DIGERATI

Como fazer parte de um time de desenvolvedores de sistema Pág.6

Geek especial

Ano III - Número 28 - R$ 11,90

#28

ISSN 1517-4390

9 771517 439003 28

Live CD Slax

DISTRO LINUX BOOTÁVEL COMPLETA BASEADA NO SLACKWARE

Construa seu próprio LINUX

FERRAMENTAS, TUTORIAIS E CÓDIGOS PARA:

- Criar sua distro personalizada
- Criar um sistema operacional do zero e desenvolver sistemas em Assembly e C

+ 5 SISTEMAS COMPLETOS

• Athene • FreeDOS • BraSO • MenuetOS • ReactOS

Veja mais no verso

Emuladores de Sistema

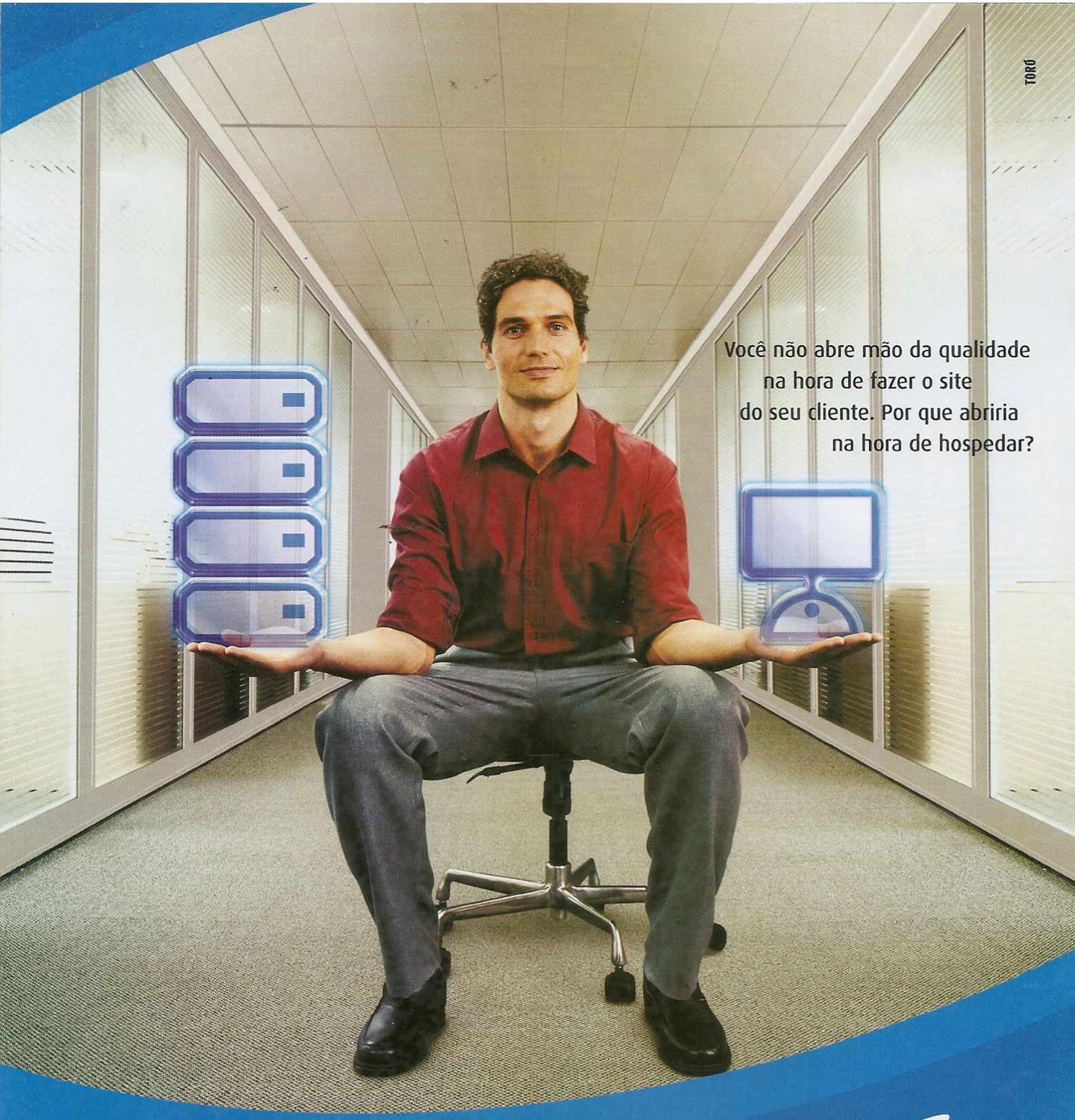
Programas para rodar

- Windows no Linux
- Linux no Windows
- Win95 no XP
- Aplicativos DOS no XP

e muito mais

Bônus:

- Mais de 150 apostilas
- SO para Dreamcast
- Ferramentas para melhorar o desempenho do PC



Você não abre mão da qualidade na hora de fazer o site do seu cliente. Por que abriria na hora de hospedar?

HOSPEDE O SITE DO SEU CLIENTE COM O MELHOR CUSTO-BENEFÍCIO: TODA A QUALIDADE DO PRESENÇA WEB A PARTIR DE R$ 19,90 POR MÊS*.

Alta disponibilidade, segurança, facilidade de utilização e garantia de escalabilidade para qualquer volume de informações. É isso que o seu cliente ganha colocando o site na Internet com a gente. E, para você, desenvolvedor, o Presença Web tem um programa especial de parceria, em que você ganha mais e todo mês, enquanto seu cliente estiver aqui. Conheça e cadastre-se já para começar a ganhar.

br turbo empresas

NOSSO DOMÍNIO É A INTERNET.

0800 61 8700

www.brturboempresas.com.br

BrasilTelecom

*Promoção válida até 31/7/05 para as novas assinaturas do pacote Presença Web Simples. Valor válido para os 12 primeiros meses do contrato.

Geek ESPECIAL

Diretor editorial: Alessio Fon Melozo
Coordenador editorial: Hudson de Almeida
Editor de arte: Daniel Brito

REDAÇÃO
Editor-assistente: James Della Valle
Reportagem: Cristine Vecchi e Emersom Satomi
Arte: Sérgio Bergocce, Felipe Luiz Fatarelli e Fabio F. Souza
Revisão e checagem: Elisabete B. Pereira e Sirlene S. Farias

COLABORARAM NESTA EDIÇÃO
Alexnaldo C. dos Santos e Angelo Valle (texto)

MULTIMÍDIA
Rafael Scopel (coordenador), Aleksandro Botelho e Cleber Faria (seleção de conteúdo) e Felipe Carmo (design e programação)

PUBLICIDADE
Cassetari Assessoria em Comunicação

ATENDIMENTO AO LEITOR – SUPORTE
Horário de atendimento das 9 às 21h
e-mail: atendimento@digerati.com.br, suporte@digerati.com.br, tel.: (11) 3217-2626

EDIÇÕES ANTERIORES
Atendimento a jornaleiros: (11) 3217-2606
Canais de vendas: (11) 3217-2600
e-mail: vendas@digerati.com.br, fax: (11) 3217-2647
Site: www.lojadigerati.com.br

CONTATO
Redação: R. Haddock Lobo, 347, 12º andar, São Paulo - SP, CEP 01414-001, tel.: (11) 3217-2600, fax: (11) 3217-2617
Publicidade: (11) 3217-2627
e-mail: publicidade@digerati.com.br
Representante comercial nos EUA: USA-Multimedia
tel.: +1-407-903-50000, Ramal: 222
e-mail: info@multimediausa.com
Marketing: (11) 3217-2600
e-mail: marketing@digerati.com.br
Circulação: (11) 3217-2719
e-mail: circulacao@digerati.com.br

DIGERATI

GEEK ESPECIAL (ISSN 1517-4395)
é uma publicação da editora Digerati.
Distribuidor exclusivo para todo o Brasil: Fernando Chinaglia Distribuidora S.A. Tel.: (21) 3879-7766.
Impressão: Oceano Indústria Gráfica Ltda.

DIGERATI É UMA EDITORA DO GRUPO DOMO

GRUPO DOMO

Presidente: Alessandro Gerardi

Conselho editorial: Alessandro Gerardi, Luís Afonso G. Neira, Alessio Fon Melozo, William Nakamura

ANER
www.aner.org.br

No início, eram apenas calculadoras gigantes, sem alma. Nos anos 50, operadores tinham de fornecer todas as instruções manualmente para que computadores conseguissem carregar aplicações. Dez anos depois, a coisa evoluiu. Começaram a surgir os primeiros sistemas operacionais.

Essa é uma linha evolutiva onde revoluções não faltam. Passa por aqui – na digitação deste texto – e vai muito além.

Por essas e outras é que dedicamos esta edição à nobre tarefa hacker de colocar alma na máquina: o desenvolvimento de sistemas operacionais.

A introdução para esse universo está nas próximas páginas, desde como montar sua própria distro GNU/Linux – o neto pródigo do primeiro sistema operacional-padrão para computadores: o Unix –, até a criação de um sistema básico do zero, em C e Assembly.

No mundo da programação, poucas tarefas são tão desafiantes e enriquecedoras como a criação de um SO. Envolve a manipulação hands-on de bits, barramentos, trilhas de disco, processamento... Mas foi para isso mesmo que inventaram a programação. Então, mãos à obra e pau na máquina!

A redação

Introdução p.04
Conheça o funcionamento de um sistema operacional

Software livre p.06
Contribua com projetos em andamento e enriqueça seu currículo

Tutorial p.14
Desenvolva sistemas em C e Assembly

Monte seu Linux p.24
Crie sua própria distribuição utilizando o *Linux From Scratch*

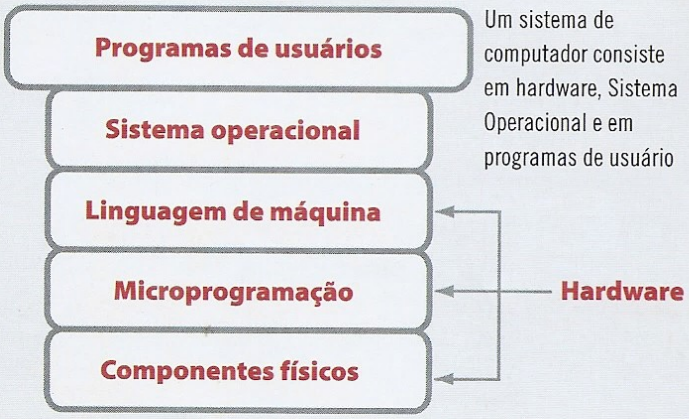
Guia do CD p.29
Confira o conteúdo do CD-ROM que acompanha esta edição

O que é um Sistema Operacional

Conheça a história e o funcionamento dos sistemas operacionais antes de começar a criá-los

Por Alexnaldo C. Santos *

Modernos computadores são fabricados para executar uma ou mais tarefas através da manipulação de seus componentes. A necessidade de gerenciamento de seus componentes e expansão de suas tarefas, sem necessariamente refazer a máquina, levou ao desenvolvimento de uma arquitetura que permitisse que a máquina fosse manipulada através de programação. Dessa forma, ela teria um microprograma (microprogramação) que faria a interpretação de um conjunto de comandos (linguagem de máquina) para manipular seus componentes. Em suma, um sistema complexo. Escrever programas que controlassem todos esses componentes e usá-los corretamente era um trabalho extremamente difícil. Para resolver tal problema era necessário a criação de um programa que fizesse uma abstração de toda essa complexibilidade. A partir daí criou-se o conceito de Sistema Operacional, que funciona como uma máquina virtual entre os programas e os componentes da máquina real. Veja adiante como ficou essa abstração em um sistema de computador.



Um sistema de computador consiste em hardware, Sistema Operacional e em programas de usuário

A história dos Sistemas Operacionais

A história dos Sistemas Operacionais está interligada ao surgimento dos computadores que, em meados da década de 40, funcionavam a válvulas e painéis de conectores, e sua programação era feita diretamente em linguagem de máquina (não existia ainda a linguagem Assembly). O programador tinha de inserir manualmente suas conexões no computador e esperar horas pelo resultado, que só seria positivo caso nenhuma das milhares de válvulas estivesse queimada. Todo esse processo era para executar simples cálculos numéricos. Logo em seguida, surgiram os cartões perfurados possibilitando que os programas fossem gravados e lidos quando necessário. Qualquer problema no programa implicaria a utilização de outro cartão.

Com o surgimento dos transistores, em meados da década de 50, o modelo de programação do computador mudou radicalmente, tornou-o mais confiável, permitindo sua fabricação para revenda. Devido ao preço exorbitante pelo que era vendido, sua aquisição era restrita a grandes corporações, universidades e órgãos do governo. Nessa época, nasceram as linguagens Assembly e Fortran, contudo, para executar uma tarefa, ainda era necessário utilizar os cartões perfurados. Uma simples tarefa consistia em escrever o programa em papel, perfurar os cartões, levá-los à sala dos computadores e, então, esperar horas pelo resultado. Enquanto um programa era executado, outro programador não podia nem mesmo deixar o seu cartão para execução em seguida. A fim de tornar isso possível, surgiu o conceito de sistema de processamento em lote (batch system), que permitia que os cartões fossem colocados em cima de uma bandeja para serem lidos em fitas magnéticas. Em seguida cada fita era lida, executada e seu resultado era impresso após horas. Sistemas Operacionais típicos eram o

FMS (Fortran Monitor System) e o IBSYS da IBM.

Esse processo perdurou até meados da década de 60, tendo a IBM como líder no segmento, que introduziu o computador System/360 com o uso de Circuitos Integrados. Na época, utilizava-se o Sistema Operacional OS/306 cuja finalidade era, em teoria, permitir que um programa rodasse sem problema em qualquer um dos computadores da linha 360, 370, 4300, 3080 e 3090. Essa pretensão de permitir portabilidade de programas tornou o OS/360 complexo de desenvolver e manter. No OS/360, nasceu o conceito de multiprogramação, que permitia que, quando um programa esperasse pelo processamento de E/S (entrada/saída), a CPU(unidade central de processamento) pudesse ser utilizada por outro. Note que, se um programa usasse a CPU por tempo indeterminado, o outro jamais teria a chance de usá-la. Era necessário um sistema que permitisse o compartilhamento de tempo, e foi exatamente isso que o pessoal do MIT(Massachusetts Institute of Technology) fez, porém, como os computadores da época não tinham segurança entre os processos (um programa na memória podia prejudicar o outro), não difundiu muito o recurso. Mais adiante, surgiu o Sistema Operacional Multics (MULTiplexed Information and Computing Service) do desenvolvimento conjunto do MIT, Bell Labs e General Electric, com um modelo que permitia com que vários usuários compartilhassem tempo simultaneamente.

