

NOTICIAS DEL KERNEL

Fusiones de Driver más Rápidas

Recientemente, Linus Torvalds ha reducido los obstáculos requeridos para incluir código en el kernel. Cuando apareció Linux, una de las prioridades principales de Linus era mejorar las contribuciones. Para lograrlo, puso énfasis en responder a cualquier parche que viniese desde la comunidad, aceptando muchos e incluso realizando estadísticas a mano de los que recibía. En los 90, a medida que aumentaban las contribuciones, su capacidad de respuesta disminuyó, siendo más frecuente ver cómo descartaba parches que no le gustaban. Linus también comenzó a insistir en que el diseño y los algoritmos debían ser bonitos y en que las diferentes partes del kernel se comunicasen de forma natural a éstos.

Cuando Linus adoptó las ramas del kernel “estable” y de “desarrollo” mantuvo su insistencia en el buen gusto, pero se hizo incluso más estricto durante el ciclo “estable”. Con la estructuración abierta de la contribución de código del kernel en una jerarquía de mantenedores y “tenientes”, Linus comenzó a crear una cultura de adhesiones a sus preferencias en el código, (algo que para otra gente podía generar barreras de entrada), además de hacer el trabajo técnico de

codificar y revisar el código en busca de bugs. Con el uso de BitKeeper y git, la cultura de Linus de “buen gusto al codificar” podía extender su distribución a proyectos particulares que trabajaban aisladamente, en los que contribuyentes individuales podían revisar el trabajo de otros sin necesidad de ser parte previamente de la rama principal del kernel.

Durante la rama 2.6, Linus abandonó los incómodos vaivenes entre las ramas estable y de desarrollo, reinstaurando un conjunto de micro-forks en la rama de desarrollo, en las que la rama principal 2.6 nunca abandonó la fase de desarrollo y en la que cada lanzamiento generaba un nuevo fork estable, sólo para arreglar los bugs. La decisión de abandonar los ciclos originales estable/desarrollo marcó una fase de reconsideración de un conjunto de problemas. Una de las principales justificaciones para el cambio fue que las distros de Linux siempre generaban sus propios parches específicos por encima de los lanzamientos oficiales del kernel. Esto hizo que al menos algunos de los esfuerzos en estabilidad fueran debatibles, ya que las distribuciones a menudo usaban parches novísimos que no habían podido probarse y revisarse suficientemente como un kernel estable oficial.

Debido a que la responsabilidad de proporcionar verdadera estabilidad siempre fallaba por parte de las distribuciones de Linux, Linus les pasó formalmente esta responsabilidad y la levantó de su equipo principal de desarrollo. Esta decisión indicaba un nuevo método en el desarrollo del kernel, en el que no se abandonaba la necesidad de estabilizar aspectos del kernel, pero en el que se ponía especial hincapié en permitir a los colaboradores que se despreocupasen un poco.

Ahora, Linus ha comenzado a aceptar parches de drivers que contienen problemas obvios. Recientemente, Adrian Bunk se quejó acerca del envío de un parche que se aceptó en el kernel a pesar de tener más de 250 errores “checkpatch” y más de 2000 warnings. En el pasado, estos problemas tenían que arreglarse antes de que el driver pudiese ser aceptado en la rama. Parte de la justificación de Linus para el cambio en la política estriba en el consenso que existe acerca de que el código tiene muchas más probabilidades de ser probado y arreglado en la rama oficial que fuera de ella.

De igual manera, se supone que el usuario medio confía en el kernel de su distribución, en lugar de en la versión oficial, mientras que los desarrolladores del kernel usan principalmente la versión oficial y pueden lidiar mejor con drivers menos pulidos.

Linus señala que esta nueva política apunta al código de los drivers, que por definición es periférico al cuerpo principal del código. Para las interioridades del kernel, presumiblemente existen estándares aún más altos para el código al que se permite ser incorporado. El código de drivers suele funcionar independientemente y no interfiere con otras partes del kernel.

