Linux do zero

www.mentebinaria.com.br

Autor: Fernando Mercês (fernando em mentebinaria.com.br)
Primeira versão: setembro de 2008
Versão: 1.1 (publicada em outubro de 2008)

Índice

Capítulo 1
1.1 O papel do sistema operacional.....1
1.2 Breve história do Linux.....1
1.3 O papel do kernel.....2
1.4 Distribuições GNU/Linux.....2

Capítulo 2
2.1 Planejamento e instalação.....3
2.2 Particionamento.....3
2.3 Sistemas de arquivos.....4
2.4 Gerenciadores de incialização.....6
2.5 Pacotes.....7

Capítulo 3
3.1 O básico do sistema.....8
3.2 Estrutura de diretórios.....8
3.3 Níveis de operação.....8
3.4 Operações básicas do shell.....9

Capítulo 4
Hardware e drivers.....15
4.1 Comandos de gerenciamento de hardware.....15
4.2 O sistema /proc.....17
4.3 Módulos.....19

Capítulo 5
Gerenciamento de pacotes.....20
5.1 DPKG.....20
5.2 RPM.....21
5.3 APT.....21
5.4 YUM.....23
5.5 Tarballs.....23

Capítulo 6
Controle de acesso.....25
6.1 Usuários e grupos.....25
6.2 Arquivos e diretórios.....27

Capítulo 7
Montagem e formatação de mídias.....29
7.1 Partições de disco.....29
7.2 Disquetes.....30
7.3 Cds/DVDs.....30
7.4 Mídias USB.....30

Capítulo 8
Configuração de rede.....31
8.1 Comandos de configuração.....31
8.2 Arquivos de configuração.....32

Capítulo 9
Ambiente gráfico.....34
9.1 Servidores X.....34
9.2 Gerenciadores gráficos.....34

Símbolos e convenções utilizados na apostila

Para otimizar o texto da apostila, alguns símbolos e convenções foram utilizados:

negrito – Usado para chamar a atenção para determinada palavra.
Itálico – Usado para nomes de comandos, pacotes e arquivos em geral.
- Evidencia um comando que deve ser iniciado pelo usuário root (administrador do sistema).
$ - Evidencia um comando que pode ser iniciado por um usuário comum.

 **- Exibe uma dica ou informação útil sobre o assunto em contexto.**

Capítulo 1 - O Papel do sistema operacional

O sistema operacional (SO) detém um papel básico no conceito dos computadores: criar uma interface entre o hardware e o usuário. Seu propósito é estabelecer um ambiente para que o usuário possa executar programas (software). O SO precisa oferecer um ambiente *conveniente* para o uso do PC e, em segundo plano, *eficiente*.

Um sistema computacional é dividido em quatro partes básicas, ordenadas do nível mais alto para o mais baixo: o usuário, os softwares, o SO e o hardware.

É o SO quem gerencia o hardware do PC e aloca seus recursos para que os programas os utilizem. Por sua vez, os programas fornecem uma interface para que nós os utilizemos.

Programar um SO significa escrever um software (porque o SO não deixa de ser um software) em linguagem que o hardware entenda. Essa linguagem é a linguagem de máquina e o hardware que deve entendê-la é o microprocessador.

A interação Linguagem de máquina x Microprocessador

O processador (ou microprocessador) é um componente eletrônico. E como todos os outros componentes eletrônicos, ele só entende pulsos elétricos. Portanto, a linguagem que é capaz de dizer algo ao microprocessador é o binário, que tem apenas dois "comandos": 0 e 1, ou seja, ausência ou presença de energia elétrica.

Claro que é humanamente inviável criar um programa somente com zeros e uns. Por isso, o sistema hexadecimal foi utilizado para programar o microprocessador (já que qualquer número hexa pode ser convertido em binário) e posteriormente, uma linguagem de programação chamada Assembly (ou linguagem de montagem). Todos os comandos desta linguagem possuem equivalentes em hexadecimal, que por sua vez, possuem equivalentes em binário, que enfim, tornam possível a compreensão por parte do microprocessador (lembre-se que, neste contexto, binário nada mais é que ausência ou presença de energia).

1.1 Breve história do Linux

Há alguns anos um sistema operacional chamado UNIX fazia muito sucesso, mas era proprietário (e caro). Um professor universitário desenvolveu uma versão reduzida, mas de código-aberto deste potente sistema e chamou-a de MINIX. Como o nome sugere, um "mini-UNIX". Este SO deveria ser utilizado em faculdades para fins de estudo, somente. Foi então que um dos universitários usuários do MINIX resolveu criar seu próprio SO, com base no MINIX e na tentativa de implementar as melhores funções do UNIX, só que gratuitamente, surgiu o Free UNIX. Este estudante era Linus Torvalds que logo após de divulgar seu Free UNIX recebeu uma ordem judicial por causa do nome escolhido (lembre-se que o UNIX é proprietário), mas isso não era problema. Bastava mudar o nome. Surge então o Linux 1.0.

E os programas? Ele desenvolveu o núcleo do SO (conhecido também como kernel), que sabia gerenciar o hardware mas que software existia na época para Linux? A resposta é nenhum, claro. Ele então resolveu portar (adaptar o código-fonte e recompilar) pequenos softwares que já funcionavam no UNIX e no MINIX. Um deles foi o bash (um interpretador de comandos) e o outro foi o gcc (o compilador C do UNIX).

1.2 O papel do kernel

Como mencionado anteriormente, o Linux é um kernel e não um SO. Lembre-se que por definição um SO deve interagir com o hardware e o usuário mas sem nenhum programa, como o usuário interage? No mínimo é necessário um interpretador de comandos, como o COMMAND.COM do antigo MS-DOS e o bash, que foi portado para o Linux. Se existem formas de se interagir com este kernel, por via de programas, aí sim nomeamos o conjunto de Sistema Operacional.

1.3 Distribuições GNU/Linux

Linus Torvalds usou muitos programas de uma empresa chamada Free Software Foundation, que sonhava em desenvolver um sistema operacional, mas começou o desenvolvimento pelos programas e não pelo kernel. A FSF iniciou um projeto chamado GNU, onde desenvolveu vários aplicativos gratuitos e Linus os usou para distribuir junto com o seu kernel Linux. Desde então ele já tinha um SO, mas em troca, o nome foi GNU/Linux. Este é o SO. Linux é o kernel, puramente.

Rapidamente empresas, universidades e grupos de programadores começaram a anexar seus próprios programas ao kernel e distribuir o conjunto. Surgem então as distribuições GNU/Linux, que avançam até hoje. Em 2006 o site DistroWatch (http://distrowatch.com) já apontava mais de 350 distribuições Linux ativas. Exemplos de tradicionais distribuições Linux são o Slackware, o Red Hat, o Debian e o SUSE.

Abaixo uma lista com as principais distribuições Linux e seus respectivos sites oficiais.

Arch Linux – www.archlinux.org
Debian GNU/Linux – www.debian.org
Damn Small Linux – www.damnsmalllinux.org
Fedora Core – fedoraproject.org
Gentoo – www.gentoo.org
Knoppix – www.knoppix.de
Kurumin – www.guiadohardware.net/kuruminlinux
Linux Mint – www.linuxmint.com
Mandriva – www.mandriva.com
OpenSUSE – www.opensuse.org
Red Hat – www.redhat.com
Slackware – www.slackware.com
SUSE – www.novell.com/linux
Ubuntu – www.ubuntu.com

Capítulo 2 - Planejamento e instalação

Para um melhor aproveitamento do sistema, é essencial planejar sua implementação. O Linux provê um sistema muito flexível que pode ser instalado em várias arquiteturas facilmente mas, se feito um planejamento antes da implementação, pode ficar ainda melhor. Um bom exemplo disso é o esquema de particionamento e consumo de memória, que podem gerar resultados desagradáveis se mal planejados ou ignorados.

2.1 Particionamento

Na implementação de todos os sistemas operacionais, raramente é recomendável que utilizemos somente uma partição e, se desejar mais segurança, é altamente recomendável que usemos mais de um disco rígido. No Linux não é diferente, inclusive, criar uma partição extra para a memória virtual (swap) é obrigatório.

Swap file (ou arquivo de troca) é o arquivo usado para simular memória RAM. No Windows 2000 e superiores, ele fica no raiz da partição do sistema e chama-se *Pagefile.sys*. Em versões antigas do Windows, ele fica no diretório de instalação do Windows e chama-se *Win386.swp*. Já no Linux, toda uma partição é utilizada como área de troca e esta partição chama-se *swap*.

Um detalhe interessante é que o Windows usa o arquivo de paginação (um outro nome para o arquivo de troca / memória virtual) desde a inicialização do sistema. Sua otimização é baseada na regra de prioridade de uso dos dados, ou seja, se há um dado em memória que não é utilizado por muito tempo, este vai para o swap, para liberar memória física. O problema é que quando este dado precisar ser acessado, ele vai ter de ser recuperado do swap, que está no HD, para a memória RAM novamente. Como o HD é muito mais lento que a memória RAM, há uma grande perda de performance. Já o Linux, só utiliza o swap se a memória física estiver totalmente cheia. Como desvantagem, a memória pode ficar cheia de dados que você não vai mais utilizar na sessão atual (por escolha sua, pois o "lixo" sempre é removido).

