

# NOTICIAS DESDE EL KERNEL

## Sus Labores

El kernel siempre está siendo sometido a tareas de limpieza para mantenerlo como los chorros del oro. Recientemente, la mayoría de las referencias que aún le quedaban a DriverFS fueron eliminadas por Eike Beer.

A DriverFS se le viene denominando SysFS desde hace ya bastante tiempo, pero varios documentos y archivos fuente seguían refiriéndose a él por su antiguo nombre, ya fuese porque arreglar las referencias al nombre no era muy importante o bien porque los mantenedores originales habían migrado a otros proyectos.

Adrian Bunk, por su parte, lleva ya tiempo trabajando en la completa eliminación de los ya obsoletos drivers de sonido OSS y, recientemente, ha programado el borrado de varios de ellos. Al decir obsoletos, se refiere a que existen sustitutos ALSA completos y funcionales listos para asumir sus funciones.

Desafortunadamente, no siempre está claro cómo de bien ha de funcionar un driver ALSA antes de considerarse un sustituto adecuado. Por ejemplo, algunos de estos drivers ALSA son mucho mayores que sus equivalentes OSS, lo que puede suponer un problema para la gente que trabaje en sistemas embebidos, aún si la versión ALSA soporta todo lo soportado por el anterior driver OSS. A pesar de las dificultades, Adrian desmantela pieza a

pieza OSS. No es un trabajo muy satisfactorio, pero como resultado aporta un kernel más limpio y feliz.

Russell King también trabaja para limpiar el kernel de código obsoleto, lo que incluye un buen número de funciones que se habían quedado anticuadas desde hace tiempo. Por desgracia, en algunos casos, la eliminación de este código rompe varios drivers como los de MWave y ibmasm.

La situación actual con funciones en desuso se ha vuelto complicada debido al hecho de que muchos de los drivers afectados no estaban siendo mantenidos, y por lo tanto no se actualizaban. El resultado final de todo esto ha sido que Max Asbock ha corregido ibmasm, mientras que AlanCox ha parcheado MWave.

Aparte de trabajar en OSS, Adrian también se ha hecho responsable del trivial patch Monkey de Rusty Russell. El Trivial Patch Monkey es un sistema semi-automático que recoge parches del kernel extremadamente simples y obvios para evitar que cualquier otro tenga que ocuparse de hacerles el seguimiento y reenviarlos si no se aplican la primera vez.

El motivo original por el que Rusty empezó el Trivial Patch Monkey era precisamente debido a que muchos de estos parches no se aplicaban la primera vez y los desarrolladores encontraban frustrante el tener que reenviar parches obviamente correctos una y otra vez.

El desarrollo inicial del patch monkey se llevó a cabo en una época en la que Linux Torvalds encontraba dificultoso mantenerse al día con el vasto número de parches que se le remitían. Una de las soluciones para aliviar las tensiones de los desarrolladores era la de utilizar control de versiones. La otra era la de administrar los parches más sencillos y obvios con el Trivial Patch Monkey, que Rusty mantuvo durante años y que ahora es responsabilidad de Adrian.

## Sistema de Ficheros eCryptFS

Phillp Hallewell y Michael Halcrow han sometido la versión 0.1 de eCryptFS a evaluación con vistas a su inclusión en el árbol oficial del kernel. Ésta es la versión simplificada de lo que esperan del sistema, y han elegido someterlo a examen en este formato para facilitar el análisis y depurado del diseño básico. Si aprueba el examen y se incluye en el kernel, planean empezar a incluir prestaciones adicionales a la infraestructura básica.

La idea fundamental de eCryptFS es la de hacer la encriptación de desencriptación completamente transparente para las aplicaciones del usuario. La encriptación se hace para cada fichero individual, con los metadatos criptográficos encapsulados dentro del mismo fichero. Esto permite a los usuarios tratar ficheros eCryptFS exactamente igual que otros tipos de ficheros no encriptados. Los ficheros pueden ser copiados de un lugar a otro, incluso a dominios no confiables, y seguirán siendo legibles exclusivamente por los usuarios con las credenciales criptográficas correctas. De esta manera, eCryptFS se comporta de manera similar a GNUPG u otras herramientas de encriptación con claves públicas.

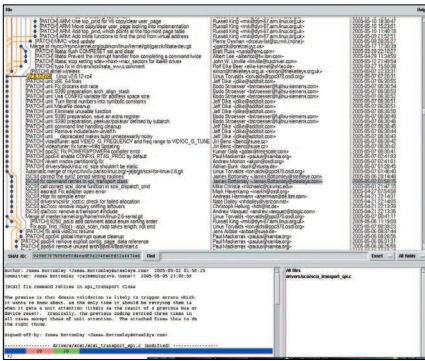
Debido a esta flexibilidad en el diseño básico, existe una gran anchura de manga para especificar las políticas de seguridad en un sistema dado. De momento, sin

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.





**Figura 1: Git es un juego de herramientas de bajo nivel muy poderosas.**

embargo, Phillip y Michael han aportado exclusivamente el soporte por montaje hasta que las características y diseño básicos puedan ser probadas adecuadamente. Parece claro que ya existe código adicional listo para ser incluido a medida que la oferta actual se gana su aprobación.