Linus dice que es importante que los drivers se hayan probado correcta-

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.

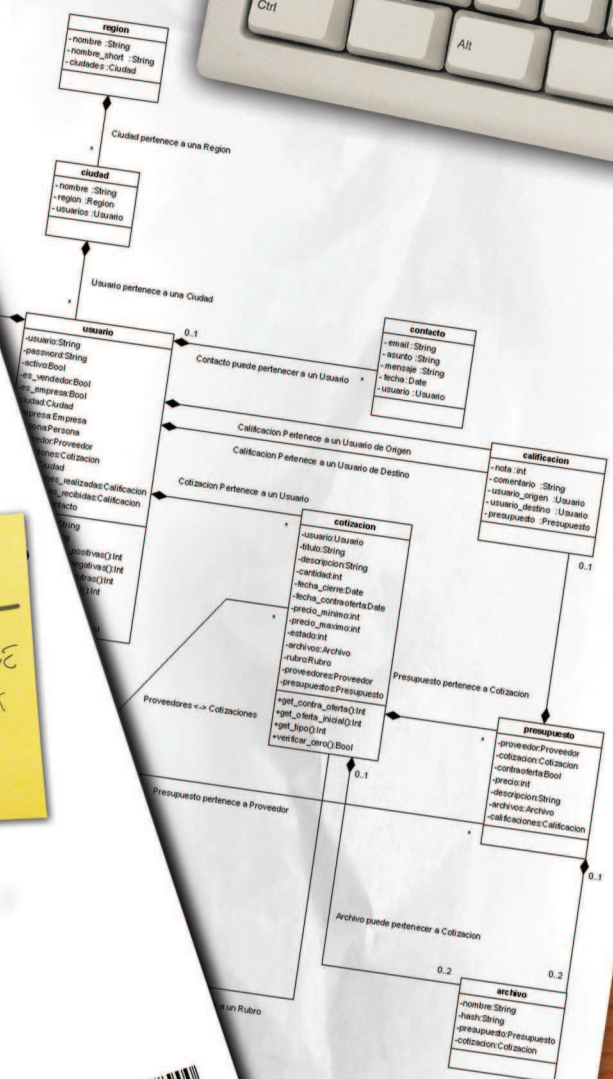


sc y HD DVD
 Año XVII | OCTUBRE 2008 | Nº 324
Java-Server
 Descubre todos los secretos de la arquitectura de tres capas con J2EE
 Sun Microsystems
 100 Herramientas para el desarrollo
 6 Antivirus para tu sistema
 12 GB de espacio en disco
 324 GB de capacidad de almacenamiento

ENTREGAR A PRIMERA HORA!

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

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?

mente antes de incluirlos en el kernel. No le preocupa qué aspecto tenga el código, pero quiere asegurarse de que funciona y de que no se cuelgue mucho o de, que suponga pérdida de datos a los usuarios.

Esta nueva dirección no le simplifica la vida a Adrian Bunk, que frecuentemente tiene que escudriñar a través de grandes cantidades de código del kernel para eliminar algún fragmento defectuoso. De hecho, identificar el código a eliminar del kernel puede convertirse en una tarea cada vez más complicada a medida de que se añade código menos limpio y se baje la barrera de entrada. Por otro lado, Greg Kroah-Hartman, Jeff Garzik y Arjan van de Ven opinan que la nueva dirección mejorará el kernel. Sin otras quejas mayores, incluso Adrian no parece muy preocupado, y la discusión se ha desplazado a cómo solucionar el script "checkpatch" de forma independiente a esta nueva política.

Indudablemente, va a entrar en el kernel mucho más código de drivers, y el "equipo estable" y las distribuciones continuarán su labor para que el usuario medio tenga una buena experiencia. Igualmente, en el futuro habrá otros cambios en la manera en que se desarrolla el kernel, ajustando los problemas que puedan surgir con esta nueva filosofía. ■

Supervisión de Fusiones del Subsistema

Ahora que es mucho más fácil incluir código en el kernel, Andrew Morton tuvo algunos recelos y quiso asegurarse de que no se rompía todo. Su idea original fue crear una rama git linux-next por donde los parches

podieran pasar en su camino hacia Linus, pero esa idea cambió rápidamente para enfocarse sólo en los subsistemas.

