Guia de Referência

Mandriva Linux 2006

http://www.mandriva.com

Guia de Referência: Mandriva Linux 2006
Publicado Setembro 2005

por NeoDoc (`http://www.neodoc.biz`)Camille Bégnis, Christian Roy, Fabian Mandelbaum, Roberto Rosselli del Turco, Marco De Vitis, Alice Lafox, John Rye, Wolfgang Bornath, Funda Wang, Patricia Pichardo Bégnis, Debora Rejnharc Mandelbaum, Mickael Scherer, Jean-Michel Dault, Lunas Moon, Céline Harrand, Fred Lepied, Pascal Rigaux, Thierry Vignaud, Giuseppe Ghibò, Stew Benedict, Francine Suzon, Indrek Madedog Triipus, Nicolas Berdugo, Thorsten Kamp, Fabrice Facorat, Xiao Ming, Snature , Guylhem Aznar, Pavel Maryanov, Annie Tétrault, Aurelio Marinho Jargas, Felipe Arruda, Marcia Gawlak Hoshi, Bob Rye, Jean-Luc Borie, e Roberto Patriarca

Aviso Legal

Sobre a Elaboração deste Manual

Este manual é escrito e mantido pela NeoDoc (`http://www.neodoc.biz`). Traduções são realizadas pela NeoDoc, Mandriva e outros tradutores.
Este documento foi escrito em DocBook XML. O conjunto de arquivos envolvido foi gerenciado com o uso do Sistema de Criação de Conteúdo Colaborativo Borges (`http://sourceforge.net/projects/borges-dms`). Os arquivos-fonte XML foram processados pelo xsltproc, e `jadetex` (para a versão eletrônica) utilizando uma versão customizada das folhas de estilo de Norman Walsh. As imagens de captura de tela foram realizadas com o `xwd` ou GIMP e convertidas com o `convert` (do pacote ImageMagick). Todos estes programas são software livre e estão todos disponíveis na sua distribuição Mandriva Linux.

Índice

Prefácio 1
1. Sobre o Mandriva Linux 1
1.1. Entrando em Contato com a Comunidade Mandriva Linux 1
1.2. Junte-se ao Clube! 1
1.3. Assinando o Mandriva Online 2
1.4. Comprando Produtos Mandriva 2
1.5. Contribuindo com o Mandriva Linux 2
2. Sobre este Guia de Referência 2
3. Nota do Editor 3
4. Convenções utilizadas neste livro 4
4.1. Convenções Tipográficas 4
4.2. Convenções Gerais 5
I. O Sistema Linux 7
1. Conceitos Básicos de Sistemas UNIX® 7
1.1. Usuários e Grupos 7
1.2. Arquivos 9
1.3. Processos 11
1.4. Uma Breve Introdução à Linha de Comando 11
2. Discos e Partições 17
2.1. Estrutura de um Disco Rígido 17
2.2. Convenções para nomenclatura de Discos e Partições 19
3. Organização de Diretórios e Arquivos 21
3.1. Dados Compartilhados/Não Compartilhados, Estáticos/Dinâmicos 21
3.2. O Diretório Raiz: / 21
3.3. /usr: O Enorme 22
3.4. /var: Dados que Podem Mudar Durante o Uso 22
3.5. /etc: Arquivos de Configuração 23
4. O Sistema de Arquivo do Linux 25
4.1. Comparando Alguns Sistemas de Arquivo 25
4.2. Tudo é um Arquivo 27
4.3. Links 28
4.4. Pipes “Anônimos” e Pipes Nomeados 29
4.5. Arquivos Especiais: Arquivos de Caractere e de Bloco 31
4.6. Links Simbólicos, Limitação de “Hard” Links 31
4.7. Atributos de Arquivo 32
5. O Sistema de Arquivos /proc 35
5.1. Informação Sobre Processos 35
5.2. Informação sobre Hardware 36
5.3. Exiba e altere parâmetros do kernel 39
II. Linux em Profundidade 41
6. Sistemas de Arquivo e Pontos de Montagem 41
6.1. Princípios 41
6.2. Particionando um Disco Rígido, Formatando uma Partição 43
6.3. Os Comandos mount e umount 43
7. Introdução a Linha de Comando 47
7.1. Utilitários para Tratamento de Arquivos 47
7.2. Tratando Atributos de Arquivo 49
7.3. Casamento de Padrões no Shell 51
7.4. Redirecionamentos e Pipes 52
7.5. Auto Completar na Linha de Comando 53
7.6. Iniciando e Tratando Processo em Segundo Plano: Controle de Trabalhos 54
7.7. Algumas Palavras ao Final 55
8. Edição de Textos: Emacs e VI 57
8.1. Emacs 57
8.2. Vi: o ancestral 60
8.3. Uma última palavra... 64
9. Utilitários na Linha de Comando 65

9.1. Operações com Arquivo e Filtragem 65
9.2. find: Encontrando Arquivos 70
9.3. Agendamento de Inicialização de Comandos 72
9.4. Arquivamento e Compactação de Dados 74
9.5. Muito, Muito Mais... 76
10. Controle de Processo 77
10.1. Mais Sobre os Processos 77
10.2. Informação sobre Processos: ps e pstree 77
10.3. Envando Sinais para Processos: kill, killall e top 78
10.4. Configurando Prioridade para Processos: nice, renice 79
11. Os Arquivos de Inicialização: init sysv 81
11.1. Introdução 81
11.2. Níveis de Execução 81
12. Acesso Remoto Seguro 85
12.1. Configuração do Servidor SSH 85
12.2. Configuração do Cliente SSH 85
12.3. Copiando Arquivos Entre um Sistema Remoto 86
13. Gerenciamento de Pacotes na Linha de Comando 87
13.1. Instalando e Removendo Pacotes 87
13.2. Gerenciamento de Mídias 87
13.3. Dicas e Truques 88
A. Glossário 91
Índice Remissivo 109

Lista de Tabelas

4-1. Características dos Sistemas de Arquivo 26

Prefácio

1. Sobre o Mandriva Linux

O Mandriva Linux é uma distribuição GNU/Linux produzida pela Mandriva S.A., que surgiu na internet em 1998 com o objetivo primário de fornecer um GNU/Linux amigável. Os dois pilares da Mandriva são trabalho de código aberto e colaborativo.

Em 7 de abril de 2005 a Mandrakesoft mudou o seu nome para Mandriva para refletir a fusão com a empresa Conectiva, com sede no Brasil. Seu produto principal, o Mandrakelinux, tornou-se o Mandriva Linux.

1.1. Entrando em Contato com a Comunidade Mandriva Linux

Existem vários locais na internet onde você pode buscar por informações sobre o Mandriva Linux. Se você deseja saber mais sobre a empresa Mandriva, acesse o nosso web site (`http://www.mandriva.com/`). Você também pode verificar o web site da distribuição Mandriva Linux (`http://www.mandrivalinux.com/`) e seus derivados.

A Mandriva Expert (`http://www.mandrivaexpert.com/`) é a plataforma de suporte da Mandriva. Ela oferece uma nova experiência baseada na confiança e no prazer de recompensar os outros por suas contribuições.

Nós também convidamos você a participar de várias listas de discussão (`http://www.mandriva.com/community/resources/newsgroups`) onde a comunidade Mandriva Linux demonstra a sua vivacidade e sabedoria.

Lembre-se também de conectar à nossa página de segurança (`http://www.mandriva.com/security`). Esta página reúne todo o material relacionado a segurança sobre as distribuições Mandriva Linux. Você irá encontrar avisos de erros e falhas de segurança, assim como procedimentos para a atualização do kernel, uma lista de discussão e o Mandriva Online (`https://online.mandriva.com/`). Esta página é essencial para qualquer administrador de servidores ou usuário preocupado com segurança.

1.2. Junte-se ao Clube!

A Mandriva oferece uma grande variedade de vantagens através do Mandriva Club (`http://club.mandriva.com`):

- download de software comercial normalmente disponível apenas em pacotes no varejo, como drivers para equipamentos especiais, aplicações comerciais, freeware e versões de demonstração;
- vote e sugira novos softwares através do nosso sistema de votação;
- acesse mais de 50,000 pacotes RPM para todas as distribuições Mandriva Linux
- ganhe descontos para produtos e serviços na Mandriva Store (`http://store.mandriva.com`);
- tenha acesso a uma lista de mirrors exclusiva para membros do clube;
- leia artigos e participe de fóruns em diversos idiomas.
- acesse a Base de Conhecimento (`http://club.mandriva.com/xwiki/bin/view/KB/`) da Mandriva, um web site do tipo wiki que hospeda documentação sobre diversos assuntos, como administração, conectividade, resolução de problemas e mais;
- converse com os desenvolvedores da Mandriva Linux no Club Chat (`https://www.mandrivaclub.com/user.php?op=clubchat`);
- aumente os seus conhecimentos sobre GNU/Linux através das lições de treinamento à distância da Mandriva (`http://etraining.mandriva.com/`).

Participando da Mandriva através do Mandriva Club você estará contribuindo diretamente para o aperfeiçoamento da distribuição Mandriva Linux e nos ajudando a oferecer o melhor desktop GNU/Linux para os nossos usuários.

Uma nova versão do Mandriva Club está a caminho. Você pode vê-la através da nova URL (`http://club-beta.mandriva.com/`). Uma nova Base de Conhecimento (`http://club-beta.mandriva.com/xwiki/bin/view/KB/`) também está sendo repensada.

1.3. Assinando o Mandriva Online

A Mandriva oferece uma maneira muito conveniente de manter o seu sistema automaticamente atualizado, mantendo bugs afastados e corrigindo falhas de segurança. Visite o web site Mandriva Online (`https://online.mandriva.com//`) para aprender mais sobre este serviço.

1.4. Comprando Produtos Mandriva

Os usuários do Mandriva Linux podem comprar produtos on-line através da Mandriva Store (`http://store.mandriva.com/`). Você não irá apenas encontrar software Mandriva Linux, sistemas operacionais e CDs "live" (como o Move), mas também ofertas especiais para assinantes, suporte, softwares e licenças de terceiros, documentação, livros relacionados a GNU/Linux e outros produtos da Mandriva.

1.5. Contribuindo com o Mandriva Linux

As habilidades de muitas pessoas talentosas que utilizam o Mandriva Linux podem ser muito importantes no desenvolvimento do sistema Mandriva Linux:

- **Empacotamento.** Um sistema GNU/Linux é feito principalmente de programas recolhidos da internet. Eles têm que ser empacotados para poderem funcionar juntos.
- **Programação.** Existem vários, vários projetos diretamente apoiados pela Mandriva: encontre o que mais lhe agrada e ofereça a sua ajuda para o(s) desenvolvedor(es) principais.
- **Internacionalização.** Você pode nos ajudar a traduzir páginas web, programas e documentação.

Consulte a página dos projetos de desenvolvimento (`http://qa.mandriva.com/`) para saber como você pode contribuir para a evolução da Mandriva Linux.

2. Sobre este Guia de Referência

Este *Guia de Referência* é direcionado para as pessoas que desejam entender melhor o sistema Mandriva Linux, e para quem quer tirar uma vantagem maior de suas grandes capacidades. Depois de ler este manual, nós esperamos que você esteja com mais facilidade em relação à administração diária de seu computador com GNU/Linux. Aqui está uma visão geral de seus dois componentes, junto com uma breve descrição de cada capítulo que ele contém:

- Na primeira parte, (*O Sistema Linux*), nós apresentamos o sistema GNU/Linux para você. Discutimos a sua arquitetura, os principais sistemas de arquivo disponíveis e alguns aspectos peculiares como o sistema de arquivos `/proc`.

  No primeiro capítulo (Capítulo 1) nós fazemos uma apresentação do paradigma UNIX® quando falamos mais especificamente do mundo GNU/Linux. Nós examinamos os utilitários de manipulação de arquivo assim como algumas funcionalidades úteis oferecidas pelo shell. Então temos um capítulo complementar (Capítulo 2) onde explicamos como os discos rígidos são gerenciados no GNU/Linux. Também trabalhamos com particionamento do disco rígido.

  Nós exploramos a organização da árvore de arquivos no Capítulo 3. Os sistemas UNIX® tendem a crescer muito, mas cada arquivo possui o seu lugar em diretórios específicos. Após ler este capítulo, você saberá onde procurar por arquivos de acordo com a função deles no sistema.

  O próximo capítulo fala de sistemas de arquivo (Capítulo 4). Após apresentar os sistemas de arquivo disponíveis, nós dicutimos os tipos de arquivo e alguns conceitos e funcionalidades adicionais como inodes e

pipes. O capítulo seguinte (Capítulo 5) apresenta um sistema de arquivos especial (e virtual) do GNU/Linux chamado `/proc`.

- A segunda parte (*Linux em Profundidade*) trata de tópicos mais práticos. Nós falamos sobre a relação entre sistemas de arquivos e pontos de montagem, como utilizar a linha de comando em suas tarefas diárias, como editar arquivos de configuração com editores leves e poderosos, e mais.

  Nós cobrimos os tópicos sobre ***sistemas de arquivo*** e ***pontos de montagens*** (Capítulo 6) definindo ambos os termos, bem como explicando-os com exemplos reais.

  Então trabalhamos com a interface de linha de comando (Capítulo 7). Nós discutimos utilitários de manipulação de arquivos, como os comandos `mkdir` e `touch`, e como mover, deletar e copiar arquivos e diretórios no sistema de arquivos. Nós também falamos de atributos de arquivos e como gerenciá-los com comandos como o `chown` e o `chgrp`. Então entramos no assunto de caracteres curinga, redirecionamento e pipes, completação na linha de comado, e também controle básico de processos.

  O próximo capítulo aborda a edição de textos (Capítulo 8). Já que a maior parte dos arquivos de configuração do UNIX® são arquivos de texto, você irá querer ou precisar utilizar um ***editor de texto*** eventualmente. Você irá aprender como utilizar dois dos mais famosos editores de texto do mundo UNIX® e GNU/Linux: o poderoso Emacs, desenvolvido por Richard M. Stallman, e o bom e velho Vi, escrito em 1976 por Bill Joy.

  Você teria então que executar uma manutenção básica no seu sistema. Os próximos dois capítulos apresentam usos práticos da linha de comando (Capítulo 9), e controle de processo (Capítulo 10) em geral.

  O próximo capítulo (Capítulo 11) apresenta o procedimento de inicialização do Mandriva Linux, e como utilizá-lo eficientemente. Nós falamos sobre o init (o processo que possibilita a inicialização do seu sistema) e níveis de execução diferentes que você possa querer utilizar (especialmente para tarefas de manutenção). Nós também explicamos brevemente como utilizar o drakxservices para gerenciar serviços.

  No próximo capítulo (Capítulo 12) nós explicamos como acessar de maneira segura um sistema remoto (através do `ssh`) para executar tarefas de manutenção, executar programas, etc. Nós descrevemos uma visão geral do esquema de conexão e então uma configuração básica do cliente/servidor do `ssh`. O uso do `scp` também é discutido.

  Por fim, nós terminamos este livro com um capítulo dedicado ao gerenciamento de pacotes através da linha de comando (Capítulo 13). Nele você vai aprender a utilizar o programa urpmi e sua contraparte, o urpme. Nós também explicamos como). Nele você vai aprender a utilizar o programa urpmi e sua contraparte, o urpme. Nós também explicamos como). Nele você vai aprender a utilizar o programa urpmi e sua contraparte, o urpme. Nós também explicamos como). Nele você vai aprender a utilizar o programa urpmi e sua contraparte, o urpme. Nós também explicamos como). Nele você vai aprender a utilizar o programa urpmi e sua contraparte, o urpme. Nós também explicamos como). Nele você vai aprender a utilizar o programa urpmi e sua contraparte, o urpme. Nós também explicamos como). Nele você vai aprender a utilizar o programa urpmi e sua contraparte, o urpme. Nós também explicamos como). Nele você vai aprender a utilizar o programa urpmi e sua contraparte, o urpme. Nós também explicamos como gerenciar fontes de mídia.

3. Nota do Editor

Na filosofia do código aberto, contribuidores são muito bem-vindos! Atualizar a doumentação do Mandriva Linux é uma tarefa e tanto. Você pode ajudar de várias formas. Na verdade, a equipe de documentação está procurando constantemente por voluntários talentosos para nos ajudar com as seguintes tarefas:

- elaboração ou atualização;
- tradução;
- edição;
- programação XML/XSLT.

Se você tem bastante tempo, pode escrever ou atualizar um capítulo inteiro; se você fala outro idioma além do inglês, pode nos ajudar a traduzir os nossos manuais; se você tem idéias de como melhorar o nosso conteúdo, conte-nos; se você possui habilidade com programação e quer nos ajudar a aperfeiçar o Sistema de Criação de Conteúdo Colaborativo Borges (C3S) (`http://sourceforge.net/projects/borges-dms`), junte-se a nós. E não hesite em nos contactar se você encontrar qualquer erro na documentação, pois assim podemos corrigí-los!

Para qualquer informação sobre o projeto de documentação do Mandriva Linux, por favor entre em contato com o administrador da documentação (`mailto:documentation@mandriva.com`) ou visite a pági-

na do Projeto de Documentação do Mandriva Linux (`http://qa.mandriva.com/twiki/bin/view/Main/DocumentationTask/`).

Note que desde junho de 2004 a documentação do Mandriva Linux e o desenvolvimento do Borges é feito pela empresa NeoDoc (`http://www.neodoc.biz`).

4. Convenções utilizadas neste livro

4.1. Convenções Tipográficas

Para realmente diferenciar as palavras especiais do fluxo do texto, a equipe de documentação utiliza diferentes interpretações. A tabela a seguir demonstra um exemplo de cada palavra especial ou grupo de palavras com a sua interpretação atual e o que cada uma significa.

Exemplo Formatado	**Significado**
inode	Utilizado para enfatizar um termo técnico explicado no Apêndice A.
`ls -lta`	Utlizado para comandos e seus argumentos. (veja Seção 4.2.1).
`um_arquivo`	Utilizado para nomes de arquivo. Também poed ser encontrado em nome de pacotes RPM.
ls(1)	Referência para uma `man` page. Para ler a página, simplesmente digite `man 1 ls`, na linha de comando.
`$ ls *.pid`	Formatação usada para representações textuais do que pode estar sendo exibido na sua tela, incluindo interações com o computador, listagem de programas, e outros.
`localhost`	Informações literais que não se encaixam em nenhuma das categorias definidas anteriormente. Por exemplo, uma palavra-chave retirada de um arquivo de configuração.
OpenOffice.org	Define nomes de aplicação. Dependendo do contexto, o nome da aplicação e o do comando podem ser o mesmo, mas formatados de maneira diferente. Por exemplo, a maior parte dos comandos estão escritos com letras minúsculas, enquanto nomes de aplicações normalmente começam com letra maiúscula.
Aquivos	Indica entrada de menus ou rótulos da interface gráfica. A letra sublinhada, quando presente, informa um atalho, que pode ser acessado pressionando a tecla **Alt** mais a letra em questão.
Le petit chaperon rouge	Identifica palavras em outros idiomas.
Aviso!	Reservedo para avisos especiais, para dar ênfase à importância das palavras. Leia em voz alta.

Este ícone destaca uma nota. Geralmente, é uma remarca no contexto atual, dando informação adicional.

Representa uma dica. Ele pode ser um conselho geral de como executar uma ação específica, ou uma bela funcionalidade que poderá tornar a sua vida mais fácil, como teclas de atalho, por exemplo.

Tenha bastante cuidado ao ver este ícone. Ele sempre significa que uma informação bastante importante sobre um tópico específico que precisa ser abordado.

4.2. Convenções Gerais

4.2.1. Sinopse dos Comandos

O exemplo abaixo demonstra os símbolos que você encontrará quando o autor descreve os argumentos de um comando:

```
comando <argumento não literal> [--opção={arg1,arg2,arg3}] [argumento opcional ...]
```

Essas convenções são padronizadas e você poderá encontrá-las em outros lugares, como as páginas do manual.

Os símbolos "<" (menor que) e ">" (maior que) denotam um argumento **mandatório** a não ser copiado verbatim, mas a ser substituído de acordo com as suas necessidades. Por exemplo, `<nome_do_arquivo>` refeer ao nome de um arquivo. Se o nome é `foo.txt` você deve digitar `foo.txt`, e não `<foo.txt>` ou `<nome_do_arquivo>`.

Os colchetes ("[ ]") denotam argumentos opcionais, que podem ou não ser incluídos no comando.

As reticências ("...") significam um número arbitrário de argumentos que podem ser incluídos.

As chaves ("{ }") contém os argumentos autorizados neste lugar específico. Um deles deverá ser colocado aqui.

4.2.2. Anotações Especiais

De tempo em tempo você será direcionado a pressionar, por exemplo, as teclas **Ctrl**-**R**, o que significa que você deve pressionar e manter pressionada a tecla **Ctrl** e então pressionar a tecla **R** em seguida. O mesmo caso se aplica para as teclas **Alt** e **Shift**.

Nós utilizamos letras maiúsculas para representar as teclas, mas isto não significa que você tem que digitá-las em maiúscula também. Entretanto, pode haver programas em que **R** não significa o mesmo que **r**. Você será informado quando estiver trabalhando com estes programas.

Em relação aos menus, acessar o item Arquivo→Recarregar configuração do usuário (**Ctrl-R**) significa: clique no texto Arquivo exibido no menu (normalmente localizado na diagonal superior esquerda da janela). Então, quando o menu desdobrar, clique no item Reacrregar configuração do usuário. Além disto, você é informado que pode utilizar a combinação de teclas **Ctrl**-**R** para obter o mesmo resultado.

4.2.3. Usuários Genéricos do Sistema

Sempre que possível nós utilizamos dois usuários genéricos em nossos exemplos:

Fulano Pingus	usuario1	Este é o nosso usuário padrão, utilizado na maior parte dos exemplos neste livro.
Ciclano Pingus	usuario2	Este usuário pode ser criado mais tarde pelo administrador do sistema e, às vezes, é utilizado para variar durante o texto.

Capítulo 1. Conceitos Básicos de Sistemas UNIX®

O nome "UNIX®" deve lhe ser familiar, caso utilize um sistema UNIX® no trabalho. Neste caso esse capítulo pode ser de pouco interesse para você.

Para aqueles que nunca utilizaram um sistema UNIX®, a leitura desse capítulo é necessária. A compreensão dos conceitos que aqui serão introduzidos irá responder a um grande número de questões comumente formuladas pelos iniciantes no mundo **GNU/Linux**. Da mesma maneira, alguns desses conceitos irão ajudá-lo a resolver a grande maioria dos problemas que irá encontrar no futuro.

1.1. Usuários e Grupos

Uma vez que eles tem uma influência direta em todos os outros conceitos, essa seção irá introduzir os conceitos de usuários e grupos, que são de grande importância para o usuário.

O Linux é um sistema ***multiusuário***, e para utilizar a sua máquina com GNU/Linux, você deve ter uma ***conta*** na máquina. Durante a instalação, quando criou um usuário, na realidade você criou uma conta de usuário. Caso não se lembre, foram perguntados alguns itens:

- o "nome real" do usuário (que atualmente pode ser o que você quiser);
- um nome para o ***login***;
- e uma ***senha***.

Os dois parâmetros principais são o nome de login (comumente abreviado para login) e a senha. Os dois são necessários para acessar o sistema.

Quando um usuário é criado, um grupo padrão também é criado. Mais tarde veremos que os grupos são úteis quando desejamos compartilhar arquivos com outras pessoas. Um grupo pode conter quantas pessoas quanto necessário, e é muito comum observar essa separação em grandes sistemas. Por exemplo, em uma universidade, podemos ter um grupo para cada departamento, outro grupo para os professores, e assim por diante. O contrário também é verdade: um usuário pode ser membro de mais de um grupo. Um professor de matemática, por exemplo, pode ser membro do grupo professores e também pertencer ao grupo de estudantes de matemática.

Uma vez vistas as informações básicas, vejamos agora como entrar no sistema, ou efetuar o login.

Se a interface gráfica for inicializada automaticamente durante o boot do sistema, sua tela inicial se parecerá com a Figura 1-1.

Figura 1-1. Sessão de Login no Modo Gráfico

Para entrar no sistema, ou efetuar seu login, você deve primeiro selecionar sua conta na lista. Uma nova janela será mostrada, requisitando sua senha . Note que você terá que digitar sua senha às escuras, uma vez que os caracteres digitados serão mostrados na tela como asteriscos (*) enquanto você digitar. Você também pode selecionar o gerenciador de janelas que deseja. Assim que estiver pronto, clique no botão Login.

Se você estiver no console ou no modo "texto" irá ver algo semelhante ao que segue:

```
Mandriva Linux release 2006.0 (CodeName) for i586
 Kernel 2.6.12-6mdk on an i686 / tty1
 [machine_name] login:
```

Para entrar no sistema, digite o seu login (nome da conta) no prompt do login: e aperte **Enter**. A seguir o programa de login (chamado login) irá mostrar o prompt da Password: e esperar que você entre com a senha. Da mesma maneira que no modo gráfico, não serão mostrados os caracteres da senha, nem mesmo os asteriscos.

Note que você pode entrar no sistema (efetuar o login) várias vezes com a mesma conta em outros ***consoles*** e também no X (modo gráfico). Cada sessão que você abrir é independente das outras, além disso também é possível abrir várias sessões do X ao mesmo tempo (embora não seja recomendado por consumir muitos recursos do sistema). Por padrão, o Mandriva Linux tem seis ***consoles virtuais*** e um sétimo reservado para a interface gráfica. Você pode alternar entre eles pressionando as teclas **Ctrl-Alt-F<n>**, onde `<n>` é o número do console para o qual deseja ir. Por padrão, a interface gráfica está no console número 7. Entretanto, para alternar para o segundo console, você deve pressionar as teclas **Ctrl**, **Alt F2**.

Durante a instalação, o programa DrakX também requisitou a senha para um usuário especial: `root`. Esse é o administrador do sistema, que provavelmente será você mesmo. Para a segurança do seu sistema, é muito importante que a conta do usuário `root` seja protegida por uma senha boa e difícil de adivinhar.

Se você entra regularmente no seu sistema utilizando o usuário ***root***, é muito fácil cometer um erro que pode deixar o seu sistema inutilizável: um simples erro pode ocasionar isso. Em particular, se você não atribuir uma senha para a conta de usuário `root`, então **qualquer** usuário poderá alterar **qualquer** parte do seu sistema (mesmo outros sistemas operacionais que existam na sua máquina!). Naturalmente essa não é uma boa idéia.

É importante mencionar que internamente o sistema não o identifica pelo seu login. Ao invés disso, é utilizado um número único atribuído ao nome: o *User ID* (UID) . Da mesma maneira, os grupos são identificados pelo seu *Group ID* (GID) e não pelo seu nome.

1.2. Arquivos

Em comparação ao Windows® e muitos outros ***sistemas operacionais***, o tratamento de arquivos é bem diferente do utilizado no GNU/Linux. Nesta seção iremos cobrir a grande maioria das diferenças. Para mais informações, por favor veja o Capítulo 4.

As maiores diferenças são resultado direto do fato de que o Linux é um sistema multiusuário: cada arquivo é propriedade exclusiva de um usuário e um grupo. Um fato que não havíamos mencionado a respeito dos usuários é que cada um tem um diretório pessoal (chamado de ***diretório home***). O usuário é o proprietário desse diretório e de todos os arquivos criados nele. Também vale notar que esses arquivos também pertencem a um grupo e que esse é o grupo primário ao qual o usuário pertence. Como mencionado anteriormente (veja Seção 1.1), um usuário pode pertencer a mais de um grupo ao mesmo tempo.

Entretanto, não seria de muita utilidade se essa fosse a única idéia de posse de arquivo. Como dono dos arquivos, o usuário pode (e deve) atribuir **permissões** aos arquivos. Essas permissões distingüem tres categorias de usuários: o **dono (owner)** do arquivo, qualquer usuário pertencente ao **grupo** associado ao arquivo (também chamado de ***grupo do dono (owner group)***) porém não sendo o dono, e **outros**, que inclui qualquer usuário que não o dono do arquivo, nem um membro do grupo pertence ao dono do arquivo.

Existem três tipos de permissões diferentes:

1. **Leitura (Read)** (`r`): permite ao usuário ler o conteúdo de um arquivo. No caso de um diretório, permite que o usuário liste seu conteúdo (i.e. os arquivos contidos no diretório).
2. **Escrita (Write)** (`w`): permite a modificação do conteúdo do arquivo. Para um diretório, a permissão de escrita permite que o usuário acrescente ou remova arquivos desse diretório, mesmo não sendo o dono dos arquivos.
3. **eXecução (eXecute)** (`x`): permite que um arquivo seja executado (normalmente só arquivos executáveis tem essa permissão atribuída). Para um diretório, essa permissão faz com que o usuário possa ***percorrer*** o diretório, ou seja, entrar e sair do diretório. Observe que essa permissão é diferente da permissão de leitura: você pode ser capaz de percorrer um diretório embora continue incapaz de ler seu conteúdo!

Cada uma das combinações de permissão é possível. Por exemplo, você pode permitir que somente você leia um arquivo enquanto restringe o acesso a todos os outros usuários. Como dono do arquivo, também pode modificar o grupo atribuído ao arquivo (se e somente for membro do novo grupo).

Vejamos um exemplo com um arquivo e um diretório. A seguir pode ser visto o resultado do comando `ls -l` executado na ***linha de comando***:

```
$ ls -l
total 1
-rw-r-----   1 usuario1    users           0 Jul  8 14:11 um_arquivo
drwxr-xr--   2 usuario2    users        1024 Jul  8 14:11 um_diretorio/
$
```

Os resultados do comando `ls -l` (da esquerda para a direita):

- Os primeiros dez caracteres representam o tipo de arquivo e as permissões associadas a ele. O primeiro caracter é o tipo do arquivo: se for um arquivo regular, será representado por um hífen (`-`). No caso de um diretório, o caracter mais à esquerda será um `d`. Existem outros tipos de arquivo, que serão discutidos mais tarde. Os próximos nove caracteres representam as permissões associadas com aquele arquivo. Esses nove caracteres representam três grupos de três permissões cada. O primeiro grupo representa as permissões associadas ao dono do arquivo; o segundo as permissões associadas aos usuários do grupo ao qual o arquivo está associado; e as últimas três permissões se aplicam a todos os outros usuários. Um hífen (`-`) indica que a permissão não está atribuída.
- A seguir temos o número de **links** para o arquivo. Mais tarde veremos que o identificador único de um arquivo não é o seu nome, mas sim um número (o ***número do inode***), e que é possível que um arquivo no disco tenha vários nomes. No caso de um diretório, o número de links tem um significado especial, que será discutido mais para frente.
- Os próximos dados encontrados são o nome do proprietário seguido pelo nome do grupo ao qual o arquivo está associado.

- Ao final, são mostrados o tamanho do arquivo (em ***bytes***) e a data da sua última modificação, seguidos do nome do arquivo ou diretório.

Vejamos com mais detalhes as permissões associadas a cada um desses arquivos. Inicialmente, precisamos remover o primeiro caracter que representa o tipo, então, para o arquivo `um_arquivo`, temos as seguintes permissões: `rw-r-----`. Dividindo as permissões em grupos, temos:

- Os três primeiros caracteres (`rw-`) representam as permissões do dono do arquivo, neste caso, usuario1. Sendo assim, o usuário usuario1 tem permissão para ler o arquivo (`r`) e modificar seu conteúdo (`w`) porém, não tem permissão para executá-lo (`-`).
- os próximos três caracteres (`r--`) são aplicáveis para qualquer usuário que não usuario1, mas que seja membro do grupo `users`. Estes usuário têm permissão para ler o arquivo (`r`), entretanto não poderão alterá-lo nem mesmo executá-lo (`--`).
- os últimos três caracteres (`---`) representam as permissões para todos os usuários que não usuario1 e os membros do grupo `users`. Estes usuários não possuem permissão alguma sobre o arquivo, para eles o arquivo estará "invisível".

Para o diretório `a_directory`, temos as permissões `rwxr-xr--`, então:

- usuario2, sendo o proprietário do diretório, pode listar os arquivos contidos nele (`r`), adicionar ou remover arquivos do diretório (`w`), e também pode percorrê-lo (atravessá-lo) (`x`).
- Qualquer usuário que não usuario2, porém membro do grupo `users`, será capaz de listar os arquivos nesse diretório (`r`), porém não poderá adicionar e/ou remover arquivos (`-`), e ainda será capaz de percorrê-lo (`x`).
- Qualquer outro usuário será capaz de listar o conteúdo do diretório (`r`). Uma vez que não têm as permissões `wx`, não serão capazes de escrever arquivos ou entrar no diretório.

Existe **uma** exceção à essas regras: `root`. O usuário `root` pode mudar os atributos (permissões, dono e grupo do dono) de todos os arquivos, mesmo que ele não seja o proprietário, sendo assim, pode tornar-se dono do arquivo! O usuário `root` pode ler arquivos que não tem permissão de leitura para ele, percorrer diretórios que normalmente não teria acesso, e assim por diante. E se o usuário `root` precisar de uma permissão, ele simplesmente pode adicioná-la a si mesmo. O usuário `root` tem o controle completo do sistema, o que envolve uma dose de confiança na pessoa que possui a senha de `root`.

Por final, é importante notar as diferenças entre os nomes de arquivos presentes nos mundos UNIX® e Windows®. Para alguns, o UNIX® possui uma maior flexibilidade e menos limitações.

- Um nome de arquivo pode conter quaisquer caracteres, incluindo os não imprimíveis, exceto pelo caracter ASCII 0, que denota o final de uma string, e da **/**, que é o separador de diretórios. Além disso, pelo fato de o UNIX® ser sensível à capitulação (case sensitive), os arquivos `readme` e `Readme` são diferentes, porque `r` e `R` são considerados dois caracteres **diferentes** nos sistemas baseados no UNIX®.
- Como pudemos observar, o nome de arquivo não necessáriamente deve conter uma extensão, a não ser que seja de sua preferência colocá-la. As extensões de arquivo não identificam o conteúdo do arquivo no GNU/Linux, nem na maioria dos outros sistemas operacionais. As chamadas "extensões de arquivos" são bastante convenientes. O ponto (`.`) no UNIX® é tão somente um caracter como os outros, porém tendo também um significado especial. No UNIX®, nomes de arquivo começando com um ponto são considerados "arquivos ocultos"[1], que também incluem nomes de diretórios que começam com um "`.`"

Entretanto é necessário observar que muitas aplicações gráficas (gerenciadores de arquivos, aplicações de escritório, etc.) atualmente utilizam suas extensões para identificar seus arquivos. Sendo assim, é uma boa idéia utilizar nomes de arquivo com suas respectivas extensões para as aplicações que suportam-nas.

1. Por padrão, arquivos ocultos não serão mostrados em um gerenciador de arquivos, a não ser que você configure-o para tal. Em um terminal, você deve digitar o comando `ls -a` para ver todos os arquivos ocultos juntos com os arquivos normais. Essencialmente esses arquivos contém informações de configuração. A partir do seu diretório `home`, veja os arquivo `.mozilla` ou `.openoffice` como exemplo.

1.3. Processos

Um ***processo*** define uma instância de um programa sendo executado e também seu ***ambiente***. Apenas mencionaremos as principais diferenças entre GNU/Linux e Windows® (por favor, veja Capítulo 10 para mais informações).

A principal diferença está diretamente relacionada ao conceito de **usuário**: cada processo é executado com as permissões e direitos do usuário que o iniciou. Internamente o sistema identifica os processos com um número único, chamado *ID do processo*, ou PID. A partir desse PID, o sistema sabe quem (isto é, que usuário) iniciou o processo e um outro tanto de informações, que o sistema precisa para verificar a validade do processo. Voltemos ao exemplo do arquivo `a_file`. O usuário usuario2 será capaz de abrir esse arquivo no ***modo somente leitura***, e não no ***modo de leitura-escrita*** porque as permissões associadas ao arquivo não o permitem. novamente a exceção à regra é o usuário `root`.

Por causa disso, o GNU/Linux é virtualmente imune aos vírus. Para operarem, os vírus devem infectar arquivos executáveis. Como um usuário, você não tem acesso de escrita aos arquivos vulneráveis do sistema, sendo assim o risco é reduzido sobremaneira. Em geral, os vírus são extremamente raros no mundo UNIX®. Existem alguns poucos vírus conhecidos para o Linux, e são inofensivos quando executados como usuário normal. Somente um usuário pode danificar o sistema ativando um desses vírus: o usuário `root`.

Interessantemente, existem softwares anti-vírus para GNU/Linux, porém, a maioria deles é para arquivos provenientes dos sistemas operacionais DOS/Windows®! Por que então existem anti-virus executando no GNU/Linux se tem seu foco no DOS/Windows®? Cada vez mais podemos observar sistemas GNU/Linux agindo como servidores de arquivos para máquinas que utilizam Windows®. Esses servidores se utilizam do software Samba (Compartilhando Arquivos e Impressoras, capítulo do *Guia do Servidor*), para fornecer o serviço para as máquinas Windows®.

O Linux torna fácil o controle de processos. Uma das maneiras de controlar um processo é através da utilização de "sinais", que permitem suspender ou matar um processo, enviando o sinal correspondente ao processo. Entretando, você está limitado a enviar sinais aos seus próprios processos. Com exceção para o usuário `root`, o Linux e outros sistemas baseados no UNIX® não permitem que você envie sinais a processos que foram iniciados por outros usuários. No Capítulo 10, você irá aprender como obter o PID de um processo e como enviar um sinal para ele.

1.4. Uma Breve Introdução à Linha de Comando

A linha de comando é a maneira mais direta de enviar comandos à sua máquina. Ao utilizar a linha de comando no GNU/Linux, você perceberá rapidamente que ela é muito mais poderosa e funcional que outros prompts de comando que pode ter encontrado anteriormente. Essa superioridade é devido ao fato de ter acesso não somente a todos os programas do ambiente X, mas também a milhares de outros utilitários no modo texto (em contrapartida ao modo gráfico) que não tem equivalentes gráficos, com suas várias opções e possíveis combinações, que seriam sobremaneira difíceis de serem acessadas na forma de botões ou menus.

Reconhecidamente a grande maioria das pessoas precisa de um pouco de ajuda para começar. Se você ainda não estiver trabalhando no console e estiver utilizando uma interface gráfica, a primeira coisa a fazer é iniciar um emulador de terminal. Acesse o menu principal do GNOME, KDE ou qualquer outro gerenciador de janelas que estiver utilizando e irá encontrar um número de emuladores no menu Sistema+Terminais. Escolha um, por exemplo Konsole ou RXvt. Dependendo da sua interface de usuário, haverá um ícone que identifica claramente o terminal no painel (Figura 1-2).