Ken Thompson, cientista da Bell Labs, escreveu uma versão monousuária do Multics – nasceu em seguida o famoso Unix. O que parecia uma maravilha inicialmente terminou em caos, pois, como o código-fonte era aberto, várias empresas criaram suas próprias versões, incompatíveis entre si. Tal compatibilidade só foi atingida quando o IEEE (Institute of Electrical and Electronics Engineers) criou o padrão Posix, que a maioria das versões do Unix suportam até hoje.

Com o surgimento dos computadores pessoais, cresceu a necessidade de Sistemas Operacionais mais amigáveis, na época a AT&T era a detentora dos direitos sobre o Unix. Nessa época a Microsoft comprou uma licença para produzir sua própria versão, denominada Xenix e, logo em seguida, juntou-se à IBM. Sem precisar mais manter compatibilidade com o padrão Posix, produziu o MS-DOS e vendeu o Xenix à SCO. O MS-DOS não tinha modo protegido, partiram então para o desenvolvimento de uma versão que tivesse essa proteção. Ao mesmo tempo criaram o Windows 1.0, um simples ambiente gráfico que rodava sobre o MS-DOS. Antes de lançar a versão 2.0 do Windows, a Microsoft rompeu o contrato com a IBM e lançou a versão 3.1, tornando-se líder de mercado em Sistemas Operacionais.

A AT&T só licenciou o código-fonte do Unix até a versão 6. Dessa forma, a partir da versão 7.0 ele passou a ser comercializado, com conseqüente proibição do estudo do código-fonte. Andrew S. Tanenbaum, professor na época, resolveu desenvolver um Sistema Operacional do zero, para que seus alunos pudessem estudar, mantendo compatibilidade a nível de usuários com o Unix 7.0. Nasceu então o projeto Minix (miniunix). Para difundir o Minux, criou-se uma lista de discussão na Usenet, porém, por ser didático, muitos recursos sugeridos não era implementados por Tanenbaum. Um estudante finlandês, Linus Torvalds, decidiu escrever um close do Minix para que pudesse colocar recursos, em vez de ser uma ferramenta didática e limitada ao estudo. Nascia o Linux.

Presente e futuro dos Sistemas Operacionais

A Microsoft continua líder no fornecimento de Sistemas Operacionais. O Linux tem crescido, mas com um problema parecido ao do Unix pois, como o seu código-fonte é aberto, qualquer pessoa pode modificá-lo, contudo, a distribuição do kernel (núcleo do Sistema Operacional) continua controlada por Linus Torvalds e sua equipe de colaboradores. A instalação e a configuração é seu maior problema, principalmente pelo fato de que cada distribuição instala um conjunto diferente de aplicativos, ou seja, um programa feito para rodar no Linux poderá não funcionar imediatamente em distribuições distintas e obrigar o usuário a configurá-lo corretamente, o que não ocorre na grande maioria dos Sistemas Operacionais comerciais. Para profissionais da área de informática, isso não é um problema, mas para usuários domésticos, que é a grande maioria, isso torna seu uso inviável. Enquanto houver distribuições diferentes, veremos um crescimento abaixo do possível e poucas empresas desenvolvedoras de programas portando seus aplicativos para Linux. Sendo um Sistema Operacional open source, o Linux se tornou o maior defensor da bandeira de livre uso de software e código-fonte aberto e abriu espaço para o surgimento de novos Sistemas Operacionais, alguns com o intuito de ficarem famosos, outros simplesmente para estudos. Na próxima matéria, iremos fazer um estudo do que é necessário para criar um Sistema Operacional, tomando como base o BraSO (Sistema Operacional Brasileiro). Até lá.

Bibliografia

Tanenbaum, Andrew S. e Woodhull, Albert S. Sistemas Operacionais: projeto e implementação, 2ª ed. Bookman, 2000.

Silberschatz, Abraham; Galvin, Peter Baer; Gagne, Greg; Wile John y & Sons Operating System Concepts: 7ª ed.

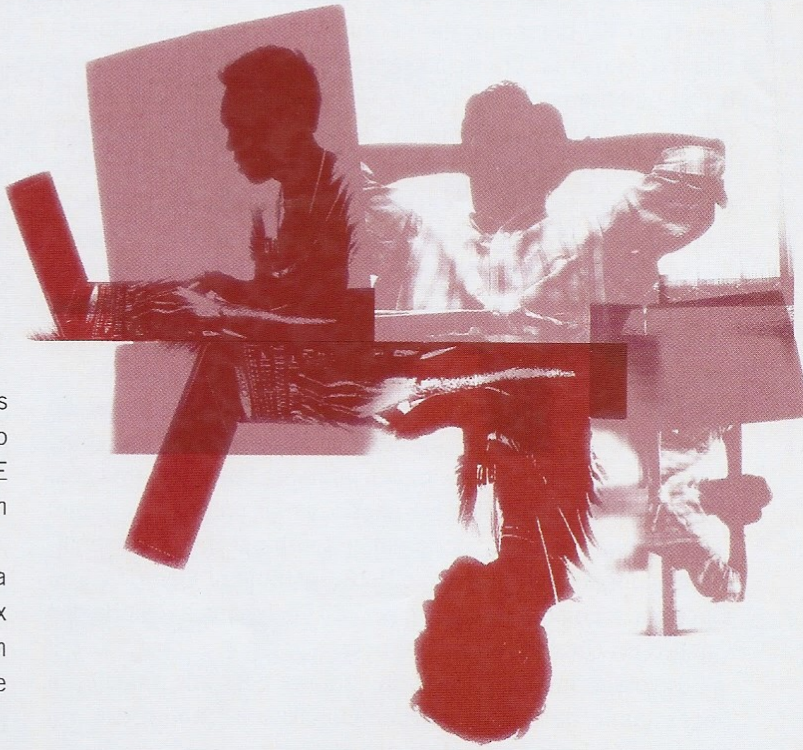
Kaufmann, Morgan Computer Organization and Design: The Hardware/Software Interface, 3ª ed.

* Alexnaldo é mestrando em modelagem computacional e trabalha como arquiteto de software para a Sismont Consultoria e Netwatts Tecnologia na Web. *alexnaldocs@hotmail.com*

APRENDA CRIANDO CÓDIGO ABERTO

Por Emersom Satomi

Criar um sistema do zero é impossível sem ampla experiência. Um bom apredizado está nos projetos open source



Como adquirir conhecimento profundo sobre sistemas operacionais e seu desenvolvimento? Um passo decisivo pode ser o envolvimento com projetos open source. E esse é um universo de portas abertas para os que têm talento e veia geek, independente do país de origem.

Entre os projetos em que há programadores brasileiros com a mão na massa, está o próprio kernel Linux, distros GNU/Linux consagradas, OpenBSD, entre outros. As recompensas podem ser muitas: experiência técnica de alto nível, convívio estimulante

O que se ganha com código aberto?

1 - Experiência prática. Há créditos para todos os programadores responsáveis pelas atualizações, o que enriquece bastante o currículo.

2 - É possível continuar contribuindo com esses projetos no trabalho, já que cada vez mais empresas estão se beneficiando e apoiando tais iniciativas.

3 - Convívio estimulante, já que esse universo é riquíssimo em talentos. Fora a Internet, acontecem também encontros de grupo e conferências.

4 - Para os altruístas, realização por doar seu trabalho e transformá-lo em algo útil e livre, que pode ser usufruído por qualquer um.

Como começar

Triagem

A triagem de bugs é um bom começo para se familiarizar com um projeto open source. É necessário criar uma conta no sistema de registro de bugs, no site do projeto (geralmente, bugzilla.nomedosite.org). Depois, ler os relatórios de bugs dos usuários, agrupar as ocorrências repetidas – já que usuáàios descrevem os bugs de maneira diferente –, e encaminhar para um dos desenvolvedores, seguindo a lista de prioridades de correção de bugs.

Patches

Ao isolar um erro no código-fonte e corrigi-lo, é necessário criar um patch, um arquivo que contém as diferenças em relação ao código original. Depois é preciso enviar esse

(ainda que seja on-line), grandes desafios cumpridos, além da realização pessoal e profissional.

O maior sacrifício costuma ser o tempo livre. Mas isso nunca foi problema para quem realmente se interessa por computadores. A polêmica está presente na completa doação do código produzido. Críticos argumentam que o programador perde autonomia e poder diante de sua obra. Defensores elogiam o caráter comunitário, social e altruísta dessa iniciativa. Mas o que ninguém pode negar é a riqueza de conhecimento e experiência adquiridas no desenvolvimento de um sistema operacional livre.

REQUISITOS

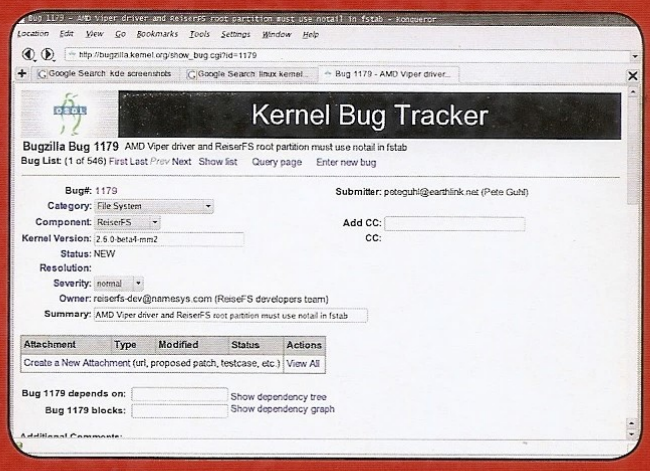
Além do espírito hacker, ler e escrever em inglês é fundamental. Pelo menos para a participação em projetos famosos, internacionais e com muitos desenvolvedores. Neles, essa é a língua oficial, ainda que grande parte dos colaboradores não tenha o inglês como idioma nativo.

Noções avançadas e amplas de programação não são totalmente necessárias já que, normalmente, cada um responde por uma área bem específica. Mas conhecimento profundo do projeto que você pretende participar é fundamental.

Foi esse conhecimento que levou Mauro Carvalho Chehab, 37 anos, a tornar-se o mantenedor da seção Video4Linux do kernel Linux apenas alguns meses depois de sua primeira contribuição.

Chehab é engenheiro no setor de telecomunicações e professor de cursos de graduação em redes, segurança e programação C, e de pós-graduação na área de informática, em Brasília. Contribui com trabalho open source desde maio deste ano. Seu projeto traz compatibilidade a diversas placas de captura de TV e o kernel Linux.

Outro programador em uma posição de destaque é Pedro Martelletto, 22 anos. Atualmente, ele é o único brasileiro que



O sistema online de registro e busca de bugs do kernel Linux, pode ser acessado em *http://bugzilla.kernel.org*

'A recompensa principal é o conhecimento'



Gustavo Franco, 21 anos, é desenvolvedor do Debian GNU/Linux, coordenador de tecnologia da Rede de Informações para o Terceiro Setor (RITS) e atua em diversos projetos de inclusão digital e divulgação do software livre, no Rio de Janeiro

Aos nove anos, Gustavo já devorava apostilas de DOS. Pouco depois aprendeu Basic sozinho, o que o levou a criar programas que geravam ruídos controláveis no speaker do PC (em uma época em que o único som que saía do micro eram "bips").

Além da área de empacotamento no Debian, também atua em subprojetos como o python-modules, que ele mesmo criou, para melhorar a manutenção de pacotes da linguagem Python .

Gustavo não se esquece dos primeiros contatos com o time de desenvolvedores do Debian, por IRC: "São muitas pessoas, é realmente como descobrir uma nova civilização com sua própria organização, tradição e tudo mais. Fiquei impressionado e tentei aprender o máximo que pude em menos tempo possível para poder colaborar de alguma forma".

No ano passado, ele teve a oportunidade de participar do maior evento de desenvolvedores Debian de todo o mundo, a Debconf, que aconteceu em Porto Alegre. "Em poucos dias, aprendi muito mais do que só desenvolver software, mas como uma comunidade diversificada pode colaborar em prol de um bem comum. Fiz novas amizades e, claro, corrigi diversos bugs."

Apesar da experiência, Franco tem outras realizações. "A recompensa principal é conhecimento, mas a minha participação no projeto tem sido um diferencial em meu currículo. Quando falo de conhecimento, também incluo outras culturas, metodologias de trabalho e outros fatores".

Sua dica para os interessados: "Procure entender como os próprios desenvolvedores enxergam o projeto. São eles que fazem a coisa continuar, mantêm os mais de 15 mil softwares e toda a estrutura funcionando, com a ajuda dos usuários. Identifique no projeto suas áreas de interesse e converse com quem está envolvido. Não há dúvida de que haverá trabalho. Procure saber se isso está documentado em algum lugar e comece. Se não estiver, será um bom começo auxiliar na documentação."

contribui com o desenvolvimento do OpenBSD, um sistema operacional derivado do Unix como o GNU/Linux, mas considerado mais estável. Além disso, possui menor número de desenvolvedores e usuários.

Marteletto dedica o tempo que lhe sobra do estudo e trabalho na PUC do Rio de Janeiro ao projeto. Para isso, as linguagens de programação C e Lua (criada na própria PUC) já fazem parte de seu dia-a-dia, além da compilação constante de novas versões e da análise das mudanças no CVS (gerenciador de atualizações de código-fonte).

Para ser aceito como desenvolvedor oficial no OpenBSD, Marteletto precisou mostrar todo seu conhecimento sobre a dinâmica interna do kernel e sua relação com a userland (áreas do sistema fora do kernel). Depois que provou seu talento, não lhe faltou apoio e incentivo para que enviasse mais contribuições.

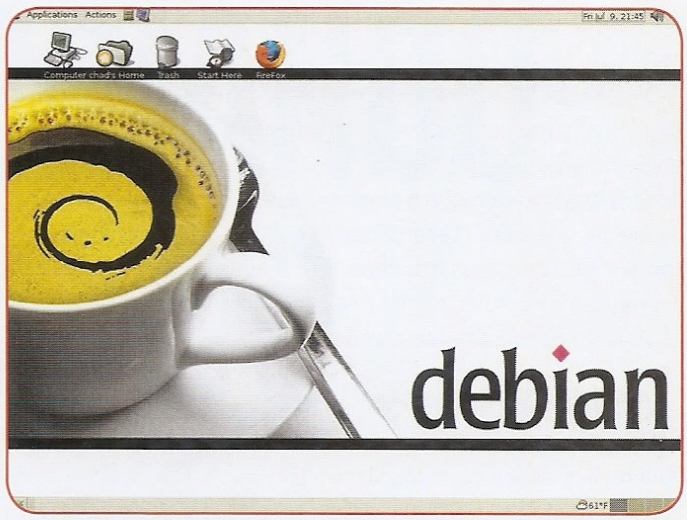
DISTROS

Outra opção enriquecedora para quem deseja aprender e contribuir é participar do desenvolvimento de alguma distro Linux. Aqui também conta muito conhecer o projeto em questão, toda a ideologia open source e de programação.

