Guia do Administrador de Sistemas Linux

Do original
Linux System Administrator's Guide

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

```
Wirzenius, Lars
   Guia do Administrador de Sistemas Linux / Lars Wirzenius;
   tradução de Conectiva Informática.
   São Paulo : Conectiva, 1998.

   Título original: Linux System Administrator's Guide.
   Bibliografia.

   1. LINUX (Sistema operacional de computador)
2. Redes de computadores
3. UNIX (Sistema operacional de computador)
I. Título

98-4859                         CDD-005.43
ISBN:  85-87118-02-1
```

Índices para catálogo sistemático:
1. LINUX : Sistema operacional : Computadores :
Processamento de dados 005.43

Conectiva Informática
Rua Rubens Elke Braga, 558
CEP: 80.220.320, Parolin - Curitiba - Paraná
Telefone/Fax: (041) 332-2074

Editoração, Diagramação, Supervisão, Revisão: Conectiva Informática

Tradução/Revisão do Guia do Administrador de Sistemas Linux:
Rodrigo Stulzer Lopes <rodrigo@conectiva.com.br>
Sandro Nunes Henrique <sandro@conectiva.com.br>

Tradução/Revisão dos HOWTO's:
Lisiane Sztoltz <lisiane@conectiva.com.br>
Paulo Renato Silva de Rezende <paulo@conectiva.com.br>
Rodrigo Stulzer Lopes <rodrigo@conectiva.com.br>
Sandro Nunes Henrique <sandro@conectiva.com.br>

Tradução/Revisão das páginas de manual:
Eliphas Levy da Silva Theodoro Queiroz <eliphas@conectiva.com.br>
Jorge Godoy <jorge@bestway.com.br>
Lisiane Sztoltz <lisiane@conectiva.com.br>
Marcus Brito <pazu@linuxbr.com.br>
Paulo Renato Silva de Rezende <paulo@conectiva.com.br>
Rafael Caetano dos Santos <rcaetano@linux.ime.usp.br>
Rodney Wagner Miyakawa <rodney@conectiva.com.br>
Rodrigo Stulzer Lopes <rodrigo@conectiva.com.br>
Sandro Nunes Henrique <sandro@conectiva.com.br>

Guia do Administrador de Sistemas Linux

Lars Wirzenius

Projeto de Documentação do Linux

Esta é a versão 0.6.2 do Guia do Administrador de Sistemas Linux.
Impresso em 18 de novembro de 1999.

Traduzido e Adaptado pela Conectiva.

Os fontes do documento original em LaTeX e outros formatos podem ser encontrados na Internet, via ftp anônimo em `http://metalab.unc.edu`, no diretório `/pub/Linux/docs/LDP`. Encontra-se também disponível nos formatos Postscript e TeX .dvi. A página oficial deste livro encontra-se em `http://www.iki.fi/liw/linux/sag/`. A versão atualizada também pode ser encontrada naquela localização.

O autor gostaria de ser avisado das modificações, traduções e versões impressas. Obrigado.

A Conectiva, no seu papel de buscar popularizar o uso do Linux no Brasil, buscou traduzir da melhor forma o conteúdo desta publicação do LDP - Projeto de Documentação do Linux, assim como agregou comentários e informações adicionais que pudessem enriquecer o seu uso.

Quaisquer incorreções ou problemas detectados, pedimos que sejam comunicados por email para `info@conectiva.com.br`, a fim de que os ajustes sejam realizados para as próximas versões.

Agradecemos a todos aqueles que têm participado ativamente no desenvolvimento dos trabalhos de tradução, internacionalização, divulgação e adaptação do Linux à realidade brasileira, pois muito de nosso esforço está calcado no processo participativo desta comunidade.

Esperamos que este livro seja de utilidade aos administradores de sistemas que busquem uma ferramenta de auxílio às suas atividades diárias, e que possa enriquecer e facilitar os seus conhecimentos.

Linux: Quem Compara, Usa!

Conectiva 1998

Este livro é dedicado a todos aqueles que lutam pelo desenvolvimento do Linux no Brasil

Breve Sumário

Sumário

Capítulo 1

Introdução

No começo o arquivo era vazio e não tinha forma; e o vazio estava sobre a face dos bits. E os Dedos do do Autor moveram-se sobre o teclado. E o Autor disse: que haja palavras, e fizeram-se as palavras.

Este manual descreve aspectos da administração de sistemas do Linux. Ele está voltado para usuários com pouco conhecimento de administração de sistemas, mas que tenham um conhecimento mínimo. Este manual não descreve como instalar o Linux, o qual é apresentado no Guia de Instalação e Introdução ao Linux. Veja a seguir maiores informações sobre os manuais do Linux.

Administração de sistemas é o conjunto de tarefas necessárias para manter o computador em boas condições de uso. Isso inclui atividades como efetuar cópias de segurança (e restaurá-las, se necessário), instalar novos programas, criar contas de usuários (e apagá-las quando não mais forem necessárias), garantir que os sistemas de arquivos estejam íntegros, e assim por diante. Se um computador fosse, por exemplo, uma casa, a administração de sistemas poderia ser comparada à manutenção, a qual inclui a limpeza, o conserto de vidraças quebradas e outras tarefas do gênero. A administração de sistemas não é chamada de manutenção, porque encerra outros aspectos, não abrangidos por este conceito, que é muito simples. [1]

A estrutura deste manual permite que muitos capítulos possam ser utilizados de maneira independente, isto é, caso sejam necessárias por exemplo informações sobre cópias de segurança, pode-se ler somente o capítulo 10. Espera-se com isto tornar este livro um manual de referência e treinamento, e dentro do possível, que somente partes pequenas sejam consultadas quando necessário, ao invés de exigir a sua leitura completa. Ainda assim o fato deste manual ser uma ferramenta de ajuda e de referência é, para mim, uma feliz coincidência.

[1] Existem algumas pessoas que chamam administração de manutenção, mas isso é porque elas nunca leram este manual. Pobres criaturas.

Este manual não pretende ser usado como única ferramenta de consulta. Muitas informações importantes para a administração de sistemas podem ser encontradas na documentação do Linux. Além disso o administrador do sistema é somente um usuário com privilégios e tarefas especiais. Um recurso muito importante são as páginas de manual on-line, que devem servir de referência sempre que um comando não for familiar.

O processo de tradução e adaptação do manual buscou acrescentar os comandos e HOWTOS[2] relacionados aos temas aqui abordados, os quais estão relacionados a partir do Apêndice C.

Enquanto este manual está voltado para o Linux, princípios gerais de administração podem ser utilizados em outros sistemas UNIX. Infelizmente pelo fato das variações serem grandes entre diferentes versões do UNIX, a possibilidade de cobrir completamente o tema é muito pequena. O próprio Linux é difícil de ser abrangido em todas as suas possibilidades, dada a natureza de seu desenvolvimento.

Não há uma distribuição oficial do Linux, diferentes pessoas utilizam diferentes configurações, e muitas pessoas têm configurações personalizadas. Este livro não pretende fixar-se em uma distribuição específica, apesar do autor utilizar o Debian GNU/Linux e o tradutor o Conectiva Linux. Sempre que possível procurar-se-á apresentar as diferenças entre elas.

Tentou-se descrever como as coisas funcionam, ao invés de se descrever "cinco passos simples" para cada tarefa. Isso significa que muitas informações aqui podem não ser necessárias para todos os leitores. Estas partes podem ser simplesmente ignoradas para quem utiliza um sistema pré-configurado. Uma leitura completa, naturalmente, aprimorará o conhecimento do sistema e poderá tornar a administração mais agradável.

Como todo o desenvolvimento do Linux, que funciona através do trabalho voluntário, este trabalho foi realizado porque achei que seria agradável e porque eu achei que ele deveria ser realizado. Ainda assim, como em qualquer trabalho voluntário há limites de tempo despendido e de conhecimento. Isso significa que este manual pode não ser necessariamente tão bom quanto ele poderia se um especialista fosse pago para escrevê-lo e ainda gasto alguns anos para deixá-lo perfeito. Eu penso, é óbvio, que ele seja muito bom, mas esteja alerta!

Outro ponto que deve ser ressaltado é que muitas coisas já documentadas em outros manuais não foram aqui exploradas profundamente. Isso aplica-se especialmente à documentação de programas específicos, como por exemplo o uso detalhado do `mkfs` [3]. Eu descrevi somente o propósito do programa e o seu uso como ferramenta dentro dos princípios deste manual. Para maiores informações, eu sugiro ao caro leitor

[2] Como Fazer

[3] na edição brasileira adaptada as páginas de manual mais importantes são apresentados no Apêndice E

o uso de outros manuais. Normalmente estes manuais fazem parte do Projeto de Documentação do Linux.

Mesmo que eu tenha tentado fazer deste manual o melhor possível, gostaria de ouvir qualquer sugestão para que possa melhorá-lo. Erros de linguagem, informações incorretas, idéias para novos tópicos, seções reescritas, informações sobre como os diversos sistemas UNIX funcionam, são bem-vindos. Você pode me contatar através da World Wide Web em `http://www.iki.fi/liw/mail-to-lasu.html`. É necessário ler esta página para não ser pego pelos meus filtros de email. Para sugestões em português, por favor envie as mensagens para `info@conectiva.com.br`.

Muitas pessoas ajudaram no desenvolvimento deste livro, direta ou indiretamente. Eu gostaria de agradecer especialmente a Matt Welsh pela inspiração e liderança no LDP, Andy Oram por levar-me a trabalhar com o seu valioso retorno, Olaf Kirch por mostrar que isso poderia ser executado, e Adam Richter da Yggdrasil e diversas outras pessoas por me mostrarem que este trabalho poderia ser útil e interessante.

Stephen Tweedie, H. Peter Anvin, Rémy Card, Theodore Ts'o, e Stephen Tweedie que me emprestaram seus trabalhos [4] (e tornaram este livro menor e mais rico). Estou muito agradecido e peço desculpas por versões anteriores que possam estar sem os devidos créditos.

Adicionalmente gostaria de agradecer a Mark Komarinski por enviar seu material em 1993 e às muitas colunas de administração de sistemas do Linux Journal. Elas foram muito instrutivas e inspiradoras.

Muitos comentários úteis foram enviados por um grande número de pessoas. Não poderia enumerar todos eles, mas alguns, em ordem alfabética devem ser lembrados: Paul Caprioli, Ales Cepek, Marie-France Declerfayt, Dave Dobson, Olaf Flebbe, Helmut Geyer, Larry Greenfield e seu pai, Stephen Harris, Jyrki Havia, Jim Haynes, York Lam, Timothy Andrew Lister, Jim Lynch, Michael J. Micek, Jacob Navia, Dan Poirier, Daniel Quinlan, Jouni K Seppänen, Philippe Steindl, G.B. Stotte. Peço desculpas àqueles que por ventura não tenham sido citados.

Convenções tipográficas

Negrito Usado para evidenciar **novos conceitos**, **ALERTAS**, e **palavras-chaves** em uma linguagem.

itálico Usado para *enfatizar* um texto, e ocasionalmente para frases ou introduções no início de uma seção.

oblíquo Usado para evidenciar **meta-variáveis** no texto, especialmente na re-

[4] uma comparação entre os sistemas de arquivos xia e ext2, a lista de dispositivos e uma descrição do sistema de arquivos ext2. Estes não mais fazem parte do livro.

presentação da linha de comando. Por exemplo,

`ls -l` *foo*

onde *foo* "representará" um nome de arquivo, como em `/bin/cp`.

`Constante` Usado para representar interação com o terminal, como em

```
$ ls -l /bin/cp
-rwxr-xr-x   1 root     root        23124 ago 20 00:03 /bin/cp
```

Também usado para exemplos de código, podendo ser um fonte C, shell script, ou qualquer outro. Para mostrar arquivos em geral, como arquivos de configuração. Quando necessário, estes exemplos ou figuras poderão estar envoltos numa fina caixa de texto.

`Tecla` Representa uma tecla a pressionar. Você normalmente verá esta situação da seguinte forma:

Pressione `return` para continuar.

O Projeto de Documentação do Linux (LDP)

O Projeto de Documentação do Linux, ou LDP, é uma equipe de escritores, revisores e editores que trabalham juntos para produzir uma documentação completa do sistema operacional Linux. O coordenador geral do projeto é Greg Hankins.

Este é um manual de um conjunto distribuído pelo LDP, incluindo Guia do Usuário Linux, Guia do Administrador de Sistemas, Guia do Administrador de Redes e o Guia dos Hackers para o Kernel. Estes manuais estão disponíveis em fontes LaTeX, .dvi e postscript via FTP anônimo através do `sunsite.unc.edu`, no diretório `/pub/Linux/docs/LDP`.

Nós encorajamos qualquer um com pendor para escrita ou editoração a juntar-se a nós para melhorar a documentação do Linux. Caso você tenha um endereço email, pode contactar Greg Hankins em `gregh@sunsite.unc.edu`.[5]

[5] ou a Conectiva em `info@conectiva.com.br`

Os Versos do LDP[6]

A wondrous thing,
and beautiful,
'tis to write,
a book.

I'd like to sing,
of the sweat,
the blood and tear,
which it also took.

It started back in,
nineteen-ninety-two,
when users whined,
"we can nothing do!"

They wanted to know,
what their problem was,
and how to fix it
(by yesterday).

We put the answers in,
a Linux f-a-q,
hoped to get away,
from any more writin'.

"That's too long,
it's hard to search,
and we don't read it,
any-which-way!"

Then a few of us,
joined together
(virtually, you know),
to start the LDP.

We started to write,
or plan, at least,
several books,
one for every need.

The start was fun,
a lot of talk,
an outline,
then a slew.

Then silence came,
the work began,
some wrote less,
others more.

A blank screen,
oh its horrible,
it sits there,
laughs in the face.

We still await,
the final day,
when everything,
will be done.

Until then,
all we have,
is a draft,
for you to comment on.

[6] O autor prefere manter-se anônimo. Eles foram enviados à lista de discussão do LDP por Matt Welsh. O texto foi mantido em inglês para manter as rimas dos versos

Capítulo 2

Visão Geral de um Sistema Linux

Deus olhou para tudo o que fez,
e viu que estava muito bom.
Genesis 1:31

Este capítulo apresenta uma visão geral do Linux. Primeiro, os principais serviços disponibilizados pelo sistema operacional são descritos. Na seqüência, os programas que implementam estes serviços são descritos porém sem um detalhamento muito profundo. O propósito deste capítulo é visualizar o sistema como um todo, pois cada parte será descrita em detalhes mais adiante.

2.1 Diversas partes de um sistema operacional

Um sistema operacional UNIX consiste de um **núcleo (kernel)** e alguns **programas do sistema**. Há ainda **aplicações** que executam diversas tarefas. O kernel é o coração do sistema operacional[1]. Ele mantém o controle dos arquivos em disco, inicializa programas e executa-os de forma concorrente, aloca memória e outros recursos para os diversos processos, recebe e envia pacotes da rede, e assim por diante. O kernel faz muito pouco sozinho, na verdade ele provê ferramentas com as quais os serviços podem ser desenvolvidos. Ele ainda evita que qualquer um possa acessar diretamente os componentes de hardware, forçando o uso das ferramentas disponíveis. Deste modo, o kernel mantém uma certa proteção entre os usuários. As ferramentas disponibilizadas pelo kernel são utilizadas através das **chamadas ao sistema (system calls)**. Para maiores informações veja o Apêndice E.

Os programas de sistema utilizam as ferramentas disponibilizadas pelo kernel para implementar os diversos serviços necessários ao sistema operacional. Programas de

[1]Na verdade ele é erroneamente considerado como o próprio sistema operacional. Um sistema operacional disponibiliza muito mais serviços que o kernel somente.

sistema e todos os demais programas, são executados 'sobre o kernel', o qual é denominado **modo usuário**[2]. A diferença entre programas de sistema e aplicações está no seu objetivo: as últimas visam que coisas úteis sejam realizadas (brincar, caso seja um jogo), enquanto os primeiros são necessários para manter o sistema operacional funcionando. Um editor de textos é uma aplicação, o `telnet` é um programa de sistema. A diferença é algumas vezes muito pequena, e somente é importante para classificadores compulsivos.

Um sistema operacional pode ainda conter compiladores e suas bibliotecas correspondentes (gcc e C no caso do Linux), ainda que nem todas as linguagens devam ser parte integrante do sistema operacional. Documentação e até mesmo jogos podem fazer parte. Tradicionalmente o sistema operacional tem sido definido como o conteúdo nos discos de instalação, o que no Linux pode variar bastante, já que ele pode ser encontrado em diversos sites ao redor do mundo.

2.2 Partes importantes do kernel

O kernel do Linux é formado por diferentes partes: gerenciador de processos, gerenciador de memória, controle de dispositivos de hardware, controle de sistemas de arquivos, gerenciador de rede e vários outros. A figura 2.1 mostra alguns deles.

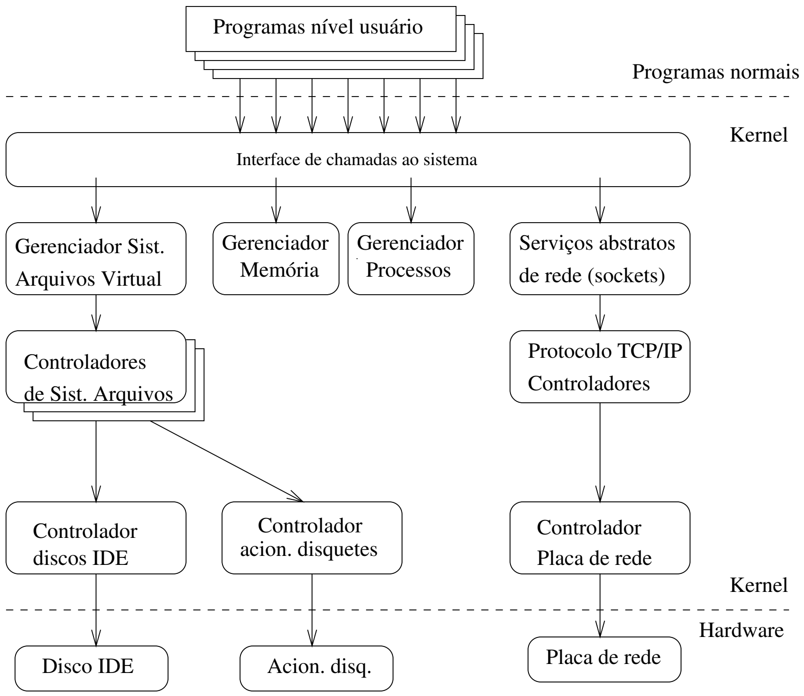

Figura 2.1: Partes importantes do kernel do Linux.

Provavelmente as partes mais importantes do kernel (nada funciona sem elas) são

[2] **user mode** no original

o gerenciador de memória e o gerenciador de processos. O gerenciador de memória é o responsável pela atribuição de áreas de memória e áreas de troca (swap) para os processos, para os demais componentes do kernel e para o cache de disco. O gerenciador de processos cria processos e implementa multitarefa através da troca dos processos ativos no processador.

No nível mais baixo, o kernel contém controladores (drivers) para cada tipo de hardware suportado. Uma vez que existem inúmeros hardwares, o número de controladores também é muito grande. A similaridade de funcionalidades permite que se tenham classes de controladores para suportar funções similares, ou seja, cada membro de uma determinada classe de hardwares tem a mesma interface com o resto do kernel, mas diferem somente no que é necessário para implementar as suas características específicas. Por exemplo todos os discos rígidos parecem iguais para o resto do kernel, isto é todos têm operações como 'inicialize o dispositivo' , 'leia o setor N' e 'grave o setor N'.

Alguns serviços de software disponibilizados pelo kernel têm propriedades similares, e podem ser também derivados em classes. Por exemplo, os diversos protocolos de rede foram abstraídos em uma interface de programação, a biblioteca BSD de socket. Outro exemplo é a camada do **sistema virtual de arquivos** (VFS), a qual abstrai as operações dos diversos sistemas de arquivos de suas implementações. Quando alguma entidade tenta utilizar um sistema de arquivos, a requisição é direcionada ao VFS, o qual a reencaminha para o controlador adequado ao sistema de arquivos solicitado.

2.3 Principais serviços em um sistema Unix

Esta seção descreve como partes importantes dos serviços UNIX são implementadas. Elas são descritas mais detalhadamente nos próximos capítulos.

2.3.1 `init`

O mais importante serviço isolado disponível em um sistema UNIX é denominado `init`. O `init` é o primeiro processo iniciado em todos os sistemas UNIX, após a carga do kernel do sistema. Quando o `init` é disparado, ele continua o processo de inicialização do sistema, executando diversas tarefas necessárias ao funcionamento do Linux (checando e montando sistemas de arquivos, iniciando servidores, etc).

A lista exata de tarefas que o `init` realiza depende do estado em que ele está. O `init` provê o conceito de modo monousuário, no qual ninguém, além do `superusuário (root)`, pode entrar no sistema, sendo utilizado somente o shell do console do sistema, ou o modo **multiusuário**, o modo padrão de início do sistema. Algumas versões generalizam isso, denominando-os **níveis de execução**; sendo os modos mono e multiusário considerados como níveis, podendo existir diversos outros, como por exemplo, aquele que permita a execução do X (interface gráfica) na console.

Em uma operação normal, o `init` certifica-se que o `getty` está sendo executado (para permitir o acesso dos usuários) e adota processos órfãos (processos cujos programas que os iniciaram (chamados pais) morreram (não estão mais sendo executados)). No UNIX *todos* os processos *devem* estar em uma estrutura do tipo árvore, devendo todos eles terem um "pai".

Quando um sistema é desligado, é o `init` que se encarrega de finalizar todos os outros processos, desmontar todos os sistemas de arquivos e parar o processador junto com tudo o mais que esteja sendo executado.

2.3.2 Acesso a partir de terminais

Os accessos a partir de terminais (via linhas seriais) e através do console (quando não se está executando o X), são disponibilizados pelo `getty`. O `init` inicia uma instância distinta do `getty` para cada terminal, no qual o acesso é permitido. O `getty` lê o nome do usuário e executa o programa de acesso `login`, o qual por sua vez processa a validação da senha do usuário. Caso usuário e senha sejam válidos, o `login` inicia o shell do usuário. Quando um shell é finalizado, isto é, o usuário sai do sistema, ou quando o `login` é finalizado porque usuário ou senha são incorretos, o `init` inicia uma nova instância do `getty`. O kernel não tem conhecimentos dos acessos, pois tudo isso é controlado pelos programas de sistema.

2.3.3 Syslog

O kernel e muitos programas do sistema produzem avisos, mensagens de erros, e outros alertas. É importante que estas mensagens possam ser lidas outra hora, mesmo que isto demore muito tempo deste o ocorrido. Para que isso seja possível elas devem ser gravadas em um arquivo. O programa responsável por este arquivamento é o `syslog`. Ele pode ser configurado para gravar as mensagens em diferentes arquivos de acordo com o nível de importância ou origem. As mensagens do kernel são freqüentemente direcionadas para arquivos distintos, pois estas mensagens são normalmente mais importantes e necessitam ser lidas com mais freqüência para se evitar/sanar problemas.

2.3.4 Comandos periódicos: `cron` e `at`

Tanto administradores do sistema como usuários comuns necessitam executar periodicamente determinados programas. O administrador do sistema quer executar um comando para limpar os diretórios que contenham arquivos temporários antigos (`/tmp` e `/var/tmp`), evitando assim que os discos rígidos fiquem cheios de arquivos sem utilidade, uma vez que nem todos os programas apagam os arquivos temporários após a sua execução.

O `cron` foi criado para atender a essa demanda. Cada usuário tem uma tabela do cron (`crontab`) à sua disposição, onde se inclui os comandos desejados e a freqüência com que eles devem ser executados. O daemon do `cron` encarrega-se de executar os comandos nos dias e horários definidos.

O comando `at` é similar ao `cron`, mas está limitado a uma única execução: o comando é executado em um determinado horário fornecido, mas não repetidas vezes.

2.3.5 Interface Gráfica

O UNIX e o Linux não incorporam a interface do usuário ao kernel. Ela é implementada pelos programas a nível de usuário, tanto no modo texto como no ambiente gráfico.

Esta solução gera uma maior flexibilidade para o sistema, mas, pela facilidade de implementação, pode-se ter uma multiplicidade exagerada de interfaces, uma para cada programa, tornando o sistema mais complexo de ser utilizado.

O ambiente gráfico usado com o Linux é chamado X Window (X). O X na verdade não implementa a interface gráfica, somente disponibiliza um sistema de janelas, onde outros programas podem implementar um ambiente gráfico . Algumas das interfaces gráficas mais populares para o Linux são: KDE, Gnome, After Step, Window Maker, fvwm e fvwm95.

2.3.6 Rede

Rede pode ser definida como a conexão entre dois ou mais computadores, de forma que eles possam comunicar-se entre si. Os métodos de conexão e comunicação são um tanto complexos, porém os resultados são muito úteis.

Os Sistemas UNIX têm muitas características de rede. Os serviços básicos de impressão, cópias de segurança, sistemas de arquivos, etc, podem ser realizados através da rede. Isso pode tornar a administração do sistema mais simples, uma vez que permite a sua administração de forma centralizada, mantendo os benefícios do processamento distribuído e do uso de microcomputadores, gerando menores custos e melhor tolerância a falhas.

Ainda que este livro aborde superficialmente o tema, veja o Guia do Administrador de Redes para maiores informações, inclusive uma descrição mais profunda sobre o funcionamento de redes.

2.3.7 Acesso via Rede

Os acessos via rede são um pouco diferentes dos acessos normais, onde há uma linha serial distinta para cada terminal, através da qual é possível acessar o sistema. Nos casos de acesso via rede, para cada usuário que conecta-se ao sistema, há uma conexão virtual separada, podendo haver[3] um número ilimitado delas [4]. Não é mais possível executar um `getty` para cada conexão virtual possível. Existem muitas formas diferentes de conectar-se via rede, sendo o `telnet` e `rlogin` os mais utilizados em redes TCP/IP.

Acessos via rede têm, ao invés de inúmeros `getty`, um servidor controlando o acesso (o `telnet` e o `rlogin` utilizam servidores distintos), o qual monitora as tentativas de conexão. Quando algum usuário tenta conectar-se, o servidor inicia uma nova instância de si próprio para controlar esta conexão enquanto o servidor original continua monitorando novas tentativas. As novas instâncias funcionam de maneira similar ao getty.

2.3.8 Sistemas de Arquivos em Rede (NFS)

Uma das maiores facilidades disponibilizadas pelos serviços de rede é o compartilhamento de arquivos através de **sistemas de arquivos em rede**. O mais usado é conhecido como NFS (Network File System), tendo sido desenvolvido pela Sun.

Com um sistema de arquivos em rede, qualquer operação com arquivos executada por um programa em uma máquina, é enviada pela rede para outro computador. Esse procedimento faz o programa "pensar" que todos os arquivos encontram-se no mesmo equipamento onde ele está sendo executado. Isso torna o compartilhamento de informações muito simples, já que não requer nenhuma modificação nos programas utilizados.

2.3.9 Correio Eletrônico

O correio eletrônico é normalmente o mais importante meio de comunicação entre os usuários. Uma mensagem eletrônica é armazenada em um arquivo, formatada de maneira especial, e programas de correio específicos são utilizados para ler e enviar mensagens.

Cada usuário tem uma **caixa de entrada de mensagens** [5] (um arquivo com um formato especial) onde todas as novas mensagens são armazenadas. Quando alguém envia uma nova mensagem, o programa de correio localiza a caixa do destinatário e

[3] teoricamente

[4] Bem, pelo menos um número bem grande. A banda das redes ainda é um recurso escasso. Sendo assim utiliza-se limites práticos para o número de conexões simultâneas que podemos ter em uma conexão de rede.

[5] mailbox

adiciona a nova mensagem ao arquivo de mensagens. Caso a caixa de entrada esteja em outra máquina, a mensagem é enviada através da rede para o equipamento que mantém os arquivos de correio do destinatário.

O sistema de correio é constituído por uma série de programas. A entrega de mensagens para usuários remotos é realizada por um programa específico (o **agente de transferência de mensagens** ou **MTA**, como por exemplo o `sendmail` ou `smail`), enquanto os programas de usuários são também distintos (**agente de correio de usuário MUA**, p. ex. `pine`, `elm` ou `kmail`). As caixas postais são normalmente armazenadas no diretório `/var/spool/mail`.

2.3.10 Impressão

Somente um usuário pode utilizar uma impressora a cada vez, mas seria antieconômico não compartilhar esse recurso entre os demais usuários da rede. A impressora é gerenciada por um software que implementa o conceito de **fila de impressão**: todos os trabalhos são colocados em uma fila e toda vez que a impressora finaliza um trabalho de impressão, o seguinte é iniciado automaticamente. Isso evita que os usuários tenham que organizar esta fila ou que haja problemas no controle do uso da impressora[6].