O Linux gerencia os dispositivos de armazenamento nomeando-os no formato /dev/xdy, onde "x" é substituído pelo tipo do dispositivo ("s" para dispositivos SATA, SCSI ou memórias flash e "h" para dispositivos IDE). O "y" é substituído por uma letra que varia alfabeticamente de acordo com a posição do dispositivo na placa-mãe (IDE primária, secundária, slave, master, porta SATA 0, 1, etc). Por exemplo, um HD IDE, conectado à IDE primária da placa-mãe será reconhecido como /dev/hda. Se fosse um dispositivo SATA conectado à porta SATA 0, este seria o /dev/sda.

As partições são acessadas no formato /dev/xdyn, onde "n" é o número da partição, de acordo com sua posição no HD e seu tipo. Para partições primárias, os números variam de 1 a 4, já que os HDs só podem ter quatro partições primárias. Para as lógicas, de 5 em diante.

Por exemplo, a segunda partição primária de um HD IDE que está conectado à IDE secundária será /dev/hdb2. Já a primeira partição lógica de um HD SATA, na porta SATA 0 será a /dev/sda5. Os pen-drives seguem a mesma lógica, mas eles geralmente têm uma só partição e são sempre reconhecidos como dispositivos SATA/SCSI. Se você tiver um HD SATA com duas partições primárias (/dev/sda1 e /dev/sda2), a partição do seu pen-drive será a /dev/sdb1.

As mídias ópticas (CD-ROM, DVD, etc) não podem ser particionadas. Portanto, seu acesso só depende do barramento ao qual está conectado (IDE, SATA ou SCSI). Por exemplo, um drive DVD-RW na posição slave da IDE primária será acessado por /dev/hdb.

Abaixo, uma tabela que relaciona os HDs e suas partições:

Barramento	Posição	Prioridade	Nome	Partições primárias / acesso	Partições lógicas
IDE	Primária	Master	/dev/hda	/dev/hda1 /dev/hda2 /dev/hda3 /dev/hda4	/dev/hda5 /dev/hda6 etc...
IDE	Primária	Slave	/dev/hdb	/dev/hdb1 /dev/hdb2 /dev/hdb3 /dev/hdb4	/dev/hdb5 /dev/hdb6 etc...
IDE	Secundária	Master	/dev/hdc	/dev/hdc1 /dev/hdc2 /dev/hdc3 /dev/hdc4	/dev/hdc5 /dev/hdc6 etc...
IDE	Secundária	Slave	/dev/hdd	/dev/hdd1 /dev/hdd2 /dev/hdd3 /dev/hdd4	/dev/hdd5 /dev/hdd6 etc...
SATA/SCSI	0	-	/dev/sda	/dev/sda1 /dev/sda2 /dev/sda3 /dev/sda/4	/dev/sda5 /dev/sda6 etc...
SATA/SCSI	1	-	/dev/sdb	/dev/sdb1 /dev/sdb2 /dev/sdb3 /dev/sdb4	/dev/sdb5 /dev/sdb6 etc...

Agora que já sabemos como os dispositivos são nomeados, podemos definir as partições para a instalação do sistema. O mínimo que o Linux exige para ser instalado são duas partições (uma para o sistema raiz e outra para o swap). Mas se você quiser que seus arquivos pessoais fiquem numa partição separada (o que pode ser muito útil), basta criar uma partição a mais e montar nela o diretório /home. Não se preocupe se você não souber o que é o /home, muito menos o que é montagem. Veremos isso mais adiante.

Neste curso básico, não abordaremos o particionamento mais complexo, que pode exigir mais de cinco partições no disco. Esse esquema é útil para servidores e sistemas de alto desempenho e confiabilidade mas foge do nosso escopo.

2.2 Sistemas de arquivos

No mundo Linux, sistemas de arquivos podem aparecer de duas maneiras: o conjunto de regras para armazenar dados numa partição ou os diretórios montáveis nestas partições (como o / e o /home). Nesta seção, trataremos do primeiro caso.

Você certamente já ouviu falar em formatação. Esta palavra vem da idéia de criar um formato, ou seja, definir um sistema de arquivos e não de apagar.

Há dois tipos de formatação: a lógica e a física. A extinta formatação física servia para preencher todos os setores de um disco rígido com zeros, incluindo seu MBR (Master Boot Record ou Setor principal de inicialização) e recalcular a relação CHS, que relaciona cilindros, cabeças e

setores. Já a lógica, que é a que mais usamos, é somente aplicável às partições (e não ao disco) e serve para criar um filesystem na partição desejada. Se houver arquivos nesta partição, eles não serão apagados, mas ficarão inacessíveis porque foram gravados utilizando o filesystem anterior e o novo não os conhece.

É por este motivo que dados de uma partição formatados logicamente podem ser recuperados: Os dados não são apagados. A formatação só define um filesystem e esta operação é feita no início do disco e não no disco inteiro.

O Linux pode ser instalado em muitos sistemas de arquivos diferentes. Alguns exemplos são ext2, ext3, ReiserFS, XFS e JFS. Abordaremos aqui os filesystem ext3 e o ReiserFS, por serem os mais comuns.

O ReiserFS é mais rápido que o ext3 na maioria dos casos e também faz com que arquivos pequenos ocupem menos espaço do que ocupariam num sistema ext3. Ambos possuem uma tecnologia de recuperação de arquivos em caso de falta de energia chamada journaling, que consiste em criar um tipo de log (registro de ocorrências) para poder recuperar o arquivo depois de uma falha elétrica ou desligamento repentino por qualquer outro motivo.

Estes filesystems são baseados em inodes, que são pequenos arquivos que armazenam informações sobre cada arquivo no sistema. Essas informações são chamadas de metadados.

O inode de um arquivo é um conjunto de metadados que define, por exemplo, os atributos do arquivo, quem é o dono, a hora de sua última modificação, as permissões de acesso a este arquivo, dentre outras informações.

Agora imagine que você digitou um texto e mandou salvar. Neste exato momento, o fornecimento de energia foi interrompido. O arquivo não foi salvo mas seu inode sim. Isso gerará uma inconsistência no seu filesystem.

O journaling consiste em fazer atualizações dos metadados e registrá-las em um log, antes que os dados sejam salvos. Ou seja, ao clicar em Salvar, primeiro o inode foi criado, depois o journaling entrou em ação e registrou no log que os metadados do arquivo foram alterados e que a situação anterior é que tais metadados nem existiam. Quando a energia for restabelecida, a condição anterior do arquivo (inexistente) será restaurada. Você perdeu o que salvou. Fato. Mas manteve o filesystem consistente.

O ext3 faz journaling nos metadados e nos dados. Já o ReiserFS, somente nos metadados.

O ext3 reserva 5% do espaço total da partição para controle, enquanto o ReiserFS reserva aproximadamente 33 MB, independente do tamanho da partição.

Para resolver inconsistências, um comando útil é o *fsck* (filesystem checker), equivalente ao *chkdsk*/*scandisk* do Windows. Ele deve ser rodado como root e com a partição de destino desmontada. Sua sintaxe é # fsck <partição> (no formato /dev/xdyn).

2.3 Gerenciadores de inicialização

Alguém tem que dizer para o BIOS (Basic Input/Output System) que existe um kernel de SO numa partição específica e este alguém é um gerenciador de boot. No mundo Linux, há dois gerenciadores amplamente utilizados: o LILO (Linux Loader) e o GRUB (GRanted Unified Bootloader). Ambos também possuem a capacidade de inicializar outros sistemas operacionais tais como Windows, FreeBSD e outros.

Estes bootloaders podem ser instalados no MBR (Master Boot Record) ou no primeiro setor de uma partição primária.

O LILO consiste em três partes: o gerenciador de boot em si, um arquivo de configuração (/etc/lilo.conf), um arquivo de mapeamento (/boot/map) que contém a localização do kernel, e o comando *lilo*, que lê o arquivo de configurações e utiliza estas informações para atualizar o arquivo de mapeamento e/ou (re)instalar o gerenciador de boot.

O GRUB tem algumas vantagens em relação ao LILO. Uma delas é que se você instalar uma nova versão de kernel, não precisará reinstalar o gerenciador de boot. Outra é que em desastres o GRUB oferece uma interface de texto, que aceita comandos de manutenção. Estes são muito úteis para recuperar o gerenciador de boot em caso de falhas.