Gustavo Franco, do Rio de Janeiro, contribui para o Debian GNU/Linux desde os 17 anos. No começo de 2005, aos 21, foi admitido como um dos desenvolvedores oficiais do projeto.

"Precisei comprovar que conhecia a filosofia, que estava apto tecnicamente e que era uma pessoa confiável para ter acesso aos servidores do projeto e tudo mais", conta. Nesse teste, teve de atualizar programas do sistema, isolando e corrigindo bugs, além de adaptar e "empacotar" novos programas. Sua admissão envolveu um processo de avaliação que durou, praticamente, quatro anos.

Atualmente, ele lida principalmente com as linguagens Perl, Python e C, compilação, e com as políticas de inclusão e manutenção do Debian. "No entanto, um novo candidato não precisa ter um profundo entendimento de diversas linguagens. Conhecer a documentação do projeto, ter noções de make e shell script já são um excelente começo", aconselha Franco.



O sistema operacional Debian GNU/Linux (rodando o ambiente desktop Gnome). Gustavo Franco, do Rio de Janeiro, é um dos desenvolvedores desse sistema

Hoje, o trabalho de Gustavo é "empacotar" novos programas para o sistema operacional Debian GNU/Linux, o que inclui avaliar o código-fonte, a licença, as dependências (outros programas necessários para que o primeiro rode), e todas as alterações necessárias para essa inclusão, tarefa nada simples já que, no caso do Debian, os critérios são bem mais elevados do que em outras distros.

Outra clássica distribuição GNU/Linux com participação brasi-

patch para a mailing-list de desenvolvedores do projeto. Se vários outros desenvolvedores aprovarem sua sugestão (e o critério costuma ser bem elevado), incluindo o responsável, ela pode ser incorporada a uma nova versão do programa. De acordo com o número de patches aceitos, seu moral vai subindo no grupo e você pode se tornar um desenvolvedor oficial.

CVS

Em projetos grandes, é necessário estar familiarizado com programas de gerenciamento de versões do código-fonte em rede, responsáveis por automatizar os processos envolvidos na criação e envio de patches. Os mais usados são CVS (*www.cvshome.org*) e Subversion (*www.subversion.tigris.org*).

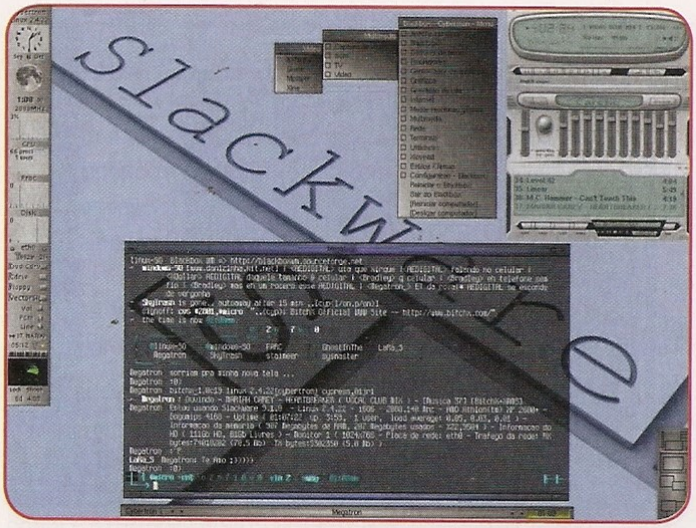
Onde contribuir

GNU/Linux

Distros GNU/Linux e os serviços em torno delas formam o setor open source que mais cresce, com dezenas de milhares de desenvolvedores. O trabalho necessário divide-se entre o kernel Linux (*www.tux.org/lkml*), as ferramentas GNU (*www.gnu.org/*) que completam o sistema e as centenas de distros (*http://distrowatch.com*). No projeto Debian, citado na reportagem, links úteis são a documentação sobre como colaborar (*www.debian.org/devel/join*), o site dos novos mantenedores (*http://nm.debian.org*) e o repositório de subprojetos/outros (*http://alioth.debian.org*).

BSD

Os sistemas Unix baseados no BSD (Berkeley Software Dis-



Uma das interfaces gráficas do sistema operacional Slackware Linux. Roberto Freires Batista, o Piter Punk, de São Carlos (SP) é um dos membros mais ativos do projeto

leira em destaque é o projeto Slackware. Roberto Freires Batista, mais conhecido como Piter Punk, está entre seus desenvolvedores. Piter, inclusive, chegou a substituir o criador do projeto Patrick Volkerding, junto a outros brasileiros, durante um retiro do fundador por problemas de saúde.

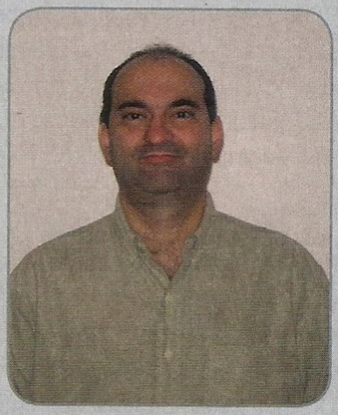
Esse esquema voluntário acaba influenciando bastante o dia-a-dia dos participantes. O movimento open source, por exemplo, inspirou Gustavo Franco, do Debian, a procurar um trabalho compatível com isso. Hoje, Franco é coordenador de tecnologia da Rede de Informações para o Terceiro Setor (RITS), entidade responsável por uma rede de informações para movimentos sociais e ONGs. "Tenho a sorte de, atualmente, ser empregado de uma

tribution) são considerados mais estáveis para aplicações críticas e pesadas do que o GNU/Linux. Possui desenvolvimento mais centralizado e uma licença que permite a ocultação do código. As principais variantes são FreeBSD (*www.freebsd.org*), OpenBSD (*www.openbsd.org*), NetBSD (*www.netbsd.org*) e o Apple Darwin (*www.opendarwin.org*).

Mais alternativos ainda

Há o ReactOS, um sistema aberto compatível com programas para Windows (*www.reactos.com*) e o FreeDOS, clone open source do DOS (*www.freedos.org*). Sistemas mais alternativos ainda, desenvolvidos praticamente como hobbies, também oferecem muito aprendizado para os desenvolvedores. Entre eles, há o MenuetOS (*www.menuetos.org*), e o

'As comunidades open source são bastante receptivas'



Mauro Carvalho Chehab, 37 anos, é mantenedor da seção Video4Linux do kernel Linux, engenheiro no setor de telecomunicações e professor de redes, segurança e programação C em cursos de graduação e pós-graduação na área de informática

Em março deste ano, o engenheiro e professor Mauro Carvalho Chehab, 37 anos, comprou uma placa de captura de vídeo e FM. Mas ela não funcionou 100% sob o GNU/Linux. Como se trata de um sistema operacional de código aberto, ele pôs a mão na massa e aperfeiçoou o código para que a peça funcionasse.

Essa foi sua primeira contribuição para o projeto Video4Linux, que dá suporte para placas de captura de TV e rádio no kernel Linux. Dois meses depois, Chehab já era o mantenedor oficial do projeto.

Fora a programação de drivers e do núcleo dessa seção do kernel, é ele quem coordena os colaboradores, resolvendo divergências de opinião e dando direções para o aprimoramento dos drivers.

O primeiro computador em que Chehab colocou as mãos foi um CP-500, em 1984. Foi nele que aprendeu a programar Basic. Pouco depois, passou para Assembly. Primeiro como diversão, mas pouco depois seus programas de autodiscagem passaram a integrar o pacote de softwares de uma fabricante de modems. Hoje, é um entusiasta do software livre. "A filosofia open source me ajuda a identificar potenciais soluções para meu ambiente profissional", conta Chehab. "O trabalho voluntário fornece recompensas nem sempre tangíveis em bens materiais. No caso do desenvolvimento do Linux, o prêmio é produzir um código de qualidade, utilizado mundialmente".

Sua dica: "Para quem quiser contribuir, o ideal é buscar uma comunidade que já esteja atuando no desenvolvimento. Embora seja possível criar um módulo novo, é mais interessante iniciar em uma estrutura já pronta, com auxílio da comunidade. Há diversas informações interessantes no diretório de documentação do kernel, que é uma leitura obrigatória. Também há boas referências na página Wiki do Video4Linux (www.linuxtv.org/v4lwiki/index.php/Main_Page)".

organização que não impede que o resultado daquilo que desenvolvo no trabalho seja distribuído de maneira livre". Ele também atua em diversos projetos de inclusão digital e divulgação do software livre, no Rio de Janeiro.

Já Pedro Marteletto, do OpenBSD, ainda não chegou a receber convites profissionais devido a seu envolvimento com esse sistema, apesar de essa experiência ajudar em seu trabalho. "Contribuo porque acho interessante e divertido. E a diversão e saciedade que obtenho já são suficientes para me recompensar", conta.

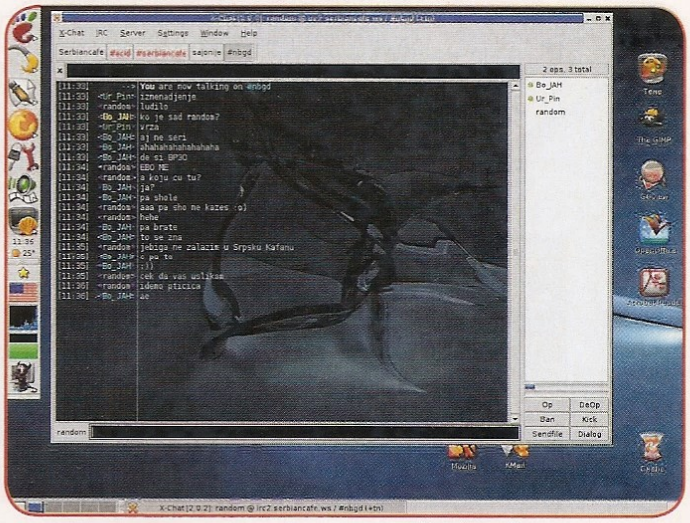
ORGANIZAÇÃO

A organização de todo o desenvolvimento de um sistema open source é uma das etapas mais complexas, à primeira vista. Mas quem participa costuma se entusiasmar com o fluxo livre de informações e o feedback instantâneo. Isso é possível com comunicação on-line (por chat IRC), a ausência de barreiras entre os departamentos e a forte união dos membros para um objetivo comum, coisas raras em um trabalho "normal".

No caso da distribuição Debian GNU/Linux, por exemplo, há mais de mil desenvolvedores. Alguns ocupam posições-chave, para decisões mais globais, e o líder é eleito por todos. Os outros membros organizam-se de acordo com as necessidades dos usuários.

Segundo Gustavo, o ponto forte de um grande projeto open source como o Debian é a comunicação efetiva entre diversos departamentos. Atualmente, o modelo influencia grandes corporações, como IBM, HP e Sun, que precisam de um melhor fluxo de informações entre os setores, conforme aponta.

"No Debian, jamais alguém irá buscar uma solução 'fora de casa', se alguém já a desenvolveu ou pode desenvolvê-la internamente. Empresas terceirizam serviços e projetos de maneira desnecessária, pois há talento internamente para tal. Além de um erro gerencial, é uma falha de comunicação e divulgação interna



Grande parte da comunicação entre os desenvolvedores de um projeto open source é feito através de chat, pelo protocolo IRC

de aptidões, objetivos e projetos em curso", diz Gustavo.

Grandes projetos de software livre otimizam o fluxo de informações com listas de discussão por e-mail, canais de chat separados por assunto, sites de edição colaborativa (tanto de desenvolvedores como de usuários) e muito engajamento, já que no mundo do software livre um colaborador costuma "vestir a camisa" muito mais do que um funcionário faria por sua empresa.

Entre esses projetos, alguns contam com a mítica figura do "ditador benevolente", o líder admirado, com poderes absolutos.

Tal fato ocorre, por exemplo, no kernel Linux, assim como na distribuição Slackware. O criador dessa distro, Patrick Volkerding, está para o Slackware assim como Linus Torvalds está para o kernel Linux.

"O Patrick manda – e manda bem. Já a gente (os desenvol-



O MenuetOS é um sistema escrito todo em Assembly

SkyOS (*http://syllable.sourceforge.net*). Uma lista, que inclui também Linux e BSD, tem links para dezenas de sistemas está em *www.microsuck.com*.

Ambientes Desktop

As interfaces gráficas são a porta de entrada de usuários para o sistema operacional. Sua funcionalidade acaba se confundindo com o próprio sistema operacional, já que a interface atua como intermediária entre o usuário e o sistema de fato. As mais famosas são KDE (*www.kde.org*), Gnome (*www.gnome.org*) e XFCE (*www.xfce.org*).

Ajuda para outras áreas

1 - Documentação

Esse é um dos setores que mais precisam de colaboradores.

vedores) faz sugestões e bug fixes. Aliás, coisa que todo slacker (usuário do Slackware) pode fazer", conta Piter Punk.

No setor em que ele mesmo é o mantenedor, o slackpkg, Piter também tem muito trabalho. "Olho o que me enviam (sugestões, bugs etc.), respondo e-mails e tento solucionar problemas que aparecem, além de mesclar as features (recursos) que o povo pede".

Trabalho parecido tem Mauro Carvalho Chehab, mantenedor do Video4Linux. Nesse projeto há desenvolvedores voltados para cada uma das famílias de chipset das placas de captura. Os patches que envolvem pequenas alterações são aplicados diretamente por cada um deles.

"Já quando o patch requer maiores alterações, analisamos o circuito e discutimos um pouco, até obter uma forma mais definitiva. Nesses casos, normalmente, a versão fina fica por minha conta", explica Chehab.

Os métodos são semelhantes mas, dependendo do projeto, sua aplicação varia nos detalhes. Pedro Marteletto, do OpenBSD, resume a organização do projeto da seguinte maneira: alguém sugere uma alteração para o responsável de alguma das áreas do sistema. Ele, então, precisa argumentar sobre a necessidade dessa mudança e testa repetidamente sua versão. Vai colecionando a aprovação de vários membros, até que recebe o OK para incorporar a mudança na versão oficial.