Figura 1-2. O Ícone do Terminal no Painel do KDE

Quando você iniciar esse emulador de terminais, estará na realidade utilizando um shell, este é o nome do programa com o qual estará interagindo. Após iniciar o shell, você se encontrará diante do ***prompt*** de comando:

```
[usuario1@localhost usuario1]$
```

Assumimos que o nome do seu usuário é usuario1 e o nome de sua máquina é `localhost` (que é o caso quando sua máquina não faz parte de nenhuma rede). Após o prompt existe o espaço para que você digite

seus comandos. Note que quando você entrou no sistema como usuário `root`, o caracter do prompt `$` passa a ser um `#` (isso é verdade somente quando utilizar a configuraçào padrão, uma vez que pode configurar todos esses detalhes no GNU/Linux). Para se tornar o usuário `root`, execute o comando `su` após iniciar um shell.

```
[usuario1@localhost usuario1]$ su
# Enter the root password; (a senha não será mostrada na tela)
Password:
# exit (or Ctrl-D) volta à conta de usuário comum
[root@localhost usuario1]# exit
[usuario1@localhost usuario1]$
```

Quando você ***executa*** um shell pela primeira vez, normalmente estará no seu diretório `home/`. Para mostrar o nome do diretório corrente, execute o comando `pwd` (que quer dizer *Print Working Directory - Mostre o Diretório Corrente*):

```
$ pwd
/home/usuario1
```

A seguir veremos alguns comandos básicos que são muito úteis.

1.4.1. cd: Change Directory (Mude de Diretório)

O comando `cd` é semelhante ao utilizado no DOS, só que com mais funcionalidades. Ele faz exatamente o que seu acrônimo indica, muda o diretório corrente. Você pode utilizar o `.` e `..`, para indicar o diretório corrente e o diretório pai respectivamente. Digitando o comando `cd` sozinho você voltará ao seu diretório home. Executando `cd -` voltará ao último diretório visitado. E finalmente, pode especificar o diretório home do usuário usuario2 digitando `cd ~usuario2` (o `~` sozinho indica seu próprio diretório `home/`). Observe que, como usuário comum, você não pode entrar no diretório `home/` de outro usuário (a não ser que ele autorize explicitamente ou esta seja a configuração padrão do sistema), até que se torne o usuário `root`. Sendo assim, mude para o usuário `root` e pratique:

```
$ su -
Password:
# pwd
/root
# cd /usr/share/doc/HOWTO
# pwd
/usr/share/doc/HOWTO
# cd ../FAQ-Linux
# pwd
/usr/share/doc/FAQ-Linux
# cd ../../../lib
# pwd
/usr/lib
# cd ~usuario2
# pwd
/home/usuario2
# cd
# pwd
/root
```

Agora, volte ao usuário normal executando o comando `exit` e pressionando a tecla **Enter** (ou simplesmente pressionando **Ctrl-D**).

1.4.2. Variáveis de ambiente e o comando echo

Todos os processos tem suas ***variáveis de ambiente*** e o shell permite que você visualize-as diretamente utilizando o comando `echo`. Algumas variáveis de ambiente interessantes:

1. `HOME`: essa variável de ambiente contém uma string que representa o caminho para o seu diretório home.
2. `PATH`: ela contém a lista de todos os diretórios nos quais o shell irá procurar por executáveis quando você digitar um comando. Note que, diferentemente do DOS, por padrão, o shell **não** irá procurar por comandos no diretório corrente, a não ser que ele esteja listado nessa variável!
3. `USERNAME`: esta variável contém o nome de login.

4. `UID`: essa variável contém o ID do usuário.
5. `PS1`: determina como o seu prompt irá ser mostrado, é freqüentemente uma combinação de seqüências especiais de caracteres. Para mais informações você pode ler a página de manual bash(1) ***página de manual*** digitando o comando `man bash` em um terminal.

Para que o shell mostre o valor de uma variável, você deve colocar um `$` na frente do nome da variável. Vejamos um exemplo com o comando `echo`:

```
$ echo Hello
Hello
$ echo $HOME
/home/usuario1
$ echo $USERNAME
usuario1
$ echo Hello $USERNAME
Hello usuario1
$ cd /usr
$ pwd
/usr
$ cd $HOME
$ pwd
/home/usuario1
```

Como pode ver, o shell substitui o valor da variável antes que execute o comando. De outra maneira, o comando `cd $HOME` do exemplo não teria funcionado. De fato, o shell substitui primeiro a variável `$HOME` com o seu valor (`/home/usuario1`), então a linha se torna `cd /home/usuario1`, que é o que queríamos. A mesma coisa acontece com o exemplo do comando `echo $USERNAME`.

Se uma das variáveis de ambiente não existir, você pode criá-la temporariamente digitando `export NOME_VAR_AMB=value`. Uma vez feito isso, pode verificar se ela foi criada da seguinte maneira:

```
$ export USERNAME=usuario1 $ echo $USERNAME usuario1
```

1.4.3. cat: Mostre o Conteúdo de Um ou Mais Arquivos na Tela

Sem muito mais para explicar, esse comando simplesmente faz o seguinte: ele mostra o conteúdo de um ou mais arquivos para a saída padrão, normalmente a tela:

```
$ cat /etc/fstab
# This file is edited by fstab-sync - see 'man fstab-sync' for details
/dev/hda2 / ext3 defaults 1 1
/dev/hdc /mnt/cdrom auto umask=0022,user,iocharset=utf8,noauto,ro,exec,users 0 0
none /mnt/floppy supermount dev=/dev/fd0,fs=ext2:vfat,--,umask=0022,iocharset=utf8,sync 0 0
none /proc proc defaults 0 0
/dev/hda3 swap swap defaults 0 0
$ cd /etc
$ cat modprobe.preload
# /etc/modprobe.preload: kernel modules to load at boot time.
#
# This file should contain the names of kernel modules that are
# to be loaded at boot time, one per line.  Comments begin with
# a '#', and everything on the line after them are ignored.
# this file is for module-init-tools (kernel 2.5 and above) ONLY
# for old kernel use /etc/modules

nvidia-agp
$ cat shells
/bin/bash
/bin/csh
/bin/sh
/bin/tcsh
```

1.4.4. less: um Paginador

O nome é uma brincadeira com a palavra relacionada ao primeiro paginador utilizado no UNIX® chamado `more`. Um ***paginador*** é um programa que permite ao usuário visualizar arquivos longos página por página (mais precisamente, tela por tela). O motivo de examinarmos o comando `less` ao invés do comando `more` é que `less` é mais intuitivo. Você pode utilizar o comando `less` para visualizar aquivos grandes que não caibam em uma única tela. Por exemplo:

```
less /etc/termcap
```

Para navegar no arquivo, utilize as teclas cima e baixo. Pressione a tecla **Q** para terminar a execução. O comando `less` pode fazer mais coisas que isso: pressione **H** para uma tela de ajuda ser mostrada com as possíveis opções.

1.4.5. ls: Listing Files - Listando Arquivos

O comando `ls` (*LiSt*) é equivalente ao comando `dir` do DOS, porém com muito mais funcionalidades. De fato, ele pode fazer mais porque os arquivos também podem. A sintaxe do comando `ls` é:

```
ls [opções] [arquivo|diretorio][arquivo|diretorio...]
```

Se nenhum arquivo ou diretório for especificado na linha de comando, o `ls` irá listar os arquivos no diretório corrente. Existe um grande número de suas opções, iremos descrever apenas algumas delas:

- `-a`: lista todos os arquivos, incluindo ***arquivos ocultos***. Lembre-se que no UNIX®, os arquivos ocultos são aqueles que começam com um `.`; a opção `-A` lista "quase" todos os arquivos, ou seja, lista todos os arquivos que a opção `-a` listar, com exceção do "." e ".."
- `-R`: lista recursivamente, i.e todos os arquivos e subdiretórios de diretórios digitados na linha de comando.
- `-h`: se o tamanho do arquivo for mostrado, será mostrado em um formato legível, perto de cada arquivo. Isso quer dizer que você irá ver os tamanhos dos arquivos utilizando sufixos para os tamanhos, como "K", "M" and "G", por exemplo, "234K" e "132M". Note que esses tamanhos se referem a potência de 2 e não potência de 10. Sendo assim 1K é realmente 1024 bytes ao invés de 1000 bytes.
- `-l`: mostra informações adicionais sobre os arquivos, como as permissões associadas a ele, o dono e grupo do dono, o tamanho do arquivo e a data da última modificação.
- `-i`: mostra o número do inode (o número único do arquivo no sistema de arquivos, veja mais em Capítulo 4) perto de cada arquivo.
- `-d`: trata diretórios na linha de comando como se fossem arquivos normais ao invés de listar seu conteúdo.

Alguns Exemplos:

- `ls -R`: lista recursivamente o conteúdo do diretório corrente.
- `ls -ih imagens/ ..`: lista o número do inode e o tamanho em um formato legível para cada arquivo no diretório `imagens/`, no diretório pai e no diretório corrente.
- `ls -l imagens/*.png`: lista todos os arquivos no diretório `imagens/` cujos nomes terminem em `.png`, incluindo o arquivo `.png`, se ele existir.

1.4.6. Atalhos de Teclado Úteis

Existe um grande número de atalhos disponíveis, a sua principal vantagem é que eles economizam bastante tempo de digitação. Essa seção assume que você está utilizando o shell fornecido com o Mandriva Linux, bash, embora essas teclas de atalho também funcionem em outros terminais.

Primeiro: as setas. O bash mantém um histórico dos comandos anteriores, que podem ser vistos utilizando-se das teclas para cima e para baixo. Você pode voltar um número máximo de linhas definido na variável de ambiente `HISTSIZE`. Além disso, o histórico é perene entre uma e outra sessão, sendo assim, não irá perder os comandos digitados nas sessões anteriores.

As setas para esquerda e direita movimentam o cursor para a esquerda e direita na linha corrente, permitindo editar os comandos. Porém, existe mais a ser editado do que simplemente mover um caracter por vez: **Ctrl**-

A e **Ctrl**-**E**, por exemplo, movem o cursor para o início e fim da linha corrente. As teclas **Backspace** e **Del** agem como esperado. **Ctrl**-**K** irá apagar da posição corrente até o final da linha, e **Ctrl**-**W** irá apagar a palavra imediatamente antes do cursor (assim como **Alt**-**Backspace**).

Digitando **Ctrl**-**D** em uma linha em branco irá fechar a sessão corrente, o que é bem menor que ter que digitar `exit`. **Ctrl**-**C** irá interromper o comando em execução, exceto se você estiver editando a linha de comando, caso no qual o processo de edição será cancelado e você retornará ao prompt. **Ctrl**-**L** limpa a tela. **Ctrl**-**Z** interrompe temporariamente uma tarefa, suspendendo-a. Esse atalho é muito útil quando esquecemos de digitar o caracter "&" após o comando. Por exemplo:

```
$ xpdf MyDocument.pdf
```

Conseqüentemente você não poderá utilizar mais seu shell, uma vez que a tarefa de interface foi alocada para o processo do xpdf. Para colocar a tarefa em segundo plano e recuperar seu shell, simplesmente digite **Ctrl**-**Z** e, a seguir, o comando `bg`.

Finalmente, as teclas **Ctrl**-**S** e **Ctrl**-**Q**, que são utilizadas para suspender e restaurar a saída para a tela. Elas não são utilizadas freqüentemente, porém você pode digitar por engano **Ctrl**-**S** (além do mais, **S** e **D** estão perto uma da outra no teclado). Então, se você se encontrar em uma situação onde digita e não consegue ver nenhum caracter aparecendo no Terminal, tente pressionar **Ctrl**-**Q**. Note que todos os caracteres que foram digitados entre o **Ctrl**-**S** não desejado e o **Ctrl**-**Q** serão mostrados de uma vez só.

Capítulo 2. Discos e Partições

Este capítulo contém informação para aqueles que simplesmente desejam conhecer mais a respeito dos detalhes técnicos subjacentes do sistema. Este capítulo fornecerá uma completa descrição do esquema de particionamento de um PC. Portanto este capítulo será bastante útil se você está planejando configurar manualmente seu disco rígido.

2.1. Estrutura de um Disco Rígido

Um disco é fisicamente dividido em setores. Uma sequência de setores podem formar uma partição. Falando de forma aproximada, você pode criar tantas partições quanto você deseje, até um limite de 67 (3 partições primárias e uma partição secundária contendo até 64 partições lógicas): cada partição e considerada como um único disco rígido.

2.1.1. Setores

Simplificando, um disco rígido é meramente uma sequência de setores, que são as menores unidades de dados em um disco rígido. O tamanho típico de um é 512 bytes. Os setores de um disco rígido de "n" setores são numerados de "0" a "n-1".

2.1.2. Partições

O uso de múltiplas partições permite que você crie vários discos rígidos virtuais em seu disco físico real. Isto possui muitas vantagens:

- Diferentes sistemas operacionais usando diferentes estruturas de disco (chamadas ***sistemas de arquivos***): este é o caso com Windows® e o GNU/Linux. Tendo múltiplas partições em um disco rígido também permite que você instale vários sistemas operacionais em um mesmo disco físico.
- Por razões de performance, um sistema operacional pode preferir ter diferentes discos com diferentes sistemas de arquivos pois eles podem ser usados para coisas completamente diferentes. Um exemplo é o GNU/Linux que requer uma segunda partição chamada Swap. Esta partição é utilizada pelo gerenciador de memória virtual como memória virtual.
- Mesmo que todas as partições usem o mesmo sistema de arquivos, esta provado que é útil separar as diferentes partes do seu OS em diferentes partições. Uma simples configuração de exemplo seria separar os arquivos em duas partições: uma para dados pessoais, e outra para os seus programas. Isto permite que você atualize o seu OS, apagando completamente a partição que contém os programas enquanto mantém a partição de dados pessoais a salvo.
- Devido aos erros físicos em um disco rígido estarem normalmente localizados em setores adjacentes e não espelhados pelo disco, distribuir os arquivos entre partições diferentes pode limitar a perda de dados se o seu disco rígido ficar danificado fisicamente.

Normalmente, o tipo de partição especifica qual o sistema de arquivos que a partição deve contar. Cada sistema operacional precisa reconhecer os seus tipos de partições, mas não os dos outros. Por favor veja em Capítulo 6, e em Capítulo 4, para maiores informações.

2.1.3. Definindo a Estrutura do seu Disco

2.1.3.1. O modo mais simples

Este cenário implicaria apenas em duas partições: uma para o espaço de Swap, e outra para os arquivos[1], chamado `root` e rotulado como `/`.

Uma regra empírica é configurar a partição de swap com duas vezes o tamanho da RAM memória (por exemplo se você tiver 128 MB de memória RAM o tamanho do swap deverá ser de 256 MB). Entretanto para configurações com grande quantidade de memória (512 MB ou mais), esta regra não é crítica, e tamanhos menores são aceitáveis. Por favor tenha em mente que o tamanho da partição de swap pode ser limitado dependendo de que plataforma você esteja usando. Por exemplo, este limite é de 2GB em um x86, PowerPC e MC680x0; é de 512MB na MIPS; é de 128GB na Alpha e é de 3TB na Ultrasparc. Tenha em mente também que quanto maior a partição de swap, maior será a quantidade de recursos do OS (notadamente memória RAM) necessários a gerência desta partição.

2.1.3.2. Um Outro Esquema Comum

Separar os dados de programa. Para ser mais eficiente, normalmente definimos mais partições para separar os sistemas e programas dos dados. A partição de sistema conteria os programas necessários ao início do sistema e para manutenção básica.

Portanto nós podemos definir quatro partições:

Swap

A partição de Swap cujo tamanho será aproximadamente o dobro do tamanho da memória física RAM.

Raiz: `/`

A mais importante partição. Não contém apenas dados críticos e programas, mas também atua como ponto de montagem para outras partições (veja Capítulo 6).

A necessidade em termos de tamanho de uma partição raiz não é grande, 400MB geralmente são suficientes. Entretanto, se você planeja instalar aplicativos comerciais, que normalmente são localizados no diretório `/opt`, você precisrá aumentar o tamanho da partição raiz de acordo. Alternativamente, você pode criar uma partição separada para o `/opt`.

Dados estáticos: `/usr`

A maioria dos pacotes instalam a maioria dos seus arquivos executáveis e de dados sobre o diretório `/usr`. A vantagem de criar uma partição separada é que esta permite que você facilmente compartilhe estes dados com outras máquinas da rede.

O tamanho recomendado depende dos pacotes que você deseja instalar, e pode variar de 100MB para uma instalação bem enxuta, até vários GB para uma instalação completa. Um tamanho meio-termo de dois ou três GB (dependendo do seu disco rígido) normalmente é suficiente.

Diretórios Home: `/home`

Este diretório contém os diretórios pessoais de todos os usuários armazenados em sua máquina. O tamanho da partição depende do número de usuários armazenados e suas necessidades.

Uma outra solução consiste em **não** criar a partição separada para o arquivos do diretório `/usr`: `/usr` pode simplesmente ser um diretório dentro da partição raiz (`/`), entretanto você precisaria aumentar o tamanho da partição raiz (`/`) de acordo.

1. o sistema de arquivos padrão do Mandriva Linux é chamado `ext3`

Finalmente, você também pode criar somente as partições Swap e `root` (`/`), no caso de você não ter certeza do que você fará com seu computador. Neste caso, sua partição raiz `/home` seria localizada dentro da partição `root`, assim como seriam o diretório `/usr` e os demais diretórios.

2.1.3.3. Configurações Exóticas

Quando tivermos que configurar nossa máquina para um uso específico — como para um servidor web server ou um firewall — as necessidades são radicalmente diferentes daquelas para uma máquina desktop padrão. Por exemplo, um servidor FTP provavelmente necessitará de uma grande partição separada para o diretório `/var/ftp`, enquanto que o diretório `/usr` poderá ser relativamente pequeno. Nestas situações, sugerimos que você pense cuidadosamente nas suas necessidades antes de iniciar o processo de instalação.

É possível reconfigurar o tamanho da maioria das partições, no caso de você precisar ou usar um esquema de particionamento diferente sem a necessidade de reinstalar seu sistema e sem perder seus dados. Por favor consulte *Gerenciando Partições* o manual sobre *Guia do Usuário* .

Com alguma prática, você será capaz de mover uma partição cheia para uma árvore em um novo disco.

2.2. Convenções para nomenclatura de Discos e Partições

O GNU/Linux usa um método lógico para nomear as partições. Primeiro, durante a numeração das partições, ele ignora o tipo de sistema de arquivos de qualquer partição que você tenha. Segundo, ele nomeia as partições de acordo como o disco em que elas estão localizadas. Este é o processo como os disco são nomeados:

- Os dispositivos IDE master e slave primários (sejam eles um disco rígido, drive de CD-ROM ou qualquer outra coisa) são nomeados de `/dev/hda` e `/dev/hdb` respectivamente.
- Na interface secundária, o master é nomeado de `/dev/hdc` e o slave é nomeado de `/dev/hdd`.
- Se o seu computador contém outras interfaces IDE (por exemplo a interface IDE presente em algumas placas Soundblaster), estes discos serão nomeados de `/dev/hde`, `/dev/hdf`, etc. Você pode também possuirá interfaces IDE adicionais se você tiver controladoras RAID.
- Os discos SCSI são nomeados de `/dev/sda`, `/dev/sdb`, etc., na ordem do seu posicionamento no barramento SCSI (dependendo do aumento de IDs). O drives de CD-ROM SCSI são nomeados de `/dev/scd0`, `/dev/scd1`, sempre na ordem do seu posicionamento no barramento SCSI.

Se você possuir discos SATA IDE, o esquema de nomenclatura SCSI se aplica a eles.

As partições são nomeadas depois do disco em que são encontradas, na seguinte forma (neste exemplo usaremos partições em um disco IDE master primário mas as mesmas regram se aplicam a todos os outros tipos de discos):

- As partições primárias (ou extendidas) são nomeadas de `/dev/hda1` até `/dev/hda4`, quando estiverem presentes.
- Partições lógicas, se existiram, serão nomeadas de `/dev/hda5`, `/dev/hda6`, etc. na ordem do seu posicionamento na tabela lógica de partições.

Desta forma o GNU/Linux nomeará as partições da seguinte forma:

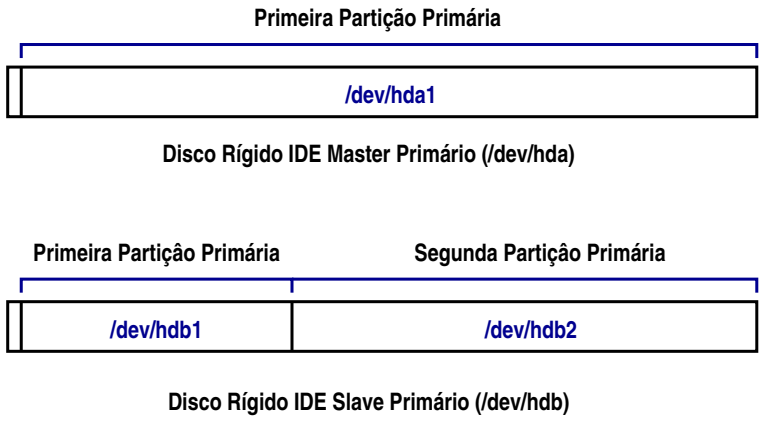


Figura 2-1. Primeiro Exemplo de Nomenclatura de Partições de um GNU/Linux

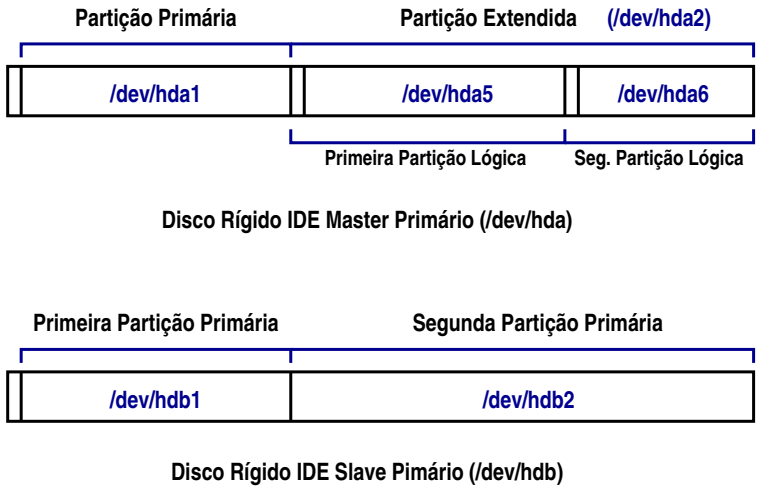


Figura 2-2. Segundo Exemplo de Nomenclatura de Partições de um GNU/Linux

Com este conhecimento em mãos, você será capaz de nomear várias partições e discos rígidos quando você tiver que manipulá-los. Você também verá os nomes de partições no GNU/Linux mesmo que ele não saiba inicialmente como gerenciá-los (ele ignora o fato de que eles não são partições nativas do GNU/Linux).

Mandriva Linux agora usa o udev (vide link udev FAQ (`http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev-FAQ`) para maiores informações). Isto garante plena compatibilidade com o esquema descrito acima e com os padrões como o Linux Standards Base Project (`http://www.linuxbase.org/`). Cada dispositivo é dinamicamente adicionado ao sistema assim que ele se torne disponível ou seja necessário.

Capítulo 3. Organização de Diretórios e Arquivos

Hoje em dia, um sistema UNIX® é grande, bastante grande. Isto é particularmente verdade para o GNU/Linux: a quantidade de programas disponíveis tornaria o sistema muito difícil de ser gerenciado caso não houvessem regras para governar a localização de arquivos e diretórios.

O padrão mais aceito é o FHS (Filesystem Hierarchy Standard) cuja versão 2.3 foi lançada em janeiro de 2004. O documento que descreve este padrão está disponível na Internet em diversos formatos no sítio chamado Pathname (`http://www.pathname.com/fhs/`). Este capítulo vai apenas oferecer uma breve descrição do FHS, o que deverá ser o suficiente para mostrar que diretório mais provavelmente contém um dado arquivo, ou onde um certo arquivo deverá ser colocado.

3.1. Dados Compartilhados/Não Compartilhados, Estáticos/Dinâmicos

Dados em um sistema UNIX® podem ser classificados de acordo com o seguinte critério: dados compartilhados podem ser comuns a vários computadores em uma rede, enquanto que dados não compartilhados são específicos para cada máquina. Dados estáticos não podem ser modificados durante o uso normal, enquanto que dados dinâmicos podem. À medida que formos explorando a estrutura da árvore de diretórios, iremos classificar cada um deles em uma destas categorias.

Estas classificações são apenas uma recomendação. Não é obrigatório utilizá-las, mas sua adoção vai facilitar muito o gerenciamento do sistema. Além disto, tenha em mente que a distinção entre estático e dinâmico apenas se aplica ao uso normal do sistema, e não à sua configuração. Quando um programa é instalado, sem dúvida que diretórios "normalmente" estáticos serão modificados, como por exemplo `/usr`.

3.2. O Diretório Raiz: /

O diretório raiz contém toda a hierarquia do sistema de arquivos. Ele portanto não pode ser classificado pois seus sub-diretórios podem ou não ser estáticos ou compartilhados. A seguir temos uma lista dos principais diretórios e sub-diretórios junto com a sua classificação:

- `/bin`: arquivos binários essenciais. Este diretório contém os comandos básicos que são utilizados por todos os usuários e que são necessários para a operação do sistema: `ls`, `cp`, `login`, etc. Estático, não compartilhável.
- `/boot`: contém os arquivos exigidos pelo carregador do sistema GNU/Linux (GRUB ou LILO para Intel, yaboot para PPC, etc.). Ele pode ou não conter o kernel, mas caso o kernel não esteja localizado neste diretório ele precisará estar no diretório raiz. Estático, não compartilhável.
- `/dev`: arquivos de dispositivo do sistema (`dev` como em *DEVices*). Alguns arquivos armazenados no diretório `/dev` são obrigatórios, como `/dev/null`, `/dev/zero` e `/dev/tty`. Estático, não compartilhável.
- `/etc`: contém todos os arquivos de configuração específicos para a máquina. Este diretório não pode conter arquivos binários. Estático, não compartilhável.
- `/home`: é onde todos os diretórios e arquivos pessoais dos usuários do sistema estão armazenados. Este diretório pode ou não ser compartilhado (algumas redes grandes o exportam via NFS). Os arquivos de configuração das suas aplicações favoritas (como leitores de e-mail ou navegadores) estão localizados neste diretório e começam com um ponto ("."). Por exemplo, os arquivos de configuração do Mozilla estão no diretório `.mozilla`. Dinâmico, compartilhável.
- `/lib`: contém bibliotecas que são essenciais para o sistema; também armazena os módulos do kernel no sub-diretório `/lib/modules/VERSÃO_DO_KERNEL`. Possui todas as bibliotecas exigidas pelos programas dos diretórios `/bin` e `/sbin`. O carregador/ligador (*loader/linker*) opcional em tempo de execução `ld*` bem como a biblioteca C dinâmica `libc.so` também devem ser colocados neste diretório. Estático, não compartilhável.
- `/mnt`: diretório contendo os pontos de montagem para sistemas de arquivos montados temporariamente como `/mnt/cdrom`, `/mnt/floppy`, etc. O diretório `/mnt` também é usado para montar diretórios temporários (um chaveiro USB será montado em `/mnt/removable`, por exemplo). Dinâmico, não compartilhável.

- `/opt`: contém pacotes que não são essenciais para a operação do sistema, sendo reservado para pacotes suplementares; programas como Adobe Acrobat Reader são comumente instalados aqui. O FHS recomenda que arquivos estáticos (binários, bibliotecas, páginas de manual, etc.) instalados na estrutura `/opt` sejam armazenados em `/opt/nome_do_pacote` e que os arquivos de configuração específicos sejam colocados em `/etc/opt`. Estático, não compartilhável.
- `/root`: diretório pessoal do usuário `root`. Dinâmico, não compartilhável.
- `/sbin`: contém arquivos binários do sistema essenciais para a inicialização. A maior parte destes programas somente pode ser executada pelo usuário `root`. Um usuário normal pode até executá-los, mas é possível que não tenham efeito. Estático, não compartilhável.
- `/tmp`: diretório cujo objetivo é armazenar arquivos temporários que os programas possam eventualmente criar. Dinâmico, não compartilhável.
- `/usr`: explicado em mais detalhes em Seção 3.3. Estático, compartilhável.
- `/var`: diretório para dados que podem ser modificados em tempo real por programas (como servidores de correio eletrônico, programas de auditoria, servidores de impressão, etc.). Dinâmico. Seus vários sub-diretórios podem ser compartilháveis ou não.

3.3. /usr: O Enorme

O diretório `/usr` é o local principal de armazenamento de aplicações. Os arquivos binários contidos aqui não são necessários para a inicialização ou manutenção do sistema, portanto a hierarquia `/usr` pode estar, e muitas vezes está, localizada em um sistema de arquivos separado. Por causa do seu tamanho usualmente grande, `/usr` possui sua própria hierarquia de sub-diretórios. Nós vamos mencionar apenas alguns:

- `/usr/X11R6`: a hierarquia completa do X Window System. Todos os binários e bibliotecas necessários para a operação do X (incluindo o servidor X) devem ser armazenados aqui. O diretório `/usr/X11R6/lib/X11` contém todos os detalhes de configuração do X que não variam de uma máquina para outra. Configurações específicas para cada computador devem ser armazenadas em `/etc/X11`.
- `/usr/bin`: contém a maior parte dos binários do sistema. **Qualquer** programa que não for necessário para a manutenção do sistema e também não for um programa de administração da máquina deve ser colocado neste diretório. As únicas exceções são programas que são compilados e instalados pelo próprio usuário, que devem ser armazenados em `/usr/local`.
- `/usr/lib`: contém todas as bibliotecas necessárias para executar programas armazenados em `/usr/bin` e `/usr/sbin`. Também há uma ligação simbólica em `/usr/lib/X11` apontando para `/usr/X11R6/lib/X11`, o diretório que contém as bibliotecas do X Window System (mas apenas se X estiver instalado). [1]
- `/usr/local`: este é o diretório onde o usuário deverá instalar quaisquer aplicações que forem compiladas a partir do código fonte. O programa de instalação deverá criar a hierarquia necessária.
- `/usr/share`: este diretório contém todos os dados de somente leitura e independentes de arquitetura necessários pelas aplicações instaladas em `/usr`. Aqui também são encontradas as informações de fuso horário e internacionalização (`zoneinfo` e `locale`).

Devemos citar ainda os diretórios `/usr/share/doc` e `/usr/share/man`, que contêm, respectivamente, a documentação das aplicações e as páginas de manual do sistema.

3.4. /var: Dados que Podem Mudar Durante o Uso

O diretório `/var` contém todos os dados operacionais dos programas em execução no sistema. Ao contrário dos dados existentes no diretório `/tmp`, estes dados precisam ser mantidos no caso de uma reinicialização da máquina. Há vários sub-diretórios, muitos bastante úteis:

- `/var/log`: contém os arquivos de registro do sistema que podem ser analisados pelo administrador para identificar e depurar problemas (`/var/log/messages` e `/var/log/kernel/errors` para mencionar apenas dois).

1. Note que o Mandriva Linux agora utiliza Xorg em vez de X Window System como o Sistema de Janelas X padrão.

- `/var/run`: usado para saber que processos estão sendo executados no sistema desde sua inicialização, permitindo a intervenção do administrador no caso de uma troca de ***nível de execução*** (veja Capítulo 11).
- `/var/spool`: contém os arquivos de trabalho do sistema que estão esperando por algum tipo de ação ou processamento. Por exemplo, `/var/spool/cups` contém os arquivos de trabalho do servidor de impressão, enquanto que `/var/spool/mail` contém os arquivos de trabalho do servidor de correio eletrônico (por exemplo, todas as mensagens que estão chegando e saindo do sistema).

3.5. /etc: Arquivos de Configuração

O diretório `/etc` é um dos diretórios mais essenciais para os sistemas UNIX® pois contém todos os arquivos de configuração específicos para a máquina. **Nunca** remova este diretório para ganhar espaço! Da mesma forma, se você deseja remanejar sua estrutura de diretórios para outras partições, lembre-se que `/etc` não pode ser colocado em uma partição separada: ele é necessário para a inicialização do sistema e deve estar presente na partição raiz no momento do boot.

Aqui estão alguns arquivos importantes:

- `passwd` e `shadow`: estes são arquivos texto que contêm todos os usuários do sistema e suas senhas criptografadas. O arquivo `shadow` somente estará presente se senhas estilo shadow estiverem em uso, o que é na verdade o padrão da instalação por motivos de segurança.
- `inittab`: este é o arquivo de configuração para o `init` que é fundamental para a inicialização do sistema.
- `services`: este arquivo contém uma lista dos serviços de rede existentes.
- `profile`: este é o arquivo de configuração geral para o shell. Suas configurações podem ser sobrescritas por arquivos de configuração específicos para cada shell. Por exemplo, `.bashrc` para o shell bash.
- `crontab`: este é o arquivo de configuração para o `cron`, o programa responsável pela execução periódica de comandos.

Alguns sub-diretórios existem para programas que requerem um número grande de arquivos de configuração. Isto se aplica ao X Window System, por exemplo, que armazena todos os seus arquivos de configuração no diretório `/etc/X11`.

Capítulo 4. O Sistema de Arquivo do Linux

O seu sistema GNU/Linux está em seu disco rígido dentro de um sistema de arquivos. Neste capítulo nós iremos discutir os vários aspectos de sistemas de arquivo disponíveis para o GNU/Linux, assim como as possibilidades que eles oferecem.

4.1. Comparando Alguns Sistemas de Arquivo

Durante a instalação, você pode escolher sistemas de arquivo diferentes para as suas partições, pois elas serão formatadas utilizando algoritmos diferentes.

A não ser que você seja um especialista, escolher um sistema de arquivo não é simples. Nós iremos dar uma olhada rápida em alguns sistemas de arquivo, dos quais todos estão disponíveis no Mandriva Linux.

4.1.1. Sistemas de Arquivo Diferentes Utilizáveis

4.1.1.1. Ext2

O Second Extended File System (sua forma abreviada é ext2FS ou simplesmente ext2) tem sido o sistema de arquivo padrão do GNU/Linux por muitos anos. Ele substituiu o Extended File System (que é de onde vem o "Second" no nome do sistema). O ext2 corrige alguns problemas e limitações de seu predecessor.

O ext2 respeita os padrões comuns do sistema de arquivos do UNIX®. Desde o seu início, ele estava destinado a desenvolver enquanto ainda oferecer uma grande robustez e boa performance.

Ele precisa ser desmontado para ser redimensionado.

4.1.1.2. Ext3

Como o seu nome sugere, o Third Extended File System é o sucessor do ext2. Ele é compatível com o último mas aprimorado por incorporar o suporte a ***journaling***.

Uma das maiores falhas dos sistemas de arquivo "tradicionais" como o ext2 é a sua baixa tolerância à quedas repentinas do sistema (queda de energia ou travamentos causados por software). Falando de maneira geral, uma vez que o sistema é reiniciado, estes tipos de evento envolvem um exame muito longo da estrtura do sistema de arquivos e tentativas de corrigir erros, que às vezes resultam em um sistema corrompido. Isto pode causar a perda parcial ou total de dados gravados.

O Journaling soluciona este problema. Para simplificar, vamos dizer que o que estamos fazendo é armazenar as ações (como a gravação de um arquivo) **antes** de realmente executarmos. Nós podemos comparar esta funcionalidade com aquele capitão de barco que mantém um diário de bordo onde anota todos os eventos que acontecem diariamente. O resultado: um sistema de arquivo sempre coerente. E se um problema ocorrer, uma verificação é muito rápida e os reparos eventuais muito reduzidos. Então o tempo gasto para verificar um sistema de arquivo é proporcional ao uso atual e não relacionado ao seu tamanho.

Sendo assim, o ext3 oferece um sistema de arquivo com journaling enquanto mantém a estrutura do ext2, garantindo uma compatibilidade excelente. Isto torna muito fácil trocar de ext2 para ext3 e voltar novamente.

Assim como o ext2, ele precias ser desmontado para ser redimensionado.

4.1.1.3. ReiserFS

Ao contrário do ext3, o `reiserfs` foi escrito do zero. Ele é um sistema de arquivos com journaling como o ext3, mas a sua estrutura interna é radicalmente diferente porque ele usa conceitos de árvores binárias inspiradas nos softwares de base de dados e também possui um tamanho de bloco variável, tornando-o otimizado para o uso com vários (milhares ou centenas de milhares) arquivos pequenos. Ele também possui uma boa performance com arquivos grandes, o que o torna adequado para vários usos.

Ele pode ser redimensionado "on the fly", sem a necessidade de desmontar o sistema de arquivos.

4.1.1.4. JFS

O JFS é o sistema de arquivo com journaling desenvolvido e utilizado pela IBM. Antes ele era proprietário e fechado, mas a IBM decidiu abrí-lo para o movimento de software livre. Sua estrutura interna é similar a do `reiserfs`.

Ele não pode ser redimensionado no GNU/Linux.

4.1.1.5. XFS

O XFS é o sistema de arquivo com journaling desenvolvido pela SGI e utilizado pelo sisema operacional Irix. Assim como o JFS, o XFS também era proprietário e fechado, até que a SGI resolveu abrí-lo para o movimento de software livre. Sua estrutura interna possui muitas funcionalidades diferentes, como suporte real-time para transferência de dados via rede, extensões e sistemas de arquivo para clusters (mas não na versão gratuita).

Com o GNU/Linux ele pode ser redimensionado somente para um tamanho maior. Você não pode reduzir. O redimensionamento somente pode ser feito com o sistema de arquivos montado.

4.1.2. Diferenças entre Sistemas de Arquivo

	Ext2	**Ext3**	**ReiserFS**	**JFS**	**XFS**
Estabilidade	Excelente	Muito Bom	Bom	Mediano	Bom
Ferramentas para restaurar arquivos apagados	Sim (complexo)	Sim (complexo)	Não	Não	Não
Tempo de reinicialização após travamento	Demorado, até mesmo muito demorado	Rápido	Muito rápido	Muito rápido	Muito rápido
Estado das informações após um travamento	Falando no geral, bom. Mas alto risco de perda total ou parcial de informações	Muito bom	Mediano[a]	Muito bom	Muito bom
Suporte a ACL	Sim	Sim	Não	Não	Sim

	Ext2	Ext3	ReiserFS	JFS	XFS
Notas: a. é possível melhorar os resultados na recuperação de falhas aplicando o journaling nos **dados** e não apenas nos **metadados**, adicionando a opção *data=journal* no /etc/fstab.					

Tabela 4-1. Características dos Sistemas de Arquivo

O tamanho máximo de um arquivo depende de vários parâmetros (por exemplo, o tamanho de bloco no ext2/ext3), e deve desenvolver de acordo com a versão do kernel e da arquitetura.