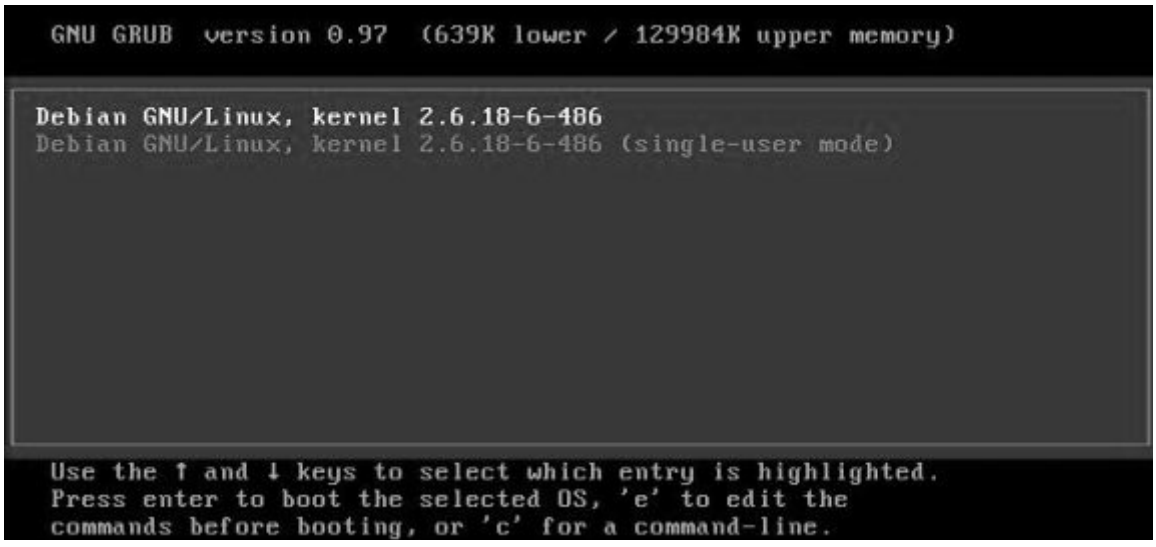
Assim como o LILO, o GRUB também é dividido em partes. São elas:

stage1 – parte executável que reside no MBR ou no primeiro setor da partição. Esta parte é necessária porque a parte principal do GRUB é muito grande para caber num setor só. Este estágio 1 é usado para transferir a execução para o estágio 1.5 ou 2.

stage1.5 – carregado somente se o sistema exigir. Este estágio é específico para cada sistema de arquivos que o GRUB pode carregar. Por exemplo, o arquivo */boot/grub/reiserfs_stage1_5* é o executável do estágio 1.5 para o filesystem ReiserFS.

stage2 – este executável é a parte principal do gerenciador de boot. Ele é quem exibe o menu, chama o *initrd*, carrega o kernel, etc.

Abaixo o menu do GRUB instalado num PC somente com o Debian GNU/Linux:



Um ponto importante tratando-se do GRUB é que ele tem sua própria convenção e

nomenclatura para as partições, onde diz que o primeiro disco rígido é hd0 (independente do disco ser IDE, SATA ou SCSI). O segundo será o hd1 e assim por diante.

Para as partições, basta acrescentar uma vírgula seguida de um número e por tudo entre parênteses. Assim, a primeira partição do primeiro HD é a (hd0,0). A segunda partição deste mesmo HD é a (hd0,1) e assim por diante. Seguindo esta lógica, você seria capaz de definir como o GRUB chama a terceira partição de um segundo HD? Se você respondeu (hd1,2), parabéns! Se errou, releia o trecho acima porque este conceito é essencial para usar o GRUB.

O arquivo de configuração do GRUB fica em */boot/grub/menu.lst* e seu arquivo de mapas é o */boot/grub/device.map*.

Em nosso curso, adotaremos o GRUB como gerenciador de boot. Para saber mais sobre ele, consulte o artigo "Tudo sobre o GRUB" no endereço www.mentebinaria.com.br/index.php?option=com_content&task=view&id=62&Itemid=5

2.4 Pacotes

No mundo Linux, a forma na qual baixamos os programas e bibliotecas são arquivos especiais, chamados de pacotes. Os dois tipos de pacotes existentes são o DEB e o RPM. O primeiro foi criado para o Debian e o segundo, para o Red Hat. No entanto, é imensa a quantidade de distribuições que os utilizam já que tudo que segue a licença GNU/GPL pode ser reutilizado livremente.

Por exemplo, para baixar o software Wireshark, para Linux, uma das opções é baixar o pacote DEB (se estivermos usando uma distribuição baseada no Debian). Outra opção é baixar o pacote RPM (no caso de distribuições baseadas no Rad Hat ou Slackware). Depois disso, basta instalar o pacote com o gerenciador de pacotes, que veremos no capítulo 5.

Uma opção mais avançada é baixar o código-fonte do Wireshark e compilá-lo. Todos esses métodos serão vistos no capítulo 5 mas a compilação de programas exige conhecimentos fora do escopo básico deste SO.

Capítulo 3 - O básico do sistema

O Linux possui muitas particularidades úteis e sua operação consciente é essencial para a segurança do sistema. Neste capítulo veremos o que chamo de "básico técnico". Preste bastante atenção nele pois a partir daqui você vai precisar de todos os conceitos bem sólidos em sua mente.

3.1 Estrutura de diretórios

A árvore de diretórios do Linux, assim como toda árvore do mundo, tem origem na raiz. Este diretório raiz é representado pelo caractere "/".

Cada diretório possui uma função específica, que veremos agora:

/ - raiz do sistema. Todos os diretórios abaixo são sub-diretórios deste.
/bin – Armazena arquivos executáveis (também conhecidos como binários).
/boot – Contém o kernel e os arquivos que gerenciam o boot.
/cdrom – É um link (atalho) para o diretório /media/cdrom.
/dev – Mantém os arquivos usados para conexão com os dispositivos de hardware.
/etc – Centraliza (quase) todas as configurações dos programas.
/home – Armazena os arquivos pessoais do usuário. É como o "Meus documentos" do Windows.
/initrd – Cria um ramdisk para carregar o kernel.
/lib – Armazena as bibliotecas, conhecidas como libs, que são equivalentes às DLLs do Windows.
/lost+found – Mantém arquivos recuperados após falhas. Equivalentes aos .CHK do Windows.
/media – Ponto de montagens para mídias removíveis, somente (disquetes, pen-drives, CD-ROM...)
/mnt – Ponto de montagens para HDs.
/opt – Armazena programas que não são nativos da distribuição.
/proc – Não armazena arquivos e sim referências à informações que o kernel produz em tempo real.
/root – O /home do usuário root. Mantém seus arquivos pessoais separados dos de outros usuários.
/sbin – Armazena os executáveis que só o root pode utilizar.
/srv – Armazena arquivos compartilhados, de websites, servidores FTP, etc.
/sys – Neste diretório é montado um filesystem chamado sysfs, que o kernel utiliza.
/tmp – Contém arquivos temporários. Este tem seu conteúdo excluído a cada inicialização.
/usr – Mantém a maioria dos dados do sistema, em modo somente leitura.
/var – Armazena dados variáveis, como logs e spool de impressão.

Lembre-se que você deve conhecer cada diretório da estrutura de diretórios do Linux. Este é um assunto muito importante que o acompanhará em toda a jornada de aprendizado.

3.3 Níveis de operação (runlevels)

No Linux existem 7 runlevels, que vão de 0 a 6. São eles:

0 – desligado
1 – monousuário
2 – multiusuário
3 – multiusuário com rede
4 – reservado local
5 – multiusuário com ambiente gráfico
6 – reinicialização

O runlevel 0 é o estado desligado. Assim, se você mandar seu sistema operar no runlevel 0, ele desligará. Parece bobo, mas é necessário para o desligamento do sistema.

O runlevel 1 é para manutenção no sistema e só suporta que um usuário efetue login no sistema. Logicamente, este usuário é o root, o administrador e "todo-poderoso" do mundo Linux.

O runlevel 2 permite que outros usuários também façam login, mas somente login local.

O runlevel 3 difere do 2 porque também permite o login remoto e acesso à rede.

O runlevel 4 é um modo onde só o administrador local do sistema pode interagir. Ele difere-se do runlevel 1 em poucos detalhes.

O runlevel 5 é usado quando se está usando o ambiente gráfico.

O runlevel 6 reinicia o sistema e segue a mesma lógica de importância que o runlevel 0.

O comando que altera o runlevel atual é o init. E sua sintaxe é:

 # init <runlevel>

Experimente alternar entre os runlevels mas saiba mas saiba que as distribuições Debian-based não respeitam este padrão. No Debian, os runlevels de 2 a 4 são idênticos ao 5.

3.4 Operações básicas no shell

Antes começar, vamos definir o que é um shell. Este é um software interpretador de comandos que roda sobre o kernel Linux. É um meio de comunicação entre você e o kernel. Um dos shells mais usados no mundo Linux atualmente é o BASH (Bourne Again SHell) e é nele que concentraremos nossos estudos. Este executável está localizado em */bin/bash*. Uma lista com todos os shells disponíveis em sua distribuição é mantida em */etc/shells*. Se não estiver satisfeito você pode instalar ou remover mais shells, de acordo com seu gosto ou necessidade.

Já sabemos o que é um shell, agora vamos explorar as características internas do BASH que o tornam uma poderosa linguagem de programação e eterno aliado ao administrador ou usuário do sistema Linux.

O BASH possui comandos internos (também chamados de built-ins), como rotinas de loop, seleção e condição. Neste curso, abordaremos os comandos básicos e não abordaremos o bash scripting (programação de scripts com BASH).

A sintaxe padrão do shell é:

$ comando <parâmetros>

Perceba o espaço entre os dois. Primeiro digitamos o comando e depois suas opções (também chamadas de parâmetros). Por exemplo, para listar o conteúdo de um diretório podemos comandar:

$ ls

Mas se quisermos uma lista detalhada, temos que informar algum parâmetro que o comando ls permita. O parâmetro "-l" pode ser usado assim:

$ ls -l

Outro importante assunto são os metacaracteres (*, ? e []). Se eles estiverem presentes na linha de comando, o shell os interpreta da seguinte forma:

* - ocorrência de qualquer caractere em qualquer quantidade.
? - ocorrência qualquer caractere único.
() - ocorrência de um intervalo personalizado.

