Notícias do Kernel

Adeus ao driver Gamma DRM

O suporte à placa de vídeo GMX 2000 da 3dlabs foi removido do kernel 2.6. A placa é tão rara que os mantenedores do driver acham que virtualmente ninguém a usa. Ao mesmo tempo, deixar o driver no kernel faz com que o código global seja difícil de manter.

Dave Airlie, um dos desenvolvedores do projeto, indicou que o esforço envolvido na manutenção de todos os casos especiais começou a pesar muito mais do que o benefício de suportar hardware que provavelmente ninguém mais usa. A decisão de marcar o driver como "BROKEN" no kernel será seguida por uma faxina geral para remoção das "gambiarras" espalhadas pelo código fonte para que a GMX 2000 funcione. Depois disso, espera-se que o driver pare de funcionar; será, então, removido definitivamente.

O desejo de suportar itens obscuros de hardware possui raízes profundas na comunidade de desenvolvedores Linux e em todo o universo geek, mas essa "urgência" sempre foi quebrantada por opositores ativos como Linus Torvalds. Proclamando a beleza da praticalidade e seu imenso valor a despeito de um conceito abstrato do que é o "correto", Linus sempre teceu críticas aos puristas. Mas, com isso, criou uma certa cultura de desenvolvimento que se move extremamente rápido. Isso não significa que hardware que pouca gente usa não seja bem-vindo no kernel do Linux. Pelo contrário, há muitos casos de hardware antigo sendo ativamente suportado por hackers do kernel. Mas os esforços para suportar esse tipo de hardware devem ser pesados contra outros fatores, como o número existente de usuários daquele item.

Ocasionalmente acontece de um novo driver ser desenvolvido com um "esqueleto" mais complexo do que o necessário porque o autor já o deixou preparado para expansões futuras. Em quase todos esses casos, o autor é instado a reduzir a complexidade final até que ela seja realmente necessária. Isso sempre resulta em uma certa quantidade de trabalho perdido por parte do desenvolvedor do driver, mas também significa que o kernel não será sobrecarregado com um código muito pesado, que dificultará modificações no momento em que as falhas surgirem. Em vez disso, o projeto e desenvolvimento do driver corresponderá às necessidades do momento, com maior liberdade de decisão sobre a melhor solução do que ocorreria caso o desenvolvimento inicial fosse baseado num "chute" sobre as possíveis falhas e novas implementações.

Dave e o resto dos desenvolvedores do driver para a GMX 2000 seguem essa nova tradição. Ao mesmo tempo, ainda não ficou claro se esse driver deve ser, realmente, removido. À medida que os desenvolvedores vão avançando em direção ao objetivo de apagar o código completamente, os usuários reais da placa podem "sair da moita" inesperadamente para reclamar suporte. Nesse ponto, pode ser necessária uma reavaliação da situação do driver: se deve ser mesmo apagado, reescrito ou revertido ao estado anterior, funcional mas difícil de manter. ■

Novo mantenedor do I2O

Markus Lidel fez uma profunda reorganização no código do I2O no kernel 2.6 e, por isso, foi nomeado seu mantenedor oficial. Alan Cox foi o mantenedor antes de Markus mas, como em quase todos os projetos tocados por ele, tudo era muito provisório e apenas para manter o código funcionando enquanto um mantenedor definitivo não fosse encontrado. Os melhoramentos de Markus seguem o conceito tradicional de divisão das versões do kernel em "estável" e "em desenvolvimento". Como seu trabalho acabou parando na série estável, em teoria ele não deverá fazer alterações maciças que possam tornar o sistema menos estável.

Andrew Morton, entretanto, vem instituindo várias novas idéias na filosofia padrão de desenvolvimento do kernel. Uma das mais interessantes é que a série 2.6 não colocará a estabilidade acima de outros interesses. Em vez disso, Morton quer que novas funcionalidades e melhoramentos de código continuem a ser adicionados ao kernel 2.6 num futuro próximo.