TEMPO

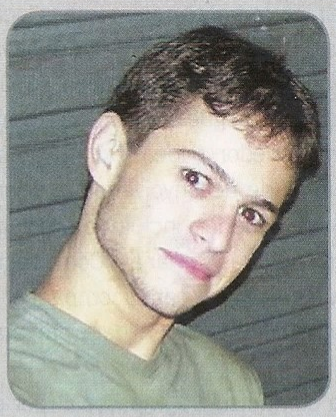
Boa parte das dificuldades que se apresentam nesse tipo de desenvolvimento coletivo tem a ver com tempo, já que trabalhar de graça nas folgas do emprego ou estudo pode se tornar uma situação bem complicada.

A principal dificuldade de Marteletto, do OpenBSD, é justamente essa. "Embora seja muito divertido e aprazível, esse esforço, infelizmente, não me dá dinheiro nem créditos na faculdade". Ele



O ambiente Desktop KDE funciona em GNU/Linux, várias versões de BSD e outros sistemas

'Open source é minha filosofia por um motivo: eficiência'



Pedro Marteletto, 22 anos, é o único brasileiro membro do time de desenvolvimento do sistema operacional OpenBSD. Estuda e trabalha na PUC-Rio

Pedro Martelletto, 22 anos, costuma discordar dos fãs mais ardorosos do software livre, cuja obstinação beira o fanatismo. Ele é um adepto desse método de trabalho por razões práticas. Software livre simplesmente funciona, segundo o estudante. "Diria que o método open source não afeta minha filosofia de trabalho. Ela é minha filosofia de trabalho por um motivo muito simples: eficiência".

Marteletto estuda e trabalha na PUC-Rio e, atualmente, é o único desenvolvedor brasileiro do OpenBSD, um sistema operacional baseado no BSD Unix. Faz isso desde maio de 2004, e seu foco de trabalho são sistemas de arquivos (filesystem), no entanto, atua também no desenvolvimento geral do kernel.

Sua primeira contribuição "de verdade", como diz, para o kernel OpenBSD aconteceu há cerca de um ano. Foi uma modificação para que fossem geradas estatísticas separadas de I/O (input/output) por disco rígido. Tudo começou quando ajudava um amigo. "Ele tentava portar uma aplicação de monitoramento de disco e me pediu ajuda para fazê-la funcionar corretamente no OpenBSD, pois queria um patch para a aplicação, que a fizesse mostrar as estatísticas de I/O unificadas. Preferi alterar o kernel, uma vez que esse tipo de informação (saber quantas operações de leitura/escrita foram efetuadas em um disco) é interessante e boa de se ter".

Marteletto sente-se recompensado por estar efetuando um trabalho em equipe, em que conhece muita gente, faz novos amigos e discute muitas idéias. "Aprende-se (por mal ou por bem) a trabalhar em grupo, a ouvir críticas, enfim, todas essas coisas de praxe em qualquer atividade coletiva".

Como dica aos interessados, ele recomenda um mergulho de cabeça. "Não tenha medo, muito menos vergonha de estar errado e de ser corrigido. Contribua. Acredito que nós, brasileiros, temos muito mais potencial para contribuir do o que já foi feito até agora. É hora de transformar o potencial em prática, já".

passa pelo menos quatro horas por dia apenas lendo e analisando códigos. Já quando está programando alguma mudança, esse total sobe para seis horas.

No Video4Linux, Chehab sente dificuldade em lidar com pessoas de todo o globo. "É comum trocar informações com pessoas localizadas na Europa e na Ásia, cuja diferença de fuso é significativa, dificultando a comunicação eletrônica", aponta. "No fundo, é mais uma questão de quanto tempo você consegue dedicar. Não existe uma exigência mínima. Como mantenedor, o ideal é que, a cada nova versão de kernel, no ambiente de desenvolvimento, sejam produzidos novos patches pelo mantenedor, incluindo o código estável. Isso acontece a cada uma ou duas semanas. Como regra geral, se gasta pelo menos meia hora por dia para acompanhar o que está acontecendo e um tempo maior para consolidar e preparar os patches", conta Chehab.

Gustavo, do Debian, aponta a comunicação impessoal (por e-mail, chat etc.) e a concretização de objetivos como as principais dificuldades. "Dentro de um ambiente com metas tão diversas, a maior dificuldade é conseguir ter um foco, estabelecer objetivos e alcançá-los. A comunicação impessoal também é empecilho para alguns, mas pequenos grupos reúnem-se de maneira mais freqüente, amenizando esse efeito colateral de um projeto que tem desenvolvedores de todo o mundo".

Gustavo já dedicou e dedica muito tempo ao Debian, mas diz que hoje não precisa sacrificar seu emprego nem todo seu tempo livre.

Piter Punk também sofre com a falta de tempo para o Slackware. Todo seu tempo livre acaba indo para esse "hobby", que inclui outras atividades ligadas ao software livre, como escrever artigos, moderar listas de discussão, responder dúvidas. "Às vezes, alguém reporta um bug que não consigo reproduzir de jeito algum. Por exemplo, bugs em um hardwares que não posso adquirir", conta Piter.

Caso você se interesse por algum projeto, mas está com dificuldades de por a mão na massa por falta de documentação, essa é uma boa área para começar.

2 - Tradução

O processo de internacionalização de um sistema é outra área chave, já que pode aumentar a base de usuários e, conseqüentemente, mais desenvolvedores serão atraídos, acelerando a evolução do sistema.

3 - Design

As interfaces gráficas vêm evoluindo, devido ao crescente aumento de designers e artistas que contribuem. Mas há uma necessidade crescente desse tipo de trabalho. Créditos e reconhecimento são dados da mesma maneira para designers ou programadores.

'Faço o que gosto'



Piter Punk, 27 anos, é game designer e desenvolvedor da distribuição GNU/Linux Slackware

Roberto Freires Batista, ou Piter Punk como é mais conhecido, chegou a substituir o lendário Patrick Volkerding, criador de uma das mais consagradas distribuições GNU/Linux, o Slackware.

Foi durante a época em que Patrick teve problemas de saúde e teve que deixar o projeto, temporariamente, para se tratar. Isso aconteceu no final do ano passado. Piter, então, juntamente com outros membros do GUS-BR (grupo de usuários brasileiros da distribuição Linux Slackware) ficaram responsáveis pela correção de bugs e atualizações de segurança da distribuição.

Apesar de enriquecedora, não foi uma tarefa nada simples. "Sinceramente, espero nunca estar naquela situação novamente, e que o Patrick viva para sempre", brinca Piter.

Ele mora em São Carlos, interior de SP. É também onde trabalha, desenvolvendo games para dispositivos móveis como celulares e handhelds.

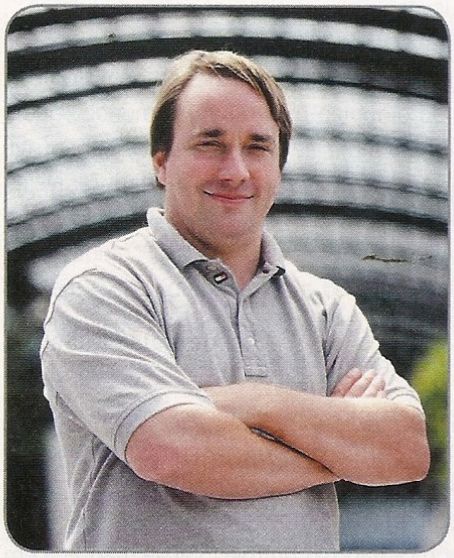
Fora isso, Piter ajuda no desenvolvimento do Slackware e é o criador e mantenedor do slackpkg, um sistema de gerenciamento de pacotes para essa distribuição, que instala ou remove programas e suas dependências. O aplicativo foi escrito todo em bash shell script.

Isso foi há dois anos, mas antes já contribuía com a distribuição. Participa de maneira tão ativa que nem se lembra de seu primeiro patch. "Provavelmente deve ter sido algo que não foi aceito. A primeira coisa que me lembro de ter mandado foram algumas traduções, ou correções de traduções".

Piter se diverte com o projeto. "Bom, faço o que gosto. Isso é legal. Mesmo antes de colaborar, eu passava boa parte do meu tempo instalando e testando softwares e pesquisando coisas estranhas".

Sua dica para quem deseja mergulhar nesse universo: "Não desista. Estude bastante e corra atrás".

Como funciona o desenvolvimento do kernel Linux



O finlandês Linus Torvalds, criador do Linux, é quem dá a última palavra sobre as modificações no kernel

1 - O criador do Linux, Linus Torvalds, com a ajuda de seus colaboradores próximos, define objetivos para a próxima versão. Por exemplo, três anos atrás foi decidido que o kernel (então, na versão 2.4) precisava suportar melhor laptops e se "fortalecer" para rodar bancos de dados gigantescos. Hoje, esses são objetivos consolidados na versão 2.6.

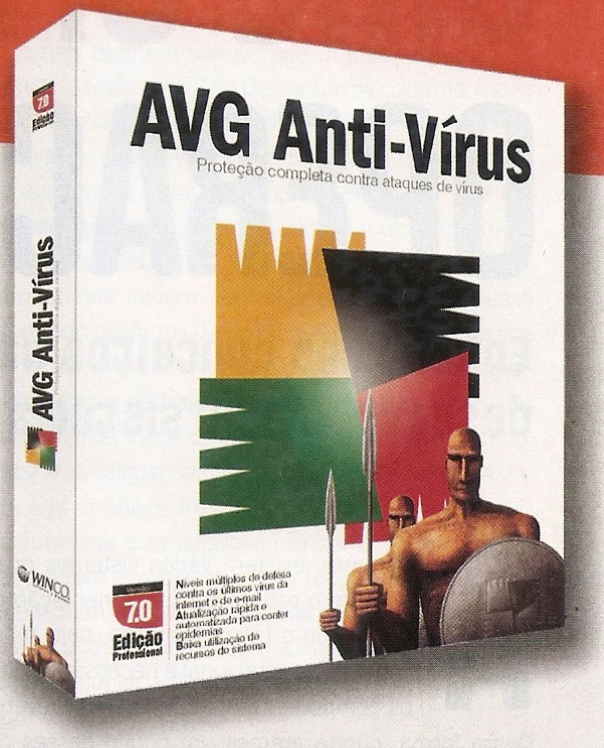
2 - Programadores contratados por grandes empresas de tecnologia (são eles que dão boa parte das contribuições) começam a criar os patches que vão concretizar esses objetivos. Esses patches são enviados aos mantenedores do kernel, cada um responsável por uma área específica, que podem aprovar ou não as atualizações.

3 - Já programadores que produzem patches para o kernel em seu tempo livre fazem a mesma coisa. Foram eles, por exemplo, que criaram a capacidade para gerenciar melhor o uso da energia das baterias dos laptops. Eles também enviam seus patches para os mantenedores.

4 - Os mantenedores, que também trabalham em companhias de TI, analisam e pedem melhorias nesses patches. Quando o resultado é satisfatório, as atualizações são repassadas para Torvalds e seu braço-direito: Andrew Morton.

5 - A dupla de comando, então, examina o código, pede melhorias e, quando satisfeitos, incluem as mudanças em uma versão teste, a qual é lançada para que milhares de entusiastas e colaboradores possam testá-la e enviar relatórios de bugs. Os erros são corrigidos e uma nova versão do kernel é lançada para ser incluída nas futuras distribuições GNU/Linux.

TRANSFORME SEU PC EM UMA FORTALEZA.



Promoção AVG
VALE 10%
CUPOM DESCONTO

Finalmente a oportunidade que você esperava para ter o melhor antivírus do mercado pelo menor preço! Comprando o AVG antivírus pelo site www.ciadosoftware.com.br/avg você terá 10% de desconto.

Cia. do Software
Distribuindo Soluções

AVG Antivírus

www.ciadosoftware.com.br/avg

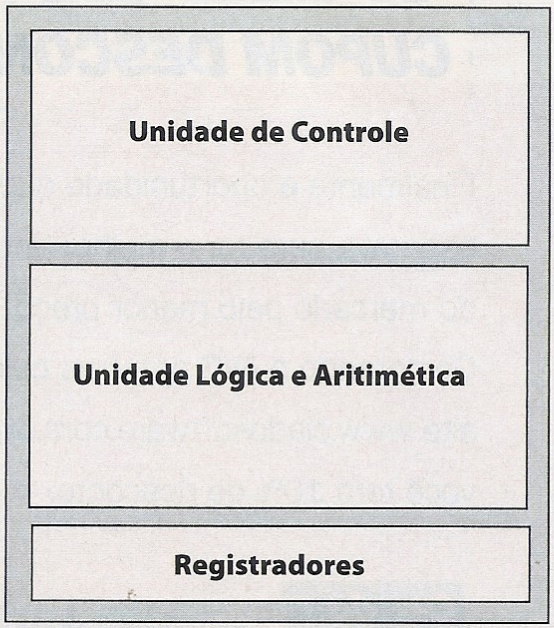
CRIE UM SISTEMA OPERACIONAL

Entenda os conceitos básicos de interação com a máquina e desenvolva seu sistema em C e Assembly

Por Alexnaldo C. Santos

Atualmente, existem vários sistemas operacionais que permitem o estudo e a implementação de seu código-fonte, como o Linux. Contudo, para modificar ou criar um sistema operacional é necessário entender como ele e os componentes do computador devem se relacionar. Dessa forma, conheceremos um pouco desses componentes e seus relacionamentos.

CPU (Unidade central de processamento) – É o "cérebro" do computador, responsável pela execução dos programas carregados na memória. A execução de um programa é feita passo-a-passo, comando por comando, apesar de não percebermos na prática.

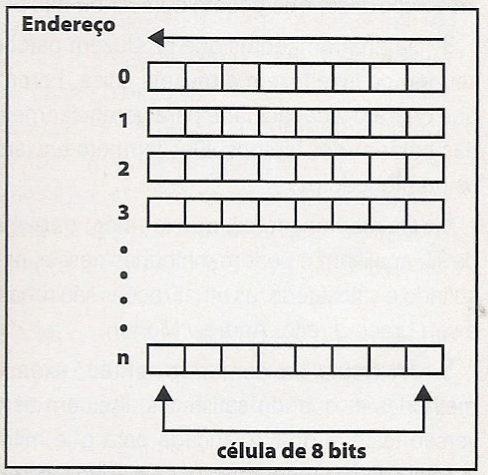


Organização da CPU (Unidade central de processamento)

É possível notar que para termos "multiprocessamento real" é necessário mais de uma CPU na mesma máquina. Ou seja, os computadores pessoais atuais fazem um "multiprocessamento irreal" por meio do compartilhamento de tempo, em que para cada processo na memória é dado o controle total da máquina em intervalos de tempos indeterminados. Esse processo é chamado de "escalonamento" e é vital em bons sistemas operacionais.