## El Estado de GIT

A los desarrolladores de git no hay quien los pare. El proyecto ha sobrepasado todas las expectativas, incluyendo las del mismo Torvalds, y para cuando leáis esto, probablemente se haya pasado a la versión 1.0. Y, a pesar de lo anterior, git sigue manteniéndose fiel a sus objetivos originales, en el sentido de que no aporta un interfaz sencillo tipo CVS que cualquiera pueda coger y ejecutar, sino que, tal y como Linux lo describió en un principio, se trata más bien de una serie de comandos de bajo nivel, sobre los cuales cualquiera puede escribir una serie de operaciones de alto nivel más amigables. A este respecto, Cogito sigue siendo el front-end más popular para git y provee de un sencillo interfaz a lo CVS que resalta la potencia subyacente.

Si bien Linux hace tiempo que abandonó el cargo de líder del proyecto de git, sigue siendo una importante influencia. Además de escribir código, tal vez su mayor contribución sea la de mantener encarrilado el proyecto. A pesar de ser un proyecto relativamente pequeño, la elegancia de su diseño base no siempre es obvia y, en ocasiones, algunos desarrolladores se han encontrado resolviendo problemas que en realidad no existen. Por ejemplo, el protocolo de git no manejaba lo que alguna gente necesitaba y se abrió una discusión sobre la posibilidad de reemplazarlo y de cómo afectaría esto en términos de compatibilidad retroactiva, hasta

que Linux apareció por allí y dijo: “¡Chicos, chicos! De hecho el protocolo está diseñado para ser extensible”.

Uno de los problemas persistentes con git es el seguimiento de renombrado de ficheros. En la práctica, a menos que se desee utilizar un trozo de la historia del renombrado para algo, git administra sin ningún tipo de fallo el renombramiento. En otras palabras, en el nivel más básico, si se renombra un fichero en repositorio git, éste es capaz de hacer el seguimiento. Sin embargo, el interfaz diseñado para presentar estos cambios ante el usuario aún está por escribir y no parece una tarea sencilla. El problema radica en la insistente idea de Linus de que la función de detección de renombramiento pertenece al front-end. En su opinión, confiar en el usuario para que le diga cuándo se ha producido un renombramiento daría lugar a muchos casos de error, en parte debido al hecho de que en general los usuarios ni siquiera se dan cuenta de cómo se trasladan datos entre ficheros. Sin embargo, según Linus, la pista está ahí, dispuesta a ser seguida cuando el usuario lo requiera.

El debate sobre el renombramiento es continuo, sin que nadie, ni siquiera el propio Linus, sepa a ciencia cierta qué es lo que se requiere para realizar un seguimiento de renombramientos preciso a la manera propuesta. A pesar de haber presentado descripciones detalladas de porciones de la solución, sigue bloqueado en algunos aspectos, y con éstos, todos los demás. Sin embargo, el debate posee las características de las cosas en las que Linus ha insistido en el pasado. Se obstina en que algo es correcto, a pesar de que contradice el saber popular y de que nadie, a excepción de él, ve cómo puede resolverse. A continuación se desencadenan peleas y flamewars. Al final todas las piezas acaban encajando y se convierten en el nuevo saber popular.

Mientras tanto, algunos, como Jeff Garzik, están involucrados de manera muy activa en desarrollo utilizando git. Hace poco Jeff migró el desarrollo de ethtool a un repositorio git y se ha estado planteando algunas modificaciones inusuales para el propio git. Su última idea loca es añadir soporte para un repositorio git de red puro. Hasta el momento, git se ha presentado como un sistema distribuido, en el sentido de que no hay repositorio central y que cada nodo actúa como el servidor para su propia versión de un proyecto dado. Jeff, por su lado, propone deshacerse

incluso de la noción de servidor de repositorio. En vez de que cada máquina sirva su propio repositorio, participarían muchos sistemas, sin que existiera una única máquina esencial para el mantenimiento de todo el conjunto. Los usuarios obtendrían la versión más reciente del proyecto consultando una red git, en lugar de a una única máquina en Internet. O una red git consultaría a otra y los repositorios de proyectos flotarían de un lado del mundo al otro sobre alas binarias.

Es una idea desquiciada, pero también lo es git. Y lo es el seguimiento de renombramientos en el front-end. Y también lo es la implementación del control de las revisiones como una serie de interfaces similares a llamadas al sistema. Todo es una locura... pero es lo mejor que tiene.

## Subsistema de Suspensión de Software

Rafael J. Wysocki ha partido el código de la suspensión de software en dos subsistemas independientes y, gracias a la aprobación de Pavel Machek, es probable que ésta sea la dirección que tome el árbol oficial. El objetivo es simplificar el código y hacerlo más coherente, para, con el tiempo, mover varias porciones al espacio de usuario.

El subsistema primario es administrador de instantáneas, que crea las estructuras de datos que deben preservarse durante la suspensión del sistema. El subsistema secundario, y el que con toda probabilidad acabe en el espacio de usuario, es el administrador del swap que recoge lo que le entrega el código de la instantánea y lo escribe en el swap antes de la suspensión. Al reactivarse el sistema, el subsistema lee los datos del swap y recrea el sistema en ejecución.

Este método le permite a Rafael eliminar tres cuartas partes de los datos de la instantánea que graba en el disco, lo que hace posible contar con más memoria durante la reactivación. Asimismo se elimina una restricción de tamaño que se imponía a los datos y desaparece la necesidad de algunas variables globales (lo que siempre es bueno).

Rafael indica que su idea sigue siendo un prueba de concepto y que quedan por probar algunos detalles como las rutas de errores. Sin embargo, el diseño en sí se ha encontrado con una aprobación generalizada y es de esperar que será la dirección elegida para swusp.