Notícias do kernel

Git

O trabalho em torno das ferramentas de controle de versões *git* e *Cogito* continua a todo vapor. A lista de discussão do git recebe de dois a três **megabytes** de mensagens por semana. O código fonte recebeu mais de mil e trezentos patches nos primeiros dois meses de existência do projeto.

Outros projetos ligados ao kernel nos quatro cantos do mundo estão migrando do *BitKeeper* para o git. O *NTFS* foi um dos primeiros a adotá-lo, migrando no início de maio – o git tinha então apenas um mês de vida. Desde então, a fila de fãs ensandecidos só cresce: *libata*, *JFS*, o driver de acesso à rede, *ALSA*... Até a árvore da série 2.4 mantida por Marcelo Tosatti o adotou, bem como os mantenedores da versão estável **w.x.y.z**, a chamada "árvore dos otários", mantida por Greg Kroah-Hartman e Chris Wright. Andrew Morton decidiu não usar nenhum controle de versões em sua famosa árvore *-mm*, mas Matthias Urlichs escreveu um *script* para "enfiá-la", junto com todos os patches, em um repositório git.

O BitKeeper virtualmente desapareceu das paragens do kernel do Linux. Sua documentação foi removida da árvore e todas as menções a ela foram simplesmente apagadas – nem a velha União Soviética expurgava com tal eficiência. É impressionante a velocidade com que o desenvolvimento do kernel voltou à normalidade – até melhor que o normal. Alguns desenvolvedores afirmaram que em alguns aspectos o trabalho com a dupla git/Cogito é muito mais eficiente do que com o BitKeeper.

Mesmo assim, a súbita ausência do BitKeeper – e a urgência de se criar um substituto – atrasou o cronograma do próprio kernel, particularmente a versão 2.6.12. A procrastinação foi tamanha que levou alguns dos desenvolvedores a se perguntar se algum dia ele seria lançado. Depois de alguns cálculos e discussões, chegou-se à conclusão de que a nova versão poderia ser aprontada em uma semana, no máximo duas – o que quer dizer que, quando esta revista estiver nas bancas, o kernel 2.6.12 já estará nas ruas. A versão *release candidate* 2.6.12-rc3 foi a primeira a usar o git. Desde então, já houve três versões *-rc*, todas administradas via git. Em um determinado momento, Andrew Morton sugeriu que a versão 2.6.12-rc6 (já disponível) fosse a última antes do 2.6.12. Vamos ver se conseguem.

Nesse meio tempo, uma nova safra de ferramentas e serviços auxiliares está sendo cultivada ao redor do git. Martin J. Bligh e uma turma da IBM engendraram um sistema automático de testes para avaliar as versões do kernel – abrangendo os lançamentos oficiais e os *snapshots* noturnos gerados pelo git. O site kernel.org, lar-doce-lar de todos os kernels do Linux, improvisou uma página que permite aos interessados vasculhar as muitas dúzias de repositórios hospedados por lá. Para conferir, visite kernel.org/git. Também estão disponíveis várias ferramentas de pesquisa a repositórios git, algumas inclusive baseadas em web. A lista de discussões, antes usada para anunciar mudanças no BitKeeper, foi reciclada para registrar os *commits* dos patches para o git.

Obviamente, o próprio git está em constante desenvolvimento – e por gente de peso. O próprio Linus, pai da criança, tem papel importante em seu crescimento; entretanto, agora que a idéia principal foi transformada em realidade, ele está tentando associar várias partes do projeto a outras pessoas. Uma delas é Petr Baudis, que mantém o Cogito. Junio C. Hamano também é extremamente ativo, implementando muitos dos recursos do git. Além deles, há uma multidão de outros contribuidores fortemente ativos. O mais provável é que nos próximos meses a participação de Linus no git e no Cogito seja cada vez mais esparsa e o projeto ganhe vida própria. Entretanto, não há dúvida de que o filho mais novo de Torvalds vai cegamente atender a todos os desejos de seu progenitor. ■

O estado atual da versão estável

Quando a nova série estável **w.x.y.z** emergiu do mosaico de considerações a respeito da direção que o kernel 2.6 deveria seguir, Linus Torvalds não acreditou que ela tivesse muito sucesso. Para ele, o trabalho de manter essa árvore seria por demais oneroso e ninguém se apresentaria para a missão – ou, se houvesse loucos suficientes, ninguém agüentaria por muito tempo. Para piorar, Linus criou uma série de requisitos bem estritos, indicando exatamente que tipo de *patch* poderia ser aceito na árvore **w.x.y.z** e quanto tempo eles deveriam ficar em "quarentena" antes de serem aceitos. Talvez por isso, o criador a tenha chamado de "árvore dos otários".

Mas Greg Kroah-Hartman e Chris Wright decidiram se apresentar para a empreitada, e desde então produziram nada menos que doze versões. Houve algumas rusgas no início – patches que foram aceitos mas que Linus achou que deveriam ter sido melhor debatidos. Isso foi contornado e o desenvolvimento na **w.x.y.z** segue de vento em popa numa rotina bastante previsível. Greg e Chris anunciam um dado conjunto de patches e os colocam em discussão, a cambada quebra o pau por alguns dias e, baseados nos hematomas, definem como será a versão final e sua data de lançamento.