A CPU é composta por várias partes distintas. A unidade de controle é responsável pela busca de instruções e determinação de seus tipos. A unidade lógica e aritmética, pelas operações matemáticas básicas (adição, subtração, soma e divisão) e pela aritmética binária, necessária à execução de instruções. Existe ainda uma memória pequena de alta velocidade, usada para resultados temporários e certas informações de controle, a qual é composta por um conjunto de registradores, cada um com uma função bem definida.

Memória – É o local onde os programas e dados são armazenados. Não há diferença para a CPU no processo de execução, ou seja, se no lugar de um código de execução estiver um dado inválido, a CPU tentará executá-lo de qualquer forma (provavelmente a máquina irá tra-

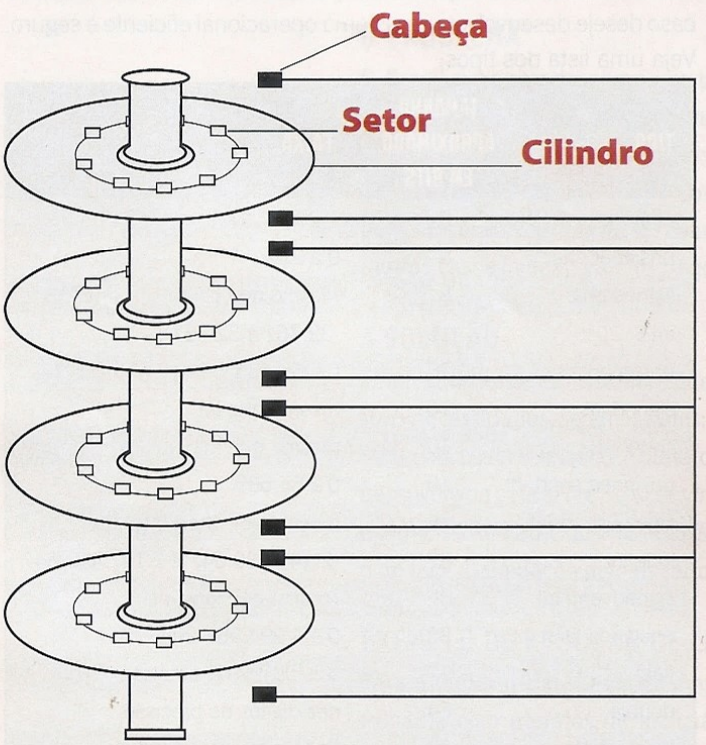


Computadores atuais têm arranjo de memória de 8 bits, alinhados da direita para esquerda

var). É responsabilidade do sistema operacional controlar onde devem ficar "dados" e "programas". A unidade de armazenamento da memória é o bit (binário), que pode ser 0 ou 1. A memória é composta por um conjunto de células, todas com a mesma quantidade de bits por célula, cada qual armazenando uma parte da informação. Para acessar uma célula de memória é necessário saber seu número, ou seja, seu endereço.

A célula é a menor unidade endereçável, ou seja, não podemos acessar diretamente um bit dentro dela, apenas o todo. A partir da padronização do tamanho da célula em todos os computadores atuais, denominou-se byte o conjunto de oito bits. Os bytes são agrupados em palavras, ou seja, um computador de 16 bits (duas palavras) manipulará dois bytes de uma única vez, enquanto os de 32 bits manipularão quatro bytes por vez, conseqüentemente mais rápido. Computadores pessoais são de 32 bits, porém já existem CPUs funcionando a 64 bits. Um sistema operacional é desenvolvido levando-se em conta essas informações para um melhor gerenciamento dos dados na memória.

Discos Magnéticos – Infelizmente, a memória é volátil e seus dados são perdidos quando o computador é desligado. Como uma forma de armazenamento permanente existe o disco rígido, um pedaço de metal formado por círculos concêntricos chamados trilhas, normalmente com dois lados. O disco possui uma cabeça que se movimenta em relação ao centro. A distância entre o centro e a posição acessada é chamada de cilindro. Uma unidade de disco com n pratos (2n lados) terá 2n cabeças e, portanto, 2n trilhas por cilindro.



A estrutura interna de um disco rígido

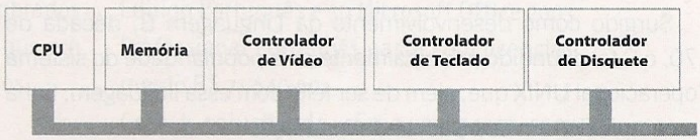
As trilhas são divididas em setores, os quais contêm, normalmente, 512 bytes cada. O processo de leitura e gravação dos dados consiste em fornecer número do cilindro e cabeça – que juntos formam uma trilha –, número do setor, número de palavras, endereço da memória e tipo da operação (leitura ou gravação). Sistemas operacionais definem sua própria unidade de armazenamento (sempre múltiplo de 512). Dessa forma, quanto maior for esse valor, maior será a velocidade de leitura e menor será a velocidade de gravação devido à latência rotacional (processo de posicionamento da cabeça para leitura/gravação). Note que em discos de alta capacidade essa perda só será percebida perto do preenchimento total da unidade.

Sistemas operacionais devem controlar eficientemente essa latência, posicionando os dados de um arquivo tão perto uns dos outros quanto possível. A distância entre dados comuns armazenados é o que chamamos de fragmentação.

Para determinar a capacidade de armazenamento de um disco, deve-se fazer o seguinte cálculo: trilhas * setores * 512 * cabeças.

Agora que você já conhece três itens básicos veremos, rapidamente, como funciona o processo de Entrada/Saída (E/S) de dados. O primeiro passo é saber que nem toda Entrada é fornecida pelo usuário e nem toda Saída é destinada ao usuário, isto é, alguns processos podem ocorrer sem a interferência humana. Uma câmera digital acoplada ao computador tem uma Entrada automatizada, por exemplo.

Os componentes do computador são interligados por meio de canais, chamados de barramentos. Dessa forma, quando a CPU quer executar E/S, envia um programa especial ao canal para executá-lo, e libera a CPU para uso. Ao término, o canal envia um sinal chamado interrupção, obrigando a CPU a respondê-lo. Quanto maior for o número de barramentos especializados, maior será o potencial de processamento. Por exemplo: barramento para memória, barramento para E/S etc. Computadores pessoais têm apenas um barramento conectando todos os componentes.



Exemplo de uma estrutura de barramento simples

CARREGAMENTO DO SISTEMA OPERACIONAL

O processo de carregamento do sistema operacional é simples e direto. Ao iniciar a máquina, um programa gravado em uma memória especial localizada na placa-mãe, denominado BIOS (Basic Input Output System), verifica em que local deve procurar o sistema operacional (disquete, disco rígido etc.), então, lê o setor 0, trilha 0 e cabeça 0, ou seja, somente 512 bytes no início. Esses dados são colocados no endereço absoluto de memória 0000:7C00h. Em seguida, é feita uma verificação na posição 0000:7DFEh, que deve conter a informação AA55h, a qual indica uma assinatura de boot, caso contrário, uma mensagem de erro será exibida (por

exemplo, "Sem sistema operacional"). Se tudo estiver correto, a BIOS passa o controle da máquina para posição de memória 0000:7C00h iniciando, assim, o sistema operacional.

Por causa dessa limitação da BIOS, todo sistema operacional é carregado em duas ou mais fases. Um pequeno programa é feito, em Assembly (devido ao tamanho necessário), para ficar no local indicado, e fará a leitura e a execução da outra parte, normalmente feita em C.

É importante saber que a linguagem de programação Assembly é fundamental no processo de desenvolvimento de um sistema operacional, apesar de ser complexa e pouco legível. Por meio dela, programamos praticamente em linguagem de máquina, o que nos dá controle total da estrutura final do programa que será gerado. O segundo passo é saber por que, normalmente, somente essa parte inicial (ou rotinas que requerem extremo controle e performance do código) é feita em Assembly e as demais, em linguagem C. Isso se dá pelo fato de que programar diretamente em linguagem de máquina torna a leitura e manutenção do código mais lento e complexo. Veja, a seguir, uma comparação de códigos:

```
mov     eax,10
add     eax,5
```

Código em Assembly para somar 10 e 5 e colocar no registrador eax (entenda aqui como uma variável qualquer)

```
int eax = 10+5;
```

O mesmo código escrito em linguagem C

Dessa forma, entende-se por que, normalmente, usamos linguagem C no desenvolvimento de um sistema operacional e Assembly em casos de extrema necessidade. Agora, conheceremos um pouco dessa linguagem e sua história.

INTRODUÇÃO À LINGUAGEM C NO DESENVOLVIMENTO DE UM SISTEMA OPERACIONAL

Surgido como desenvolvimento da Linguagem B, década de 70, o C foi difundido principalmente pela popularidade do sistema operacional UNIX que, além de ser feito com essa linguagem, tinha o compilador em seu pacote de instalação.

Na mesma época, foi criada uma linguagem chamada Pascal que, apesar de mais estruturada, fácil de aprender e de uma elegância incontestável, não emplacou. O sucesso da linguagem C ocorreu devido ao fato de que os programadores de sistemas operacionais precisavam de uma linguagem extremamente flexível para portar facilmente o sistema voltado a vários tipos de máquina (a estrutura da linguagem foi construída de acordo as necessidades). O compilador Pascal era pago, já o C era distribuído gratuitamente. Esse fato foi crucial para que estudantes e profissionais da época passassem a estudar e usar a linguagem C, em vez da linguagem Pascal.

Agora que já sabemos um pouco da história da linguagem C, aprenderemos alguns cuidados ao utilizá-la no desenvolvimento de um sistema operacional.

COMPILADOR

Por padrão, os compiladores C colocam um cabeçalho no início de seu programa para que o sistema operacional tenha informações vitais de como executá-lo.

Porém, lembre-se de especificar para o compilador retirar esses dados, para que seu programa seja gerado em modo "flat", isto é, sem cabeçalho ou qualquer outra informação específica para o sistema operacional atual rodando o compilador. Uma dessas informações é o nome da primeira rotina a ser executada, normalmente "main".

Controle – É você quem tem o controle da máquina. Dessa forma, a maioria das rotinas disponíveis pela linguagem não funcionará. Veja um exemplo:

```
#include <stdio.h>
  void main(void)
  {
        printf(" olá mundo !");
  }
```

Aqui haverá um problema porque a rotina "printf" é específica para o sistema operacional em que o compilador está instalado e não funcionará se for executada diretamente em nível de máquina. Você terá de colocar, manualmente, os caracteres na posição correta da memória que está relacionada ao vídeo

Tipos de Dados – Você deve saber manipular corretamente todos os tipos de dados da linguagem. Eles são manipulados por meio de variáveis. Cada uma suporta um tipo de dado distinto (ou similar) e caso deseje desenvolver um sistema operacional eficiente e seguro. Veja uma lista dos tipos:

TIPO	TAMANHO APROXIMADO EM BITS	FAIXA
char	8	-127 a 127
unsigner char	8	0 a 255
signed char	8	mesmo que char
int	16	-32.767 a 32.767
unsigned int	16	0 a 65.535
signed int	16	mesmo que int
short int	16	mesmo que int
unsigned short int	16	0 a 65.535
signed short int	16	mesmo que short int
long int	32	-2.147.483.647 a 2.147.483.647
signed long int	32	mesmo que long int
unsigned long int	32	0 a 4.294.967.295
float	32	seis dígitos de precisão
double	64	dez dígitos de precisão
long double	80	dez dígitos de precisão

PROJETO EMPRESA LEGAL

CUSTOS MAIS BAIXOS EM TI

legalidade nos softwares

Tenha um sistema operacional para a sua empresa,
personalizado com os aplicativos necessários
para o dia-a-dia. Reduza custos, e fique legal.

Qualquer empresa preocupada em manter-se dentro da lei deve reservar especial atenção aos softwares nos seus computadores. Um sistema simples idealizado pela 4Free Soluções Inteligentes, com apoio da Digerati, oferece uma solução para legalização de softwares, aliando transição rápida e eficaz a mínimo investimento.

O PROBLEMA

O Brasil é apontado como um dos países com maior índice de pirataria de software no mundo. Um dos grandes impulsionadores dessa situação são os altos preços cobrados pelos desenvolvedores de software, que em muitos casos inviabilizam os negócios.

A SOLUÇÃO

Empresários querem sistemas que atendam suas necessidades, sejam baratos, estáveis e ofereçam suporte técnico. A plataforma Linux desenvolvida pela 4Free Soluções Inteligentes é muito semelhante ao Windows ® e por isso é de fácil assimilação pelo usuário

EXEMPLO DE REDUÇÃO DE CUSTOS

Para atender o Geranegócio, empresa que trabalha com projetos de inclusão digital, a 4Free desenvolveu o Ger@Linux, um sistema operacional robusto, com aplicativos de escritório. "Com o sistema desenvolvido o Geranegócio® reduziu em 70% os gastos que teria com licenciamento", afirma Aquiles Siqueira, da empresa.

PODE SER MUITO SIMPLES FAZER UMA ECONOMIA DE 70%

DESCUBRA COMO FAZER

Uma companhia com 30 estações rodando sistema operacional Windows XP Home Edition Português e 30 Microsoft Office 2003 Professional Português gasta com licenciamento R$ 71.640,00.

Com a solução da 4Free, os gasto com licenciamento é zero. A empresa fará investimentos somente na adaptação ao sistema operacional linux, aplicativos e treinamento especializado. Entre em contato e descubra como dar esse passo.

4free Soluções Inteligentes

www.4free.com.br

atendimento@4free.com.br - (11) 6283-1012

Debug – Não existe mais um "debug" para facilitar sua vida. É fundamental ter cuidado ao programar cada linha, pois o resultado não será uma mensagem de erro explicando detalhes do problema, mas o travamento completo da máquina.

Memória – Será necessário aprender a acessar diretamente dados na memória. Então, devemos entender um pouco do conceito de "ponteiros".

Ponteiros – É o endereço na memória de uma variável. Será utilizado para acessar, rapidamente, itens dentro de uma matriz, alterar (dentro de uma rotina) os parâmetros passados a ela, e trabalhar com estruturas dinâmicas de dados, por exemplo, listas encadeadas. Os operadores de ponteiro são:

&	Operador unário que devolve o endereço na memória de seu operando.
*	Operador unário que devolve o valor da variável localizada no endereço que o segue.