O software de fila de impressão também grava os trabalhos de impressão em disco, ou seja os textos são mantidos em um arquivo enquanto estejam na fila de impressão. Isso permite que uma aplicação transfira rapidamente os trabalhos de impressão para o software gerenciador da fila, não tendo que esperar que o trabalho de impressão seja finalizado para continuar a sua execução.

2.4 A estrutura do sistema de arquivos

O sistema de arquivos é dividido em diversas partes, normalmente utilizando uma estrutura em árvore, com um sistema de arquivos chamado raiz (root), simbolizado por '/'. Debaixo deste podemos ter diversos outros: `/bin`, `/lib`, `/etc`, `/dev`, etc.... Um sistema de arquivos `/usr` pode conter programas e dados não atualizáveis, o `/var` pode conter dados variáveis (como arquivos de log) e um sistema `/home` pode ter arquivos pessoais de todos os usuários. Dependendo da configuração de hardware e das definições do administrador do sistema, a divisão pode variar substancialmente, sendo possível inclusive que todos os conteúdos estejam em um único sistema de arquivos.

No capítulo 3 apresentaremos a estrutura dos sistemas de arquivos em mais deta-

[6] Ao invés disso, as pessoas fazem uma fila *em frente* à impressora, esperando pelos seus impressos, já que eles não sabem quando a fila irá liberar seus trabalhos. Isto ajuda muito nas relações sociais do escritório!

lhes, sendo que o documento que descreve o Sistema de Arquivos Padrão do Linux (FSSTND) cobre o tema mais profundamente.

Capítulo 3

Visão Geral da Árvore de Diretórios

Dois dias depois, lá estava Pooh,
sentado em seu banco, balançando suas pernas,
e ao seu lado, quatro potes de mel...
(A.A. Milne)

Este capítulo descreve as partes importantes da árvore de diretórios do Linux, baseado no Sistema de Arquivos Padrão do Linux - FSSTND. Ele descreve a forma usual de dividir a árvore de diretórios em diferentes sistemas de arquivos com diferentes propósitos, justificando a adoção de cada divisão. Algumas opções alternativas de particionamento são também descritas.

3.1 Background

Este capítulo está de certa forma baseado no Sistema de Arquivos Padrão do Linux (FSSTND), versão 1.2, (veja `fsstnd-1.2`) o qual busca definir uma padronização para a organização da árvore de diretórios no Linux. Tal padrão tem a função de facilitar o desenvolvimento e o porte para o Linux, assim como a administração de máquinas Linux, uma vez que tudo estará localizado em caminhos conhecidos. Não há obrigatoriedade da adoção desta padronização, nem nada que obrigue quem quer que seja a utilizá-la, além do bom senso, uma vez que ela tem o apoio da maioria, senão da totalidade, das distribuições Linux. Portanto não é aconselhável alterar o padrão da árvore de diretórios, a menos que se tenha razões muito fortes para isso. O FSSTND busca manter os padrões antigos do UNIX, conciliando-os com novas demandas, fazendo com o que o Linux torne-se mais familiar para aqueles com alguma experiência em outros UNIX e vice e versa.

Este capítulo não é tão detalhado como o FSSTND. Um administrador de sistemas deverá ler aquela especificação para um completo entendimento.

Também não são descritos todos os arquivos em detalhes. A intenção é de apresentar uma visão geral, sob um ponto de vista de sistemas de arquivos. Maiores informações sobre cada arquivo podem ser encontradas em outros pontos deste manual ou nas páginas de manual do Linux.

A árvore de diretórios deve ser dividida em partes menores, cada uma com o seu próprio disco ou partição, facilitando questões como limites de tamanho de disco ou a geração de cópias de segurança. Os sistemas de arquivo mais importantes são o raiz (root), `/usr` , `/var` , e `/home` (vide figura 3.1). Cada parte tem um propósito diferente. O desenho da árvore foi idealizado para funcionar tanto em máquinas que necessitem compartilhar partes do sistema de arquivos com autorização somente de leitura (por exemplo um CD-ROM), assim como na rede via NFS.

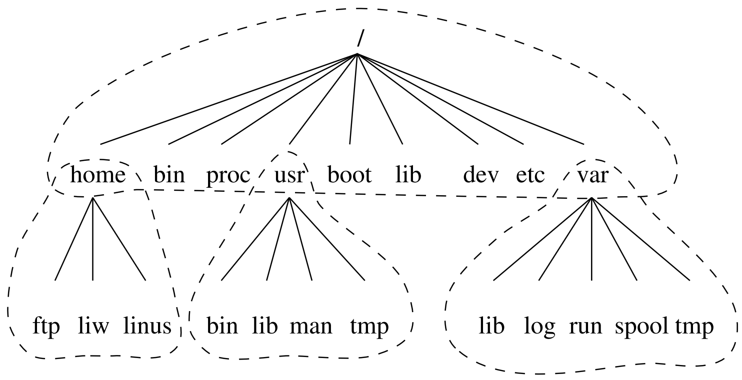

Figura 3.1: Partes de uma árvore de diretório do Unix. Linhas tracejadas indicam os limites das partições.

As regras para tal divisão são descritas a seguir.

- O sistema de arquivos raiz é específico de cada máquina (ele é geralmente armazenado em um disco local, apesar de poder ser um disco em memória (ramdisk) ou uma unidade de rede) e contém os arquivos necessários para a inicialização do sistema e para permitir o acesso aos demais sistemas de arquivos. O conteúdo do sistema de arquivos raiz deverá ser suficiente ainda para que o sistema possa ser executado em modo monousuário. Ele contém ainda ferramentas para resolver eventuais problemas de disco e recuperar arquivos a partir de cópias de segurança.

- O sistema de arquivos `/usr` contém os comandos, bibliotecas, páginas de manual e outros arquivos imutáveis, necessários para a operação normal do sistema. Os arquivos no `/usr` não devem ser específicos para uma determinada máquina e não devem ser alterados durante o uso normal. Isso permite que estes arquivos possam ser compartilhados através da rede, o que pode ser bastante interessante pela possível economia de espaço em disco, (o `/usr` pode conter facilmente centenas de megabytes) torna a administração mais simples (uma vez que somente o

/usr na máquina principal necessita ser atualizado e mantido em dia[1]). Mesmo que este sistema de arquivos esteja no disco local, ele deve ser montado com autorização somente de leitura, para diminuir a chance de corrupção de arquivos durante um eventual acidente.

- O sistema de arquivos /var contém arquivos mutáveis, como diretórios de arquivos temporários, spool (para correio eletrônico, news, impressões, etc), logs, páginas de manual formatadas, etc... Tradicionalmente o conteúdo do /var era montado em algum lugar sob o /usr, porém isso tornava impossível montar o /usr com permissões somente de leitura.

- O sistema de arquivos /home contém os diretórios pessoais dos usuários, ou seja, todas as informações armazenadas por estes no sistema. Separar diretórios pessoais em sua própria árvore de diretórios ou sistema de arquivos simplifica a geração de cópias de segurança, uma vez que outras partes do sistema normalmente não necessitam de cópias com a mesma freqüência. Um /home muito grande pode ser dividido em diversos sistemas de arquivos que requerem um nível adicional na sua denominação, p. ex., /home/alunos ou /home/funcionários.

Apesar das diferentes partes acima serem chamadas de sistemas de arquivos, não há obrigatoriedade que elas estejam assim separadas. Elas podem estar facilmente no mesmo sistema de arquivos em uma pequena máquina utilizada por um único usuário que deseje mantê-las de uma forma mais simplificada. A árvore de diretórios pode estar dividida ainda em diferentes sistemas de arquivos, dependendo do tamanho de cada disco e de quanto espaço será alocado para cada finalidade. A parte importante é que os nomes padrão funcionam, mesmo que o /var e o /usr estejam na mesma partição, os arquivos /usr/lib/libc.a e /var/adm/messages apontarão para os locais corretos, pois somente é necessário mover os dados do /var para /usr/var e fazer um link simbólico do /var para o /usr/var.

A estrutura dos sistemas de arquivos UNIX prevê um agrupamento de acordo com o propósito. Os comandos estão todos em uma determinada área, todos os arquivos de dados em uma outra, documentação em uma terceira, e assim por diante. Uma alternativa seria agrupar os arquivos de acordo com os programas a que eles pertençam, isto é, os arquivo do Emacs estariam todos em um único diretório, todos os arquivos do TEX em outro, e assim por diante. O problema dessa abordagem reside no compartilhamento de arquivos (um diretório de programas contém todos os tipos de arquivos: estáticos, dinâmicos, compartilhados e não compartilhados,...) e algumas vezes em encontrar alguns desses arquivos (por exemplo uma página de manual a ser pesquisada em um grande número de diretórios é um pesadelo para a administração).

[1] note que ao utilizar sistemas baseados na tecnologia rpm, os pacotes devem ser instalados somente na máquina principal

3.2 O Sistema de arquivos raiz (root)

O sistema de arquivos raiz deve geralmente ser pequeno, pois ele contém somente os arquivos fundamentais para a inicialização e funcionamento do sistema. Um sistema pequeno e pouco modificado tem uma chance maior de não ter problemas. Um sistema de arquivos raiz corrompido normalmente significa que o sistema não poderá ser inicializado, exceto através de medidas especiais (como por exemplo, por disquetes).

O diretório raiz geralmente não contém nenhum arquivo, exceto, em algumas distribuições, pela imagem de inicialização (boot) do sistema chamada `/vmlinuz`[2]. Todos os outros arquivos estão em subdiretórios do raiz:

`/bin` Comandos necessários durante a inicialização do sistema que podem ser utilizados pelos usuários (provavelmente após a ativação).

`/sbin` Similar ao `/bin`, porém os comandos não são destinados aos usuários comuns, apesar de poderem ser utilizados por estes se necessário.

`/etc` Arquivos de configuração específicos da máquina.

`/root` O diretório pessoal do `superusuário` (root).

`/lib` Bibliotecas compartilhadas necessárias aos programas no sistema de arquivos raiz.

`/lib/modules` Módulos dinâmicos e carregáveis pelo kernel, especialmente aqueles necessários para inicializar o sistema em caso de acidentes (por exemplo programas de controle de sistemas de arquivos de redes).

`/dev` Arquivos de dispositivos.

`/tmp` Arquivos temporários. Programas que são executados após a ativação do sistema devem usar o `/var/tmp` e não o `/tmp`, uma vez que provavelmente encontrarão mais espaço disponível neste sistema de arquivos.

`/boot` Arquivos utilizados pelo gerenciador de inicialização do sistema, p. ex., LILO. Imagens do kernel são normalmente mantidas aqui ao invés de no diretório raiz. Caso existam inúmeras imagens, o diretório pode facilmente crescer demasiadamente, sendo aconselhável mantê-lo em um sistema de arquivos à parte. Outra razão é estar seguro de que as imagens do kernel estejam nos primeiros 1024 cilindros de um disco IDE.

`/mnt` Ponto para montagens temporárias realizadas pelo administrador do sistema. Programas usualmente não prevêem a montagem automática

[2] No Conectiva Linux a imagem do kernel encontra-se no diretório `/boot`

no `/mnt`. Este deve estar dividido em subdiretórios, como por exemplo `/mnt/dosa` sendo uma unidade de disquetes utilizando um sistemas de arquivos MS-DOS, e `/mnt/exta` pode ser a mesma unidade, porém com uma extensão ext2.

`/proc`, `/usr`, `/var`, `/home` Pontos de montagem para outros sistemas de arquivos.

3.2.1 O diretório `/etc`

O diretório `/etc` contém inúmeros arquivos. Alguns deles estão descritos a seguir. Para outras descrições é necessário determinar a quais programas eles pertencem e ler as páginas de manual daqueles programas. Para sistemas que utilizam a tecnologia RPM, basta executar "`rpm -qf arquivo`" que o sistema lhe dirá a qual pacote o arquivo pertence. Muitos arquivos de configuração de programas de rede estão no `/etc`, assim como encontram-se descritos no Guia do Administrador de Redes[3].

`/etc/rc` ou `/etc/rc.d` ou `/etc/rc?.d` São scripts (arquivos de lote) ou os diretórios de scripts que são executados no início do sistema ou na mudança de nível do sistema. Veja o capítulo 7 sobre o processo `init` para maiores informações.

`/etc/passwd` A base de dados de usuários, cujos campos definem nome do usuário, nome real, diretório pessoal, senha criptografada e outras informações específicas de cada usuário. O formato está documentado na página de manual do comando *passwd* ou no Apêndice E.

`/etc/fdprm` A tabela de parâmetros de disquetes. Descreve os diferentes formatos disponíveis. É utilizada pelo programa `setfdprm`. Para maiores informações veja página de manual.

`/etc/fstab` Lista os sistemas de arquivos montados automaticamente, durante a inicialização do sistema, pelo comando `mount -a` (no script de inicialização `/etc/rc.d/rc.sysinit` ou similar). No Linux, contém ainda informações sobre as áreas de troca (swap) usadas automaticamente pelo comando `swapon -a`. Veja a a seção 4.8.5 e a página de manual do *mount* para maiores informações.

`/etc/group` Similar ao `/etc/passwd`, porém descreve grupos ao invés de usuários. Veja a página de manual do *group* para maiores informações.

`/etc/inittab` Arquivo de configuração do `init`.

`/etc/issue` Saída do programa `getty` antes do prompt de acesso ao sistema. Nor-

[3]"The Linux Network Administrator's Guide" de Olaf Kirch, traduzido pela Conectiva Informática e pertencente a Edição Servidor do Conectiva Linux.

malmente contém uma breve descrição ou mensagens de boas-vindas. O conteúdo fica a critério do administrador do sistema[4].

`/etc/motd` A **mensagem do dia**, automaticamente apresentada após um acesso bem sucedido. Conteúdos ficam a critério do administrador do sistema. Normalmente são usadas para enviar informações para os usuários, como por exemplo avisos de desligamentos planejados.

`/etc/mtab` Lista dos sistemas de arquivos montados. Inicialmente configurado por scripts e posteriormente atualizado pelo comando `mount`. Usado quando uma listagem dos sistemas de arquivos é necessária, como por exemplo, pelo comando `df`.

`/etc/shadow` Arquivo de senhas em sistemas onde o `shadow` esteja instalado. Esta opção move as senhas criptografadas do `/etc/passwd` para o arquivo `/etc/shadow` o qual somente pode ser lido pelo superusuário. O `shadow` dá uma maior segurança às senhas dos usuários.

`/etc/login.defs` Arquivo de configuração do programa `login`.

`/etc/termcap` O arquivo de configuração de terminal. Descreve as "seqüências de escape" para os diversos tipos de terminais. Os programas ao invés de escreverem diretamente uma seqüência que funcione em determinado tipo de terminal, devem buscar a seqüência correta no `/etc/termcap`. Desta maneira os programas funcionam na maioria dos terminais. Veja as páginas de manual do *termcap*, *curs_termcap* e *terminfo* para maiores informações.

`/etc/printcap` Similar ao `/etc/termcap`, mas direcionado à impressoras e com sintaxe diferente.

`/etc/profile`, `/etc/bashrc` Arquivos executados pelo Bourne Shell e Bash, no momento do login do usuário. Estes arquivos permitem ao administrador manipular o ambiente de trabalho de todos os usuários do sistema. Veja as páginas de manual dos interpretadores de comandos para detalhes específicos.

`/etc/securetty` Identifica terminais seguros, ou seja, aquele nos quais é permitido que o `root` acesse o sistema. Normalmente somente os consoles virtuais são listados, tornando muito mais difícil obter os privilégios de superusuário do sistema através da rede ou de uma conexão via modem.

`/etc/shells` Lista as shells válidas. O comando `chsh` permite que os usuários mudem sua shell de login, porém, somente para aquelas listadas neste

[4] No Conectiva Linux, o `/etc/issue` é inicializado no arquivo `/etc/rc.d/rc.local`

arquivo. O servidor de transferência de arquivos, `ftpd`, checará se a shell do usuário está listada no `/etc/shells` e não aceitará acessos, a menos que ela esteja presente neste arquivo.

3.2.2 O diretório /dev

O diretório `/dev` contém arquivos especiais (drivers) de controle de todos os dispositivos. Estes arquivos utilizam uma denominação especial descrita na lista de dispositivos (veja `device-list`). Os controladores são criados durante a instalação, e posteriormente através do script `/dev/MAKEDEV`. O `/dev/MAKEDEV.local` é um script escrito pelo administrador para criar somente os dispositivos locais que não façam parte do padrão do `MAKEDEV`.

3.3 O sistema de arquivos /usr

O sistema de arquivos `/usr` é grande, uma vez que a maioria dos programas estão ali instalados. Os arquivos do `/usr` são normalmente instalados pela sua distribuição do Linux. Programas locais são instalados no `/usr/local`[5]. Isso torna a atualização do sistema possível para uma nova versão da distribuição ou mesmo para uma distribuição completamente nova, sem que seja necessário instalar todos os programas novamente. Alguns dos subdiretórios do `/usr` estão listados a seguir (outros menos importantes podem ser encontrados no FSSTND para maiores informações).

`/usr/X11R6` Arquivos do X Window. Para simplificar o desenvolvimento e a instalação do X, seus arquivos não foram integrados ao restante do sistema. Há uma árvore de diretórios sob `/usr/X11R6` similar ao `/usr`.

`/usr/X386` Similar ao `/usr/X11R6`, mas para o X11 Release 5.

`/usr/bin` Praticamente todos os comandos de usuários. Alguns outros podem ser encontrados em `/bin` ou `/usr/local/bin`.

`/usr/sbin` Comandos de administração do sistema que não necessitem estar no sistema de arquivos raiz, como servidores.

`/usr/man`, `/usr/info`, `/usr/doc` Páginas de manual, documentos do GNU/Info e diversos outros documentos, respectivamente.

`/usr/include` Arquivos header para a linguagem de programação C. Este diretório deveria estar sob o `/usr/lib`, porém por tradição tem sido mantido neste local.

[5] pacotes adicionais são instalados atualmente no `/opt`

`/usr/lib` Arquivos estáticos de dados para programas e sub-sistemas, incluindo alguns arquivos de configuração globais. O nome `lib` vem de library (biblioteca); originalmente as bibliotecas de programação eram armazenadas neste subdiretório.

`/usr/local` O local para programas instalados localmente e outros arquivos.

`/usr/share/magic` O arquivo de configuração para o comando `file`. Contém as descrições de vários formatos de arquivos nas quais o `file` pode descobrir o tipo do arquivo. Veja as páginas de manual do *magic* e do *file* para maiores informações.

3.4 O sistema de arquivos `/var`

O sistema de arquivos `/var` contém dados que são alterados quando o sistema está sendo executado. É específico de cada sistema, ou seja, não compartilhado através da rede com outros equipamentos.

`/var/catman` Um cache para páginas de manual que são formatadas quando solicitadas. Os fontes para as páginas são armazenados em `/usr/man/man*`, sendo que algumas páginas podem vir pré-formatadas, estando armazenadas em `/usr/man/cat*`. Outras páginas necessitam ser formatadas quando acessadas pela primeira vez. A versão formatada fica armazenada em `/var/catman`, fazendo com que outro usuário que acesse a página, tenha esta pronta e disponível. O diretório `/var/catman` é freqüentemente esvaziado, da mesma forma que os arquivos temporários.

`/var/lib` Arquivos que mudam enquanto o sistema está ativo.

`/var/local` Dados variáveis para programas que estejam instalados no `/usr/local`. Note que mesmo quando instalado localmente estes programas podem usar outros subdiretórios do `/var`, como por exemplo o `/var/lock`.

`/var/lock` Arquivos de bloqueio. Muitos programas seguem uma convenção de se criar arquivos no `/var/lock` para indicar que eles estão utilizando um dispositivo em particular ou um determinado arquivo. Outros programas perceberão a existência do arquivo de bloqueio e não tentarão utilizar o mesmo dispositivo ou arquivo.

`/var/log` Arquivos de históricos de vários programas, especialmente o `login` (`/var/log/wtmp`, o qual registra todas as entradas e saídas do sistema) e o `syslog` (`/var/log/messages`, o qual contém as mensagens do kernel e programas do sistema). Os arquivos no `/var/log` podem crescer indefinidamente com freqüência e podem requerer limpezas periódicas[6].

`/var/run` Arquivos que contêm informações sobre o sistema e que são válidas até a próxima inicialização. Por exemplo, o `/var/run/utmp` contém informações sobre os usuários atualmente conectados.

`/var/spool` Diretórios para mails, news, filas de impressão e outros trabalhos em fila. Cada fila tem o seu próprio subdiretório sob o `/var/spool`, p. ex., as mensagens de correio eletrônico estão armazenadas em `/var/spool/mail`.

`/var/tmp` Arquivos temporários que sejam muito grandes ou que necessitem existir por um período maior que o definido para o `/tmp`. Note que o administrador do sistema pode não permitir arquivos muitos antigos também no `/var/tmp`.

3.5 O sistema de arquivos /proc

O `/proc` é um sistema de arquivos ilusório. Na verdade ele não existe em um disco rígido. É criado em memória pelo kernel, sendo usado para disponibilizar informações sobre o sistema (originalmente sobre os processos, daí a origem de seu nome). Alguns dos arquivos mais importantes e diretórios são detalhados abaixo. O `/proc` é descrito mais detalhadamente na página de manual *proc*.

`/proc/1` Um diretório com as informações do processo número 1 `(init)`. Cada processo tem um diretório embaixo do /proc, cujo nome equivale ao seu número de identificação.

`/proc/cpuinfo` Informações sobre o processador, tais como tipo, fabricante, modelo, e performance.

`/proc/devices` Lista dos controladores de dispositivo configurados no kernel atualmente em execução.

`/proc/dma` Mostra quais canais DMA[7] estão sendo utilizados no momento.

`/proc/filesystems` Sistemas de arquivos configurados no kernel.

`/proc/interrupts` Mostra quais interrupções estão em uso e quantas vezes foram chamadas.

`/proc/ioports` Quais portas de entrada e saída estão em uso no momento.

[6] neste caso é conveniente instalar o pacote `logrotate`

[7] Direct Memory Access: acesso direto à memória por dispositivo periférico inteligente, de entrada e saída

`/proc/kcore` Imagem da memória física do sistema. Tem exatamente o mesmo tamanho da memória física, mas não ocupa toda aquela área. Ela é gerada em tempo de execução, através do acesso dos programas. Lembre-se: a menos que você copie esta imagem para outro lugar, nada abaixo do `/proc` ocupa espaço em disco.

`/proc/kmsg` Mensagens de saída do kernel. São direcionadas também para o `syslog`.

`/proc/ksyms` Tabela de símbolos para o kernel.

`/proc/loadavg` A 'carga média' do sistema, contendo três indicadores de quanto trabalho o sistema está executando no momento.

`/proc/meminfo` Informações sobre o uso de memória, tanto física como de swap.

`/proc/modules` Descreve quais módulos estão carregados no momento.

`/proc/net` Informações sobre a situação dos protocolos de rede.

`/proc/self` Um link simbólico para o diretório de processos do programa que está olhando para o `/proc` no momento. Quando dois processos acessam o `/proc`, estes recebem diferentes links. Isto é feito para facilitar a vida dos programas quando estes olham o seu diretório de processos.

`/proc/stat` Diversas estatísticas sobre o sistema, como o número de paginações desde seu início, E/S, processos executados, etc...

`/proc/uptime` O tempo que o sistema está ativo.

`/proc/version` Versão do kernel do sistema.

Note que enquanto os arquivos acima são de fácil leitura, por estarem em formato texto, não quer dizer que eles sejam facilmente lidos. Há muitos comandos que fazem mais que somente ler os arquivos acima e formatar o seu conteúdo para um melhor entendimento. Por exemplo o programa `free` lê o conteúdo do `/proc/meminfo` e converte as informações dadas em bytes para kilobytes, com algumas pequenas informações adicionais.

Capítulo 4

Usando discos e outros dispositivos de armazenamento

Em um disco vazio pode-se pesquisar eternamente.

Ao instalar ou atualizar o sistema, é necessário preparar os discos rígidos para isso. Deve-se criar sistemas de arquivos para que arquivos possam ser armazenados e reservar espaços extras para as diferentes partes do sistema.

Este capítulo explica todas estas atividades iniciais. Normalmente, uma vez que o sistema esteja configurado, não é necessário executar esta atividade novamente, exceto ao se utilizar disquetes. Será necessário retornar a este capítulo caso um novo disco seja adicionado ao sistema ou se queira refinar a utilização do disco já instalado.

As atividades básicas na administração dos discos são:

- Formatar o disco. Nesta atividade várias tarefas são executadas, como por exemplo a checagem de setores defeituosos. (A formatação hoje não é mais necessária para diversos discos rígidos).
- Particionar o disco. Indicado caso estejam previstas a execução de diversas tarefas independentes entre si. Uma razão para o particionamento é manter diferentes sistemas operacionais no mesmo disco. Outra razão é a manutenção de arquivos de usuários separados dos arquivos do sistema, simplificando a geração de cópias de segurança e auxiliando na proteção dos arquivos do sistema operacional.
- Gerar um sistema de arquivos (de determinado tipo) para cada disco ou partição. O disco em si não tem significado para o Linux, até que um sistema de arquivos seja criado; assim arquivos podem ser criados e acessados.
- Montar diferentes sistemas de arquivos para formar uma única árvore de di-

retórios, automática ou manualmente montadas (note que sistemas de arquivos montados manualmente, necessitam ser desmontados da mesma forma).

O capítulo 5 contém ainda informações sobre memória virtual e cache de disco, os quais são conhecimentos importantes para o correto uso do disco.

Este capítulo explica o que você precisa saber sobre discos rígidos, disquetes, CD-ROM's e unidades de fita.

4.1 Dois tipos de dispositivos

O UNIX, e também o Linux, reconhecem dois tipos de dispositivos: de acesso randômico em blocos (como discos rígidos) e dispositivos a caractere (como fitas e linhas seriais), alguns dos quais podem ser de acesso randômico e outros de acesso seqüencial. Cada dispositivo suportado é representado no sistema de arquivos como um **arquivo de dispositivo** (**device**). Ao ler ou gravar em um arquivo de dispositivo, as informações vêm e vão para o dispositivo por ele representado. Desta forma nenhum programa especial (e nenhum método especial de programação, como descobrir interrupções ou vasculhar portas seriais) é necessário para acessar os dispositivos. Por exemplo para enviar um arquivo para uma impressora pode-se simplesmente executar:

```
$ cat nome_do_arquivo > /dev/lp1
$
```

e o conteúdo do arquivo será impresso (obviamente o arquivo deve estar em um formato inteligível para a impressora). Note que não é exatamente uma boa idéia terse diversos arquivos direcionados para a impressora através do `cat`. Deve-se utilizar o comando `lpr`, o qual assegura que somente um arquivo será impresso por vez, e que enviará o arquivo seguinte imediatamente após a conclusão da impressão do anterior. Algo similar é necessário para a maioria dos dispositivos. Na verdade, raramente os arquivos de dispositivos são utilizados diretamente e você não precisa se preocupar com eles.

Desde que os dispositivos são apresentados como arquivos no sistema de arquivos (no diretório `/dev`), é fácil verificar quais dispositivos existem utilizando-se por exemplo o `ls` ou outro similar. Na saída do comando `ls -l` a primeira coluna indica o tipo do arquivo e suas permissões. Veja um exemplo ao se examinar um dispositivo serial:

```
$ ls -l /dev/cua0
crw-rw----   1 root     uucp       5,  64 set 23 13:48 /dev/cua0
$
```

O primeiro caracter na primeira coluna, '`c`' em `crw-rw--`, mostra-nos o tipo do arquivo, neste caso um dispositivo de caracter. Para arquivos comuns o primeiro caracter é um '`-`', para diretórios será '`d`' e para dispositivos em blocos será '`b`'. Verifique a página de manual do *ls* para maiores detalhes.

Veja que normalmente todos os arquivos de dispositivos existem mesmo que o dispositivo em si não esteja instalado. Mesmo que exista um arquivo `/dev/sda`, não significa que realmente haja um disco SCSI instalado. Ter-se todos os arquivos de dispositivos, torna os programas de instalação mais simples, assim como facilita a adição de novos componentes de hardware, pois não há necessidade de encontrar os parâmetros corretos para criar os arquivos dos novos dispositivos.

4.2 Discos rígidos

Esta seção introduz terminologias relacionadas com discos rígidos. Caso estes termos e conceitos já sejam de seu conhecimento, continue sua leitura na próxima seção.

A figura 4.1 apresenta um esquema das partes importantes de um disco rígido. Ele consiste em um ou mais **pratos** circulares[1], cada um com uma ou ambas as **faces** cobertas por uma substância magnética usada para a gravação de dados. Para cada face, há uma **cabeça de leitura e gravação** que examina ou altera os dados ali gravados. Os pratos giram em um eixo comum e a velocidades superiores a 5.400 rotações por minuto, sendo que discos de alta performance atingem velocidades maiores. As cabeças movimentam-se ao longo do raio dos pratos; este movimento combinado com a rotação dos pratos permite o acesso pelas cabeças a qualquer parte da superfície.