No kernel 2.6.X o limte do dispositivo de bloco pode ser extendido utilizando um kernel compilado com o suporte a Large Block Device habilitado (CONFIG_LBD=y). Para mais informações, consulte Adding Support for Arbitrary File Sizes to the Single UNIX Specification (http://www.unix.org/version2/whatsnew/lfs.html), Large File Support in Linux (http://www.suse.com/~aj/linux_lfs.html), e Large Block Devices (http://www.gelato.unsw.edu.au/IA64wiki/LargeBlockDevices). Com isto e um sistema de arquivo com suporte, você poderá ter muitos TB sem a necessidade de executar qualquer truque especial do sistema de arquivo, como é feito pelo JFS.

4.1.3. E Sobre Performance?

É sempre muito difícil comparar performance entre arquivos de sistema. Todos os testes possuem as suas limitações e os resultados devem ser interpretados com precaução, comparações feitas há alguns meses ou semanas atrás já são bastante velhas. Não vamos esquecer que o hardware de hoje (especialmente se tratando da capacidade de discos rígidos e performance das controladoras de disco) alavancou consideravelmente as diferenças entre os vários sistemas de arquivo.

Cada sistema oferece vantagens e desvantages. Na verdade, tudo depende de como você usa a sua máquina. Uma simples máquina para desktop ficará boa com o ext2. Para um servidor, um sistema de arquivo com journaling como o ext3 é recomendado. O reiserfs, por causa de sua concepção, é melhor se utilizado em um servidor de banco de dados. O JFS é o preferido em casos onde a transferência de dados do sistema de arquivos é o ponto principal. O XFS é interessante se você precisa de suas funcionalidades avançadas. Para uso "normal", os quatro sistemas de arquivo dão aproximadamente o mesmo resultado e todos eles posuem opções diferentes para ajustar o sistema de arquivo para um caso particular. Verifique a documentação do sistema de arquivos para mais informações.

4.2. Tudo é um Arquivo

No *Guia do Usuário* introduzimos os conceitos de permissão de acesso e posse dos arquivos, mas agora que entendemos o ***sistema de arquivo*** do UNIX® (e também os sistemas de arquivo do Linux) precisamos redefinir o conceito sobre "o que é um arquivo".

Aqui, "tudo" **realmente** significa tudo. Um disco rígido, uma partição no disco, uma porta paralela, uma conexão a um web site, uma placa de rede: todos estes são arquivos. Até mesmo diretórios são arquivos. O Linux reconhece vários tipos de arquivos além dos diretórios e arquivos padrões. Perceba que tipo de arquivo aqui não significa o **conteúdo** de um arquivo: para o GNU/Linux e qualquer sistema UNIX®, um arquivo, quer ele seja uma imagem PNG, um arquivo binário ou qualquer outra coisa, é apenas um fluxo de bytes. Diferenciar os arquivos de acordo com o seu conteúdo é papel das aplicações.

4.2.1. Os Diferentes Tipos de Arquivo

Quando você executa ls -l, o primeiro caractere identifica o tipo do arquivo. Nós já vimos dois tipos de arquivos: arquivos regulares (-) e diretórios (d). Você também pode achar outros tipos se você navegar pela árvore de arquivo e listar conteúdo de seus diretórios:

1. ***Arquivos de caractere***: eles são arquivos especiais do sistema (como o /dev/null, que nós já discutimos anteriormente), ou periféricos (portas seriais ou paralelas), que possuem em comum o fato de que os seus conteúdos (se é que possuem algum) não são ***armazenados*** na memória. Estes arquivos são identificados pela letra c.

2. ***Arquivos de Bloco***: estes arquivos são periféricos, e ao contrário dos arquivos de caractere, estes possuem os seus conteúdos armazenados em memória. Por exemplo, alguns arquivos desta categoria são: discos rígidos, partições de discos, drives de disquete, drives de CD-ROM e outros dispositivos de armazenamento. Arquivos como `/dev/hda` e `/dev/sda5` são exemplos de arquivos de bloco. Estes arquivos são identificados pela letra `b`.
3. ***Links simbólicos***: estes arquivos são muito comuns e bastante utilizados no procedimento de inicialização do Mandriva Linux (veja Capítulo 11). Como o nome já diz, o propósito destes arquivos é de referenciar outros arquivos de uma maneira simbólica, o que significa que eles são arquivos cujos conteúdos são caminhos para arquivos diferentes. Eles também podem estar apontando para um arquivo que não existe. Eles são frequentemente chamados de ***soft links***, e são identificados pela letra `l`.
4. ***Named pipes***: em caso de você estar se perguntando, sim, estes são muito similares ao pipe utilizado nos comandos shell, mas com a diferença de que estes possuem nomes. Entretanto eles são muito raros e talvez você não encontre um durante a sua viagem na árvore de arquivo. Estes arquivos são identificados pela letra `p`. Veja a Seção 4.4.
5. ***Sockets***: este é o tipo de arquivo para todas as conexões de rede, mas apenas algumas delas possuem nomes. What's more, há tipos diferentes de sockets e somente um pode ser referenciado, mas isto está além do escopo deste livro. estes arquivos são identificados pela letra `s`.

Aqui está um exemplo de cada arquivo:

```
$ ls -l /dev/null /dev/sda  /etc/rc.d/rc3.d/S20random /proc/554/maps \
/tmp/ssh-usuario1/ssh-510-agent
crw-rw-rw-    1 root     root       1,   3 May  5  1998 /dev/null
brw-rw----    1 root     disk       8,   0 May  5  1998 /dev/sda
lrwxrwxrwx    1 root     root           16 Dec  9 19:12 /etc/rc.d/rc3.d/
  S20random -> ../init.d/random*
pr--r--r--    1 usuario1  usuario1         0 Dec 10 20:23 /proc/554/maps|
srwx------    1 usuario1  usuario1         0 Dec 10 20:08 /tmp/ssh-usuario1/
  ssh-510-agent=
$
```

4.2.2. Inodes

Inodes são, de acordo com o paradigma de que "Tudo é um Arquivo", uma parte fundamental de qualquer sistema de arquivo UNIX®. A palavra ***inode*** é uma abreviação para "Information NODE" (Nó de informação).

Inodes são armazenados em disco em uma tabela de inode. Eles existem para todos os tipos de arquivos que possam estar amazenados em um sistema de arquivo, incluindo diretórios, pipes nomeados, arquivos de caracteres e outros. O que nos leva a esta outra frase famosa: "O inode é o arquivo". Inodes são como o UNIX® indentifica um arquivo dee uma maneira única.

Não, você não entendeu errado: no UNIX®, você **não identifica um arquivo pelo seu nome**, mas pelo seu número de inode[1]. A razão para isto é que o mesmo arquivo pode ter vários nomes, ou até mesmo nenhum. No UNIX®, um nome de arquivo é apenas uma entrada em um inode de diretório. Uma entrada desta é chamada de link. Vamos olhar os links para mais detalhes.

4.3. Links

A melhor maneira de entender o que é um link é olhando um exemplo. Vamos criar um arquivo (normal):

```
$ pwd
/home/usuario1/example
$ ls
$ touch a
$ ls -il a
32555 -rw-r--r--  1 usuario1 usuario1 0 Aug  6 19:26 a
```

1. **Importante**: note que os números de inode são únicos **por sistema de arquivo**, o que significa que um inode com o mesmo número pode existir em outro sistema de arquivo. Isto nos leva a diferença entre inodes em disco e inodes na memória. Enquanto dois inodes em disco podem ter o mesmo número se eles estiverem em sistemas de arquivo diferentes, inodes em memória possuem um número único em todo o sistema. Uma solução para obter exclusividade, por exemplo, é associar o número do inode em disco com o identificador do dispositivo de bloco.

A opção `-i` do comando `ls` imprime o número do inode, que é o primeiro campo na saída. Como você pode ver, antes de criarmos o arquivo `a`, não havia arquivos no diretório. O outro campo de interesse é o terceiro, que indica o número de links do arquivo (links de inode, na verdade).

O comando `touch a` pode ser separado em duas ações distintas:

- criação de um inode, para o qual o sistema operacional deu o número 32555, e que o seu tipo é o de um arquivo normal;
- criação de um link para este inode, chamado `a`, no diretório atual (`/home/usuario1/example`). Assim sendo, o arqivo `/home/usuario1/example/a` é um link para o inode de número 32555, e é atualmente o único, pois o contador de link indica `1`.

Mas agora, se digitarmos:

```
$ ln a b
$ ls -il a b
32555 -rw-r--r--  2 usuario1 usuario1 0 Aug  6 19:26 a
32555 -rw-r--r--  2 usuario1 usuario1 0 Aug  6 19:26 b
$
```

Nós criamos outro link para o mesmo inode. Como você pode ver, nós não criamos um arquivo chamado `b`. Ao invés disso, nós adicionamos um link para o inode de número 32555 no mesmo diretório, e atribuímos o nome `b` a este novo link. Você pode ver na saída do `ls -l` que o contador de link para o inode indica 2 em vez de 1.

Agora, se fizermos:

```
$ rm a
$ ls -il b
32555 -rw-r--r--  1 usuario1 usuario1 0 Aug  6 19:26 b
$
```

Nós podemos verificar que, mesmo após apagarmos o "arquivo original", o inode continua existindo. Mas agora, o único link para ele é o arquivo chamado `/home/usuario1/example/b`.

Então um arquivo no UNIX® não possui nome; mas sim um ou mais ***link***(s) em um ou mais diretórios.

Os próprios diretórios também são armazenados em inodes. A contagem de links para eles é de acordo com o número de subdiretórios que eles possuem. E por isso há pelo menos dois links por diretório: o próprio diretório (representado pela entrada `.`) e o seu diretório pai (representado por `..`). Então um diretório com dois subdiretórios terá no mínimo quatro links: `.`, `..` e os links para cada subdiretório.

Exemplos típicos de arquivos que não estão referenciados (ex.: não possuem nome) são conexões de rede. Você nunca verá o arquivo correspondente a sua conexão ao web site da Mandriva Linux (`http://www.mandrivalinux.com`) na sua árvore de arquivo, não importa qual diretório você procure. De maneira similar, quando você usa um ***pipe*** no shell, o inode correspondente ao pipe existe, mas nao está referenciado. Arquivos temporários sao outros exemplos de inodes sem nomes. Você cria um arquivo temporário, abre ele, e então remove-o. A arquivo existe enquanto está aberto, mas ninguém mais pode abrí-lo (já que não existe nome para abrí-lo). Desta forma, se a aplicação travar, o arquivo temporário é removido automaticamente.

4.4. Pipes "Anônimos" e Pipes Nomeados

Vamos voltar ao exemplo dos pipes, já que eles são bastante interessantes e também um bom exmplo das noções de links. Quando você usa um pipe em uma linha de comando, o shell cria o pipe para você e o coloca para funcionar, então o comando antes do pipe escreve nele e o comando depois do pipe lê a partir dele. Todos os pipes, quer sejam anônimos (como aqueles utilizados pelo shell) ou nomeados (veja abaixo) trabalha como FIFOs (First In, First Out - Primeiro que entra, primeiro que sai). Nós já vimos alguns exemplos de como usar pipes no shell, mas vamos dar outra olhada para o propósito de nossa demonstração:

```
$ ls -d /proc/[0-9] | head -5
/proc/1/
/proc/2/
/proc/3/
/proc/4/
/proc/5/
```

Uma coisa que você não vai perceber neste exemplo (porque ela acontece rápida demais para percebermos) é que a escrita em pipes é bloqueada. Isto significa que quando o comando `ls` escreve no pipe, ele fica bloqueado até que um processo na outra ponta leia o pipe. Para visualizar o efeito, você pode criar pipes nomeados, que ao contrário dos pipes utilizados pelo shell possuem nomes (ex.: eles são referenciados, enquanto os pipes do shell não são)[2]. O comando para criar um pipe nomeado é o `mkfifo`:

```
$ mkfifo a_pipe
$ ls -il
total 0
169 prw-rw-r--  1 usuario1 usuario1 0 Aug  6 19:37 a_pipe|
  #
  # Você pode ver que o contador de link é 1, e que a saída mostra
  # que  o arquivo é um pipe ('p').
  #
  # Você também pode usar o ln aqui:
  #
$ ln a_pipe the_same_pipe
$ ls -il
total 0
169 prw-rw-r--  2 usuario1 usuario1 0 Aug  6 19:37 a_pipe|
169 prw-rw-r--  2 usuario1 usuario1 0 Aug  6 19:37 the_same_pipe|
$ ls -d /proc/[0-9] >a_pipe
  #
  # O processo está bloqueado, já que não umprocesso para ler na
  # outra ponta.
  # Digite Control Z para suspender o processo...
  #
[1]+  Stopped                 ls -F --show-control-chars --color=auto -d /proc/[0-9] >a_pipe
  #
  # ...Então coloque-o em background:
  #
$ bg
[1]+ ls -F --show-control-chars --color=auto -d /proc/[0-9] >a_pipe &
  #
  # agora leia-o a partir do pipe...
  #
$ head -5 <the_same_pipe
  #
  # ...o processo de escrita termina
  #
/proc/1/
/proc/2/
/proc/3/
/proc/4/
/proc/5/
[1]+  Done                    ls -F --show-control-chars --color=auto -d /proc/[0-9] >a_pipe
$
```

De maneira similar, leituras também estão bloqueadas. Se nós executarmos os comandos acima em ordem reversa, nós veremos que os blocos do `head`, esperando por algum processo para dar a ele algo para ler:

```
$ head -5 <a_pipe
#
# Blocos do programa, suspenso com: C-z
#
[1]+  Stopped                 head -5 <a_pipe
#
# Coloque-o no background...
#
$ bg
[1]+ head -5 <a_pipe &
#
# ...e alimente-o :)
#
$ ls -d /proc/[0-9] >the_same_pipe
/proc/1/
/proc/2/
/proc/3/
/proc/4/
/proc/5/
[1]+  Done                    head -5 <a_pipe
$
```

2. Existem outras diferenças entre os dois tipos de pipes, mas estão fora do escopo deste livro.

Você também pode constatar um efeito não desejado no exemplo anterior: o comando `ls` terminou antes que o comando `head` o pegasse. A consequência disto é que você foi devolvido imediatamente para o prompt, mas o `head` foi executado depois e você só viu sua saída depois de retornar.

4.5. Arquivos Especiais: Arquivos de Caractere e de Bloco

Como já mencionado anteriormente, estes arquivos são criados pelo sistema ou por periféricos na sua máquina. Nós também já mencionamos que o conteúdo dos arquivos de bloco são armazenados em buffer, enquanto os arquivos de caractere não. Para ilustrarmos isto, insira um diquete no drive e digite o seguinte comando duas vezes:

```
$ dd if=/dev/fd0 of=/dev/null
```

Você deve ter observado o seguinte: a primeira vez que o comando foi executado, todo o conteúdo do disquete foi lido. Na segunda vez que você executou o comando, não houve acesso algum ao drive de disquete. Isto aconteceu porque o conteúdo do disquete foi armazenado em buffer na primeira vez que você executou o comando — e você não alterou nada no disquete entre os dois comandos.

Mas, se você quiser imprimir um arquivo grande desta maneira (sim, vai funcionar):

```
$ cat /a/big/printable/file/somewhere >/dev/lp0
```

O comando vai levar o mesmo tempo para ser executado uma, duas, ou cinqüenta vezes. Isto porquê o `/dev/lp0` é um arquivo de caractere, e o seu conteúdo não é armazenado em buffer.

Existe um lado bom no fato dos arquivos de bloco serem armazenados em buffers: não apenas as leituras são feitas do buffer, mas as escritas também são armazenadas. Isto permite que a escrita em disco seja assíncrona: quando você escreve um arquivo em disco, a operação de escrita propriamente dita não é imediata. Ela irá acontecer quando o kernel Linux decidir executar a escrita no hardware. É claro que, se você precisar ele poderá ser sobrescrito para um determinado sisteam de arquivos; dê uma olhada nas opções `sync` e `async` na man page mount(8) e também na Seção 4.7 para mais detalhes.

Finalmente, cada arquivo especial possui um número ***principal*** e um número ***secundário***. Na saída do `ls -l`, eles aparecem no lugar do tamanho, já que o tamanho para estes arquivos é irrelevante:

```
$ ls -l /dev/hdc /dev/lp0
brw-rw----  1 usuario1 cdrom 22, 0 Feb 23 19:18 /dev/hdc
crw-rw----  1 root root 6, 0 Feb 23 19:17 /dev/lp0
```

Aqui, o número principal e o secundário do `/dev/hdc` são 22 e 0, enquanto para o `/dev/lp0`, eles são 6 e 0. Note que estes números são únicos por categoria de arquivo, o qual significa que pode haver um arquivo de caractere com número principal 22 e secundário 0, e de maneira similar, pode haver um arquivo de bloco com principal 6 e secundário 0. Estes números existem por uma simples razão: eles permitem ao kernel associarem as operações corretas a estes arquivos (isto é, aos periféricos que estes arquivos se referem): você não gerencia um drive de disquete da mesma maneira que um disco rígido SCSI, por exemplo.

4.6. Links Simbólicos, Limitação de "Hard" Links

Aqui nós temos que enfrentar um conceito errado muito comum, mesmo entre usuários UNIX®, que está principalmente ligado ao fato de que links da maneira como vimos até agora (erroneamente chamados de "hard" links) estão somente associados com arquivos regulares (e nós vimos que este não é o caso — já que até mesmo links simbólico são referenciados "linked"). Mas isto requer que primeiro expliquemos o que são links simbólicos ("soft" links ou "symlinks").

Links simbólicos são arquivos de um tipo específico, cujo conteúdo é uma string arbitrária que aponta para um arquivo existente ou não. Quando você menciona um link simbólico na linha de comando ou em um programa, na verdade você está acessando o aquivo para o qual ele aponta, se ele existir. Por exemplo:

```
$ echo Hello >myfile
$ ln -s myfile mylink
$ ls -il
total 4
169 -rw-rw-r--   1 usuario1  usuario1  6 Dec 10 21:30 myfile
416 lrwxrwxrwx   1 usuario1  usuario1  6 Dec 10 21:30 mylink -> myfile
```

```
$ cat myfile
Hello
$ cat mylink
Hello
```

Você pode ver que o tipo de arquivo para `mylink` é `l`, de *Link* simbólico. Os direitos de acesso para um link simbólico não são significantes: eles serã sempre `rwxrwxrwx`. Você também pode ver que ele **é** um arquivo diferente de `myfile`, já que o número de seu inode é diferente. Mas ele refere ao arquivo simbolicamente, então quando você digita `cat mylink`, você irá na verdade exibir o conteúdo do arquivo `myfile`. Para demonstrar que um link simbólico possui uma string arbitrária, nós podemos fazer o seguinte:

```
$ ln -s "I'm no existing file" anotherlink
$ ls -il anotherlink
418 lrwxrwxrwx    1 usuario1     usuario1           20 Dec 10 21:43 anotherlink
-> I'm no existing file
$ cat anotherlink
cat: anotherlink: No such file or directory
$
```

Mas os links simbólicos existem porque eles podem suprir várias limitações encontradas nos links normais (""hard""):

- Você não pode criar um link para um inode em um diretório que está em um sistema de arquivo diferente do inode. A razão é simples: o contador do link é armazenado no próprio inode, e não podem ser compartilhados entre sistemas de arquivo. Mas os links simbólicos permitem isto;
- Você não pode referenciar diretórios para evitar criar círculos no sistema de arquivo. Mas você pode fazer um link simbólico apontar para um diretório e utilizá-lo como se ele fosse um diretório de verdade.

Links simbólicos são então muito úteis em diversas circunstâncias, e normalmente, as pessoas tendem a utiliza-los para referenciar arquivos até mesmo quando um link normal poderia ser utilizado. Uma vantagem do link normal, é a de que você não perde o arquivo quando o "original" é apagado.

Por fim, se você observou com cuidado, você sabe que o tamanho de um link simbólico é apenas o tamanho da string dentro dele.

4.7. Atributos de Arquivo

Da mesma forma que o FAT possui atributos de arquivo (arquivo, arquivo do sistema, invisível, apenas leitura), um sistema de arquivos GNU/Linux também possui os seus, mas eles são diferentes. Nós iremos descrevê-los brevemente para completar o assunto, mas eles são raramente utilizados. Entretanto, se você realmente quer um sistema seguro, continue a ler.

Há dois comandos para manipular atributos de arquivo: `lsattr` e `chattr`. Você provavelmente já matou a questão, `lsattr` "LiSta" atributos, enquanto `chattr` modifica-os ("CHanges"). Estes atributos apenas podem ser configurados em diretórios e arquivos normais. A seguir vamos descrever alguns dos atributos possíveis, mas você deve olhar na man page chattr(1) para obter uma lista completa:

1. `A` ("no **A**ccess time"): se um arquivo ou diretório possuir este atributo configurado, não importa quando ele for acessado, seja para leitura ou escrita, a data do último acesso não será alterada. Isto pode ser útil, por exemplo, em arquivos e diretórios que são freqüentemente acessados para leitura, especialmente porque este parâmetro é o único a mudar em um inode uando ele é aberto para leitura.
2. `a` ("**a**ppend only"): se um arquivo possui este atributo configurado e é aberto para escrita, a única operação possível é adicionar dados ao seu conteúdo anterior. em um diretório, isto significa que você apenas pode adicionar arquivos nele, mas não renomear ou apagar os já existentes. Apenas o `root` pode ativar ou desativar este atributo.
3. `d` ("no **d**ump"): `dump` é o utilitário padrão do UNIX® para backups. Ele copia qualquer sistema de arquivo que tenha o contador dump igual a 1 no `/etc/fstab` (veja o capítulo Capítulo 6). Mas se um arquivo ou um diretório possuir este atributo, ele não será considerado quando o dump estiver trabalhando. Perceba que para diretórios, isto também inclui todos os subdiretórios e arquivos abaixo dele.

4. `i` ("**i**mmutable"): um arquivo ou diretório com este atributo configurado não pode ser modificado: ele não pode ser renomeado, nenhum link pode ser criado para ele[3] e ele não pode ser removido. Apenas o `root` pode configurar ou retirar este atributo. Note que isto também previne mudanças na data de acesso e, então você não precisa configurar o atributo `A` quando o `i` estiver configurado.
5. `s` ("**s**ecure deletion"): quando um arquivo ou diretório com este atributo é apagado, os blocos que ele ocupava são sobrescritos com zeros.
6. `S` ("**S**ynchronous mode"): quando um arquivo ou diretório possui este atributo configurado, todas as modificações nele são sincronizadas e escritas ao disco imediatamente.

Por exemplo, você pode querer configrar o atributo `i` em arquivos essenciais ao sistema para evitar algumas surpresas desagradáveis. Também considere o trecho sobre o atributo `A` na man page: isto evita muita operação de disco e, em particular, pode poupar o uso de bateria em laptops.

3. Esteja certo do que significa "adicionar um link", tanto para um arquivo quanto para um diretório!

Capítulo 5. O Sistema de Arquivos /proc

O sistema de arquivo `/proc` é específico do GNU/Linux. É um sistema de arquivos virtual, então os arquivos que você encontrar neste diretório não ocupam nenhum espaço no seu disco rigido. Esta é uma maneira muito conveniente de se obter informações sobre o sistema, especialmente pelo fato de que a maioria dos arquivos neste diretório podem ser entendidos por humanos (bem, com uma pequena ajuda). Muitos programas na verdade coletam informações dos arquivos no `/proc`, formatam da maneira deles e então exibem os resultados. Há alguns programas que exibem informações sobre processos (`top`, `ps` e amigos) que fazem exatamente isto. O `/proc` é também uma boa fonte de informação sobre o seu hardware, e assim como os programas que exibem os processos, existem vários programas que são apenas interfaces para as informações armazenadas no `/proc`.

Há também um subdiretório especial, o `/proc/sys`. Ele permite que você exiba e modifique os parâmetros do kernel, efetivando as alterações imediatamente.

5.1. Informação Sobre Processos

Se você listar o conteúdo do diretório `/proc`, você verá vários diretório que possuem números como nome. Estes são os diretórios contendo informações de todos os processos que estão sendo executados no sistema:

```
$ ls -d /proc/[0-9]*
/proc/1/    /proc/302/  /proc/451/  /proc/496/  /proc/556/  /proc/633/
/proc/127/  /proc/317/  /proc/452/  /proc/497/  /proc/557/  /proc/718/
/proc/2/    /proc/339/  /proc/453/  /proc/5/    /proc/558/  /proc/755/
/proc/250/  /proc/385/  /proc/454/  /proc/501/  /proc/559/  /proc/760/
/proc/260/  /proc/4/    /proc/455/  /proc/504/  /proc/565/  /proc/761/
/proc/275/  /proc/402/  /proc/463/  /proc/505/  /proc/569/  /proc/769/
/proc/290/  /proc/433/  /proc/487/  /proc/509/  /proc/594/  /proc/774/
/proc/3/    /proc/450/  /proc/491/  /proc/554/  /proc/595/
```

Note que, como um usuário, você só pode (logicamente) exibir informações relacionadas aos seus processos, e sobre os processos dos outros usuários. Então, acesse o sistema como `root` e veja quais informações estão disponíveis sobre o processo 1, que é o processo do init e o responsável por iniciar todos os outros:

```
$ su
Password:
# cd /proc/1
# ls -l
total 0
-r--------  1 root root 0 Aug 15 18:14 auxv
-r--r--r--  1 root root 0 Aug 15 18:14 cmdline
lrwxrwxrwx  1 root root 0 Aug 15 18:14 cwd -> //
-r--------  1 root root 0 Aug 15 18:14 environ
lrwxrwxrwx  1 root root 0 Aug 15 18:14 exe -> /sbin/init*
dr-x------  2 root root 0 Aug 15 18:14 fd/
-rw-r--r--  1 root root 0 Aug 15 18:14 loginuid
-r--r--r--  1 root root 0 Aug 15 18:14 maps
-rw-------  1 root root 0 Aug 15 18:14 mem
-r--r--r--  1 root root 0 Aug 15 18:14 mounts
-rw-r--r--  1 root root 0 Aug 15 18:14 oom_adj
-r--r--r--  1 root root 0 Aug 15 18:14 oom_score
lrwxrwxrwx  1 root root 0 Aug 15 18:14 root -> //
-rw-------  1 root root 0 Aug 15 18:14 seccomp
-r--r--r--  1 root root 0 Aug 15 18:14 stat
-r--r--r--  1 root root 0 Aug 15 18:14 statm
-r--r--r--  1 root root 0 Aug 15 18:14 status
dr-xr-xr-x  3 root root 0 Aug 15 18:14 task/
-r--r--r--  1 root root 0 Aug 15 18:14 wchan
#
```

Cada diretório contém as mesmas entradas. Aqui está uma descrição breve de algumas delas:

1. `cmdline`: este (pseudo-)arquivo contém toda a linha de comando utilizada para invocar o processo. Ela não está formatada: não há espaços entre o programa e seus argumentos, e nem caractere de nova linha no final. Para visualizá-la, você pode usar: `perl -ple 's,\00, ,g' cmdline`.
2. `cwd`: este link simbólico aponta para o diretório de trabalho atual do processo ("cwd - current working directory (diretório de trabalho atual)").

3. `environ`: este arquivo contém todas as variáveis de ambiente definidas para este processo, na forma `VARIÁVEL=valor` . Similar ao `cmdline`, a saída não é formatada de nenhuma maneira: as variáveis não são separadas em novas linhas, e também não há indicação de nova linha no final. Uma maneira de vê-la é através do comando: `perl -ple 's,\00,\n,g' environ`.
4. `exe`: este é um link simbólico apontando para o arquivo executável do processo.
5. `fd`: este subdiretório contém a lista dos descritores de arquivo atualmente abertos pelo processo. Veja abaixo.
6. `maps`: quando você exibe o conteúdo deste pipe nomeado (com o comando `cat` por exemplo), você pode ver as partes do espaço de endereço do processo que está mapeado para um arquivo. Da esquerda para a direita, os campos são: o espaço de endereço associado a este mapeamento, as permissões associadas a estes mapeamento, a marca no início do arquivo onde começa o mapeamento, o número principal e secundário (em hexadecimal) do dispositivo em que o arquivo mapeado está localizado, o número do inode do arquivo, e finalmente o nome do arquivo propriamente dito. Quado o dispositivo é 0 e não há número de inde ou nome do arquivo, este é um mapeamento anônimo. Veja mmap(2).
7. `root`: este é um link simbólico que aponta para o diretório raiz utilizado pelo processo. Normalmente ele será `/`, mas veja chroot(2) para mais informações.
8. `status`: este arquivo contém diversas informações sobre o processo: o nome do executável, seu estado atual, seu PID e PPID, seu UID e GID real e efetivo, seu uso de memória, e outras informações. Note que os arquivos `stat` e `statm` são obsoletos. A informação que eles contém está agora armazenada no `status`.

Se nós listarmos o conteúdo do diretório `fd` para um processo escolhido aleatoriamentes nós iremos obter isso:

```
# ls -l /proc/8141/fd/
total 4
lrwx------  1 usuario2 usuario2 64 Aug  4 09:05 0 -> /dev/tty1
lrwx------  1 usuario2 usuario2 64 Aug  4 09:05 1 -> /dev/tty1
lrwx------  1 usuario2 usuario2 64 Aug  4 09:05 2 -> /dev/tty1
l-wx------  1 usuario2 usuario2 64 Aug  4 09:05 3 -> /home/usuario2/seti32/lock.sah
#
```

Na verdade, esta é a lista de descritores de arquivo aberta pelo processo. Cada descritor aberto do arquivo é exibido por um link simbólico, onde o nome é o número descritor, e aponta para o arquivo aberto por ele[1]. Note as permissões nos links simbólicos: este é o único lugar onde elas fazem sentido, já que elas representam as permissões do arquivo aberto que corresponde ao descritor.

5.2. Informação sobre Hardware

Aparte dos diretórios associados a processos, o `/proc` também contém uma miríade de informação sobre o hardware na sua máquina. Uma listagem de arquivos no diretório `/proc` exibe o seguinte:

```
$ ls -d [a-z]*
acpi/      diskstats    iomem      locks       pci            sysvipc/
asound/    dma          ioports    mdstat      scsi/          tty/
buddyinfo  driver/      irq/       meminfo     self@          uptime
bus/       execdomains  kallsyms   misc        slabinfo       version
cmdline    fb           kcore      modules     splash         vmstat
config.gz  filesystems  keys       mounts@     stat
cpuinfo    fs/          key-users  mtrr        swaps
crypto     ide/         kmsg       net/        sys/
devices    interrupts   loadavg    partitions  sysrq-trigger
$
```

Por exemplo, se olharmos no conteúdo de `/proc/interrupts`, nós podemos ver que ele contém a lista de interrupções que estão sendo utilizadas pelo sistema, junto com o periférico que as utiliza. Similarmente, `ioports` contém a lista de intervalos de endereços de entrada/saída que estão ocupados, e por último, o `dma` faz o mesmo para os canais DMA. Então, quando quiser localizar um conflito, olhe o conteúdo destes três arquivos:

```
$ cat interrupts
           CPU0
```

1. Se você se lembra o que foi explicado na Seção 7.4, você sabe o que significam os descritores `0`, `1` e `2`. O descritor `0` é a entrada padrão, `1` é a saída padrão e `2` é o erro padrão.

```
  0:     543488          XT-PIC  timer
  2:          0          XT-PIC  cascade
  5:        109          XT-PIC  ohci_hcd:usb2, eth1
  7:          1          XT-PIC  parport0
  8:          0          XT-PIC  rtc
  9:       3432          XT-PIC  acpi, NVidia CK8
 10:      52855          XT-PIC  ehci_hcd:usb3, eth0
 11:       7538          XT-PIC  libata, ohci_hcd:usb1
 12:       1386          XT-PIC  i8042
 14:         20          XT-PIC  ide0
 15:       5908          XT-PIC  ide1
NMI:          0
LOC:          0
ERR:          0
MIS:          0

$ cat ioports
0000-001f : dma1
0020-0021 : pic1
0040-0043 : timer0
0050-0053 : timer1
0060-006f : keyboard
0070-0077 : rtc
0080-008f : dma page reg
00a0-00a1 : pic2
00c0-00df : dma2
00f0-00ff : fpu
0170-0177 : ide1
01f0-01f7 : ide0
0376-0376 : ide1
0378-037a : parport0
037b-037f : parport0
03c0-03df : vesafb
03f6-03f6 : ide0
03f8-03ff : serial
0778-077a : parport0
0970-0977 : 0000:00:0b.0
  0970-0977 : sata_nv
09f0-09f7 : 0000:00:0b.0
  09f0-09f7 : sata_nv
0b70-0b73 : 0000:00:0b.0
  0b70-0b73 : sata_nv
0bf0-0bf3 : 0000:00:0b.0
  0bf0-0bf3 : sata_nv
0cf8-0cff : PCI conf1
4000-407f : motherboard
  4000-4003 : PM1a_EVT_BLK
  4004-4005 : PM1a_CNT_BLK
  4008-400b : PM_TMR
  4020-4027 : GPE0_BLK
4080-40ff : motherboard
  4080-40ff : pnp 00:00
4200-427f : motherboard
  4200-427f : pnp 00:00
4280-42ff : motherboard
  4280-42ff : pnp 00:00
4400-447f : motherboard
  4400-447f : pnp 00:00
4480-44ff : motherboard
  44a0-44af : GPE1_BLK
5000-503f : motherboard
  5000-503f : pnp 00:01
5100-513f : motherboard
  5100-513f : pnp 00:01
9000-9fff : PCI Bus #02
  9000-907f : 0000:02:07.0
    9000-907f : 0000:02:07.0
ac00-ac0f : 0000:00:0b.0
  ac00-ac0f : sata_nv
b000-b07f : 0000:00:0b.0
  b000-b07f : sata_nv
b800-b8ff : 0000:00:06.0
  b800-b8ff : NVidia CK8
bc00-bc7f : 0000:00:06.0
  bc00-bc7f : NVidia CK8
```

```
c000-c007 : 0000:00:04.0
  c000-c007 : forcedeth
c400-c41f : 0000:00:01.1
f000-f00f : 0000:00:09.0
  f000-f007 : ide0
  f008-f00f : ide1

$cat dma
 3: parport0
 4: cascade
$
```

Ou, de maneira mais simlpes, use o comando `lsdev`, que reúne informação destes arquivos e ordena-as por periféricos, o que é, sem dúvida, mais conveniente.[2]:

```
$ lsdev
Device            DMA   IRQ  I/O Ports
------------------------------------------------
0000:00:01.1                 c400-c41f
0000:00:04.0                 c000-c007
0000:00:06.0                 b800-b8ff bc00-bc7f
0000:00:09.0                 f000-f00f
0000:00:0b.0                 0970-0977 09f0-09f7 0b70-0b73 0bf0-0bf3 ac00-ac0f b000-b07f
0000:02:07.0                   9000-907f     9000-907f
cascade             4     2
CK8                          9
dma                          0080-008f
dma1                         0000-001f
dma2                         00c0-00df
eth0                        10
eth1                         5
forcedeth                      c000-c007
fpu                          00f0-00ff
GPE0_BLK                       4020-4027
GPE1_BLK                       44a0-44af
i8042                       12
ide0                        14  01f0-01f7 03f6-03f6   f000-f007
ide1                        15  0170-0177 0376-0376   f008-f00f
keyboard                     0060-006f
motherboard                  4000-407f 4080-40ff 4200-427f 4280-42ff 4400-447f 4480-44ff 5000-503f 5100-513f
NVidia                         b800-b8ff   bc00-bc7f
ohci_hcd:usb1               11
parport0            3        7  0378-037a 037b-037f 0778-077a
PCI                          0cf8-0cff 9000-9fff
pic1                         0020-0021
pic2                         00a0-00a1
PM1a_CNT_BLK                   4004-4005
PM1a_EVT_BLK                   4000-4003
PM_TMR                         4008-400b
pnp                            4080-40ff   4200-427f   4280-42ff   4400-447f   5000-503f   5100-513f
rtc                          8  0070-0077
sata_nv                        0970-0977   09f0-09f7   0b70-0b73   0bf0-0bf3   ac00-ac0f   b000-b07f
serial                       03f8-03ff
timer                        0
timer0                       0040-0043
timer1                       0050-0053
vesafb                       03c0-03df
$
```

Uma listagem exaustiva dos arquivos demoraria demais, então aqui vai uma descrição dos principais:

- `cpuinfo`: este arquivo contém, como o próprio nome diz, informação sobre o(s) processador(es) presente(s) em sua máquina.
- `modules`: este arquivo contém a lista de módulos que estão sendo utilizadas pelo kernel, junto com a contagem de uso de cada um. Na verdade, esta é a informação utilizada pelo comando `lsmod` que a exibe de uma maneira mais inteligível.
- `meminfo`: este arquivo contém informação sobre o uso de memória no momento em que voce exibe o conteúdo dele. O comado `free` exibe a mesma informação de uma maneira mais fácil de se entender.

2. `lsdev` faz parte do pacote `procinfo`

- `apm`: se você possui um laptop, exibir o conteúdo deste arquivo irá permitir qe você visualize o estado da bateria. Você pode ver quando o AC está plugado, o nível de carga da bateria, e se a APM BIOS de seu laptop suportar (infelizmente nem todos possuem suporte), o tempo restante da bateria em minutos, etc. O arquivo não é muito legível por si só, então talvez você prefira utilizar o comando `apm`, o qual oferece a mesma informação em um formato que pode ser entendido por humanos.

  Note que computadores modernos agora oferecem suporte a ACPI em vez de APM. Veja abaixo.

- `bus`: este subdiretório contém informação sobre todos os periféricos encontrados em diferentes barramentos na sua máquina. A informação normalmente não é legível, e na maior parte das vezes ela é reformatada com utilitários externos: `lspcidrake`, `lspnp`, etc.

- `acpi`: vários dos arquivos e diretórios contidos são úteis especialmente para laptops, onde você pode selecionar várias opções para economizar energia. Note que é mais fácil modificar estas opções com uma aplicação de alto nível, como as que estão inclusas nos pacotes `acpid` e `kapacity`.

  As entradas mais interessantes são:

  `battery`

  Mostra quantas baterias estão no laptop, e informações relacionadas como a quantidade de carga restante, capacidade máxima, etc.

  `button`

  Permite que você controle ações associadas a botões "especiais" como power, sleep, lid, etc.

  `fan`

  Exibe o estado dos ventiladores do seu computador, se estão funcionando ou não, e permite que você pare/inicie eles de acordo com um certo critério. O nível de controle dos ventiladores em sua máquina depende da placa-mãe.