Exemplo:

```
void main(void)
 {
        int *a; // declara 'a' como um
ponteiro
        int b=10;

        a = &b;       // 'a' agora tem o
endereço de 'b', não o valor.
        *a = 15;      // nesse caso é o
mesmo que 'b=15;'
 }
```

Obs: no código real, os comentários não devem pular de linha

Operação binária – Em muito casos, será necessário efetuar operações binárias. Já aprendemos que a menor unidade de dados é o bit e que todas as informações são representadas na memória bit a bit (veja em "memória"). Diante dessas informações, precisamos testar, atribuir ou deslocar bits com facilidade. Pensando nisso, a linguagem C tem um conjunto de operadores bit a bit. Vejamos a seguir:

OPERADOR	AÇÃO
&	AND
!	OR
^	OR exclusivo (XOR)
~	Complemento de um
>>	Deslocamento à esquerda
<<	Deslocamento à direita

Exemplo: (x = 1 & 2) = 0

```
0 0 0 0 0 0 1   - número '1' em binário
0 0 0 0 0 1 0   - número '2' em binário
X = 0 0 0 0 0 0 0 0
```

O resultado será 0. O processo é o mesmo utilizado para comparar valores booleanos (verdadeiro e falso)

Conversão de Tipos – Aprenda a fazer conversão de tipos de dados com segurança. Normalmente, você irá se deparar com uma variável de um tipo, precisando colocar seu valor em outra variável de outro tipo. Tome cuidado sempre que fizer isso com uma variável de origem maior que a variável de destino, pois dados serão perdidos.

```
void main(void)
 {
        int a=512;
        char b;

        b = a;  // 'char' tem limite de 256.
O resultado será a perda da metade do valor.
 }
```

Velocidade – Lembre-se sempre de que está fazendo um sistema operacional. Com isso, é aconselhável fazer a rotina em Assembly do que tornar o código C lento e complexo de se ler. Apesar da flexibilidade da linguagem C, você não tem o controle total do código como tem em Assembly.

Variáveis – Sempre que possível, inicialize as variáveis com tipos padrões a fim de evitar erros condicionais. Alguns compiladores C obrigam a declaração de variáveis com valores ou eles mesmos colocam esses valores. Veja o que pode acontecer:

```
void main(void)
 {
        int a;
        int b;
        a=b; // código simples, mas você
sabe o conteúdo da variável 'a' ?
 }
```

PRÁTICA

Vimos que para desenvolver um sistema operacional é necessário conhecer seus componentes e suas interações. Entretanto, há pela frente mais complexidades e desafios. As necessidades de conhecimento surgirão durante o aprendizado e a construção do sistema operacional. Não reinvente a roda, veja como outros sistemas foram feitos, analise seus códigos-fonte, e adéqüe às suas necessidades. Então, vamos à prática.

BRASO - SISTEMA OPERACIONAL BRASILEIRO

No final do ano de 2000, resolvi projetar um pequeno sistema operacional como estudo e incentivar novas pesquisas na área. Em conjunto com um amigo, Marcos Aurélio Almeida da Silva, conseguimos fazer um pequeno protótipo, divido em dois módulos: um para a inicialização da máquina e outro para carregar o sistema e configurar o ambiente.

Antes de iniciarmos, copie o código completo do BraSO no CD (seção Sistemas Operacionais) ou no site *www.geocites.com/kundera_braso*. Baixe também as ferramentas necessárias ao desenvolvimento.

Inicialização da máquina – Como vimos anteriormente, temos de fazer um programa com exatos 512 bytes (veja detalhes em "Carregamento do Sistema Operacional"), nossa única opção é fazê-lo diretamente em Assembly, então, vamos ao código comentado.

Primeiro, o programa de boot:

```
; Arquivo < boot.asm >
; Versao  < 0.1 >
; Modificacoes :
;       31/12/2000:10:30, 0.0.1 - versao inicial
;02/01/2001:19:17, 0.0.2    - vai para o modo 80x25 e tira o cursor
;
; ************************************
; Este programa eh responsavel pelo carregamento do sistema
; operacional BraSO. O mesmo ficara gravado no setor 1, trilha 0
; cabeca 0 ; local onde a BIOS ira fazer a busca e carrega-lo
; para o endereco absoluto 0000:7C00h.
; ************************************
; a GRANDE parte deste programa foi retirado do EduOS, ficando aqui registrado
; o agradecimento a Cornelis Frank <Frank.Cornelis@rug.ac.be>


LOADED                  EQU     7C00h
END_BIOS_DATA           EQU     0500h
BUFFER_SEGMENT          EQU     07E0h
; erros possiveis
No386                   EQU     '3'
ErrorLoadingRootDir     EQU     'R'
OSFILENotFound          EQU     '0'
ErrorLoadingFATs        EQU     'F'
ErrorLoadingCluster     EQU     'C'
InternalFATError        EQU     'I'

bits    16              ; modo real 16 bits

org     7C00h   ; estamos em 0000h:7C00h
        ; Cebecalho para compatibilidade com MS-DOS FAT12
        ; para mais informacoes, consulte o arquivo <../dos/boot.html>
    jmp short   Inicio
    nop
    db      'BraSO   ' ; 8-bytes para o nome
    dw      512  ; tamanho do setor em bytes
    db      1           ; setores por cluster
    dw      1           ; clusters reservados
    db      2               ; numero de FATs
    dw      224             ; entrada do diretorio principal
    dw      2880          ; total de setores
    db      0F0h            ; formato
    dw      9              ; setores por FAT
    dw      18           ; setores por trilha
    dw      2               ; lados
    dd      0              ; setores ocultos
    dd      0               ; mais setores
    db      0               ; Id do driver
    db      0               ; reservado
    db      4           ; assinatura de boot
    db      435101793       ; numero do volume serial
    db      'NO NAME    '       ; nome do label (11 bytes)
    db      'FAT12   '      ; sistema de arquivo (8 bytes)

    Inicio:
     mov    ax, 0x0003
     int    0x10          ; Limpa a tela e assegura que estamos
                          ; no modo 80x25

     mov    ah, 0x01
     mov    cx, 0x2000
     int    0x10        ; Desabilita o cursor

            xor     ax,ax           ; seta pilha para 0000h:7C00h
            cli     ; nada de interrupcoes
            mov     ss,ax           ; mudamos o endereco da pilha para evitar
                                    ; que percamos dados no endereco 0000:0500h
            mov     sp,LOADED
            sti                     ; pilha
```

```
agora tem 30646 bytes disponiveis

    push    cs                  ; registrador de
dados para o local correto
    pop     ds

    mov     bp,dx               ; armazena o numero
do driver(i.e.0=disquete,80=Hd) em BP
    ; O numero do driver sera passado para o
                                ; OSFILE

    ; verifica se o processador eh 386+
    mov     ah,01110000b ; Tentando setar o
IOPL & NT flags.
    push    ax                  ; (IOPL no bits
12-13 & NT no bit 14)
    popf                        ; Se nao estao
setados, nao eh um 386
     pushf
     pop     ax
     and     ah,01110000b       ; Verifica se
IOPL & NT ainda estao setados
     jz      ShowError286       ; Nao pode
pular sobre mais de 128 bytes usando
     ; um jump no 286. Entao, dois jumps

     push    cs ; modifica o segmento Extra
     pop     es

     ; carrega o diretorio principal - INT
13h/02h - READ SECTORS
     mov     byte [ErrorCode],ErrorLoadingRo
otDir ; Codigo de Erro
     mov     ax,020Eh                   ;
0Eh = 224 entradas * 32 bytes por entrada /
512
              mov       bx,END_BIOS_DATA
; ES:BX local do buffer
    m o v                 c x , 0 0 0 2 h
    ; Trilha 0, Setor 2
                  m o v           d h , 0 1 h
          ; Lado 1, DL aindo contem o numero
do driver
                    i n t             1 3 h
          ; 0Eh+02h=16, entao nao teremos
"track boundary"(limite de trilha)
         jc    ShowError286                ;
Pula para a rotina se algum erro ocorreu

    ; procura pelo OSFILE no diretorio principal
    mov     di,END_BIOS_DATA
    mov     dx,224                  ;
Numero de entradas no diretorio principal
    mov     byte [ErrorCode],OSFILENotFound
; Codigo de Erro
    Search:
    mov     cx,11               ; 8 + 3 nome do
arquivo + extensao
    mov     si,FileName         ; DS:SI ponteiro
para o OSFILE
     rep    cmpsb               ; verifica se o
arquivo existe
     jz     Found       ; achou ? pula codigo
     dec    dx                  ; Nao ? ainda tem
entradas validas ?
    jz      ShowError286        ; Nao ? Entao a
ultima entrada foi checada
    add     di,21               ; Sim ? verifica
na proxima entrada
    add     di,cx               ; adiciona o resto
dos bytes que nao condizem
    jmp     Search              ; procura novamente

    ShowError286:               ; Nos temos
que colocar um segundo jump aqui porque o
     JMP    ShowError    ; 286 nao pode dar
um pulo sobre 128 bytes

    Found:          ; Ok! arquivo encontrado
                    ; DI eh o ponteiro
para os atributos do arquivo

     mov    eax,[di+17]         ; Pega o ta-
manho do arquivo em bytes (dword)
    mov     [KrnSize], eax  ; Salva o tamanho
em bytes para passar ao OSFILE
    add     eax,512-1           ; Calcula o numero
de setores usados
    shr     eax,9               ; EAX <- EAX / 512
    push    ax                  ; Salva o numero
de setores a serem lidos
    mov     ax,[di+15]          ; Primeiro Cluster
do OSFILE
     push   ax   ; Salva o primeiro cluster

     ; Carrega a primeira FAT na memoria -
usando a INT 13h/02h
     mov    ax,0209h            ; ler 9 setores
     mov    bx,END_BIOS_DATA; ES:BX local do
buffer
     mov    cx,0002h    ; Trilha 0, Setor 2
     mov    dx,bp       ; Numero do driver
```

```
        xor     dh,dh           ; tamanho 0
        int     13h
        jnc     FATloaded   ; Se CF, tentar ler
a segunda FAT

        mov     byte [ErrorCode],ErrorLoadingFATs
; Codigo de Erro
        ; Carreaga a segunda FAT na memoria -
usando INT 13h/02h
        ; Devido a um "track boundary" vai ocorrer
na segunda FAT
        ; nos temos que usar duas leituras.
        mov     ax,0208h    ; Ler os 8 primeiros
setores
        mov     cl,0Bh          ; Trilha ainda
eh 0, Setor 0Bh = 2 + 9
        int     13h             ; DL & DH ainda
estao corretos
        jc      ShowError286    ; Nao pode ser a
segunda FAT ?Erro...
        mov     ax,0201h        ; Ultimo setor a
ser lido
        mov     bx,END_BIOS_DATA + 512*8 ; 512 *
8 = primeiros 8 setores
        mov     cl,01h          ; Trilha ainda
eh 0, Setor 1
        mov     dh,01h          ; Lado 1, DL ainda
esta correto
        int     13h
        jc      ShowError       ; Nao pode ser a
segunda FAT ?Erro...

    FATloaded:
        ; Carrega o OSFILE usando FAT12
        pop     ax                      ; AX =
Primeiro Cluster
        pop     cx                      ; CX =
Numero de clusters a serem lidos
        mov     bx,BUFFER_SEGMENT       ; ES pon-
teiro para o final do setor de boot ((07E0h)
        mov     es,bx
    NextCluster:
        mov     byte [ErrorCode], ErrorLoading-
Cluster
            ; AX = Corrente cluster a ser lido
                    ; ES = Segmento do buffer
                        ; CX = Numero de
clusters a serem lidos
        pusha                   ; Talvez nao
altere AX,ES,CX (so 1 byte)
        add     ax,31           ; Converte o
Cluster para setores logicos
        ; 31 = SetoresReservados + SetoresOcultos
            ; + FATS * SetoresPorFAT + Dir - 2
        xor     dx,dx           ; Converte setores
logicos para CHS
        mov     bx,18 ; CL := ((AX-1) MOD 18)+1
        div     bx          ; CH := (AX-1) DIV 36
        mov     ch,al               ; DH := ((AX-1)
DIV 18) AND 1
        inc     dx
        mov     cl,dl
        and     AL,1                ; DH = Lado
        mov     dx,bp ; Carrega numero do drive
        mov     dh,al               ; CH = Trilha
        shr     ch,1                ; CL = Setor

        mov     ax,0201h        ; Ler 1 setor
usando INT 13h/02h
        xor     bx,bx  ; ES:0000h local do buffer
        int     13h
        jc      ShowError   ; Falha na leitura
do cluster
        popa                Restaura AX, ES, CX

        ; Pega o proximo Cluster usando FAT12
        mov     byte [ErrorCode], InternalFATError
        mov     bx,ax
        shl     bx,1
        add     bx,ax
        shr     bx,1        ;  BX = AX * 3 / 2;
        and     ax,1
        mov     ax,[END_BIOS_DATA + BX]
        jz      EvenCluster ; um MOV nao altera
o flag Z
        shr     ax,4
    EvenCluster:
        and     ax,0000111111111111b    ; Entrada
da FAT so leva 12 bits
        cmp     ax,0000h        ; Se Cluster livre
        je      ShowError
        cmp     ax,0FF7h        ; Se Cluster ruim
        je      ShowError

        mov     bx,es       ; Incrementa ES para
o proximo
        add     bx,512/16       ; bloco de memoria
que sera preenchido
        mov     es,bx
    loop    NextCluster         ; Ate CX=0,
ultimo setor lido
```

```
    mov     dx,bp                   ; Passa o
numero do drive par o OSFILE
    ; O numero do drive sera guardado em bp

    mov     ax,BUFFER_SEGMENT       ; Segmento
    mov     ds,ax
    mov     es,ax
    mov     fs,ax       ; so para completar...
nem precisava
    mov     gs,ax                   ; idem..
    mov     eax,[KrnSize]           ; eax sera
passado para o OSFILE com o seu
    ; tamanho em bytes
            push    eax
    ; os seguinte parametros estarao dispo-
niveis ao OSFILE
    ; DL  = numero do drive
    ; EAX = tamanho em bytes do OSFILE
    jmp     BUFFER_SEGMENT:0000h        ;Bye..
bye...boot

    ; procedure locais
    ShowError:
    mov     cx,MessageSize
    mov     si,Message
    .ShowNextChar
    lodsb
    mov     ah,0Eh              ; apresenta ca-
racter usando INT 10h/0Eh
    mov     bx,0007h ; Pagina 0, Cor normal
    int     10h
    loop    .ShowNextChar       ; ate apre-
sentar toda a mensagem

    xor     ah,ah               ; espera pres-
sionar uma tecla
    int     16h

    xor     dx,dx           ; primeiro disquete
    int     19h         ; forca BIOS a executar
todo novamente

    ; dados locais
    Message   db    13,"Sem o BraSO ou falha
no disquete. Codigo '"
    ErrorCode       db      No386,"'.",10,13
; No386 eh o primeiro erro
    db      'Troque o disquete e pressione
ENTER...',10,10,13
    MessageSize   equ     $ - Message

    FileName        db      'BRASO   SYS' ;
deve ser 11 bytes
    KrnSize         dd      0
    TIMES   510-($-$$) DB 0     ; posiciona-
mento da assinatura
    DW      0xAA55      ; assinatura de boot

    ; ---- NOTE ----
    ; INT 13h usa a DMA para transferir os
dados do disquete
    ; para a memoria.
    ; A DMA tem duas restricoes:
    ; -Nao pode transferir mais de 64K de uma vez
    ;       - Nao pode ocorrer "page bounda-
ry"( limite de pagina )
    ; Pagina x eh definida como :
    ; Pagina x = x000:0000 - x000:FFFF
; Nos usandos o buffer no endereco 0000:7E00
    ; Podemos ter problemas com "page boundary"?
    ; O problema pode ocorrer se o buffer
para o cluster corrente
    ; tentar ultrapassar 0000:FFFF. Mas como
a nossa transferencia comeca em 0000:7E00  e
termina em 0000:FFFF,
    ; entao nos nao teremos problema de "page
boundary".
```

