Notícias do Kernel

GIT

Mês passado o controle de versões do kernel estava de pernas pro ar, pois Linus havia abandonado o *BitKeeper*. Não haviam outras alternativas à altura e dezenas de opções estavam sob o escrutínio de Linus. Nenhuma, entretanto, parecia satisfatória. No espaço de poucas semanas, entretanto, o panorama mudou completamente. Do caos emergiu a luz, sob a forma de uma entidade espiritual chamada *git* (pronunciada com um 'g' forte). O git revelou-se a nós, mortais, não só como a escolha natural de Linus para o porvir mas, também, como um grande avanço na arena dos sistemas livres para controle de versões. Todos os concorrentes ficaram para trás, comendo poeira. Mesmo as soluções mais antigas como o *arch* e o *darcs* estão colocando a mão na consciência e se integrando ao git por debaixo dos panos. Algo vultoso está acontecendo.

Quando Linus iniciou o desenvolvimento do git, imaginava que ele fosse um mero sistema de arquivos com comandos de baixo nível para rastrear o conteúdo de um dado diretório. Como descrito pelo Criador, esses comandos poderiam ser abstraídos como um tipo de 'chamadas de sistema' (*system calls*) para operações básicas. Linus nunca quis que o git possuísse um conjunto de recursos para controle de versões digno desse nome. Sua esperança era a de que alguma boa alma usasse o git como fundação para criar um sistema verdadeiro – preferencialmente na forma de scripts – que fizesse o que se espera dele.

No momento, o projeto *Cogito*, de Petr Baudis, parece ser o que mais se aproxima do que Linus sonha todas as noites. É bastante provável que seja essa a ferramenta que usaremos para controlar o desenvolvimento do kernel. Porém (tudo tem um porém...) estamos nos primeiros minutos do primeiro tempo e muita coisa – boa ou ruim – pode acontecer. A única coisa que parece moderadamente certa é a de que o git terá um futuro glorioso no desenvolvimento do kernel. Quiçá em todo lugar...

Alguém poderia perguntar: "Mas que demônios está acontecendo aqui"? Durante todo o tempo (anos!) em que Linus usou o BitKeeper, legiões de fundamentalistas do software livre – e, diga-se de passagem, excelentes desenvolvedores – trabalharam feito escravos para criar uma alternativa livre ao BitKeeper. Entretanto, nenhum deles chegou nem perto de algo minimamente usável. Agora, dias após o BitKeeper deixar o palco, um substituto sólido e confiável cai de pára-quedas no meio da cena. O que aconteceu? A resposta é: muita coisa!

- **1.** Nenhuma das alternativas livres possuía o benefício de saber, exatamente, o que Linus queria. O chefe, de tempos em tempos, divulgava suas idéias e reclamações a respeito dos vários sistemas livres existentes, mas segredava apenas aos desenvolvedores do BitKeeper o que ele realmente queria, chegando a fazer análises específicas do produto. Com essa sonegação de informações, todas as alternativas livres acabavam por desperdiçar um tempo enorme em recursos desnecessários enquanto inadvertidamente negligenciavam o que era realmente importante.
- **2.** Linus desenhou o sistema de arquivos git para ser rápido e para permitir o desenvolvimento distribuído. Ele não se distraiu com recursos mais complicados que pareciam supérfluos para ele. Em vez disso, ateve-se à sua idéia principal e não permitiu que o desenvolvimento saísse de seu controle. O projeto avançou, portanto, de forma bastante objetiva.
- **3.** Todo mundo colaborou. Quando Linus anunciou que pretendia "dar um tempo" do kernel para trabalhar no git, centenas de desenvolvedores talentosos (incluindo aí algumas sumidades) o seguiram, trabalhando incansavelmente dia e noite para entender a proposta do mestre e criar um exoesqueleto em volta de seu rebento.

Um dos argumentos de Larry McVoy, CEO da BitMover, em favor do BitKeeper era o fato de que o modelo aberto de desenvolvimento não era apropriado para criar uma ferramenta complexa como o BitKeeper, com todas as arestas aparadas, detalhes de implementação horrorosos e um trabalho duro e maçante – em sua opinião, como os desenvolvedores de SL podem "escolher o serviço", ninguém iria pegar tal tarefa enfadonha e espinhosa. Por anos e anos o senhor McVoy impingiu o BitKeeper goela abaixo dos desenvolvedores do kernel como prova cabal de que os métodos do software livre eram, por definição, ineficientes. E por anos e anos ninguém teve a competência para provar que ele estava errado.

Mesmo agora, a velocidade de desenvolvimento do git deve muito ao BitKeeper, já que os desenvolvedores estão acostumados com seu "jeitão" e seus recursos. Com toda a certeza, a incapacidade dos desenvolvedores de todas as alternativas livres ao BitKeeper em conseguir algo que chegasse pelo menos perto deve continu-

ar a ser objeto de discussão e pesadelos em toda a comunidade, um fato que não deve ser esquecido. Mesmo que a coisa funcione com o git, algo tem que mudar na motivação dos desenvolvedores de agora em diante.