  `processor`

  Há um subdiretório para cada CPU na sua máquina. Opções de controle variam de um processador para outro. Processadores móveis possuem mais funcionalidade, incluindo:

  - possibilidade de utilizar vários estados de alimentação, equilibrando entre performance e consumo de energia;
  - possibilidade de alterar a frqüência do clock para reduzir o consumo de energia por parte do CPU;

  Note que existem muitos processadores que não oferecem estas possibilidade.

  `thermal_zone`

  Informação sobre o quão quente o seu sistema/processador está.

5.3. Exiba e altere parâmetros do kernel

O papel do subdiretório `/proc/sys` é reportar diferentes parâmetros do kernel, e permitir que você interaja com alguns deles. Ao contrário de todos os outros arquivos no `/proc`, alguns arquivos neste diretório podem ser modificados, mas somente pelo `root`.

Uma lista de diretórios e arquivos iria levar muito tempo para ser descrita, principalmente porque o conteúdo dos diretórios depende do sistema e assim a maior parte dos arquivos seriam utilizados apenas para aplicações muito específicas. Entretanto, aqui estão dois usos comuns deste subdiretório:

1. Permitir roteamento: até mesmo se o kernel padrão do Mandriva Linux estiver configurado para fazer o roteamento, você pode autorizá-lo explicitamente a fazer isso. Para isto, você tem que digitar o seguinte comando como `root`:

   ```
   $ echo 1 >/proc/sys/net/ipv4/ip_forward
   ```

   Substitua o 1 por 0 se você quer proibir o roteamento.

2. Prevenindo o IP spoofing: o IP spoofing consiste em fazer alguém acreditar que um pacote vindo do outro lado do mundo foi originado na interface por onde ele chega. Esta técnica é normalmente utilizada por ***crackers*** [3]. Você pode fazer o kernel prevenir este tipo de invasão. Digite:

   ```
   $ echo 1 >/proc/sys/net/ipv4/conf/all/rp_filter
   ```

   e este tipo de invasão se torna impossível.

Estas mudanças irão permanecer somente enquanto o sistema estiver sendo executado. Se o sistema for reiniciado, então os valores voltarão à configuração padrão. Para reiniciar os valores para outra coisa que não os padrões durante o processo de inicialização, você pode adicionar o comando que digitou no prompt do shell, no arquivo `/etc/rc.d/rc.local`, assim você evita de ter que digitar ele sempre. Outra solução é modificar o `/etc/sysctl.conf`, veja sysctl.conf(5) e sysctl(8) para mais informações.

3. Mas não por ***hackers***!

Capítulo 6. Sistemas de Arquivo e Pontos de Montagem

Como vimos no Capítulo 2, todos os arquivos do sistema são organizados em uma única árvore. E cada vez que queremos acessar um dispositivo removível (como um CD-ROM) ou um local remoto em um servidr de arquivos, o conteúdo será literalmente "montado" em algum local da árvore.

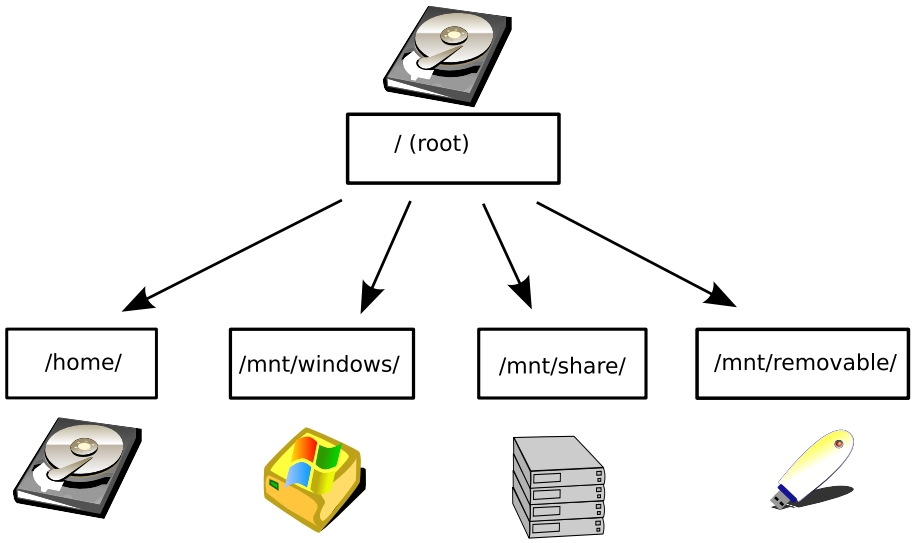


Figura 6-1. Pontos de Montagem

A Figura 6-1 mostra: a raiz, uma partição GNU/Linux que hospeda outra partição Linux em `/home/` e também uma Windows®, um compartilhamento remoto de um servidor de arquivos (Windows® ou UNIX®), e um chaveiro USB. Hoje em dia muitos dispositivos podem ser montados em um sistema de arquivos GNU/Linux, incluindo quase todos os tipos de sistemas de arquivos, WebDAV e até mesmo coisas mais exóticas, como o e-mail do Google™...

Para um melhor entendimento dos conceitos sobre pontos de montagens, nós elaboramos este capítulo baseado em um caso prático. Suponha que você acabou de comprar um disco rígido novo, sem partições. Sua partição Mandriva Linux está completamente cheia, e como você precisa de mais espaço, você decide mover uma sessão inteira da estrutura da árvore[1] para o seu disco rígido novo. como o seu disco tem uma capacidade de armazenamento grande, você decide mover o seu maior diretório para ele: `/usr`.

Nós usaremos este exemplo a partir da Seção 6.2, mas antes teremos um pouco de teoria.

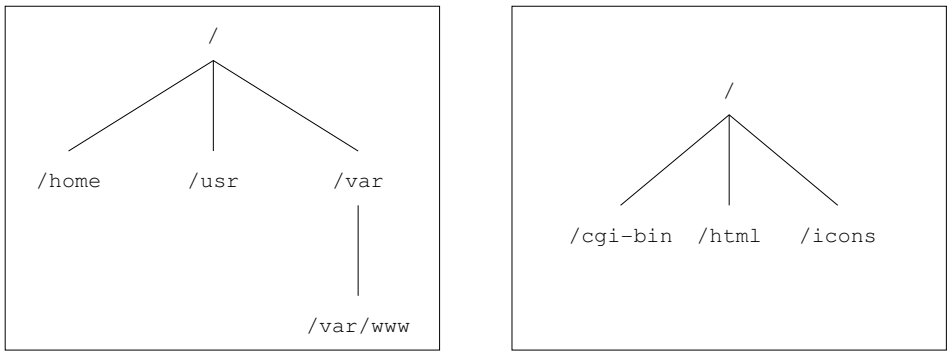
6.1. Princípios

Todo disco rígido pode ser dividido em partições, e cada uma contém um sistema de arquivos. Enquanto o Windows® atribui uma letra para cada um destes sistemas de arquivos (bem, na verdade apenas para aqueles que ele reconhece), o GNU/Linux possui uma única estrutura de árvore para os arquivos, e cada sistema de arquivos está ***montado*** em algum local desta árvore.

Assim como o Windows® precisa de um drive `C:`, o GNU/Linux deve montar a raiz de sua árvore de arquivos (`/`) em uma partição que contém o ***sistema de arquivo raiz***. Uma vez montada a raiz você pode montar outros sistemas de arquivos na árvore em vários ***pontos de montagem*** dentro dela. Qualquer diretório abaixo da estrutura raiz pode funcionar como um ponto de montagem, e você pode montar o mesmo sistema de arquivos diversas vezes em diferentes pontos de montagem.

Isto permite uma grande flexibilidade de configuração. Por exemplo, se vocês for configurar um servidor web, é recomendado dedicar uma partição inteira para o diretório onde ficarão os dados do servidor web. Este diretório, normalmente, é o `/var/www` e age como um ponto de montagem para a partição. Você deve considerar uma partição `/home` grande se planeja baixar bastante sofwares, armazenar trabalhos ou documentos pessoais, fotos, arquivos de músicas, etc... Você pode ver na Figura 6-2 e Figura 6-3 como fica o sistema antes e depois de montar os sistemas de arquivo.

1. Nosso exemplo assume que toda a árvore está construída em uma única partição.



Root file system
(already mounted)

File system containing files
of directory "/var/www"
(not yet mounted)

Figura 6-2. Um sistema de arquivos não montado ainda

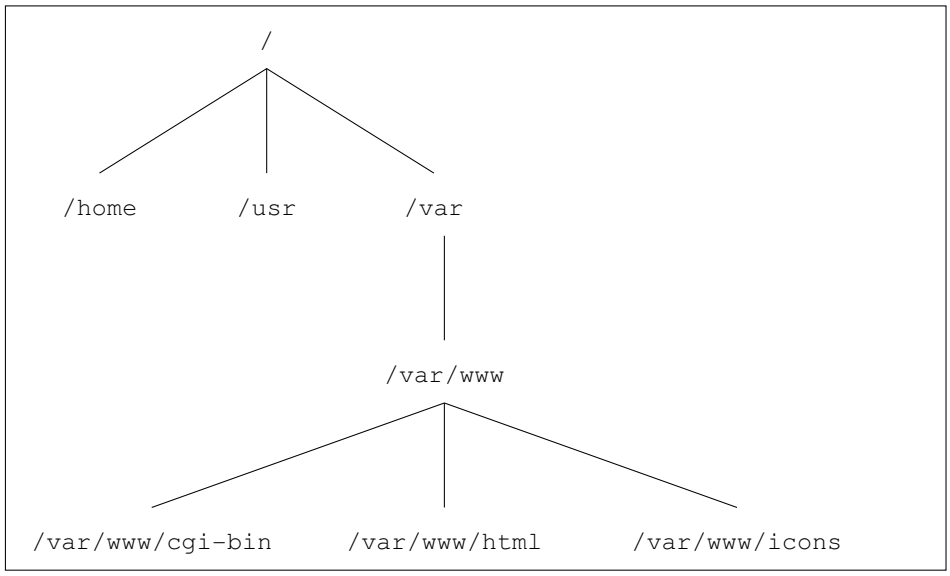


Figura 6-3. O sistema de arquivos agora está montado

Como você pode imaginar, isto oferece um número de vantagens: A estrutura da árvore será sempre a mesma, seja para um único sistema de arquivos ou extendido em vários. Esta flexibilidade permite mover a parte chave da estrutura da árvore para outra partição quando o espaço se tornar escasso, que é exatamente o que vamos fazer aqui.

Há duas coisas que você precisa saber sobre pontos de montagem:

1. O diretório que trabalhará como ponto de montagem deve existir.
2. E este diretório, preferencialmente, **deveria estar vazio**: se um diretório escolhido para ponto de montagem já contém arquivos e subdiretórios, estes ficarão "escondidos" pelo novo sistema de arquivos montado. Os arquivos não serão apagados, mas também não estarão acessíveis até que você libere o ponto de montagem.

Na verdade é possível acessar as informações "escondidas" pelo sistema de arquivo montado. Você simplesmente pode montar o diretório escondido com a opção `--bind`. Por exemplo, se você montou um diretório em `/hidden/directory/` e quer acessar o seu conteúdo original em `/new/directory`, você teria que executar:

```
mount --bind /hidden/directory/ /new/directory
```

6.2. Particionando um Disco Rígido, Formatando uma Partição

Há duas coisas que você não pode esquecer enquanto lê esta sessão: um diso rígido está dividido em partições, e cada uma destas partições possui um sistema de arquivos. O seu disco rígido novo não possui nenhum dos dois, então nós começamos com o particionamento. Para proceder, você deverá estar logado como `root`.

Primeiro você tem que saber o "nome" do disco rígido. (i.e.: para qual arquivo ele está designado). Suponha que o novo drive está configurado como na IDE primária. Neste caso, ele será conhecido pelo sistema como `/dev/hdb`.[2]. Por favor, recorra ao capítulo *Gerenciando Partições* do *Guia do Usuário*, que explicará como particionar um disco. O DiskDrake também irá criar o sistema de arquivos para você, então, assim que o particionamento e criação do sistema de arquivos estiverem concluídos, nós podemos prosseguir.

6.3. Os Comandos mount e umount

Agora que o sistema de arquivos foi criado, você pode montar a partição. Inicialmente ela estará vazia, já que não foi possível ter acesso ao sistema de arquivos para adicionar arquivos nele. O comando para montar um sistema de arquivo é o `mount`, e a sua sintaxe é a seguinte:

```
mount [opções] <-t tipo> [-o opções de montagem] <dispositivo> <ponto de montagem>
```

Neste caso nós queremos montar nossa partição temporariamente em `/mnt/new` (ou qualquer outro ponto de montagem que você tenha escolhido: lembre-se que o ponto de montagem deve existir). O comando para montar a nossa partição recém-criada é:

```
$ mount -t ext3 /dev/hdb1 /mnt/new
```

A opção `-t` serve para especificar que tipo de sistema de arquivos é utilizado pela partição. Os sistemas de arquivos que você irá encontrar com mais freqüência são o ext2FS (O sistema de arquivos do GNU/Linux) ou o ext3FS (uma versão aperfeiçoada do ext2FS com capacidades de journaling), VFAT (para quase todas as partições DOS/Windows®: FAT 12, 16 ou 32), NTFS (para versões novas do Windows®) e ISO9660 (sistema de arquivos para CD-ROM). Se você não especificar qualquer tipo, o `mount` irá tentar adivinhar qual sistema de arquivos é utilizado pela partição através da leitura do superblock.

A opção `-o` é utilizada para especificar uma ou mais opções de montagem. As opções apropriadas irão depender do sistema de arquivos que está sendo utilizado. Verifique a man page do mount(8) para mais detalhes.

Agora que você montou a sua nova partição, é hora de copiar todo o diretório `/usr` para ela:

```
$ (cd /usr && tar cf - .) | (cd /mnt/new && tar xpvf -)
```

Agora que os arquivos estão copiados, nós podemos desmontar a nossa partição. Para fazer isto, utilize o comando `umount`. A sintaxe é simples:

```
umount <ponto de montagem|dispositivo>
```

Então para desmontar a nossa nova partição nós podemos digitar:

```
$ umount /mnt/new
```

ou:

```
$ umount /dev/hdb1
```

2. Determinar o nome de um disco é explicado na Seção 2.2.

Às vezes pode acontecer de um dispositivo (normalmente o CD-ROM) estar ocupado. Se isto acontecer a maior parte dos usuários resolveria este problema reiniciando o seu computador. Porém, se o `umount /dev/hdc` falhar, então você pode tentar o `umount` "preguiçoso". A sintaxe é bastante simples:

```
umount -l <ponto de       montagem|dispositivo>
```

Este comando desconecta o dispositivo e fecha todos os acessos abertos a ele que sejam possíveis. Normalmente você pode ejetar o disco utilizando o comando `eject <ponto de montagem|dispositivo>`. Se o comando `eject` não funcionar e você não quiser reiniciar o seu computador, utilize o `umount -l`.

Já que esta partição irá se "tornar" nosso diretório `/usr`, nós precisamos informar isto ao sistema. Vamos então editar o arquivo `/etc/fstab`. Ele torna possível a automatização da montagem de certos sistemas de arquivo, especialmente na inicialização do sistema. Ele contém várias linhas descrevendo os sistemas de arquivo, seus pontos de montagem e outras opções. O arquivo se parece com o seguinte:

```
/dev/hda2 / ext3 defaults 1 1
/dev/hdd /mnt/cdrom auto umask=0022,user,iocharset=utf8,noauto,ro,exec,users 0 0
/dev/fd0 /mnt/floppy supermount dev=/dev/fd0,fs=ext2:vfat,--,umask=0022,iocharset=utf8,sync 0 0
/dev/hda1 /mnt/windows ntfs umask=0,nls=utf8,ro 0 0
none /proc proc defaults 0 0
/dev/hda3 swap swap defaults 0 0
```

Em cada linha encontramos:

- o dispositivo que possui o sistema de arquivos;
- o ponto de montagem;
- o tipo do sistema de arquivo;
- as opções de montagem;
- o ***parâmetro*** do utilitário de backup `dump`;
- ordem de verificação do `fsck` (*FileSystem CheckK*).

Há **sempre** uma entrada para o sistema de arquivo raiz. As partições Swap são especiais, já que não são visíveis na estrutura da árvore, e o campo do ponto de montagem para esta partição contém a palavra-chave `swap`. Assim como o sistema de arquivos `/proc`, ele será descrito com mais detalhes no Capítulo 5. Outro sistema de arquivo especial é o `/dev/pts`.

Note também que o seu sistema deve ter entradas adicionadas e removidas automaticamente deste arquivo. Isto é feito através do `fstab-sync`, um comando que recebe eventos especiais do sistema da Camada de Abstração de Hardware (HAL), e manipula o arquivo `/etc/fstab`. Dê uma olhada na página do manual fstab-sync(8) para mais detalhes.

Voltando para a nossa mudança no sistema de arquivos, neste ponto nós movemos toda a hierarquia de `/usr` para `/dev/hdb1` e nós queremos que esta partição seja montada como `/usr` quando o computador iniciar. Para efetuar isto, adicione a seguinte entrada em qualquer lugar do arquivo `/etc/fstab`:

```
/dev/hdb1 /usr ext3 defaults 1 2
```

Agora a partição será montada sempre que o seu computador iniciar, e terá verificação de erros quando necessário.

Se a sua partição não é do tipo ext3FS você terá que alterar para o tipo correto. As opções mais comuns são `ext2` e `reiserfs`. Note também que o último campo tem o valor `2`. Isto significa que a partição será verificada após todas as outras entradas com o valor `1`, e depois de outros sistemas de arquivos com a mesma prioridade no mesmo disco rígido que apareçam antes no arquivo `/etc/fstab`. Somente a partição root (`/`) deve ter o valor `1`.

Há duas opções especiais: `noauto` e `users`. A opção `noauto` especifica que o sistema de arquivos não deve ser montado na inicialização, e é montado somente quando você ordenar. A opção `users` diz que qualquer usuário pode montar e desmontar o sistema de arquivos. Estas duas opções são normalmente utilizadas para os drives de CD-ROM e disquete. Há também outras opções para o `/etc/fstab` que você pode encontrar na página do manual (fstab(5)).

Uma vantagem em usar o `/etc/fstab` é que ele simplifica a sintaxe do comando `mount`. Para mountar um sistema de arquivos descrito no `/etc/fstab`, você pode indicar o ponto de montagem ou o dispositivo. Para montar um disquete, por exemplo, você pode utilizar:

```
$ mount /mnt/floppy
```

ou:

```
$ mount /dev/fd0
```

Para finalizar, vamos revisar o que nós já fizemos. Nós copiamos o `/usr` e modificamos o `/etc/fstab` para que a nova partição fosse montada durante a inicialização. Mas no mometo, os arquivos do `/usr` antigo ainda estão no seu local original no drive, então nós precisamos apagá-los para liberar espaço (que era, antes de mais nada, o nosso objetivo inicial).

- Para fazer isto, você primeiro deve mudar para o modo monousuário, executando o comando `telinit 1` na linha de comando. Ela irá parar todos os serviços e prevenir que usuários conectem à máquina.
- Depois, nós apagamos todos os arquivos no diretório `/usr`. Lembre-se que nós ainda estamos falando do diretório "antigo", já que o novo, e maior, ainda não está montado: `rm -Rf /usr/*`.
- Finalmente, nós montamos o novo diretório `/usr`: `mount /usr`.

Agora, volte ao modo multi-usuário (`telinit 3` para o modo texto ou `telinit 5` para o modo gráfico) e, se não tiver nenhuma tarefa administrativa para realizar, você deve deixar de usar a conta do usuário `root`.

Capítulo 7. Introdução a Linha de Comando

No Capítulo 1 você aprendeu como executar um shell. Neste capítulo, iremos mostrar como trabalhar com o shell.

A maior propriedade do shell é o número de utilitários existentes: existem milhares deles, cada utilitário dedicado a executar uma tarefa especial. Iremos ver apenas um pequeno número desses utilitários. Uma das grandes vantagens do UNIX® é a possibilidade de combinar esses utilitários como veremos a seguir.

7.1. Utilitários para Tratamento de Arquivos

Nesse contexto, tratamento de arquivos significa copiar, mover e apagar arquivos. Mais tarde veremos maneiras de mudar os atributos dos arquivos (dono, permissões).

7.1.1. mkdir, touch: Criando Diretórios e Arquivos Vazios

O programa `mkdir` (*MaKe DIRectory*) é utilizado para criar diretórios. Sua sintaxe é simples:

```
mkdir [options] <diretório> [diretório ...]
```

Apenas uma das opções do `mkdir` merece destaque: a opção `-p`. Ela faz duas coisas:

1. cria os diretórios pai caso eles não existam previamente, ou seja, cria toda a estrutura de diretórios anterior ao diretório especificado, mesmo que essa não exista. Sem essa opção, o `mkdir` simplesmente irá falhar, dizendo que esse diretório não existe;
2. retorna silenciosamente caso o diretório a ser criado já exista. De maneira análoga, se você não especificar a opção `-p`, o `mkdir` irá retornar uma mensagem de erro dizendo que o diretório já existe.

Alguns exemplos:

- `mkdir foo`: cria o diretório `foo` no diretório corrente;
- `mkdir -p images/misc docs`: cria o diretório `misc` no diretório `images`. Primeiro cria o diretório `images` se ele não existir, observe a opção `-p` e a seguir cria diretório `misc`; também cria o diretório `docs` no diretório corrente.

Inicialmente o comando `touch` não foi concebido para criar arquivos, mas para alterar as datas de acesso e modificação do arquivo[1]. Entretanto o comando `touch` irá criar os arquivos listados como arquivos vazios caso eles não existam. A sintaxe do comando `touch` é:

```
touch [opções] arquivo [arquivo...]
```

Executando o comando:

```
touch arquivo1 images/arquivo2
```

serão criados dois arquivos vazios, o `arquivo1` no diretório corrente e o arquivo `arquivo2` no diretório `images`, caso esses arquivos não existam previamente.

1. No sistema UNIX®, existem três estampas de tempo distintas para cada arquivo: a data do último acesso ao arquivo (`atime`), isto é, a data da última vez que o arquivo foi aberto para leitura ou escrita; a data da última modificação dos atributos do *inode* (`ctime`); e finalmente a data da última modificação do **conteúdo** do arquivo (`mtime`).

7.1.2. rm: Apagando Arquivos ou Diretórios

O comando `rm` (*ReMove*) é equivalente aos comandos `del` e `deltree` do sistema DOS, só que com mais opções. Sua sintaxe é a seguinte:

```
rm [opções] <arquivo|diretório> [arquivo|diretório...]
```

Opções:

- `-r`, ou `-R`: apaga recursivamente. Esta opção é **obrigatória** para apagar um diretório, esteja ele vazio ou não. Entretanto, você pode utilizar o comando `rmdir` para apagar um diretório vazio.
- `-i`: pede confirmação antes de apagar cada arquivo. Note que, por razões de segurança, por padrão no Mandriva Linux, o comando `rm` é um ***alias*** para `rm -i`, (da mesma maneira existem *aliases* para os comandos `cp` e `mv`). A utilidade desses *aliases* pode variar de acordo com a sua utilização. Se desejar removê-los, você pode criar um arquivo `~/.alias` vazio, que irá evitar que as configurações globais do sistema sejam aplicadas à sua conta. Alternativamente você pode editar seu arquivo `~/.bashrc` e desabilitar alguns dos *aliases* globais adicionando a seguinte linha: `unalias rm cp mv`
- `-f`, exatamente o oposto da opção `-i`, a opção `-f` força a remoção dos arquivos ou diretórios, mesmo que o usuário não tenha acesso de escrita nos arquivos[2].

Alguns exemplos:

- `rm -i images/*.jpg arquivo1`: apaga todos os arquivos cujos nomes terminem em `.jpg` no diretórios `images` e apaga o arquivo `arquivo1` no diretório corrente, pedindo confirmação para apagar cada arquivo. Responda com a tecla **y** para confirmar a remoção, ou com a tecla **n** para cancelar.
- `rm -Rf images/misc/ arquivo*`: apaga, sem pedir confirmação, todo o conteúdo do diretório `misc/` no diretório `images/`, junto com todos os arquivos do diretório corrente cujos nomes começem com `arquivo`.

Usando o comando `rm` a remoção dos arquivos é **irreversível**. Não existe nenhuma maneira de restaurá-los! (Bem, atualmente existem várias maneiras para fazer isso, mas não é uma tarefa trivial e normalmente necessita de uma preparação prévia do sistema antes da remoção do arquivo.) Não hesite em usar a opção `-i` para garantir que você não irá apagar nenhum arquivo por engano.

7.1.3. mv: Movendo ou Renomeando Arquivos

A sintaxe do comando `mv` (*MoVe*) é a seguinte:

```
mv [opções] <arquivo|diretório> [arquivo|diretório ...] <destino>
```

Observe que quando você move vários arquivos o destino deve ser um diretório. Para renomear um arquivo, você simplesmente move-o para o novo nome.

Algumas opções:

- `-f`: força a operação — nenhum aviso é dado no caso de algum arquivo existente esteja para ser sobrescrito.
- `-i`: exatamente o oposto. Pergunta ao usuário, requisitando confirmação antes de sobrescrever algum arquivo.
- `-v`: modo ***verborrágico*** ou extenso (*verbose*) indica todas as mudanças e atividades.

Alguns Exemplos:

- `mv -i /tmp/pics/*.png .`: move todos os arquivos que estejam no diretório `/tmp/pics/` cujos nomes terminem com `.png` para o diretório corrente (`.`), pedindo confirmação antes de sobrescrever qualquer arquivo existente.

2. É suficiente ter acesso de escrita no **diretório** para que o usuário possa apagar arquivos dentro dele, mesmo que não seja o proprietário dos arquivos.

- `mv foo bar`: renomeia o arquivo arquivo `foo` para `bar`. Se `bar` for um diretório existente, o resultado desse comando será mover o arquivo `foo` ou todo o diretório (o diretório e todos os arquivos pertencentes a ele, recursivamente) para dentro do diretório `bar`.
- `mv -vf arquivo* images/ trash/`: move, sem pedir confirmação, todos os arquivos do diretório corrente cujos nomes começem com `arquivo`, juntamente com o diretório `images/` para o diretório `trash/` e diretório, e mostre cada operação executada.

7.1.4. cp: Copiando Arquivos e Diretórios

O comando `cp` (*CoPy*) é equivalente aos comandos `copy` e `xcopy` do DOS, porém com mais opções. Sua sintaxe é a seguinte:

```
cp [opções] <arquivo|diretório> [arquivo|diretório ...] <destino>
```

Aqui temos as opções mais utilizadas do comando `cp`:

- `-R`: cópia recursiva; **obrigatória** para a cópia de um diretório, mesmo que seja um diretório vazio.
- `-i`: pede confirmação antes de sobrescrever qualquer arquivo que esteja para ser sobrescrito.
- `-f`: ao contrário da opção `-i`, sobrescreve qualquer arquivo existente sem pedir confirmação alguma.
- `-v`: modo verborrágico ou extenso, mostra todas as ações executadas pelo comando `cp`.

Alguns Exemplos:

- `cp -i /timages/* images/`: copia todos os arquivos no diretório `/timages/` para o diretório `images/` localizado no diretório corrente. Pede confirmação se um arquivo estiver para ser sobrescrito.
- `cp -vR docs/ /shared/mp3s/* mystuff/`: copia o diretório `docs` e todos os arquivos no diretório `/shared/mp3s` para o diretório `mystuff`.
- `cp foo bar`: cria uma cópia do arquivo `foo` com o nome `bar` no diretório corrente.

7.2. Tratando Atributos de Arquivo

Os comandos mostrados aqui são utilizados para mudar o dono ou grupo do dono de um arquivo ou ainda as suas permissões. Vimos as diferentes permissões que um arquivo pode ter na Capítulo 1.

7.2.1. chown, chgrp: Mudam o Dono ou o Grupo de Um ou Mais Arquivos

A sintaxe do comando `chown` (*CHange OWNer*) é mostrada a seguir:

```
chown [opções] <usuário[:grupo]> <arquivo|diretório> [arquivo|diretório...]
```

As opções do comando `chown` incluem:

- `-R`: Operação Recursiva. Utilizado para mudar o dono de todos os arquivos e subdiretórios em um dado diretório.
- `-v`: modo verborrágico. Mostra todas as ações executadas pelo comando `chown`; informa quais arquivos tiveram sua posse mudada como resultado do comando e quais arquivos não foram modificados.
- `-c`: semelhante à opção `-v`, porém, informa somente os arquivos que foram modificados.

Alguns Exemplos:

- `chown nobody /shared/book.tex`: muda o dono do arquivo `/shared/book.tex` para o usuário `nobody`.
- `chown -Rc usuario1:music *.mid concerts/`: muda a posse de todos os arquivos no diretório corrente cujos nomes terminem com `.mid` e de todos os arquivos e subdiretórios no diretório `concerts/` para o usuário `usuario1` e grupo `music`, informando somente quais arquivos foram afetados pelo comando.

O comando `chgrp` (*CHange GRouP*) permite que você modifique o grupo do dono do arquivo (ou arquivos); sua sintaxe é bem parecida com a sintaxe do comando `chown`:

```
chgrp [opções] <grupo> <arquivo|diretório> [arquivo|diretório...]
```

As opções para esse comando são as mesmas do comando `chown`, e ele é utilizado de uma maneira muito similar. Dessa forma, o comando `chgrp disk /dev/hd*` muda o grupo de todos os arquivos no diretório `/dev` cujos nomes começem com `hd` para o grupo `disk`.

7.2.2. chmod: Mudando as Permissões de Arquivos e Diretórios

O comando `chmod` (*CHange MODe*) tem uma sintaxe bem própria. Sua sintaxe geral é:

```
chmod [opções] <modo a ser modificado> <arquivo|diretório> [arquivo|diretório...]
```

Porém, o que distingue ele dos demais são as diferentes formas que o modo a ser modificado pode tomar. O modo pode ser especificado de duas maneiras:

1. Em formato octal. As permissões do dono, correspondem aos dígitos com o formato `<x>00`, onde `<x>` corresponde à permissão atribuída: 4 para permissão de leitura, 2 para permissão de escrita e 1 para permissão de execução. De maneira similar, as permissões do grupo tem o formato `<x>0` e as permissões para "outros" usuários tem o formato `<x>`. Então, tudo o que você têm a fazer é somar as permissões atribuídas para obter o modo correto. Assim sendo, as permissões `rwxr-xr--` correspondem a 400+200+100 (permissões do dono, `rwx`) +40+10 (permissões do grupo, `r-x`) +4 (permissões para outros, `r--`) = 754; desta maneira, as permissões são expressas em termos absolutos. Isso quer dizer que as permissões anteriores são substituídas incondicionalmente;
2. com expressões. Nesta forma, as permissões são expressas por uma seqüência de expressões separadas por vírgulas. Sendo assim, uma expressão tem a seguinte forma: `[categoria]<+|-|=><permissões>`.

   A categoria pode ser uma ou mais das seguintes:

   - `u` (*Usuário*, permissões para o dono);
   - `g` (*Grupo*, permissões para o grupo do dono);
   - `o` (*Outros*, permissões para os "outros").

   Se nenhuma categoria for especificada, as mudanças serão aplicadas a todas as categorias. Um `+` atribui a permissão, um `-` remove a permissão e um `=` atribui a permissão exatamente como foi passada na linha de comando. Finalmente, a permissão é uma ou mais das seguintes:

   - `r` (*Leitura*);
   - `w` (*Escrita*) ou;
   - `x` (*eXecução*).

As principais opções são similares às opções dos comandos `chown` e `chgrp`:

- `-R`: muda as permissões recursivamente.
- `-v`: modo verborrágico. Mostra as ações tomadas na execução para cada arquivo.
- `-c`: semelhante à opção `-v`, porém mostra somente os arquivos afetados pelo comando.

Exemplos:

- `chmod -R o-w /shared/docs`: remove recursivamente a permissão de escrita para `outros` em todos os arquivos e subdiretórios no diretório `/shared/docs/` .
- `chmod -R og-w,o-x private/`: recursivamente, remove a permissão de escrita para `grupo` e `outros` em todo o diretório `private/`. Remove também a permissão de execução para `outros` no diretório `private/`.

- `chmod -c 644 misc/arquivo*`: muda as permissões de todos os arquivos no diretório `misc/` cujos nomes começem com `arquivo` para `rw-r--r--` (i.e. permissão de leitura para todos e permissão de escrita somente para o dono do arquivo), e indica somente os arquivos afetados pela execução do comando.

7.3. Casamento de Padrões no Shell

Você provavelmente já utiliza o ***casamento*** de padrões sem mesmo conhecê-lo. Quando especifica um arquivo no Windows®, ou quando procura por um arquivo, você utiliza o caractere `*` para espedificar uma string variável. Por exemplo, `*.txt` "casa" com todos os arquivos cujos nomes terminem em `.txt`. Também utilizamonos dele várias vezes na última seção. Entretanto, existem muitos outros casamentos de padrão além do `*`.

Quando você digita um comando como `ls *.txt` e pressiona **Enter**, a tarefa de encontrar quais arquivos casam com o padrão `*.txt` não é feita pelo comando `ls`, mas sim pelo próprio shell. Aqui se faz necessária uma pequena explicação de como a linha de comando é interpretada pelo shell. Quando você digita:

```
$ ls *.txt
readme.txt  recipes.txt
```

a linha de comando é inicialmente dividida em palavras (`ls` e `*.txt` nesse exemplo). Quando o shell encontra um `*` em uma palavra, ele irá interpretar a palavra inteira como um casamento de padrões e irá substituir ela pelos nomes de todos os arquivos que casem com aquele padrão. Então, o comando, exatamente antes de ser executado pelo shell se tornou `ls readme.txt recipe.txt`, que retorna o resultado esperado. Outros caracteres fazem com que o shell também reaja dessa maneira:

- `?`: casa com um e somente um caractere, independente de qual caractere seja;
- `[...]`: casam qualquer caractere encontrado nos parênteses. Os caracteres podem referenciar ou uma faixa de valores (i.e. `1-9`) ou ***valores discretos***, ou ainda ambos. Por exemplo: `[a-zBE5-7]` irá casar todos os caracteres que estejam entre `a` e `z`, um `B`, um `E`, um `5`, um `6` ou um `7`;
- `[!...]`: casa qualquer caractere que **não** esteja dentro dos colchetes. `[!a-z]`, por exemplo, irá casar qualquer caractere que não seja uma letra minúscula[3];
- `{c1,c2}`: casa `c1` ou `c2`, onde `c1` e `c2`também são padrões de casamento, o que quer dizer que você pode escrever, por exemplo, `{[0-9]*,[acr]}`.

Alguns padrões e seus significados:

- `/etc/*conf`: todos os arquivos no diretório `/etc` cujos nomes terminem em `conf`. Pode casar com `/etc/inetd.conf`, `/etc/conf.linuxconf`, **e também** `/etc/conf` se esse arquivo existir. Lembre-se que o `*` também pode casar com qualquer string vazia.
- `image/{cars,space[0-9]}/*.jpg`: todos os nomes que terminem em `.jpg` nos diretórios `image/cars`, `image/space0`, (...), `image/space9`, caso esses diretórios existam.
- `/usr/share/doc/*/README`: todos os arquivos com nome `README` em todos os subdiretórios de primeiro nível abaixo do diretório `/usr/share/doc`. Irá casar com `/usr/share/doc/mandriva/README`, porém, não irá casar com `/usr/share/doc/myprog/doc/README`.
- `*[!a-z]`: todos os arquivos com nomes que **não** terminem com uma letra mínuscula no diretório corrente.

3. Cuidado! Enquanto que isso é verdade para a maioria das linguagens, pode não ser verdade para a sua própria configuração de linguagem (`localização`). Isso depende da **ordem de combinação**. Em algumas configurações de linguagens, `[a-z]` irá casar a, A, b, B, (...), z. E lembre-se do fato que algumas linguagens também tem caracteres acentuados ...

7.4. Redirecionamentos e Pipes

7.4.1. Um Pouco Mais Sobre Processos

Para compreender o princípio dos redirecionamentos e pipes precisamos explicar uma noção a respeito de processo que ainda não foi apresentada. A maioria dos processos UNIX® (incluindo também as aplicações gráficas mas exclui a maioria dos daemons) usam um mínimo de três descritores de arquivo: entrada padrão, saída padrão e saída de erro padrão. Seus números respectivos são 0, 1 e 2. Em geral, esses três descritores estão associados com o terminal no qual o processo foi iniciado, com a entrada sendo a partir do teclado. O objetivo de redirecionamentos e pipes é para redirecionar esses descritores. Os exemplos nesta seção irão ajudá-lo a compreender melhor esse conceito.

7.4.2. Redirecionamentos

Imagine, por exemplo, que você quer uma lista de arquivos que terminem em `.png`[4] no diretório `images`. Essa lista é muito grande, então você quer armazená-la em um arquivo para vê-la com mais calma mais tarde. Você pode digitar o seguinte comando:

```
$ ls images/*.png 1>file_list
```

A saída padrão desse comando (`1`) é redirecionada (`>`) para o arquivo com nome `file_list`. O operador `>` é o operador de redirecionamento de saída. Se o arquivo para o qual o redirecionamento foi apontado não existir, ele é criado, mas, se ele já existir, seu conteúdo anterior é sobrescrito. Entretanto, o descritor padrão, redirecionado pelo operador é o descritor da saída padrão e não necessita ser especificado na linha de comando. Então, podemos simplesmente escrever:

```
$ ls images/*.png >file_list
```

e o resultado será exatamente o mesmo. Então, a seguir você pode analisar o conteúdo do arquivo utilizando um visualizador de arquivos como o comando `less`.

Agora, imagine que você quer saber quantos desses arquivos existem. Ao invés de contá-los manualmente, você pode usar o utilitário chamado `wc` (*Contador de Palavras - Word Count*) com a opção `-l`, que escreve na saída padrão o número de linhas no arquivo. Uma solução é a seguinte:

```
$ wc -l 0<file_list
```

e esse comando irá retornar o resultado desejado. O operador `<` é o operador de redirecionamento de entrada, e o descritor redirecionado por padrão é a entrada padrão, i.e. `0`, e você simplesmente precisa escrever a seguinte linha de comando:

```
$ wc -l <file_list
```

Agora imagine que você quer remover todas as "extensões" de arquivo e colocar o resultado em um outro arquivo. Uma ferramenta para fazer isso é o comando `sed` (*Editor de Fluxos - Stream EDitor*). Você simplesmente redireciona a entrada padrão do comando `sed` para o arquivo `file_list` e redireciona sua saída para o arquivo de resultados, i.e. `the_list`:

```
$ sed -e 's/\.png$//g' <file_list >the_list
```

e a sua lista está criada, pronta para ser vista com qualquer visualizador a qualquer hora.