Compilando o boot.asm – O compilador Assembly utilizado é o Nasm, atualmente, o mais fácil de aprender e usar, além de suas extensas funcionalidades. No site do BraSO, há um link direto para download.

```
C:\braso\boot\nasm -f bin boot.asm -o boot.img
```

Esse processo criará o arquivo 'boot.img' que será a imagem do BraSO para inicialização

Gerando o disquete para boot – Por motivos de segurança, o boot é gerado somente para disquetes, a fim de evitar que você perca seu disco rígido acidentalmente. O arquivo com a imagem do BraSO será copiado para o disquete no setor 1, trilha 0 e cabeça 0. Portanto temos de usar um programa especial chamado 'partcopy', que nos permite manipular com facilidade setores, trilhas e cabeças, aleatoriamente.

```
C:\braso\boot\partcopy boot\boot.img 0
200 -f0 0
```

Esse comando copia a imagem para o disquete

Agora que já temos o disquete preparado para a inicialização, vejamos o trecho inicial do código-fonte do núcleo do BraSO, também feito em Assembly.

```
; Arquivo < BraSO.asm >
```

```
    ; Versao  < 0.1 >
    ; Modificacoes :
    ; 31/12/2000:14:48, 0.0.1 - versao inicial
    ; 02/01/2001:19:17, 0.0.2  - adicionado
o codigo para reprogramar o
    ;                    PIC e o IDT
    ;       < descricao das modificacoes >
    ; ************************************
    ; Este modulo eh responsavel pela prepa-
racao de embiente do BraSO :
    ;
    ; 0. Habilitar A20
    ; 1. Modo Protegido
    ; 2. Inicializar a PIC
    ; 3. Configurar a IDT
    ; 4. Configurar a PIT
    ; 5. Configurar a TSS
    ; 6. Carregar e passar o controle para
o Init.sys
    ; ************************************
    bits    16  ; ainda estamos em modo real
    org     0h

    jmp     inicio          ; pula cabecalho
    db 'SYS'        ; veja explicacao em "bra-
so/kernel/video/video.asm"

    %macro  Print 1
            mov     si,%1
            call    PrintMessage
    %endmacro

    ; rotina em 32bits

    %macro  Print32 2
     mov    ecx,%2
            mov     esi,(%1+Loaded)
    call    (PrintMessage32+Loaded)
    %endmacro

    ; includes
    %include "gdtnasm.inc"

    Loaded          equ     7E00h  ; caso o Bra-
SO.sys seja carregado em outro local
    ; modifique aqui tambem
    ; definicoes da GDT
    ; inicialmente o kernel tera dois sele-
tores FLAT
    NULL_SELETOR equ 00h ; nunca pode ser usado!
    FLAT_CODE       equ     08h
    FLAT_DATA       equ     10h

    inicio:
      push   cs  ; coloca o valor certo em ds
      pop    ds
      mov [BootDrive], dl
            ; apresenta mensagem de inicio
            Print   InitMessage
      ; re-inicia a controladora de disco
      xor    ax,ax   ; funcao 00h, dl ainda eh
o numero do drive
      int    0x13                ; BIOS int
      ; Prepara entrada para modo protegido
      cli    ;{0}, o kernel retornara as ints
```

Confira o código completo no CD-ROM dessa edição, na categoria Sistemas operacionais.

COMPILANDO O NÚCLEO BRASO.ASM

Compilamos o núcleo seguindo o mesmo modelo visto anteriormente.

```
C:\braso\kernel\nasm -f bin braso.asm -o
braso.sys
```

Faremos uma simples cópia de arquivo para disquete: C:\braso\kernel\copy braso.sys a:

Testando o BraSO – Para testarmos, é necessário que o boot esteja configurado para pesquisar o sistema operacional primeiro no drive de disquete. Reinicie a máquina e veja um sistema simples sendo carregado, exibindo uma mensagem e parando em seguida. Os próximos passos só dependem de você. Boa sorte!

REFERÊNCIAS

Andrew S. Tanenbaum e Albert S. Woodhull. Sistemas Operacionais: projeto e implementação. Bookman, 2000. 2ª ed.

Abraham Silberschatz, Peter Baer Galvin, Greg Gagne. Operating System Concepts. John Wiley & Sons, 7th ed.

Morgan Kaufmann. Computer Organization and Design: The Hardware. Software Interface, 3rd ed.

Andre S. Tanenbaum. Organização Estrutura de Computadores. Livros Técnicos e Científicos Editora, 3ª ed.

Herbert Schildt. C Completo e Total. Makron Books, 3ª ed.

BraSO – Sistema Operacional Brasileiro: www.geocities.com/kundera_braso

COMO MONTAR SEU PRÓPRIO LINUX

Por Angelo Valle *

Além de mais velocidade e estabilidade, fazer sua própria distro é um processo muito enriquecedor para quem quer entender sistemas operacionais e sua relação com a máquina

Quais são as vantagens de criar seu próprio sistema? São muitas. Velocidade, estabilidade, liberdade e o mais importante: conhecimento. Aqui, aprenderemos algumas maneiras interessantes para preparar seu sistema hospedeiro e como usar o livro Linux From Scratch (disponível em inglês em www.linuxfromscratch.org ou em português: http://lfs-br.codigolivre.org) para construir seu sistema-base.

Linux From Scratch significa Linux do zero, do rascunho, do nada. Essa é a idéia do livro, que explica o passo-a-passo da criação de um sistema Linux, o qual depende apenas de um sistema hospedeiro e dos referidos códigos-fonte para a compilação de seus pacotes.

Quais as dificuldades? Para um usuário pouco experiente serão muitas. É comum que na primeira fase do projeto esbarremos em problemas de dependências, pois faltam bibliotecas de desenvolvimento e até mesmo programas. Nada que não se resolva rapidamente. Nesses casos, o ideal é entrar em contato com a comunidade LFS-BR através do site http://lfs-br.codigolivre.org.br/ e informar detalhadamente a mensagem de erro recebida. Também não deixe de se cadastrar na lista de discussão no site do projeto LFS-BR.

Construir um LFS lhe dará uma noção maior de como as coisas acontecem. É uma experiência válida até para os mais veteranos. Trata-se de um livro muito interessante e que certamente mudará sua forma de ver os sistemas Linux.

O livro LFS assume que você já tenha um Linux instalado e que já possua conhecimento necessário para a manipulação do sistema. Por isso, são necessários alguns pré-requisitos:

1 - Experiência em compilação e instalação de programas a partir do código-fonte

2 - Configuração e compilação do kernel

3 - Conhecimento básico de sintaxe de shell

4 - Conhecimento sobre alguns comandos UNIX (echo, sed, cat etc.)

Caso não se sinta preparado, antes de prosseguir leia alguns tutoriais e ganhe segurança para iniciar sua construção. Uma boa recomendação é o Guia Foca Linux (presente no CD-ROM desta edição).

Antes de tudo, faça um backup de todos os seus documentos importantes. Trabalharemos com reparticionamento de disco e não queremos perder nada com possíveis erros provocados por uma eventual falta de atenção. Todo o processo de criação se dará em uma partição separada.

O mínimo de espaço recomendado é 1.3 GB, o suficiente para os códigos-fonte e pacotes já compilados para um sistema-base. Caso sua intenção seja criar um sistema completo, com interface gráfica e vários serviços rodando, é interessante que se reserve pelo menos 10 GB de espaço.

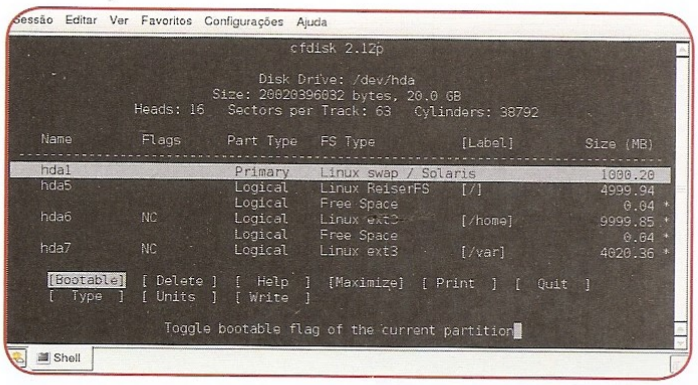
Configurando a tabela de partições

Usaremos o programa cfdisk para criar sua nova partição. Se não tiver espaço livre em seu HD, use o programa qtparted para

redimensionar as partições atuais e abrir mais espaço, sempre respeitando quantidade de dados gravados nessas partições.

O funcionamento de ambos é bem dedutível, porém é preciso ter cuidado para não apagar partições já utilizadas. Memorize onde está sua nova partição (hda1, hda5, hda6 etc.). Após a criação, salve-a e reinicie seu micro.

Esses passos devem ser efetuados como usuário root, ou seja: qualquer erro poderá ser fatal.



Agora que temos nossa tabela de partições preparada para receber nosso ambiente de desenvolvimento, formataremos nossa nova partição. Utilizaremos o formato ext3, em que hdXX corresponde à localização de sua nova partição (/dev/hda6, por exemplo)

```
# mkfs.ext3 /dev/hdXX
```
(cria uma partição no sistema de arquivos ext3)

Criando o ponto de montagem para a nova partição

Ainda como usuário root, crie um diretório /mnt/lfs:

```
# mkdir /mnt/lfs
```
(cria um subdiretório lfs no diretório mnt)

Agora, adicionaremos no arquivo /etc/fstab as informações necessárias para que esse diretório seja facilmente acessível por nosso usuário lfs. O arquivo fstab é um arquivo de texto, em que cada linha corresponde a um ponto de montagem em seu HD. Um exemplo de fstab é:

```
/dev/hda1 none    swap    sw      0       0
/dev/hda5 /       reiserfs notail 0       1
```

Supondo ainda que sua nova partição seja a hda6 o conteúdo desse fstab seria:

```
/dev/hda1 none    swap    sw      0       0
/dev/hda5 /       reiserfs notail 0       1
/dev/hda6 /mnt/lfs ext3   defaults 0      2
```

Para alterar esse arquivo, use um editor de textos qualquer como usuário root. O mcedit é bem interessante e bastante amigável.

Após criar a nova entrada no fstab, montaremos nossa nova partição. Note que, como ela já está setada, não será necessário montar a cada vez que o sistema for iniciado, pois o fstab garante que a partição terá seu ponto de montagem durante o boot.

```
# mount /mnt/lfs
```
(monta a partição correspondente de acordo com a configuração do arquivo fstab)

Se a montagem correu bem, nenhum erro será apresentado na tela. Caso tenha problemas, verifique os passos acima. Criaremos um diretório para colocar os códigos-fonte de nosso projeto:

```
# mkdir /mnt/lfs/sources
```
(cria o subdiretório sources no subdiretório /mnt/lfs)

Criando e setando permissões para o usuário de desenvolvimento

Trabalharemos com um novo usuário chamado lfs. Para ele, criaremos um grupo também com o nome de lfs. Veja, a seguir, os comandos que devem ser executados como root:

```
# groupadd lfs
```
(cria o grupo lfs)

```
# useradd -s /bin/bash -g lfs -m -k /dev/null lfs
```
(cria o usuário lfs com bash, home e não copia o padrão de home do sistema atual)

Grupo e usuário criados. Hora de acertar as permissões e definir variáveis:

```
# chown lfs.lfs /mnt/lfs/sources
```
(dá a "posse" do subdiretório citado ao usuário e grupo lfs)

```
# export LFS=/mnt/lfs
```
(define o patch para a partição de desenvolvimento)

```
# echo $LFS
```
(testa o patch que acabamos de criar. Tem de retornar à partição em questão)

Importante: confira se o comando "echo $LFS" retorna o diretório /mnt/lfs.

Agora, logue-se com seu usuário lfs. Em seguida, entre no diretório de desenvolvimento e tente criar um arquivo. Se tudo foi configurado corretamente, poderemos baixar os fontes necessários para o início do projeto:

estão montadas. Verifique o capítulo 9.3 e veja a forma correta para o procedimento antes de rebootar.