Por exemplo, para listar todos os arquivos que comecem com a palavra "mente", usamos:

*$ ls mente**

Se quiséssemos somente os arquivos que terminam em "mente", usamos:

*$ ls *mente*

Agora, os que contenham a palavra "mente":

*$ ls *mente**

Para listar os arquivos que terminam em "mente" seguido de um único caractere qualquer:

$ ls mente?

Para lista os arquivos que tenham seu nome no formato "<número>mente", usamos:

$ ls [0-9]mente

Agora vamos ver alguns dos principais comandos, que usaremos bastante:

cat
Este comando é equivalente ao type, do MS-DOS. Ele recebe como entrada um arquivo e exibe seu conteúdo na saída padrão (stdout) que, por definição, é a tela do monitor.

```
fernando@turion64:~> cat scripts/perlprint.pl
#!/usr/bin/perl
print "Bem-vindo ao Perl.\n";
$num=0x41;
$numb=0326;
print qq/O número é $numb\n/;
```

No exemplo acima, o conteúdo do arquivo perlprint.pl, que está no diretório /home/fernando/scripts é exibido na tela.

ls
Este comando é largamente utilizado. Ele lista o conteúdo de um diretório.

```
fernando@turion64:~/museum> ls
icq v1.02b.exe  winamp v0.20.exe  winzip32.exe
opera v3.0.exe  winrar v2.00.exe
fernando@turion64:~/museum> ls -l
total 2756
-rw-r--r-- 1 fernando users  726753 2008-05-27 22:12 icq v1.02b.exe
-rw-r--r-- 1 fernando users 1190889 2008-05-27 22:08 opera v3.0.exe
-rw-r--r-- 1 fernando users  137216 2008-05-27 22:06 winamp v0.20.exe
-rw-r--r-- 1 fernando users  453501 2008-05-27 22:10 winrar v2.00.exe
-rw-r--r-- 1 fernando users  286720 2008-05-27 22:06 winzip32.exe
fernando@turion64:~/museum> ls -lah
total 2,8M
drwxr-xr-x  2 fernando users 4,0K 2008-07-12 20:49 .
drwxr-xr-x 64 fernando users 4,0K 2008-09-12 11:59 ..
-rw-r--r--  1 fernando users 710K 2008-05-27 22:12 icq v1.02b.exe
-rw-r--r--  1 fernando users 1,2M 2008-05-27 22:08 opera v3.0.exe
-rw-r--r--  1 fernando users 134K 2008-05-27 22:06 winamp v0.20.exe
-rw-r--r--  1 fernando users 443K 2008-05-27 22:10 winrar v2.00.exe
-rw-r--r--  1 fernando users 280K 2008-05-27 22:06 winzip32.exe
```

No exemplo acima, mostrei o uso do ls com os parâmetros mais comuns:

-l – exibe uma lista detalhada.
-a – exibe também os arquivos ocultos.
-h – formato mais amigável nos tamanhos dos arquivos.

mkdir
Cria diretórios.

```
fernando@turion64:~/museum> mkdir curso
fernando@turion64:~/museum> ls -l
total 2760
drwxr-xr-x 2 fernando users    4096 2008-09-12 13:24 curso
-rw-r--r-- 1 fernando users  726753 2008-05-27 22:12 icq v1.02b.exe
-rw-r--r-- 1 fernando users 1190889 2008-05-27 22:08 opera v3.0.exe
-rw-r--r-- 1 fernando users  137216 2008-05-27 22:06 winamp v0.20.exe
-rw-r--r-- 1 fernando users  453501 2008-05-27 22:10 winrar v2.00.exe
-rw-r--r-- 1 fernando users  286720 2008-05-27 22:06 winzip32.exe
```

rmdir
Remove diretórios.

```
fernando@turion64:~/museum> rmdir curso
fernando@turion64:~/museum> ls -ls
total 2756
 716 -rw-r--r-- 1 fernando users  726753 2008-05-27 22:12 icq v1.02b.exe
1168 -rw-r--r-- 1 fernando users 1190889 2008-05-27 22:08 opera v3.0.exe
 140 -rw-r--r-- 1 fernando users  137216 2008-05-27 22:06 winamp v0.20.exe
 448 -rw-r--r-- 1 fernando users  453501 2008-05-27 22:10 winrar v2.00.exe
 284 -rw-r--r-- 1 fernando users  286720 2008-05-27 22:06 winzip32.exe
```

cp
Copia arquivos e também diretórios inteiros (com o parâmetro -R).

```
fernando@turion64:~/museum> cp winzip32.exe zip.exe
fernando@turion64:~/museum> ls -l
total 3040
-rw-r--r-- 1 fernando users  726753 2008-05-27 22:12 icq v1.02b.exe
-rw-r--r-- 1 fernando users 1190889 2008-05-27 22:08 opera v3.0.exe
-rw-r--r-- 1 fernando users  137216 2008-05-27 22:06 winamp v0.20.exe
-rw-r--r-- 1 fernando users  453501 2008-05-27 22:10 winrar v2.00.exe
-rw-r--r-- 1 fernando users  286720 2008-05-27 22:06 winzip32.exe
-rw-r--r-- 1 fernando users  286720 2008-09-12 13:27 zip.exe
```

rm
Remove arquivos ou diretórios inteiros (com o parâmetro -R).

```
fernando@turion64:~/museum> rm zip.exe
fernando@turion64:~/museum> ls -l
total 2756
-rw-r--r-- 1 fernando users  726753 2008-05-27 22:12 icq v1.02b.exe
-rw-r--r-- 1 fernando users 1190889 2008-05-27 22:08 opera v3.0.exe
-rw-r--r-- 1 fernando users  137216 2008-05-27 22:06 winamp v0.20.exe
-rw-r--r-- 1 fernando users  453501 2008-05-27 22:10 winrar v2.00.exe
-rw-r--r-- 1 fernando users  286720 2008-05-27 22:06 winzip32.exe
```

tail
Exibe as últimas 10 linhas de um arquivo (muito útil para checar logs).

```
turion64:~ # tail /var/log/messages
Sep 12 12:24:18 turion64 syslog-ng[1934]: STATS: dropped 0
Sep 12 12:24:40 turion64 smartd[2709]: Device: /dev/sda, 1 Currently unreadab
le (pending) sectors
Sep 12 12:24:40 turion64 smartd[2709]: Device: /dev/sda, SMART Usage Attribut
e: 194 Temperature_Celsius changed from 134 to 130
Sep 12 12:54:40 turion64 smartd[2709]: Device: /dev/sda, 1 Currently unreadab
le (pending) sectors
Sep 12 12:54:40 turion64 smartd[2709]: Device: /dev/sda, SMART Usage Attribut
e: 194 Temperature_Celsius changed from 130 to 127
Sep 12 13:08:01 turion64 sudo: fernando : TTY=pts/1 ; PWD=/home/fernando ; US
ER=root ; COMMAND=/bin/bash
Sep 12 13:24:18 turion64 syslog-ng[1934]: STATS: dropped 0
Sep 12 13:24:40 turion64 smartd[2709]: Device: /dev/sda, 1 Currently unreadab
le (pending) sectors
Sep 12 13:24:40 turion64 smartd[2709]: Device: /dev/sda, SMART Usage Attribut
e: 194 Temperature_Celsius changed from 127 to 125
Sep 12 13:30:44 turion64 sudo: fernando : TTY=pts/1 ; PWD=/home/fernando/muse
um ; USER=root ; COMMAND=/bin/bash
```

df
Exibe estatísticas da utilização dos discos rígidos. O parâmetro "-h" orienta o comando a exibir unidades de grandeza mais amigáveis (K para quilobytes, M para megabytes e assim por diante).

```
fernando@turion64:~> df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda6              20G  5,6G   14G  30% /
udev                  437M  104K  437M   1% /dev
/dev/sda7              46G   43G  641M  99% /home
/dev/sda2              60G   33G   27G  56% /windows/C
```

free
Exibe estatísticas da utilização de memória. O parâmetro "-m" exibe a saída em megabytes.

```
fernando@turion64:~> free -m
             total       used       free     shared    buffers     cached
Mem:           873        781         91          0         26        350
-/+ buffers/cache:        404        468
Swap:         2055          0       2055
```

halt
Desliga o sistema.

reboot
Reinicia o sistema.

fgrep
Este é um filtro rápido, que pode ser utilizado para filtrar o conteúdo de um arquivo, de acordo com uma ocorrência.

```
fernando@turion64:~> fgrep "Bem" scripts/perlprint.pl
print "Bem-vindo ao Perl.\n";
```

cd
Entra num diretório.

```
fernando@turion64:~> cd scripts
fernando@turion64:~/scripts> cd /var
fernando@turion64:/var>
```

echo
Usado para exibir um texto estático ou o conteúdo de uma variável.

```
fernando@turion64:/var> echo "Acesse: www.mentebinaria.com.br"
Acesse: www.mentebinaria.com.br
fernando@turion64:/var> echo $PATH
/usr/bin:/usr/local/bin:/bin:/usr/bin/X11:/usr/X11R6/bin:/usr/games:/opt/kde3
/bin:/usr/lib64/jvm/jre/bin:/usr/lib/mit/bin:/usr/lib/mit/sbin
```

A variável PATH no Linux, assim como no Windows é uma variável de ambiente que armazena os caminhos conhecidos para os programas executáveis. Graças a ela, podemos digitar um comando usando somente seu nome ao invés do caminho completo. Sem ela, o comando *ls* deveria ser comandado na forma completa: */bin/ls*.

