las condiciones bajo las cuales se pueda modificar, nadie, menos el poseedor del copyright, tiene permiso para cambiarla. Por tanto, cambiar la licencia del kernel de V2 a V3 requeriría en teoría ponerse en contacto con los miles de desarrolladores que han contribuido elementos al kernel y pedirles que relicencien su código bajo la GPL3. Y luego todos esos miles de personas tendrían que consentir a la petición unánimemente.

Una alternativa sería insistir en que todas las nuevas contribuciones se enviaran con una licencia dual, con la licencia 2 y la 3. Al cabo del tiempo (o al menos en el mundo de los sueños imposibles), todo el código cubierto por la licencia 2 acabaría

reemplazado por código cubierto por la licencia 3.

El argumento de que todo código bajo licencia GPL se licencia habitualmente bajo la versión 2 “o cualquier versión posterior a discreción de su creador” no se aplica, porque el kernel no utiliza esas palabras cuando identifica su licencia. Se especifica de manera explícita la versión 2 de la licencia porque Linus Torvalds no confiaba en que Richard Stallman produjera una versión aceptable de la versión 3. Y, aunque la GPL especifica que el poseedor del copyright no puede imponer restricciones adicionales, el evitar la oración “cualquier versión posterior” no introduce ninguna nueva restricción, al no ser este texto parte de la licencia. Y al no ser este texto parte de la licencia, es una mera sugerencia cuando se licencia algo bajo la GPL.

En conclusión, tendrían que darse una increíble serie de circunstancias para que el kernel cambiara de licencia o, incluso, que se actualizara a una nueva versión de la licencia actual.

## DRIVER FORCEDETH YA NO ES EXPERIMENTAL

Adrian Bunk ha emitido un parche para que no se liste el driver Forcedeth como “experimental” durante la configuración del kernel. El driver parece ser ya bastante estable, tal y como confirman varios usuarios después de la aplicación del parche.

Otra parte del parche hace que ya no se muestre el mensaje descriptivo de una sola línea que menciona que el código se ha escrito a partir de ingeniería inversa aplicada a su homólogo binario de NVIDIA. Adrian opinaba que el usuario sólo necesitaba saber lo que hacía el driver, no como se desarrolló.

Sin embargo, Lee Revell señaló que esa información seguía siendo útil, aunque sólo fuera para saber a qué proveedor, supuestamente amistoso con Linux, convenía boicotear. Arthur Othieno aportó su granito de arena indicando que el texto sobre la ingeniería inversa se podía conservar en el texto de ayuda del driver, en vez de mostrarlo en la descripción de la configuración, y Alistair John Stracham mencionó que no toda NVIDIA era hostil al código abierto, ya que había al menos dos desarrolladores de la empresa que contribuyeron en la escritura del driver libre.

## ESTADO DE DESARROLLO DEL 2.4

El kernel 2.4 de Linux ha estado por ahí desde hace más de 5 años y el desarrollo lleva tiempo ralentizándose. Pero mucha gente sigue utilizando la versión 2.4 del kernel, sobre todo a la vista del abandono del ciclo de desarrollo par/impar utilizado anteriormente. Para muchos usuarios, el árbol 2.4 es el único en el que se puede confiar al ser el único que se ha sometido a un extenso esfuerzo de estabilización.

El argumento en contra es que las distribuciones hacen su propio trabajo de estabilización y que la serie w.x.y.z también se ocupa de la estabilización, pero todos estos esfuerzos se concentran en el uptime. El código base del árbol 2.6 no es estable en sí mismo, sino que cambia constantemente y esto es lo que pone nervioso a mucha gente y haciéndoles dudar antes de migrar al árbol 2.6.

Recientemente, Willy Tarreau envió un mensaje donde comentaba una docena de parches para el 2.4, principalmente para soporte de drivers, mejoras en la seguridad y optimización de memoria. Este mensaje

La lista de correo del kernel de Linux comprende lo principal de las actividades de desarrollo de Linux. El volumen del tráfico es inmenso, alcanzándose a menudo los diez mil mensajes semanales. Mantenerse al día de todo lo que sucede en el desarrollo del kernel es casi imposible para una sola persona.