Pode ser útil também redirecionar a saída padrão de erros. Por exemplo, você quer saber em quais diretórios abaixo do diretório `/shared` você não consegue acessar: uma solução é listar o diretório recursivamente e redirecionar os erros para um arquivo, enquanto não mostra na saída padrão:

```
$ ls -R /shared >/dev/null 2>errors
```

4. Você pode pensar que é loucura dizer "arquivos terminados em `.png`" ao invés de "imagens PNG". Entretanto, mais uma vez, os arquivos sob o UNIX® somente tem uma extensão por convenção: as extensões não são, de maneira alguma, uma forma de definir um tipo de arquivo. Um arquivo cujo nome termine em `.png` pode perfeitamente ser uma imagem JPEG, um arquivo de uma aplicação, um arquivo texto ou qualquer outro tipo de arquivo. O mesmo é verdade sob Windows®!

a saída padrão será redirecionada (>) para `/dev/null`, um arquivo especial no qual, qualquer coisa que seja escrita nele será descartada (i.e. a saída padrão não é mostrada) e a saída de erros padrão (`2`) é redirecionada (>) para o arquivo `errors` .

7.4.3. Pipes

Os "pipes" (termo em inglês) são, de algumas maneiras uma combinação dos redirecionamentos de entrada e saída padrão. O princípio é como se fosse um cano físico, daí o nome (pipe é cano em inglês): um processo envia dados em uma ponta do pipe e o outro processo lê os dados na outra ponta. O operador pipe é representado pelo caractere `|`. Voltemos então ao exemplo anterior, o da lista de arquivos. Suponha que você quer encontrar diretamente quantos arquivos existem no diretório, sem armazenar a lista em um arquivo temporário. Para executar essa ação, digite o seguinte comando:

```
$ ls images/*.png | wc -l
```

que irá passar a saída padrão do comando `ls` (i.e. a lista dos arquivos) redirecionando-a para a entrada padrão do comando `wc`. Esse comando dará o resultado esperado.

Você também pode gerar um arquivo de saída com uma lista de nomes de arquivo "sem extensões" utilizando o seguinte comando:

```
$ ls images/*.png | sed -e 's/\.png$//g' >the_list
```

ou, se você quiser visualizar a lista diretamente, sem armazená-la em um arquivo:

```
$ ls images/*.png | sed -e 's/\.png$//g' | less
```

Os pipes e redirecionamentos não estão restritos somente a texto legível. Por exemplo, o seguinte comando digitado a partir de um Terminal:

```
$ xwd -root | convert - ~/my_desktop.png
```

irá gravar uma foto da tela no arquivo `my_desktop.png` [5] no seu diretório pessoal.

7.5. Auto Completar na Linha de Comando

A função de ***completar*** é muito útil e todos os shells modernos (incluindo o bash) tem ela embutida. A sua finalidade é que o usuário tenha o mínimo de trabalho possível. A melhor maneira de ilustrar a função de completar é dando um exemplo.

7.5.1. Exemplo

Imagine que seu diretório pessoa contém um arquivo com o nome `arquivo_com_nome_muito_grande_impossivel_de_digitar` e você quer ver seu conteúdo. Suponha então, que você tenha no mesmo diretório um outro arquivo chamado `arquivo_texto`. Você está no seu diretório pessoal, então digite a seguinte seqüência de comandos:

```
$ less ar<TAB>
```

(i.e., digite `less ar` e então pressione a tecla **Tab**). O shell irá expandir a linha de comando para você:

```
$ less arquivo_
```

e também dar-lhe uma lista das possíveis escolhas (na sua configuração padrão, que pode ser modificada). Então, digite a seguinte seqüência de teclas:

```
$ less arquivo_c<TAB>
```

e o shell irá expandir a linha de comando para dar o resultado que você espera:

5. sim, ele realmente será uma imagem PNG (de qualquer maneira, o pacote ImageMagick precisa estar instalado para executar esse comando).

```
$ less arquivo_com_nome_muito_grande_impossivel_de_digitar
```

Tudo o que você precisa é então pressionar **Enter** para confirmar e ler o arquivo.

Use the **q** key to exit the file viewer.

7.5.2. Outros Métodos de Completar

A tecla **Tab** não é a única maneira de ativar a função completar, embora seja a maneira mais comum. Como regra geral, a palavra a ser completada deve ser um nome de comando para a primeira palavra na linha de comandos (`nsl<TAB>` retornará `nslookup`), e um nome de arquivo para todas as outras posições, a não ser que uma palavra seja precedida por um caractere "magico" como `~`, `@` ou `$`, casos em que o shell irá tentar completar com um nome de usuário, um nome de máquina ou uma variável de ambiente respectivamente.[6]. Existe também um caractere mágico para completar um nome de arquivo (`/`) e um comando para recuperar um comando do histórico (`!`).

As outras duas maneiras de ativar a função de completar são as seqüências **Esc-<x>** e **Ctrl-X-<x>**, onde **<x>** é um dos caracteres mágicos já mencionados. **Esc-<x>** irá tentar retornar com uma única completude. Se falhar, irá completar a palavra com a maior subpalavra na lista de opções. Um ***beep*** indica que a opção não é única, ou simplesmente que não existe nenhuma opção correspondente. A seqüência **Ctrl-X-<x>** mostra a lista de possíveis escolhas sem tentar nenhuma das possibilidades. Pressionar a tecla **Tab** é semelhante a pressionar sucessivamente as seqüências de teclas **Esc-<x>** e **Ctrl-X-<x>**, onde o caractere mágico depende do contexto.

Thus, one way to see all the environment variables defined is to type the sequence **Ctrl-X-$** on a blank line. Another example: if you want to see the man page for the `nslookup` command, you simply type `man nsl` then **Esc-!**, and the shell will automatically complete the command to `man nslookup`.

7.6. Iniciando e Tratando Processo em Segundo Plano: Controle de Trabalhos

Você provavelmente notou que ao entrar com um comando no Terminal, você normalmente tem que esperar comando terminar antes de o shell retornar o controle para você. Isso indica que você mandou executar o comando em ***primeiro plano***. Entretanto, existem ocasiões e em que esse não é o comportamento desejado.

Suponha, por exemplo, que você decida copiar recursivamente um diretório grande para um outro. Você também decidiu ignorar os erros, então redirecionou a saída de erros para `/dev/null`:

```
cp -R images/ /shared/ 2>/dev/null
```

Um comando como esse pode demorar vários minutos até terminar de ser executado. Você então tem duas soluções: a primeira é violenta, e implica em parar (matar) o comando e reexecutá-lo quando tiver tempo. Para fazer isso, pressione as teclas **Ctrl-C**: isso irá terminar o processo e levá-lo de volta ao prompt. Mas espere, não faça-o ainda! Leia mais um pouco.

Suponha que você quer que o comando continue executando enquanto faz alguma outra coisa. A solução então é colocá-lo para executar em ***segundo plano***. Para fazer isso, pressione as teclas **Ctrl-Z** para suspender o processo:

```
$ cp -R images/ /shared/ 2>/dev/null
# Type C-z here
[1]+  Stopped                 cp -R images/ /shared/ 2>/dev/null
$
```

e novamente você é levado ao prompt. O processo está parado, esperando que seja reiniciado (como mostrado pela palavra-chave `Stopped`). Isso é exatamente o que deseja fazer, porém, é claro, em segundo plano. Digite o comando `bg` (de *Segundo Plano - BackGround*) para executar o processo em segundo plano:

```
$ bg
```

6. Lembre-se: o sistema UNIX® diferencia caixa alta e caixa baixa. As variáveis de ambiente `HOME` e `home` não são a mesma variável.

```
[1]+ cp -R images/ /shared/ 2>/dev/null &
$
```

O processo então irá começar a executar em segundo plano, como indicado pelo caractere `&` (e comercial) ao final da linha. Você então será levado novamente ao prompt e poderá continuar trabalhando. Um processo que executa como tarefa de segundo planou, ou em segundo plano, é chamado de ***job*** ou trabalho de segundo plano.

É claro, você pode iniciar processos diretamente como tarefas de segundo plano, simplesmente adicionando um caractere **&** ao final da linha de comando. Por exemplo, você pode iniciar um comando para copiar um diretório em segundo plano simplesmente digitando:

```
$ cp -R images/ /shared/ 2>/dev/null &
```

Se você quiser, também pode restaurar esse processo para primeiro plano e esperar ele terminar, utilizando o comando `fg` (*Primeiro Plano - ForeGround*). Para colocá-lo em segundo plano novamente, digite as teclas **Ctrl**-**Z**, `bg`.

Você pode iniciar vários processos dessa maneira: cada comando terá um número de trabalho (job). O comando do shell `jobs` lista todos os trabalhos associados com o shell corrente. O trabalho precedido de um caractere + indica o último processo que foi iniciado como tarefa de segundo plano. Para restaurar um trabalho particular no primeiro plano, você pode digitar `fg <n>` onde `<n>` é o número do trabalho ou job, i.e. `fg 5`.

Observe que você também pode suspender ou iniciar aplicações de ***tela cheia*** desta maneira, como o comando `less` ou um editor de texto como o Vi, e restaurá-los no primeiro plano se desejar.

7.7. Algumas Palavras ao Final

Como você pôde ver, o shell é muito geral e seu uso efetivo é uma questão de prática. Neste capítulo relativamente longo, nós mencionamos somente alguns dos comandos disponíveis: o Mandriva Linux tem milhares de utilitários, e mesmo o mais experiente dos usuários não faz uso de todos eles.

Existem utilitários para todos os gostos e propósitos: você tem utilitários para tratamento de imagens (como o comando `convert` mencionado anteriormente, mas também tem o GIMP modo de ***lote*** e muitos utilitários para gerenciamento de ***imagens***), som (Ogg Vorbis codificadores, CD de áudio; tocadores), gravadores de CD, clientes de e-mail, clientes de FTP e mesmo navegadores www (como `lynx` ou `links`), sem mencionar todas as ferramentas administrativas.

Mesmo que aplicações gráficas com funções equivalentes existam, normalmente elas são interfaces gráficas construídas sobre esses mesmos utilitários. Além disso, os utilitários de linha de comando tem a vantagem de operar em um modo não interativo: você ode iniciar a gravação de um CD e então sair do sistema com a confiança de que a gravação será executada (veja a página de manual do comando nohup(1) man ou a página de manual do comando screen(1) man page).

Capítulo 8. Edição de Textos: Emacs e VI

Como declarado na introdução, a edição de textos[1] é uma funcionalidade fundamental quando se usa um sistema UNIX®. Os dois editores que nós vamos dar uma olhada rápida são um pouco difícil de usar inicialmente, mas assim que você souber o básico, ambos provarão ser ferramentas muito poderosas. Isto se deve à grande variedade de modos de edição disponíveis, que fornecem funcionalidades de edição específicas para vários tipos de arquivo (Perl, C++, XML, etc.).

8.1. Emacs

O Emacs é provavelmente o editor de textos mais poderoso que existe. Ele pode fazer quase tudo e é infinitamente extensível através da linguagem de programação baseada em lisp que ele possui embutida. Com o Emacs, você pode navegar na internet, ler o seu e-mail, participar de grupos de notícias Usenet, fazer café, e muito mais. Isto não significa que você irá aprender como fazer tudo isto neste capítlo, mas já vai ser um bom começo abrir o Emacs, editar um ou mais arquivos, salvá-los e então sair do editor.

Se, depois de ler este capítulo você quiser aprender mais sobre o Emacs, você pode dar uma olhada no Tutorial de Introdução ao Emacs (`http://www.lib.uchicago.edu/keith/tcl-course/emacs-tutorial.html`).

8.1.1. Pequena Apresentação

Você pode executar o Emacs da seguinte maneira na linha de comando:

```
emacs [arquivo1] [arquivo2...]
```

O Emacs irá abrir cada arquivo indicado como argumento em um buffer separado. Se mais de dois arquivos forem indcados na linha de comando, a janela será automaticamente dividida em duas e, uma parte dela mostrará o último arquivo especificado, enquanto a outra parte mostrará uma lista dos buffers disponíveis com os outros aquivos. Se você executar o Emacs sem especificar nenhum arquivo na linha de comando, você será posicionado em um buffer chamado `*scratch*`. Quando o Emacs for executado no X, menus estarão disponíveis e poderão ser acessados com o mouse, já no modo texto, você pode acessar os menus com a tecla **F10**. Neste capítulo nós vamos nos concentrar em trabalhar com o teclado e sem qualquer menu.

8.1.2. Começando com o Emacs

Agora é hora de experimentarmos na prática. Para o nosso exemplo vamos começar abrindo dois arquivos, `arquivo1` e `arquivo2`. Se estes arquivos não existem, eles serão criados assim que você escrever alguma coisa neles:

```
$ emacs arquivo1 arquivo2
```

Digitando este comando, a seguinte janela será aberta:

1. "Editar texto" significa modificar o conteúdo de um arquivo contendo somente letras, dígitos e sinais de pontuação. Ele não contém informação de layout como fontes, etc. Arquivos como estes podem ser mensagens de e-mail, códigos-fonte, documentos ou até mesmo arquivos de configuração.

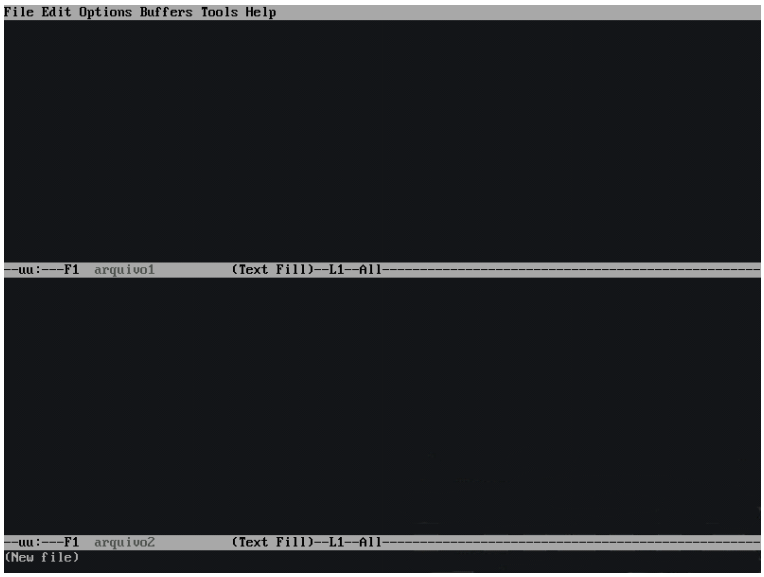


Figura 8-1. Editando dois arquivos ao mesmo tempo

Como você pode ver, dois buffers foram criados. Um terceiro é apresentado na parte de baixo da tela (onde está escrito `(New file)`). Esse é o mini-buffer. Este buffer não pode ser acessado diretamente, porém você será "convidado" pelo Emacs a escrever nele durante operações interativas. Para alterar o buffer atual, tecle **Ctrl**-**X**-**O**. Você digita textos da mesma forma que em um editor "normal", apagando caracteres com a tecla **Del** ou **Backspace**.

Para se movimentar dentro do texto, você pode utilizar as setinhas, ou então alguma das seguintes combinações: **Ctrl**-**A** para ir ao início da linha, **Ctrl**-**E** para ir ao fim da linha, **Alt-<** ou **Ctrl**-**Home** para ir ao início do buffer e **Alt->** ou **Ctrl**-**End** para ir ao fim do buffer. Há várias outras combinações, até mesmo uma para cada setinha[2].

Quando você estiver pronto para salvar as alterações em disco, digite **Ctrl**-**X** **Ctrl**-**S**, ou se você quiser salvar o conteúdo do buffer em outro arquivo, digite **Ctrl**-**X** **Ctrl**-**W**. O Emacs irá pedir que você informe o nome do arquivo onde o conteúdo deve ser salvo. Você pode utilizar ***completação*** para facilitar, pressionando a tecla **Tab** da mesma maneira que no bash.

8.1.3. Lidando com os buffers

Se você quiser, pode alternar para um modo em que apenas um buffer é exibido. Há duas maneiras de fazer isto:

- Se você está no buffer que quer esconder: digite **Ctrl**-**X** **0**.
- Se você está no bufer que vai permanecer na tela: digite **Ctrl**-**X** **1**.

Há duas maneiras de exibir uma buffer novamente na tela:

- digite **Ctrl**-**X** **B** e entre com o nome do buffer que você deseja, ou
- digite **Ctrl**-**X** **Ctrl**-**B**. Isto irá abrir um novo buffer chamado `*Buffer List*`. Você pode navegar neste buffer utilizando a seqüência **Ctrl**-**X** **O**, e então selecionar o buffer que você deseja e pressionar a tecla **Enter**, ou então digitar o nome do buffer no mini-buffer. O buffer `*Buffer List*` volta ao segundo plano assim que você tiver feito a sua escolha.

Se você já terminou um arquivo e quer se livrar do bufer associado a ele, digite **Ctrl**-**X** **K**. O Emacs então irá te perguntar qual buffer ele deve fechar. Por padrão, este será o buffer que você está localizado atualmente. Se você quer se livrar de um buffer diferente daquele sugerido, entre com o nome do buffer diretamente ou pressione **Tab**: O Emacs irá abrir ainda outro buffer chamado `*Completions*` dando a lista das escolhas possíveis. Confirme a escolha com a tecla **Enter**.

2. O Emacs foi desenvolvido para trabalhar em uma variedade grande de máquinas, algumas das quais não possuem as teclas de setas no teclado. Isto se aplica ainda mais ao Vi.

Você também pode recuperar dois buffers visíveis na tela a qualquer momento. Para fazer isto, digite **Ctrl**-**X** **2**. Por padrão, o novo buffer criado será uma cópia do buffer atual (que possibilita, por exemplo, editar várias partes de um arquivo grande em diversos lugares "ao mesmo tempo"). Para mover entre os buffers, utilize os comandos previamente mencionados.

Você pode abrir outros arquivos a qualquer momento, utilizando **Ctrl**-**X** **Ctrl**-**F**. O Emacs irá perguntar pelo nome do arquivo e você pode novamente utilizar a completação se você achar mais conveniente.

8.1.4. Copiar, Cortar, Colar, Procurar

Suponha que você se encontre na seguinte situação: Figura 8-2.

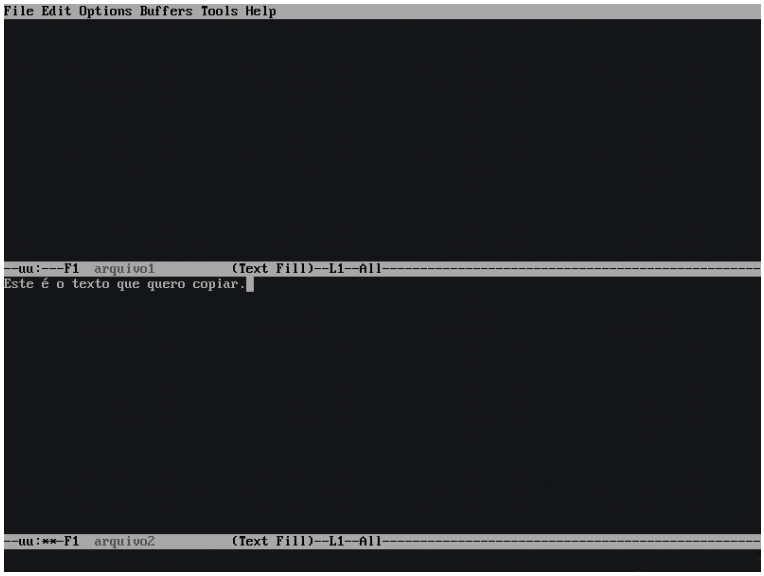


Figura 8-2. Emacs, antes de copiar o bloco de texto

Antes de mais nada, você precisará selecionar o texto que você quer copiar. Neste exemplo nós queremos copiar a sentença toda. O primeiro passo é colocar uma marca no início da área. Assumindo que o cursor está na posição mostrada na Figura 8-2, a seqüência de comando seria **Ctrl**-**Space** (**Control** + Barra de espaço). O Emacs irá mostrar a mensagem `Mark set` no mini-buffer. A seguir, mova para o início da linha com **Ctrl**-**A**. A área selecionada para copiar ou cortar é toda a área localizada entre a marca e a posição atual do cursor, então neste caso será toda a linha do texto. Há duas seqüências de comandos disponiveis: **Alt**-**W** (para copiar) ou **Ctrl**-**W** (para cortar). Se você copiar, o Emacs irá retornar rapidamente para a posição da marca, e assim você pode ver a área selecionada.

Finalmente, vá para o buffer onde você quer colocar o texto e tecle **Ctrl**-**Y**. Isto irá retornar o seguinte resultado:

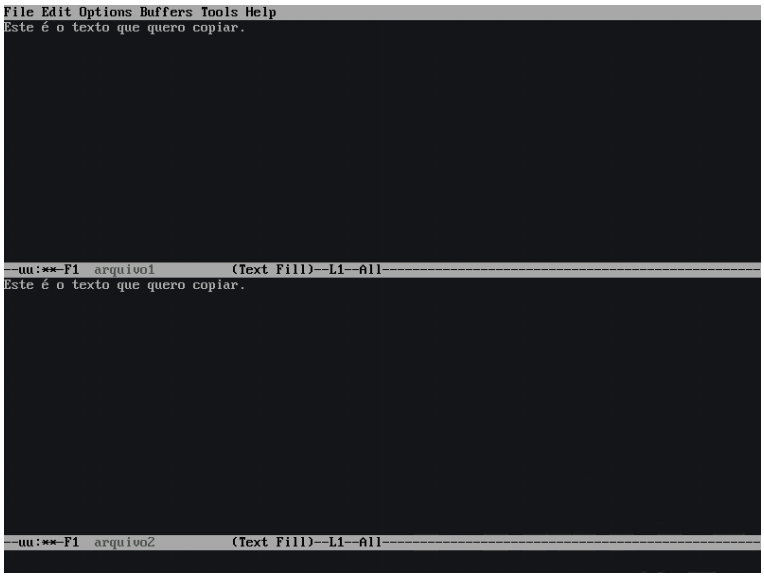


Figura 8-3. Copiando texto com o emacs

Na verdade, o que você fez foi copiar texto para o ***kill ring*** do Emacs. Este kill ring contém todas as as áreas copiadas ou recortadas desde que o Emacs iniciou. **Qualquer** área recém-copiada ou recortada é colocada no topo do kill ring. A sequência **Ctrl-Y** somente "cola" a área que estiver no topo. Se você quiser acessar qualquer uma das outras áreas, pressione **Ctrl-Y** e então **Alt-Y** até que você consiga o texto desejado.

Para procurar por um texto, vá para o bufer desejado e digite **Ctrl-S**. O Emacs irá solicitar a você a string pela qual deseja buscar. Para continuar com uma busca pela mesma string no buffer atual, pressione **Ctrl-S** novamente. Quando o Emacs atingir o final do buffer e não encontrar mais ocorrêcias, você pode digitar **Ctrl-S** novamente para reiniciar a busca do início do buffer. Pressionando **Enter** a busca será encerrada.

Para procurar e substituir, digite **Alt-%**. O Emacs irá pedir que você informe a string de procura, o que deve substituir esta string, e pedir uma confirmação para cada ocorrência que encontrar.

Digitar **Ctrl-X U** ou **Ctrl-Shift--** irá desfazer a última modificação que você fez no arquivo. Você pode desfazer quantas operações quiser.

8.1.5. Sair do emacs

A tecla de atalho para sair do Emacs é **Ctrl-X Ctrl-C**. Se você não salvou as alterações, o Emacs irá lheperguntar se você quer ou não salvar o conteúdo dos buffers.

8.2. Vi: o ancestral

O Vi foi o primeiro editor em tela cheia criado. É um dos principais alvos daqueles que não gostam do UNIX®, mas também é um dos melhores argumentos dos que defendem o sistema: apesar de ser complicado para aprender, ele é uma ferramenta extremamente poderosa uma vez que você adquira o hábito de utilizá-lo. Com pouco uso do teclado um usuário do Vi pode mover montanhas, e pouquíssimos editores, com exceção do Emacs, podem alegar fazer o mesmo.

Na verdade a versão oferecida com o Mandriva Linux é o Vim, que significa *VI iMproved* (VI melhorado), mas nós iremos nos referir a ele como Vi ao longo deste capítulo.

Se você quiser aprender mais sobre o Vi, você pode dar uma olhada no Hands-On Introduction to the Vi Editor (`http://www.library.yale.edu/wsg/docs/vi_hands_on/`)ou na página do Vim (`http://www.vim.org/`).

8.2.1. Modo de Inserção, Modo de Comando, Modo de Execução...

Para começar a utilizar o Vi nós utilizamos uma linha de comando semelhante aquela do Emacs. Então vamos voltar aos nossos dois arquivos e digitar:

```
$ vi arquivo1 arquivo2
```

Neste ponto, você vai estar olhando para uma janela parecida com esta:

Figura 8-4. Posição de início do VIM

Agora você está no ***modo de comando*** em frente do primeiro arquivo aberto. Neste modo você não pode inserir um texto em um arquivo. Para isto você precisa mudar para o ***modo de inserção***.

Aqui estão alguns atalhos para a inserção de texto:

Por favor, note que a combinação de teclas deve ser digitada exatamente como indicada neste capítulo, o Vi distingue letras maúsculas de letras minúsculas e, então, o comando **a** não é o mesmo que o comando **A**.

- **a** e **i**: para inserir texto depois e antes do cursor (**A** e **I** insere texto no final e no início da linha atual);
- **o** e **O**: para inserir texto acima e abaixo da linha atual.

No modo de inserção você verá a string `--INSERT--` aparecer na parte de baixo da tela (então você pode saber em que modo está). Este é o único modo em que você pode inserir texto. Para retornar ao modo de comando, pressione a tecla **Esc**.

No modo de inserção você pode utilizar as teclas **Backspace** e **Del** para apagar texto. As setas direcionais irão permitir que você se movimente no texto, tanto no Modo de Comando como no Modo de Inserção. No modo de comando temos também outra combinação de teclas que nós iremos ver mais tarde.

O modo de execução é acessado pressionando a tecla **:** no modo de comando. Um **:** irá aparecer no lado esquerdo da parte de baixo da tela com o cursor posicionado após ele. O Vi irá considerar tudo o que você digitar até pressionar **Enter** como um comando. Se você apagar tudo, incluindo o **:**, você irá retornar ao modo de comando e o cursor irá retornar à sua posição original no texto.

Você pode utilizar a completação de comandos no modo de execução, digite as primeiras letras do comando e pressione a tecla **Tab** para completá-lo.

Para salvar as alterações de um arquivo digite **:w** no modo de comando. Se você quiser salvar o conteúdo do buffer para outro arquivo, digite **:w <nome_do_arquivo>**.

8.2.2. Trabalhando com Buffers

Para alternar, no mesmo buffer, entre os arquivos que foram indicados na linha de comando, digite **:next** para ir ao próximo arquivo e **:prev** para o arquivo anterior. Você também pode utilizar **:e <nome_do_arquivo>**, que deixa você escolher entre alternar para o arquivo desejado se ele já estiver aberto, ou abrir outro arquivo. Você também pode utilizar completação.

Assim como no Emacs, você pode ter vários buffers exibidos na tela. Para isto, utilize o comando **:split**.

Para alternar entre os buffers, digite **Ctrl-w j** para ir ao buffer abaixo ou **Ctrl-w k** para ir ao buffer acima. Você também pode utilizar a tecla para cima e para baixo ao invés das teclas **j** ou **k**. O comando **:close** esconde um buffer e o comando **:q** fecha.

Você deve estar ciente que se você tentar esconder ou fechar um buffer sem salvar as modificações, o comando não irá ser executado e o Vi irá exibir esta mensagem:

```
No write since last change (use ! to override)
```

Neste caso, faça como foi indicado peli Vi e digite **:q!** ou **:close!**.

8.2.3. Edição de Textos e Comandos de Movimentação

Além das teclas **Backspace** e **Del** no modo de edição, o Vi possui muitos outros comandos para apagar, copiar, colar e substituir textos em modo de comando. Todos os comandos que serão mostrados a seguir estão separados em duas partes: a ação que será executada e seu efeito. A ação pode ser:

- **c**: substituir (*Change*). O editor apaga o texto requisitado e volta para o modo de inserção após o comando.
- **d**: apagar (*Delete*);
- **y**: copiar ("Yank"). Nós vamos ver este na próxima seção.
- **.**: repete a última ação.

O efeito define que grupo de caracteres o comando irá agir sobre.

- **h**, **j**, **k**, **l**: um caractere à esquerda, abaixo, acima, à direita[3]
- **e**, **b**, **w**: para o final, início da palavra atual e início da próxima palavra.
- **^**, **0**, **$**: para o primeiro caractere não branco da linha atual, o início e o fim da linha atual.
- **f<x>**: vai para apróxima ocorrência do caractere **<x>**. Por exemplo, **fe** move o cursor para a próxima ocorrência do caratere `e`.
- **/<string>**, **?<string>**: para a próxima e última ocorrência da string ou expressão regular `<string>`. Por exemplo, **/foobar** move o cursor para a próxima ocorrência da palavra `foobar`.
- {, }: para o início e para o final do parágrafo atual;
- **G**, **H**: para o final do arquivo, para o começo da tela.

Cada um destes caracteres de "efeito" ou comandos de movimentação pode precedido por um número. Para o **G**, ("Go") isto referencia o número da linha no arquivo. Baseado nesta informação, você pode fazer todos os tipos de combinação.

Aqui estão alguns exemplos:

- **6b**: move 6 palavras para trás;
- **c8fk**: apaga todo o texto até a oitava ocorrência do caractere `k` e entao entra no modo de inserção;
- **91G**: vai para a linha 91 do arquivo;

3. Um atalho para o **dl** (apague um caractere a frente) é o **x**; um atalho para o **dh** (apague um caractere para trás) é o **X**; **dd** apaga a linha atual.

- **d3$**: apaga até o final da linha atual mais as próximas duas linhas.

Enquanto muitos destes comandos não são muito intuitivos, a melhor forma de se lembrar dos comandos é utilizando-os. Mas você pode ver que a expressão "mover montanhas com poucas teclas" não é um exagero.

8.2.4. Cortar, Copiar, Colar

O Vi contém um comando que nós já vimos para copiar texto: o comando **y**. Para cortar texto, simplesmente use o comando **d**. Há 27 memórias ou buffers para armazenar texto: uma memória anônima e 26 memórias nomeadas com as 26 letras, em caixa baixa, do alfabeto do inglês.

Para utilizar a memória anônima você entra com o comando da maneira "como ele é". Assim o comando **y12w** irá copiar as 12 palavras após o cursor na memória anônima[4]. Utilize **d12w** se você quer recortar esta área.

Para utilizar uma das 26 memórias nomeadas, entre com a sequência **"<x>** antes do comando, onde **<x>** indica o nome da memória. Entã, para copiar as mesmas 12 palavras na memória `k`, você escreveria **"ky12w**, ou **"kd12w** para recortá-las.

Para colar o conteúdo da memória anônima, utilize o comando **p** ou **P**, para inserir texto depois ou antes do cursor. Para colar o conteúdo de uma memória nomeada, utilize **"<x>p** ou **"<x>P** da mesma maneira (por exemplo **"dp** irá colar o conteúdo da memória `d` depois do cursor).

Vamos olhar um exemplo:

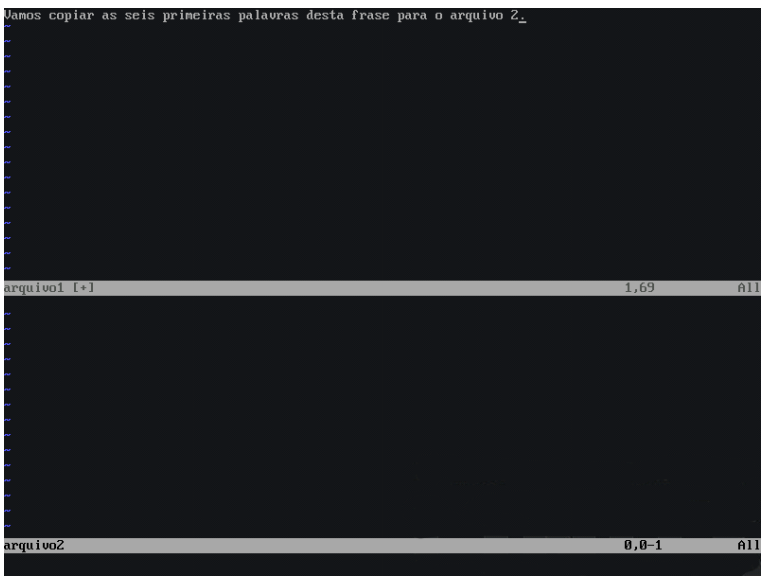


Figura 8-5. VIM, antes de copiar o bloco de texto

Para continuar esta tarefa, nó vamos:

- Copiar novamente as seis primeiras palavras da sentença na memória `r` (por exemplo): **"ry6w**[5];
- alterne para o buffer `file2`, que está localizado abaixo: **Ctrl**-**w** **j**;
- cole o conteúdo da memória `r` antes do cursor: **"rp**.

Nós obtivemos o resultado esperado, como mostrado na Figura 8-6.

4. Mas somente se o cursor estiver posicionado no início da primeira palavra!
5. **y6w** literalmente significa: "Yank 6 words (extrair 6 palavras)".

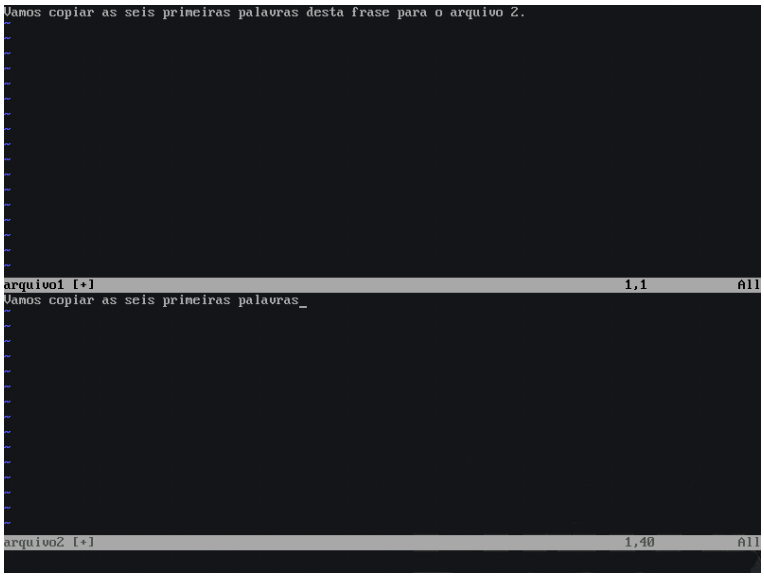


Figura 8-6. VIM, depois de ter copiado o bloco de texto

Procurar por texto é muito simples: no modo de comando, você simplesmente digita **/** seguido da string de busca, e então pressiona a tecla **Enter**. Por exemplo, **`/festa`** irá procurar pela string `festa` a partir da posição atual do cursor. Pressionando **n** a busca será repetida e o cursor levado para a próxima ocorrência, e se você chegar ao fim do arquivo, a busca irá começar novamente do início. Para procurar em sentido inverso, utilize **?** ao invés de **/**.

8.2.5. Saindo do Vi

O comando para sair é o **`:q`** (na verdade, este comando fecha o buffer atual, como nós já vimos, mas se ele é o único buffer aberto, você irá sair do Vi). Há um atalho: como na maior parte das vezes você edita apenas um arquivo, para sair, utilize:

- **`:wq`** ou **`:x`** para salvar as alterações e sair (uma solução mais rápida é **Z Z**), ou
- **`:q!`** para sair sem salvar.

Note que se você possui vários buffers abertos, o **`:wq`** irá salvar somente o buffer ativo e então fechá-lo.

8.3. Uma última palavra...

É claro, nós falamos mais do que era necessário aqui (o primeiro objetivo era editar um arquivo de texto), mas esperamos ter conseguido mostrar para você algumas das possibilidades destes editores. Há muito mais para ser falado sobre eles, como se pode ver pelo número de livros dedicados ao Vi e ao Emacs.

Reserve um tempo para absorver todas estas informações, escola um deles, ou aprenda apenas o que você acha necessário. Mas ao menos você sabe que, quando quiser, poderá ir mais adiante.

Capítulo 9. Utilitários na Linha de Comando

O propósito deste capítulo é apresentar algumas ferramentas de linah de comando úteis para o uso diário.

Uma dos pontos fortes principais no GNU/Linux é o uso de ferramentas simples para realizar tarefas complexas. Nós mostramos a você como juntar comandos e como limpar a saída para torná-la mais visível (veja Seção 7.4). Agora é hora de aprender sobre algumas ferramentas úteis que lhe darão muito controle e produtividade.

Este capítulo é uma espécie de exercício para você entender completamente o uso de cada função de comando. Então cada comando será ilustrado por um exemplo. Não tenha medo de parar e consultar a página do manual para qualquer um destes comandos. No final de cada seção você encontrará as seções "VEJA TAMBÉM" com referências cruzadas para outros comandos interessantes. Você tem um novo lugar para explorar o seu sistema GNU/Linux!

9.1. Operações com Arquivo e Filtragem

A maior parte do trabalho em linha de comando é feito em arquivos. Nesta seção nós iremos mostrar a você como observar e filtrar o conteúdo de um arquivo, como extrair a informação necessária de arquvo utilizado um único comando, e como ordenar facilmente o conteúdo de um arquivo.

9.1.1. cat, tail, head, tee: Comandos para Impressão de Arquivos

Estes comandos possuem quase a mesma sintaxe: `nome_do_comando [opção(ões)] [arquivo(s)]`, e podem ser utilizados em um pipe. Todos eles são utilizados para imprimir parte de um arquivo de acordo com um certo critério.

O `cat` concatena arquivos e imprime o resultado para a saída padrão, a qual é normalmente a tela do seu computador. Este é um dos comandos mais utilizados. Por exemplo, você pode usar:

```
# cat /var/log/mail/info
```

para imprimir o conteúdo do arquivo de log de um daemon de correio na saída padrão[1]. O comando `cat` possui uma opção muito útil(`-n`) que permite imprimir também o número de cada linha do arquivo.

Alguns arquivos, como os logs de daemons (se estiverem em execução) normalmente são bem grandes em tamanho[2] e imprimí-los na tela completamente não é uma idéia muito útil. De maneira geral, você somente precisa ver algumas linhas do arquivo. Você pode usar o comando `tail` para fazer isso. O seguinte comando irá mostrar, por padrão, as últimas 10 linhas do arquivo `/var/log/mail/info`:

```
# tail /var/log/mail/info
```

Arquivos como os de log normalmente variam dinamicamente porque o daemon adiciona constantemente ações e eventos ao arquivo. Para observar interativamente estas mudaas, você pode obter vantagem da opção `-f`:

```
# tail -f /var/log/mail/info
```

Neste caso todas as mudanças no arquivo `/var/log/mail/info` serão imediantamente apresentadas na tela. Utilizar o comando `tail` com a opção `-f` é muito útil quando você quer saber como o seu sistema funciona. Por exemplo, observando o arquivo `/var/log/messages`, você pode acompanhar mensagens do sistema e vários daemons.