O processador (CPU) e o disco comunicam-se através da **controladora de discos**. Isso retira dos outros componentes do computador o trabalho de saber como acessar o dispositivo, uma vez que as controladoras para diferentes tipos de discos podem utilizar a mesma interface com o restante do computador. Assim, o computador pode dizer "ei disco, me dê o dado que quero", ao invés de enviar uma série de sinais elétricos para mover as cabeças para o local adequado e esperar pelo correto posicionamento dos pratos e todos os demais passos necessários. Na realidade a interface da controladora é bastante complexa, mas muito menos do que poderia ser um acesso direto ao disco. A controladora pode ainda executar outras tarefas como cache e realocação de setores defeituosos.

Estas são as informações necessárias para entender o funcionamento do hardware. Há ainda muitos outros aspectos, como o motor que gira os pratos e move as cabeças, e os componentes eletrônicos que controlam as partes mecânicas do disco, porém estes aspectos não são relevantes para o processo de entendimento dos princípios básicos do disco.

[1] Os pratos são feitos de um material rígido como o alumínio, o que dá ao disco rígido o seu nome

A superfície dos pratos é normalmente dividida em anéis concêntricos, chamados **trilhas** e estes por sua vez estão divididos em **setores.** Esta divisão é utilizada para especificar as localizações dentro do disco e para alocação de espaço para os arquivos. Para encontrar um determinado ponto em um disco, deve-se dizer "face 3, trilha 5, setor 7". Normalmente o número de setores é o mesmo para todas as trilhas, mas alguns discos rígidos possuem um número maior de setores nas trilhas mais externas (todos os setores têm o mesmo tamanho físico, assim as trilhas externas comportam mais dados). Tipicamente um setor comporta 512 bytes de dados. O disco por si só não pode manipular quantidades menores de dados que um setor.

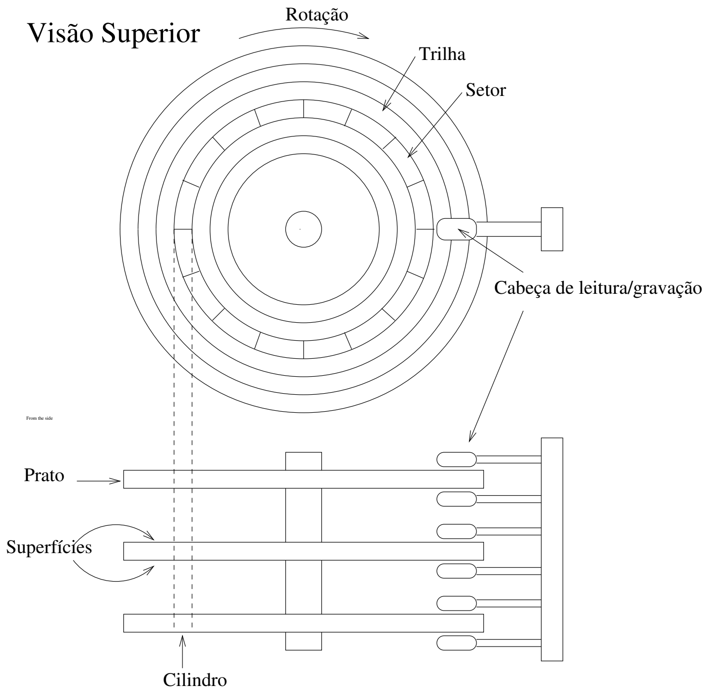

Figura 4.1: Desenho esquemático de um disco rígido

Cada face é dividida em trilhas (e setores) do mesmo modo. Assim quando uma cabeça está sobre uma trilha, as cabeças das outras faces estão na mesma trilha correspondente. O conjunto destas trilhas formam um **cilindro.** O movimento das cabeças de uma trilha (ou cilindro) para outra utiliza um determinado intervalo de tempo. Colocando-se os dados que são normalmente acessados de forma conjunta (digamos um arquivo) em localizações contíguas dentro do disco, o movimento das cabeças será menor, melhorando a performance do sistema. Nem sempre é possível armazenar os arquivos desta forma; arquivos que estão dispostos em diversos locais do disco são chamados de **fragmentados**.

O número de faces (ou cabeças, que é o mesmo), cilindros e setores varia bastante. A especificação deste número é chamada **geometria** do disco rígido. A geometria é normalmente arquivada de forma especial, em uma posição da memória alimentada por pilhas, chamada **CMOS RAM**, da qual o sistema operacional pode recuperar durante a inicialização do sistema.

Infelizmente a BIOS[2] tem um desenho limitado, o que torna impossível especificar na CMOS RAM um número de trilhas maiores que 1.024, o que é muito pouco para discos maiores. A saída encontrada para isso foi a controladora "mentir" sobre a geometria e **converter os endereços** dados pelo computador em endereços corretos. Por exemplo, consideremos um disco rígido que tenha 8 cabeças, 2048 trilhas e 35 setores por trilha[3]. A controladora pode informar que o disco tem 16 cabeças, 1024 trilhas e 35 setores por trilha, não excedendo o limite do número de trilhas e dividindo à metade o número de trilhas e duplicando o de cabeças para chegar à capacidade exata do disco. Na realidade a matemática pode ser um pouco mais complicada, porque os números podem não ser tão exatos como no nosso exemplo, porém estes detalhes não são relevantes para o entendimento do princípio. Esta conversão distorce a visão do disco pelo sistema operacional, tornando impraticável o uso de todos os dados em um cilindro para o incremento da performance.

A conversão é um problema somente em discos IDE. Discos SCSI têm um número seqüencial de setores (a controladora traduz o número seqüencial do setor em um endereço composto por cabeça, cilindro e setor), e usa um método completamente diferente de comunicação entre a CPU e a controladora, não tendo este tipo de problema. Note que de qualquer forma o computador pode também não saber a real geometria de um disco SCSI.

O Linux com freqüência não conhecerá a real geometria de um disco, seus sistemas de arquivos sequer tentarão manter os arquivos em um único cilindro. Ao invés disso ele tentará utilizar setores consecutivos para os arquivos, o que proporciona quase a mesma performance. O tema complica-se com caches e acessos automáticos feitos pela controladora.

Cada disco rígido é representado por um arquivo de dispositivo distinto. Eles podem (normalmente) ser dois ou quatro para discos IDE, e são conhecidos como `/dev/hda`, `/dev/hdb`, `/dev/hdc` e `/dev/hdd`, respectivamente. Discos SCSI são conhecidos como `/dev/sda`, `/dev/sdb`, `/dev/sdc` e /dev/sdd. Convenções similares para nomes existem para outros tipos de discos; veja `device-list` para mais informações. Note que os arquivos de dispositivos de discos rígidos dão acesso a todo o disco, sem preocupar-se com as partições (que serão discutidas adiante), e pode ser simples desorganizar as partições ou as informações nelas contidas caso não se tenha cuidado. Os arquivos de dispositivos de discos são normalmente utilizados somente para a obtenção de dados do registro mestre de inicialização (MBR).

[2]A BIOS é um programa básico do computador, armazenada em ROM. Ela cuida, entre outras coisas, do estágio inicial de inicialização do sistema

[3]Os números são fictícios

4.3 Discos flexíveis

Um disquete consiste em uma membrana flexível coberta em um ou ambos os lados com uma substância magnética similar a de um disco rígido. O disquete por si só não tem uma cabeça de leitura e gravação, a qual está na unidade de disquetes. Um disquete corresponde a um prato do disco rígido, porém é removível e uma unidade de disquetes pode tratar diferentes disquetes, enquanto um disco rígido é uma unidade indivisível.

Como um disco rígido, um disquete está dividido em trilhas e setores (e as trilhas de ambos os lados do disquete formam um cilindro), mas em menor quantidade que em um disco rígido.

Uma unidade de disquetes pode normalmente tratar diferentes tipos de disquetes; por exemplo, uma unidade de $3\frac{1}{2}$ polegadas pode usar tanto discos de 720 Kb como de 1.44 Mb. Porém o dispositivo terá que operar de forma um pouco diferente e o sistema operacional deverá saber qual o real tamanho do disquete. Para tanto há diversos arquivos de dispositivos para unidades de disquetes, uma para cada combinação de dispositivo e tipo de disquete. O `/dev/fd0H1440` é o primeiro dispositivo de disquetes (`fd0`), que serve para uma unidade de $3\frac{1}{2}$ polegadas, de alta densidade (`H` - high) e com a capacidade de 1440 Kb (`1440`).

Os nomes dos arquivos de controle de unidades de disquetes são complexos, e no Linux há tipos especiais de dispositivos que podem detectar automaticamente o tipo de disco na unidade. Isso funciona através da leitura do primeiro setor de um disquete recém inserido, usando diferentes tipos de dispositivos, até que o dispositivo correto seja encontrado. Isso naturalmente requer que o disquete esteja formatado. Os dispositivos automáticos são chamados `/dev/fd0`, `/dev/fd1` e assim por diante.

Os parâmetros que os dispositivos automáticos utilizam podem ser configurados através do programa `setfdprm`. Isso pode ser útil caso não se utilizem tamanhos usuais de discos, ou se, por exemplo, eles têm um número diferente de setores, ou se a autodetecção falhou por alguma razão e o dispositivo correspondente não existe.

O Linux pode manusear diversos formatos de disquetes não padronizados. Alguns desses formatos requerem a utilização de programas especiais de formatação. Não comentaremos esses tipos, porém você pode verificar o arquivo `/etc/fdprm`. Ele especifica os parâmetros que o comando `setfdprm` reconhece.

O sistema operacional deve saber toda a vez que o disquete for substituído na unidade, para evitar, por exemplo, manter dados no cache de um disquete não mais disponível. Infelizmente a linha de sinalização utilizada para transmitir esta informação algumas vezes pode não estar disponível ou pior nem sempre haverá notificação ao se usar o dispositivo dentro do MS-DOS. Caso problemas estranhos estejam ocorrendo, esta pode ser uma possível razão. O único meio de corrigir este problema é substituir a unidade de disquetes.

4.4 CD-ROM's

O leitor de CD-ROM utiliza um leitor ótico circular. A informação é gravada na superfície do disco[4] em pequenos 'orifícios' alinhados ao redor de uma espiral do centro para a borda. O dispositivo direciona um feixe de laser na espiral para poder ler os dados contidos no disco. Quando o laser atinge um dos 'orifícios', o laser é refletido de uma determinada forma; quando ele atinge a superfície lisa é refletido de outra. Isso permite a criação de um código de bits e portanto de informação. O restante é simples, mecânico.

Leitores de CD-ROM são lentos, quando comparados a discos rígidos. Onde um disco rígido típico tem um tempo médio de acesso de menos de 15 milisegundos, um CD-ROM rápido pode chegar a décimos de segundos a cada leitura. A taxa atual de transferência está em torno das centenas de kilobytes por segundo. A lentidão dos CD-ROM significa que eles não são tão agradáveis de usar quanto os discos rígidos. Porém para instalações de novos programas, os CD-ROMs são muito indicados , uma vez que a velocidade não é essencial neste momento. Algumas distribuições do Linux disponibilizam sistemas de arquivos prontos para execução em CD-ROMs, tornando a instalação mais simples e economizando bastante espaço em disco.

Há muitas formas de arrumar os dados em um CD-ROM. O mais popular é o especificado pelo padrão internacional ISO 9660. Este padrão especifica um sistema de arquivos mínimo, que é mais rudimentar do que o do MS-DOS. Por outro lado, é tão simples que qualquer sistema operacional poderá mapeá-lo em modo nativo.

Para uso normal no UNIX, o sistema de arquivos ISO 9660 não é muito útil, então uma extensão chamada `Rock Ridge` foi desenvolvida. A Rock Ridge permite nomes longos de arquivos, links simbólicos e uma série de outras facilidades, tornando o CD-ROM similar a qualquer sistema de arquivos UNIX. E melhor ainda, um sistema de arquivos Rock Ridge é ainda um sistema de arquivos válido para o ISO 9660, tornando-o passível de utilização também sob sistemas não-UNIX. O Linux suporta ambos os formatos: ISO 9660 e Rock Ridge, sendo as extensões reconhecidas automaticamente.

O sistema de arquivos é somente metade da questão. Muitos CD-ROMs contêm dados que requerem programas especiais de acesso e muitos desses programas podem não ser compatíveis com o Linux (exceto, possivelmente, sob o dosemu, a emulação MS-DOS do Linux).

Um leitor de CD-ROM é acessado através do arquivo de dispositivo correspondente. Há muitas formas de conectar uma unidade de CD-ROM ao computador: via SCSI,

[4]Na verdade, no disco metálico dentro da proteção plástica

via placa de som, ou via EIDE. Os detalhes de hardware estão fora do escopo deste livro. Porém o tipo de conexão definirá o arquivo de dispositivo. Veja `device-list` para maiores detalhes.

4.5 Fitas

Uma unidade de fitas utiliza fitas similares[5] aos cassetes usados para música. Uma fita é serial por natureza, o que significa que para obter qualquer informação ali armazenada, é necessário passar por todas as outras partes que estejam antes do ponto desejado. Um disco pode ser acessado randomicamente, isto é, pode-se buscar um dado diretamente, independente do ponto em que esteja no disco. O acesso serial torna as fitas bastante lentas, quando comparadas com disco rígidos.

Por outro lado, as fitas são relativamente baratas, uma vez que elas não necessitam ser rápidas. Elas podem ter tamanhos extremamente longos e conseqüentemente podem reter uma grande quantidade de dados. Isso torna as fitas muito apropriadas para atividades como arquivamentos ou para se gerar cópias de segurança, as quais não requerem grandes velocidades, mas possuem o benefício de baixo custo e alta capacidade de armazenamento.

4.6 Formatação

Formatar é o processo de inicializar uma mídia magnética, gerando-se as marcas de trilhas e setores. Antes de um disco ser formatado, a sua superfície magnética é uma confusão de sinais magnéticos. Após a sua formatação, alguma ordem é dada a esse caos, basicamente delineando-se as linhas onde as trilhas estarão presentes e onde elas estarão divididas em setores. O detalhamento deste processo mostra algumas diferenças em relação ao princípio, porém dentro do escopo deste manual isto não é relevante. O que realmente importa é que um disco não pode ser utilizado até que seja formatado.

A terminologia pode ser um pouco confusa aqui: no MS-DOS a palavra formatação é utilizada com a abrangência também da criação dos sistemas de arquivos (a qual será discutida a seguir). Estes dois processos estão geralmente combinados, especialmente em disquetes. Quando uma distinção faz-se necessária, a real formatação é chamada de **formatação de baixo nível**, enquanto que a geração de sistemas de arquivos é denominada de **formatação de alto nível**. No UNIX as duas são denominadas: formatação e criação de sistemas de arquivos, e é assim que utilizaremos neste livro.

Para discos IDE e alguns SCSI, a formatação é realizada na fábrica e não necessita

[5] Mas completamente diferentes, é lógico!

ser repetida; por isso não há porque preocupar-se com o tema. Na verdade formatar um disco rígido pode causar problemas no seu funcionamento, por exemplo, por ser necessário formatar o disco de uma forma especial ou diferente para permitir a realocação automática de setores defeituosos.

Discos que necessitem ser formatados freqüentemente requerem um programa especial, uma vez que a interface de formatação lógica dentro do dispositivo varia de disco para disco. O programa de formatação é normalmente fornecido com a controladora ou através de um programa DOS, sendo que nenhum dos dois tende a ser de simples utilização sob o Linux.

Durante a formatação pode-se encontrar pontos ruins no disco, chamados de **blocos defeituosos**[6] ou **setores defeituosos**[7]. Algumas vezes isso é administrado pelo próprio dispositivo, mas mesmo assim, caso o seu número seja muito grande, algo precisará ser feito para evitar o uso destas partes do disco. A lógica para se fazer isso reside nos sistemas de arquivos; como adicionar estas informações ao sistema de arquivos está descrito abaixo. Alternativamente pode-se criar uma pequena partição que abrigue somente os setores defeituosos do disco, sendo essa abordagem mais adequada caso existam porções substanciais do disco com defeitos, uma vez que o sistema de arquivos pode ter problemas com grandes áreas de blocos defeituosos.

Disquetes são formatados com o comando `fdformat`. O arquivo de dispositivos da unidade de disquetes é informado através de parâmetro. O seguinte comando é dado para se formatar um disco de $3\frac{1}{2}$ polegadas e de alta densidade na primeira unidade de disquetes:

```
$ fdformat /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... read: Success
$
```

Note que caso você necessite autodetectar o tipo de dispositivo (por exemplo `/dev/fd0`), *deve-se* configurar os parâmetros com o comando `setfdprm` primeiramente. Para conseguir o mesmo resultado do comando anterior, deve-se executar os seguintes passos:

```
$ setfdprm /dev/fd0 1440/1440
$ fdformat /dev/fd0
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... read: Success
$
```

[6] **bad blocks**
[7] **bad sectors**

Normalmente é mais conveniente escolher o arquivo de dispositivo que se adapte com o tipo de disquete. Note que não é recomendado formatar disquetes com uma capacidade maior de armazenamento do que aquela original de fábrica.

O `fdformat` validará o disquete, isto é, verificará os blocos defeituosos, testando-os diversas vezes (pode-se ouvir isto, pois o barulho da unidade cresce substancialmente). Se o disquete tem somente problemas superficiais (como sujeira na cabeça de leitura e gravação, alguns erros podem ser falsos), o `fdformat` será executado normalmente, mas se um erro real for encontrado, o processo será finalizado imediatamente. O kernel apresentará mensagens para cada erro que seja encontrado, as quais serão apresentadas na console, ou, caso o `syslog` esteja sendo usado, no arquivo `/var/log/messages`. O `fdformat` não identificará nem apresentará a localização exata do erro, pois um disquete é barato o suficiente para que possa ser substituído imediatamente.

```
$ fdformat /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... read: Unknown error
$
```

O comando `badblocks` pode ser utilizado para pesquisar qualquer disco ou partição na detecção de blocos defeituosos (inclusive em um disquete). Ele não formata o disco, portanto pode ser utilizado para sistemas de arquivos já existentes. O exemplo a seguir verifica um disquete de $3\frac{1}{2}$ polegadas que apresentou dois blocos com defeitos:

```
$ badblocks /dev/fd0H1440 1440
718
719
$
```

A saída do programa apresenta os números dos blocos defeituosos encontrados. Muitos sistemas de arquivos podem evitar tais blocos. Eles mantêm uma lista destes, a qual é inicializada quando o sistema de arquivos é gerado e posteriormente atualizada. A pesquisa inicial por blocos defeituosos pode ser realizada pelo comando `mkfs` (que inicializa sistemas de arquivos), mas posteriormente deve ser realizada pelo comando `badblocks` e novos blocos podem ser adicionados com o comando `fsck`. Estes comandos serão descritos posteriormente.

Muitos discos novos avisam sobre blocos defeituosos e tentam consertá-los, usando em seu lugar um bloco especial, que não tenha problemas e foi reservado para estas situações. Isso é transparente para o sistema operacional e tal facilidade deve estar documentada no manual do disco rígido. Mesmo este tipo de disco pode falhar caso o número de blocos defeituosos cresça demasiadamente, apesar que nesta altura o disco deverá estar tão ruim que você já estará pensando em substituí-lo.

4.7 Partições

Um disco rígido pode ser divido em diversas **partições**. Cada partição funciona como se fosse um disco rígido em separado. A idéia é que se há somente um disco rígido, e você deseja ter dois sistemas operacionais instalados, pode-se dividí-lo em duas partições. Cada sistema operacional teria a sua partição própria, sem interferir nos dados do outro sistema. Desta forma eles podem coexistir pacificamente no mesmo dispositivo. Sem o particionamento deveriam haver dois discos, um para cada sistema.

Disquetes não são particionados. Não há razões técnicas contrárias, porém pelo fato de serem tão pequenos, partições não têm aplicação prática nesta mídia. CD-ROMs também normalmente não são particionados, uma vez que é muito simples utilizá- los como um grande disco, e dificilmente haverá mais de um sistema operacional em um CD.

4.7.1 O MBR, setores de inicialização e tabela de partições

As informações de como um disco rígido está particionado são armazenadas no seu primeiro setor (isto é, no primeiro setor, da primeira trilha da primeira face do primeiro disco). O primeiro setor é o **master boot record**[8] (MBR) do disco. Este é o setor que a BIOS lê e inicializa, quando a máquina é ligada. O master boot record contém um pequeno programa que lê a tabela de partições, verifica qual partição está ativa (isto é, marcada como inicializável) e lê o primeiro setor desta partição, o **setor de inicialização**[9] da partição (o MBR também é um setor de inicialização, mas tem um nome diferente por executar diferentes funções). O setor de inicialização contém outro pequeno programa que lê a primeira parte do sistema operacional armazenado naquela partição (assumindo que ela seja inicializável, ou seja, que possa iniciar o sistema operacional) e ativa o sistema operacional lá armazenado.

O esquema de particionamento não é definido pelo hardware, muito menos pela BIOS. É somente uma convenção que diversos sistemas operacionais seguem. Os poucos que não a seguem são considerados exceções. Alguns sistemas operacionais suportam partições, mas ocupam uma partição do disco e usam métodos internos de particionamento naquela partição. Esse tipo de partição pode conviver pacificamente com outros sistemas operacionais, inclusive com o Linux, e não requerem medidas especiais, porém um sistema operacional que não suporte partições não poderá coexistir no mesmo disco.

Como medida preventiva é uma boa idéia anotar os dados das partições em um pedaço de papel, assim, mesmo que a tabela de partições esteja corrompida, não se perderão todos os arquivos. (Uma tabela defeituosa pode ser consertada com o

[8] **registro mestre de inicialização**
[9] **boot sector**

`fdisk`). Informações relevante são fornecidas pelo comando `fdisk -l`:

```
$ fdisk -l /dev/hda

Disk /dev/hda: 15 heads, 57 sectors, 790 cylinders
Units = cylinders of 855 * 512 bytes

   Device Boot  Begin   Start     End  Blocks   Id  System
/dev/hda1           1       1      24   10231+  82  Linux swap
/dev/hda2          25      25      48   10260   83  Linux native
/dev/hda3          49      49     408  153900   83  Linux native
/dev/hda4         409     409     790  163305    5  Extended
/dev/hda5         409     409     744  143611+  83  Linux native
/dev/hda6         745     745     790   19636+  83  Linux native
$
```

4.7.2 Partições estendidas e lógicas

O esquema original para discos rígidos dos PCs permite somente quatro partições. Isso rapidamente tornou-se muito reduzido para o atendimento das necessidades do mundo real, principalmente porque as pessoas desejavam utilizar diversos sistemas operacionais (Linux, MS-DOS, OS/2, Minix, FreeBSD, NetBSD, ou Windows/NT, para enumerar alguns), assim como ter diversas partições em um único sistema operacional pode ser uma boa idéia. Por exemplo a área de swap, por questões de performance, deve ter a sua própria partição no Linux, ao invés de utilizar a partição principal.

Para solucionar estes problemas de projeto, foram criadas as **partições estendidas**. Este truque permite que a **partição primária** seja particionada em subpartições. A partição primária subdividida é uma partição estendida, as subpartições são chamadas **partições lógicas**. Elas comportam-se exatamente como partições primárias, mas são criadas de forma diferente, não havendo diferença de velocidade entre elas.

A estrutura de partições de um disco rígido pode aparentar algo similar à figura 4.2. O disco é dividido em três partições primárias, a segunda das quais é dividida em duas partições lógicas. Parte do disco não está particionado. O disco como um todo e cada partição primária tem um setor de inicialização.

4.7.3 Tipos de Partições

As tabelas de partições (uma no MBR e outras para as partições estendidas) contêm um byte que identifica o tipo da partição. É uma tentativa de identificar o sistema

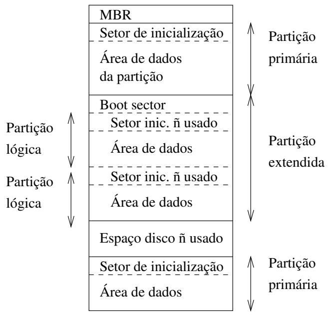

Figura 4.2: Exemplo de particionamento de disco.

operacional que utiliza aquela partição ou qual é a sua finalidade. O propósito é evitar que sistemas operacionais utilizem inadvertidamente a mesma partição. Na verdade, os sistemas operacionais não se importam com o byte de tipo de partição (o Linux na verdade nem toma conhecimento dele). Pior, alguns deles o utilizam incorretamente como algumas versões do DR-DOS.

Não há um padrão internacional para especificar o que cada byte significa, mas os mais comuns e aceitos estão descritos na tabela 4.1. A mesma lista está disponível no programa `fdisk` do Linux.

Tabela 4.1: Tipos de partição (extraído do programa `fdisk`).

0	Empty	40	Venix 80286	94	Amoeba BBT
1	DOS 12-bit FAT	51	Novell?	a5	BSD/386
2	XENIX root	52	Microport	b7	BSDI fs
3	XENIX usr	63	GNU HURD	b8	BSDI swap
4	DOS 16-bit <32M	64	Novell	c7	Syrinx
5	Extended	75	PC/IX	db	CP/M
6	DOS 16-bit ≥32M	80	Old MINIX	e1	DOS access
7	OS/2 HPFS	81	Linux/MINIX	e3	DOS R/O
8	AIX	82	Linux swap	f2	DOS secondary
9	AIX bootable	83	Linux native	ff	BBT
a	OS/2 Boot Manag	93	Amoeba		

4.7.4 Particionando um disco rígido

Há muitos programas para criação e remoção de partições. A maioria dos sistemas operacionais tem o seu próprio, e pode ser uma boa idéia usá-los, para evitar situações não suportadas pelos outros. Muitos programas são chamados `fdisk`, inclusive o programa do Linux, ou suas variações. Detalhes sobre a utilização do `fdisk` são apresentadas na sua página de manual ou no Apêndice E. O comando `cfdisk` é similar ao `fdisk` porém mais agradável, por possuir uma interface mais amigável ao usuário. No Conectiva Linux pode ainda ser encontrado o `DiskDruid`, um particionador extremamente poderoso no momento da instalação do sistema.

Ao utilizar discos IDE, a partição de inicialização (aquela que contém os arquivos de imagem do kernel do sistema) deve residir inteiramente nos primeiros 1024 cilindros. Isso deve-se ao fato que o disco é usado através da BIOS na inicialização do sistema (antes do sistema entrar em modo protegido) e a BIOS não pode gerenciar mais que 1024 cilindros. Pode-se ainda usar uma partição que esteja somente parcialmente nos 1024 cilindros iniciais, desde que todos os arquivos necessários à inicialização do sistema estejam dentro do limite da BIOS. Porém esta é uma prática *desaconselhada* pois novas compilações do kernel ou fragmentações do disco podem colocar os arquivos fora deste limite. De qualquer forma há que se estar seguro de que os arquivos necessários à inicialização do sistema estejam nos primeiros 1024 cilindros.

Algumas versões novas de BIOS e de discos IDE conseguem efetivamente gerenciar discos com mais de 1024 cilindros. Caso se tenha um sistema desses disponível, este problema não existirá e os arquivos de inicialização podem estar em qualquer ponto do disco.

Cada partição deve ter um número par de setores, uma vez que os sistemas de arquivos do Linux usam blocos de 1 Kb de tamanho, ou seja dois setores. Um número ímpar resultará na não utilização do último setor. Isso não é necessariamente um problema, porém é bastante desagradável e algumas versões do `fdisk` podem alertar sobre o problema.

Alterar o tamanho de uma partição normalmente requer que seja efetuada uma cópia de segurança de tudo que se deseja salvar e que esteja presente na partição (preferencialmente do disco todo como medida preventiva), apagá-la, criar uma nova, e restaurar os arquivos na nova partição. Se a partição estiver crescendo, pode-se, também, ter que ajustar o tamanho de partições adjacentes (assim como das cópias de segurança).

Uma vez que alterar o tamanho das partições é bastante trabalhoso, é preferível têlas bem dimensionadas desde o início, ou ter-se um eficiente sistema de cópias de segurança implementado. Caso a instalação seja a partir de uma mídia que não requeira muita intervenção humana (CD-ROM comparado com disquetes), é fácil testar diferentes configurações num primeiro momento, já que não há dados para serem copiados, tornando menos trabalhoso modificar o tamanho das partições diversas vezes.

Há um programa para o MS-DOS chamado `fips`, que redimensiona uma partição FAT/FAT32 sem que seja necessário reinstalar o sistema.

4.7.5 Dispositivos e partições

Cada partição e partição estendida tem o seu próprio arquivo de dispositivo. A convenção da nomenclatura destes arquivos é formada por um número adicionado após o nome do disco, variando de 1 a 4 para partições primárias (independentes de quantas elas sejam) e de 5 a 8 para as partições lógicas (independente de onde elas residam). Por exemplo o `/dev/hda1` é a primeira partição primária no primeiro disco IDE e `/dev/sdb7` é a terceira partição estendida no segundo disco SCSI. A lista de dispositivos em `device-list` traz mais informações.