Sin embargo Zack Brown es uno de los pocos valientes que lo intentan y a partir de ahora, podrá leerse lo último de las discusiones y decisiones con respecto del kernel de Linux llevados de la mano de este experto.

Zack ha publicado un resumen online semanal llamado “The Kernel Traffic Newsletter” durante cinco años. Linux Magazine te trae ahora la quintaesencia de las actividades del kernel de Linux del mayor especialista en el tema.



# Bienvenidos a



## LinuxLand

The Linux Business World

### Central de Soluciones Corporativas Linux



#### Red Hat Enterprise Linux

La solución más completa del líder en software corporativo Linux. Las dos soluciones disponibles, AS y ES, aportan el soporte más fiable para la explotación de sistemas y aplicaciones de misión crítica.



#### VMware GSX Server 3

VMware corre sistemas operativos y aplicaciones corporativas Linux y Windows en un mismo servidor físico. Instale sistemas en minutos, acceda a los dos entornos sin reiniciar, desarrolle y pruebe software en cualquier plataforma.



#### Novell Suite para PYMES

Completa colección de productos probados y diseñados para las necesidades y presupuestos de las PYMES. La solución completa servidor-escritorio para la pequeña y mediana empresa.

[www.linuxland.es](http://www.linuxland.es)

suscitó un interés inmediato y desencadenó peticiones de más soporte a diferentes características, así como indicadores a más parches que podrían ser recogidos y puestos a disposición de los usuarios en un repositorio centralizado.

Es difícil imaginar que estos parches sean incluidos en el árbol oficial en una fecha tan tardía, pero es posible que esta rama de desarrollo no oficial del kernel 2.4 se consolide como un proyecto más serio y obtenga abundantes seguidores mientras que el 2.6 continúe navegando a la deriva.

## REQUISITOS GPL PARA LA EXPORTACIÓN DE SÍMBOLOS

Desde hace ya tiempo, el kernel restringe el código de terceros que pretenden enlazar con él y utilizar sus símbolos (variables y funciones). Estos símbolos son necesarios para que los drivers de terceros interactúen con el kernel profundo. Pero muchas partes del kernel sólo exportan sus símbolos a código que ha establecido una variable de licencia para indicar que se licencia bajo la GPL.

Esto, de hecho, es un artilugio bastante ingenioso, a la vez que ha inspirado algunos interesantes intentos de sortearlo, tales como añadir extraños caracteres a la variable de licencia que satisficiera al kernel, sin indicar que nada del código fuera GPL. Todas estas áreas grises se resolvían a medida que aparecían, pero uno no deja de sorprenderse ante la audacia de las empresas que liberan drivers sólo en formato binario que consiguen manipular la variable de licencia justo de la manera correcta.

En estos últimos años estos intentos han ido decreciendo y distanciándose en el tiempo, pero la pregunta que ahora se plantea es ¿qué pasa con el software propietario producido por compañías amistosas con Linux? Hace poco, Sven Schmidt, de AVM, se quejó de que Greg Kroah-Hartman había alterado los requisitos de la variable de licencia del subsistema USB haciendo que fuera imposible ejecutar drivers USB sin licencia GPL en un kernel no parcheado. Sven recordó a todo el mundo que AVM había sido muy generoso dando soporte Linux y ofreciendo a sus clientes drivers tanto para Windows como para Linux, pero que ahora tendrían que abandonar el soporte para todos los drivers de sus dispositivos USB.

Hubo varias respuestas a esto. La tradicional, en boca de Valdis Kletnieks, es que AVM simplemente debería liberar sus drivers bajo la GPL y se acabaría el problema. Pero Greg proporcionó una respuesta más técnica: dijo que de hecho, con la llegada de `usbfs` y `libusb`, ya era posible escribir drivers USB en el espacio de usuario sin por ello sacrificar velocidad. AVM podría distribuir todos sus drivers de código cerrado para espacio de usuario sin preocuparse por la accesibilidad a los símbolos del kernel.