Sistema-base instalado e bootável

Você conseguiu! Resta "ser contado" (se tornar um usuário registrado) e fazer parte da comunidade LFS. O conhecimento que você adquiriu poderá ajudar outras pessoas que passarão pelo mesmo percurso. Para "ser contado" basta preencher o formulário em:

http://linuxfromscratch.org/cgi-bin/lfscounter.cgi

Listas de discussão

Português: *http://listas.codigolivre.org.br/mailman/listinfo/lfs-br-i18n*
Inglês: *http://linuxfromscratch.org/mailman/listinfo/lfs-support/*
FAQ: *www.linuxfromscratch.org/faq/*

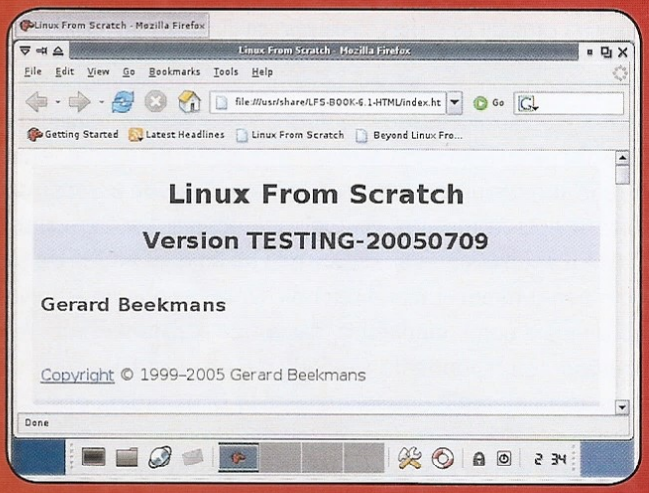
* Angelo Valle é consultor de redes Linux, desenvolvedor LAMP (Linux, Apache, MySQL, PHP), administrador de redes e colaborador do projeto LFS-BR. Site: www.avalle.org



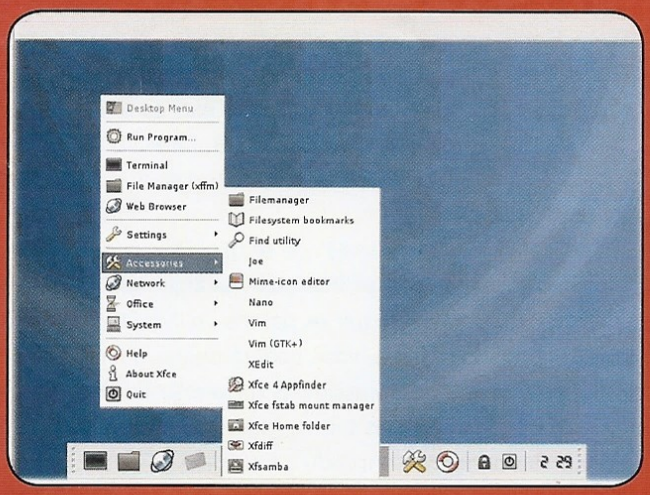
O gerenciador de arquivos do XFCE

Live CD é opção para construção de sistema

O projeto Linux From Scratch (www.linuxfromscratch.org/livecd/) dispõe de um Live CD que pode ajudar bastante na construção de seu sistema. Ele boota independentemente do sistema instalado no HD e também dá uma idéia de como fica um sistema baseado no LFS.



O Mozilla Firefox também está no Live CD



A interface gráfica XFCE rodando pelo Live CD do Linux From Scratch



A imprescindível linha de comando, por onde o "trabalho sujo" é feito

CONTEÚDO DO CD

Confira os programas especialmente selecionados para você

Nesta edição da Geek Especial, você encontra ferramentas para criar seu próprio sistema operacional ou para modificar um já existente. Há também ferramentas para manutenção de sistemas, emuladores e muito mais.

Para ter acesso a esse conteúdo, basta acessar a interface do CD que acompanha a revista. É muito simples: basta inseri-lo na gaveta do seu CD-ROM e aguardar alguns instantes até a abertura completa da tela principal. Caso o recurso de auto-reprodução não esteja ativado, você pode abrir o CD clicando em Executar, no menu Iniciar do Windows, e digitando a letra correspondente ao seu drive de CD-ROM (D:\ ou E:\, por exemplo). Escreva o comando "BROWSER.EXE" e pronto! A interface será exibida na tela e estará pronta para a navegação.



EMULADORES

Qemu - Possibilita rodar alguns sistemas operacionais dentro do Linux (ou ele mesmo) por meio de um HD virtual.

Mac-on-Linux - Aplicativo que possibilita rodar o Mac OS no Linux/ppc. Requer Linux.

DOSEMUE - Permite que o usuário do Linux utilize comandos do DOS no Linux ou instale algum aplicativo simples, feito para DOS, e o execute.

DOSbox - Emulador de DOS com recursos para emulação de jogos antigos. Traz novos comandos no shell, emulação correta da Creative GameBlaster, suporte para gravação de áudio em arquivos WAV, screenshot, MSCDEX para montar CD-ROMs e diretórios, como falsas unidades de CD-ROM, emulação de MPU401, cache de diretório para suportar longos nomes de arquivos e montar diretórios em outras plataformas.

Bochs - PC emulator desenvolvido em C++ que permite rodar as mais populares plataformas. Inclui emulação da CPU Intel x86, dispositivos I/O e BIOS. Pode ser compilado para emular CPUs 386, 486 ou Pentium. É capaz de rodar diversos sistemas operacionais, inclusive Linux, Windows 95, DOS e Windows NT 4.0.

FERRAMENTAS PARA MANUTENÇÃO DE SISTEMAS

TweakAll 3 - Configurações extras para melhorar a performance do Windows, como boot e outras.

DESTAQUE

VMWare Workstation 5.0 - Máquina virtual. Permite rodar outros sistemas operacionais que estejam no mesmo computador ou em rede. É preciso se cadastrar no site do desenvolvedor para receber o serial que desbloqueia o programa. www.vmware.com/vmwarestore/newstore/wkst_eval_register.jsp

Wine - Rode softwares do Windows. O programa é uma implementação das APIs do Windows, mas com código livre.

Xebian - sistema operacional baseado no Debian. O Xebian foi desenvolvido para ser usado no console Microsoft Xbox.

Ietweak 2 1.0 - Programa desenvolvido para configurar e personalizar o Microsoft Internet Explorer.

RegKey Backup 1.0 - Faz backup e recupera chaves e subchaves do Registro do Windows. Ainda que não seja um editor de Registro, é uma ferramenta fácil de usar para recuperar chaves que foram modificadas por programas ou pela Web.

ANÁLISE DE SISTEMAS

Aida32 - System Information - Informações do sistema, diagnósticos e comparativos para plataforma Windows. O programa mostra os detalhes de todos os componentes de seu PC. Os resultados podem ser enviados para a tela, impressora, arquivo de texto, HTML, CVS ou XML, traz informações remotas do sistema e gerenciamento da rede. Recomendado para usuários corporativos.

Intel Processor Utility - Programa que identifica seu processador Intel e verifica se ele está operando na freqüência correta. Com Pentium III, ele avisa se o processador está 'overclocked'. Suporta Pentium IV, mas em Windows 2000 ou NT. É necessário ter os direitos de administrador.

CoolBeans - Esse programa monitora o uso do processador, a memória e o cache de disco. Traz ainda diversas opções de cores e tamanhos de janela.

CPUSpeed - Veja a velocidade real de seu computador (CPU).

CPUInfo - Exibe informações detalhadas sobre o(s) processador(es), como o tipo e fabricante, a taxa de clock, cache nível 1 e 2 etc.

FreeMeter - Confira gráficos sobre utilização do processador e do espaço em discos rígidos e removíveis, utilização da memória, tráfego da rede e taxas de velocidade das transferências dos discos. Roda tanto local como em unidades mapeadas da rede. Os gráficos são em tempo real e apresentam o que realmente acontece no momento.

HotCPU Tester - Teste seu PC para detectar possíveis erros. O programa permite verificar como está seu micro testando a CPU e todas as partes da placa-mãe, além de detectar erros e diagnosticar sistemas quanto à estabilidade.

Performance Test - Faça uma série de testes de velocidade em seu computador (Benchmarks). Esse programa verifica CPU, memórias, discos, CD, DVD, MMX e DirectX e faz comparações com PCs-padrão para você verificar como anda a potência do seu. Os resultados podem ser gravados facilmente.

Watch - Monitoração do sistema durante a instalação de um software ou um hardware. Ele detecta todas as mudanças efetuadas no HD (criação de arquivos e pastas), arquivos INI e no registro. Tudo é guardado em um banco de dados e pode ser acessado posteriormente.

ClearInfo - Programa para monitorar dados, como a taxa de uso da CPU, da memória virtual e da memória física. Também exibe dados como o nome da CPU, nome do usuário logado, versão do Windows, display de vídeo etc.

MemViewer - Esse programa mede a quantidade de memória utilizada por seu PC e a apresenta em forma de barra na bandeja do Windows. Suporta skins e pode ser posicionado nas bordas da tela, na posição horizontal ou vertical.

Samurize - O Samurize fornece todas as informações sobre seu computador em seu desktop. Ele informa sobre uso de CPU, tráfego de rede, temperatura do sistema, memória utilizada e muito mais. Roda em Windows XP/2000.

Linux Monitor 2.ORC1 - Monitora e detecta os erros do sistema. Ele checa os espaços em disco e a quantidade de "tranqueiras" instaladas em sua máquina. Depois, cria um relatório de erros que o sistema produz. A versão do servidor dessa aplicação pode fazer uso tanto do syslog, como do PostgreSQL para criar registro dos erros.

Nulasys - Esse programa exibe algumas informações de seu computador, como velocidade atual do processador, vendor name, cpu name, generic name, freqüência do monitor, nome da placa de vídeo e a quantidade de memória RAM.

Linux Monitor 2.ORC1 - Monitora e detecta os erros do sistema. Ele checa os espaços em disco e a quantidade de "tranqueiras" instaladas em sua máquina. Depois, cria um relatório de erros que o sistema produz. A versão do servidor dessa aplicação pode fazer uso tanto do syslog, como do PostgreSQL, para criar registro dos erros.

Sistemas operacionais

ReactOS - Projeto para desenvolver um sistema operacional de qualidade e compatível com as aplicações e drivers do Microsoft Windows.

MenuetOS - Sistema Operacional 32 bits totalmente escrito em Assembly que cabe em um único disquete. Ele é multitarefa, possui interface gráfica de até 16 milhões de cores em 1.280x1.024 pixels, editor e compilador de Assembly, Ethernet: TCP / UDP / ICMP / IP, servidor HTTP, MP3 e de e-mail.

FreeDOS - O FreeDOS é uma reimplementação livre do MS-DOS-GPL. Requer gravador de CD.

Athene - Sistema operacional desenvolvido pela Rocklyte Systems para uso doméstico ou em escritório. É freeware (para uso não comercial) e Requer disquete.

BraSO - O Braso é um sistema operacional microkernel que tem por objetivo principal ajudar àqueles que gostam ou querem aprender como fazer um sistema operacional. O sistema ainda está em fase experimental e foi escrito em Assembly e C.

Apostilas

Coleção de apostilas e tutorias de linguagens C, C++, Csharp, Windows, Linux e Assembly.

SO para Consoles

KOS - Sistema operacional desenvolvimento para o Dreamcast.

NewOS - Sistema operacional para Dreamcast com ênfase no design e na portabilidade. Foi desenvolvido quase totalmente em C/C++, com uma pequena parte em Assembler.

Chegou Vintage Motors

revista + vídeo



apaixonados por carros

A Vintage Motors traz o que existe de melhor no antigomobilismo nacional e internacional. Além dos vídeos de matérias exclusivas que acompanham cada edição.

Já nas bancas

www.vanblad.com.br

VAN BLAD

Geek especial

DESTAQUES

VMWare Workstation 5.0 - Máquina virtual. Permite rodar outros sistemas operacionais que estejam no mesmo computador ou em rede. Se você usa o Linux e quer rodar o Windows que está instalado no mesmo computador, basta abrir uma janela e o sistema do Windows estará lá para você rodar.

Wine - Rode softwares do Windows. O programa é uma implementação das APIs do Windows, mas com código livre.

Xebian - Sistema operacional baseado no Debian. O Xebian foi desenvolvido para ser usado no console Microsoft Xbox.

MATERIAL DIDÁTICO

Mais de 140 apostilas

• Assembly • C • Linux • Lógica • Pascal/Delphi • Windows

Emuladores

Qemu - Possibilita rodar alguns sistemas operacionais dentro do Linux (ou ele mesmo) por meio de um HD virtual.

Mac-on-Linux - Aplicativo que possibilita rodar o Mac OS no Linux/ppc. Requer Linux.

DOSEMUE - Permite que o usuário do Linux utilize comandos do DOS nesse Sistema Operacional ou instale algum aplicativo simples, feito para DOS, e o execute.

DOSbox - Emulador de DOS com recursos para emulação de jogos antigos.

Sistemas operacionais

ReactOS - Projeto para desenvolver um sistema operacional de qualidade e compatível com as aplicações e drivers do Microsoft Windows.

MenuetOS - Sistema operacional 32 bits totalmente escrito em Assembly que cabe em um único disquete. Ele é multitarefa, possui interface gráfica de até 16 milhões de cores em 1.280x1.024 pixels, editor e compilador de Assembly, Ethernet: TCP / UDP / ICMP / IP, servidor HTTP, MP3 e de e-mail.

FreeDOS - O FreeDOS é uma reimplementação livre do MS-DOSGPL. Requer gravador de CD.

Athene - Sistema operacional desenvolvido pela Rocklyte Systems para uso doméstico ou em escritório. É freeware (para uso não comercial) e requer disquete.

OSKit - Conjunto de ferramentas e tutoriais para a construção de sistemas operacionais.

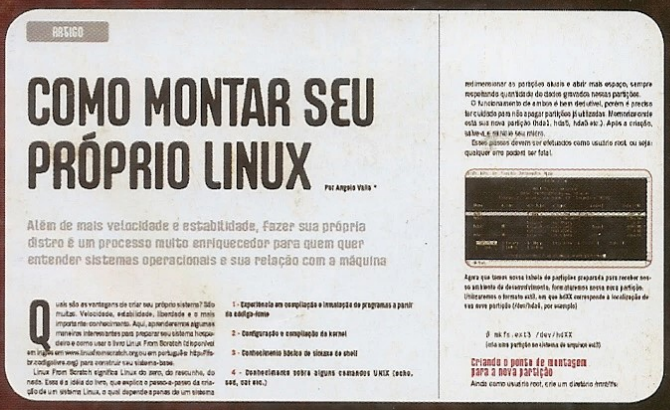
REQUISITOS MÍNIMOS

Para rodar o CD sem nenhum problema, é necessário que seu computador tenha a seguinte configuração mínima:

- Processador Pentium II ou superior
- 64 MB de memória RAM
- 16 MB de memória RAM de vídeo em resolução de 800x600 pixels
- Placa de som
- Drive de CD-ROM com velocidade 2x

Atenção: esses requisitos mínimos são focados na interface do CD. As configurações podem variar de acordo com os programas.

Como montar seu Linux


BÁSICO

COMO MONTAR SEU PRÓPRIO LINUX

Por Angelo Viola *

Além de mais velocidade e estabilidade, fazer sua própria distro é um processo muito enriquecedor para quem quer entender sistemas operacionais e sua relação com a máquina

Conheça os procedimentos necessários para montar sua distro Linux

Software livre


SOFTWARE LIVRE

APRENDA CRIANDO CÓDIGO ABERTO

Por Estevam Sabino

Criar um sistema do zero é impossível sem ampla experiência. Um bom aprendizado está nos projetos open source

Enriqueça seu currículo participando de grupos de criação de sistemas operacionais