4.8 Sistema de arquivos

4.8.1 O que é um sistema de arquivos?

Um **sistema de arquivos** é o método e a estrutura de dados que um sistema operacional utiliza para administrar arquivos em um disco ou partição, ou seja, a forma pela qual os arquivos estão organizados em um disco. A expressão também é utilizada para se referenciar a uma partição ou disco que seja usado para armazenar os arquivos ou outros tipos de sistemas de arquivos. Alguém pode dizer "eu tenho dois sistemas de arquivos", significando que tem duas partições nas quais armazena arquivos ou aquela pessoa está usando o "sistema de arquivos estendido", exemplificando o tipo do sistema de arquivos.

A diferença entre um disco ou partição e um sistema de arquivos é bastante significativa. Poucos programas (inclusive os programas que criam sistemas de arquivos) operam diretamente em setores não inicializados de um disco ou partição, e caso exista um sistema de arquivos ele será destruído ou danificado seriamente. A maioria dos programas trabalham em um sistema de arquivos e não funcionam em uma partição que não contenha um (ou que contenha um de tipo errado).

Antes de uma partição ou disco ser usado como um sistema de arquivos ele necessita ser inicializado, e a estrutura básica de dados necessita ser gravada no disco. Este processo é chamado **criação de um sistema de arquivos**.

Muitos sistemas de arquivos do UNIX tem uma estrutura similar, apesar dos detalhes exatos variarem um pouco. Os conceitos básicos são **superbloco**, **inode**, **bloco de dados**, **bloco de diretórios** e **bloco de indireção**. O superbloco contém as informações sobre o sistema de arquivos como um todo, como por exemplo seu tamanho (a informação exata depende do sistema de arquivos). Um inode contém as informações sobre um determinado arquivo, exceto seu nome. O nome está armazenado no diretório, junto com o número do inode. Uma entrada de diretório é formada pelo nome e pelo número do inode que representa o arquivo. O inode contém o número de diversos blocos de dados usados para armazenar as informações do arquivo. Há espaço somente para uns poucos números de blocos de dados no inode, e, caso um número maior seja necessário, mais espaço para ponteiros será alocado dinamicamente. Estes blocos alocados dinamicamente são blocos de indireção que, como o nome indica, contêm endereços para outros blocos.

Sistemas de arquivos UNIX normalmente permitem a criação de **espaços vazios** em um arquivo (isso é feito com o comando `lseek`), o que significa que o sistema de arquivos simplesmente 'simula' que uma determinada parte do arquivo possui somente bytes vazios, mas nenhum setor de disco é reservado (o que significa que o arquivo usará um pouco menos de espaço em disco). Isso ocorre freqüentemente para pequenos binários, bibliotecas compartilhadas, algumas bases de dados e outros casos especiais. Espaços são implementados através do armazenamento de valores especiais como endereços de blocos de dados no bloco de indireção ou no inode. Este endereço especial significa que não há blocos de dados realmente alocados para aquela parte do arquivo, ou seja há um 'buraco' no arquivo.

Espaços vazios são razoavelmente úteis. No sistema do autor, uma medição mostrou uma economia de pelo menos 4 Mb de espaço em disco, de um total de 200 Mb utilizado com o sistema. Veja que este sistema contém relativamente poucos programas e não possui bases de dados. A ferramenta de medição está descrita no Apêndice A.

4.8.2 Diversidade de sistema de arquivos

O Linux suporta diversos tipos de sistemas de arquivos. Dentre esses destacamos:

minix
: O mais antigo e presumivelmente o mais confiável, mas bastante limitado em características (algumas datas não aparecem , máximo de 30 caracteres para nome de arquivos, etc...) e restrito em armazenamento (no máximo 64 Mb por sistema de arquivos).

xia
: Uma versão modificada do sistema de arquivos minix, o qual aumenta os limite de nomes de arquivos e de sistemas de arquivos, mas não introduz novas facilidades. Não é muito popular, mas comenta-se que funcione muito bem.

ext2
: O mais poderoso e popular sistema de arquivos nativo do Linux. Desenhado para ser facilmente compatível com os avanços das novas versões, sem a necessidade de criar novamente os sistemas de arquivos já existentes.

ext
: Uma versão antiga do `ext2` que não é mais compatível com versões atuais. É raro vê-la instalada em sistemas novos e mesmo os mais antigos têm sido convertidos para `ext2`.

Adicionalmente há o suporte a diversos outros sistemas de arquivos, para simplificar a troca de informações com outros sistemas operacionais. Estes sistemas de arquivos funcionam como se fossem nativos, exceto pela perda de algumas facilidades presentes no UNIX, ou apresentam algumas particularidades.

msdos	Compatibilidade com MS-DOS (e OS/2 e Windows NT) através de sistemas de arquivos FAT/FAT32.
umsdos	Sistemas de arquivos MS-DOS estendidos para suportar nomes longos, donos, permissões, links e arquivos de dispositivos do Linux. Isso permite que um sistema de arquivos `msdos` possa ser usado como se fosse um sistema Linux, removendo a necessidade de uma partição distinta para o Linux.
iso9660	O sistema de arquivos padrão do CD-ROM. A extensão Rock Ridge que permite nomes longos também é suportada automaticamente.
nfs	Sistemas de arquivos em redes que permitem o compartilhamento e o fácil acesso aos arquivos entre diversos computadores da rede.
hpfs	O sistema de arquivos do OS/2.
sysv	Sistema de arquivos do System V/386, Coherent e Xenix.

A opção do sistema de arquivos a ser usado depende da situação. Caso a compatibilidade ou outras razões tornem um dos sistemas de arquivos não nativos necessário, então este deve ser utilizado. Caso a opção seja livre, então provavelmente a decisão mais acertada seja usar o ext2, uma vez que ele traz diversas facilidades sem sofrer perda de performance.

Há ainda o sistema de arquivos `proc`, normalmente acessível através do diretório `/proc`, o qual não é um sistema de arquivos real, apesar de parecer um. O sistema de arquivos `proc` torna mais simples o acesso a determinadas estruturas do kernel, como por exemplo a lista de processos. Isso torna a estrutura de dados parecer um sistema de arquivos que pode ser manipulado com as ferramentas usuais. Por exemplo para obter uma lista de todos os processos pode-se usar o comando:

```
$  ls -l /proc
total 0
dr-xr-xr-x   4 root     root           0 Jan 31 20:37 1
dr-xr-xr-x   4 liw      users          0 Jan 31 20:37 63
dr-xr-xr-x   4 liw      users          0 Jan 31 20:37 94
dr-xr-xr-x   4 liw      users          0 Jan 31 20:37 95
dr-xr-xr-x   4 root     users          0 Jan 31 20:37 98
dr-xr-xr-x   4 liw      users          0 Jan 31 20:37 99
```

```
-r--r--r--   1 root     root            0 Jan 31 20:37 devices
-r--r--r--   1 root     root            0 Jan 31 20:37 dma
-r--r--r--   1 root     root            0 Jan 31 20:37 filesystems
-r--r--r--   1 root     root            0 Jan 31 20:37 interrupts
-r--------   1 root     root      8654848 Jan 31 20:37 kcore
-r--r--r--   1 root     root            0 Jan 31 11:50 kmsg
-r--r--r--   1 root     root            0 Jan 31 20:37 ksyms
-r--r--r--   1 root     root            0 Jan 31 11:51 loadavg
-r--r--r--   1 root     root            0 Jan 31 20:37 meminfo
-r--r--r--   1 root     root            0 Jan 31 20:37 modules
dr-xr-xr-x   2 root     root            0 Jan 31 20:37 net
dr-xr-xr-x   4 root     root            0 Jan 31 20:37 self
-r--r--r--   1 root     root            0 Jan 31 20:37 stat
-r--r--r--   1 root     root            0 Jan 31 20:37 uptime
-r--r--r--   1 root     root            0 Jan 31 20:37 version
$
```

(Podem haver alguns arquivos a mais que não correspondem a processos, porém o exemplo acima foi resumido por questões didáticas).

Note que apesar de ser chamado como sistema de arquivos, o `proc` não acessa o disco rígido. Ele existe somente no kernel do sistema. Toda vez que alguém tente acessar alguma parte do sistema de arquivos `proc`, o kernel torna esta parte visível em algum lugar. Então mesmo que pareça existir um arquivo com diversos megabytes denominado `/proc/kcore`, ele não utiliza um único byte do disco rígido.

4.8.3 Qual sistema de arquivos deve ser usado?

Faz pouco sentido utilizar muitos sistemas de arquivos diferentes. Atualmente o ext2fs é o mais popular, e provavelmente a decisão mais acertada. Dependendo da necessidade de recursos adicionais na manutenção das estruturas, velocidade, segurança, estabilidade, compatibilidade e diversas outras razões, pode ser desejável utilizar outro tipo de sistema de arquivos. Isto deve ser decidido caso a caso.

4.8.4 Criando um sistema de arquivos

Sistemas de arquivos são criados, ou melhor inicializados, através do comando `mkfs`. Na verdade há um comando específico para cada tipo de sistema de arquivos. O `mkfs` é somente uma interface que executa o programa adequado ao tipo desejado. O tipo é selecionado através da opção `-t`.

Os programas acionados pelo `mkfs` podem ter uma interface de linha de comando ligeiramente diferente. As opções comuns e mais importantes estão resumidas a seguir (veja a página de manual ou consulte o Apêndice E).

`-t` *tipo_sa* Seleciona o tipo do sistema de arquivos

`-c` Pesquisa blocos defeituosos e inicializa a lista de blocos ruins

`-l` *nome arquivo* Lê a lista de blocos ruins a partir do arquivo informado

Para criar um sistema de arquivos do tipo ext2 em um disquete, basta executar o comando:

```
$ fdformat -n /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
$ badblocks /dev/fd0H1440 1440 > bad-blocks
$ mkfs -t ext2 -l bad-blocks /dev/fd0H1440
mke2fs 1.12, 9-Jul-98 for EXT2 FS 0.5b, 95/08/09
Linux ext2 filesystem format
Filesystem label=
360 inodes, 1440 blocks
72 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
1 block group
8192 blocks per group, 8192 fragments per group
360 inodes per group

Writing inode tables: done
Writing superblocks and filesystem accounting information: done
$
```

Inicialmente o disquete foi formatado (a opção `-n` inibe a checagem de blocos defeituosos). Após o comando `badblocks` faz a checagem e redireciona as mensagens de saída para o arquivo **bad_blocks**. Finalmente o sistema de arquivos é criado, com a lista de blocos defeituosos inicializada com qualquer coisa que o comando `badblocks` tenha encontrado.

A opção `-c` poderia ter sido utilizada em conjunto com o comando `mkfs`, ao invés do `badblocks` e de um arquivo em separado, conforme o exemplo a seguir:

```
$ mkfs -t ext2 -c /dev/fd0H1440
mke2fs 1.12, 9-Jul-98 for EXT2 FS 0.5b, 95/08/09
Linux ext2 filesystem format
Filesystem label=
360 inodes, 1440 blocks
72 blocks (5.00%) reserved for the super user
First data block=1
```

```
Block size=1024 (log=0)
Fragment size=1024 (log=0)
1 block group
8192 blocks per group, 8192 fragments per group
360 inodes per group

Checking for bad blocks (read-only test): done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
$
```

A opção `-c` é mais conveniente que o uso do comando `badblocks` em separado, porém este último é necessário para uma verificação mais precisa, após a criação do sistema de arquivos.

O processo de preparar sistemas de arquivos em discos rígidos ou partições é o mesmo para disquetes, exceto pelo fato da formatação não ser necessária.

4.8.5 Montando e desmontando

Antes de um sistema de arquivos poder ser utilizado, ele necessita ser **montado**. O sistema operacional executa diversas verificações para estar seguro de que tudo está funcionando bem. Uma vez que todos os arquivos no UNIX estão em uma única árvore de diretórios, a operação de montagem fará com que o novo sistema de arquivos pareça um subdiretório existente em algum sistema de arquivos já montado.

Por exemplo, a figura 4.3 mostra três diferentes sistemas de arquivos, cada qual com o seu próprio diretório raiz. Quando os dois últimos são montados sob o `/home` e `/usr`, respectivamente, no primeiro sistema de arquivos, temos somente uma única árvore de diretórios e suas entradas, conforme a figura 4.4.

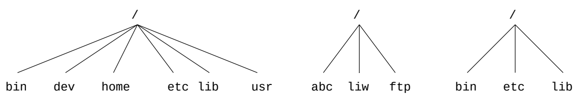

Figura 4.3: Três sistemas de arquivos distintos.

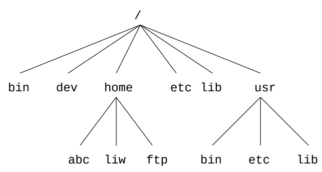

Figura 4.4: `/home` e `/usr` foram montados.

As montagens podem ser executadas, por exemplo, da seguinte forma:

$ *mount /dev/hda2 /home*

$ *mount /dev/hda3 /usr*

$

O `mount` tem dois argumentos. O primeiro é o arquivo de dispositivo correspondente ao disco ou partição que contenha o sistema de arquivos. O segundo é o diretório sob o qual ele será montado. Após este comando o conteúdo dos dois sistemas de arquivos parecerão simplesmente dois diretórios: `/home` e `/usr` respectivamente. Pode-se então dizer que "`/dev/hda2` **está montado no** `/home`" e similarmente para o `/usr`. Para examinar estes sistemas de arquivos, pode-se acessar estes diretórios exatamente da mesma forma que qualquer outro. É importante ressaltar a diferença entre o dispositivo `/dev/hda2` e o diretório montado `/home`. Enquanto o primeiro dá acesso ao dados brutos do disco, o segundo permite o acesso aos arquivos contidos no mesmo disco. O diretório montado é chamado **ponto de montagem**.

O Linux suporta diversos tipos de sistemas de arquivos. O `mount` tenta verificar qual o tipo de sistema de arquivos que está sendo montado. Pode-se utilizar opcionalmente a opção `-t` *tipo* para especificar o tipo diretamente. Isso pode ser útil quando a verificação automática não funcionar corretamente. Por exemplo para montar um disquete MS-DOS, pode-se usar o seguinte comando:

$ *mount -t msdos /dev/fd0 /floppy*

$

O diretório montado não necessita estar vazio, na verdade ele somente deve existir. Apesar disso, todos os arquivos abaixo deste diretório estarão inacessíveis enquanto o sistema de arquivos é montado (qualquer arquivo que já tenha sido aberto estará disponível. Arquivos que contêm links com outros diretórios também podem ser acessados através daqueles nomes). Nenhum dano será causado ao sistema de arquivos e isso pode, por vezes, ser bastante útil. Por exemplo caso se deseje ter o `/tmp` e `/var/tmp` como sinônimos, faz-se uma ligação simbólica do `/var/tmp` para `/tmp`. Quando o sistema é inicializado, antes que o sistema de arquivos `/usr` seja montado, o diretório /var/tmp residente no diretório raiz é usado em seu lugar. Quando o `/usr` é montado, ele tornará o diretório `/var/tmp` no raiz indisponível. Caso este diretório não existisse no raiz seria impossível usar arquivos temporários antes de montar o `/var`.

Caso não se pretenda gravar absolutamente nada no sistema de arquivos, pode-se usar o parâmetro `-r` para montar o sistema de arquivos permitindo somente operações de leitura. Isso fará com que o kernel do sistema bloqueie qualquer tentativa de gravação naquele sistema de arquivos. Montagens de leitura somente são necessárias para mídias não graváveis como por exemplo CD-ROMs.

O leitor mais atento perceberá um pequeno problema logístico. Como o primeiro sistema de arquivos (chamado **raiz**, por conter o diretório raiz (/)) é montado, uma vez que obviamente ele não pode ser montado a partir de outros sistemas de arquivos? Isso é feito de maneira automática.[10] O sistema de arquivos raiz é montado automaticamente durante a inicialização do sistema operacional e pode-se estar certo de que ele sempre estará disponível, pois de outra forma o sistema não poderá ser inicializado. O nome do sistema de arquivos que é montado como / é compilado junto com o kernel do sistema, configurado no LILO ou através do comando `rdev`.

O sistema de arquivos raiz normalmente é montado com permissões somente de leitura. Os scripts de inicialização executarão o comando `fsck` para verificar sua integridade e se por ventura há algum problema. Caso tudo corra bem, ele será montado novamente, agora com permissões de gravação. O programa `fsck` não deve ser executado em sistemas de arquivos montados, pois quaisquer alterações no sistema durante a sua execução *causará* problemas. Uma vez que o sistema de arquivos raiz é montado inicialmente no modo somente de leitura, o `fsck` pode ser executado e regularizar qualquer anormalidade encontrada, pois, após a remontagem com permissões de gravação, os eventuais ajustes serão efetuados automaticamente.

Em muitos sistemas, há outros sistemas de arquivos que devem ser montados automaticamente durante a sua inicialização. Estes sistemas de arquivos estão especificados no arquivo `/etc/fstab` (veja a página de manual do *fstab* ou Apêndice E). Os detalhes exatos de como cada sistema de arquivos será montado depende de muitos fatores e pode ser efetuado de acordo com a necessidade de cada administrador.

Quando um sistema de arquivos não necessita mais estar montado, ele pode ser desmontado através do comando `umount`[11]. Este comando necessita um argumento: ou o arquivo de dispositivo ou o ponto de montagem. Por exemplo para desmontar os diretórios do exemplo anterior deve-se executar:

```
$ umount /dev/hda2
$ umount /usr
$
```

Para maiores informações sobre este comando, leia a página de manual. É fundamental que um disquete sempre seja desmontado. *Não se deve simplesmente retirar o disquete da unidade*! Dados podem estar em memória e ainda não terem sido gravados. Isso ocorre após o sistema executar um `sync`[12] Retirá-lo antes pode danificar o conteúdo do disquete. Caso se esteja somente lendo do disquete, pode não ser tão crítico, mas caso se esteja gravando algum conteúdo o resultado pode ser catastrófico.

Montar e desmontar sistemas de arquivos requer privilégios de superusuário, isto é, somente o usuário `root` pode fazê-lo. A razão para isso é que se qualquer usuário

[10] Para mais informações, veja os fontes do kernel ou o Kernel Hackers'Guide

[11] O comando deveria ser `unmount`, mas o `n` desapareceu misteriosamente na década de 70 e não foi encontrado desde então. Por favor, devolva-o para a Bell Labs, NJ, se encontrá-lo.

[12] O `umount` encarrega-se de executá-lo

pudesse montar um disquete como um sistema de arquivos, seria fácil criar um Cavalo de Tróia disfarçado como `/bin/sh`, ou outro programa qualquer. Ocasionalmente os usuários precisam utilizar disquetes. Seguem algumas formas de resolver isto:

- Dar aos usuários a senha do `superusuário`. Esta é obviamente a pior solução, mas é a mais simples. Deve funcionar bem caso não haja outras necessidades de segurança no equipamento, no caso de máquinas fora da rede ou sistemas de uso pessoal.
- Usar um programa como o `sudo` que permite aos usuários utilizar o comando `mount`. Ainda não é uma solução segura, mas não dá privilégios diretos de superusuário a todos os usuários[13].
- Indicar o uso das ferramentas `mtools`, um pacote de manipulação de sistemas de arquivos DOS, sem ter que montá-los. Isso funciona bem para arquivos MS-DOS, mas ainda não é a solução ideal.
- Listar as unidades de disco flexível e os pontos de montagem permitidos no arquivo `/etc/fstab`, juntamente com as opções corretas.

Esta última alternativa pode ser implementada adicionando-se uma linha similar à seguinte no `/etc/fstab`:

```
/dev/fd0 /floppy msdos user,noauto 0 0
```

As colunas são: dispositivo a ser montado, diretório de montagem, tipo do sistema de arquivos, opções, freqüência da realização de cópias de segurança (usado pelo `dump`), e número de passagem do `fsck` (especifica a ordem em que o sistema de arquivos deve ser verificado após a inicialização. 0 significa sem checagem).

A opção `noauto` inibe a montagem automática na inicialização do sistema (ou seja inibe a ação do comando `mount -a` no arquivo fstab). A opção `user` permite que qualquer usuário possa montar o sistema de arquivos e, por razões de segurança, desabilita a execução de programas (em modo normal ou através de setuid) e a interpretação de dispositivos a partir do sistema de arquivos montado. Após isso, qualquer usuário pode montar um disquete com um sistema de arquivos DOS utilizando o comando:

```
$ mount /floppy
$
```

O disquete pode (e deve) ser desmontado com o comando `umount`.

Caso queira ter acesso a diferentes tipos de disquetes, é necessário ter diversos pontos de montagem. Os parâmetros podem ser diferentes para cada um. Para dar acesso a disquetes DOS e ext2, pode-se configurar o `/etc/fstab` da seguinte maneira:

[13] São necessários diversos segundos até que o usuário ache um jeito de burlar a segurança do sistema

```
/dev/fd0    /dosfloppy    msdos   user,noauto  0  0
/dev/fd0    /ext2floppy   ext2    user,noauto  0  0
```

Para sistemas de arquivos DOS (não somente disquetes), provavelmente queira-se restringir o acesso usando opções de sistemas de arquivos como `uid`, `gid` e `umask` descritos na página de manual do comando `mount`. Caso estas precauções não sejam tomadas, qualquer usuário poderá no mínimo ler todos os arquivos DOS daquele sistema de arquivos, o que não parece ser uma boa idéia.

4.8.6 Verificando a integridade de um sistema de arquivos com `fsck`

Sistemas de arquivos são criaturas complexas, e tendem, com o tempo ou o mau uso, a ter erros. A validação e correção de erros de um sistema de arquivos podem ser realizadas através do comando `fsck`. Ele pode ser configurado para resolver alguns tipos de problemas que encontre e para alertar o usuário sobre erros incorrigíveis. Felizmente os programas que implementam os sistemas de arquivos são muito testados e é raro ter-se problemas desta natureza. Os erros comuns são ocorrências causadas por falhas de energia, falhas de hardware ou erros de operação.

Muitos sistemas estão configurados para executar o `fsck` automaticamente no momento de sua inicialização, assim vários erros podem ser encontrados e corrigidos desta forma, antes da utilização do sistema. O uso de sistemas de arquivos corrompidos tende a piorar o problema: se as estruturas de dados contêm erros, utilizar o sistema de arquivos pode provocar uma perda ainda maior. O `fsck` pode demorar ao executar em grandes sistemas de arquivos e pelo fato de que problemas raramente ocorrem quando se desliga a máquina apropriadamente, alguns truques são usados para evitar a checagem nestes casos. O primeiro é baseado na existência do arquivo `/fastboot`; caso ele exista, nenhuma checagem é realizada. O segundo consiste em utilizar o `e2fsck` (a versão do `fsck` para sistemas de arquivos do tipo ext2). Os sistemas de arquivos ext2 têm uma marca especial no superbloco, a qual informa se o sistema foi desligado de maneira apropriada. Caso positivo, o programa não checará todo o sistema de arquivos. O uso do `/fastboot` depende dos scripts de inicialização do sistema. O `e2fsck` não irá validar o sistema de arquivos, desde que o sistema tenha sido desligado corretamente. Para forçar sua validação, utilize a opção `-f` do `e2fsck`. Veja a página de manual do `e2fsck` ou o Apêndice E.

A checagem automática funciona somente para os sistemas de arquivos que são montados automaticamente na inicialização do sistema. Para os demais, o programa `fsck` deve ser executado manualmente.

Caso o `fsck` encontre problemas irreparáveis, será necessário um conhecimento profundo de como os sistemas de arquivos funcionam e em especial do tipo de sistema corrompido. Outra alternativa muito saudável é ter cópias de segurança do sistema!

A segunda opção é mais simples de ser posta em prática, já a primeira pode depender de pesquisas em grupos de notícias, listas de discussão e serviços de suporte, caso você não tenha o conhecimento necessário. Eu gostaria de falar mais a você sobre este assunto, mas meus conhecimentos neste tópico não são muito profundos. O programa `debugfs` de Theodore T'so pode ajudar nesta tarefa.

O `fsck` deve ser executado somente em sistemas de arquivos desmontados e nunca naqueles que já estejam montados (exceto o sistema de arquivos raiz, montado com permissões somente de leitura, durante a inicialização do sistema). Isso deve-se ao fato do `fsck` acessar diretamente os dados do sistema, podendo inclusive modificá-los sem que o sistema operacional perceba. Você terá sérios problemas, caso o sistema operacional fique confuso.

4.8.7 Checando erros de disco com `badblocks`

É aconselhável checar periodicamente o disco rígido. Isto poder ser feito através da execução do comando `badblocks`. Ele gera uma lista com os números dos blocos defeituosos encontrados. Esta lista pode servir de entrada para o comando `fsck`. O `fsck` grava nas estruturas de dados do sistema de arquivos os blocos que tenham problemas, assim, o sistema não utilizará estas áreas para armazenar dados. O seguinte exemplo mostra como isso pode ser feito:

```
$ badblocks /dev/fd0H1440 1440 > bad-blocks
$ fsck -t ext2 -l bad-blocks /dev/fd0H1440
Parallelizing fsck version 1.12 (9-Jul-98)
e2fsck 1.12, 9-Jul-98 para o sistema de arquivos EXT2 0.5b, 95/08/09
Pass 1: Checking inodes, blocos, and sizes
Pass 2: Checking diretório structure
Pass 3: Checking diretório connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information

/dev/fd0H1440: ***** SISTEMA DE ARQUIVOS FOI MODIFICADO *****
/dev/fd0H1440: 11/360 arquivos (0.0% não-contíguos), 63/1440 blocos
$
```

Caso o `badblocks` ache um bloco defeituoso que já foi utilizado pelo sistema, o `e2fsck` tentará mover o bloco para outro local. Caso o problema seja realmente grave e não somente um erro superficial, o arquivo poderá estar definitivamente corrompido.

4.8.8 Evitando fragmentação

Quando um arquivo é gravado em disco, ele nem sempre pode ser gravado em blocos consecutivos. Um arquivo que não esteja armazenado em blocos consecutivos estará **fragmentado**. O tempo de leitura é maior para arquivos fragmentados, uma vez que as cabeças do disco rígido terão que se movimentar mais. É desejável evitar a fragmentação, apesar de não ser exatamente um problema quando se utiliza um bom buffer de dados de leitura do tipo read-ahead.

Os sistemas de arquivos ext2 tentam manter a fragmentação no mínimo, mantendo todos os blocos de um arquivo juntos, mesmo que eles não possam ser armazenados em setores consecutivos. O ext2 sempre aloca o bloco livre mais próximo dos demais blocos do arquivo. É raro o ext2 preocupar-se com fragmentação. Há um programa para desfragmentação de sistemas de arquivo do tipo ext2. Veja `ext2-defrag` na bibliografia.

Para sistemas DOS há diversos programas de desfragmentação que movem os blocos dentro do sistema de arquivos, diminuindo o nível de fragmentação. Para outros sistemas de arquivos, a desfragmentação pode ser realizada através da realização de cópias de segurança do sistema de arquivos, recriação do sistema e restauração dos arquivos. Realizar uma cópia de segurança do sistema de arquivos antes de desfragmentá-lo é uma boa medida, uma vez que muitas coisas podem ocorrer durante esse processo.

4.8.9 Outras ferramentas para todos os sistemas de arquivos

Algumas ferramentas são ainda muito úteis no gerenciamento de sistemas de arquivos. O comando `df` apresenta o espaço livre em disco para um ou mais sistemas de arquivos. O comando `du` mostra o espaço ocupado por um diretório e todos os seus arquivos, e pode ser usado para encontrar os grandes usuários de espaço em disco.

O `sync` força a gravação de todos os blocos ainda não gravados e que estejam em memória (cache, veja a seção 5.6). É raro executá-lo manualmente, pois o servidor `update` faz isso automaticamente. O `sync` pode ser útil em catástrofes, no caso do `update` ou o `bdflush` morrerem, ou quando é preciso encerrar o sistema rapidamente.

4.8.10 Outras ferramentas para sistemas de arquivos ext2

Em adição à ferramenta usada para a criação de sistemas de arquivos `mke2fs` e ao verificador `e2fsck` acessados diretamente ou através de uma interface, o sistema de arquivos ext2 tem algumas ferramentas que podem ser de grande utilidade.

O `tune2fs` ajusta os parâmetros do sistema de arquivos. Alguns parâmetros interessantes são:

- Um contador do número de vezes que o sistema de arquivos foi montado. O `e2fsck` força uma checagem toda vez que o sistema seja montado n vezes, mesmo que o indicador de normalidade esteja assinalado. Para um sistema que é usado em desenvolvimento ou testes, pode ser uma boa idéia reduzir este número.
- Tempo máximo entre as verificações. O `e2fsck` pode ainda forçar o tempo máximo entre verificações, mesmo que o indicador de normalidade esteja assinalado e o sistema de arquivos não seja montado freqüentemente. Esta opção pode ser desabilitada, a critério do administrador.
- Número de blocos reservado para o `superusuário`. O ext2 reserva alguns blocos para o `root`, pois, mesmo que o sistema de arquivos esteja cheio, o root ainda pode administrar o filesystems sem apagar nenhum dado. O espaço reservado padrão é de 5%, o qual na maioria dos discos, não representa perda considerável de dados. Note que disquetes não necessitam de blocos reservados.