Esto podría haber calmado los ánimos, pero, como suele ocurrir cuando se habla del kernel, había mucho más en juego. Por ejemplo (y no fueron los ingenieros de AVM quienes lo mencionaron), se recordó que en la FAQ del kernel se dice desde hace algunos años que no se harían más símbolos sólo para software GPL y que los desarrolladores de software propietario deberían sentirse cómodos al contribuir sus drivers. La respuesta de Greg era que esa sección de la FAQ databa de hace ocho años y que debería actualizarse para que reflejara la realidad.

Sven también señaló que, incluso si los drivers en el espacio de usuario fueran posibles, no asegurarían la fiabilidad de aplicaciones con parámetros temporales críticos de la misma manera que lo harían en el kernel. Asimismo dijo que el tránsito al espacio de usuario implicaría un tremendo esfuerzo de desarrollo. Finalmente, como suele ser habitual, declaró que AVM también quería proteger su “propiedad intelectual”. Lo de siempre, vamos.

## GIT SE ACTUALIZA

El sistema de control de revisiones, git, continúa su desarrollo a velocidad de vértigo. Para empezar, Linus Torvalds y otros han estado pensando en dar soporte a subproyectos, permitiendo que una parte del repositorio git se pueda separar de manera independiente del resto. Linus, de hecho, lo ve desde un punto de vista inverso. Según su razonamiento, el valor del soporte a subproyectos reside en el hecho de que para proyectos que contienen muchos subproyectos, la nueva característica permite sacar todo el repositorio sin tener que teclear muchos comandos diferentes para obtener cada subproyecto. Por tanto, git probablemente acabe soportando subproyectos de una manera u otra.

El proyecto X.org está empezando a migrar al menos parte de su código base a

git, comenzando por el núcleo del X server. Keith Packard lidera este esfuerzo y planea migrar todos los módulos de X.org de los que es personalmente responsable. Otros módulos podrán elegir su propio control de revisiones y el código muerto probablemente nunca se migre.

Se está planteando si se implementa la funcionalidad que permita el bloqueo de ficheros para su edición. Esto implicaría emitir un comando “edit” sobre el fichero que se edita para que git supiera exactamente qué ficheros han sido alterados. BitKeeper se comportaba de una manera similar y la ganancia de velocidad era considerable. Linus no tiene claro si desea ir en esa dirección, más que nada porque git bajo Linux ya es increíblemente rápido, gracias en parte a la veloz función de evaluación de estado `lstat()` residente en el kernel de Linux. Desafortunadamente, bajo sistemas operativos con rendimientos de sistemas de ficheros menos eficientes, el beneficio de esta velocidad se pierde.

Git podría empezar a soportar la clonación poco profunda, lo que permitiría a usuarios clonar sólo el histórico más reciente de un repositorio dado. Junio C. Hamano, el mantenedor de git, ha propuesto esta característica ya presente en CVS como medio de protección para desarrolladores contra el volumen descomunal al cual puede crecer un proyecto. Incluso un proyecto pequeño, a lo largo del tiempo, puede crecer hasta adquirir un histórico tremendo, hasta un tamaño que de hecho sería difícil de contener en el sistema doméstico de un colaborador. La clonación poco profunda permitiría al desarrollador escoger la cantidad del histórico relevante a su contribución.

Aneesh Kumar ha estado trabajando en gitview, otro navegador de repositorios git, que se suma a la lista conformada por `qgit`, `gitk` y otros. Eric Wong, a su vez, ha escrito `git.svn`, una herramienta que permite a los desarrolladores que trabajan en un solo proyecto utilizar git o Subversion, según su elección. Lo hizo principalmente para poder utilizar git él mismo en proyectos que se almacenaban en repositorios Subversion. Junio ha creado un área *contrib* del árbol de fuentes para herramientas interesantes y/o experimentales. La más reciente de estas herramientas es una interfaz emacs para git, de Alexandre Julliard.