Existem centenas de comandos no Linux, cada um com dezenas de opções. Uma lista bem completa pode ser encontrada em www.oreillynet.com/linux/cmd

Capítulo 4 - Hardware e drivers

Uma das grandes vantagens deste SO é ter um gerenciamento de hardware muito bem definido, o que facilita a consulta às informações dos dispositivos de hardware. Para o Linux, tudo é um dispositivo (device, em inglês). Para ver o conteúdo atual da memória, por exemplo, basta comandar *# cat /dev/mem*. Só não espere entender, porque não há texto puro na memória.

Analise o conteúdo do diretório */dev* e tente identificar outros itens de hardware.

4.1 Comandos de gerenciamento de hardware

Já falamos dos comandos *df* e *free*, mas existem outros que facilitam a vida na hora de consultar informações ou configurar itens de hardware. São eles:

lspci
Famoso comando que lista todos os dispositivos conectados ao barramento PCI do PC.

O barramento PCI é o "centro nervoso" dos microcomputadores atuais. Nele estão conectados quase todos os dispositivos. Não confunda barramento PCI com slots PCI. As placas nesses slots também estão conectadas à este barramento, mas ele já existe dentro do PC.

```
turion64:~ # lspci
00:18.0 Host bridge: Advanced Micro Devices [AMD] K8 [Athlon64/Opteron] Hyper
Transport Technology Configuration
00:18.1 Host bridge: Advanced Micro Devices [AMD] K8 [Athlon64/Opteron] Addre
ss Map
00:18.2 Host bridge: Advanced Micro Devices [AMD] K8 [Athlon64/Opteron] DRAM
Controller
00:18.3 Host bridge: Advanced Micro Devices [AMD] K8 [Athlon64/Opteron] Misce
llaneous Control
0e:00.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL8101E PCI Exp
ress Fast Ethernet controller (rev 01)
14:00.0 Ethernet controller: Atheros Communications Inc. AR242x 802.11abg Wir
eless PCI Express Adapter (rev 01)
1a:04.0 CardBus bridge: Texas Instruments PCIxx12 Cardbus Controller
1a:04.1 FireWire (IEEE 1394): Texas Instruments PCIxx12 OHCI Compliant IEEE 1
394 Host Controller
1a:04.2 Mass storage controller: Texas Instruments 5-in-1 Multimedia Card Rea
der (SD/MMC/MS/MS PRO/xD)
1a:04.3 SD Host controller: Texas Instruments PCIxx12 SDA Standard Compliant
SD Host Controller
```

lsusb
Lista os dispositivos conectados ao barramento USB.

```
turion64:~ # lsusb
Bus 006 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 002: ID 062a:0003 Creative Labs
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

dmidecode
Oferece uma extensa lista, com detalhadas informações sobre cada item de hardware no PC. Esta lista é tão grande que seu uso mais comum é especificando o item desejado com o parâmetro "-t".

```
turion64:~ # dmidecode -t memory
# dmidecode 2.9
SMBIOS 2.4 present.

Handle 0x0011, DMI type 16, 15 bytes
Physical Memory Array
        Location: System Board Or Motherboard
        Use: System Memory
        Error Correction Type: None
        Maximum Capacity: 4 GB
        Error Information Handle: Not Provided
        Number Of Devices: 4

Handle 0x0012, DMI type 17, 27 bytes
Memory Device
        Array Handle: 0x0011
        Error Information Handle: No Error
        Total Width: 128 bits
        Data Width: 64 bits
        Size: 512 MB
        Form Factor: DIMM
        Set: 1
        Locator: S1
        Bank Locator: DIMM1
        Type: DDR2
        Type Detail: Synchronous
        Speed: Unknown
        Manufacturer: Not Specified
        Serial Number: Not Specified
        Asset Tag: Not Specified
        Part Number: Not Specified
```

4.2 O sistema /proc

Além destes comandos, temos uma outra fonte de informação que é o sistema de arquivos /proc. Este pode nos fornecer preciosas informações sobre o hardware do PC. Veja alguns exemplos:

Informações sobre o microprocessador:

```
turion64:~ # cat /proc/cpuinfo
processor       : 0
vendor_id       : AuthenticAMD
cpu family      : 15
model           : 72
model name      : AMD Turion(tm) 64 X2 Mobile Technology TL-52
stepping        : 2
cpu MHz         : 800.000
cache size      : 512 KB
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 2
fpu             : yes
fpu_exception   : yes
cpuid level     : 1
```

Informações de uso das interrupções (IRQs):

```
turion64:~ # cat /proc/interrupts
           CPU0       CPU1
  0:    2718437          0  local-APIC-edge      timer
  1:      34391         31   IO-APIC-edge      i8042
  8:          1          0   IO-APIC-edge      rtc0
  9:       5768         92   IO-APIC-fasteoi   acpi
 12:      36675        440   IO-APIC-edge      i8042
 14:      32315        223   IO-APIC-edge      pata_atiixp
 15:          0          0   IO-APIC-edge      pata_atiixp
 16:          1      47763   IO-APIC-fasteoi   ohci_hcd:usb1, HDA Intel
 17:          0          2   IO-APIC-fasteoi   ohci_hcd:usb2, ohci_hcd:usb4
 18:          0          3   IO-APIC-fasteoi   ohci_hcd:usb3, ohci_hcd:usb5,
fglrx[0]@PCI:1:5:0
 19:          1      93703   IO-APIC-fasteoi   ehci_hcd:usb6, wifi0
 20:          0          1   IO-APIC-fasteoi   yenta
 21:          2          1   IO-APIC-fasteoi   ohci1394
 22:      18151       5491   IO-APIC-fasteoi   ahci, sdhc0:slot0, tifm_7xx1
4348:          0          0   PCI-MSI-edge      eth0
NMI:          0          0   Non-maskable interrupts
```

Informações de uso dos endereços de E/S:

```
turion64:~ # cat /proc/iomem
00000000-0009dbff : System RAM
0009dc00-0009ffff : reserved
000d0000-000fffff : reserved
00100000-37e6ffff : System RAM
  00200000-0044f862 : Kernel code
  0044f863-0062f997 : Kernel data
  007e6000-008b5bd7 : Kernel bss
37e70000-37e82fff : ACPI Tables
37e83000-37e84fff : ACPI Non-volatile Storage
37e85000-3fffffff : reserved
50000000-53ffffff : PCI Bus #1a
  50000000-53ffffff : PCI CardBus #1b
54000000-540fffff : PCI Bus #0e
  54000000-5401ffff : 0000:0e:00.0
58000000-5bffffff : PCI CardBus #1b
e0000000-efffffff : reserved
  e0000000-e1afffff : PCI MMCONFIG 0
f0000000-f7ffffff : PCI Bus #01
```

Informações sobre as partições do sistema:

```
turion64:~ # cat /proc/partitions
major minor  #blocks  name

   8     0  156290904 sda
   8     1    1536000 sda1
   8     2   61903661 sda2
   8     3          1 sda3
   8     5    2104483 sda5
   8     6   20972826 sda6
   8     7   48797406 sda7
```

Alguns programas não-nativos como o *lshw, hwinfo* e o *memtest86+* também podem ser úteis na gerência de hardware. Em sistemas Debian, eles podem ser instalado facilmente pelos repositórios APT com o comando *# apt-get install lshw hwinfo memtest86+.*

4.3 Módulos

Os drivers, no Linux, são usados através de módulos. Na verdade, podemos dizer que os drivers são chamados de módulos. Na maioria dos casos, o Linux já vem com os módulos necessários para reconhecer e gerenciar grande parte dos dispositivos de hardware mas pode acontecer de um dispositivo não ter módulo nativo no sistema para seu funcionamento. Neste caso, há três alternativas:

- ✔ Alguns fabricantes oferecem módulos pré-compilados (em forma de pacote) para download. Basta baixá-lo e instalá-lo (você verá como instalar pacotes no capítulo 5).

- ✔ Se os fabricantes, ou até mesmo terceiros, oferecerem o código-fonte do módulo para download, você terá que baixar o tarball (código-fonte compactado), descompactar, compilar este módulo e carregá-lo em memória.

- ✔ Outra opção é usar o driver de Windows para este dispositivo, com algum software Linux que permita este recurso (*ndiswrapper* é um deles, mas só para placas de rede).

Não vamos abordar a compilação de módulos pois é um assunto mais complexo e não se encaixa em um contexto de curso básico de Linux. Não se preocupe com isso por agora pois a maioria dos hardware existentes já são suportados pelo Linux nativamente e a tendência é aumentar cada vez mais esta lista de hardwares compatíveis. ;o)

Capítulo 5 - Gerenciamento de pacotes

Conforme dito antes, um pacote pré-compilado pode ser do tipo DEB ou RPM. As formas de gerenciamento destes dois tipos de pacotes são diferentes. O primeiro é gerenciado por um programa chamado DPKG. O segundo, por um programa que tem o mesmo nome do tipo, ou seja, RPM.

5.1 DPKG (Debian PacKaGe)

Este é o gerenciador de pacotes nativo do Debian, que utilizamos para instalar os pacotes .deb baixados da internet ou adquirido através de outros meios (CD-ROM, etc). Sua sintaxe é:

 # dpkg <opções> nomedopacote.deb

As opções mais comuns do DPKG são:

-P pacote – Remove um pacote instalado e todos os seus arquivos de configuração.
-L pacote – Mostra quais são e onde estão todos os arquivos que o pacote criou no sistema.
-l – Lista todos os pacotes instalados e desinstalados.
-i pacote – Instala um pacote .deb.
-I pacote – Exibe informações sobre um pacote.
-c pacote – Exibe o que será criado pelo pacote. Útil para saber o que será alterado em seu sistema, antes da instalação do pacote.

É importante lembrar das bibliotecas (libraries) neste momento, uma vez que muitos pacotes necessitam de bibliotecas para serem instalados. Por exemplo, o pacote *vi*, um clássico editor de textos, depende de uma biblioteca chamada *libdl.so.2*. Se ela não estiver instalada, não será possível instalar o vi. Essa relação chama-se **dependência de pacotes**. Em nosso exemplo, para resolver este problema, basta baixar o pacote *libdl.so.2.deb* e instalá-lo **antes** do vi. Mas e se o pacote *libdl.so.2.deb* depender de outra lib, que não está instalada? Bem, neste caso você terá que instalá-la.

A dependência de pacotes pode gerar um trabalho árduo e ainda, se uma library for removida, os pacotes que dependem dela irão parar de funcionar. Por isso, gerenciar bem as dependências é uma tarefa importantíssima no mundo Linux e essa tarefa cabia à você, até a chegada do APT, que veremos mais adiante.

Alguns pacotes ao serem instalados exibem telas de configuração onde o usuário seta várias opções para o funcionamento do pacote em questão. Todavia, depois que a configuração é finalizada você pode desejar voltar nela para mudar algo. Para isso existe um variante do DPKG chamado de *dkpg-reconfigure* e sua sintaxe é:

 # dpkg-reconfigure <nomedopacote>

Lembre-se sempre que ele existe!

5.2 RPM (Red Hat Package Manager)

Este é o gerenciador de pacotes nativo do Red Hat mas foi adotado em muitas distribuições como Slackware e SUSE.

Sua sintaxe é:

 # rpm <opções> nomedopacote.rpm OU nomedopacote

As opções do RPM mais comuns são:

-i pacote – instala um pacote RPM.
-qa – mostra os pacotes instalados.
-e pacote – remove um pacote.
-F pacote – atualiza um pacote instalado.
-U pacote – atualiza um pacote instalado. Se este não estiver instalado, instala-o.

O exemplo abaixo instala o pacote do Java no sistema:

```
turion64:/home/fernando/pacotes # rpm -iv jre-6u6-linux-amd64.rpm
Preparing packages for installation...
jre-1.6.0_06-fcs
Unpacking JAR files...
        rt.jar...
        jsse.jar...
        charsets.jar...
        localedata.jar...
```

Perceba que adicionei o parâmetro "-v" neste comando. Ele coloca o RPM em modo verbose (detalhado) e seu uso é encorajado em todas as operações e não só na instalação de pacotes.

5.3 APT (Advanced Package Tool)

Esta é a "menina dos olhos" no que diz respeito ao gerenciamento de pacotes no Debian. Ele é um front-end para o DPKG mas também tem funções próprias. As principais facilidades que ele oferece são:

- ✔ O APT se conecta aos servidores de sua distribuição, baixa o pacote para você e o instala. Isso tudo com uma linha de comando. Isso elimina o trabalho de procurar um pacote .deb na internet, baixar e instalar via DPKG.

- ✔ Resolve as dependências automaticamente. Assim, no caso da instalação do vi apresentada quando falamos do DPKG, a *libdl.so.2* seria instalada automaticamente ao instalar o vi, assim como todas as outras dependências.

- ✔ Mantém uma lista local com todos os pacotes disponíveis para download que constam nos repositórios de sua distribuição, o que permite uma busca muito otimizada.

- ✔ Pode remover os pacotes por completo, sem comprometer a estabilidade do sistema, uma

vez que as dependências são gerenciadas.

Com todos esses recursos, só nos resta ver como o APT faz tudo isso. Para começar, em um sistema que tenha o APT instalado, existe um arquivo */etc/apt/sources.list*. Este é o arquivo de fontes, onde ficam os endereços e opções dos repositórios (sites HTTP e FTP que mantêm os pacotes de sua distribuição para download). Um exemplo de um *sources.list* segue abaixo:

```
ortserver:~# cat /etc/apt/sources.list
deb http://ftp.debian.org etch main contrib
deb http://ftp.debian.org etch main contrib non-free
deb http://security.debian.org/ etch/updates main contrib
#deb-src http://security.debian.org/ etch/updates main contrib
```

A sintaxe geral do APT é:

 # apt-get/cache <ação> <nomedopacote>

Segue a lista de comandos mais usados:

apt-get update – Atualiza a lista local de pacotes disponíveis. Este comando lê o seu sources.list em busca de endereços de repositórios de pacotes. Então ele baixa um arquivo compactado que contém uma lista em texto com os nomes, versões e descrição de todos os pacotes que cada repositório possui, para armazenar em cache local.

apt-get upgrade – Atualiza os pacotes instalados no sistema, com base no seu cache local. Isso quer fizer que se um pacote presente no seu sistema está numa versão anterior ao mesmo pacote listado no cache local, este comando irá baixar o novo pacote e instalá-lo, substituindo o antigo.

apt-get dist-upgrade – Atualiza o sistema em si, instalando novas versões de kernel (se disponíveis) e outros componentes críticos.

apt-get install <pacote> - Instala um pacote e todas as suas dependências.

apt-get remove <pacote> - Remove um pacote instalado.

apt-get remove --purge <pacote> - Remove um pacote instalado e todos os seus arquivos de configuração.

apt-cache search <expressão> - procura por um pacote onde o nome ou a descrição case com a expressão informada. Por exemplo, para procurar programas que editem imagens você pode usar *$ apt-cache search "image editor"*. Veja o exemplo de uma busca pela palavra "wireshark":

```
ortserver:~# apt-cache search wireshark
ethereal - dummy upgrade package for ethereal -> wireshark
ethereal-common - dummy upgrade package for ethereal -> wireshark
ethereal-dev - dummy upgrade package for ethereal -> wireshark
tethereal - dummy upgrade package for ethereal -> wireshark
tshark - network traffic analyzer (console)
wireshark - network traffic analyzer
wireshark-common - network traffic analyser (common files)
wireshark-dev - network traffic analyser (development tools)
```

apt-cache show <pacote> - Exibe informações sobre o pacote em questão.

apt-cache depends <pacote> - Exibe as dependências do pacote.

Sempre atualize seu cache local de pacotes com o comando "apt-get update" antes de instalar um novo pacote para certificar-se de que está usando a última versão disponível no repositório de sua distribuição.

5.4 YUM (YellowDog Updater Modified)

Este gerenciador de pacotes também gerencia as dependências automaticamente e seu funcionamento é bem similar ao do APT. O motivo de eu citá-lo nesta apostila é que ele é largamente utilizado pelo Fedora Core.

Sua sintaxe é:

 # yum <ação> <pacote>

Ações:

list – Mostra os pacotes disponíveis nos repositórios.
search <expressão> - procura por um pacote onde o nome ou a descrição case com a expressão informada.
install <pacote> - instala um pacote.
update – Atualiza o sistema.
info <pacote> - Exibe detalhes sobre o pacote.

5.4 Tarballs

Um tarball é um arquivo compactado e empacotado que contém um código-fonte de um programa, um *Makefile* e alguma documentação. Uma das vantagens de se distribuir um software neste formato é que ele pode ser utilizado em qualquer distribuição (diferente de um pacote DEB ou RPM, que necessita ser usado na distribuição correta para o qual foi feito). Essa vantagem existe porque o código-fonte do software está no tarball. A desavantagem prática, logicamente, é ter de compilar este código para começar a usar o programa.

Muitos administradores de sistema preferem compilar todos os softwares de seus sistemas para obter um melhor desempenho e remover ou adicionar opções e alterações no código-fonte antes da compilação.

Geralmente a extensão do tarball (apesar de o Linux ignorar extensões) é .tar.gz ou .tgz.

A extensão é usada para dizer ao usuário que é um arquivo empacotado com o TAR e compactado com o GZip. É uma "estética útil".

Para descompactar e desempacotar um tarball, geralmente usamos o próprio programa *tar*, da seguinte maneira:

$ tar -xvzf <tarball.tar.gz>
Este comando vai criar um diretório com o mesmo nome do tarball, que conterá os arquivos descompactados e no decorrer da descompactação, o caminho completo de cada

arquivo criado é exibido na tela.

Abaixo um exemplo de descompactação de um tarball:

```
turion64:/home/fernando/pacotes # tar -xvzf iptraf-3.0.0.tar.gz
iptraf-3.0.0/
iptraf-3.0.0/CHANGES
iptraf-3.0.0/Documentation/
iptraf-3.0.0/Documentation/README
iptraf-3.0.0/Documentation/iptraf.8
iptraf-3.0.0/Documentation/rvnamed.8
iptraf-3.0.0/Documentation/iptraf-configmenu.png
iptraf-3.0.0/Documentation/iptraf-dstat1.png
iptraf-3.0.0/Documentation/iptraf-editfilter.png
iptraf-3.0.0/Documentation/iptraf-filtermenu.png
```

Ao final da descompactação, você terá todo o código-fonte do programa e poderá compilá-lo para uso. No entanto, antes disto, você deve ter instalados o compilador necessário e todas as dependências que este código-fonte exige. Esta é uma tarefa minuciosa, que não abordaremos neste curso básico, mas saiba que é possível. ;o)

Capítulo 6 – Controle de acesso

Este é um assunto prazeroso do mundo Linux. É graças às suas permissões que vírus em ambientes Linux são inviáveis, que a segurança é reforçada e que a estabilidade do sistema é garantida.

Cada arquivo e diretório possui um conjunto de permissões (leitura, escrita e execução) que controlam o seu acesso a partir de três fontes (dono, grupo e outros). Entender esses conceitos é essencial.

6.1 Usuários e grupos

Os usuários do Linux são armazenados, por padrão, no arquivo */etc/passwd* enquanto os grupos, em */etc/group*.

Um trecho exemplo de um arquivo *passwd* é mostrado abaixo.