Veja a página de manual do *tune2fs*.

O comando `dumpe2fs` apresenta algumas informações sobre um sistema de arquivos ext2, a maioria extraída a partir do superbloco. A figura 4.5 apresenta um exemplo. Algumas informações são extremamente técnicas e exigem um certo conhecimento de como o sistema de arquivos funciona, mas muitas das informações são do alcance da maioria dos administradores.

O `debugfs` é um programa destinado à validação de um sistema de arquivos. Ele permite o acesso direto às estruturas de dados armazenadas em disco e pode ser usado para reparar problemas em discos, que talvez nem mesmo o `fsck` pode corrigir automaticamente. Também é conhecido por seu uso na recuperação de arquivos apagados. O `debugfs` exige um alto nível de conhecimento das tarefas que estão sendo executadas; uma falha no entendimento pode destruir todos os dados do seu disco!

Os comandos `dump` e `restore` podem ser usados para geração de cópias de segurança de um sistema de arquivos ext2. Eles são versões específicas para o ext2 de tradicionais ferramentas do UNIX. Veja o capítulo 10 para maiores informações sobre cópias de segurança.

4.9 Discos sem sistemas de arquivos

Nem todos os discos ou partições são usados como sistemas de arquivos. Uma partição swap, por exemplo, não conterá um sistema de arquivos. Muitos disquetes são usados como uma emulação de um dispositivo de fita, podendo o comando `tar` ou outro qualquer gravar os dados diretamente no disco, sem um sistema de arquivos. Os disquetes de inicialização do Linux não contêm um sistema de arquivos, somente a imagem do kernel do sistema.

A não utilização de um sistema de arquivos libera um maior espaço do disco para outras tarefas, uma vez que o sistema de arquivos utiliza estruturas internas para garantir a integridade dos dados. Isso também torna o disco mais compatível com outros sistemas: por exemplo o formato de arquivos gerado pelo `tar` é o mesmo em todos os sistemas, enquanto que os sistemas de arquivos são diferentes na maioria. Um administrador rapidamente acostuma-se ao uso de discos sem sistemas de arquivos, caso necessário. Disquetes inicializáveis do Linux também não têm sistemas de arquivos, apesar de ser possível.

Uma razão para usar discos de forma direta é a de fazer imagens deles. Por exemplo, se um disco tem parte do sistema de arquivos danificada, é uma boa idéia fazer-se uma cópia antes de tentar consertá-lo, podendo o trabalho ser reiniciado caso as tentativas tenham danificado os dados restantes. Uma forma de fazer-se isso é através do comando `dd`:

```
$ dd if=/dev/fd0H1440 of=floppy-image
2880+0 records in
2880+0 records out
$ dd if=floppy-image of=/dev/fd0H1440
2880+0 records in
2880+0 records out
$
```

O primeiro `dd` gera uma imagem exata do disquete para o arquivo `floppy-image`, o segundo grava a imagem para o disquete. (o usuário presumivelmente substituiu o disquete antes do segundo comando. De outra forma o procedimento torna-se inútil).

4.10 Alocando espaço em disco

4.10.1 Esquemas de particionamento

Não é fácil particionar um disco de uma maneira perfeita. Pior, não há uma forma 'certa' de fazê-lo, devido ao número de fatores envolvidos.

O modo tradicional sugere um pequeno sistema de arquivos raiz, que contém os diretórios `/bin`, `/etc`, `/dev`, `/lib`, `/tmp` e outros necessários para o sistema ser inicializado e para que ele permaneça ativo. Desta forma o sistema de arquivos raiz (na sua própria partição ou no seu próprio disco) contém todo o necessário para manter o sistema ativo. A razão para isso é que se o sistema de arquivos raiz é pequeno e pouco usado, a chance dele ter problemas é muito pequena e caso aconteçam problemas com o sistema de arquivos, eles são sanados mais rapidamente. Após, criam-se partições separadas para a árvore de diretórios sob o `/usr`, para os diretórios pessoais (normalmente sob o `/home`) e para a área de swap. Separando-se a área de diretórios pessoais em sua própria partição torna as cópias de segurança mais simples, uma vez que não é necessário copiar os programas (que estão normalmente no `/usr`). Em um ambiente de rede é possível ainda compartilhar o `/usr` entre diversas máquinas (usando NFS por exemplo), reduzindo-se assim o espaço total de disco requerido entre todas as máquinas, gerando economia de dezenas ou centenas de megabytes, dependendo do número de máquinas.

O problema de ter-se muitas partições é que elas dividem o espaço livre total do disco em partes menores. Hoje, que discos e sistemas operacionais são mais confiáveis, muitos preferem ter somente uma partição com todos os arquivos. Por outro lado, é mais fácil fazer cópias de segurança (e restaurações) de partições pequenas.

Para um disco pequeno (assumindo que você não usa os fontes do kernel), a melhor solução é provavelmente ter somente uma partição. Para discos grandes, é preferível ter-se poucas partições grandes, para os casos em que algo de anormal ocorra. Note que 'pequenas' ou 'grandes' são usadas aqui em um sentido relativo, já que dependem das necessidades e dos equipamentos disponíveis.

Caso haja vários discos disponíveis é aconselhável ter-se o sistema de arquivos raiz (inclusive o `/usr`) em um disco e os diretórios pessoais em outro.

É interessante ainda estar preparado para experimentar diferentes tipos de particionamento. Pode ser bastante trabalhoso instalar um sistema desde seu início diversas vezes, mas desta maneira você acabará sabendo que está fazendo a coisa certa.

4.10.2 Requisitos de espaço

A distribuição Linux que você instalar trará algumas indicações do espaço necessário para sua instalação. Programas instalados em separado provavelmente façam a mesma coisa. Isso ajudará no planejamento da utilização do espaço em disco, porém deve-se estar preparado para o futuro, reservando-se algum espaço extra para as demandas que surgirão.

A quantidade de espaço necessária para os usuários depende das tarefas que eles executarão. Algumas pessoas fazem somente edição de textos e necessitam de alguns megabytes, outros podem fazer processamento de imagens e utilizarem gigabytes.

Por oportuno ao comparar tamanhos de arquivos em kilobytes e megabytes e espaço em disco dado em megabytes, é importante saber que estas unidades podem ser diferentes. Alguns fabricantes de discos deixam a entender que um kilobyte é igual a 1.000 bytes e um megabyte igual a 1.000 kilobytes, enquanto o resto do mundo usa 1.024 em ambos os fatores. Em assim sendo um disco de aparentemente 345 Mb, tem na verdade 330 Mb disponíveis.

A alocação de espaço em disco para a área de swap (troca) é discutida na seção 5.5.

4.10.3 Exemplos de alocação de espaço em disco

Eu estou acostumado a um disco rígido de 109 Mb, porém tenho agora um disco de 330 Mb, e apresentarei como este disco está particionado [14].

Particionei o disco de 109 Mb de diversas formas à medida que minhas necessidades e o sistema operacional que eu usava foram mudando. Apresentarei dois cenários básicos. Primeiro, executando o MS-DOS em conjunto com o Linux. Para isso precisei de 20 Mb para conter, no mínimo, o MS-DOS, um compilador C, um editor, alguns utilitários, um programa em desenvolvimento, e algum espaço em disco livre para não me sentir claustrofóbico. Para o Linux foi reservado um espaço de 10 Mb para a área de troca e os 79 Mb restantes, foram alocados em uma única partição com todos os arquivos do Linux. Foi experimentado o particionamento em raiz, `/usr` e `/home`, mas como havia pouco espaço disponível em disco, uma única partição tornou-se a opção mais interessante.

Quando o DOS não era mais necessário, o disco foi reparticionado, e foi criada uma área de *swap* de 12 Mb, e novamente o espaço restante foi alocado em uma única partição.

O disco de 330 Mb foi particionado em diversas partições:

5 MB	sistema de arquivos raiz
10 MB	partição de swap
180 MB	sistema de arquivos `/usr`
120 MB	sistema de arquivos `/home`
15 MB	partição de testes

A partição de testes está designada para atividades que requerem uma partição própria, como por exemplo, testar diferentes distribuições do Linux ou comparar a velocidade de diferentes sistemas de arquivos. Quando não preciso de nada especial, ela é utilizada como *swap* (eu gosto de ter *muitas* janelas abertas).

[14] atualmente os discos rígidos são muito maiores, mas as *proporções* entre os espaços permanecem as mesmas

4.10.4 Adicionando mais espaço em disco ao Linux

Adicionar mais espaço ao Linux é simples, ao menos após o hardware ter sido adequadamente instalado (a instalação do hardware está além do escopo deste livro). Basta formatar, criar as partições e os sistemas de arquivos conforme descrito anteriormente, e adicionar as linhas adequadas ao `/etc/fstab` para a montagem automática dos novos sistemas de arquivos.

4.10.5 Dicas para economizar espaço

A melhor dica para economizar espaço em disco é evitar a instalação de programas desnecessários. Muitas distribuições do Linux têm opções de instalação parcial do sistema. Verifique os pacotes disponíveis, e analise as suas necessidades. Você irá perceber que não necessita de muitos deles (talvez a maioria). Isso ajudará a salvar muito espaço em disco, já que muitos programas são grandes. Mesmo que você necessite de um determinado programa, talvez somente alguns componentes já sejam suficientes. Por exemplo, alguma documentação on-line pode ser desnecessária, assim como alguns dos arquivos Elisp para o GNU Emacs, alguns fontes do X11, ou bibliotecas de programação.

Caso não seja possível desinstalar algum pacote, você pode pensar em compactá-los. Programas de compactação como `gzip` ou `zip` irão compactar (e descompactar) arquivos em grupo ou individuais. O programa `gzexe` irá compactar e descompactar programas automaticamente para o usuário (programas sem uso são compactados, e descompactados quando utilizados). O programa experimental `DouBle` irá compactar todos os arquivos do sistema de arquivos, automaticamente e de forma transparente para os usuários e programas que os utilizem (princípio similar ao produto Stacker para MS-DOS, por exemplo).

```
dumpe2fs 1.12, 9-Jul-98 for EXT2 FS 0.5b, 95/08/09
Filesystem volume name:   <none>
Last mounted on:          <not available>
Filesystem UUID:          ce3ad49a-1ce7-11d2-8afa-000021a28984
Filesystem magic number:  0xEF53
Filesystem revision #:    0 (original)
Filesystem features:      (none)
Filesystem state:         not clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              78312
Block count:              313236
Reserved block count:     15661
Free blocks:              85681
Free inodes:              78270
First block:              1
Block size:               1024
Fragment size:            1024
Blocks per group:         8192
Fragments per group:      8192
Inodes per group:         2008
Inode blocks per group:   251
Last mount time:          Sun Sep 27 11:26:23 1998
Last write time:          Tue Sep 29 00:11:23 1998
Mount count:              1
Maximum mount count:      20
Last checked:             Sun Sep 27 11:25:52 1998
Check interval:           15552000 (6 months)
Next check after:         Fri Mar 26 11:25:52 1999
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)


Group 0: (Blocks 1 -- 8192)
  Block bitmap at 4 (+3), Inode bitmap at 5 (+4)
  Inode table at 6 (+5)
  7923 free blocks, 1997 free inodes, 2 directories
  Free blocks: 270-8192
  Free inodes: 12-2008
```

Figura 4.5: Exemplo da saída do programa `dumpe2fs`

Capítulo 5

Gerenciamento de Memória

Minnet, jag har tappat mitt minne,
är jag svensk eller finne
kommer inte ihåg...
(Bosse Österberg)

Esta seção descreve as características do gerenciamento de memória do Linux, isto é, a memória virtual e o cache de disco. Os aspectos e considerações que o administrador de sistemas necessita estão descritos neste capítulo.

5.1 O que é memória virtual?

O Linux tem suporte a **memória virtual**, isto é, ele utiliza o disco como extensão da memória RAM, fazendo com que o tamanho de memória disponível cresça consideravelmente. O kernel irá gravar o conteúdo de blocos de memória sem uso no disco rígido, podendo aquela área agora ser utilizada para outros propósitos. Quando o conteúdo original faz-se necessário, eles são lidos e colocados novamente na memória. Isso é feito de maneira totalmente transparente para o usuário. Programas sendo executados sob o Linux somente vêem a quantidade total de memória disponível, sem perceber que partes dela residem no disco de tempos em tempos. Obviamente, ler e gravar em disco é muito mais lento (na ordem de 1.000 vezes mais lento) que utilizar memória real, tornando portando os programas também mais lentos. A parte do disco que é usada como memória virtual é chamada **área de troca** ou **área de swap** (swap space).

O Linux pode usar tanto um arquivo normal de um sistema de arquivos quanto uma partição separada para a área de troca. Uma partição dedicada é mais rápida, porém é mais fácil mudar o tamanho de um arquivo de troca (não há necessidade de reparticionar o disco inteiro, e possivelmente instalar o sistema desde o início).

Quando se sabe a quantidade de troca necessária, o mais indicado é utilizar uma partição dedicada, mas em caso de dúvida, deve-se utilizar um arquivo de troca inicialmente, até que se esteja seguro sobre a quantidade necessária. Neste hora é indicada a criação de uma partição específica.

Cabe salientar que o Linux permite a utilização simultânea de diversas partições e arquivos de troca. Isso significa que caso seja necessário ocasionalmente uma área de troca maior, pode-se criar um arquivo com esta finalidade, ao invés de gerar uma partição com todo o espaço necessário.

Nota sobre termos em sistemas operacionais: a ciência da computação normalmente faz distinção entre troca (gravar todo o conteúdo de um processo em uma área de troca) e paginação (gravar partes fixas da memória, normalmente poucos kilobytes, de cada vez). Paginação é normalmente mais eficiente e é o que o Linux faz, mas por tradição a terminologia no Linux foi mantida como troca (swap)[1].

5.2 Criando uma área de troca

Um arquivo de troca é um arquivo comum, sem nenhum tratamento especial para o kernel. A única coisa que importa para o kernel é que este não possua espaços, e ele foi preparado para uso pelo utilitário `mkswap`. Ele deve residir em um disco local, ainda que possa estar em um sistema de arquivos que foi montado via NFS devido a razões de implementação.

A questão sobre ausência de espaços é importante. O arquivo de troca reserva um espaço em disco para que o kernel possa rapidamente trocar uma página sem ter que passar por todos os ítens necessários quando da alocação de um setor para um arquivo. O kernel simplesmente utiliza qualquer setor que tenha sido alocado para o arquivo de troca. Como um espaço vazio no arquivo significa que não há setores de disco alocados (naquele local), não é interessante para o kernel tentar utilizá-lo.

Uma boa maneira de criar uma área de troca sem espaços vazios é utilizar o seguinte comando:

```
$ dd if=/dev/zero of=/extra-swap bs=1024 count=1024
1024+0 registros de entrada
1024+0 registros de saída
$
```

Onde `/extra-swap` é o nome do arquivo de troca e o tamanho é dado pelo parâmetro `count=`. O tamanho mais indicado deve ser múltiplo de 4, uma vez que o kernel utiliza

[1]Apesar de deixar algumas pessoas horrorizadas!

páginas de memória de 4 Kb de tamanho. Caso não seja múltiplo alguns kilobytes não serão utilizados.

Uma partição de troca também não tem nada de especial. É criada como qualquer outra partição, a única diferença é que ela é acessada de maneira direta, isto é, não contém nenhum sistema de arquivos. É aconselhável marcar a partição de troca como de tipo 82 (Linux swap); isso tornará claro a listagem das partições, mesmo que não seja necessário para o kernel.

Após criar a partição de troca ou um arquivo de troca, é necessário gravar uma assinatura no seu início, que contém algumas informações administrativas utilizadas pelo kernel. O comando utilizado é o `mkswap`, da seguinte forma:

```
$ mkswap /extra-swap 1024
Setting up swapspace, size = 1044480 bytes
$
```

Note que a área de troca ainda não está sendo utilizada; ela existe, mas o kernel não a está utilizando como memória virtual.

Deve-se ser extremamente cuidadoso com o comando `mkswap`, uma vez que ele não checa se o arquivo ou a partição já está sendo utilizada para outra finalidade. *Pode-se facilmente sobrescrever arquivos importantes ou mesmo partições inteiras com o* ***mkswap***. Felizmente este comando é necessário somente durante a instalação do sistema.

O gerenciador de memória do Linux limita o tamanho da área de troca em cerca de 127 Mb (por várias razões técnicas, o limite atual é $(4096-10)\times 8\times 4096 = 133890048$ bytes, ou mais precisamente 127.6875 Mb). Pode-se porém utilizar até 16 áreas de troca simultaneamente, totalizando cerca de 2 Gb[2].

5.3 Usando a área de troca

Uma área de troca é ativada através do comando `swapon`. Este comando diz ao kernel que a área de troca pode ser usada. O caminho para a área de troca é dado como um argumento. Para iniciar o processo de troca em um arquivo temporário pode-se usar o seguinte comando:

```
$ swapon /extra-swap
$
```

[2]Um gigabyte aqui, outro ali e estaremos, logo logo, falando de *memória de verdade*

Áreas de troca podem ser usadas automaticamente ao serem listadas no arquivo `/etc/fstab`:

```
/dev/hda8       none       swap       sw    0    0
/swapfile       none       swap       sw    0    0
```

O script de inicialização `/etc/rc.d/rc.sysinit` executará o comando `swapon -a`, inicializando as áreas de troca listadas no arquivo `/etc/fstab`. O `swapon` é ainda usado para se adicionar novas áreas de troca, sem precisar reinicializar o sistema.

Pode-se monitorar o uso das áreas de troca através do comando `free`. Ele apresentará o total de troca disponível:

```
$ free
             total       used       free     shared    buffers     cached
Mem:         63140      60328       2812      32528       5248      18020
-/+ buffers/cache:      37060      26080
Swap:       104384      15288      89096
$
```

O primeiro item da saída (`Mem:`) mostra a memória física. A coluna de total não apresenta a memória física usada pelo kernel, a qual gira em torno de um megabyte. A coluna *used* apresenta a memória utilizada. A coluna de memória livre (`free`) mostra a quantidade de memória sem uso. A coluna de memória compartilhada (`shared`) mostra a memória utilizada por diversos processos; quanto mais, melhor. A coluna de buffers apresenta o tamanho atual do buffer cache de disco.

A última linha (`Swap:`) mostra informações similares sobre a área de troca. Se todos os dados desta linha estiverem zerados, sua área de troca não está ativada.

As mesmas informações podem ser obtidas através do comando `top` ou através do arquivo `/proc/meminfo` no sistema de arquivos `/proc`.

Uma área de troca pode ser desativada através do comando `swapoff`. Normalmente isso não é necessário, exceto para áreas temporárias. Qualquer página em uso na área será devolvida à memória primeiro, e caso não haja memória física disponível para manter todas as páginas, o Linux começará a ter problemas e não conseguirá executar diversos comandos; após um período ele pode se recuperar, mas neste meio tempo estará indisponível. Deve-se checar, com o comando `free`, se há memória livre suficiente para remover a área de troca em uso.

Todas as áreas de troca que são usadas automaticamente através do comando `swapon -a` podem ser removidas pelo comando `swapoff -a`; ele verifica o conteúdo do arquivo `/etc/fstab` para descobrir o que deve ser removido. As áreas de troca ativadas manualmente permanecerão em uso.

Algumas vezes uma grande quantidade de área de troca pode estar em uso, apesar de existir uma grande quantidade de memória física livre. Isso pode ocorrer por exemplo quando em determinado momento há necessidade de efetuar uma troca, porém logo após um processo que ocupava muita memória física foi concluído, liberando a memória física em uso. Os dados que foram gravados na área de troca não são automaticamente retornados para a memória física até que isso seja necessário, podendo a memória física ficar um longo tempo sem uso. Não há com o que se preocupar em relação a esta situação, porém é bom saber o que está ocorrendo.

5.4 Compartilhando áreas de troca com outros sistemas operacionais

Memória virtual é uma funcionalidade disponível em diversos sistemas operacionais. Uma vez que cada sistema somente a utiliza quando estiver rodando, ou seja, nunca ao mesmo tempo, pois não é possível executar mais de um sistema ao mesmo tempo, uma idéia mais eficiente seria compartilhar uma única área de troca. Isso é possível, mas requer um conhecimento avançado. O Tips-HOWTO (veja o Apêndice D) contém algumas orientações sobre a forma de implementar isto.

5.5 Alocando áreas de troca

Algumas pessoas dirão que se deve alocar o dobro da memória física para a área de troca, mas esta é uma regra simplista. Seguem sugestões de como se fazer isto corretamente:

1. Estime o total de memória necessária. Esta é a maior quantidade de memória que você provavelmente irá precisar em um determinado momento, ou seja, a soma dos requisitos de memória de todos os programas que podem rodar simultaneamente. Isso pode ser feito rodando-se todos os programas passíveis de execução ao mesmo tempo em seu sistema.

   Por exemplo, caso se deseje executar o X, deve-se alocar no mínimo 8 Mb para ele, o gcc usa muitos megabytes (alguns arquivos necessitam grandes quantidades de memória, até dezenas de megabytes, mas normalmente ao redor de 4 Mb deve ser suficiente, e assim por diante. O kernel utiliza em torno de 1 Mb por si só, e o interpretador de comandos padrão e outros pequenos utilitários usam alguns poucos kilobytes (digamos 1 Mb se somados). Não há necessidade de ser exato, estimativas aproximadas são suficientes, mas neste caso deve-se sempre estimar pelo máximo.

   Lembre-se que se há diversos usuários utilizando o sistema ao mesmo tempo, todos eles irão consumir memória. Porém, caso duas pessoas executem o mesmo programa ao mesmo tempo, o total de memória consumido não será o dobro, pois as páginas de código e bibliotecas compartilhadas existem em somente uma instância. Os comandos `free` e `ps` são muito úteis para estimar a memória necessária.

2. Adicione uma margem de segurança à estimativa do passo 1. Isso porque as estimativas do tamanho dos programas estarão provavelmente erradas. Certamente há outros programas que serão necessários e provavelmente não foram lembrados. Também é bom ter-se uma certa margem de segurança. Alguns megabytes devem dar conta do recado (é melhor alocar uma área de troca maior do que uma área muito pequena, mas também não há necessidade de se alocar o disco inteiro, afinal, a área de troca é um espaço perdido, caso não utilizada). Como é melhor trabalhar com números inteiros, pode-se arredondar o valor para o próximo megabyte.

3. Baseado nos cálculos acima, você irá descobrir quanta memória será necessária no total. Então, para alocar a área de troca, basta somente subtrair o tamanho da memória física do total de memória necessária; assim obtém-se o tamanho necessário para a área de troca (em alguns sistemas UNIX é necessário alocar espaço para uma imagem da memória física. Neste caso o valor encontrado no passo 2 é o valor necessário para a área de troca).

4. Caso a área de troca seja muito maior que a memória física (algumas vezes maior), será necessário aumentar o total de memória física instalada, pois de outra forma a performance será razoável.

É aconselhável ter ao menos alguma área de troca, mesmo que os cálculos indiquem que nenhuma área é necessária. O Linux usa a área de troca de forma agressiva, desta maneira o máximo possível de memória física pode permanecer livre. O Linux irá transferir para a área de troca as páginas de memória que não estejam sendo utilizadas, mesmo que a memória física não esteja sendo requisitada para outra tarefa. Isso evita esperas quando a memória for necessária — a troca pode ser feita mais cedo, quando o disco está disponível.

A área de troca pode ser dividida em diversos discos. Isso pode eventualmente aumentar a performance, dependendo da velocidade relativa dos discos e do tipo de acesso aos mesmos. Pode-se experimentar algumas formas, mas é importante salientar que as experiências podem ser bastante complexas. Não se deve crer que uma maneira é superior à outra pois nem sempre isso será verdade.

5.6 O cache

A leitura do disco[3] é muito lenta quando comparada com o acesso a memória real. Adicionalmente, é comum ler-se a mesma parte do disco diversas vezes durante um período de tempo relativamente curto. Por exemplo, alguém pode primeiro ler um email, depois respondê-lo em um editor e finalmente salvar o mail em uma pasta para arquivamento. Ou então considere quão freqüentemente o comando `ls` é executado em um sistema com diversos usuários. Ao ler as informações do disco e armazenando-os em memória até que não seja mais necessário, pode-se agilizar todos os acessos posteriores ao primeiro. Isso é chamado de **buffering de disco**, e a memória usada para este fim é chamada de **buffer cache**.

Uma vez que a memória é, infelizmente, um recurso escasso, o buffer cache não pode ser muito grande. Ele pode não conter todos os dados que os programas necessitem. Quando o cache está cheio, os dados sem uso há mais tempo são descartados e a memória é então liberada para uso por dados novos.

Buffering de disco funciona também para gravação. Por um lado, dados que são gravados são também lidos com muita freqüência (por exemplo o código fonte de um programa, primeiro é salvo em um arquivo, e depois é lido pelo compilador), então, colocar os dados gravados em um cache é também uma boa idéia. Por outro lado, colocar os dados a serem gravados somente em memória, sem gravá-los efetivamente em disco, agiliza o programa que deveria gravá-los. A gravação pode então ser feita em segundo plano, sem diminuir a velocidade de gravação dos outros programas.

Muitos sistemas operacionais têm cache de buffer (apesar de às vezes receber outro nome), mas nem todos funcionam dentro dos mesmos princípios. Alguns são chamados **write-through**[4]: os dados são gravados no disco imediatamente (e são mantidos no cache obviamente), outros são chamados **write-back**[5], isto é, a gravação somente é feita após intervalos de tempo. O write-back é mais eficiente, mas mais suscetível a erros: caso uma máquina tenha algum problema, ou a energia elétrica for cortada em um mau momento, ou o disquete for removido sem ser desmontado, as mudanças registradas no cache são perdidas. Isso pode significar que o sistema de arquivos pode não funcionar bem, talvez devido a dados não gravados que continham informações críticas à atualização do sistema de arquivos.

Por causa disso, nunca deve-se desligar um equipamento sem seguir os procedimentos adequados (veja o capítulo 6), ou remover um disquete da unidade sem que o dispositivo esteja desmontado. O comando `sync` esvazia o buffer, isto é, força a gravação de dados ainda não atualizados, podendo ser utilizado quando se quer ter certeza que tudo está em segurança e atualizado. Em sistemas UNIX tradicionais, há um programa chamado `update` que roda em segundo plano, executando um `sync` a ca-

[3]Exceto a RAM disk, por razões óbvias
[4]de **gravação imediata**
[5]de **gravação posterior**

da 30 segundos. O Linux tem ainda um outro programa chamado `bdflush`, o qual executa uma sincronização imperfeita mais freqüentemente para evitar travamentos súbitos devido ao grande tráfego de entrada e saída que o `sync` pode, ocasionalmente, gerar.

No Linux, o `bdflush` é ativado pelo `update`. Não há razão para preocupar-se com isso, mas se o `bdflush` morrer por alguma razão, o kernel emitirá um aviso sobre a ocorrência e ele deverá ser reinicializado executando-se o `/sbin/update`.

O cache não põe arquivos no buffer, mas sim blocos, que são as menores unidades de E/S (no Linux, eles têm normalmente 1 Kb). Dessa forma, diretórios, superblocos, outros sistemas de arquivos e discos sem sistemas de arquivos são armazenados no cache.

A eficiência do cache é definida basicamente pelo seu tamanho. Um cache pequeno é quase inútil: ele manterá poucos dados e toda a informação será retirada do cache antes de ser usada novamente. O tamanho correto depende do volume de dados que é lido e gravado simultaneamente e qual a freqüência que o mesmo dado é acessado. A única forma de saber é experimentando.

Caso o cache tenha um tamanho fixo, é melhor não definí-lo com um tamanho muito grande, pois isso pode diminuir demasiadamente a memória livre e causar um volume excessivo de troca (e conseqüentemente deixar o sistema mais lento). Para tornar mais eficiente o uso da memória real, o Linux automaticamente usa toda a memória livre como cache de buffer, mas também o diminui quando os programas necessitam de mais memória.

No Linux, você não precisa fazer nada para utilizar o cache, tudo acontece automaticamente. Exceto por seguir os procedimentos de desligamento e remoção de disquetes, não há com o que se preocupar.

Capítulo 6

Iniciando e encerrando o sistema

Start me up
Ah... you've got to... you've got to
Never, never never stop
Start it up
Ah... start it up, never, never, never
You make a grown man cry,
you make a grown man cry
(Rolling Stones)

Esta seção explica o que acontece quando o Linux é inicializado ou desligado, e como isto pode ser feito adequadamente. Caso os procedimentos adequados não sejam seguidos, arquivos podem ser corrompidos ou perdidos.

6.1 Uma visão geral da inicialização e encerramento do sistema

O ato de ligar o computador e carregar[1] o sistema operacional é chamado **booting** (inicialização). O nome vem da imagem do computador levantando-se, mas o ato em si mesmo é um pouco mais realista.