Uma das razões para essa decisão, ou pelo menos um dos fatores para suavizar o risco, é a existência de distribuições como Debian e Red Hat. Essas distribuições sempre empacotam kernels modificados em vez dos "oficiais", e agora Andrew, com total apoio de Linux Torvalds, decidiu colocar grande parte do fardo da estabilidade do kernel oficialmente nas costas dessas distros. Uma maneira filosófica de encarar o problema é como uma extensão do modelo de desenvolvimento já existente.

Até agora as distribuições têm sido, por assim dizer, "cidadãos de segunda classe" no desenvolvimento do kernel, usando o kernel oficial como base para suas próprias versões modificadas. Com esse novo modelo, as distros aumentam em importância no processo de construção da versão oficial. O kernel "oficial" distribuído no site *kernel.org* é agora uma versão mais de desenvolvimento, menos destinada às massas do que antes. Em vez disso, e além do desenvolvimento adicional, as distros modificarão esse kernel para deixá-lo estável. Nesse novo modelo, os únicos kernels realmente estáveis serão os incluídos nos CDs de instalação das distribuições Linux!

O trabalho feito por Markus no I2O tem a mesma natureza da "plástica" no subsistema de memória virtual das primeiras versões da séria 2.4. Enquanto aquela decisão encontrou forte resistência, o trabalho de Markus é agora, em muitos aspectos, mera rotina. A remodelação profunda do código do kernel em séries estáveis já não é mais tabu. Se esse modelo de desenvolvimento persistirá na série 2.7 ou se a 2.8 seguirá os mesmos passos da 2.6, ninguém sabe.

Parece que estamos entrando num período de mudanças rápidas e, por vezes, fugazes no modelo de desenvolvimento do Linux. Sem sombra de dúvida, os métodos usados num futuro próximo serão drasticamente diferentes dos atuais. Será interessante acompanhar o processo e julgar seus resultados. ■

Desenvolvedores removendo seu código do kernel?

Uma controvérsia estrambótica tomou forma nos círculos de desenvolvedores do kernel a respeito da exclusão do driver PWC, que dá suporte a certas webcams com chipset Philips. Por muitos anos, o driver ficou dividido em uma parte de código aberto e outra de código fechado. A porção livre disponibilizava um "gancho" (*hook*) no kernel que permitia à sua colega binária e fechada comunicar-se diretamente com o kernel. A divisão em duas partes era considerada necessária pelo mantenedor para suportar câmeras que usavam os chips Philips, que por sua vez comunicavam-se unicamente com a parte binária do driver.

Entretanto, esse tipo de hook no kernel é expressamente proibido e Linus Torvalds já deu voz a vários protestos contra tais coisas, consideradas tentativas de contornar os requisitos legais básicos da licença GPL. De fato, a mera possibilidade – mesmo sem os *hooks* – de módulos binários estarem ligados ao kernel é considerada uma violação em potencial da GPL. Linus decidiu, nos primeiros tempos do Linux, que isso seria permitido. Provavelmente hoje se arrepende amargamente disso: alguns afirmam que ele nunca teve o direito de permitir tal coisa. O debate só poderia ser resolvido num tribunal – e um caso como esse ainda não ocorreu.

Hooks para módulos binários permitem que sejam usados em qualquer versão do kernel. Sem os ganchos, o módulo tem que ser recompilado para cada versão diferente de kernel. Uma das razões pelas quais os hooks para módulos de código fechado não são permitidos é a de encorajar os fabricantes de hardware a produzir drivers de código aberto e dificultar a geração de lucro (às custas da comunidade) pela venda de seu módulo binário. Greg Kroah-Hartmann, o homem que faz a ponte entre os desenvolvedores do driver PWC e o kernel, finalmente decidiu remover o hook em uma versão recente do driver, mantido por Nemosoft Unv. A resposta de Nemosoft foi exigir que seu driver fosse removido inteiramente do kernel em vez de sobreviver por lá de forma incompleta – sem a parte de código fechado, a funcionalidade da câmera é reduzida.

