

# NOTICIAS DEL KERNEL

## Estado del Kernel 2.2

En Agosto de 2007, Xose Vázquez Pérez preguntó acerca del estado de la rama 2.2 del kernel y señaló que la versión 2.2.26 se había lanzado el 25 de Febrero de 2004. Por otro lado, la última versión candidata de la 2.2.27 era del 25 de Enero de 2005. Willy Tareau contestó que toda nueva versión de la rama 2.2 del kernel podría inducir a los usuarios a creer que es usable. Sin embargo, puntualizó que, de momento, muchas soluciones de agujeros de seguridad no se han incluido en la rama, y que simplemente está demasiado desactualizada como para continuar su mantenimiento.

Xose aceptó esa explicación en aquel momento, pero recientemente ha continuado sugiriendo que el kernel 2.2 debería ser eliminado de la página principal de kernel.org. Si está tan desactualizado que nadie debería usarlo ni parchearlo, claramente no debería publicarse en kernel.org, según su opinión. Esto parecía lo lógico. Sin embargo, en el momento de escribir este artículo, el kernel 2.2 aún se lista en kernel.org junto con el resto de ramas del kernel. ■

## Ingenioso Método para Programar Eliminación de Código

Matthew Wilcox ha tenido una brillante idea para ahorrar a Andrew Morton un pequeño dolor de cabeza. La lista actual de las funcionalidades del kernel que se encuentran programadas para ser eliminadas se guarda en un único archivo denominado *feature-removal-schedule.txt*. Como parte de la labor rutinaria de los hackers del kernel, éstos han ido añadiendo elementos al final de este archivo. El problema es que todo el mundo añade sus cambios al archivo en forma de parche, por lo que muchos de estos parches entran en conflicto con otros porque todos intentan añadir texto distinto en el mismo

sitio del archivo. Como resultado, Andrew ha estado aparentemente resolviendo eso a mano, lo cual le resulta bastante molesto.

La idea de Matthew para ayudar a Andrew es modificar las herramientas del parcheo para que hagan lo correcto de manera transparente. Por ejemplo, la herramienta diff produce un parche que contiene líneas de contexto acerca de los parches que produce, de manera que la herramienta de parcheo puede aplicar el parche en el sitio adecuado del archivo. La herramienta diff también lleva un seguimiento del estado "anterior y posterior" de la parte del archivo a modificar, pero con los cambios que Matthew propone, el estado "anterior" está vacío.

Según esto, su idea es poner un separador sencillo, como "- - - - -", entre las entradas y, lo que es más importante, al final del archivo. De este modo, la herramienta diff no sólo no tiene estado "anterior" para el parche, sino que sólo tendrá este separador genérico para proporcionar contexto a sus parches. Como señala Matthew, esto hará que la herramienta de parcheo inserte cada entrada nueva de manera aleatoria entre dos entradas adyacentes del archivo.

Esto supone un elegante truco, y además resulta de agradecer cuando puede ahorrar tiempo a alguien. Irónicamente, con la herramienta git no se producen los mismos errores que con diff y patch, pero como Andrew no usa aún git para esta parte de su trabajo con el kernel, esta sencilla solución puede funcionar. ■

## Nuevo Código General de Debugging

Thomas Gleixner ha propuesto una nueva y genial infraestructura para el depurado del kernel. Su idea es mantener una lista hashada de los objetos del kernel y llevar a cabo chequeos cada vez que se tocan o la memoria se

libera, de manera que las banderas rojas se identifican antes de que un bug pueda causar un kernel panic u otras consecuencias fatales. Estos chequeos no encontrarán todos los bugs, pero cuando marquen una bandera roja, casi seguro que será porque se ha detectado un bug verdadero en alguna parte. El plan de Thomas sería mantener el código de debug en el kernel, desde donde podría habilitarse fácilmente. El kernel no se ejecutaría con el código de debug habilitado por defecto, ya que ralentizaría todo el sistema.

El apoyo inicial al trabajo de Thomas fue bueno, aunque Greg Kroah-Hartman sugirió algunos chequeos adicionales. Andi Kleen también sugirió incorporar las funcionalidades de un viejo parche de Chris Mason, en las que un hilo en segundo plano aloja la memoria, la marca y la chequea periódicamente para ver si se ha corrompido. Debido a que la memoria sólo se usaría para las pruebas, cualquier código que se corrompiera no causaría necesariamente un problema inmediato para el sistema en ejecución, por lo que detectar la corrupción proporcionaría al usuario una valiosa información de debug que se podría guardar en logs antes de cualquier potencial problema.

Es muy probable que el trabajo de Thomas se acepte para incluirlo en el kernel en algún momento, y posiblemente continuará ampliándose con estas y otras sugerencias. ■