Se você utilizar o `tail` com mais de um arquivo ele irá exibir o nome do arquivo em uma linha antes de apresentar o seu conteúdo. Isto também funciona com a opção `-f` e é uma valiosa funcionalidade para ver como diferentes partes do sistema interagem.

Você pode utilizar a opção `-n` para exibir as últimas n linhas de um arquivo. Por exemplo, para exibir as últimas duas linhas, voc<7869> deveria executar:

1. Alguns exemplos nesta seção são baseadas em trabalho de verdade e arquivos de log de servidores (serviços, daemons, etc.). Certifique-se de que o syslogd (que permite o registro das atividades), e o daemon correspondente (no nosso caso, o Postfix) esteja sendo executado, e que você esteja como `root`.É claro, você pode sempre aplicar os nossos exemplos em outros arquivos.
2. Por exemplo, o arquivo `/var/log/mail/info` contém informação sobre todos os e-mails enviados, mensagens sobre e-mails recebidos por usuários com o protocolo POP, etc.

```
# tail -n2 /var/log/mail/info
```

Assim como em outroscomandos, você poderia utilizar opções diferentes ao mesmo tempo. Por exemplo, utilizando junto as opções `-n2` e `-f`, você começa com as duas últimas linhas do arquivo e continua a ser informado assim que novas linhas foem exibidas ao arquivo de log.

O comando `head` é similar ao `tail`, mas ele exibe as primeiras linhas de um arquivo. O comando a seguir irá exibir, por padrão, as dez primeiras linhas do arquivo `/var/log/mail/info`:

```
# head /var/log/mail/info
```

Assim como o `tail` você pode utilizar a opção `-n` para especificar o número de linhas a ser exibido. Por exemplo, para imprimir as duas primeiras, digite:

```
# head -n2 /var/log/mail/info
```

Você também pode utilizar estes comandos juntos. Por exemplo, se você quiser exibir somente as linhas 9 e 10 de um arquivo, você pode usar um comando onde o `head` irá selecionar as 10 primeiras linhas e passá-las através de um pipe para o comando `tail`.

```
# head /var/log/mail/info | tail -n2
```

A últma parte irá então selecionar as duas últimas linhas e irá exibí-las na tela. Da mesma maneira você pode selecionar a vigésima linha a partir do fim do arquivo:

```
# tail -n20 /var/log/mail/info |head -n1
```

Neste exemplo nós dizemos ao `tail` para selecionar as últimas vinte linhas do arquivo e passá-las através do pipe para o comando `head`. Então o `head` exibe na tela a primeira linha das informações obtidas.

Vamos supor que queremos exibir o resultado do último exemplo na tela e salvá-lo no arquivo `resultados.txt`. Para isso, vamos utilizar o comando `tee`, que possui a seguinte sintaxe:

```
tee [opções] [arquivo]
```

Agoranós podemos alterar o comando anterior desta maneira:

```
# tail -n20 /var/log/mail/info |head -n1|tee resultados.txt
```

Vamos pegar um outro exemplo. Nós queremos selecionar as últimas 20 linhas, salvá-las no arquivo `resultados.txt`, mas imprimir na tela somente a primeira linha das 20 selecionadas. Então deveríamos executar:

```
# tail -n20 /var/log/mail/info |tee resultados.txt |head -n1
```

O comando `tee` possui uma opção poderosa (`-a`) que lhe possibilita adicionar dados a um arquivo existente.

Na próxima seção nós iremos ver como utilizar o comando grep como um filtro para separar as mensagens do Postfix de mensagens geradas por outros serviços.

9.1.2. grep: Localizando Strings em um Arquivo

Nem o nome e nem o acrônimo ("General Regular Expression Parser" - "Analisador de Expressões Regulares") são muito intuitivos, mas o que ele faz e o seu uso são muito simples: o grep procura em um ou mais arquivos por um padrão padrão passado como um argumento. A sua sintaxe é

```
grep [opções] <padrão> [um ou mais arquivo(s)]
```

Se for indicado mais de um arquivo, o nome de cada arquvo é mecionado no começo de cada linha do resultado. Você pode usar a opção `-h` para evitar que o nome seja exibido ou então a opção `-l` para retornar apenas o nome dos arquivos. O padrão é uma expressão regular, embora na maior parte das vezes ela seja uma única palavra. As opções mais comuns são:

- `-i`: ignora a diferença entre letras maiúsculas e minúsculas durante a busca;
- `-v`: inverte a pesquisa. Exibe linhas que **não** combinam com o padrão;

- `-n`: exibe o número da linha para cada linha encontrada;
- `-w`: indica para o grep que o padrão deve ser uma palavra inteira.

Então vamos voltar a analizaro arquivo de log do servidor de e-mails. Nós queremos encontrar todas as linhas no arquivo `/var/log/mail/info` que contém o padrão `postfix`. Então nós digitamos este comando:

```
# grep postfix /var/log/mail/info
```

Se quisermos encontrar todas as linhas que NÃO possuem o padrão `postfix`, nós utilizamos então a opção `-v`:

```
# grep -v postfix /var/log/mail/info
```

O comando grep pode ser utilizado com o pipe.

Vamos supor que queremos encontrar todas as mensagens sobre os e-mails enviados com sucesso. neste caso temos que filtrar todas as linhas que foram adicionadas ao arquivo de log pelo servidor de e-mails (ou seja, que contenham o padrão `postfix`) e elas devem conter ma mensagem sobre o sucesso do envio (`status=sent`)[3]:

```
# grep postfix /var/log/mail/info |grep status=sent
```

Neste caso o comando grep é utilizado duas vezes. É permitido, mas não muito elegante. O mesmo resultado pode ser alcançado com o utilitário `fgrep`. O `fgrep` é na verdade um método simples de chamar o `grep -F`. Primeiro nós precisamos criar um arquivo contendo cada padrão em um linha. Este arquivo pode ser criado da seguinte maneira (nós usaremos `padrao.txt` como o nome do arquivo:

```
# echo -e 'status=sent\npostfix' >./padroes.txt
```

Verifique o resultado com o comando `cat`. O `\n` é um padrão especial que significa "nova linha".

assim vamos executar o próximo comando com o `padroes.txt` e o aplicativo `fgrep` em vez de chamar duas vezes o grep:

```
# fgrep -f ./padrao.txt /var/log/mail/info
```

O arquivo `./padrao.txt` pode conter quantos padrões você quiser. Por exemplo, para selecionar as mensagens sobre e-mails enviados com sucesso para `usuario2@mandriva.com`, seria o suficiente adicionar este padrão no nosso arquivo `./padroes.txt`, executando este comando:

```
# echo 'usuario2@mandriva.com' >>./padroes.txt
```

É claro que você pode combinar o grep com o `tail` e `head`. Se nós queremos encontrar mesagens sobre o penúltimo e-mail enviado para `usuario2@mandriva.com`, nós digitaríamos:

```
# fgrep -f ./padroes.txt /var/log/mail/info | tail -n2 | head -n1
```

aqui nós aplicamos o filrto descrito acima e direcionamos o resultado em um pipe para os comandos `tail` e `head`. Eles selecionam a penúltima linha da lista.

9.1.3. Expressões Regularem e Filtros com o egrep

Com o grep nós estamos presos com padrões e dados estáticos. Como nós poderíamos encontrar cada e-mails enviados para qualquer funcionário da "ABC Company"? Listar todos os e-mails deles não seria uma tarefa fácil já que poderíamos acabar esquecendo um ou ter que analisar o arquivo de log manualmente.

Assim como o `fgrep`, o grep também possui um atalho para o comando `grep -E`: `egrep`. Ele recebe expressões regulares em vez de padrões, oferecendo uma interface mais poderosa para a busca de texto.

Além do que nós mencionamos na Seção 7.3 sobre os caracteres curingas, aqui estão mais algumas expressões regulares:

- `[:alnum:]`, `[:alpha:]` e `[:digit:]` podem ser utilizados para evitar que você tenha que definir manualmente classes de caractere que representem, respectivamente, todas as letras mais todos os números,

3. Embora seja possível filtrar apenas pelo padrão do do status, iremos fazer a busca desta maneira para lhe apresentar um novo comando com este exemplo.

todas as letras (em caixa baixa ou alta), e todos os números. Estas expressões também oferecem um bônus adicional: eles incluem caracteres de internacionalização e respeitam a localização do sistema.

- `[:print:]` representa todos os caracteres que podem ser exibidos na tela.
- `[:lower:]` e `[:upper:]` representam lodas as letras de caixa baixa e alta.

Há mais classes disponíveis e você pode ver todas elas na página do manual do egrep(1). As citadas acima são as mais utilizadas.

Uma expressão regular pode ser acomanhada de diversos operadores de repetição:

?

O item que preceder este operador é opcional, isto é, pode haver zero ou uma ocorrência do caractere, mas não mais do que uma.

*

O item que estiver precedendo este operador pode aparecer zero ou mais vezes.

+

Indica que o item pode aparecer uma ou mais vezes.

{n}

Informa que o item deve aparecer exatamente `n` vezes.

{n,}

Combina `n` ou mais vezes o item preecdente a este operador.

{n,m}

Busca por no mínimo `n` ocorrências do item precedente, mas não mais do que `m` vezes.

Se você colocar uma expressão regular dentro de parênteses, você pode reaproveitá-la em outro local. Vamos dizer que você especificou a expressão `[:alpha:]+`, que poderia representar uma palavra. Então, se você quiser detectar palavras que aparecem duas vezes você poderia colocar esta expressão dentro ed parênteses e fazer uma referência a ela com `\1` que indica o primeiro grupo do padrão. Você pode ter até 9 destas "memórias".

```
$ echo -e "abc def\nabc abc def\nabc1 abc1\nabcdef\nabcdabcd\nabcdef abcef" > arquivoteste
$ egrep "([[:alpha:]]+) \1" arquivoteste
abc abc def
$
```

Os caracteres `[` e `]` fazem parte do nome do grupo, então nós precisamos incluí-lo para usar esta classe de caracteres. O primeiro `[` informa que nós estamos utilizado um grupo de caracteres, o segundo faz parte do nome do grupo, e então há os caracteres `]` que fecham cada um dos grupos abertos.

A linha apresentada no resultado é a única que atende o padrão de dois grupos de letras separados por um espaço. Nenhuma outra linha do arquivo combinou com a expressão regular.

Você também pode utilizar o caractere `|` para informar que o padrão deve casar com a expresão à esquerda ou à direita do `|`. Utilizando o mesmo arquivo que criamos no exemplo anterior (`arquivoteste`), você poed tentar procurar por expressões que contenham palavras duplicadas ou palavras duplicadas que contenham números:

```
$ egrep "([[:alpha:]]+) \1|([[:alpha:][:digit:]]+) \2" arquivoteste
abc abc def
abc1 abc1
$
```

Note que para o segundo grupo entre parênteses nós utilizamos `\2`, caso contrário ele não combinaria com o padrão que esperávamos. Uma expressão mais eficiente, para este caso em particular, seria:

```
$ egrep "([[:alnum:]]+) \1" testfile
abc abc def
abc1 abc1
$
```

Finalmente, para combinar certos caracteres será necessário "escapá-los", colocando uma contrabarra antes deles. Estes caracteres são: `?`, `+`, `{`, `|`, `(`, `)` e `\`. Quando quiser buscar por estes caracteres, você terá que escrever: `\?`, `\+`, `\{`, `\|`, `\(`, `\)`, e `\\`.

Este simples truque pode evitar que você tenha palavras repetidas em seu texto.

Expressões regulares em todas as ferramentas devem seguir estas regras, ou ao menos regras similares a estas. Dedicar um pouco de tempo para entendê-las irá ajudar muito quando você trabalhar com ferramentas como o `sed`. O `sed` permite que você anipule texto, alterando-o com o uso de expressões regulares e outras coisas.

9.1.4. wc: Contando Elementos em Arquivos

O comando `wc` (*Word Count - Contagem de Palavras*) é utilizado para contar o número de linhas, palavras e caracteres nos arquivos. Ele também pode informar o tamanho da maior linha. Sua sintaxe é:

```
wc [opções] [arquvo(s)]
```

As seguintes opções são bastante utilizadas:

- `-l`: imprime o número das linhas;
- `-w`: imprime o número de palavras;
- `-m`: imprime o número total de caracteres;
- `-c`: imprime o número de bytes;
- `-L`: imprime o tamanho da maior linha no texto.

O comando `wc`, por padrão, imprime o número de linhas, palavras e caracteres. Aqui estão alguns exemplos de uso:

Se quisermos encontrar o número de usuários em nossos sistema, por exemplo, nós poderíamos digitar:

```
$ wc -l /etc/passwd
```

Se quisermos saber o número de processadores em nosso sistema, nós podemos executar:

```
$ grep "model name" /proc/cpuinfo |wc -l
```

Na seção anterior nós obtivemos uma lista de mensagens sobre e-mails enviados com sucesso para os endereços listados no arquivo `./padroes.txt`. Se quisermos saber quantas mensagens ele contém, podemos redirecionar o resultado do filtro para o comando `wc` através de um pipe:

```
# fgrep -f ./padroes.txt /var/log/mail/info | wc -l
```

9.1.5. sort: Ordenando o Conteúdo de um Arquivo

Aqui está a sintaxe deste poderoso utilitário para ordenação[4]:

```
sort [opções] [arquivo(s)]
```

Vamos considerar a ordenação em parte do arquivo `/etc/passwd`. Como você pode ver, o arquivo não é ordenado:

```
$ cat /etc/passwd
```

Se nós quisermos ordená-lo pelo campo de `login`, nós podemos digitar:

```
$ sort /etc/passwd
```

4. Nós vamos discutir o brevemente o `sort` aqui. Livros inteiros podem ser escritos sobre as suas funcionalidades.

O comando `sort` ordena as informações em ordem crescente iniciando pelo primeiro campo por padrão (no nosso caso, o campo `login`). Para ordenar em ordem decrescente, use a opção `-r`:

```
$ sort -r /etc/passwd
```

Todo usuário possui um `UID` informado no arquivo `/etc/passwd`. O seguinte comando ordena um arquivo em ordem crescente utilizando o campo `UID`:

```
$ sort /etc/passwd -t":" -k3 -n
```

aqui nós utilizamos as seguintes opções do `sort`:

- `-t":"`: tells `sort` that the field separator is the `":"` symbol;
- `-k3`: means that sorting must be done on the third column;
- `-n`: says that the sort is to occur on numerical data, not alphabetical.

Também pode ser ordenado de maneira decrescente:

```
$ sort /etc/passwd -t":" -k3 -n -r
```

O `sort` também possui outras duas opções importantes:

- `-u`: elimina entradas duplicadas durante a ordenação;
- `-f`: trata de maneira igual os caracteres em caixa baixa ou alta.

Finalmente, se quisermos encontrar o usuário com o maior `UID`, nós podemos utilizar este comando:

```
$ sort /etc/passwd -t":" -k3 -n |tail -n1
```

onde ordenamos o arquivo `/etc/passwd` em ordem crescente de acordo com a coluna `UID`, e redirecionamos o resultado através de um pipe para o comando `tail`, que imprime o último ítem da lista.

9.2. find: Encontrando Arquivos

O `find` é um utilitário de longa data no UNIX®. Seu papel é o de procurar recursivamente u ou mais dirtórios e arquivos que atendam a um certo conjunto de critérios. Embora ele seja muito útil, possui uma sintaxe um tanto obscura, e a sua utilização requer um pouco de prática. A sintaxe geral é:

```
find [opções] [diretórios] [critério1] ... [critérioN] [ação]
```

Se você não especificar qualquer diretório, o `find` irá buscar no diretório atual. Se você não especificar um critério, isto é o equivalente a dizer que todos os arquivos devem ser encontrados. As opções, critérios e ações são tão numerosas que nós vamos mencionar apenas algumas. Estas são algumas opções:

- `-xdev`: não procura em diretórios localizados em outros sistemas de arquivo.
- `-mindepth <n>`: desce pelo menos `n` níveis abaixo do diretório especificado antes de procurar pelos arquivos.
- `-maxdepth <n>`: procura por arquivos que estão localizados no máximo a `n` níveis abaixo do diretório especificado.
- `-follow`: seguir link simbólico caso ele aponte para diretórios. Por padrão, o `find` não segue links.
- `-daystart`: quando utilizar testes baseados em tempo (veja abaixo), esta opção considera a data como o início do dia atual, ao invés do padrão (24 horas antes do horário atual).

Um critério pode ser um ou mais testes ***atômicos***. Alguns testes úteis são:

- `-type <tipo_de_arquivo>`: busca por um determinado tipo de arquivo. `tipo_de_arquivo` pode ser um dos seguintes: `f` (arquivo comum), `d` (diretório), `l` (link simbólico), `s` (socket), `b` (arquivo de bloco), `c` (arquivo de caractere) ou `p` (pipe nomeado).

- `-name <padrão>`: busca arquivos cujos nomes combinam com o padrão fornecido. Com esta opção, o padrão é tratado como os ***caracteres curinga*** (veja Seção 7.3).
- `-iname <pattern>`: igual ao `-name`, mas ignora a diferença entre letras maiúsculas e minúsculas.
- `-atime <n>`, `-amin <n>`: procura arquivos que foram acessados pela última vez há `n` dias atrás (`-atime`) ou `n` minutos atrás (`-amin`). Você também pode especificar `<+n>` ou `<-n>`, o que significa que a busca será feita para arquivos acessados no máximo ou no mínimo há `n` dias/miniutos atrás.
- `-anewer <um_arquivo>`: encontra arquivos que foram acessados mais recentemente do que `um_arquivo`.
- `-ctime <n>`, `-cmin <n>`, `-cnewer <arquivo>`: o mesmo que `-atime`, `-amin` e `-anewer`, mas aplica ao último horário em que o conteúdo do arquivo foi modificado.
- `-regex <padrão>`: o mesmo que `-name`, mas o padrão `padrão` é tratado como uma ***expressão regular***.
- `-iregex <padrão>`: o mesmo que `-regex`, mas ignorando letras a diferença entre letras maiúsculas e minúsculas.

Há muitos outros testes, veja find(1) para mais detalhes. Para combinar os testes, você pode usar:

- `<c1> -a <c2>`: verdadeiro se `c1` e `c2` são verdadeiros; `-a` é implícito, então você digitar `<c1> <c2> <c3>` se você quer testar `c1`, `c2` e `c3`.
- `<c1> -o <c2>`: verdadeiro se `c1` ou `c2` são verdadeiros, ou ambos. Note que `-o` possui uma ***precedência*** menor do que `-a`, então se você quiser encontrar arquivos que case com o padrão `c1` ou `c2` e também com o critério `c3`, você terá que utilizar parênteses: `( <c1> -o <c2> ) -a <c3>`. Você deve ***escapar*** (desativar) os parênteses, pois senão eles serão interpretados pelo shell!
- `-not <c1>`: inverte o teste `c1`, desta forma, `-not <c1>` é verdadeiro quando `c1` é falso.

Finalmente, você pode especificar uma ação para cada arquivo encontrado. O mais utilizados são:

- `-print`: apenas imprima o nome de cada arquivo na saída padrão. Esta é a ação padrão.
- `-ls`: exiba na saída padrão o equivalente ao comando `ls -ilds` aplicado em cada um dos arquivos encontrados.
- `-exec <linha_de_comando>`: executa o comando `linha_de_comando` em cada arquivo encontrado. A `linha_de_comando` deve terminar com um `;`, que você deve escapar para que o shell não interprete-a. A posição do arquivo é marcada com `{}`. Veja os exemplos de utilização.
- `-ok <comando>`: o mesmo que `-exec` mas pede por uma confirmação para cada comando.

A melhor forma de consolidar todas as opções e parâmetros é com alguns exemplos. Para encontrar todos os diretórios em `/usr/share`, por exemplo, digitaríamos:

```
find /usr/share -type d
```

Vamos supor que você tenha um servidor HTTP. Todos os seus arquivos HTML estão em `/var/www/html`, que é também o diretório onde você está posicionado no momento. Você quer encontrar todos os arquivos que no tiveram os seus conteúdos modificados por um mês. E como você tem páginas de vários autores, alguns arquivos possuem a extensão `html` e outros a extensão `htm`. E voce quer criar um link para estes arquivos no diretório `/var/www/obsolete`. Você poderia digitar[5]:

```
find \( -name "*.htm" -o -name "*.html" \) -a -ctime -30 \
-exec ln {} /var/www/obsolete \;
```

Este é um exemplo um tanto complexo, e requer uma pequena explicação. O critério é o seguinte:

```
\( -name "*.htm" -o -name "*.html" \) -a -ctime -30
```

que faz o que queremos: ele encontra todos os arquivos cujos nomes terminam em `.htm` ou `.html` (“`\( -name "*.htm" -o -name "*.html" \)`”), **e** (`-a`) que não foram modificados nos últimos 30 dias, o que é aproximadamente um mês(`-ctime -30`). Note os parênteses: eles são necessários aqui porque o `-a` tem uma precedência mais alta. Se não tiver nenhum parêntese, seriam encontrados todos os arquivos que terminam com `.htm`, mais todos os arquivos `.html` que não foram modificados no último mês, e isso não é o que esperávamos. Note também que os parênteses fora escapados: se nós tivéssemos utilizado `( .. )` ao invés de

5. Note que este exemplo requer que `/var/www` e `/var/www/obsolete` estejam no mesmo sistema de arquivo!

`\( .. \)`, o shell teria interpretado-os e tentado executar `-name "*.htm" -o -name "*.html"` em um sub-shell... Outra solução seria colocar os parênteses entre aspas, mas uma contrabarra aqui é preferível, já que temos que isolar apenas um caractere.

E finalmente, temos o comando que deverá ser executado para cada arquivo:

```
-exec ln {} /var/www/obsolete \;
```

Aqui você também tem que escapar o caractere `;`. Caso contrário o shell iria interpretá-lo como um separador de comando. Se acontecer de você esquecer, o `find` irá reclamar que está faltando um argumento para o `-exec`.

Um último exemplo: você tem um diretório enorme (`/shared/images`) que contém todo tipo de imagens. Você normalmente usa o comando `touch` para atualizar o horário de um arquivo chamado `stamp` neste diretório, e assim você tem uma referência de tempo. Você quer encontrar todos os arquivos de imagem ***JPEG*** que são mais novos do que o arquivo `stamp`, mas como você obtém arquivos de várias fontes, estes arquivos possuem as extensões `jpg`, `jpeg`, `JPG` ou `JPEG`. Você também quer evitar a busca no diretório `old` e que a lista de arquivos seja enviada por e-mail para vocês, e seu nome de usuário é **`usuario2`**:

```
find /shared/images -cnewer     \
     /shared/images/stamp       \
     -a -iregex ".*\.jpe?g"     \
     -a -not -regex ".*/old/.*" \
       | mail usuario2 -s "New images"
```

É claro que este comando não é muito útil se você tem que digitá-lo sempre, pois você gostaria que ele fosse executado regularmente. Uma maneira simples de ter o comando sendo executado periodicamente é através do uso do cron, como terá sua explicação mostrado a seguir.

9.3. Agendamento de Inicialização de Comandos

9.3.1. crontab: Reporting ou Editando o Arquivo crontab

O `crontab` permite que você execute comandos em um intervalo regular de tempo com o bônus adicioal de que você não precisa estar logado para que isto aconteça. O `crontab` irá enviar para você, por e-mail, a saída do comando. Você pode especificar os intervalos em minutos, horas, dias e até meses. Dependendo das opções, o `crontab` irá agir de maneira diferente:

- `-l`: exibe o conteúdo do arquivo `crontab`;
- `-e`: edita o arquivo `crontab`;
- `-r`: remove o arquivo `crontab`;
- `-u <usuário>`: aplica uma das opções acima para o usuário indicado. Somente o `root` pode fazer isto.

Vamos começar editando um arquivo `crontab`. Se você digitar `crontab -e`, o editor de sua preferência será aberto, caso você tenha configurado as variáveis de ambiente `EDITOR` ou `VISUAL`, senão o Vi será o editor utilizado. Uma linha no arquivo `crontab` é feita de seis campos. Os primeiros cinco campos são os intervalos de tempo para minutos, horas, dias do mês, mês e dias da semana. O sexto campo é o comando a ser executado. Linhas que começam com um `#` são consideradas comentários e serão ignoradas pelo `crond` (o programa que é responsável por executar os arquivos `crontab`). Este formato é um pouco diferente para o sistema `crontab`, em `/etc/crontab`. Lá, o sexto campo é o nome do usuário que deveria ser utilizado para iniciar o programa no sétimo campo. Ele deveria ser utilizado somente para tarefas administrativas, e para processos em execução de usuários que existem somente para aumentar a segurança do sistema (como o suário de um anti-vírus ou um usuário criado para executar um servidor de banco de dados). Aqui está um exemplo do `crontab`:

Para imprimir o arquivo de uma maneira legível, nós tivemos que quebrar as linhas que eram grandes demais. Então, alguns blocos devem ser digitados em uma única linha. Quando o caractere `\` finaliza uma linha, significa que esta linha possui continuação. Esta convenção funciona com os arquivos `Makefile` e no shell, assim como em outros contextos.

```
# Se você não deseja receber e-mail,
# comente a seguinte linha
#MAILTO="seu_endereço_de_e-mail"
#
# Avise a cada 2 dias sobre novas imagens,
#   como no exemplo acima - e depois disto, "crie novamente"
#   o arquivo de "data". O "%" é tratado como uma nova
#   linha, o que lhe possibilita colocar vários comandos
#   em uma mesma linha.
0 14 */2 * *  find /shared/images              \
  -cnewer /shared/images/stamp                 \
  -a -iregex ".*\.jpe?g"             \
  -a -not -regex                               \
    ".*/old/.*"%touch /shared/images/stamp
#
# Todo Natal toque uma melodia :)
0 0 25 12 * mpg123 $HOME/sounds/merryxmas.mp3
#
# Toda terça-feira às 17h, exiba a lista de compras...
0 17 * * 2 lpr $HOME/shopping-list.txt
```

Há várias outras maneiras de especificar intervalos diferentes das apresentadas neste exemplo. Você pode especificar um conjunto de ***valores discretos*** separados por vírgulas (`1,14,23`) ou um intervalo (`1-15`), ou até combinar as duas formas (`1-10,12-20`), opcionalmente com um valor de step, que pulará a seqüência com o valor indicado (`1-12,20-27/2`).

9.3.2. at: Agendando uma Comando uma Única Vez

Pode ser que você queira agendar um comando para um determinado dia, mas sem uma certa freqüência, como ser lembrado de um compromisso às 18:00 horas de oje. Você executa o X, o pacote `X11R6-contrib` é instalado e então você é avisado às 17:30 que você precisa ir. Para isso, o `at` é exatamente o que você preisa:

```
$ at 5:30pm
 # You're now in front of the "at" prompt
at> xmessage "Hora de ir! Compromisso às 18:00 horas"
 # Type CTRL-d to exit
at> <EOT>
job 1 at 2005-02-23 17:30
$
```

Você pode especificar o tempo de outras formas:

- `now + <intervalo>`: significa o intervalo indicado a partir do momento em que o comando foi executado. A sintaxe para o intervalo é `<n> (minutes|hours|days|weeks|months)`. Por exemplo, você pode especificar `now + 1 hour` (uma hora a partir de agora), `now + 3 days` (três dias a partir de agora) e assim por diante.
- `<horário> <dia>`: especifica a data de maneira completa. O parâmetro `<horário>` é obrigatório. O `at` aceita o parâmtero de diversas maneiras: você pode digitar `0100`, `04:20`, `2am`, `0530pm`, `1800`, ou um dos três valores especiais: `noon` (meio-dia), `teatime` (hora do chá - 16:00 hrs) ou `midnight` (meia-noite). O parâmetro `<dia>` é opcional. Você pode indicá-lo de várias formas: a norte-americana, por exemplo, onde `12/20/2001` apontaria para o dia 20 de dezembro de 2001, ou a forma européia, mais parecida com a nossa, `20.12.2001`. Você pode omitir o ano, mas somente na forma européia: `20.12`. Você também pode especificar o mês com a abreviação em inglês dos meses: tanto `Dec 20` como `20 Dec` são validos.

O `at` também aceita algumas opções diferentes:

- `-l`: exibe a lista de tarefas agendadas; o primeiro campo é o número da tarefa. É equivalente ao comando `atq`.
- `-d <n>`: remove a tarefa `<n>` da fila. Você pode obter o número da tarefa através do comando `atq`. É equivalente ao comando `atrm <n>`.

Como sempre, veja a página do manual do comando at(1) para mais informações.

9.4. Arquivamento e Compactação de Dados

9.4.1. tar: Tape ARchiver

Assim como o `find`, o `tar` é um utilitário UNIX® de longa data, então a sua sintaxe é um pouco especial:

```
tar [opções] [arquivos...]
```

Vamos apresentar abaixo uma lista de algumas opções disponíveis para o `tar`. Note que todas elas possuem uma opção equivalente com um formato lngo, mas você terá que pesquisá-las em tar(1), já que não trataremos delas neste guia.

O hífen (`-`) que precede as opções no formato curto não é mais utilizado pelo `tar`, sendo exigido apenas para as opções no formato longo.

- `c`: utilizado para criar um novo arquivo tar.
- `x`: utilizado para extrair os arquivos existentes em um arquivo tar.
- `t`: lista os arquivos existentes em um arquivo tar.
- `v`: aumenta a verbosidade. Lista os arquivos que estão sendo adicionados ou extraídos de um arquivo tar. Se utilizado em conjunto com a opção `t` (veja abaixo), é exibida uma listagem longa dos arquivos.
- `f <nome_do_arquivo>`: cria um arquivo com o nome `nome_do_arquivo`, extrai ou lista os arquivos contidos em `nome_do_arquivo`. Se este parâmetro for omitido, o arquivo padrão será o `/dev/rmt0`, que é, de maneira geral, um arquivo especial associado a um ***streamer***. Se o valor do parâmerto for `-` (um hífen), a entrada ou a saída (dependendo da operação que você está realizando) será associaad à entra ou saída padrão.
- `z`: informa ao `tar` que o arquio a ser criado deve ser comprimido com o `gzip`, ou que o arquivo a ser extraído está comprimido com o `gzip`.
- `j`: faz o mesmo que a opção `z`, mas utiliza o `bzip2` para comprenssão do arquivo.
- `p`: preserva o atributo dos arquivos que são extraídos, incluindo o dono do arquivo, o último acesso e outros. Muito útil para cópias de sistemas de arquivo.
- `r`: adiciona os arquivos indicados na linha de comando a um arquivo tar exisente. Note que o arquivo tar **não** deve estar comprimido para poder receber novos arquivos.
- `A`: concatena o conteúdo de arquivos tar passados na linha de comando ao arquivo tar indicado com a opção `f`. Assim como na opção `r`, o arquivo tar não deve estar comprimido.

Há muitas outras opções que não estão listadas aqui, e você pode encontrar uma lista completa em tar(1). Veja, por exemplo, a opção `d`.

Vamos seguir com um exemplo. Digamos que você queira arquivar todas as imagens do diretório `/shared/images`, em um arquivo comprimido com o `bzip2` e chamado `imagens.tar.bz2` dentro do seu diretório `/home`. Você então digitaria:

```
 #
 # Nota: você deve estar no diretório em que
 #  estão os arquivos que você quer compactar!
 #
$ cd /shared
$ tar cjf ~/imagens.tar.bz2 images/
```

Como você pode ver, nós utilizamos três opções aqui: `c` informa ao `tar` que nós queremos criar um arquivo, a opção `j` serve para comprimí-lo com o `bzip2`, e `f ~/imagens.tar.bz2` que o arquivo será criado em nosso diretório home e que deverá ser chamado `imagens.tar.bz2`. Agora nós podemos fazer um teste com o arquivo pedindo para listar o seu conteúdo:

```
 #
 # Volte ao diretório home do usuário
 #
```

```
$ cd
$ tar tjvf imagens.tar.bz2
```

Aqui nós dizemos ao `tar` para listar (`t`) os arquivos compactados em `imagens.tar.bz2` (`f imagens.tar.bz2`), informando que este arquivo foi compactado com o `bzip2` (`j`), e que quermos uma listagem longa (`v`). Agora, digamos que você apagou o diretório das imagens. Felizmente o seu arquivo ainda está intacto e agora você quer extrair o conteúdo dele para onde ele estava antes, no diretório `/shared`. Mas como você quer continuar a verificar as imagens novas com o comando `find`, você precisará preservar os atributos das imagens quando extraí-las:

```
 #
 # acesse o diretório onde você quer extrair os arquivos
 #
$ cd /shared
$ tar jxpf ~/imagens.tar.bz2
```

Pronto, imagens estão recuperadas!

Mas vamos supor que você queira extrair do arquivo apenas o diretório `images/cars` e mais nada. Assim você pode executar:

```
$ tar jxf ~/imagens.tar.bz2 images/cars
```

Se você tentar fazer uma cópia de seguraça de arquivos especiais, o `tar` irá copiá-los da maneira como eles são, arquivos especiais, e não irá copiar os seus conteúdos.Sendo assim, você pode fazer uma cópia do arquivo `/dev/mem`, por exemplo, em um arquivo tar. Ele também lida de maneira correta com links. Para links simólicos, dê uma olhada na opção `h` na página do manual do programa.

9.4.2. bzip2 e gzip: Compressão de Dados Programs

Nós já vimos estas aplicações trabalhando em conjunto com o `tar`. Mas diferentemente do WinZip® no Windows®, o arquivamento e a compresão aqui são feitos por utilitários separados: `tar` para o arquivamento, e os dois programa que vamos ver agora para a compressão de dados: `bzip2` e `gzip`. Você também pode utilizar outras ferramentas para compressão, como o `zip`, `arj` ou `rar`, pois eles também estão disponíveis no GNU/Linux, embora sejam raramente utilizados.

O `bzip2` foi escrito como um substituto ao `gzip`. Sua taxa de compressão é melhor que a do `gzip`, mas em contrapartida, ele exige mais recursos do sistema. Além disso, o `gzip` ainda é bastante utilizado por motivos de compatibilidade com sistemas antigos.

Ambos os comandos possuem uma sintaxe semelhante:

```
gzip [opções] [arquivo(s)]
```

Se nenhum nome de arquivo for indicado, tanto o `gzip` quato o `bzip2` irão receber os dados da entrada padrão e enviar o resultado para a saída padrão. Assim, você pode usar os dois comandos em redirecionamentos com pipes. Ambos os comandos também possuem algumas opções em comum:

- `-1`, ..., `-9`: indica a taxa de compressão. Quanto maior o número, maior a compressão. E por isso o processo também será mais lento.
- `-d`: descomprimir arquivo(s).
- `-c`: exibe na saída padrão o resultado da compactação/descompactação dos arquivos indicados nos parâmetros.

Por padrão o `gzip` e o `bzip2` apagam o(s) arquivo(s) que eles compactaram (ou descompactaram) se você não usar a opção `-c`. Você pode evitar isso no `bzip2` utilizando a opção `-k`. O `gzip` não possui uma opção equivalente.

Agora alguns exemplos. Vamos dizer que você quer compactar todos os arquivos que terminam com `.txt` no diretório em que você está utilizando o `bzip2` com compressão máxima. Você então poderia executar:

```
$ bzip2 -9 *.txt
```

Vamos supor que você quer compartilhar as suas imagens compacadas com alguém, mas esta pessoa não possui o `bzip2` no sistema, apenas o `gzip`. Você não precisa descompactar o arquivo e recompactá-lo, você pode descompacta-lo para a saída padrão, usar um pipe, compactar da entrada padrão e redirecionar a saída para o novo arquivo. Desta forma:

```
bzip2 -dc imagens.tar.bz2 | gzip -9 >imagens.tar.gz
```

Você poderia ter usado o `bzcat` em vez do `bzip2 -dc`. Há um equivalente para o `gzip` mas o seu nome é `zcat`, e não `gzcat`. Também existe o `bzless` para arquivos `bzip2` e o `zless` para a `gzip` se você quiser visualizar diretamente os arquivos compactados, sem a necessidade de descompactá-los antes. Como exercício, tente encontrar o comando que você deveria digitar para visualizar arquivos compactados sem a a necessidade de desompactá-los, e sem utilizar o `bzless` ou `zless`.

9.5. Muito, Muito Mais...

Existem tantos comandos que um bom livro sobre eles ficaria do tamanho de uma enciclopédia. Este capítulo não cobriu nem um décimo do assunto, apesar de você já poder fazer muito com o que aprendeu aqui. Se desejar, você pode ler algumas destas páginas do manual:: sort(1), sed(1) and zip(1L) (sim, você pode extrair ou criar arquivos `.zip` com o GNU/Linux), convert(1), e muitas outras. A melhor forma de se familiarizar com estas ferramentas é experimentando-as e praticando. Você provavelmente encontrará muita utilidade para elas, até mesmo em momentos inesperados. Divirta-se!

Capítulo 10. Controle de Processo

Nós já vimos na Seção 1.3 o que é um processo. Agora nós vamos aprender como listar e manipular os processos.

10.1. Mais Sobre os Processos

É possível monitorar os processos. Faze-los terminar, pausar, continuar, etc. Para entender os exemplos que vamos examinar, será útil conhecer um pouco mais sobre eles.

10.1.1. A Árvore de Processo

Assim como os arquivos, todos os processos que estão sendo executados em um sistema GNU/Linux estão organizados em forma de árvore. A raiz desta árvore é o `init`, um processo do sistema que é inicializado durante o boot. O sistema atribui um número (PID, *Process ID*) a cada processo para poder identificá-los. eles também herdam o PID dos seus processos pais (PPID, *Parent Process ID*). O `init` é o seu próprio pai: o PID e o PPID do `init` é 1.

10.1.2. Sinais

Todo processo no UNIX® pode reagir a sinais enviados para ele. Há 64 sinais diferentes que são identificados por seus números (iniciando em `1`) ou pelo seus nomes simbólicos (`SIGx`, onde `x` é o nome do sinal). Os 32 sinais "mais altos" (`33` a `64`) são sinais de tempo real e estão fora do escopo deste capítulo. Para cada um destes sinais, o processo pode definir o seu comportamento, exceto para dois deles: o sinal de número 9 (`KILL`) e número 19 (`STOP`).

O sinal `9` termina um processo de maneira irrevogável, sem dar a ele um tempo para terminar de maneira apropriada. Este é o sinal que você vai enviar para um processo que está travado ou que apresenta outros tipos de problemas. Uma lista completa de sinais pode ser visualizada como comando `kill -l`.