```
fernando@turion64:~> tail /etc/passwd
ntp:x:74:101:NTP daemon:/var/lib/ntp:/bin/false
polkituser:x:103:105:PolicyKit:/var/run/PolicyKit:/bin/false
postfix:x:51:51:Postfix Daemon:/var/spool/postfix:/bin/false
root:x:0:0:root:/root:/bin/bash
sshd:x:71:65:SSH daemon:/var/lib/sshd:/bin/false
suse-ncc:x:105:107:Novell Customer Center User:/var/lib/YaST2/suse-ncc-fakeho
me:/bin/bash
uucp:x:10:14:Unix-to-Unix CoPy system:/etc/uucp:/bin/bash
uuidd:x:101:103:User for uuidd:/var/run/uuidd:/bin/false
wwwrun:x:30:8:WWW daemon apache:/var/lib/wwwrun:/bin/false
fernando:x:1000:100:Fernando Pinheiro:/home/fernando:/bin/bash
```

Para entender a estrutura desse arquivo, precisamos saber o que cada valor separados por dois pontos significa. Analise abaixo:

login **:** senha **:** UID **:** GID **:** Informações diversas **:** diretório inicial **:** shell inicial

O campo **login**, como o nome sugere, é o nome que o usuário digita para entrar no sistema. O campo **senha** segue a mesma lógica, mas obviamente existe uma criptografia para proteção da senha do usuário. **UID** (User IDentifier) e **GID** (Group IDentifier) são números que o sistema usa para identificar os usuários e grupos. Onde está **Informações diversas**, podem aparecer informações que foram atribuídas ao usuário no momento de sua criação, como o nome completo, números de telefone, etc. O **diretório inicial** define o caminho onde o usuário estará quando fizer login (geralmente é seu diretório pessoal em /home). E por fim, o **shell inicial** define qual shell o usuário vai receber, por padrão, logo após logar-se no sistema.

Você deve ter percebido que alguns usuários têm um shell para login definido como /bin/false. Este é um shell especial, que impede qualquer login nele. Isso significa que o usuário que estiver com este shell padrão setado não conseguirá logar no sistema.

Os comandos mais comuns para se gerenciar usuários e grupos são:

adduser <usuário>
Adiciona usuários ao sistema.

useradd <usuário>
Similar ao adduser, mas não pede senha para o usuário, nem cria diretório home para o mesmo.

userdel <usuário>
Remove um usuário.

passwd <usuário>
Altera a senha de um usuário. O próprio usuário pode usar este comando, se o administrador do sistema permitir, para alterar sua própria senha.

su <usuário>
Troca seu usuário atual para o usuário especificado. É necessário saber a senha do usuário de destino.

sudo
Qualquer comando precedido deste é executado como root. Para funcionar, o usuário atual deve estar configurado no arquivo */etc/sudoers*.

w
Mostra quais usuários estão logados no sistema.

addgroup <grupo>
Adiciona um grupo vazio ao sistema.

groupdel <grupo>
Remove um grupo mas não remove os usurários que pertencem a ele.

6.2 Arquvos e diretórios

Depois de conhecer sobre os usuários e grupos, agora estudaremos efetivamente as permissões de arquivos. Cada arquivo ou diretório possui três conjuntos de permissões que relacionam-se ao dono, grupo e outros, respectivamente. Esses conjuntos são constituídos de três permissões que são a de ler, de escrever e de executar e podem ser combinadas. Para entender melhor, acompanhe o exemplo abaixo:

```
fernando@turion64:~/scripts> ls -l
total 1104
-rwxr-xr-x 1 fernando users   34372 2008-06-23 22:25 arping-2.06.tar.gz
-rw-r--r-- 1 fernando users       3 2008-07-13 21:19 arq
-rw-r--r-- 1 fernando users      24 2008-07-12 21:20 arq1
-rw-r--r-- 1 fernando users      28 2008-07-12 21:20 arq2
-rw-r--r-- 1 fernando users      66 2008-08-29 19:19 first.c
-rw-r--r-- 1 fernando users     311 2008-07-12 21:29 hextext
-rwxr-xr-x 1 fernando users 1054249 2008-06-23 22:22 ht-2.0.14.tar.gz
-rwxr-xr-x 1 fernando users     102 2008-07-29 20:20 perlprint.pl
-rw-r--r-- 1 fernando users      85 2008-07-12 20:59 texto
-rw-r--r-- 1 fernando users    1020 2008-07-12 21:38 texto2
```

Concentre-se na linha referente ao arquivo *first.c*. A coluna de permissões é a primeira e mostra o seguinte conjunto de caracteres: -rw-r--r--. Essas são as **permissões** do arquivo e esses caracteres dizem tudo que precisamos saber sobre elas. Depois temos o nome do **dono** do arquivo, que neste caso é "fernando". Logo após, o nome do **grupo** deste arquivo, que neste caso é "users".

O primeiro conjunto de três caracteres depois do hífen inicial representa as permissões do dono do arquivo. Os próximos três caracteres representam as permissões do grupo e os últimas três, dos outros usuários.

A permissão "r" significa leitura (read).
A permissão "w" significa escrita (write).
A permissão "x" significa execução (execute).

A ausência de alguma permissão é indicada por um hífen (-).

Assim, as permissões deste arquivo podem ser representadas na tabela abaixo:

Dono	Grupo	Outros
rw- (lê e escreve)	r-- (só lê)	r-- (só lê)

O formato é sempre rwx, nesta ordem. Desta forma, volte à imagem e analise as permissões do arquivo *arping-2.06.tar.gz*. Você deve ser capaz de interpretá-las.

Uma outra forma de representação das permissões de um arquivo é usando a notação

octal (que vai de 0 à 7). Para esta representação, as permissões r, w e x possuem valores correspondentes, da segunda maneira:

r	w	x
4	2	1

Para representar as permissões neste formado, somamos os valores correspondentes. Por exemplo, no arquivo *first.c* que foi usado no exemplo anterior, as permissões são rw (dono), r (grupo) e r (outros). Se somarmos estes valores separadamente para dono, grupo e outros temos:

r + w = 2 + 4 = 6
r = 4
r = 4

Agora concatenamos os resultados, o que nos dá o número 644. Portanto, podemos dizer que as permissões deste arquivo são 644. Para treinar, obtenha a permissão em octal do arquivo *arping-2.06.tar.gz*.

O comando que muda as permissões de um arquivo ou diretório é o *chmod*:

$ chmod <permissões> arquivo

Por exemplo:

$ chmod 440 meuarquivo

O comando acima definirá que o dono do arquivo pode somente ler (4), assim como o dono. Já os outros, não podem nada (0).

Você também pode alterar as permissões de um diretório e todo o seu conteúdo com a opção -R:

 # chmod -R 763 scripts/

O comando acima faz com que todos os arquivos do diretório scripts e seus subdiretórios recebam as permissões abaixo:

O número 7 define leitura (4), escrita (2) e execução (1) para o dono.
O número 6 define leitura (4) e escrita (2) para o grupo.
O número 3 define escrita (2) e execução (1) para outros.

A última permissão (3 ou -wx) nos coloca numa situação interessante. Os outros podem escrever e executar os arquivos mas não podem lê-los. Será que isso dá certo? Bom, na verdade dá certo sim. Deste jeito, outros poderão apagar arquivos mas sem ver seu conteúdo e não poderão executá-lo porque a execução precisa de permissão de leitura.

Acostume-se com as permissões no formato octal pois são largamente utilizadas e depois que você se acostumar, ficará fácil utilizá-las.

Há ainda comandos para se alterar o dono e o grupo dos arquivos e diretórios que são o *chown* e o *chgrp*, respectivamente.

Capítulo 7 - Montagem e formatação de mídias

Toda mídia de armazenamento precisa ser montada no sistema operacional para ser utilizada. Montar significa atribuir um ponto de montagem (um diretório) para que os dados da mídia possam ser acessados através dele. Assim, se você criar um diretório chamado /minha_partição e montar a sua segunda partição nele, basta entrar neste diretório para ver os dados da sua partição.

A montagem é feita com o comando *mount*. Sua sintaxe é:

 # mount -t <filesystem> <mídia> <ponto_de_montagem>

7.1 Partições de disco

Para montar uma partição /dev/sdb1 FAT-32 no diretório /mnt/sdb1, fazemos:

 # mkdir /mnt/sdb1
 # mount -t vfat /dev/sdb1 /mnt/sdb1

O primeiro comando só criou o diretório enquanto o segundo realizou a montagem no diretório criado. Agora sua partição deve estar acessível neste diretório.

Algumas opções de filesystems para o parâmetro -t são vfat, reiserfs, ext3, ntfs, iso9660, dentre outras. Comande **# man mount** para ver uma lista completa.

NOTA: Você pode usar o comando man precedendo qualquer comando que tiver dúvida para ler seu manual.

Inversamente, temos um comando para desmontagem de mídias que é o **umount**:

 # umount /dev/sdb1

O comando acima desmontará a partição */dev/sdb1* (independente do ponto onde ela está montada).

7.2 Disquetes

Os disquetes são formatados no sistemas de arquivos FAT-12, oriundo do MS-DOS e sua sintaxe para montagem é:

 # mount -t msdos /dev/fd0 /media/floppy

No caso acima, montei o primeiro drive de disquete (/dev/fd0) no diretório /media/floppy. Se este diretório não existir, o comando mount retornará um erro e a montagem não será realizada.

A formatação física de disquetes pode ser feita como comando **fdformat**, assim:

 # fdformat /dev/fd0

Você pode instalar o pacote *dosfstools* para ter disponíveis os comandos *mkfs.vfat* e *mkfs.msdos*, que também formatam disquetes de maneira lógica.

7.3 CDs/DVDs

Para montar CD-ROM ou DVD-ROM, você pode usar o atalho /media/cdrom direto:

 # mount /media/cdrom

Se preferir usar a sintaxe geral do comando *mount*, use o filesystem que este tipo de mídia utiliza, chamado ISO9660, deste jeito:

 # mount -t iso9660 /dev/cdrom /media/cdrom

7.4 Mídias USB

Conforme citado no início do capítulo, as mídias de armazenamento USB são reconhecidas pelo sistema da mesma forma que os discos SATA/SCSI. Portanto, seu formato será /dev/sdxn, onde "x" é uma letra após o seu último disco SATA/SCSI (se não houver disco deste tipo, será a letra "a") e "n" é o número da partição do seu pen-drive, que geralmente é 1. Logo, se houver dois HDs SATA na máquina, o acesso ao pen-drive será via /dev/sdc1.

O filesystem usado na opção -t do *mount* vai depender do tipo de partição do pen-drive, assim como nas partições de HDs. Por padrão, estes dispositivos vêm de fábrica com uma partição FAT-32 por questões de compatibilidade com SOs diferentes, utilizando todo o seu espaço disponível e não vejo razão para alterar isso. Neste caso a montagem será:

 # mount -t vfat /dev/sdc1 /media/pendrive

Logicamente, o diretório */media/pendrive* deve existir.

Capítulo 8 - Configuração de rede

O Linux é muito poderoso quando está em rede. Suas ferramentas de rede nos ajudam a diagnosticar problemas em ambientes mistos ou puramente de outros sistemas operacionais.

O Linux reconhece as placas de rede ethernet como eth0, eth1 e assim por diante. Já as sem-fio, são chamadas de ath0 (ou wlan0) e assim sucessivamente.

8.1 Comandos de configuração

Para checar suas configurações de rede o comando ideal é o *ifconfig*. Apesar de sua semelhança em nome com o ipconfig dos sistemas Windows, o ifconfig é capaz de alterar as configurações de IP da placa, desabilitá-la, adicionar mais de um IP na mesma placa (alias de IP), alterar de maneira lógica seu endereço MAC, dentre outras poderosas funções. Alguns exemplos de utilização são exibidos abaixo:

```
turion64:~ # ifconfig eth0 192.168.0.2/24
turion64:~ # ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:1B:38:1C:42:85
          inet addr:192.168.0.2  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:252 Base address:0xe000

turion64:~ # ifconfig eth0 192.168.0.2 netmask 255.0.0.0
turion64:~ # ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:1B:38:1C:42:85
          inet addr:192.168.0.2  Bcast:192.255.255.255  Mask:255.0.0.0
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:252 Base address:0xe000
```

No exemplo acima, primeiro configurei a interface eth0 com o IP 192.168.0.2 e máscara de 24 bits (255.255.255.0), usando o comando *# ifconfig eth0 192.168.0.2/24*. Depois, listei a configuração da interface eth0 para certificar-me de que está com o IP que coloquei anteriormente.

Mais abaixo, configurei a eth0 com o IP 192.168.0.2 e máscara de 8 bits (255.0.0.0), utilizando comando *# ifconfig eth0 192.168.0.2 netmask 255.0.0.0*. Novamente listei a configuração da eth0 para ver o resultado.

Note também que outras informações tais como HWaddr (endereço MAC) e Bcast (endereço de broadcast) também são listadas com o comando *ifconfig*.

Um outro comando muito utilizado é o *dhclient*, que atualiza os endereços obtidos de um

servidor DHCP (Dynamic Host Control Protocol) presente na rede. Se comandado sem parâmetros, o *dhclient* vai requisitar endereços para todas as interfaces de rede configuradas como DHCP. Sua sintaxe para renovar o IP de uma interface específica é:

dhcpclient <interface>

Algumas distribuições não usam o comando dhclient. Neste caso você pode usar os comandos pump ou dhcpcd.

8.2 Arquivos de configuração

Ao alterar o IP ou outra configuração com o *ifconfig*, tais alterações só serão válidas para a sessão atual do usuário, ou seja, se você deslogar e relogar, suas alterações serão perdidas. Para fazer alterações de forma fixa, é necessário escrever num arquivo. Seu nome vai depender da distribuição utilizada.

No Debian e seus derivados (Ubuntu, Kurumin, Knoppix, etc), o arquivo que mantém essas configurações é o */etc/network/interfaces*. Segue um exemplo praticamente auto-explicativo, abaixo:

```
ortserver:~# cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eth0
iface eth0 inet static
        address 10.0.0.1
        netmask 255.255.255.0
        network 10.0.0.0
        broadcast 10.0.0.255
        # dns-* options are implemented by the resolvconf package, if installed
        dns-search ort.local
auto eth1
iface eth1 inet dhcp
```

Já em sistemas baseados no Red Hat, os arquivos ficam separados. Existe um arquivo para

cada interface de rede, todos no diretório */etc/sysconfig/network/*. No caso da eth0, será o */etc/sysconfig/network/ifcfg-eth0*. Segue um exemplo:

```
turion64:~ # cat /etc/sysconfig/network/ifcfg-eth0
BOOTPROTO='dhcp'
BROADCAST=''
ETHTOOL_OPTIONS=''
IPADDR='10.0.0.2'
MTU=''
NAME='RTL8101E PCI Express Fast Ethernet controller'
NETMASK='255.0.0.0'
GATEWAY='10.0.0.1'
NETWORK=''
REMOTE_IPADDR=''
STARTMODE='auto'
USERCONTROL='no'
```

Comandos tradicionais de rede como *ping*, *route*, *netstat* e *nslookup* também estão presentes.

Capítulo 9 - Ambiente gráfico

Em nosso último capítulo, abordaremos o ambiente gráfico, essencialmente necessário quando se trata do uso doméstico do Linux.

O ambiente gráfico do Linux é uma combinação entre um servidor, um gerenciador de sessões e um gerenciador (ou cliente) gráfico. Para utilizar um ambiente gráfico é necessário instalar todos esses componentes.

9.1 Servidores X

O servidor gráfico é o software responsável por controlar o hardware de monitor, placa de vídeo, mouse, teclado e todos os dispositivos básicos de uso num ambiente gráfico. Configurações mais avançadas como freqüência de vibração horizontal, taxa de repetição de teclado ou resoluções de ponteiros de mouse também são de sua responsabilidade.

Os servidores gráficos mais conhecidos são o X Window e o X.Org. Este segundo é uma variação do primeiro, mas eles são quase idênticos. Visualmente, eles não passam de uma tela cinza, sem ícones ou janelas. Essas funções são providas pelos gerenciadores gráficos, que veremos adiante.

9.2 Gerenciadores gráficos

Os gerenciadores gráficos mais famosos são o Gnome e o KDE. Ambos rodam sobre o X Window ou X.Org. Mas antes que os gerenciadores gráficos sejam efetivamente inicializados, é necessário um gerenciador de sessão, que pode ser o XDM, o KDM, dentre outros. Em resumo, um ambiente gráfico no Linux é formado por: Servidor gráfico + Gerenciador de sessão + Gerenciador gráfico.

Em sistemas Debian-based, podemos instalar o X Window com o seguinte comando:

 # apt-get install x-window-system

Para instalar o gerenciador gráfico (o que inclui um gerenciador de sessão), use:

 # apt-get install xfce4

Este comando instalará o XFCE, um gerenciador muito mais leve que o Gnome ou KDE, mas com menos recursos.

Feita a instalação e configuração, inicie o X com:

$ startx

Referências

Sistemas Operacionais Modernos - Andrew S. Tanenbaum
Descobrindo o Linux, NOVATEC – Eriberto Mota Filho
Linux in a Nutshell - O'Reilly