Durante a inicialização, o computador carrega inicialmente uma pequena porção de código chamado de **bootstrap loader** [2], o qual carrega e inicia o sistema operacional. O bootstrap loader é normalmente armazenado em uma localização fixa no disco rígido ou disquete. A razão para estes dois passos é que o sistema operacional é grande e complicado, mas a primeira parte de código a ser carregada deve ser muito pequena

[1] Nos primeiros computadores, não bastava simplesmente ligar o computador, os operadores tinham que manualmente colocar o sistema operacional na máquina. Estas novas maravilhas que temos hoje fazem isso por si só.

[2] **autocarregador**

(poucas centenas de bytes), evitando-se a necessidade de fabricação de firmwares muito complicados.

Diferentes computadores implementam inicializações diferentes. Para PCs, o computador (a BIOS) lê o primeiro setor (chamado **setor de inicialização**) do disquete ou do disco rígido. O bootstrap loader está neste setor. Ele carrega o sistema operacional, onde quer que ele esteja no disco (ou em qualquer outro local onde ele esteja).

Após a carga do Linux, este inicializa o hardware e os controladores de dispositivos. É executado o processo `init`, o qual inicia os demais processos que permitem o acesso aos usuários e a execução de seus programas. Os detalhes desta fase serão discutidos a seguir.

No encerramento do sistema, todos os processos recebem o aviso para terminarem suas atividades (isso faz com que os arquivos sejam fechados e todos os procedimentos necessários sejam tomados pelos programas); os sistemas de arquivos e áreas de swap são desmontados e finalmente uma mensagem é apresentada na console, sinalizando que o computador pode ser desligado. Caso o procedimento correto não seja seguido, coisas muito desagradáveis podem e *vão* ocorrer; mais importante, o buffer cache do sistema de arquivos pode não ter sido descarregado, o que significa que todos os dados nele serão perdidos e o sistema de arquivos estará inconsistente e possivelmente instável.

6.2 O processo de inicialização em maiores detalhes

Pode-se inicializar o Linux a partir de um disquete ou do disco rígido. A seção de instalação do Guia de Instalação e Introdução ao Linux (`getting-started`) descreve como instalar o Linux e iniciá-lo da forma que se queira.

Quando um PC é inicializado, a BIOS executará diversos testes para checar se tudo está em perfeita ordem[3], chamando então o boot. Ela escolherá um dispositivo de disco (tipicamente o primeiro acionador de disquetes ou o primeiro disco rígido, se houver um instalado; a ordem porém pode ser configurada) e lerá o primeiro setor do dispositivo. Este é chamado de **setor de inicialização**[4], e no caso de discos rígidos é também chamado de **master boot record**, uma vez que um disco rígido pode conter diversas partições, cada qual com o seus próprios setores de inicialização.

O setor de inicialização contém um pequeno programa, (pequeno o suficiente para estar contido em um setor), o qual tem a função de ler o sistema operacional instalado e inicializá-lo. Quando a inicialização é realizada a partir de um disquete, o setor de inicialização contém um determinado código que apenas lê algumas centenas de blocos

[3]Isto é chamado de **autoteste**

[4]**boot sector**

(dependendo do tamanho do kernel) em uma determinada área da memória. Em um disquete de boot do Linux, não há sistema de arquivos, o kernel está armazenado em setores consecutivos, uma vez que isso simplifica o processo de inicialização. É possível inicializar o sistema a partir de um disco que contenha um sistema de arquivos utilizando-se o LILO[5].

Ao ser inicializado a partir de um disco rígido, o código do registro mestre de inicialização examinará a tabela de partições (também localizada no MBR), identificará a partição ativa (a partição indicada como inicializável), lerá o setor de inicialização daquela partição e iniciará o programa gravado naquele setor. O código no setor de inicialização faz exatamente o que o setor de inicialização de um disquete faz: lerá o kernel de uma partição e o executará. Os detalhes variam, uma vez que não é muito útil ter uma partição separada somente para manter uma imagem do kernel, uma vez que o código no setor de inicialização não pode simplesmente ler o disco seqüencialmente, e sim tem que encontrar os setores aonde quer que o sistema de arquivos o tenha colocado. Há diversas soluções para o problema, mas o mais comum é utilizar o LILO (os detalhes sobre como fazer isso são irrelevantes neste momento, para maiores informações verifique a documentação do LILO).

Quando o sistema é iniciado utilizando o LILO, ele inicializa o padrão. É possível configurar o LILO para iniciar diversos kernels, ou mesmo iniciar outros sistemas diferentes do Linux, sendo possível usá-lo para escolher qual sistema ou kernel utilizar na hora da inicialização do sistema. Para tanto basta pressionar `alt`, `shift` ou `ctrl` durante a inicialização do sistema (quando o LILO é carregado) e serão apresentadas as opções disponíveis, ao invés da inicialização automática com o sistema definido como padrão. Alternativamente, o LILO pode ser configurado para sempre perguntar qual sistema deve ser carregado, e após um tempo máximo configurável de espera, carregar o sistema definido como padrão.

Com o LILO é possível ainda enviar **argumentos ao kernel**, imediatamente após o nome do sistema operacional.

Note que há outros gerenciadores de inicialização, além do LILO, como por exemplo o loadlin. Informações sobre estes sistemas podem ser adicionadas em futuras versões deste manual.

Inicializar o sistema a partir de disquete ou um disco rígido tem suas vantagens e desvantagens, porém a segunda alternativa tende a ser a melhor, uma vez que evita os problemas inerentes aos disquetes, além de ser mais rápido. Ainda assim pode haver um certo nível de problemas ao se instalar o sistema a partir de um disco rígido, fazendo com que muitos usuários utilizem disquetes para uma primeira inicialização do sistema, executem a sua instalação no disco rígido, assim como a do LILO, e somente a partir daí passem a utilizar o disco rígido como unidade de inicialização do sistema.

[5]LInux LOader

Após a carga do kernel do Linux em memória, independente da forma, e o sistema estando efetivamente em execução, ocorrem os seguintes eventos:

- O kernel é instalado inicialmente em um formato compactado, sendo então descompactado de forma automática, através de um pequeno programa que está localizado no início da imagem do kernel.
- Caso se esteja utilizando uma placa de vídeo Super VGA que o Linux reconheça e que tenha modos de texto especiais (como utilizar 100 colunas por 40 linhas), o Linux pode perguntar que modo se deseja utilizar. Durante a compilação do kernel, é possível predefinir um modo de vídeo então essa pergunta nunca será apresentada. Isso pode ser feito também através do LILO ou do comando `rdev`.
- Após, o kernel verifica os demais hardwares (discos rígidos, disquetes, placas de rede...) e configura os seus controladores de dispositivos. Neste meio tempo, diversas mensagens são apresentadas, como por exemplo:

```
LILO boot:
Loading linux....
Memory: sized by int13 088h
Console: 16 point font, 400 scans
Console: colour VGA+ 80x25, 1 virtual console (max 63)
pcibios_init : BIOS32 Service Directory structure at 0x000f8630
pcibios_init : BIOS32 Service Directory entry at 0xf8080
pcibios_init : PCI BIOS revision 2.10 entry at 0xf80b0
Probing PCI hardware.
Calibrating delay loop.. ok - 66.56 BogoMIPS
Memory: 63136k/65536k available (740k kernel code, 384k reserved, 1276k data)
Swansea University Computer Society NET3.035 for Linux 2.0
NET3: Unix domain sockets 0.13 for Linux NET3.035.
Swansea University Computer Society TCP/IP for NET3.034
IP Protocols: IGMP, ICMP, UDP, TCP
VFS: Diskquotas version dquot_5.6.0 initialized
Checking 386/387 coupling... Ok, fpu using exception 16 error reporting.
Checking 'hlt' instruction... Ok.
Intel Pentium with F0 0F bug - workaround enabled.
alias mapping IDT readonly ...  ... done
Linux version 2.0.36 (root@frajola.conectiva.com.br) (gcc version 2.7.2.3) #2 se
g set 7 11:54:18 EST 1998
```

  O texto exato difere em diversos sistemas, dependendo do hardware disponível, assim como da versão do Linux e de como ele foi configurado.

- O kernel tentará então montar o sistema de arquivos raiz. A partição pode ser configurada em tempo de compilação, ou a qualquer tempo através do LILO ou do `rdev`. O tipo de sistema de arquivos é detectado automaticamente. Caso a sua montagem falhe, devido ao esquecimento de inclusão deste tipo de sistema de arquivos no kernel, este dará uma mensagem de erro fatal (panic) e abortará a inicialização do sistema.

O sistema de arquivos raiz é normalmente montado com permissões somente de leitura (isto pode ser configurado também com o `rdev`). É possível, desta maneira, checar o sistema de arquivos enquanto ele está montado, porém não é uma boa idéia tentar fazer isso em sistemas de arquivos montados com permissão de escrita/leitura.

- Após isso, o kernel executará o programa `init` (localizado em `/sbin/init`) em segundo plano[6]. O `init` sempre terá o número de processo igual a 1. O `init` cuida da ativação de diversos serviços. O que ele fará exatamente depende de como está configurado; veja o capítulo 7 para maiores informações. No mínimo ele iniciará alguns servidores essenciais.
- O `init` então passará o sistema para o modo multiusuário e iniciará o `getty`, permitindo o uso de consoles virtuais e linhas seriais. O `getty` é o programa que permite que os usuários acessem o sistema. O `init` pode também inicializar outros programas, dependendo de como esteja configurado.
- Após estes passos o sistema estará ativo e pronto para uso.

6.3 Mais informações sobre o encerramento do sistema

É importante seguir corretamente os passos quando encerrar o sistema. Caso este procedimento não seja cumprido, seus sistemas de arquivos provavelmente estarão prejudicados e os arquivos poderão ficar bagunçados. Isto acontece porque o Linux tem um cache de disco que não grava os dados no disco no momento em que isto é solicitado, mas somente em intervalos de tempo. A performance do sistema aumenta consideravelmente, mas isso também pode causar a perda de dados caso você simplesmente desligue seu computador.

Outra razão contra desligar simplesmente o sistema é que em um sistema multitarefa existem diversas coisas acontecendo em segundo plano, e desligar o computador neste momento pode ser desastroso. Usando o procedimento correto para o encerramento do sistema, você garante que todos os processos em background podem salvar seus dados.

Para encerrar corretamente um sistema Linux, utilize o comando `shutdown`. Ele é usado normalmente de duas maneiras.

Caso esteja utilizando um sistema onde você é o único usuário, a maneira correta de se utilizar o `shutdown` é sair de todos os programas que porventura você esteja rodando, sair de todas as consoles do sistema, acessar uma das consoles como `root` (ou permanecer no sistema como `root` mudando o diretório corrente para o diretório pessoal do root para não ter problemas na desmontagem de sistemas de arquivos);

[6] background

executar o comando `shutdown -h now` (substitua o `now` por um `+` e um número em minutos caso você queira postergar o shutdown).

Alternativamente, se o seu sistema tem diversos usuários, use o comando `shutdown -h` *`+tempo mensagem`*, onde *tempo* é o tempo em minutos até o sistema ser parado, e *mensagem* é a mensagem que será enviada aos usuários que estão conectados neste momento ao sistema.

```
# shutdown -h +10 'Iremos instalar um novo disco.  O sistema deverá
> voltar ao normal em três horas.'
#
```

Este procedimento, avisará a todos os usuários que o sistema será desligado em dez minutos, e que eles poderão perder seus dados. O aviso é apresentado em todo o terminal que estiver conectado, inclusive `xterms`:

```
Broadcast message from root (ttyp0) Wed Aug 2 01:03:25 1995...

Iremos instalar um novo disco.  O sistema deverá
voltar ao normal em três horas.
The system is going DOWN for system halt in 10 minutes !!
```

Este aviso é automaticamente repetido algumas vezes antes do encerramento, e a intervalos cada vez menores, à medida que o tempo passa.

Quando o encerramento do sistema tem início, todos os sistemas de arquivos (exceto o raiz) são desmontados, os processos de usuários (casa haja algum ainda utilizando o sistema) são finalizados, servidores são encerrados, e finalmente o sistema de arquivos raiz é desmontado. Quando isso ocorre o processo `init` apresenta uma mensagem indicando que o computador pode ser desligado. Agora, *e jamais antes disso*, pode-se desligar o equipamento.

Algumas vezes, apesar de ser raro em bons sistemas, é impossível encerrar o sistema de forma adequada. Por exemplo, caso ocorra um erro fatal com o kernel, pode ser impossível executar qualquer novo comando, tornando o encerramento correto inviável. Tudo o que se pode fazer neste caso é esperar que nenhum dano maior ocorra e então desligar a máquina. Caso os problemas sejam menos sérios (digamos que alguém quebrou o seu teclado com um machado!), e o kernel está rodando normalmente, é aconselhável aguardar alguns minutos para que os dados sejam atualizados, esvaziando assim o cache de disco, e somente após desligar o equipamento.

Alguns usuários gostam de utilizar o comando `sync`[7] três vezes, aguardando pela gravação do buffer em disco, e desligando o equipamento em seguida. Caso não

[7] o sync descarrega os buffers do sistema

hajam programas sendo executados, esse procedimento equivale ao `shutdown`. Ainda assim nenhum sistema de arquivos é desmontado o que pode gerar problemas com o indicador de limpeza do sistema de arquivos do tipo ext2fs. Este método não é recomendado.

(A razão dos três comandos sync remonta aos primeiros tempos do UNIX, onde os comandos eram digitados separadamente, o que dava tempo suficiente para que as operações de E/S fossem finalizadas).

6.4 Reiniciando o sistema

Reinicializar[8] significa iniciar o sistema novamente. Isto pode ser conseguido encerrando o sistema, desligando o computador e então ligando-o novamente. Um método mais prático para ter o mesmo efeito consiste em executar o `shutdown` com a opção de reinicialização. Isto é conseguido executando-se `shutdown -r now`, sendo que a opção `-r` significa 'reinicialização'.

A maioria dos sistemas Linux executa `shutdown -t3 -r now` [9] quando o conjunto de teclas ctrl-alt-del são pressionadas. Este procedimento reinicializa o sistema. A ação do ctrl-alt-del é configurável e poderá ser desejável aguardar alguns minutos para a sua execução em uma máquina multiusuário. Sistemas que são fisicamente acessíveis a qualquer pessoa, devem ser configurados para não executar nada ao serem pressionadas tais teclas. Isto pode ser feito editando-se o arquivo `/etc/inittab` e comentando-se [10] a linha respectiva.

6.5 Modo monousuário

O comando `shutdown` também pode ser usado para colocar o sistema em modo monousuário, no qual ninguém pode acessar o sistema, além do superusuário `root`. Isso é muito útil para tarefas de administração que não podem ser executadas caso o sistema esteja sendo executado normalmente.

6.6 Disquetes de emergência

Nem sempre pode ser possível inicializar o sistema a partir do disco rígido. Um erro na configuração do LILO pode tornar o sistema incapaz de ser iniciado. Para essas situações, é necessário ter uma alternativa de inicialização que funcione sempre (desde

[8] Também conhecido como `rebootar`
[9] ou alguma variante
[10] com o caracter # no início da linha

que o hardware também funcione). Para PCs, isso pode significar inicializar o sistema a partir do acionador de disquetes.

Muitas distribuições permitem a criação de **disquetes de emergência** durante a instalação. É uma excelente idéia criá-los, ainda que algumas vezes ele possa conter somente o kernel do sistema, e presuma que os programas serão utilizados a partir dos discos de instalação da distribuição, para corrigir o que quer que seja. Algumas vezes isso pode não ser suficiente, como por exemplo, quando se deseja restaurar alguns arquivos a partir de cópias de segurança feitos através de softwares não presentes nos discos de instalação.

Pode ser necessário criar um disquete de inicialização personalizado. O *Bootdisk HOWTO* de Graham Chapman (`bootdisk-howto`) contém as instruções de como se fazer isso. Mas lembre-se de manter os disquetes de emergência e de boot atualizados.

Não se pode utilizar a unidade de disquetes utilizada para montar o disquete de boot para qualquer outra finalidade. Isso pode ser incoveniente caso se tenha somente uma unidade de disquetes. Ainda assim, caso haja memória suficiente, pode-se configurar o disquete de boot para carregar o seu conteúdo para a memória (o disco de inicialização do kernel necessita ser especialmente configurado para isso). Uma vez que o disquete tenha sido carregado em memória, a unidade estará liberada para montar outros discos.

Capítulo 7

init

Uuno on numero yksi

Este capítulo descreve o processo `init`, que é o primeiro processo de nível de usuário que é iniciado pelo kernel. O `init` tem diversas tarefas importantes, como iniciar o `getty` (após o qual os usuários podem acessar o sistema), implementar os níveis de execução, e tomar conta dos processos órfãos. Este capítulo explica como o `init` é configurado e como fazer uso dos diferentes níveis de execução.

7.1 O `init` vem em primeiro lugar

O `init` é um dos programas absolutamente essenciais para a operação de um sistema Linux, mas que a maioria dos usuários pode ignorar. Uma boa distribuição do Linux conterá a configuração de um `init` que funcionará com a maioria dos sistemas, e não haverá necessidade de se fazer absolutamente nada. Usualmente o administrador somente irá preocupar-se com o init caso necessite lidar com terminais seriais, modens configurados para atendimento (e não para chamadas) ou caso deseje mudar o nível de execução padrão do sistema.

Quando o kernel auto-inicializa (foi carregado em memória, começa a rodar e inicializa todos os dispositivos e estruturas de dados), ele finaliza as suas tarefas na inicialização do sistema ao iniciar o programa de nível de usuário `init`. O `init` é sempre o primeiro processo do sistema (o seu número de processo é sempre igual a 1).

O kernel procura pelo `init` em alguns diretórios que vêm sendo usados historicamente para isso, porém a localização correta no Linux é o `/sbin/init`. Caso o kernel não consiga encontrá-lo, ele executará o programa `/bin/sh` e caso isso também falhe, a inicialização do sistema é abortada.

Quando o `init` começa, ele finaliza o processo de inicialização ao executar uma série de tarefas administrativas como a checagem dos sistemas de arquivos, limpeza do `/tmp`, início de vários serviços e inicialização do `getty` para cada terminal e console virtual através dos quais os usuários serão autorizados a acessar o sistema (veja o capítulo 8).

Após o sistema ter sido adequadamente inicializado, o `init` reinicia o `getty` para cada terminal após a saída do usuário do sistema (permitindo que o próximo usuário possa acessar o sistema). Além disso, o `init` adota também todos os processos órfãos: quando um processo inicia um processo filho e é finalizado antes dele, imediatamente o processo restante torna-se filho do `init`. Isso é importante por várias razões técnicas, mas é bom conhecer para entender as listas de processos e a árvore de processos.[1]

Há poucas variantes disponíveis do `init`. Muitas distribuições do Linux usam o `sysvinit` (escrito por Miquel van Smoorenburg), o qual é baseado no `init` do System V. As versões BSD do UNIX tem um `init` diferente. A diferença reside nos níveis de execução: presentes no System V, mas não no BSD (pelo menos tradicionalmente). Essa diferença não é essencial e nós examinaremos somente o `sysvinit`.

7.2 configurando o `init` para inicializar o `getty`: o arquivo `/etc/inittab`

Quando é inicializado, o `init` lê o arquivo de configuração `/etc/inittab`. Enquanto o sistema estiver no ar, ele será lido novamente, caso seja enviado um sinal HUP[2], tornando desnecessário reinicializar o sistema para que as mudanças do `init` façam efeito.

O arquivo `/etc/inittab` é um pouco complicado. Começaremos pelo caso mais simples, ou seja, configurando as linhas do `getty`. As linhas do `/etc/inittab` consistem de quatro campo delimitados por dois pontos:

id:nível:ação:processo

Os campos são descritos a seguir. O `/etc/inittab` pode conter algumas linhas vazias, e linhas que comecem com '#' serão consideradas comentários.

id — Identifica a linha no arquivo. Para linhas referentes ao `getty` especifica o terminal em que eles são executados (os caracteres após o `/dev/tty` no nome do arquivo de dispositivo). Para outras linhas não têm efeito (exceto pelas restrições de tamanho), e devem ser únicas.

[1] o `init` por si só não pode morrer. Você não pode matá-lo mesmo com um SIGKILL.
[2] usando o comando `kill -HUP 1`, como root

nível	Os níveis de execução em que a linha deve ser considerada. Os níveis de execução são definidos através de dígitos sem delimitadores e são melhores descritos na próxima seção.
ação	Define a ação que deve ser tomada pela linha. Por exemplo, `respawn` para executar novamente o comando do próximo campo, quando este encerra seu processamento ou `once` para executá-lo somente uma única vez.
processo	O comando a ser executado.

Para iniciar o `getty`[3] no primeiro terminal virtual (`/dev/tty1`), em todos os modos de execução multiusuário (de 2 a 5), pode-se informar a seguinte linha:

```
1:12345:respawn:/sbin/getty 9600 tty1
```

O primeiro campo diz que a linha deve ser executada para `/dev/tty1`. O segundo que ele aplica-se aos níveis de execução de 1 a 5. O terceiro que o comando deve ser reinicializado quando o processo termina (ou seja quando um usuário desconectar-se de um terminal, o `getty` será executado novamente para que outro usuário possa conectar-se). O último campo executa o processo getty no primeiro terminal virtual[4].

Caso necessite adicionar terminais ou modens para atendimento de chamadas ao sistema, deve-se adicionar mais linhas ao arquivo `/etc/inittab`, uma para cada terminal ou modem. Para maiores detalhes veja as páginas de manual do *init*(8), *inittab*(5), *mingetty*(8) e *mgetty*(8).

Caso o comando falhe ao ser executado, e o `init` esteja configurado para `reinicializá-lo`, isso certamente consumirá uma grande quantidade de recursos pois o processo de iniciar o comando se repetirá indefinidamente. Para prevenir esse tipo de problema, o `init` verificará a freqüência de reinicialização do comando e caso esta seja muito grande, o `init` aguardará automaticamente por cinco minutos antes de iniciá-lo novamente.

7.3 Níveis de execução

Nível de execução é o estado do `init` e de todo o sistema que define que serviços estarão operacionais. Eles são identificados por números, de acordo com a tabela 7.1. Não há nenhum consenso de como utilizar os níveis definidos para usuário (de 2 a 5). Alguns administradores de sistema utilizam os níveis de execução para definir quais subsistemas serão executados, por exemplo se o X estará disponível ou as

[3]atualmente para terminais virtuais do console utiliza-se o `mingetty`

[4]Diferentes versões do `getty` rodam de maneira diferente. Consulte a página de manual para maiores informações

funcionalidades de rede e assim por diante. Outros têm todos os subsistemas sendo ativados e sendo finalizados individualmente, sem mudar o nível de execução, já que este pode ser um pouco complexo para controlar seus sistemas. Cada administrador deve definir qual o método mais adequado às suas necessidades, porém seguir a forma definida pela distribuição em uso deve ser o meio mais simples.

Tabela 7.1: Níveis de execução

0	desligue.
1	monousuário.
2	multiusuário, sem NFS.
3	multiusuário completo.
4	não usado.
5	X11.
6	reinicialize.

Níveis de execução são configurados no `/etc/inittab` por linhas como a seguinte:

```
l2:2:wait:/etc/rc.d/rc 2
```

O primeiro campo é um rótulo arbitrário; o segundo significa que ele se aplica ao nível de execução 2. O terceiro significa que o `init` deve executar o comando contido no quarto campo uma única vez, quando o sistema entrar neste nível, e que o `init` deve aguardar que ele seja concluído. O `/etc/rc.d/rc` executa todos comandos necessários para iniciar e parar os serviços previstos para o nível 2.

O comando no quarto campo executa todo o trabalho duro de configurar um nível de execução. Ele inicia os serviços que ainda não estejam sendo executados e finaliza os serviços que não devem rodar neste nível. Exatamente qual o comando a ser utilizado ou como o nível está configurado depende de cada distribuição do Linux.

Quando o `init` é iniciado, ele procura por uma linha no `/etc/inittab` que especifique o nível de execução padrão:

```
id:3:initdefault:
```

Pode-se informar ao init para iniciar o sistema em um outro nível de execução, passando ao kernel argumentos como `single` ou `emergency`[5]. Isso permite escolher o modo monousuário, descrito na seção 7.5.

Enquanto o sistema está sento executado o comando `telinit` pode mudar o modo de execução, o que faz com que o `init` execute o comando apropriado definido no `/etc/inittab`.

[5]argumentos para o kernel podem ser passados através do LILO

7.4 Configurações especiais no `/etc/inittab`

O arquivo `/etc/inittab` tem algumas funcionalidades especiais que permitem ações diferenciadas em situações especiais. Estas funcionalidades são definidas através de palavras chaves utilizadas no terceiro campo. Alguns exemplos:

`powerwait` Permite que o `init` encerre o sistema na falta de energia elétrica. Assume que o sistema está utilizando uma unidade de alimentação extra (no-break) e que o software da unidade informará sobre a falta de energia.

`ctrlaltdel` Permite ao `init` reinicializar o sistema, quando as teclas `control-alt-del` forem pressionadas simultaneamente. Veja que o administrador pode configurar o `C-A-D` para executar outra função. Isto é aplicável, por exemplo, nos casos em que o sistema esteja em uma localização pública.

`sysinit` Comando que deve ser executado quando o sistema for inicializado. Este comando pode limpar o conteúdo do `/tmp`, por exemplo.

Esta lista não é completa. Veja a página de manual do *inittab*(5) para todas as possibilidades e detalhes de como utilizá-las.

7.5 Iniciando em modo monousuário

Um nível de execução extremamente importante é o modo **monousuário** [6], no qual somente o administrador do sistema utiliza a máquina e o menor número possível de serviços (inclusive logins) estarão disponíveis. Este modo de execução é necessário para algumas tarefas administrativas, tais como na execução do `fsck` na partição `/usr`, isto requer que a partição esteja desmontada, o que não pode ocorrer a menos que todos os serviços do sistema estejam finalizados.

Um sistema em execução pode mudar para monousuário através do comando `telinit`. Durante a inicialização do sistema a palavra `single` ou `emergency`, na linha de comando do kernel, faz com que o `init` seja informado do nível de execução a iniciar (a linha de comando do kernel pode variar de sistema para sistema. Depende de como você está inicializando seu sistema).

A inicialização em modo monousuário pode ser necessária para executar-se o comando `fsck` manualmente, antes de qualquer montagem ou acesso a uma partição `/usr` com problemas (qualquer atividade em um sistema de arquivos inconsistente pode trazer

[6] **single user mode**, nível 1

mais problemas, devendo o `fsck` ser executado o mais rapidamente possível).

Os scripts de inicialização do `init` automaticamente entrarão em modo monousuário caso o comando `fsck` executado de forma automática apresente algum problema durante a inicialização do sistema. Esta é uma tentativa de prevenir que o sistema utilize um sistema de arquivos danificado e que o `fsck` não possa corrigir automaticamente. Tais casos são relativamente raros e usualmente envolvem um disco rígido com problemas ou uma versão experimental do kernel, porém é desejável que se esteja preparado.

Como medida de segurança, um sistema adequadamente configurado pedirá a senha do `root` antes de iniciar um interpretador em modo monousuário. De outra forma seria fácil simplesmente informar uma linha ao LILO e ganhar acesso ao sistema como `superusuário`. Caso o problema esteja no arquivo `/etc/passwd`, o melhor é ter-se à mão um disquete de inicialização.

Capítulo 8

Entrando e saindo do sistema

A mente é como um pára-quedas
Ela funciona melhor quando aberta

Esta seção descreve o que acontece quando um usuário acessa ou sai de um sistema. As várias interações entre os processos executados em segundo plano, arquivos de log, arquivos de configuração, etc... são descritos em detalhes.

8.1 Acesso via terminais

A figura 8.1 mostra como acontece o acesso via terminal. Primeiro o `init` assegura-se de que há um programa `getty` para aquele terminal ou console. O programa `getty` fica aguardando que o usuário informe que ele está pronto para acessar o sistema (normalmente através da digitação de qualquer coisa). Ao detectar um usuário, `getty` apresenta uma mensagem de boas vindas (armazenada em `/etc/issue`) e solicita a identificação do usuário e finalmente executa o programa `login`. Este recebe o nome do usuário como parâmetro e solicita ao usuário a sua senha. Caso a senha esteja correta, o `login` iniciará o interpretador de comandos (shell) que está configurado para o usuário. Caso a senha não confira ele simplesmente encerra o processo (talvez após dar ao usuário outra chance de informar seus dados). O processo init percebe que o processo foi concluído e inicia um novo getty para o terminal.

Note que o único processo novo é aquele criado pelo init (usando a chamada ao sistema `fork`); o `getty` e o `login` somente substituem o programa que estava sendo executado pelo processo (através da chamada ao sistema `exec`).