10.2. Informação sobre Processos: ps e pstree

Estes dois comandos exibem um lista de processos que estão sendo executados no sistema, de acordo com o critério que você selecionar. O `pstree` possui um saída mais limpa se comparado ao `ps -f`.

10.2.1. ps

Se executar o `ps` sem argumentos ele irá mostrar somente os processos iniciados por você e que estão no terminal que você está utilizando:

```
$ ps
   PID TTY          TIME CMD
 18614 pts/3    00:00:00 bash
 20173 pts/3    00:00:00 ps
```

Assim como muitos programas UNIX®, o `ps` possui muitas opções, das quais as mais comuns são:

- `a`: exibe os processos iniciados por todos os usuários;
- `x`: exibe os processos sem terminal de controle ou com um terminal de controle diferente do que você está utilizando;
- `u`: exibe para cada processo o seu dono e o horário em que ele foi iniciado.

Há muito mais opções. Verifique a página do manual ps(1) para mais informações.

A saída do `ps` é dividida em campos diferentes: o mais interessante é o campo `PID` que contém o identificador do processo. O campo `CMD` contém o nome do comando executado. Uma maneira muito comum de executar o `ps` é a seguinte:

```
$ ps ax | less
```

Isto lhe dá uma lista de todos os processos que estão sendo executados, assim você pode identificar um ou mais processos que estão causando problemas e, subsequentemente, terminá-los.

10.2.2. pstree

O comando `pstree` exibe processos na forma de árvore. Uma vantagem é que você pode identificar imediatamente os pais dos processos: quando você quer matar toda uma série de processos e eles são todos pais e filhos, você pode simplesmente matar o processo pai. Você usará a opção `-p` para exibir o PID de cada processo, e a opção `-u` para exibir o nome do usuário que iniciou o processo. Como a estrutura em árvore é normalmente bem grande, você precisaria chamar o `pstree` da seguinte maneira:

```
$ pstree -up | less
```

Isto lhe dá uma melhor visualização de toda a árvore de processos.

10.3. Envando Sinais para Processos: kill, killall e top

10.3.1. kill, killall

Estes dois comandos são utilizados para enviar sinais aos processos. O comando `kill` pede um número de processo como argumento, enquanto o `killall` pede o nome do processo.

Ambos os comandos podem receber opcionalmente o número do sinal que será enviado como argumento. Por padrão, ambos enviam o sinal 15 (`TERM`) para os processos relevantes. Por exemplo, se você quiser matar o processo com PID 785, entre com o comando:

```
$ kill 785
```

Se você que eviar o sinal 19 (`STOP`), execute:

```
$ kill -19 785
```

Suponha que no lugar disso, você quer matar um processo que você conhece o nome. No lugar de encontrar o número do processo utilizando o `ps`, você pode matar o processo através do nome dele. Se o nome do processo é “mozilla” você poderia executar o comando:

```
$ killall -9 mozilla
```

Não importa o que acontecer, você somente irá matar seus próprios processos (a não ser que você esteja logado como `root`), então você não precisa se preocupar com os processos dos outros usuários caso você esteja em um istema multiusuário, pois eles não serão afetados.

10.3.2. Misturando o ps e o kill: top

O `top` é um programa que simultaneamente oferece as funções do `ps` e do `kill`, e também é utilizado para monitorar processos em tempo real, oferecendo informações sobre o uso de CPU e memória, tempo de execução e outros, como mostrado na Figura 10-1.

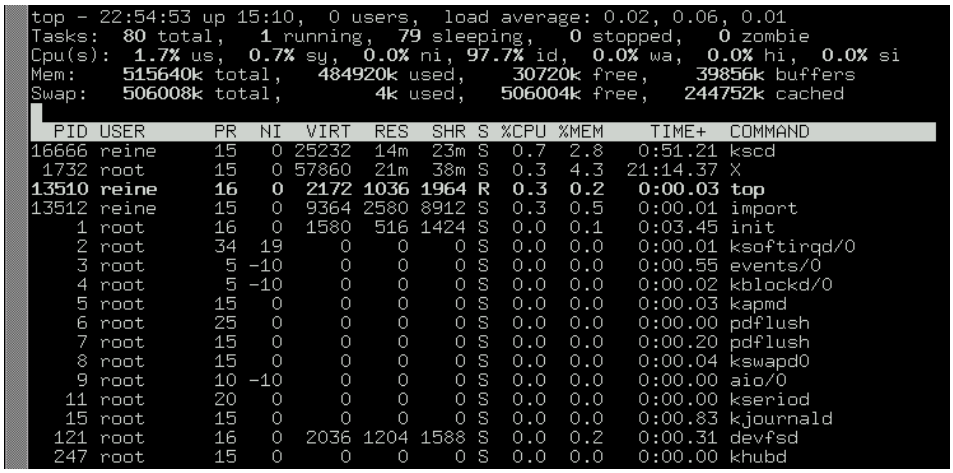


Figura 10-1. Monitorando Processos com o top

O `top` é completamente controlado pelo teclado. Você pode acessar a ajuda pressionando a tecla **h**. Seus comandos mais úteis são:

- **k**: este comando é utilizado para enviar um sinal para um processo. O `top` então irá te perguntar pelo PID do processo seguido do número ou nome do sinal que será enviado (`TERM` ou `15` por padrão);
- **M**: este comando é utilizado para ordenar os processos através da quantidade de memória que eles utilizam (campo `%MEM`);
- **P**: este comando é utilizado para ordenar os processos pelo tempo de CPU que eles utilizam (campo `%CPU`): este é o método padrão de ordenamento;
- **u**: este é utilizado para exibir os processos de um determinado usuário. O `top` irá perguntar qual é o usuário e, então, você precisará informar o nome do usuário, e não o seu UID. Se você não informar nenhum nome, todos os processos serão exibidos;
- **i**: por padrão, todos os processos, incluindo os parados, são exibidos. Este comando garante que somente os processos que estão sendo executados no momento sejam exibidos (processos cujo campo `STAT` possui o valor `R`, *Running*) e não outros. Utilizando este comando novamente o `top` voltará a exibir todos os processos.
- **r**: este comando é exibido para alterar a prioridade de um processo selecionado.

10.4. Configurando Prioridade para Processos: nice, renice

Cada processo no sistema é executado com prioridades definidas, também conhecidas como "valor nice", que pode variar de -20 (prioridade mais alta) a 19 (prioridade mais baixa). Se não definida, cada processo será executado com a prioridade 0 (a prioridade para agendamento "base"). Processos com uma prioridade maior (valor nice menor, até -20) serão agendados para executar com mais frequência do que os que possuem prioridade menor (até 19), garantindo desta forma mais ciclos do processador para eles. Usuários normais somente podem abaixar a prioridade dos processos que eles possuem dentro dos valores de 0 a 19. O superusuário (`root`) pode configurar a prioridade de qualquer processo para qualquer valor.

10.4.1. renice

Se um ou mais processos usam muito recurso do sistema, você pode alterar as suas prioridades em vez de matá-los. Para fazer isto, use o comando `renice`. A sua sintaxe é:

```
renice prioridade [[-p] pid ...] [[-g] pgrp ...] [[-u] usuário ...]
```

onde `prioridade` é o valor da prioridade, `pid` (use a opção `-p` para processos múltiplos) é o ID do processo, `pgrp` (use a opção `-g` se há mais de um) é o ID do grupo do processo, e `usuário` (`-u` para mais de um) é o dono do processo.

Vamos supor que você tenha um processo sendo executado com o PID 785, que está realizando um cálculo científico longo e complexo, e enquanto ele calcula, você quer jogar um jogo que necessitará de mais recursos livres no sistema. Então você poderia digitar:

```
$ renice +15 785
```

Neste caso o seu processo vai tomar um tempo maior para completar, mas não vai utilizar tempo de CPU de outros processos.

Se você é o administrador do sistema e percebe que algum usuário está executando muitos processos e utilizando muito recurso, você pode alterar a prioridade dos processos daquele usuário com um único comando:

```
# renice +20 -u usuario2
```

Depois disto, todos os processos do usuario2 terão a prioridade mais baixa e não irão obstruir qualquer outro processo lançado por outros usuários.

10.4.2. nice

Agora que você sabe que pode alterar a prioridade dos procesos, você pode querer executar um comando com uma prioridade definida. Para isto, utilize o comando `nice`.

Neste caso você precisará especificar o seu comando como uma opção do `nice`. A opção `-n` é utilizada para configurar o valor da prioridade. Por padrão o `nice` configura a prioridade como 10.

Por exemplo, se você quer criar uma imagem ISO de um CD-ROM de instalação do Mandriva Linux:

```
$ dd if=/dev/cdrom of=~/mandriva1.iso
```

Em alguns sistemas, o processo de copiar grande quantidade de informação de um CD-ROM IDE padrão pode utilizar muito recurso do sistema. Para evitar que a cópia bloqueie outros processos, você pode iniciar o processo com uma prioridade mais baixa, através do comando:

```
$ nice -n 19 dd if=/dev/cdrom of=~/mandriva1.iso
```

Capítulo 11. Os Arquivos de Inicialização: init sysv

O esquema de inicialização System V é herdado do UNIX® AT&T e é um dos esquemas tradicionais do UNIX® para iniciar o sistema. Ele é responsável por iniciar ou parar serviços para deixar o sistema em um de seus modos padrões. Serviços é um termo que abrange desde autenticação básica de usuários até servidor gráfico local ou serviços de internet.

11.1. Introdução

Quando o sistema inicia, e após o kernel ter configurado tudo e montado o sistema de arquivos raiz, ele executa o `/sbin/init`[1]. O `init` é o pai de todos os processos do sistema e é o responsável por conduzir o sistema ao ***nível de execução*** desejado. Nós vamos cobrir os níveis de execução depois (veja a Seção 11.2).

O arquivo de configuração do `init` é o `/etc/inittab` e tem a sua própria página de manual (inittab(5)), então nós vamos documentar alguns dos valores possíveis para configuração.

A primeira linha que deveria ser o foco da sua atenção é a seguinte:

```
si::sysinit:/etc/rc.d/rc.sysinit
```

Esta linha diz ao `init` que o `/etc/rc.d/rc.sysinit` deve ser executado assim que o sistema estiver inicializado (`si` significa *System Init*). Para determinar o nível de execução padrão, o `init` então procura pela linha contendo a palavra-chave `initdefault`:

```
id:5:initdefault:
```

Neste caso, o `init` sabe que o nível de execução padrão é 5. Ele também sabe que para entrar no nível 5, ele deve executar o seguinte comando:

```
l5:5:wait:/etc/rc.d/rc 5
```

Como você pode ver, a sintaxe para cada nível de execução é parecida.

O `init` também é responsável por reiniciar (`respawn`) alguns programas que não podem ser iniciados por qualquer outro processo. Por exemplo, cada um dos programas de login que são executados nos seis consoles virtuais são iniciados pelo `init`[2]. O segundo console virtual é identificado desta forma:

```
2:2345:respawn:/sbin/mingetty tty2
```

11.2. Níveis de Execução

Todos os arquivos relacionados a inicialização do sistema estão localizados no diretório `/etc/rc.d`. Aqui está a listagem dos arquivoss:

```
$ ls /etc/rc.d
init.d/  rc0.d/  rc2.d/  rc4.d/  rc6.d/            rc.local*    rc.sysinit*
rc*      rc1.d/  rc3.d/  rc5.d/  rc.alsa_default*  rc.modules*
```

Como já declarado anteriormente, o `rc.sysinit` é o primeiro arquivo a ser executado pelo sistema. Ele é responsável pela configuração básica da máquina: tipo de teclado, configuração de certos dispositivos, verificação de sistemas de arquivo, etc.

Então o script `rc` é executado utilizando como argumento o nível de execução desejado. Como já vimos, o nível de execução é um inteiro simples, e para cada nível `<x>` definido, deve haver um diretório `rc<x>.d` correspondente. Em uma instalação típica do Mandriva Linux, você pode ver que há seis níveis de execução:

- 0: desliga a máquina.
- 1: modo ***monousuário***. Para ser utilizado em casos de problemas graves ou recuperação do sistema.

1. E por isso é uma má idéia colocar o `/sbin` em um sistema de arquivo diferente do raiz. O kernel não montou nenhuma outra partição neste ponto, e então não conseguiria encontrar o arquivo `/sbin/init`.
2. Se você não quiser seis consoles virtuais, você pode adicionar ou remover alguns deles modificando este arquivo. Se você deseja aumentar o número de consoles, você poderá ter até 64. Mas não se esqueça que o X também é executado em um console virtual, então deixe ao menos um console livre para o X.

- 2: modo ***multiusuário***, sem suporte a rede.
- 3: modo multiusuário, com suporte a rede
- 4: não utilizado.
- 5: como o nível de execução 3, mas com a diferença de que executa a interface gráfica de login.
- 6: reinicia a máquina.

Vamos dar uma olhada no conteúdo do diretório `rc3.d`:

```
$ ls /etc/rc.d/rc3.d/
K09dm@          S12syslog@    S24messagebus@     S40atd@          S91dictd-server@
S01udev@        S13partmon@   S25haldaemon@      S55sshd@         S92lisa@
S03iptables@    S15cups@      S25netfs@          S56ntpd@         S95kheader@
S05harddrake@   S17alsa@      S29numlock@        S56rawdevices@   S99local@
S10network@     S18sound@     S33nifd@           S75keytable@
S11shorewall@   S20xfs@       S34mDNSResponder@  S90crond@
$
```

Como você pode ver, todos os arquivos neste diretório são links simbólicos, e todos eles possuem uma forma muito específica. A forma geral deles é:

```
<S|K><order><service_name>
```

O `S` significa iniciar (*Start*) o serviço, e o `K` significa matá-lo (*Kill*), parar o serviço. Os scripts são executados em uma ordem numérica ascendente, e se dois scripts possuem o mesmo número, uma ordem alfabética será utilizada. Nós também podemos ver que cada link simbólico aponta para um script localizado no diretório `/etc/init.d` (algum que não seja o script `local` que é responsável por controlar um serviço específico).

Quando o sistema entra em um certo nível de execução, ele começa executando os links `K` em ordem: o comando `rc` verifica para onde o link está apontando, e então chama o script correspondente com um único argumento: `stop`. Ele então executa os scripts `S` utilizando o mesmo método, exceto aqueles que os scripts são chamados com um parâmetro `start`.

Então, sem examinar todos os scripts, nós podemos ver que quando o sistema entra no nível de execução 3, ele primeiro executa o comando `K09dm`, (por exemplo, `/etc/init.d/dm stop`). Depois, ele executa todos os scripts `S`: primeiro o `S01udev`, que por sua vez chama o `/etc/init.d/udev start`, então o `S03iptables`, e assim por diante.

Armado com esta informação, você pode criar seu próprio nível de execução em poucos minutos (utilizando o nível 4, por exemplo), ou evitar que um serviço inicie ou pare deletando o link simbólico correspondente.

11.2.1. Configurando Serviços nos Níveis de Execução

Você também pode utilizar o comando `chkconfig` para adicionar, remover, ativar ou desativar os serviços de um certo nível. Use `chkconfig --add nome_do_serviço` para adicionar (ativar) o serviço `nome_do_serviço` para todos os níveis de execução suportados[3] e `chkconfig --del nome_do_serviço` para remover (desativar) o serviço indicado de todos os níveis de execução.

Execute `chkconfig --list` para saber quais serviços estão disponíveis, o seus nomes, e o status de cada um em todos os níveis de execução definidos.

Executando o comando `chkconfig --levels 35 sshd on` irá ativar o servidor SSH (`sshd`) nos níveis de execução 3 e 5, enquanto o comando `chkconfig --levels 3 sound off` removerá o suporte a som do nível de execução 3. Se você omitir o parâmetro `--levels lista_de_níveis`, o serviço nomeado será ativado ou desativado nos níveis 2, 3, 4 e 5. Note entretanto, que você pode terminar habilitando serviços nos niveis sem suportar devidamente estes serviços, então é melhor especificar os níveis de execução que devem ser alterados.

3. Níveis de execução "suportados" significam que, por exemplo, um serviço de rede não seria adicionado ao nível de execução 2, que não suporta rede.

11.2.2. Controlando Serviços em um Sistema em Execução

Serviços podem ser controlados em um sistema que já está em execução através do comando `service`, estejam eles configurados ou não para serem executados em um nível de execução particular. Sua sintaxe é a seguinte:

```
service nome_do_serviço ação
```

Onde `nome_do_serviço` é o nome do serviço a ser controlado, escrito da mesma forma que a listada no `chkconfig --list`, e `ação` pode ser:

start

Inicia o serviço indicado. Note que, se você tentar iniciar um serviço que já está em execução, o sistema lhe avisará disto. Se a intenção for reiniciar o serviço, utilize o `restart`, como mostrado abaixo.

stop

Pára o serviço indicado. Note que todos os usuário conectados a este serviço serão automaticamente desconectados quando você pará-lo.

restart

Pára e inicia o serviço indicado. É o equivalente a executar `service nome_do_serviço stop && service nome_do_serviço start`. Note que todos os usuários conectados ao serviço serão automaticamente desconectados quando o serviço reiniciar.

outras ações que são dependentes de serviço

Serviços diferentes suportam ações diferentes (as anteriores são suportadas por todos os tipos de serviços). Por exemplo, `reload` para recarregar o arquivo de configuração de um serviço; `force-stop` para forçar o desligamento do serviço; `status` para verificar o estado do serviço no momento; etc. Execute `service nome_do_serviço` para ser informado de todas as ações suportadas por ele.

Capítulo 12. Acesso Remoto Seguro

Administradores de sistema precisam se conectar a máquinas que estão fisicamente distantes para editar arquivos de configuração, controlar serviços, executar programas, etc. O `telnet` era utilizado para acessar sistemas remotos, entretanto ele não é uma solução segura. E já que todas as comunicações ocorrem na internet pública (e naturalmente insegura), os administradores de sistema precisam de uma solução para acesso remoto seguro. O `ssh` (que significa **S**ecure **SH**ell - Shell Seguro) permite que você acesse uma máquina remota de maneira segura, criptografando todas as comunicações.

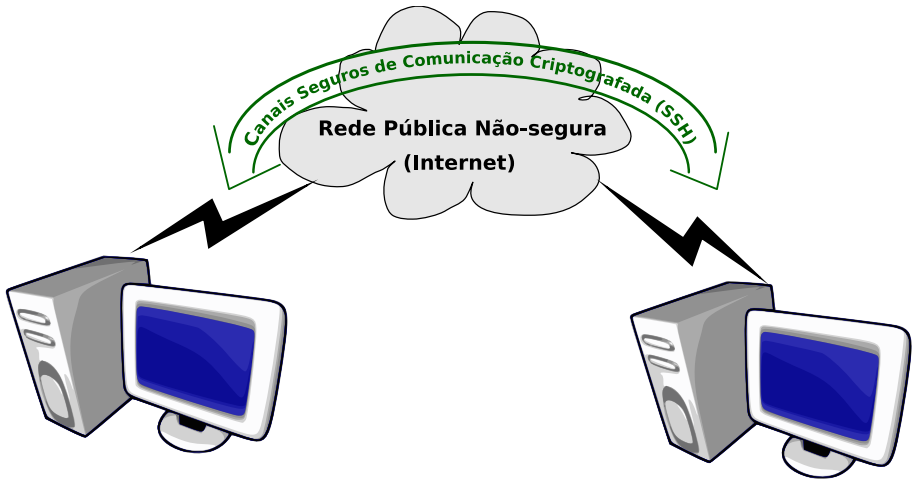


Figura 12-1. Esquema de Conexão SSH

12.1. Configuração do Servidor SSH

Por "servidor" nós queremos dizer a máquina na qual você irá conectar. Esteja certo de que o pacote `openssh-server` está instalado, e que o serviço `sshd` está sendo executado[1].

A configuração básica do servidor SSH permite aos usuários acessarem uma máquina, desde que eles tenham uma conta nela. Se você quer restringir acesso SSH para uma lista de usuários, edite o arquivo `/etc/ssh/sshd_config` e adicione ou modifique uma linha como a seguinte:

```
AllowUsers usuario1 usuario2@192.168.0.*
```

O exemplo acima irá permitir que somente os usuários usuario1 e usuario2 possam se conectar através do SSH na máquina; o usuario2 está autorizado a acessar a máquina somente a partir da rede (local) `192.168.0`.

Para proibir que os usuários possam se conectar como `root` diretamente através do SSH, altere a linha `PermitRootLogin yes` para `PermitRootLogin no`. Assim, os usuários terão que se conectar com suas contas normais e utilizarem o comando `su` para se tornarem `root`.

Verifique sshd(8) e sshd_config(5) para mais informações sobre as opções e configuração do servidor SSH.

12.2. Configuração do Cliente SSH

Por "cliente" nós queremos dizer a máquina na qual você está conectado. Certifique-se de que o pacote `openssh-clients` está instalado.

Digite `ssh username@remote_machine` para se conectar a `remote_machine` com a conta `username`. Você terá que informar o seu password no sistema remoto. Depois disto você terá acesso garantido, como se estivesse sentado em frente ao console do sistema remoto.

Se você conectar a uma ou muitas máquinas (um cenário comum para muitos administradores), a solicitação da senha pode ser contornada com o uso de chaves SSH. Utilize o comando `ssh-keygen` para gerar a sua chave, e então o comando `ssh-copy-id username@remote_machine` para copiá-la para as máquinas remotas. Quando você executar o comando `ssh-copy-id` você terá que informar a senha no sistema remoto,

1. O serviço SSH é iniciado automaticamente após instalar o pacote `openssh-server` e também é configurado para ser executado durante a inicialização do sistema.

mas apenas uma vez em cada sistema. Agora você pode fazer uma conexão SSH diretamente na máquina remota sem ter que informar a senha.

Para este mecanismo funcionar você deve executar o comando `ssh-add` e entrar com a sua passphrase — criada quando você gerou as chaves SSH — cada vez que você começar uma sessão na máquina cliente.

Se você receber uma mensagem informando que a conexão para o seu agente de autenticação não pode ser aberta, execute o comando `eval 'ssh-agent'` (perceba as crases) antes de executar `ssh-add`.

12.3. Copiando Arquivos Entre um Sistema Remoto

Para trasferir arquivos para um sistema remoto que esteja executando um servidor SSH, utilize o comando `scp`,que significa **S**ecure **C**o**P**y (cópia segura). A sintaxe do comando é a seguinte:

```
scp [opções] caminho_local [usuario@]host_remoto:[caminho_completo_no_host_remoto]
```

Se você não especificar a parte `usuario@`, então o seu login será utilizado. Se você omitir o caminho na máquina remota, o arquivo será copiado no diretório home do `usuário` no sistema remoto. Note que o dois-pontos (`:`) separa o nome do usuário e a especifação da máquina do caminho no servidor remoto.

Para transferir arquivos do sistema remoto para a máquina local, a sintaxe é a seguinte:

```
scp [opções] [usuario@]host_remoto:caminho_completo_para_o_host_remoto
```

Se o caminho de origem for um diretório, então a opção `-r` (recursivo) é obrigatória. Verifique scp(1) para mais informações sobre as opções do `scp`.

Capítulo 13. Gerenciamento de Pacotes na Linha de Comando

Programas como Rpmdrake são apenas interfaces gráficas para as poderosas ferramentas de linha de comando urpmi. Para aqueles que desejarem gerenciar os pacotes através da linha de comando (bastante útil para tarefas remotas, por exemplo), nós apresentamos aqui os comandos mais usados.

13.1. Instalando e Removendo Pacotes

Isto é feito através de dois comandos simples:

`urpmi <nome_do_pacote>`

Através deste comando o `nome_do_pacote` será instalado se existir. Se o pacote apenas contiver o texto `nome_do_pacote`, ele também será instalado. Se mais de um pacote contiver este texto, será apresentada uma lista numerada de sugestões: basta digitar o número do pacote desejado e pressionar **Enter**.

Se o pacote sendo instalado tiver dependências (ou seja, outros pacotes serão necessários), uma lista de pacotes adicionais será exibida. Analise-a e, estando satisfeito, pressione a tecla **Y** para prosseguir com a instalação.

`urpme <nome_do_pacote>`

Este comando irá remover o pacote `nome_do_pacote`. Caso outros pacotes instalados dependam desde que está sendo removido, eles serão listados juntamente com o motivo da sua remoção. Analise esta lista e pressione a tecla **Y** para remover os pacotes exibidos.

Ambos os comandos `urpmi` e `urpme` suportam o parâmetro `--auto` para a instalação e remoção automática de dependências.

Consulte as páginas de manual urpmi(8) e urpme(8) para mais informações sobre as opções disponíveis para estes comandos.

13.2. Gerenciamento de Mídias

Mídias de software são as diferentes "fontes" a partir de onde pacotes podem ser instalados. Deve existir pelo menos uma mídia definida para que o comando `urpmi` funcione. Mídias predefinidas incluem aquelas que foram usadas para instalar o sistema (rede, CD, DVD, etc.). Outras mídias devem ser definidas para a obtenção de atualizações de segurança e outras correções. Adicionar e remover mídias é fácil deste que a sintaxe seja estritamente observada e seguida.

13.2.1. Acrescentando Novas Mídias

`urpmi.addmedia <nome> <url>`

Este comando permite que uma nova mídia seja acrescentada a partir de um disco local, dispositivo removível (CD/DVD) ou a partir da rede através dos protocolos HTTP, FTP, NFS, `ssh` ou `rsync`. A sintaxe da URL varia com cada um destes tipos de mídia, portanto é recomendado que se consulte a página de manual urpmi.addmedia(8) antes de usá-la.

Se a mídia sendo acrescentada for uma mídia de atualizações, acrescente o parâmetro `--update` à linha de comando do `urpmi.addmedia`.

Recursos online como a página Easy Urpmi (`http://easyurpmi.zarb.org/`) podem ser utilizados para encontrar definições prontas de mídias com diversas aplicações úteis para seu sistema Mandriva Linux. O Clube

Mandriva (`http://club.mandriva.com/`) também oferece o módulo Urpmi media (`http://club.mandriva.com/modules.php?name=Mirrors-list`) para o teste e a contribuição de pacotes.

A lista de mídias do Clube Mandriva somente está disponível para membros.

13.2.2. Removendo Mídias

`urpmi.removemedia <nome>`

Este comando irá simplesmente remover a mídia `nome`. Se você não consegue se lembrar do nome da mídia que deseja remover, basta executar o comando `urpmi.removemedia` sem parâmetros que todas as mídias definidas serão listadas.

13.2.3. Atualizando Mídias

`urpmi.update <nome>`

Este comando analisa a mídia fornecida e atualiza a lista de pacotes associada. É útil para mídias com mudanças freqüentes, como aquelas que oferecem atualizações de segurança e demais correções. Utilize a opção `-a` para atualizar todas as mídias definidas.

13.2.4. Ordenação das Mídias

A ordem em que as mídias são definidas no arquivo `/etc/urpmi/urpmi.cfg` é importante pois ela determina a partir de que mídia a instalação será efetuada caso mais de uma forneça os pacotes necessários: eles serão instalados a partir da primeira mídia que os fornecer.

Mídias de rede são automaticamente acrescentadas antes das mídias locais. Isto é feito pois se espera que mídias de rede contenham pacotes mais atualizados que as mídias locais ou de dispositivos removíveis.

13.3. Dicas e Truques

13.3.1. Listas Resumidas vs. Completas

Ao acrescentar uma mídia, há duas opções para a lista de pacotes: resumida ou completa. Use a opção `--probe-synthesis` para tentar utilizar a lista resumida, ou `--probe-hdlist` para tentar utilizar a lista completa. Listas resumidas são bem menores que as completas, sendo mais indicadas para usuários com conexões mais lentas. Por outro lado, elas são mais limitadas em relação às informações disponíveis sobre os pacotes.

13.3.2. Encontrando o Pacote que Contém um Arquivo Específico

Você sabe que precisa de um arquivo específico no seu sistema mas não sabe que pacote o fornece? Utilize o comando `urpmf <nome_do_arquivo>` e quaisquer pacotes, instalados ou não, que contiveram o arquivo `nome_do_arquivo` serão listados.

Se você estiver usando as listas resumidas, o comando `urpmf` realizará a pesquisa apenas nos pacotes que já estiverem instalados.

Você também pode simplesmente fornecer um nome parcial. Por exemplo, o comando `urpmf salsa` exibirá uma lista de todos os pacotes que tiverem um arquivo com a palavra `salsa` no seu nome.

```
[root@test usuario1]# urpmf salsa
    kaffe:/usr/lib/kaffe/lib/i386/libtritonusalsa-1.1.2.so
    kaffe:/usr/lib/kaffe/lib/i386/libtritonusalsa.la
    kaffe:/usr/lib/kaffe/lib/i386/libtritonusalsa.so
```

13.3.3. Atualizando Pacotes

O comando a seguir irá atualizar o pacote mencionado:

```
urpmi.update -a && urpmi --update <nome_do_pacote>
```

Este comando irá automaticamente atualizar todos os pacotes possíveis da mesma forma que Mandriva Update o faria:

```
urpmi.update -a && urpmi --update --auto-select --auto
```

Se você não possui nenhuma mídia marcada especificamente como sendo de atualização, é necessário remover o parâmetro `--update` da linha de comando anterior do urpmi.

Apêndice A. Glossário

conta
Em um sistema UNIX®, é a combinação de um nome, um diretório pessoal, uma senha e um shell que permite a um usuário conectar ao sistema.

alias
Um mecanismo utilizado em um shell para substituir uma string por outra antes de executar um comado. Você pode ver todos os aliases definidos na sessão atual digitando `alias` no prompt.

ACPI
Advanced Configuration and Power Interface. Serve para configurar e gerenciar o uso de energia do computador. Ao contrário do APM, que conta somente com a BIOS, o ACPI também conta com o sistema operacional, o que torna mais simples de ser operado pelo usuário. O ACPI também oferece recursos de gerenciamento de energia para servidores e estações de trabalho.

APM
Advanced Power Management. Utilizada por algumas BIOS para fazer a máquina entrar em modo de espera após um certo período de inatividade. Em laptops, o APM também é responsável por reportar o status da bateria e (se suportado) o tempo de carga que ainda resta na bateria. Porém, laptops novos utilizam o ACPI no lugar do APM.
Ver Também: ACPI.

ARP
Address Resolution Protocol. Protocolo utilizado para mapear dinamicamente um endereço na internet para uma endereço físico (hardware) em uma rede local. O uso do protocolo é limitado a redes que possuem suporte a hardware broadcasting.

ASCII
American Standard Code for Information Interchange. O código padrão utilizado para o armazenamento de caracteres em um computador, incluindo caracteres de controle. Muitas codificações de 8-bit (como ISO 8859-1, normalmente o conjunto de caracteres padrão do Linux, a não ser que você tenha escolhido algo como UTF-8) possuem o ASCII como sua base.
Ver Também: ISO 8859, UTF-8.

linguagem assembly
É a linguagem de programação mais próxima do computador, e é por isso que é conhecida como uma linguagem de programação de "baixo nível". Assembly tem a vantagem da velocidade, já que programas assembly são escritos em termos de instruções pequenas do processador, além de não requerer "tradução" ao gerar executáveis. Sua desvantagem principal é a de que ele é dependente de processador (ou arquitetura), além de exigir bastante tempo para escrever sistemas muito complexos. Assim, assembly é a linguagem de programação mais rápida, mas também a menos portável entre as arquiteturas.

ATAPI
AT Attachment Packet Interface. Uma extensão à especificação ATA (*Advanced Technology Attachment*, mais conhecida como IDE, *Integrated Drive Electronics*) que fornece comandos extras para controlar drives de CD-ROM e de fita magnética. Controladores IDE equipados com esta extensão também são chamados de controladores EIDE (*Enhanced IDE*).
Ver Também: IDE.

ATM
Acrônimo para **Asynchronous Transfer Mode**. Uma rede ATM empacota os dados em blocos de tamanho padrão (53 bytes: 48 para os dados e 5 para o cabeçalho) que pode ser transportado eficientemente de uma ponta a outra.

atômico
Um conjunto de operações é chamado de atômico quando todas elas são executadas de uma só vez. Ou todas as operações obtém sucesso, ou nenhuma delas é levada em consideração. Pode ser utilizado para operações essenciais ou muito simples, como a soma de números integrais.

segundo plano
No contexto do shell, um processo é executado em segundo plano se você pode continuar a executar outros comandos no shell enquanto ele continua a ser executado. É o contrário do processo em primeiro plano.
Ver Também: job, foreground.

backup
Um meio de salvar informações importantes em uma mídia e local mais seguros. Os backups devem ser realizados regularmente, especialmente para as informações mais críticas e arquivos de configuração (os diretórios mais importantes para se fazer backup são o `/etc`, `/home` e `/usr/local`). Muitas pessoas utilizam o `tar` com o `gzip` ou `bzip2` para fazer o backup de diretórios e arquivos. Você também pode usar programas como o `dump` e `restore`, assim como qualquer outra solução para backup comercial ou livre.

processamento em lote
Um modo de processamento onde tarefas ou instruções que são enviadas ao CPU são executadas seqüencialmente até que todas sejam processadas.

beep
O pequeno barulho que o speaker do computador emite para lhe avisar de alguma situação ambígua, como por exemplo quando você está usando a completação de comandos pelo shell e há mais de uma opção possível para completar. Podem haver outros programas que utilizarão o beep para lhe avisar alguma situação específica.

beta testing
O nome dado ao processo de teste da versão beta de um programa. Programas normalmente são lançados em versões "alpha", "beta" e "release candidate" para que testes sejam realizados antes da versão final.

binário
No contexto de programação, binários são códigos compilados, executáveis.

bit
Quer dizer *BInary digiT*. Um único bit que pode receber os valores 0 ou 1, porque o cálculo é feito em base binária. É a unidade mais básica de informação digital.

aquivos de bloco
Arquivos cujos conteúdos são armazenados em buffers. Todas as operações de leitura/escrita são realizadas através de buffers, que permitem leitura assíncrona de dados e escrita ao hardware suportado, evitando assim que acesso desnecessário seja feito ao disco se os dados já estiverem armazenados em um buffer.
Ver Também: buffer, buffer cache, arquivos de caractere.

boot
O procedimento executado assim que o computador é ligado, onde os periféricos são reconhecidos seqüencialmente e o sistema operacional é carregado na memória.

disco de boot
Um disco inicializável contendo o código necessário para carregar um sistema operacional a partir do disco rígido.

gerenciador de inicialização
É o programa que inicia o sistema operacional. Muitos gerenciadores de inicialização dão a oportunidade de carregar mais de um sistema operacional, permitindo que você faça a sua escolha em um menu. Gerenciadores de inicialização como o GRUB e LILO são populares por sua utilidade em sistemas multi ou dual-boot.

BSD
Berkeley Software Distribution. Uma variante do UNIX® desenvolvido pelo departamento de informática da Universidade de Berkeley. Esta versão sempre foi considerada a mais avançada tecnicamente. Trouxe muitas inovações ao mundo da informática e, em particular, ao UNIX®.

buffer
Uma pequena porção de memória com tamanho fixo, que pode ser associada a um arquivo de bloco, uma tabela de sistema, um processo e etc. O buffer cache mantém a coerência de todos os buffers.
Ver Também: buffer cache.

buffer cache
Uma parte crucial do kernel de um sistema operacional, ele mantém todos os buffers atualizados, diminuindo o cache quando necessário, limpando buffers que não estão mais sendo usados e etc.
Ver Também: buffer.

bug
Comportamento ilógico ou incoerente de um programa em um caso especial, ou um comportamento que não segue a documentação ou padrões aceitos pelo programa. Novas funcionalidades, normalmente, adicionam novos bugs ao programa. Historicamente, este termo vem dos tempos dos cartões perfurados: um inseto (*bug* em inglês) caiu em um buraco do cartão e, como conseqüência, o programa agiu de maneira errada. O Almirante Grace Hopper, quando descobriu o motivo, disse "É um inseto!" (*It's a bug!*, em inglês), e desde então o termo continuou a ser utilizado. Esta é apenas uma das muitas histórias existentes para a explicação do termo *bug*.

byte
Uma seqüência de, normalmente, oito bits consecutivos, que quando convertidos para a base decimal resultam em um inteiro entre 0 e 255. Um byte é sempre "atômico" para o sistema, sendo a menor unidade endereçável.
Ver Também: bit.

caso
Quando falamos em strings, o caso é a diferença entre letras minúsculas e maiúsculas.

CHAP
Challenge-Handshake Authentication Protocol: Um protocolo utilizado por provedores de internet para autenticar seus clientes. Desta forma, um valor é enviado para o cliente (a máquina que está fazendo a conexão), a qual costuma calcular um hash baseado neste valor. O cliente envia o hash para o servidor fazer uma comparação. Este método de autenticação é diferente do PAP, em que ele faz uma autenticação de tempo em tempo após a autenticação inicial.
Ver Também: PAP.

arquivos de caractere
Arquivos cujo conteúdo não são armazenados em buffers. Quando associado a dispositivos físicos, toda a entrada/saída nestes dispositivos é realizada imediatamente. Alguns dispositivos especiais de caractere são criados pelo sistema operacional (`/dev/zero`, `/dev/null` e outros). Eles correspondem a fluxo de dados.
Ver Também: aquivos de bloco.

CIFS
Common Internet File System. O sucessor do sistema de arquivos SMB, utilizado em sistemas DOS.
Ver Também: SMB.

cliente
Um programa ou computador que conecta esporadicamente, por um certo período de tempo, a outro programa ou computador para lhe passar comandos ou obter informações. No caso de sistemas **peer to peer** como SLIP ou PPP, o cliente é definido como a ponta que iniciará a conexão, e a ponta remota que recebe a conexão é definida como servidor. É um dos componentes de um **sistema cliente/servidor**.
Ver Também: server.

sistema cliente/servidor
Sistema ou protocolo que consiste em um **servidor** e um ou mais **clientes**.

linha de comando
É fornecida por um shell e permite que o usuário entre com comandos diretamente.

modo de comando
Sob o Vi ou algum de seus clones, é o estado do programa em que quando pressionamos uma tecla nenhum caractere é inserido no arquivo que está sendo editado, mas ao invés disto realiza uma ação específica da tecla (a não ser que o clone possua comandos que podem ser mapeados novamente, e que você tenha personalizado esta configuração). Você pode sair deste modo digitando um dos comandos que retornam ao modo de inserção: **i**, **I**, **a**, **A**, **s**, **S**, **o**, **O**, **c**, **C**, ...

compilação
É o processo de transformar o código fonte, que pode ser entendido por humanos (com algum treino) e são escritos em alguma linguagem de programação (C, por exemplo), em um arquivo binário que pode ser entendido e executado pela máquina.

completação
A habilidade de um shell em expandir automaticamente uma substring para um nome de arquivo, nome de usuário ou outro item que combine com ela.

compressão
Uma forma de diminuir o tamanho de arquivos ou o número de caracteres enviados através de um link de comunicação. Entre os programas de compressão de arquivos estão o compress, `zip`, `gzip`, e `bzip2`.

console
Este é o nome dado ao que costumava ser chamado de terminais. Eles eram as máquinas (uma tela e um teclado) conectadas a um mainframe central. Nos PCs, o terminal físico é o teclado e a tela.
Ver Também: virtual console.

cookies
Arquivos temporários escritos no disco rígido local por um servidor web remoto. Ele permite que o servidor "lembre" as preferências do usuário quando este usuário conectar novamente.

datagram
Um datagram é um pacote discreto de dados e cabeçalhos que contém endereços. É a unidade básica de transmissão através de uma rede IP. Você também pode ter ouvido falar dele como um "pacote de informação".

dependências
Os estágios de compilação que precisam ser satisfeitos antes de prosseguir para outros estágios para compilar com sucesso um programa. Este termo também é utilizado quando um conjunto de programas que você deseja instalar depende de outros programas que podem ou não estar instalados em seu sistema. Neste caso uma mensagem lhe informará que será necessário "satisfazer as dependências" para continuar a instalação.

desktop
Se você estiver utilizando o X Window System, o desktop é o lugar na tela onde você trabalha e onde são exibidos os ícones e janelas.
Ver Também: virtual desktops.