Un gran problema de los subsistemas era el número de conflictos en la fusión. Cada vez que una ventana de fusión se abría en la rama de Linus, todo el mundo tenía que volverse loco intentando resolver todos los conflictos del subsistema. La idea de Andrew incluía a un voluntario manteniendo una rama que funcionase con todo el código de subsistema, ayudando a identificar los problemas de fusión, problemas de compilación, y posiblemente incluso problemas en tiempo de ejecución si los usuarios pudiesen ser requeridos para probarlo. Sin embargo, para crear esta rama linux-next, las prácticas de fusión de los mantenedores del subsistema debían alterarse significativamente.

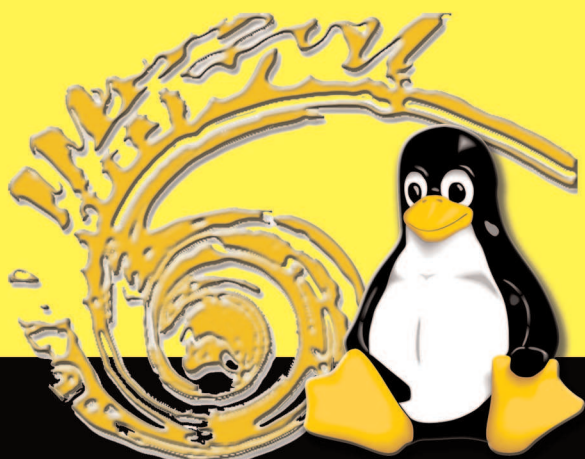
Stephen Rothwell anunció la creación de una rama linux-next para los subsistemas, así como la lista de correo linux-next@vger.kernel.org para las discusiones de la nueva rama. Stephen invitó a todos los mantenedores de subsistemas a que le enviaran las direcciones de sus ramas git o series quilt de manera que pudiese acudir a estas fuentes diariamente de forma automática. Toda rama con conflictos de fusión podría descartarse de la recompilación de ese día, y el mantenedor de ese subsistema podría ser informado automáticamente mediante email. Stephen también llevaría a cabo compilaciones automáticas para todas las arquitecturas en que fuese posible. Cualquier subsistema que no pudiese compilar, podría descartarse de la rama ese día.

Stephen no era el único voluntario: Frank Seidel, Ann Davis y Harvey Harrison también se ofrecieron como voluntarios para mantener la rama. Andrew seleccionó en última instancia a Stephen, el cual espera que los otros voluntarios puedan ayudar según las necesidades.

James Bottomley también ha estado manteniendo una rama similar a linux-next, que ha dejado el paso a la de Stephen. Andrew ya conocía el trabajo de James, pero había diferencias significativas que hacían imposible resolver los problemas que Andrew quería resolver. Por ejemplo, James no llevaba a cabo pruebas automáticas de compilación. De igual manera, su rama sólo se nutría de 46 de los cerca de 80 subsistemas que Andrew quería incluir en linux-next.

La idea de linux-next puede conducir a muchos más cambios en la manera en que se prueba y envía el código. Los mantenedores del subsistema necesitan tomar el control sobre cómo realizan los parches y qué partes del kernel tocan. Los problemas revelados por linux-next pueden acabar teniendo respuestas por defecto, como parches que se rechazan si no han estado el suficiente tiempo en la rama para ser probados. En última instancia, podría ocurrir que la decisión de Linus de bajar las restricciones en el código que entra en la rama principal terminara provocando restricciones más fuertes entre la gente que envía código si configuran cosas como en linux-next.

Claramente, el proceso de desarrollo del kernel continúa experimentando los grandes cambios que comenzaron con la versión 2.6. ■



Asociación Linux Español