Um programa em separado, para detectar usuários é necessário para linhas seriais, desde que pode ser (e tradicionalmente é) mais complicado detectar quando um terminal se torna ativo. O `getty` adapta-se ainda à velocidade e outros parâmetros de

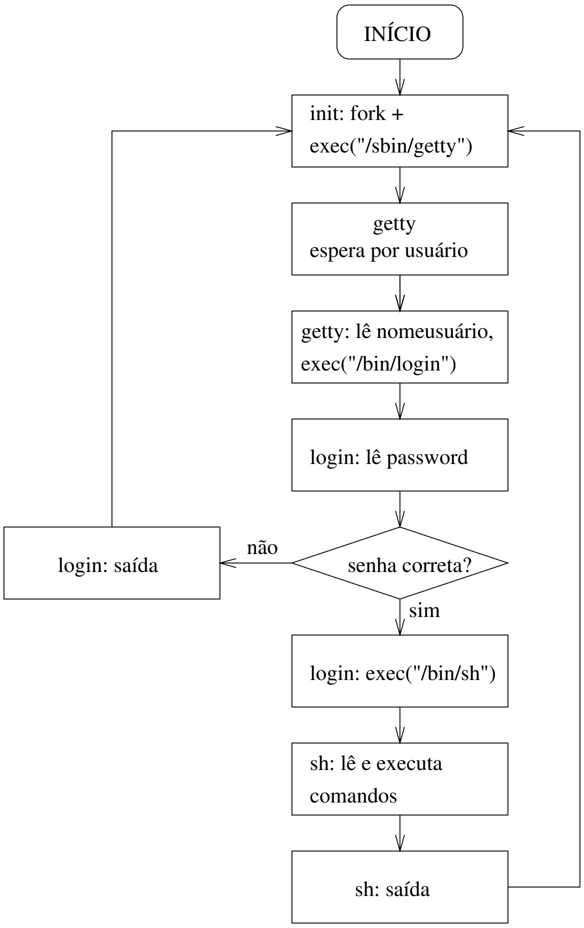

Figura 8.1: Acessos via terminal: a interação do `init`, `getty`, `login`, e shell.

comunicação, os quais são importantes em conexões discadas, onde esses parâmetros podem variar de conexão para conexão. Existem algumas versões do `getty` próprias para linhas discadas. O `mgetty` é uma delas.

Há ainda diversas versões do `getty` e `init` em uso, todas com pontos favoráveis e desfavoráveis. É aconselhável aprender sobre as versões em uso no seu sistema, e também sobre outras versões (pode-se usar o "Linux Software Map" para pesquisas). Caso não se tenha conexões discadas, não se preocupe com o `getty`, porém o `init` é ainda muito importante.

8.2 Acessos via rede

Dois computadores na mesma rede estão normalmente conectados através de um único cabo físico. Quando eles comunicam-se através da rede, os programas em cada computador que interagem na comunicação estão interligados por uma **conexão virtual**, algo como um cabo imaginário. Até onde os programas e a conexão virtual interessam, eles têm o monopólio de seu próprio cabo. Uma vez que o cabo não é real, somente imaginário, o sistema operacional de cada computador pode ter diversas conexões virtuais compartilhando o mesmo cabo físico. Desta forma usando somente um único cabo, diversos programas podem comunicar-se sem ter que se preocupar com as outras conexões. É posssível ainda ter-se diversos computadores usando o mesmo cabo; as conexões virtuais existem entre dois computadores, e os demais ignoram aquelas conexões das quais eles não fazem parte.

Esta é uma descrição resumida da realidade, porém deve ser boa o suficiente para o entendimento de porque o acesso via rede é bastante diferente de acessos normais. As conexões virtuais são estabelecidas quando há dois programas em diferentes computadores que desejam comunicar-se. Desde que, a princípio, qualquer computador pode comunicar-se com qualquer outro, há um grande número de potenciais comunicações virtuais. Devido a isso, não é muito prático iniciar um processo `getty` para cada acesso em potencial.

Há somente um único processo denominado `inetd` (correspondente ao `getty`) que administra todos os acessos de rede. Quando é detectado um pedido de acesso via rede (isto é, a detecção de uma requisição de abertura de nova conexão virtual com algum outro computador), é iniciado um novo processo que administra aquela conexão. O processo original é mantido e continua atendendo a novos pedidos de acesso.

Para tornar as coisas um pouco mais complexas, há diversos protocolos de acesso via rede. Os dois mais utilizados são telnet e rlogin. Em adição aos acessos há diversos outros tipos de conexões virtuais que podem ser realizados (como FTP, Gopher, HTTP, e outros serviços de rede). É desnecessária a manutenção de processos separados para diferentes tipos de conexões. Então, ao invés disso, há somente um processo que pode reconhecer o tipo de conexão e iniciar o tipo correto de programa para atender o serviço solicitado. Este processo é o `inetd`. Para maiores informações sobre o inetd consulte o “Guia do Administrador de Redes”.

8.3 O que o `login` faz

O `login` preocupa-se com a autenticação do usuário (assegurando que o nome do usuário e a senha conferem), por disponibilizar um ambiente inicial para o usuário através das configurações de permissões para a linha serial e início do shell.

Parte da configuração inicial é a apresentação do conteúdo do arquivo `/etc/motd` (mensagens do dia) e da checagem do correio eletrônico. Isso pode ser desabilitado através da criação de um arquivo chamado `.hushlogin` no diretório pessoal do usuário.

Caso o arquivo `/etc/nologin` exista, os acessos estarão desabilitados. Este arquivo é criado tipicamente através do comando `shutdown` e comandos relacionados. O `login` verifica a existência desses arquivos e irá recusar qualquer acesso, caso ele exista.

Neste caso, o `login` apresentará o conteúdo deste arquivo no terminal antes de ser finalizado.

As falhas de logins são registradas em um arquivo de log do sistema[1] (via `syslog`). Ele ainda armazena todos os acessos realizados pelo `superusuário`. Ambos podem ser úteis na detecção de intrusos.

Os usuários conectados no momento estão listados no arquivo /var/run/utmp. Este arquivo é válido somente até a próxima inicialização do sistema. Ele lista cada usuário e terminal (ou conexão de rede) que esteja sendo utilizado, em conjunto com algumas outras informações adicionais. Os comandos `who`, `w` e outros similares lêem o conteúdo do `utmp` a fim de verificar quem está conectado.

Todos os acessos bem sucedidos são gravados em `/var/log/wtmp`. Este arquivo é incrementado sem limites, devendo ser limpo periodicamente, por exemplo, através de uma tarefa semanal programada no `cron`[2]. O comando last mostra o conteúdo do `wtmp`.

Tanto `utmp` quanto `wtmp` estão em formato binário (vide a página de manual do `utmp`); o que torna incoveniente o seu acesso sem o uso de programas especiais.

8.4 Controle de acesso

A base de dados de usuários está tradicionalmente localizada no arquivo `/etc/passwd`. Alguns sistemas utilizam o **shadow**, o qual move as senhas para o arquivo `/etc/shadow`. Redes que tenham muitos computadores que compartilhem contas de acesso podem usar NIS ou outros métodos de armazenamento da base de dados de usuários, podendo copiar automaticamente a base de dados de uma localização central para todos os computadores da rede.

A base de dados de usuários não contém somente as senhas, mas também informações adicionais sobre os usuários, como os seus nomes reais, diretórios pessoais e shells do sistema. Estas informações devem ser públicas para que qualquer usuário possa lê-las, sendo que a senha dos usuários é armazenada encriptada. Isso produz o inconveniente de que qualquer usuário que tenha acesso ao sistema e algum conhecimento técnico, possa usar métodos criptográficos para descobrir as senhas dos demais, sem ao menos tentar acessar o sistema com outra conta. O shadow tenta evitar isso, movendo as senhas para outro arquivo, o qual somente pode ser acessado pelo superusuário (root), sendo que as senhas ainda permanecem cifradas. Cabe ressaltar que a instalação do shadow em sistemas que não têm suporte a esta funcionalidade pode ser um tanto difícil.

[1] `/var/log/messages` no Conectiva Linux

[2] Boas distribuições do Linux fazem isto automaticamente.

É importante assegurar-se e orientar os usuários para que todas as senhas existentes sejam complexas o suficiente para que não sejam facilmente quebradas. O programa `crack` pode ser usado para quebrar senhas; qualquer senha que ele conseguir quebrar, é por definição uma senha ruim. Apesar do `crack` poder ser executado por intrusos, o administrador deve executá-lo também para inibir senhas ruins. Boas senhas podem ainda ser "forçadas" pelo programa `passwd`; estas senhas são teoricamente mais difíceis de serem quebradas, pois exigem mais esforço computacional na tentativa.

A base de dados dos grupos é mantida no `/etc/group`; para sistemas com shadow, pode existir um arquivo `/etc/shadow.group`.

O `superusuário` normalmente não pode conectar-se via rede ou em determinados terminais, somente nos terminais listados no arquivo `/etc/securetty`. Isso obriga o acesso físico a um desses terminais. É possível ainda conectar-se via qualquer terminal como um outro usuário, e utilizar o comando `su` para tornar-se o `superusuário`.

8.5 Inicialização da Shell

Quando um shell interativo é iniciado, automaticamente são executados um ou mais arquivos predefinidos. Diferentes shells podem executar diferentes arquivos. Para maiores informações, deve ser examinada a documentação de cada shell.

Muitos shells executam inicialmente um arquivo global. O Bourne shell (`/bin/sh`) e seus derivados executam o `/etc/profile`, e adicionalmente executam o arquivo `.profile` no diretório pessoal do usuário. O arquivo `/etc/profile` permite que o administrador do sistema possa definir algumas variáveis de ambiente comum a todos os usuários, especialmente o PATH para inclusão de diretórios de programas e comandos em adição aos diretórios padrões. Por outro lado o arquivo `.profile` permite que o usuário possa customizar o ambiente de acordo com as suas próprias necessidades, e se necessário alterar as opções padrão do ambiente.

Capítulo 9

Gerenciando contas de usuários

As semelhanças entre os administradores de sistemas
e os traficantes de drogas:
ambos medem as coisas em K's e ambos têm usuários.
(velha e surrada piada de informática).

Este capítulo explica como criar novas contas de usuários, como modificar suas propriedades e como removê-las. Diferentes sistemas Linux têm diferentes ferramentas para execução destas tarefas.

9.1 O que é uma conta?

Quando o computador é usado por muitas pessoas é necessário diferenciar os usuários, para, por exemplo, permitir que os arquivos pessoais permaneçam pessoais! Isto é importante mesmo que o computador seja usado somente por um usuário por vez, o que ocorre em muitos microcomputadores.[1] Além disso, cada usuário tem um nome único que é utilizado na sua identificação ao acessar o sistema.

Há mais coisas relacionadas a um usuário do que somente seu nome. Uma **conta** compreende todos os arquivos, recursos e informações pertencentes a um usuário. O termo aproxima-se da sua utilização na área bancária, e como em uma conta em um banco que tem alguma disponibilidade associada, uma conta no Linux tem uma série de recursos associados, que podem ser gastos em diferentes velocidades à medida que se utilize o sistema. Por exemplo, o espaço em disco pode ter uma limitação de uso em megabytes ao dia, e o processamento pode ter um preço por segundo.

[1] Pode ser embaraçoso caso minha irmã possa ler minhas cartas de amor.

9.2 Criando um usuário

O kernel do sistema trata os usuários como meros números. Cada usuário é identificado como um número inteiro único, denominado identificação do usuário (**user id** ou **uid**), porque números são mais simples e rápidos de serem processados do que os nomes dos usuários. Um banco de dados separado do kernel relaciona cada nome, a uma identificação do usuário. O arquivo contém ainda outras informações.

Para criar um usuário, são necessárias algumas informações adicionais sobre o usuário para inclusão na base de dados e criar um diretório pessoal para ele. Pode ser necessário ainda auxiliar o usuário e configurar algum interpretador inicial para ele.

Muitas distribuições do Linux vêm com algum programa para criação de usuários. Há diversos programas disponíveis. Dois programas muito utilizados são o `adduser` e `useradd`[2]; existem ainda utilitários gráficos como o `control-panel`. Independente do programa o resultado é muito similar para todos eles. Todavia este manual descreve como fazê-lo manualmente. Mesmo que os detalhes possam parecer um pouco intrincados, esses programas fazem tudo parecer mais trivial.

9.2.1 `/etc/passwd` e outros arquivos de informação

A base de dados de usuários em um sistema UNIX é um arquivo texto, conhecido por `/etc/passwd`, o qual contém todos os nomes válidos de usuários e as informações associadas a estes. O arquivo contém um nome de usuário por linha e está dividido em sete campos delimitados por dois pontos:

1. Nome do usuário.
2. Senha em formato encriptado.
3. Identificação numérica do usuário (uid).
4. Identificação numérica do grupo (gid).
5. Nome completo ou outra descrição da conta.
6. Diretório pessoal.
7. shell de login.

O formato é explicado em mais detalhes em *passwd*(5).

Qualquer usuário de sistema pode ler o arquivo `passwd`, podendo conhecer por exemplo, o nome de outros usuários do sistema. Isso significa que a senha (segundo campo)

[2] no Conectiva Linux os dois apontam para o mesmo programa

também está disponível para qualquer usuário. As senhas são encriptadas e em teoria estarão seguras. Ainda assim a criptografia pode ser quebrada, especialmente se a senha for muito simples (por exemplo uma senha muito curta ou pertencente ao dicionário). Logo não é aconselhável manter as senhas no arquivo `passwd`.

Muitos sistemas Linux têm a opção do arquivo `shadow` para senhas. Este pode ser um meio alternativo de armazenamento: as senhas criptografadas são armazenadas em um arquivo em separado, o `/etc/shadow`, o qual pode ser lido somente pelo `root`. O arquivo `/etc/passwd` contém somente uma marca especial no campo de senha. Qualquer programa que necessite verificar a senha do usuário deve ser executado pelo `root` ou ter seu `setuid` habilitado, podendo então acessar o `shadow`. Programas normais que necessitem somente usar os demais campos do arquivo `passwd`, não precisam ter acesso às senhas dos usuários.[3]

9.2.2 Escolhendo identificações numéricas para usuário e grupo

Em muitos sistemas não faz diferença qual a identificação do usuário ou do grupo utilizadas, mas caso se esteja utilizando um sistema de arquivos em rede (NFS), será necessário ter o mesmo uid e gid em todos os sistemas. Isso deve-se ao fato do NFS identificar os usuários com uids numéricos. Caso não se esteja utilizando NFS, pode-se permitir que o programa de criação de contas escolha um número de forma automática.

Caso se esteja utilizando o NFS, terá que ser criado um mecanismo de sincronismo de informações de contas. Uma alternativa é utilizar o NIS (veja `network-admin-guide`).

Deve-se evitar a reutilização de uids numéricos e nomes de usuários, porque os novos usuários poderiam ter acesso aos arquivos dos antigos donos da conta.

9.2.3 Interpretador inicial: `/etc/skel`

Quando um diretório pessoal é criado, ele é inicializado com arquivos oriundos do diretório `/etc/skel`. O administrador do sistema pode criar arquivos no `/etc/skel` que podem disponibilizar um bom ambiente para os usuários. Por exemplo, pode-se criar um arquivo `/etc/skel/.profile` que inicializa a variável de ambiente EDITOR com algum editor que seja amigável para novos usuários.

É importante tentar manter o `/etc/skel` o menor possível, uma vez que poderá ser extremamente trabalhoso ter que atualizar todos os arquivos dos usuários já existentes. Por exemplo, se o nome do editor mudar, todos os usuários com a variável inicializada em seus arquivos `.profile` terão que alterá-las. Pode-se tentar fazer isso

[3] Sim, isto quer dizer que o `passwd` tem todas as informações referentes aos usuários, *exceto* sua senha. Esta é a maravilha do desenvolvimento.

automaticamente, mas a possibilidade de gerar algum problema para alguns usuários é muito grande.

Sempre que possível, é melhor colocar alguma configuração genérica em arquivos de configuração global, como por exemplo o `/etc/profile`. Assim é possível realizar atualizações sem alterar diretamente as configurações dos usuários.

9.2.4 Criando um usuário manualmente

Para criar um usuário manualmente, devem ser seguidos os seguintes passos:

1. Edite o arquivo `/etc/passwd` com o utilitário *vipw*(8) e adicione uma nova linha para a nova conta. Atente para a sintaxe do arquivo. *Não edite diretamente com outro editor*, pois o `vipw` trava o arquivo, então outros programas não tentarão atualizar o arquivo ao mesmo tempo. Deve-se informar '*' no campo senha, assim o acesso não será possível.
2. De forma similar, edite o arquivo `/etc/group` com o utilitário `vigr`, caso seja necessário criar também um novo grupo.
3. Crie um diretório pessoal com o comando `mkdir`.
4. Copie os arquivos do diretório `/etc/skel` para o novo diretório pessoal criado.
5. Ajuste as propriedades e permissões através dos comandos `chown` e `chmod`. A opção `-R` pode ser muito útil. As permissões corretas podem variar de um local para outro, mas normalmente os seguintes comandos devem atender:

   ```
   cd /home/novousuário
   chown -R novousuário.grupo .
   chmod -R go=u,go-w .
   chmod go= .
   ```

6. Ajuste a senha através do comando `passwd`.

Após a configuração da senha, a conta estará apta a funcionar. Não se deve alterá-la até que todos os demais passos tenham sido executados. De outra maneira o usuário poderá ter acesso ao sistema enquanto você estiver ainda copiando os arquivos.

Algumas vezes pode ser necessário criar contas anônimas que não são utilizadas por qualquer usuário. Por exemplo para configurar um servidor FTP anônimo (permitindo que qualquer pessoa possa fazer a cópia de arquivos a partir do servidor, sem a obrigatoriedade de ter-se uma conta primeiro) deve-se criar uma conta chamada `ftp`. Neste caso não é necessário configurar uma senha (último passo acima). Ao contrário, é melhor não fazê-lo, pois ninguém poderá usar a conta, a menos que antes se torne o `superusuário`, uma vez que o `root` pode transformar-sem em qualquer usuário.

9.3 Alterando as características de um usuário

Há alguns comandos para alterar as várias características de uma conta (ou seja, os campos relevantes do arquivo `/etc/passwd`):

`chfn`	Altera o campo do nome do usuário.
`chsh`	Muda o shell de login.
`passwd`	Muda a senha.

O superusuário pode usar estes comandos para alterar as propriedades de qualquer conta. Usuários normais podem somente alterar as propriedades de sua própria conta. Algumas vezes é necessário desabilitar estes comandos (com `chmod`) para usuários normais, em ambientes com muitos usuários iniciantes.

Outras tarefas deverão ser executadas manualmente. Por exemplo para alterar o nome de login do usuário é necessário editar o arquivo `/etc/passwd` diretamente (com `vipw`, lembre-se). Da mesma forma, para adicionar ou remover usuários de grupos, é necessário editar o arquivo `/etc/group` (com `vigr`). Tal procedimento tende a ser raro, devendo porém ser realizado com extrema atenção, pois a troca do nome de um usuário poderá fazer com que ele, por exemplo, não receba mais mensagens, a menos que você crie um `alias` para o email antigo.

9.4 Removendo um usuário

Para remover um usuário, deve-se inicialmente remover todos os seus arquivos, caixas de correio, alias de email, trabalhos de impressão, trabalhos programados via `cron` e `at`, e todas as referências que existirem a ele. Após, pode-se remover as linhas relevantes dos arquivos `/etc/passwd` e `/etc/group` (lembre-se de retirar quaisquer referências ao usuário em todos os grupos que ele tenha sido adicionado). É indicado antes de iniciar este processo desabilitar o usuário (veja abaixo), prevenindo assim o uso da conta enquanto ela está sendo removida.

Lembre-se que usuários podem ter arquivos fora do diretório pessoal. O comando find pode encontrá-los:

```
find / -user NomeDoUsuário
```

Note que o comando acima pode levar um tempo significativo, caso seu disco rígido seja grande. Caso se tenha discos montados em rede, (veja o capítulo 2.3.8), tem-se que tomar certas precauções em não carregar toda a rede ou o servidor.

Algumas distribuições do Linux vêm com comandos especiais para esta tarefa; verifique o comando `deluser` ou `userdel`. Ainda que esses comandos possam ser úteis, algumas vezes eles não fazem todo o trabalho necessário e pode ser interessante fazer alguns passos manualmente.

9.5 Desabilitando um usuário temporariamente

Algumas vezes é necessário desabilitar temporariamente uma conta, sem necessariamente removê-la. Por exemplo, o usuário pode não ter pago suas dívidas, ou o administrador do sistema pode suspeitar que a senha daquela conta tenha sido violada.

A melhor forma de desabilitar uma conta é alterar o shell da conta para um programa especial que somente apresente uma mensagem. Desta forma, toda vez que se tente acessar o sistema utilizando esta conta, ocorrerá uma falha, e a mensagem pode indicar, por exemplo, que o administrador do sistema seja contactado.

Pode ainda ser possível alterar o nome do usuário ou a senha para algum outro valor, mas então o usuário não saberá o que está ocorrendo, podendo gerar uma certa confusão.

Uma forma simples de criar estes programas especiais é criar 'tail scripts':

```
#!/usr/bin/tail +2
Esta conta foi fechada devido a questões de segurança.
Por favor chame 555-1234 e aguarde a chegada dos homens de preto.
```

Os primeiros dois caracteres ('`#!`') dizem ao kernel que o restante da linha é um comando que deve ser executado. O comando `tail`, neste caso, imprime, na saída padrão, todo o arquivo, exceto a primeira linha.

Caso o usuário `billg` seja suspeito de um furo de segurança, o administrador do sistema pode fazer algo como:

```
# chsh -s /usr/local/lib/no-login/security billg
# su - billg
Esta conta foi fechada devido a questões de segurança.
Por favor chame 555-1234 e aguarde a chegada do homens de preto.
#
```

O propósito do comando `su` é testar se a mudança funcionou.

Tail scripts devem ser mantidos em diretórios separados, assim os seus nomes não interferirão com os comandos normais de usuários.

Capítulo 10

Cópias de Segurança

Hardware é indeterminadamente confiável.
Software é indeterminadamente confiável.
Pessoas são indeterminadamente confiáveis.
Natureza é determinadamente confiável.

Este capítulo explica porque, como e quando fazer cópias de segurança, e como restaurar arquivos das cópias realizadas.

10.1 A importância de gerar cópias de segurança

Seus dados são valiosos. Custará tempo e muito esforço para recriá-los, e isso significa despesas ou no mínimo desgaste pessoal e lágrimas; algumas vezes, sequer eles podem ser recriados, como por exemplo o resultado de alguns experimentos. Assim sendo, é um bom investimento proteger os dados e ter procedimentos que evitem a sua perda.

Há basicamente quatro razões que podem provocar a perda de dados: falhas de hardware, problemas em programas, ação humana e desastres naturais [1]. Apesar dos hardwares modernos serem mais confiáveis, eles ainda podem ter panes espontaneamente e sem aviso prévio. O componente mais crítico de hardware para armazenamento de dados é o disco rígido, o qual é baseado em que pequenos campos magnéticos permaneçam intactos em um mundo repleto de ruídos eletromagnéticos. Softwares modernos não tendem a ser mais confiáveis, um programa estável como uma rocha é uma exceção e não uma regra. Humanos são também pouco confiáveis, quer pelos erros que possam cometer, quer por tentativas mal intencionadas de destruir os dados. A natureza pode não ser diabólica, porém pode gerar danos, mesmo quando está sendo boa (chuvas, excesso de calor,...). No final temos um pequeno milagre onde as coisas funcionam bem.

[1] A quinta razão é "qualquer outra coisa"

Cópias de segurança são uma forma de proteger os investimentos em dados. Tendo-se diversas cópias dos dados, será de menor importância se um deles for perdido, pois o custo será a restauração dos dados de uma das cópias.

É importante fazer-se cópias de maneira adequada. Como tudo o mais relacionado ao mundo físico, os backups podem falhar mais cedo ou mais tarde. Parte do trabalho da geração das cópias de segurança é estar seguro de que elas funcionem; ninguém gostaria de perceber que suas cópias não funcionam[2]. Problemas acontecem, pode haver uma pane de discos bem no meio do backup. Caso se tenha somente uma mídia para o backup ela também pode sofrer danos, sobrando somente a lembrança do grande trabalho realizado. Ou ainda, você pode notar, quando está fazendo a recuperação dos dados, que esqueceu de copiar o banco de dados de usuários com mais de 15.000 registros de clientes, que foi perdido. Mais ainda, as fitas podem estar funcionando perfeitamente, porém a última unidade da face da Terra que consegue ler o tipo de fita que você usava, está cheia de água no depósito do escritório.

Quando o tema é cópia de segurança, paranóia é a melhor descrição deste trabalho.

10.2 Selecionando o dispositivo de backup

A decisão mais importante sobre cópias de segurança é a opção da mídia a ser utilizada. É necessário considerar custos, confiabilidade, velocidade, disponibilidade e utilidade.

A questão custos é importante, uma vez que é preferível ter-se muitas vezes mais capacidade de armazenamento das cópias do que a quantidade de dados existentes. Uma mídia econômica é usualmente uma necessidade.

Confiabilidade é um ítem de extrema importância, pois uma cópia com problemas pode fazer um homem adulto chorar. Uma mídia para backups deve ser capaz de manter os dados em perfeito estado durante anos. A forma de usar a mídia afeta a sua confiabilidade. Um disco rígido é normalmente muito confiável, mas como mídia para cópias de segurança pode não sê-lo, caso esteja no mesmo computador que os discos que você está gerando as cópias.

Velocidade não é uma questão muito importante, caso a cópia possa ser realizada de forma não interativa. Não importa que uma cópia seja realizada em duas horas, desde que ela possa ser feita sem assistência. Por outro lado, caso a cópia não possa ser realizada quando o computador esteja disponível, então velocidade é um tema importante.

Disponibilidade é, obviamente, necessário, pois não se pode usar uma mídia caso ela não exista. Menos óbvia é a disponibilidade futura e em outros equipamentos

[2]Não ria. Isso acontece todos os dias.

diferentes dos que ela foi gerada. De outra forma pode não ser possível restaurar os dados após algum acidente.

Utilidade é um fator abrangente e que se relaciona à periodicidade em que as cópias são realizadas. A forma mais simples é realizar cópias da melhor maneira possível, isto é no menor período e com o maior conjunto de dados possíveis. Uma mídia não deve ser difícil ou trabalhosa de ser usada.

As alternativas típicas são disquetes e fitas. Disquetes são extremamente baratos, relativamente confiáveis, não muito rápidos, de alta disponibilidade, mas não muito úteis para grandes quantidades de dados. Fitas são baratas e algumas nem tanto assim, relativamente confiáveis e velozes, de média disponibilidade, e, dependendo do tamanho da fita, bastante confortáveis.

Há algumas outras alternativas. Não muito comuns, mas caso isso não seja um problema, podem ser uma melhor opção em muitos casos. Por exemplo, discos magneto-óticos podem ser melhores que as duas opções anteriores: têm acesso randômico, tornando a restauração mais simples e rápida como os disquetes, e podem conter uma grande quantidade de dados como as fitas.

Acrescentaríamos ainda como opções viáveis: discos rígido em outros equipamentos, CD graváveis, zip drives e similares.

10.3 Selecionando a ferramenta de backup

Há diversas ferramentas que podem ser utilizadas na geração de cópias de segurança. Algumas das ferramentas tradicionais do UNIX são os utilitários `tar`, `cpio` e `dump`. Adicionalmente, uma série de pacotes de terceiros (alguns de livre distribuição, outros comerciais) podem ser utilizados. A opção da mídia pode afetar na escolha da ferramenta.

O `tar` e `cpio` são similares, e equivalentes do ponto de vista do backup. Ambas são capazes de guardar dados em fitas e restaurá-los. Podem gravar dados em praticamente qualquer mídia, pois os controladores de dispositivos tomam conta dos acessos de baixo nível ao equipamento e todos os dispositivos tendem a ser parecidos para os programas. Algumas versões destes programas em alguns sistemas UNIX podem ter dificuldades com arquivos incomuns (links simbólicos, arquivos de dispositivos, arquivos com nomes muito longos e assim por diante), porém as versões do Linux lidam com isso corretamente.

O `dump` é um pouco diferente pois lê o sistema de arquivos diretamente, sem uso de mecanismos de indireção. Ele foi desenvolvido especificamente para a geração de cópias de segurança; `tar` e `cpio` são ferramentas para arquivamento, apesar de funcionarem também para backups.

Ler diretamente o sistema de arquivos traz algumas vantagens. Isso torna possível gerar as cópias sem afetar a data de acesso dos arquivos; no caso do `tar` e `cpio` é necessário antes montar o sistema de arquivos com permissões somente de leitura. É também mais eficiente, caso seja necessário copiar todo o sistema de arquivos, uma vez que muito menos movimento das cabeças de leitura e gravação do disco rígido será exigido. A maior desvantagem é que o dump é um programa de cópia específico para somente um tipo de sistema de arquivos; o dump do Linux entende somente os sistemas de tipo ext2.

O `dump` suporta ainda níveis de cópias, ao passo que `tar` e `cpio` têm essa funcionalidade implementada por outras ferramentas.

Uma comparação entre softwares fornecidos por terceiros está além do escopo deste livro, porém uma lista de softwares de livre distribuição pode ser encontrada no "Linux Software Map".