Há muitos detalhes espinhosos. Tecnicamente, a decisão de Nemosoft de lançar o código sob uma licença livre torna legalmente impossível revogar os direitos de qualquer um de continuar usando e distribuindo seu código, mesmo modificado. Portanto, o kernel do Linux poderia, dentro da mais completa legalidade, continuar a distribuir o código sem se importar com a opinião ou o autor. Entretanto, Linus Torvalds entendeu que era uma questão de princípios e não de licenças e, por isso, concordou em retirar completamente o driver de Nemosoft. Essa decisão, entretanto, encontrou resistências e pesada argumentação, num tiroteio vindo de todos os lados.

Alan Cox, em particular, opinou que, se Linus começar a permitir que desenvolvedores retirem suas contribuições do kernel, seria possível que uma pessoa que tenha doado código importante possa reclamá-lo de volta e, com isso, deixar o kernel inoperante. Se o próprio Alan deixar de contribuir com o Linux e levar consigo todas as suas contribuições, a próxima versão do kernel será extremamente pequena e repleta de falhas – isso se funcionar. Mas Linus considera que não é apenas uma questão de direitos do desenvolvedor, mas também de não deixar, dentro do kernel, código sem manutenção. É um conceito bem menos controverso: se Nemosoft não irá manter o código, e ninguém se prontificou a "adotá-lo", o código tem de ser exterminado.

Nos últimos meses, Luc Saillard decidiu aplicar engenharia reversa no módulo binário e lançou uma versão do driver PWC sem o hook e com os melhoramentos advindos da engenharia reversa. O resultado foi um driver que possui muito da funcionalidade do original mas sem a discutível violação de licenças. Se Nemosoft continuará a ajudar no desenvolvimento dessa nova versão ou deixará completamente a equipe é algo que não sabemos. Pode ser, inclusive, que Andrew Morton e Linus tenham outras razões para rejeitar o patch de Saillard, portanto a discussão pode continuar até que outra solução seja encontrada. A principal questão, entretanto, ainda não foi discutida: porque o gancho no kernel foi deixado lá por tanto tempo? ■

Apresentando o blktool, sucessor do hdparm

Quando Mark Lord era o mantenedor do driver IDE nos primeiros tempos do Linux, criou a famosa ferramenta hdparm, um utilitário mais ou menos específico para discos IDE que ajusta vários parâmetros de desempenho. Como muitas outras ferramentas destinadas a alterar parâmetros de hardware, o hdparm pode ser bastante perigoso se colocado em mãos inábeis - embora seja extremamente útil para quem sabe usá-lo. Entretanto, o hdparm é bastante difícil de usar e um tanto direcionado a drives IDE. Recentemente, Jeff Garzik decidiu reproduzir a funcionalidade do hdparm em um novo utilitário chamado blktool, mais fácil de usar e mais genérico que o hdparm.

O blktool é, ainda, algo direcionado ao universo IDE, mas Jeff tem planos de estender o suporte a dispositivos SCSI, I20 e RAID em um futuro próximo. Mais para a frente, o blktool pode ser melhorado para suportar qualquer dispositivo de bloco, como seu nome sugere.

Muito da complexidade do hdparm reside em sua interface na linha de comando, que Jeff está tentando deixar mais limpa. Entretanto, sua solução inicial não foi aprovada por Alan Cox e parece que haverá um certo debate sobre a interface apropriada para a ferramenta. Jeff está tentando agradar a gregos e troianos, mas sua abordagem parece resultar numa complexidade maior do que a prevista – afinal, é para ser mais fácil de usar que o hdparm, não é? O problema todo resume-se na dificuldade de organização das muitas funções e características da ferramenta em um comando com opções simples de usar. À medida que o blktool ofereça suporte a mais arquiteturas, o problema aumentará. Esperemos para ver qual será a solução adotada. ■