DHCP
Dynamic Host Configuration Protocol. Um protocolo desenvolvido para máquinas em uma rede local que possibilita aos clientes obterem dinamicamente um endereço IP e outras configurações de rede a partir de um servidor.

diretório
Parte da estrutura do sistema de arquivos. Arquivos ou outros diretórios podem ser armazenados em um diretório. Às vezes há subdiretórios dentro de um diretório, e esta estrutura é conhecida como árvore de diretórios. Se você quer ver o que tem dentro de outro diretório, você tem que listá-lo ou acessá-lo. Diretórios seguem as mesmas restrições que arquivos, embora as permissões tenham significados diferentes. Os diretórios especiais `.` e `..` referem-se ao próprio diretório e ao diretório pai (um nível acima), respectivamente. Em um ambiente gráfico os diretórios são conhecidos como pastas.

valores discretos
São valores que não são contínuos, isto é, existe algum tipo de "espaço" entre os valores consecutivos.

distribuição
Uma distribuição é feita com o kernel do Linux e utilitários, assim como programas de instalação, programas de terceiros e, às vezes, software proprietário.

DLCI
O DLCI - *Data Link Connection Identifier* - é utilizado para identificar uma única conexão virtual ponto-a-ponto através de uma rede Frame Relay. Os DLCIs são normalmente atribuídos ao provedor da rede Frame Relay.

DMA
Direct Memory Access. Uma facilidade utilizada na arquitetura do PC que permite a um periférico ler ou escrever na memória principal sem a ajuda do CPU. Periféricos PCI utilizam o barramento e não precisam da DMA.

DNS
Domain Name System. Mecanismo de nome e endereço utilizado na internet, que lhe permite mapear um domínio para um endereço de IP, permitindo que você acesse um site através do domínio, sem ter que saber o endereço IP do site. O DNS também permite pesquisa reversa, ou seja, obter o nome através do endereço IP da máquina.

DPMS
Display Power Management System. Protocolo utilizado por todos os monitores modernos para gerenciar o controle de uso de energia. Monitores que possuem esta tecnlogia são chamados de monitores verdes ("green monitors").

echo
Ocorre quando os caracteres que você digita são exibidos na tela, como quando informamos o nome de usuário durante a sessão de login, por exemplo. Alguns programas também podem mascarar o que digitamos por questões de segurança, como por exemplo quando informamos uma senha e ela é substituída por $\star$ em cada caractere que digitamos.

editor
Is a term typically used for programs which edit text files (aka text editor). The most well-known GNU/Linux editors are the GNU Emacs (Emacs) editor and the UNIX® editor Vi.

ELF
Executable and Linking Format. This is the binary format used by most GNU/Linux distributions.

email
Stands for Electronic Mail. This is a way to send messages electronically. Similar to regular mail (aka snail mail), email needs a destination and sender address to be sent properly. The sender must have an address like "sender@senders.domain" and the recipient must have an address like "recipient@recipients.domain." Email is a very fast method of communication and typically only takes a few minutes to reach anyone, regardless of where in the world they are located. In order to write email, you need an email client such as pine or `mutt` which are text-mode clients, or GUI clients such as KMail.

environment
Is the execution context of a process. It includes all the information that the operating system needs to manage the process and what the processor needs to execute the process properly.
Ver Também: process.

environment variables
A part of a process' environment. Environment variables are directly viewable from the shell.
Ver Também: process.

escape
In the shell context, is the action of surrounding a string with quotes to prevent the shell from interpreting that string. For example, when you need to use spaces in a command line and then pipe the results to some other command you have to put the first command between quotes or precede the spaces with a \ ("escape" the command) otherwise the shell will interpret it incorrectly and your command won't work as expected.

ext2
Short for the "Extended 2 file system". This is GNU/Linux's native file system and has the characteristics of any UNIX® file system: support for special files (character devices, symbolic links, etc), file permissions and ownership, and other features.

FAQ
Frequently Asked Questions. A document containing a series of questions and answers about a specific topic. Historically, FAQs appeared in newsgroups, but this sort of document now appears on various web sites, and even commercial products have FAQs. Generally, they are very good sources of information.

FAT
File Allocation Table. File system used by DOS and Windows®.

FDDI
Fiber Distributed Digital Interface. A high-speed network physical layer, which uses optical fiber for communication instead of wire. Mostly used on large networks, mainly because of its price. It is rarely seen as a means of connection between a PC and a network switch.

FHS
File system Hierarchy Standard. A document containing guidelines for a coherent file tree organisation on UNIX® systems. Mandriva Linux complies with this standard in most aspects.

FIFO
First In, First Out. A data structure or hardware buffer where items are taken out in the order they were put in. UNIX® pipes are the most common examples of FIFOs.

filesystem
Scheme used to store files on physical media (hard drive, floppy, etc.) in a consistent manner. Examples of file systems are FAT, GNU/Linux' ext2fs, ISO9660 (used by CD-ROMs) and so on. An example of a virtual filesystem is the `/proc` filesystem.

firewall
A machine or a dedicated piece of hardware which in the topology of a local network is the single connection point to the outside network, and which filters and controls the activity on some ports, or makes sure that only some specific interfaces may have access to, or can be accessed from, the outside world.

flag
Is an indicator (usually a bit) that is used to signal some condition to a program. For example, a filesystem has, among others, a flag indicating if it has to be dumped in a backup, so when the flag is active the filesystem gets backed up, and when it's inactive it doesn't.

focus
The state for a window to receive keyboard events (such as key-presses, key-releases and mouse clicks) unless they are trapped by the window manager.

foreground
In shell context, the process in the foreground is the one that is currently running and has keyboard and screen control. You have to wait for such a process to finish in order to be able to type commands again.
Ver Também: job, segundo plano.

Frame Relay
Frame Relay is a network technology ideally suited to carrying traffic which is of a bursty or sporadic nature. Network costs are reduced by having many Frame Relay customers sharing the same network capacity and relying on them wanting to make use of the network at slightly different times.

framebuffer
Projection of a video card's RAM into the machine's address space. This allows applications to access the video RAM without the chore of having to talk to the card. All high-end graphical workstations use frame buffers.

FTP
File Transfer Protocol. This is the standard Internet protocol used to transfer files from one machine to another.

full-screen
This term is used to refer to applications that take up the entire visible area of your display.

gateway
Machine or device giving a local network access to an outside network.

GFDL
The GNU Free Documentation License. The license which applies to all Mandriva Linux documentation.

GIF
Graphics Interchange Format. An image file format, widely used on the web. GIF images may be compressed or animated. Due to copyright problems it is a bad idea to use them, the recommended solution is to replace them as much as possible by the PNG format.
Ver Também: PNG.

globbing
In the shell, the ability to group a certain set of filenames with a globbing pattern.
Ver Também: globbing pattern.

globbing pattern
A string made of normal characters and special characters. Special characters are interpreted and expanded by the shell.

GNU
GNU's Not Unix. The GNU project was initiated by Richard Stallman at the beginning of the 1980s, and aimed at developing a free operating system ("free" as in "free speech"). Currently, all tools are there, except... the kernel. The GNU project kernel, Hurd, is not rock solid yet. Linux borrows, among others, two things from GNU: its C compiler, `gcc`, and its license, the GPL.
Ver Também: GPL.

GPL
General Public License. The license of the GNU/Linux kernel, it goes the opposite way to all proprietary licenses in that it applies no restrictions as to copying, modifying and redistributing the software, as long as the source code is made available. The only restriction is that the persons to whom you redistribute it must also benefit from the same rights.

GUI
Graphical User Interface. Interface to a computer consisting of windows with menus, buttons, icons and so on. A great majority of users prefer a GUI to a CLI (*Command Line Interface*) for ease of use, even though the latter is far more versatile.

guru
An expert. Used to qualify someone particularly skilled, but also of valuable help for others.

hardware address
This is a number which uniquely identifies a host in a physical network at the media access layer. Examples of this are **Ethernet Addresses** and **AX.25 Addresses**.

hidden file
Is a file which can't be "seen" when doing a `ls` command without options. Hidden files' filenames begin with a `.` and are used to store the user's personal preferences and configurations for the different programs he uses. For example, bash's command history is saved into `.bash_history`, a hidden file.

home directory
Often abbreviated as "home", this is the name for the personal directory of a given user.
Ver Também: conta.

host
Refers to a computer and is commonly used when talking about computers which are connected to a network.

HTML
HyperText Markup Language. The language used to create web documents.

HTTP
HyperText Transfer Protocol. The protocol used to connect to web sites and retrieve HTML documents or files.

icon
Is a little drawing (normally sized 16x16, 32x32, 48x48 and sometimes 64x 64 pixels) which in a graphical environment represents a document, a file or a program.

IDE
Integrated Drive Electronics. The most widely used bus on today's PCs for hard disks. An IDE bus may contain up to two devices, and the speed of the bus is limited by the device on the bus with the slowest command queue (and not the slowest transfer rate!).
Ver Também: ATAPI, SATA, S-ATA.

IMAP
Internet Message Access Protocol. A protocol which allows you to access your email messages on a remote server, without the need to transfer them locally first; as opposed to the POP mail retrieval protocol.
Ver Também: POP.

inode
Entry point leading to the contents of a file on a UNIX®-like filesystem. An inode is identified in a unique way with a number, and contains meta-information about the file it refers to, such as its access times, its type, its size, **but not its name!**

insert mode
Under Vi or any of its clones, it is the state of the program in which pressing a key will insert that character in the file being edited (except pathological cases like the completion of an abbreviation, right justify at the end of the line, ...). One gets out of it pressing the **Esc** key, (or **Ctrl-[**).

Internet
Is a huge network which connects computers around the world.

IP address
Is a numeric address consisting (in version 4, also called IPv4) of four parts which identifies your computer on a network. IP addresses are structured in a hierarchical manner, with top level and national domains, domains, sub-domains and each machine's personal address. An IP address will look something like `192.168.0.1`. A machine's personal address can be one of two types: static or dynamic. Static IP addresses are addresses which never change, they are permanently assigned. Dynamic IP addresses mean that an IP address will change with each new connection to the network. Most home users typically have dynamic IP addresses while most corporate users typically have static IP addresses.

IP masquerading
This is a technique where a firewall is used to hide your computer's true IP address from the outside. Typically, any outside network connections you make through the firewall will inherit the firewall's IP address. This is useful in situations where you may have a fast Internet connection with only one IP address but wish to use more than one computer on your internal network.

IRC
Internet Relay Chat. One of the few Internet standards for live speech. It allows for channel creation, private talks and file exchange. It also allows servers to connect to each other, which is why several IRC networks exist today: **Undernet**, **DALnet**, **EFnet** to name a few.

IRC channels
Are the "places" inside IRC servers where you can chat with other people. Channels are created in IRC servers and users join those channels so they can communicate with each other. Messages written on one channel are only visible to the people connected to that channel. Two or more users can create a "private" channel so they don't get disturbed by other users. Channel names begin with a `#`.

ISA
Industry Standard Architecture. The very first bus used on PCs, it is slowly being abandoned in favour of the PCI bus. ISA is still commonly found on SCSI cards supplied with scanners, CD writers and some other older hardware.

ISDN
Integrated Services Digital Network. A set of communication standards for voice, digital network services and video. It has been designed to eventually replace the current phone system, known as PSTN (*Public Switched Telephone Network*) or POTS (*Plain Old Telephone Service*). ISDN is known as a circuit switched data network.

ISO
International Standards Organization. A group of companies, consultants, universities and other sources which enumerates standards in various disciplines, including computing. The papers describing standards are numbered. The standard number iso9660, for example, describes the file system used on CD-ROMs.

ISO 8859
The ISO 8859 standard includes several 8-bit extensions to the ASCII character set. Especially important is ISO 8859-1, the "Latin Alphabet No. 1", which has become widely implemented and may already be seen as the *de facto* standard ASCII replacement.
ISO 8859-1 supports the following languages: Afrikaans, Basque, Catalan, Danish, Dutch, English, Faroese, Finnish, French, Galician, German, Icelandic, Irish, Italian, Norwegian, Portuguese, Scottish, Spanish, and Swedish.
Note that the ISO 8859-1 characters are also the first 256 characters of ISO 10646 (Unicode). However, it lacks the EURO symbol and does not fully cover Finnish and French. ISO 8859-15 is a modification of ISO 8859-1 to covers these needs.
Ver Também: ASCII, UTF-8.

ISP
Internet Service Provider. A company which sells Internet access to its customers, either over telephone lines or high-bandwidth circuits such as dedicated T-1 circuits, DSL or cable.

JPEG
Joint Photographic Experts Group. Another very common image file format. JPEG is mostly suited for compressing real-world scenes, and does not work very well on non-realistic images.

job
In a shell context, a job is a process running in the background. You can have several jobs running in the same shell and control each job independently.
Ver Também: foreground, segundo plano.

journaling
Journaling adds robustness to a file system, by making it transactional. Thus, instead of physically writing data at the moment it's asked for, a journal of the writes is kept, and data is written "in a block" at a later time which has also a great impact on performance and on the time needed to analyse and fix the file system, if needed.

kernel
Is the core of the operating system. The kernel is responsible for allocating resources and separating processes from each other. It handles all of the low-level operations which allow programs to talk directly to the hardware on your computer, manages the buffer cache and so on.

kill ring
Under Emacs, it is the set of text areas cut or copied since the editor was started. The text areas may be recalled to be inserted again, and the structure is ring-like.

LAN
Local Area Network. Generic name given to a network of machines connected to the same physical wiring in a reduced geographical area, such as the same office or building.
Ver Também: WAN.

launch
Is the action of invoking, or starting, a program.

library
Is a collection of procedures and functions in binary form to be used by programmers in their programs (as long as the library's license allows them to do so). The program in charge of loading shared libraries at run time is called the dynamic linker.

link
Reference to an inode in a directory, therefore giving a (file) name to the inode. Examples of inodes which don't have a link (and hence have no name) are: anonymous pipes (as used by the shell), sockets (aka network connections), network devices and so on.

linkage
The last stage of the compilation process, consisting of linking together all object files in order to produce an executable file, and matching unresolved symbols with dynamic libraries (unless a static linkage has been requested, in which case the code of these symbols will be included in the executable).

Linux
Is a UNIX®-like operating system which runs on a variety of different computers, and is free for anyone to use and modify. Linux (the kernel) was written by Linus Torvalds.

login
Connection name for a user on a UNIX® system, and the action to connect.

lookup table
Is a table which stores corresponding codes (or tags) and their meanings. It is often a data file used by a program to get further information about a particular item.
For example, HardDrake uses such a table to store a manufacturer's product codes and associated configuration information. This is one line from that table, giving information about item `CTL0001`

```
"CTL0001"      "sb"    "Creative Labs|SB16"    "sound" "HAS_OPL3|HAS_MPU401|HAS_DMA16|HAS_JOYSTICK"
```

loopback
Virtual network interface of a machine to itself, allowing the running programs not to have to take into account the special case where two network entities are in fact the same machine.

major
Number specific to the device class.

manual page
Small documents containing the definitions of a command and its usage, to be consulted with the `man` command. The first thing one should (learn how to) read when learning about a command one isn't familiar with.

MBR
Master Boot Record. Name given to the first sector of a bootable hard drive. The MBR contains the code used to load the operating system into memory or a bootloader (such as LILO), and the partition table of that hard drive.

MIME
Multipurpose Internet Mail Extensions. A string of the form `type/subtype` describing the contents of a file attached in an e-mail. This allows MIME-aware mail clients to define actions depending on the type of the file.

minor
Number identifying the specific device we are talking about.

MPEG
Moving Picture Experts Group. An ISO committee which generates standards for video and audio compression. MPEG is also the name of their algorithms. Unfortunately, the license for this format is very restrictive, and as a consequence there are still no Open Source MPEG players...

mount point
Is the place or directory where a partition or another device is attached to the GNU/Linux filesystem. For example, your CD-ROM is mounted in the `/mnt/cdrom` directory, from where you can explore the contents of any mounted CDs.

mounted
A device is mounted when it is attached to the GNU/Linux filesystem. When you mount a device you can browse its contents. This term is partly obsolete due to the "supermount" feature, so users do not need to manually mount removable media.
Ver Também: mount point.

MSS
The *Maximum Segment Size* is the largest quantity of data which can be transmitted at one time across an interface. If you want to prevent local fragmentation MSS would equal the MTU IP header.

MTU
The *Maximum Transmission Unit* is a parameter which determines the size of the largest datagram which can be transmitted by an IP interface without it needing to be broken down into smaller units. The MTU should be larger than the largest datagram you wish to transmit un-fragmented. Note, this only prevents fragmentation locally, some other link in the path may have a smaller MTU and the datagram will be fragmented there. Typical values are 1500 bytes for an Ethernet interface, or 576 bytes for a PPP interface.

multitasking
The ability of an operating system to share CPU time between several processes. At a low level, this is also known as multiprogramming. Switching from one process to another requires that all the current process context be saved and restored when this process runs again. This operation is called a context switch, and is done several times per second, thereby making it fast enough so that a user has the illusion that the operating system runs several applications at the same time. There are two types of multitasking: in preemptive multitasking the operating system is responsible for taking away the CPU and passing it to another process; cooperative multitasking is where the process itself gives back the CPU. The first variant, used by GNU/Linux, is obviously the better choice because no program can consume the entire CPU time and block other processes. The policy to select which process should be run, depending on several parameters, is called scheduling.

multiuser
Is used to describe an operating system which allows multiple users to log into and use the system at the exact same time, each user being able to do their own work independent of other users. A multitasking operating system is required to provide multiuser support. GNU/Linux is both a multitasking and multiuser operating system, as is any UNIX® system for that matter.

named pipe
A UNIX® pipe which is linked, as opposed to pipes used in shells.
Ver Também: pipe, link.

naming
A word commonly used in computing for a method to identify objects. You will often hear of "naming conventions" for files, functions in a program and so on.

NCP
NetWare Core Protocol. A protocol defined by **Novell** to access Novell NetWare file and print services.

NFS
Network File System. A network file system created by **Sun Microsystems** in order to share files across a network in a transparent way.

newsgroups
Discussion and news areas which can be accessed by a news or USENET client to read and write messages specific to the topic of the newsgroup. For example, the newsgroup `alt.os.linux.mandrake` is an alternate newsgroup (alt) dealing with the Operating System (OS) GNU/Linux (linux), and specifically, Mandriva Linux (mandrake). Newsgroups are broken down in this fashion to make it easier to search for a particular topic.

NIC
Network Interface Controller. An adapter installed in a computer which provides a physical connection to a network, such as an Ethernet card.

NIS
Network Information System. NIS was also known as "Yellow Pages", but **British Telecom** holds a copyright on this name. NIS is a protocol designed by **Sun Microsystems** in order to share common information across a NIS **domain**, which may consist of an entire LAN, or just a part of it. It can export password databases, service databases, groups information and more.

null, character
The character or byte number 0. It is used to mark the end of a string.

object code
Is the code generated by the compilation process to be linked with other object codes and libraries to form an executable file. Object code is machine readable.
Ver Também: compilação, linkage.

on the fly
Something is said to be done "on the fly" when it's done along with something else, without you noticing it or explicitly asking for it.

open source
Is the name given to free source code of a program that is made available to the development community and public at large. The theory behind this is that allowing source code to be used and modified by a broader group of programmers will ultimately produce a more useful product for everyone. Some popular open source programs include Apache, sendmail and GNU/Linux.

operating system
Is the interface between the applications and the underlying hardware. The tasks for any operating system are primarily to manage all of the machine specific resources. On a GNU/Linux system, this is done by the kernel and loadable modules. Other well-known operating systems include Amiga®OS, Mac OS®, FreeBSD®, OS/2®, UNIX®, and Windows® in all its variants.

owner
In the context of users and their files, the owner of a file is the user who created that file.

owner group
In the context of groups and their files, the owner group of a file is the group to which the user who created that file belongs.

PAP
Password Authentication Protocol. A protocol used by many ISPs to authenticate their clients. In this scheme, the client (you) sends an identifier/password pair to the server, but none of the information is encrypted. CHAP is a more secure, and thus preferred, authentication protocol.
Ver Também: CHAP.

pager
A program which displays a text file one screen at a time, making it easy to move back and forth and search for strings in this file. We suggest you to use `less`.

password
Is a secret word or combination of words or letters which is used to secure something. Passwords are used in conjunction with user logins to multi-user operating systems, web sites, FTP sites, and so forth. Passwords should be hard-to-guess phrases or alphanumeric combinations, and should never be based on common dictionary words. Passwords ensure that other people cannot log into a computer or site with your account.

patch, to patch
A file containing a list of corrections to issue to source code in order to add new features, to remove bugs, or to modify it according to one's wishes and needs. The action consisting of the application of these corrections to the archive of source code (aka "patching").

path
Is an assignment for files and directories to the filesystem. The different layers of a path are separated by the "slash" or '/' character. There are two types of paths on GNU/Linux systems. The **relative** path is the position of a file or directory in relation to the current directory. The **absolute** (or **full**) path is the position of a file or directory in relation to the root directory.

PCI
Peripheral Component Interconnect. A bus created by **Intel** which today is the standard bus for PC and other architectures. It is the successor to ISA, and it offers numerous services: device identification, configuration information, IRQ sharing, bus mastering and more.

PCMCIA
Personal Computer Memory Card International Association. More and more commonly called "PC Card" for simplicity reasons, this is the standard for external cards attached to a laptop: modems, hard disks, memory cards, Ethernet cards, and more. The acronym is sometimes humorously expanded to *People Cannot Memorize Computer Industry Acronyms*...

pipe
A special UNIX® file type. One program writes data into the pipe, and another program reads the data from the other end. UNIX® pipes are FIFOs, so the data is read at the other end in the order it was sent. Very widely used with the shell. See also **named pipe**.

pixmap
Is an acronym for "pixel map". It's another way of referring to bitmap images.

plugin
Add-on program used to display or play some multimedia content found on a web document. It can usually be easily downloaded if your browser is not yet able to display or play that kind of information.

PNG
Portable Network Graphics. Image file format created mainly for web use, it has been designed as a patent-free replacement for GIF and also has some additional features.

PnP
Plug'N'Play. First an add-on for ISA in order to add configuration information for devices, it has become a more widespread term which groups all devices able to report their configuration parameters. All PCI devices are Plug'N'Play.

POP
Post Office Protocol. One common protocol used for retrieving mail from an ISP. See IMAP for an example of another remote-access mail protocol.
Ver Também: IMAP.

porting
One of two ways to run a program on a system it was not originally intended for. For example, to be able to run a Windows®-native program under GNU/Linux (natively), it must first be ported to GNU/Linux.

PPP
Point to Point Protocol. This is the protocol used to send data over serial lines. It is commonly used to send IP packets to the Internet, but it can also be used with other protocols such as Novell's IPX protocol.

precedence
Dictates the order of evaluation of operands in an expression. For example: If you have `4 + 3 * 2` you get 10 as the result, since the multiplication has higher precedence than the addition. If you want to evaluate the addition first, then you have to add parenthesis like this: `(4 + 3) * 2`. When you do this, you'll get 14 as the result since the parenthesis have higher precedence than the addition and the multiplication, so the operations in parenthesis get evaluated first.

preprocessors
Are compilation directives which instruct the compiler to replace those directives for code in the programming language used in the source file. Examples of C 's preprocessors are `#include`, `#define`, etc.

process
In the operating system context, a process is an instance of a program being executed along with its environment.

prompt
In a shell, this is the string before the cursor. When you see it, you can type your commands.

protocol
Protocols organise the communications between different machines across a network, either using hardware or software. They define the format of transferred data, whether one machine controls another, etc. Many well-known protocols include HTTP, FTP, TCP, and UDP.

proxy
A machine which sits between a network and the Internet, whose role is to speed up data transfers for the most widely used protocols (for example, HTTP and FTP). It maintains a cache of previous requests, so that a machine which makes a request for something which is already cached will receive it quickly, because it will get the information from the local cache. Proxies are very useful on low bandwidth networks (such as modem connections). Sometimes the proxy is the only machine able to access outside the network.

pull-down menu
Is a menu that is "rolled" with a button in some of its corners. When you press that button, the menu "unrolls" itself, showing you the full menu.

quota
Is a method of restricting disk usage and put limits for users. Administrators can restrict the size of home directories for a user by setting quota limits on specific file systems.

RAID
Redundant Array of Independent Disks. A project initiated at the computing science department of Berkeley University, in which the storage of data is spread across an array of disks using different schemes. At first, this was implemented using low-cost, older, drives, which is why the acronym originally stood for *Redundant Array of Inexpensive Disks*.

RAM
Random Access Memory. Term used to identify a computer's main memory. The "Random" here means that any part of the memory may be directly accessed.

read-only mode
For a file means that the file cannot be written to. You can read its contents but you can't modify them.
Ver Também: read-write mode.

read-write mode
For a file, it means that the file can be written to. You can read its contents and modify them.
Ver Também: read-only mode.

regular expression
A powerful theoretical tool which is used to search and match text strings. It lets one specify patterns these strings must obey. Many UNIX® utilities use it: `sed`, awk, grep, perl and others.

RFC
Request For Comments. RFCs are the official Internet standard documents, published by the IETF (*Internet Engineering Task Force*). They describe all protocols, their usage, their requirements and so on. When you want to learn how a protocol works, pick up the corresponding RFC.

root
Is the superuser of any UNIX® system. Typically root (aka the system administrator) is the person responsible for maintaining and supervising the UNIX® system. This person also has complete access to everything on the system.

root directory
This is the top level directory of a filesystem. This directory has no parent directory, thus '..' for root points back to itself. The root directory is written as '/'.

root filesystem
This is the top level filesystem. This is the filesystem where GNU/Linux mounts its root directory tree. It is necessary for the root filesystem to reside in a partition of its own, as it is the basis for the whole system. It contains the root directory.

route
Is the path which your datagrams take through the network to reach their destination. It is the path between one machine and another in a network.

RPM
RPM Package Manager. A packaging format developed by **Red Hat** in order to create software packages, it is used in many GNU/Linux distributions, including Mandriva Linux.

run level
Is a configuration of the system software which only allows certain selected processes to exist. Allowed processes are defined, for each runlevel, in the file `/etc/inittab`. Usually, there are seven defined runlevels: 0, 1, 2, 3, 4, 5, 6 and switching between them can only be achieved by a privileged user by means of executing the commands `init` and `telinit`.

SATA, S-ATA
Serial ATA. The successor to the ATA specification. First generation SATA has a bandwidth of 1.5Gbps, but the serial link and underlying technologies allow for much greater bandwidths, while parallel ATA has reached its practical limits with UDMA133.
Ver Também: ATAPI, IDE.

script
shell scripts are sequences of commands to be executed as if they were sequentially entered in the console. shell scripts are UNIX®'s (somewhat) equivalent of DOS batch files.

SCSI
Small Computers System Interface. A bus with a high throughput designed to allow for several types of peripherals to be connected to it. Unlike IDE, a SCSI bus is not limited by the speed at which the peripherals accept commands. Only high-end machines integrate a SCSI bus directly on the motherboard, therefore most PCs need add-on cards.

security levels
Mandriva Linux's unique feature which allows you to set different levels of restriction according to how secure you want to make your system. There are 6 predefined levels ranging from 0 to 5, where 5 is the tightest security. You can also define your own security level.

segmentation fault
A segmentation fault occurs when a program tries to access memory that is not allocated to it. This generally causes the program to stop immediately.

server
A program or computer which provides a feature or service and awaits the connections from **clients** to execute their orders or give them the information they ask for. In the case of **peer to peer** systems such as SLIP or PPP, the server is taken to be the end of the link that is called and the end calling is taken to be the client. It is one of the components of a **client/ server system**.
Ver Também: cliente, sistema cliente/servidor.

shadow passwords
A password management suite on UNIX® systems in which the file containing the encrypted passwords is not world-readable, unlike that usually found with a normal password system. It also offers other features such as password aging.

shell
The shell is the basic interface to the operating system kernel and provides the command line where users enter commands to run programs and system commands. All shells provide a scripting language which can be used to automate tasks or simplify often-used complex tasks. These shell scripts are similar to batch files from the DOS operating system, but are much more powerful. Some example shells are bash, sh, and tcsh.

single user
Is used to describe a state of an operating system, or even an operating system itself, which only allows a single user to log into and use the system at any time.

site dependent
Means that the information used by programs such as imake and `make` to compile some source file depends on the site, the computer architecture, the computer's installed libraries, and so on.

SMB
Server Message Block. Protocol used by Windows® machines for file and printer sharing across a network.
Ver Também: CIFS.

SMTP
Simple Mail Transfer Protocol. This is the common protocol for transferring email. Mail Transfer Agents such as sendmail or postfix use SMTP. They are sometimes called SMTP servers.

socket
File type corresponding to any network connection.

soft links
Ver: symbolic links

standard error
The file descriptor number 2, opened by every process, used by convention as the file descriptor to which the process writes errors. It is usually the computer's screen.
Ver Também: standard input, standard output.

standard input
The file descriptor number 0, opened by every process, used by convention as the file descriptor from which the process receives data. It is usually the computer's keyboard.
Ver Também: standard error, standard output.

standard output
The file descriptor number 1, opened by every process, used by convention as the file descriptor in which the process prints its output. It is usually the computer's screen.
Ver Também: standard error, standard input.

streamer
Is a device which takes "streams" (not interrupted or divided in shorter chunks) of characters as its input. A typical streamer is a tape drive.

SVGA
Super Video Graphics Array. The video display standard defined by VESA for the PC architecture. The resolution was at first 800x 600 x 16 colours, quickly extended to 1024x768 x 16 colours, and beyond.

switch
Switches are used to change the behaviour of programs, and are also called command-line options or arguments. To determine if a program has optional switches which may be used, read the `man` pages or try to pass the `--help` switch to the program (i.e.. `program --help`).

symbolic links
Are special files, containing nothing but a string which references another file. Any access to them is the same as accessing the file whose name is the referenced string, which may or may not exist, and the path to which can be given in a relative or an absolute way.

target
Is the object of compilation, i.e. the binary file to be generated by the compiler.

TCP
Transmission Control Protocol. This is the most common reliable protocol which uses IP to transfer network packets. TCP adds the necessary checks on top of IP to make sure that packets are delivered. Unlike UDP, TCP works in connected mode, which means that two machines must establish a connection before exchanging data.

telnet
Creates a connection to a remote host and allows you to log into the machine, provided you have an account. Telnet is the most widely-used method of remote logins, however there are better and more secure alternatives, such as `ssh`.

theme-able
A graphical application is theme-able if it is able to change its appearance in real time. Many window managers are theme-able.

TLDP
The Linux Documentation Project. A nonprofit organisation which maintains GNU/Linux documentation. It's mostly known for documents such as HOWTOs, but it also maintains FAQs, and even a few books.
Ver Também: FAQ.

traverse
For a directory on a UNIX® system, this means that the user is allowed to go through this directory, and possibly to directories under it. This requires that the user has execute permission on this directory.

URL
Uniform Resource Locator. A string with a special format used to identify a resource on the Internet in a unique way. The resource may be a file, a server or other item. The syntax for a URL is `protocol://user:password@server.name[:port]/path/to/resource`.
When only a machine name is given and the protocol is `http://`, it defaults to retrieving the file that the server is configured to show by default, usually it the `index.html` file.

username
Is a name (or more generally a word) which identifies a user on a system. Each username is attached to a unique and single UID (user ID)
Ver Também: login.

UTF-8
Unicode Transformation Format 8. It is an octet (8-bit) lossless encoding of Unicode characters. UTF-8 encodes each Unicode character as a variable number of 1 to 4 octets, where the number of octets depends on the integer value assigned to the Unicode character. It is an efficient encoding of Unicode documents that use mostly US-ASCII characters because it represents each character in the range U+0000 through U+007F as a single octet. UTF-8 is the default encoding for XML.
Ver Também: ISO 8859, ASCII.

variables
Are strings which are used in `Makefile` files to be replaced by their value each time they appear. Usually they are set at the beginning of the `Makefile`. They are used to simplify `Makefile` and source files tree management.
More generally, variables in programming are words that refer to other entities (numbers, strings, tables, etc.) that are likely to vary while the program is executing.

verbose
For commands, the verbose mode means that the command reports to standard (or possibly error) output all the actions it performs and the results of those actions. Sometimes, commands have a way to define the "verbosity level", which means that the amount of information that the command will report can be controlled.

VESA
Video Electronics Standards Association. An industry standards association aimed at the PC architecture. For example, it is the author of the SVGA standard.

virtual console
Is the name given to what used to be called terminals. On GNU/Linux systems, you have what are called virtual consoles which enable you to use one screen or monitor for many independently running sessions. By default, you have six virtual consoles which can be reached by pressing **ALT-F1** through **ALT-F6**. There is a seventh virtual console, **ALT-F7**, which will permit you to reach a running X Window System. In X, you can reach the text console by pressing **CTRL-ALT-F1** through **CTRL-ALT-F6**.
Ver Também: console.

virtual desktops
In the X Window System, the window manager may provide you with several desktops. This handy feature allows you to organise your windows, avoiding the problem of having dozens of them stacked on top of each other. It works as if you had several screens. You can switch from one virtual desktop to another in a manner which depends on the window manager you're using.
Ver Também: window manager, desktop.

WAN
Wide Area Network. This network, although similar to a LAN, connects computers on networks which are not physically connected to the same wiring and are separated by a greater distance.
Ver Também: LAN.

wildcard
The '*' and '?' characters are used as wildcard characters and may represent anything. The '*' represents any number of characters, including no characters. The '?' represents exactly one character. Wildcards are often used in regular expressions.

window
In networking, the **window** is the largest amount of data that the receiving end can accept at a given point in time.
In the context of a graphical user environment, a window is the rectangle that occupies a given running application which usually contains a title, a menu, a status bar, and the application's work area.

window manager
The program responsible for the "look and feel" of a graphical environment, dealing with window bars, frames, buttons, root menus, and some keyboard shortcuts. Without it, it would be hard or impossible to have virtual desktops, to resize windows on the fly, to move them around, ...

workspace switcher
A little applet which allows you to switch between the available virtual desktops. It is also known as pager.
Ver Também: virtual desktops.

Índice Remissivo

>Estampa de Tempo
  ctime, 47
  mtime, 47
.bashrc, 48
acesso remoto, 85
  automação, 85
ambiente
  processo, 36
  variáveis, 12
aplicações
  ImageMagick, 53
  terminais, 53
arquivo
  apagando, 48
  atributo, 32
  atributos, 49
  bloco, 28
  caractere, 27
  criando, 47
  cópia, 49
  de bloco, 31
  de caractere, 31
  encontrar, 70
  link, 28, 28
  movendo, 48
  renomeando, 48
  socket, 28
atributos
  arquivo, 49
Borges, **??**
caracteres
  casamento, 51
caracteress
  especiais, 54
casamento
  caractere, 51
Ciclano Pingus, 5
comando
  cat, 13
  less, 14
  ls, 14
  pwd, 12
comandos
  at, 73
  bzip2, 75
  cd, 12
  chgrp, 50
  chmod, 50
  chown, 49
  cp, 49
  crontab, 72
  find, 70
  grep, 66
  gzip, 75
  init, 81
  kill, killall, 78
  less, 52
  mkdir, 47
  mount, 43
  mv, 48
  ps, 77
  rm, 48
  rmdir, 48
  scp, 86
  sed, 52
  ssh, 85
  ssh-add, 86
  ssh-keygen, 85
  tar, 74
  touch, 47
  umount, 43
  urpmi, 87
  wc, 52
console, 8
conta, 7
desenvolvimento, 2
diretório
  apagando, 48
  criando, 47
  cópia, 49
  movendo, 48
  renomeando, 48
discos, 17
  SCSI, 19
DocBook, **??**
documentação
  Mandriva Linux, 3
dono, 49
  mudança, 49
editores de texto
  Emacs, 57
  vi, 60
empacotamento, 2
Estampa de Tempo
  atime, 47
FHS, 21
Fulano Pingus, 5
GID, 8
grupo, 7
  mudança, 50
IDE
  dispositivos, 19
inode, 28
  tabela, 28
internacionalização, 2
linha de comando
  completar, 53
  introdução, 47
  utilitários, 65
link
  hard, 32
  simbólico, 31
Mandriva Expert, 1
Mandriva Club, 1
Mandriva Linux
  listas de discussão, 1
  segurança, 1
Mandriva Store, 2

memória RAM, 18
modules, 38
ordem de combinação, 51
pacotes
  gerenciamento, 87
padrão
  entrada, 52
  erro, 52
  saída, 52
partição
  home, 18
  usr, 18
partição de
  swap, 18
partições, 17, 41
  extendidas, 19
  lógicas, 19
  primárias, 19
permissões, 50
PID, 11
pipe, 53
  anônimo, 29
  arquivo, 28
  nomeado, 29
primary
  slave, 19
primário
  master, 19
processo, 11, 35, 54
processos, 77
programação, 2
prompt, 8, 11
raiz
  diretório, 21, 36
redirecionamento, 52
root
  partição, 18
runlevel, 81
senha, 8
setor, 17
shell, 11, 47
  casamento de padrões, 51
sinopse
  comando, 5
Soundblaster, 19
ssh
  chave, 85
  cliente, 85
  servidr, 85
swap, 17
Tamanho de
  Swap, 18
udev, 20
UID, 8
UNIX®, 7
usuário, 7
  root, 8
usuários
  genérico, 5
utilitários
  tratamento de arquivos, 47
valores
  discretos, 51
vírus, 11