## Distribuir el Mantenimiento de IC2

Jean Delvare ha realizado un llamamiento para ver quién quiere ser su comantenedor del subsistema IC2. Ha tenido algunos problemas para mantener el ritmo con los incesantes envíos de parches y ha pensado que otro par de ojos podrían ayudar. Un par de semanas más tarde anunció que Ben Dooks ha aceptado el cargo, y ha

cy HD DVD  
 DVD capa por sólo 5,30€  
 AÑO XVII | OCTUBRE 2008 | Nº 324

# Java-Server

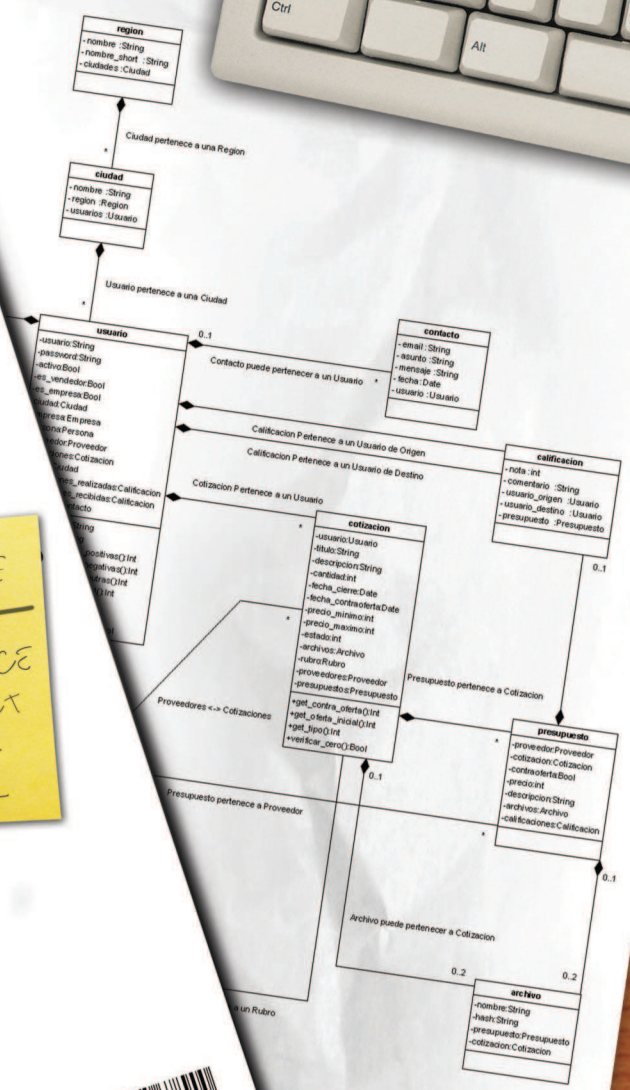
todos los secretos de arquitectura de tres con J2EE

**ENTREGAR A PRIMERA HORA!**

**IMPORTANTE**  
 CLASS E COMMERCE  
 PRODUCT: PRODUCT  
 PRECIO: NUMBER  
 STOCK: NUMBER

100 Herramientas  
 6 Antivirus  
 12 GB de RAM  
 324 GB de disco duro

Proveedor Apache para tus proyectos  
 DE VENTA SÓLO EN LA PENINSULA



Llevas días pringando con la web,  
 programando código sin parar hasta las tantas de la noche.  
 Hoy por fin has publicado el proyecto, pero...

Hay avalancha de visitas en el site  
 y tu servidor web ¡no responde!

# ¿Problemas?

enviado un parche al archivo MAIN-TAINERS con el nuevo listado. ■

### Hora de Cambiar a git para los Usuarios de Cogito

Cuando Linus Torvalds escribió git, su objetivo era un equivalente a una capa de "llamada al sistema" para control de revisiones. Su aplicación proporcionó las funcionalidades de muy bajo nivel para administrar los cambios en un directorio que cubriera sus necesidades una vez que ya no estaba disponible BitKeeper. De hecho, él veía a git como una mejora frente a BitKeeper, ya que eliminaba funcionalidades que consideraba como innecesarias y habilitó otras que BitKeeper había sido incapaz de proporcionar, como una semántica de etiquetado sensata.

Desde sus inicios, el programa git ha sido difícil de entender porque no proporciona la clase de funcionalidades completas que cualquiera podría esperar de un sistema de control de revisiones. En lugar de permitir a los usuarios teclear un comando para sincronizar su repositorio con la fuente principal,

por ejemplo, primero tienen que recabar con "fetch" los cambios del repositorio y luego deben fusionarlos con su repositorio local con otro comando. Otras acciones menos comunes son incluso más complejas de llevar a cabo. Linus hizo todo esto para mantener las operaciones flexibles y potentes. La intención no era que fuesen el front end de un repositorio. Él esperaba y animaba a otra gente a que hiciesen sus propios scripts con comandos más amigables por encima de su interfaz de "llamadas al sistema" git.

La primera y más popular de las interfaces con scripts de git fue la aplicación Cogito, la cual, durante un tiempo, parecía que podría convertirse en la herramienta principal de los usuarios a usar junto a git. Puede ser difícil hacer el seguimiento del estado de este tipo de proyectos, pero ahora resulta que Cogito ya no está mantenido, y el propio git proporcionará tanto la potente capa de back end como el amigable front end para los usuarios habituales.

El front end de git de hecho ha estado bajo desarrollo desde hace algún tiempo, denominándose capa de "porcelana". Proporciona un conjunto de comandos que son familiares para los usuarios de la mayoría de los sistemas de control de versión y que confían en los comandos de bajo nivel para su implementación. Mientras ayudaba a alguien que tenía problemas con un repositorio git y había descrito los comandos de Cogito con los que tenía problemas, Linus le comentó que "en primer lugar, realmente deberías despedirte de Cogito. Está garantizado que cualquier uso de Cogito será más difícil y menos probable que sea correcto que simplemente usar git nativo (y ya casi nadie será capaz de ayudarte, ya que no está mantenido desde hace un año)".

Con git que ya proporciona su propio front end y con Cogito que ya no se mantiene, parece que todo aquel que haya estado usando Cogito debería pasarse a git directamente. Y todo aquel que esté usando otros sistemas de versiones, también debería cambiar a git. Es realmente fantástico. ■