De qualquer forma, o advento do git é um fenômeno espantoso, muito além das chocantes expectativas de qualquer um. O muito que temos a aprender de todo esse lodaçal é deveras importante. Não perca a lição. ■

❒ Os mentores do kernel

Da série "já-não-era-sem-tempo": Matt Mackall iniciou uma nova lista de discussão chamada de *Kernel Mentors*. O propósito é simples e nobre: guiar os novos desenvolvedores na arte de ter seu código aceito na árvore principal do kernel. Como o próprio Matt reconhece, as barreiras técnicas e sociais para isso podem ser apavorantes.

Isso é verdade indiscutível, mas muitas das dificuldades encontradas não são assim tão antigas. Muitos problemas obscuros ocorrem quando os veteranos emitem sua opinião a respeito da maneira correta de se implementar uma dada porção de código – a "polidez" e a "simpatia" dos desenvolvedores mais velhos é proverbial. Muitas vezes os novatos não esperam uma crítica tão aguda a respeito de seu código e não conseguem ver lógica nenhuma nas mudanças requeridas – que muito freqüentemente exigem um retrabalho considerável.

Outras dificuldades são, também, recorrentes. Por exemplo: a quem devo encaminhar um *patch*? Como dividir o meu patch em pedaços menores? Mesmo que essas questões não pareçam ser tão difíceis assim, alguns patches ricocheteiam pra lá e pra cá até chegar ao destino apropriado; patches muito grandes podem ser difíceis de serem quebrados em pedaços menores. isso tudo é muito frustrante e desestimulante.

O que quer que seja dito sobre o assunto, aos não-iniciados as práticas de desenvolvimento do kernel parecem intangíveis. O projeto Kernel Mentors é uma iniciativa mais que bem-vinda ao universo Linux. Esperamos que se possa arrebanhar novos desenvolvedores e conduzi-los com segurança por entre os traiçoeiros picos que todos conhecemos. ■

❒ Fusão dos projetos SCSI

Os projetos *linux-iscsi* e *open-iscsi* se uniram para formar o *iSCSI*, que usará a base de código do open-iscsi como ponto de partida. Aparentemente, as discussões entre as duas equipes já estavam bem encaminhadas há algum tempo. Todos concordaram que "a união faz a força" e esperam (até que enfim) desenvolver uma pilha SCSI decente para o Linux.

Depois de decidir qual base de código seria usada como ponto de partida, começou a "briga" para escolher o sistema de controle de versões: o open-iscsi usava o *Subversion*, enquanto o linux-iscsi preferia o decano *CVS*. A curto prazo, decidiu-se pelo Subversion, mas com todo o bafafá em torno do BitKeeper e do git, qualquer coisa é possível. ■

❒ O novo ConfigFS

Joel Becker criou o sistema de arquivos *ConfigFS*, (mais uma) interface enfiada no kernel para fazer companhia a *ioctls*, *ProcFS*, *udev*, *SysFS* e *DebugFS* (sem contar o git). Nas palavras de Joel: "o ConfigFS não é um substituto para o SysFS ou o ProcFS e deve coexistir com eles". Um dos principais objetivos do ConfigFS é apresentar uma interface bastante clara e completamente automatizável por *scripts*.

O ConfigFS ainda está muito verde e muitos de seus detalhes de implementação ainda devem ser trazidos à baila. Um deles é crucial: em que pontos, exatamente, uma alteração em um sistema de arquivos ConfigFS modifica a parte do kernel que ele influencia.

Ninguém está convencido ainda de que há a necessidade de *mais um* sistema de arquivos para servir de interface ao kernel. Aparentemente, o ConfigFS foi inspirado pelo fato de que nenhuma das soluções existentes – incluindo a caçula da família, a SysFS – seja satisfatória. ■

❒ Máquinas novas no *kernel.org*

A Hewlett-Packard doou um par de servidores novinhos em folha para o kernel.org. As máquinas, modelo DL585, possuem quatro processadores Opteron cada e vêm com 24 GB de RAM e 10 TB de disco rígido. A monstruosidade de espaço de armazenamento é obtida com outro par notável, as *storage arrays* MSA-30. Cada servidor possui um par dessas belezinhas. A atualização veio em boa hora, já que o planeta todo açoita o kernel.org a cada vez que um novo núcleo do Linux é lançado. Ambos os servidores entraram em operação já no mês de abril e não foram notadas falhas graves desde então. ■

❒ Quebra de compatibilidade no FUSE

Miklos Szeredi faz algumas modificações no FUSE (***F**ilesystem in **USE**r-space*) que mudaram completamente a interface com o usuário, quebrando a compatibilidade com os sistemas existentes. As mudanças eram necessárias para que as implementações de 32 e 64 bits fossem compatíveis. Franco Broi ajudou a testar o patch.

A existência tumultuada do FUSE está mais incerta do que nunca, mas sua continuidade na árvore de Andrew Morton – a famosíssima *-mm* – garante uma base de usuários bastante sólida. A mudança proposta por Miklos parece ser aquele tipo de patch conhecido como "analgésico": a infra-estrutura tem que ser consolidada para evitar as freqüentes dores de cabeça dos desenvolvedores. Isso era um remédio que não podia mais esperar para ser receitado. ■