Inúmeros desenvolvedores como Jeff Garzik e Alan Cox louvaram o trabalho da dupla Greg & Chris, confirmando que realmente a árvore **w.x.y.z** é, sem sombra de dúvida, muito mais estável e que está pronta para o horário nobre – ou seja, os computadores das pessoas. A antiga noção – até então verdadeira – de que kernel bom é o que foi "mexido" pelas distribuições já não é mais uma verdade incontestável. O kernel já anda pelas próprias pernas sem precisar da ajuda dos "padrinhos" Debian, Red Hat e cia. ■

IDE

O *driver* para dispositivos IDE mais uma vez foi argumento para incendiar discussões entre desenvolvedores, que ultimamente andaram trocando tiros com artilharia pesada. Os mantenedores desse driver parecem estar condenados ao inferno de Dante. Desta vez Alan Cox e Bartlomiej Zolnierkiewicz estavam em lados opostos no campo de batalha. Alan – que impediu o IDE de se auto-implodir no início do período 2.6 – é de opinião de que toda a camada IDE está se degenerando em código imundo e quebrado. Já Bartlomiej – o mantenedor atual – diz que nada está quebrado, nenhuma funcionalidade foi removida e que Alan teve todas as oportunidades do mundo para reclamar quando cada patch foi liberado e anunciado na lista de discussões.

A maldição do driver IDE é algo demoníaco. Não tanto no código, mas no próprio hardware. Como se não bastassem todos os padrões oficiais (nem sempre em sintonia), ainda temos que tolerar as implementações incompatíveis. O mantenedor do IDE tem, portanto, que intuir todos os casos especiais, refinamentos, versões conflitantes, hardware difícil de identificar e toda a sorte de malefícios e, depois de tudo, fazer com que todos eles funcionem. Para piorar, o IDE é um dos pilares fundamentais do kernel do Linux. Se algo não funcionar – mesmo que seja uma pequena fração – a mortandade será gigantesca. Não há compromissos: o IDE tem que funcionar. Ponto final.

Muitos papas do desenvolvimento tentaram manter essa fatia do kernel. Mark Lord foi o maquinista durante anos, mas em um cruzamento de linhas férreas trombou com Linus Torvalds e largou a locomotiva. Depois disso Andre Hedrick deu seu sangue ao projeto, mas acabou em Arkham como os outros. Martin Dalecki, a pedido de Linus, tomou as rédeas do IDE quando começou a série 2.5. O problema de Dalecki foi seu estilo "peça redonda + buraco quadrado = marreta": em pouco tempo os desenvolvedores trataram de defenestrá-lo, já que o código do IDE estava sempre quebrado, qualquer que fosse a versão.

Depois desse caos todo, Alan tomou para si a tarefa hercúlea e consertou ele mesmo o driver, transformando o impossível em apenas ordinário. Como não queria carregar essa cruz para sempre (e ninguém pode culpá-lo…) a vaga acabou sendo preenchida por Bartlomiej.

Não está muito claro o que vai acontecer daqui para frente. Se a história nos ensinou alguma coisa, foi que Bartlomiej vai ter uma roça e tanto para carpir… ■

Violação da GPL

Carlos Silva denunciou a possibilidade de uma violação da licença GPL pela empresa Panda Software. Segundo Silva, o produto *GateDefender 8200* é baseado no Linux, mas isso não é mencionado nenhuma vez no site nem na documentação que o acompanha. Quando Carlos requisitou o código fonte do kernel do sistema para dar uma olhada, a empresa se negou a fazê-lo alegando que o kernel possui código proprietário. Se isso for verdade, é uma clara violação à licença GPL, General Public License. As violações à GPL pipocam aqui e ali com certa regularidade e costumam crescer criminosamente, mesmo depois que a empresa que cometeu a violação é avisada. Esperemos que não seja o caso desta vez. ■

Escrevendo em sistemas de arquivos só de leitura

Markus Klotzbuecher escreveu uma pequena ferramenta chamada *mini_fo* que aparentemente faz o impossível: permite que os usuários gravem dados em arquivos e sistemas de arquivos só de leitura. Não se assuste, não é uma violação de segurança e não há risco para os arquivos. O novo sistema de arquivos virtual de Markus trabalha enganando o usuário. Ele cria uma área na qual o usuário pode escrever em um sistema de arquivos apropriado e sobrepõe esses dados aos do sistema só de leitura. Os usuários parecem estar editando arquivos na área protegida mas, na verdade, estão apenas criando um *diff*.

Quando alguém tenta ler os arquivos, vê os originais só de leitura com as alterações sobrepostas, e parece mesmo que o arquivo foi alterado. Os usos para essa ferramenta incontáveis, como em sistemas embarcados que querem manter uma área protegida mas que ainda assim permitir que os usuários escrevam nela. ■

SMP em cluster

Dinakar Guniguntala tem perdido noites de sono para trazer à luz um meio de agrupar em cluster conjuntos de CPUs em sistemas SMP. Dessa forma, certos processos podem ficar restritos a um determinado conjunto de processadores. Esse é um dos muitos aprimoramentos no agendamento de processos (*scheduling*) do Linux, coisa que vem tomando forma nos últimos anos.

Entretanto o trabalho de Dinakar, mesmo sendo impressionante, causa celeuma, já que desenvolvedores do calibre de Nick Piggin e Paul Jackson apontam senões em sua implementação. Além disso, há problemas de compatibilidade entre esse patch e outros como o *hotplug* de CPUs. Ao que tudo indica, a coisa vai gravitar ao redor do kernel como um patch independente por muito tempo ainda, até que as arestas estejam aparadas e os inúmeros problemas de compatibilidade, resolvidos. ■