10.4 Cópias simples

Um esquema simples de cópia de segurança é copiar tudo uma única vez e após copiar somente os arquivos que sofreram alterações após a cópia inicial. A primeira cópia é denominada **cópia total** e as seguintes **cópias incrementais**. Uma cópia total é normalmente mais trabalhosa que a incremental, uma vez que muito mais dados são gravados e pode eventualmente não caber em uma única fita, disquete, ou qualquer que seja a mídia utilizada. A restauração de uma cópia incremental pode ser muitas vezes mais trabalhosa do que de uma cópia total. A restauração pode ser otimizada desde que sempre se copie tudo o que foi alterado desde a última cópia total; desta forma as cópias podem ser um pouco mais trabalhosas e demoradas, mas nunca será necessário restaurar mais que duas cópias (uma total e outra incremental).

Caso você deseje realizar cópias diárias e tenha seis fitas, pode-se fazer uma cópia total inicialmente (digamos na sexta-feira), e utiliza-se as fitas 2 a 5 para cópias incrementais (de segunda-feira a quinta-feira). Faz-se então uma nova cópia total com a fita 6 na segunda sexta-feira, e reinicia-se o ciclo incremental com as fitas 2 a 5. Não é necessário regravar a fita 1 até que uma nova cópia total seja gerada. Após a geração da cópia total na fita 6, pode-se manter a fita 1 em algum outro local, assim se o conjunto de fitas for perdido em um incêndio, por exemplo, ao menos algo será restaurado. Ao se fazer uma nova cópia total, utiliza-se a fita 1 e a fita 6 toma o seu lugar.

Caso se disponha de mais de 6 fitas, pode-se usar as fitas extras para cópias completas. A cada vez que seja feito uma nova cópia total, utiliza-se sempre a mais antiga. Desta forma pode-se ter cópias de várias semanas anteriores, o que pode ser útil caso se queira encontrar um arquivo antigo que foi apagado ou uma versão antiga de algum arquivo.

10.4.1 Cópias de segurança com `tar`

Uma cópia de segurança completa pode ser facilmente gerada com o `tar`:

```
# tar -create -file /dev/ftape /usr/src
tar:  Removing leading / from absolute path names in the archive
#
```

O exemplo acima utiliza a versão GNU do `tar` e sua característica para opções com nomes longos. A versão tradicional do `tar` somente entende opções descritas através de um único caracter. A versão GNU pode ainda lidar com cópias que não caibam em uma única fita ou disquete, assim como com caminhos longos; nem todas as versões podem fazer isso (o Linux somente utiliza o GNU `tar`).

Caso a cópia não caiba em uma única fita, é necessário ativar a opção `multi-volume` (`-M`):

```
# tar -cMf /dev/fd0H1440 /usr/src
tar:  Removing leading / from absolute path names in the archive
Prepare volume #2 for /dev/fd0H1440 and hit return:
#
```

Observe que deve-se formatar os disquetes antes de iniciar a cópia, ou ainda usar outra janela ou terminal virtual e fazê-lo assim que o `tar` solicitar um novo disquete.

Após a geração da cópia, deve-se verificar se tudo está em perfeitas condições. Pode-se fazer isso utilizando a opção `-compare` (`-d`):

```
# tar -compare -verbose -f /dev/ftape
usr/src/
usr/src/linux
usr/src/linux-1.2.10-includes/
....
#
```

Não executar a checagem significa que não se perceberá que as cópias não funcionam até que elas sejam necessárias.

Uma cópia incremental pode ser feita com o comando `tar` utilizando-se a opção `-newer` (`-N`):

```
# tar -create -newer '8 Sep 1998' -file /dev/ftape /usr/src -verbose
tar:  Removing leading / from absolute path names in the archive
usr/src/
usr/src/linux-2.0.36/
usr/src/linux-2.0.36/include/
usr/src/linux-2.0.36/include/asm
usr/src/linux-2.0.36/include/asm-generic/
usr/src/linux-2.0.36/include/asm-i386/
usr/src/linux-2.0.36/include/asm-i386/bugs.h
usr/src/linux-2.0.36/include/asm-i386/byteorder.h
usr/src/linux-2.0.36/include/asm-i386/pgtable.h
usr/src/linux-2.0.36/include/linux/
....
#
```

Infelizmente, o `tar` não percebe quando somente uma informação no inode foi modificada, como por exemplo, alteração de permissões de acesso, ou quando somente o nome foi alterado. Isso pode ser feito usando-se o comando `find` e comparando o sistema de arquivos atual com a lista de arquivos copiados previamente. Scripts e programas que executam essas tarefas podem ser encontrados em sites FTP do Linux.

10.4.2 Restaurando arquivos com `tar`

Deve-se usar a opção `-extract` (`-x`) para restaurar arquivos copiados com o `tar`.

```
# tar -extract -same-permissions -verbose -file /dev/fd0H1440
usr/src/
usr/src/linux-2.0.36/
usr/src/linux-2.0.36/include/
usr/src/linux-2.0.36/include/asm
usr/src/linux-2.0.36/include/asm-generic/
usr/src/linux-2.0.36/include/asm-i386/
usr/src/linux-2.0.36/include/asm-i386/bugs.h
usr/src/linux-2.0.36/include/asm-i386/byteorder.h
usr/src/linux-2.0.36/include/asm-i386/pgtable.h
usr/src/linux-2.0.36/include/linux/
....
#
```

Pode-se ainda extrair somente arquivos e diretórios específicos (inclusive seus subdiretórios), bastando nominá-los na execução do comando:

```
# tar xpvf /dev/fd0H1440 usr/src/linux-2.0.36/include/asm-i386/bugs.h
usr/src/linux-2.0.36/include/asm-i386/bugs.h
#
```

Pode-se usar a opção `-list` (`-t`), caso se queira saber quais são os arquivos presentes na cópia:

```
# tar -list -file /dev/fd0H1440
usr/src/
usr/src/linux-2.0.36/
usr/src/linux-2.0.36/include/
usr/src/linux-2.0.36/include/asm
usr/src/linux-2.0.36/include/asm-generic/
usr/src/linux-2.0.36/include/asm-i386/
usr/src/linux-2.0.36/include/asm-i386/bugs.h
usr/src/linux-2.0.36/include/asm-i386/byteorder.h
usr/src/linux-2.0.36/include/asm-i386/pgtable.h
usr/src/linux-2.0.36/include/linux/
....
#
```

Note que o `tar` sempre lê o volume seqüencialmente, tornando-se um pouco lento em grandes volumes. De qualquer forma, ao se utilizar mídias seqüenciais como unidades de fita, não é possível utilizar acessos randômicos.

O `tar` não consegue lidar adequadamente com arquivos apagados. Caso se necessite restaurar um sistema de arquivos de um backup total e de um incremental, e algum arquivo foi apagado entre as duas cópias, ele estará presente após a restauração. Isso pode ser um grande problema, caso o arquivo contenha dados sensíveis que não mais deveriam estar disponíveis.

10.5 Cópias em diversos níveis

O método de cópia simples descrito na seção anterior é muitas vezes bastante adequado para uso pessoal ou pequenas corporações. Para tarefas mais complexas o uso de cópias multinível pode ser mais adequado.

O método simples tem dois níveis: cópia completa e cópias incrementais. Isso pode ser generalizado para qualquer número de níveis. Uma cópia completa pode ser de nível 0, e os diferentes níveis incrementais podem ser 1, 2, 3, ... e assim por diante. A cada cópia incremental é copiado tudo o que foi alterado desde a cópia anterior de mesmo nível ou menor.

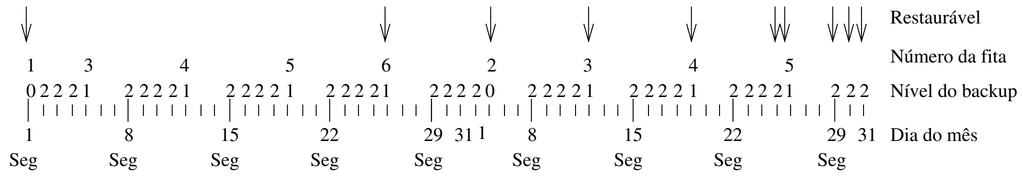

Figura 10.1: Exemplo de esquema de cópia de segurança multinível.

O propósito deste procedimento é permitir um maior histórico de cópias de forma econômica. No exemplo da seção anterior o histórico das cópias remontava à última cópia total. Isso pode ser estendido ao se ter mais fitas disponíveis, e mesmo com somente uma nova fita por semana, pode ser um pouco dispendioso. Um histórico maior pode ser útil, uma vez que arquivos corrompidos ou apagados podem passar desapercebidos por longos períodos. Mesmo uma versão não muito atualizada de um arquivo pode ser melhor que nenhuma.

Com cópias de segurança multinível, o histórico pode ser estendido de forma mais econômica. Por exemplo, caso se tenha 10 fitas disponíveis, pode-se usar as fitas 1 e 2 para cópias mensais (primeira sexta-feira de cada mês), fitas 3 a 6 para cópias semanais (as demais sextas-feiras do mês; sendo mais quatro unidades uma vez que um mês pode conter até 5 sextas-feiras); e 7 a 10 para cópias diárias (de segunda-feira a quinta-feira). Adicionando-se somente quatro fitas, pode-se estender o histórico de cópias de duas semanas para dois meses. É verdade que não se pode restaurar todas as versões de cada arquivo durante dois meses, mas o que é possível restaurar é normalmente bom o suficiente.

A figura 10.1 mostra qual o nível da cópia usado em cada dia, e quais cópias podem ser restauradas a partir do fim do mês.

Níveis de backups podem também ser usados para minimizar o tempo de restauração de sistemas de arquivos. Caso se tenha muitas cópias incrementais com um número de níveis muito grande, pode-se necessitar restaurar todos eles para reconstruir um sistema de arquivos. Ao invés disso pode-se usar níveis, e manter o número de cópias necessárias mais baixo para a restauração.

Para minimizar o número de fitas necessárias para a restauração, pode-se usar um nível menor para cada fita incremental. Porém o tempo necessário para realizar as cópias cresce (cada backup copia tudo que tenha sido alterado desde a última cópia total). Um esquema melhor é sugerido na página de manual do `dump` e apresentado na tabela 10.2. Use a seguinte ordem para os níveis de backup: 3, 2, 5, 4, 7, 6, 9, 8, 9... Isto mantém tanto o tempo de realização do backup e da restauração curtos. O máximo que se copia são dois dias. O número de fitas a serem restauradas depende do intervalo entre as cópias completas, mas certamente será menor que nos esquemas mais simples.

Uma outra estratégia pode reduzir a quantidade de trabalho necessário, mas não

Figura 10.2: Estratégia de backup utilizando diferentes níveis de cópias

Fita	Nível	Cópia (dias)	Restauração fitas
1	0	n/a	1
2	3	1	1, 2
3	2	2	1, 3
4	5	1	1, 2, 4
5	4	2	1, 2, 5
6	7	1	1, 2, 5, 6
7	6	2	1, 2, 5, 7
8	9	1	1, 2, 5, 7, 8
9	8	2	1, 2, 5, 7, 9
10	9	1	1, 2, 5, 7, 9, 10
11	9	1	1, 2, 5, 7, 9, 10, 11
...	9	1	1, 2, 5, 7, 9, 10, 11, ...

significa que há menos coisas para acompanhar. Você deve decidir se isso vale ou não a pena.

O `dump` tem suporte a cópias em multinível, já os comandos `tar` e `cpio` podem ter essa facilidade implementada através de scripts.

10.6 O que copiar

Pode-se desejar copiar tudo o que for possível. A maior exceção são softwares que podem ser facilmente restaurados, mas mesmo eles podem conter arquivos de configuração que devem ser copiados, poupando a tarefa de reconfigurá-los. Outra exceção é o sistema de arquivos `/proc`; uma vez que ele contém somente informações que o kernel gera automaticamente, *não* é uma boa idéia copiá-lo, especialmente o arquivo `/proc/kcore` que é somente uma cópia da memória física, além de ser bastante grande.

Áreas cinzentas incluem filas de impressão, arquivos de logs e muitas outras coisas no `/var`. Você deve decidir o que é importante na sua máquina.

As coisas óbvias que devem ser copiadas são os arquivos de usuários (`/home`) e os arquivos de configuração do sistema (`/etc`, mas certamente haverá outras coisas espalhadas por todos os sistemas de arquivos).

10.7 Arquivos compactados

Cópias de segurança costumam utilizar muito espaço, o que pode ser dispendioso. Para reduzir o espaço necessário, as cópias podem ser compactadas. Há diversas formas de se fazer isso. Alguns programas têm suporte à compactação; por exemplo a opção `-gzip` (`-z`) do `tar` da GNU passa o arquivo gerado para o programa de compactação `gzip`, antes de gravar o arquivo na mídia.

Infelizmente arquivos compactados podem causar certos problemas, dada a natureza de como a compactação funciona. Caso um simples bit esteja com problemas, todo o restante do arquivo compactado pode estar inutilizado. Alguns programas têm facilidades de correção, mas não há nenhum método de lidar com um grande número de erros. O que significa que se uma cópia é compactada através da maneira que o `tar` faz, com a saída sendo um único grande arquivo compactado, um simples erro pode tornar todo o restante da cópia inútil. Cópias de segurança têm que ser confiáveis, e este método de compactação não parece ser uma boa idéia.

Uma forma alternativa é compactar os arquivos separadamente. Isso significa que se um arquivo for perdido, os demais não sofrerão os efeitos colaterais. O arquivo perdido poderia de qualquer forma conter algum tipo de erro, então essa situação não parece ser menos favorável do que não utilizar nenhuma compactação. O programa `afio` (uma variante do `cpio`) pode trabalhar desta forma.

A compactação leva algum tempo, o que pode tornar o programa de cópia incapaz de gravar os dados com a velocidade de uma unidade de fita [3]. Isso pode ser evitado através do armazenamento temporário da saída (buffers de memória), de forma interna se o programa de backup tem essa funcionalidade disponível, ou através da utilização de outro programa. Isso somente se aplica a computadores muito lentos ou sobrecarregados durante a geração da cópia de segurança.

[3] Se a unidade de fita não recebe os dados à velocidade que pode tratar, ela pára; isto torna o backup mais lento ainda e pode ser ruim para a fita e para a unidade também.

Capítulo 11

Manutenção de Data/Hora

Terra,
Vagabundeando, matas o tempo,
Bailarina láctea, ébria,
Giras com teus minúsculos hóspedes,
Debruçados em tuas costas...
A Nova Poesia Brasileira - Sandro N. Henrique - 1983

Este capítulo explana como o Linux mantém a data e a hora, e o que é necessário para evitar problemas. Usualmente, não é necessário executar qualquer atividade, mas é importante entender o seu funcionamento.

11.1 Fusos horários

A medição do tempo é baseada no fenômeno regular de alternância de períodos de luz e escuridão causada pela rotação do planeta. O tempo total decorrido entre os períodos é constante, porém os períodos de luz e escuridão podem variar. Uma constante simples é o meio-dia.

O meio-dia é a hora do dia em que o Sol está em sua posição culminante. Uma vez que a terra é redonda[1], o meio-dia acontece em diferentes horários em diferentes lugares. Isso leva ao conceito de **horário local**. O homem mede o tempo em diversas unidades, muitas das quais estão baseadas em fenômenos naturais como o meio-dia. Caso você permaneça em um mesmo local menor importância tem as diferenças entre os horários locais.

Assim que você precise comunicar-se com outros locais distantes, notará a necessidade de um horário comum. Nos tempos modernos, muitos locais no mundo passaram a

[1] Segundo pesquisas recentes.

comunicar-se entre si, sendo necessário um padrão mundial de medida de tempo. Este padrão é conhecido como **horário universal** (UT ou UTC, anteriormente conhecido como o tempo medido em Greenwich ou GMT (Greenwich Mean Time), pois é baseado no horário local de Greenwich, uma localidade da Inglaterra). Quando alguém com um diferente horário local necessita comunicar-se, ele pode expressar o tempo através do horário universal, pois desta maneira não há confusão de quando determinado encontro deve acontecer.

Cada horário local é chamado de fuso horário. Enquanto que geograficamente seja possível que vários locais tenham o meio-dia ao mesmo tempo e portanto tenham a mesma zona de tempo, politicamente nem sempre isso é possível. Por diversas razões países adotam **horários de verão** para atender a demandas econômicas. Alguns países não adotam estes procedimentos. Os países que utilizam o horário de verão não têm um padrão de quando os relógios devem ser adiantados ou atrasados, mudando as regras de ano para ano. Isso faz com que algumas conversões de fusos horários definitivamente não sejam triviais.

Os fusos horários são melhor denominados através da localização ou da diferença entre o horário local e o universal. Nos Estados Unidos e em alguns outros países, os fusos horários locais têm um nome e três letras na sua abreviação. As abreviações não são únicas, porém não deveriam ser utilizadas desacompanhadas do nome do país. É melhor referir-se ao horário, digamos em Helsinki, do que no horário do Leste Europeu, uma vez que nem todos os países daquela região adotam as mesmas regras.

O Linux reconhece todos os fusos horários existentes, e estes podem ser mudados facilmente à medida que as regras mudem. Tudo o que o administrador de sistemas necessita fazer é selecionar o fuso horário apropriado. Cada usuário também pode configurar seu fuso horário próprio, o que pode ser importante uma vez que muitas pessoas trabalham com computadores através da Internet. Quando as regras do horário de verão mudam, é importante ajustar estas regras, no seu sistema local. Não há muito com o que se preocupar com a contagem de tempo em seu sistema, além de ajustar o fuso horário e atualizar eventualmente as bases de dados.

11.2 Os relógios de hardware e software

Um computador pessoal tem um relógio mantido por uma bateria. Essa bateria garante que o relógio funcionará mesmo que o restante do computador esteja desligado ou não haja eletricidade disponível. O relógio de hardware pode ser ajustado através do painel de configuração da BIOS ou através do que esteja rodando como sistema básico em seu computador.

O kernel do Linux acompanha o tempo independentemente do relógio de hardware. Durante a inicialização, o Linux configura seu próprio relógio para o mesmo horário que o relógio de hardware. Após isso o relógio do sistema anda independentemente. O Linux mantém o seu próprio relógio porque é complicado e lento verificar o horário no relógio de hardware a todo instante.

O relógio do kernel apresenta sempre o horário universal. Desta forma ele não tem que conhecer absolutamente nada sobre os fusos horários, fazendo com que a simplicidade torne o horário do sistema mais confiável e fácil de atualizar. Cada processo lida com as conversões de fusos horários de forma independente (utilizando ferramentas padrões que fazem parte do pacote de fusos horários).

O relógio de hardware pode estar nos formatos de horário local e horário universal. Normalmente, é melhor mantê-lo no formato universal, porque não será necessário atualizar o relógio de hardware quando o horário de verão começar ou terminar (UTC não tem horário de verão). Infelizmente alguns sistemas operacionais para PCs, incluindo MS-DOS, Windows, OS/2, assumem que o hardware mostra o horário local. O Linux pode trabalhar com qualquer um deles, mas caso o relógio de hardware esteja configurado para o horário local, deve-se alterá-lo quando o horário de verão inicia ou termina.

11.3 Mostrando e acertando o relógio

No Conectiva Linux, o fuso horário é definido por um link simbólico com o arquivo `/etc/localtime`. Este link aponta para um arquivo de dados de fuso horário que descreve o fuso horário local. Os arquivos de dados de fuso horário estão armazenados em `/usr/share/zoneinfo`. Outras distribuições podem apresentar diferenças.

Um usuário pode alterar o seu fuso horário privado através da variável de ambiente TZ. Caso ela não esteja inicializada, o fuso horário do sistema é assumido. A sintaxe da variável TZ é descrita na página de manual do comando *tzset*(3).

O comando `date` apresenta a data atual e o horário corrente [2]. Por exemplo:

```
$ date
sex out  9 17:39:42 EDT 1998
$
```

Isso significa Sexta, 9 de outubro de 1998, aproximadamente 20 minutos para as 6 horas da tarde, e o fuso horário "EDT" significa Horário do Leste (Eastern Daylight). O comando `date` pode apresentar ainda o horário universal:

```
$ date -u
```

[2] Note que o comando `time` não mostra a hora corrente. Ele é útil quando se quer saber quanto tempo um determinado programa gastou na sua execução

```
sex out  9 20:26:11 UTC 1998
$
```

O `date` também é usado para configurar o horário do relógio do kernel do sistema:

```
# date 10091835
sex out  9 18:35:00 EDT 1998
# date
sex out  9 18:35:00 EDT 1998
#
```

Veja a página de manual do comando `date` para maiores detalhes - a sintaxe pode ser um pouco complexa. Somente o `root` pode ajustar o relógio do sistema. Enquanto cada usuário pode ter o seu próprio fuso horário, o relógio é o mesmo para todos.

O `date` apresenta somente as configurações do relógio do sistema. O comando `clock` sincroniza os relógios de software e hardware. Ele é utilizado quando o sistema é inicializado, para ler o relógio do hardware e atualizar o relógio do sistema. Caso seja necessário ajustar ambos os relógios, primeiro deve-se ajustar o relógio do software através do comando `date` e após o relógio de hardware através do comando `clock -w`.

A opção `-u` diz ao `clock` que o relógio de hardware está no formato universal. *Deve-se* usar a opção `-u` com cuidado, caso contrário o computador ficará um tanto confuso sobre o horário correto.

O relógio deve ser alterado com cuidado. Muitas partes de um sistema UNIX necessitam do relógio para funcionarem corretamente. Por exemplo, o servidor `cron` executa comandos periodicamente. Caso o relógio seja alterado, ele pode ficar um pouco confuso sobre a necessidade ou não de executar determinado comando. Em algum UNIX mais antigo, quando alguém mudou o relógio para vinte anos no futuro, o cron tentou executar todos os comandos periódicos dos próximos vinte anos de uma única vez. Versões atualizadas do cron podem lidar com isso corretamente, mas ainda deve-se ter muito cuidado. Grandes saltos ou atrasos no relógio podem ser mais perigoso que pequenas diferenças.

11.4 Quando o relógio está errado

O relógio de software do Linux não é sempre preciso. Ele é mantido por **interrupções de tempo** periódicas geradas pelo hardware. Caso o sistema tenha muitos processos sendo executados, pode levar um intervalo de tempo maior até que a interrupção de tempo seja gerada, e o relógio de software pode estar ligeiramente atrasado. O relógio de hardware é executado de forma independente do sistema e estará portanto funcionando de maneira mais precisa. Caso o sistema seja inicializado freqüentemente (e neste caso ele provavelmente não esteja funcionando como um servidor), o relógio deve estar praticamente correto.

Caso seja necessário ajustar o relógio de hardware, o mais simples será reinicializar o computador, acessar a tela da configuração da BIOS e ajustá-lo. Isso evita qualquer problema que a mudança de horário possa causar ao sistema. Caso executar a mudança via BIOS não seja uma alternativa viável, ajuste o novo horário com os comandos `date` e `clock` (nesta ordem), porém prepare-se para reinicializar o sistema, caso coisas estranhas comecem a ocorrer.

Um computador conectado em rede (mesmo que somente através de um modem) pode checar o relógio automaticamente, comparando-o com o horário de outros computadores. Caso o outro computador seja conhecido pela acuidade de seu horário, então ambos podem manter um horário ajustado. Isso pode ser feito através dos comandos `rdate` e `netdate`. Ambos verificam o horário de um computador remoto (o `netdate` pode lidar com diversos computadores remotos), e podem ajustar o horário do computador local. Ao executar um desses comandos regularmente, o computador local terá um horário tão ajustado quanto o computador remoto.

Apêndice A

Medidas de Segurança

Este apêndice contém a parte essencial do programa para medir potenciais 'buracos' em um sistema de arquivos. O fonte do livro contém os fontes completos do programa (`sag/measure-holes/measure-holes.c`).

```
int process(FILE *f, char *filename) {
        static char *buf = NULL;
        static long prev_block_size = -1;
        long zeroes;
        char *p;

        if (buf == NULL || prev_block_size != block_size) {
                free(buf);
                buf = xmalloc(block_size + 1);
                buf[block_size] = 1;
                prev_block_size = block_size;
        }
        zeroes = 0;
        while (fread(buf, block_size, 1, f) == 1) {
                for (p = buf; *p == '\0'; )
                        ++p;
                if (p == buf+block_size)
                        zeroes += block_size;
        }
        if (zeroes > 0)
                printf("%ld %s\n", zeroes, filename);
        if (ferror(f)) {
                errormsg(0, -1, "read failed for `%s'", filename);
                return -1;
        }
        return 0;
}
```

Apêndice B

Glossário

A Biblioteca da Universidade de Unseen
decidiu unilateralmente ajudar o entendimento
produzindo um dicionário Orangotango/Humano.
Ele está trabalhando nisso por três meses.

Não foi fácil e ele chegou até 'Oook'.

(Terry Pratchett, "Men At Arms")

Esta é uma pequena lista de palavras e definições relacionadas com o Linux e com a administração de sistemas. As referências de páginas são para a primeira ou a mais importante página onde a palavra é usada.

ambição O ato de escrever frases interessantes na esperança de vê-las nos arquivos Linux.

aplicações (p. 7) Software que executa alguma atividade útil. Os resultados de um programa aplicativo são a razão primeira de usar-se um computador. Veja também programas de sistema e sistema operacional.

chamada ao sistema (p. 7) Os serviços disponibilizados pelo kernel para os aplicativos, e a forma como eles são chamados. Veja a seção 2 das páginas de manual.

glossário Lista de palavras sobre um determinado tema e suas respectivas explanações sobre seus conceitos. Não confundir com um dicionário que pode ser mais abrangente.

kernel (p. 7) Parte do sistema operacional que implementa a interação entre o hardware e o compartilhamento de recursos. Veja também programas de sistema.

programas de sistema (p. 7) Programas que implementam funcionalidades de alto nível de um sistema operacional, como por exemplo, coisas que não dependem diretamente do hardware. Podem requerer algumas vezes privilégios especiais para sua execução (por exemplo, a entrega de correio eletrônico), mas que freqüentemente agem como parte do sistema operacional (por exemplo um compilador). Veja também aplicativos, kernel, sistema operacional.

servidor Um processo executado em segundo plano, normalmente sem ser notado, até que algum evento acione a sua ação. Por exemplo, o programa de atualização acorda a cada trinta segundos e esvazia o cache de dados, e o servidor de correio eletrônico (`sendmail`) acorda toda vez que uma mensagem é enviada.

sistema de arquivos (p. 39) Os métodos e as estruturas de dados que um sistema operacional utiliza para manter controle dos arquivos em um disco ou partição; a forma pela qual os dados estão organizados no disco. Também usado para se referenciar a uma partição ou disco que é usado para armazenar os arquivos ou determinado tipo de sistema de arquivos.

sistema operacional (p. 7) Software capaz de administrar os recursos do computador (processador, memória, espaço em disco, banda de rede, e assim por diante) entre os diversos usuários e os programas aplicativos que eles executam. Controla o acesso ao sistema para disponibilizar segurança. Veja também kernel, programas de sistema e aplicativos.

Apêndice C

Como Fazer um Disco de Inicialização Linux

Este documento descreve como definir e construir o seu próprio disquete de inicialização e do sistema de arquivos raiz para o Linux. Estes discos podem ser usados como disquetes de emergência ou no teste de novos componentes do sistema. Caso não se tenha lido o FAQ do Linux e outros relacionados, como o Tutorial de Instalação Linux e o Guia de Instalação Linux, não é indicada a construção de discos de inicialização. Caso se deseje somente criar discos de emergência, veja o Anexo C.11.1 (Discos de inicialização pré-configurados).

C.1 Prefácio

Nota: este documento pode não estar atualizado. Verifique na página do Projeto de Documentação Linux `<http://sunsite.unc.edu/LDP/>` se há alguma versão mais recente deste documento.

Apesar deste documento estar em formato texto, ele pode ser visualizado de maneira muito *mais* agradável em formato Postscript (`.ps`) ou HTML, em virtude dos comandos de notação tipográfica utilizados. Nós sugerimos a utilização de um destes formatos.

C.1.1 Notas

Graham Chapman (`grahamc@zeta.org.au`) escreveu a versão original e a suportou até a versão 3.1. Tom Fawcett (`fawcett@croftj.net`) adicionou uma grande quantidade de informações do Núcleo 2.0 e é o mantenedor atual da versão 3.2, porém muito da versão original ainda permanece.

Este documento está voltado para usuários da versão **2.0 do Kernel do Linux ou posterior**. Caso se esteja utilizando uma versão anterior (1.2.xx ou mais antiga), é aconselhável utilizar versões do Bootisk-HOWTO arquivadas na *Página Pessoal de Graham Chapman* `<http://www.zeta.org.au/~grahamc/linux.html>`.

As informações aqui apresentadas estão voltadas para os usuários da plataforma **Intel**, porém muito do aqui descrito aplica-se ainda a outros processadores, porém não podemos ser conclusivos a respeito disso. Caso alguém tenha essa experiência, por favor nos contate.

C.1.2 Opiniões e Créditos

Quaisquer comentários ou sugestões, positivas ou negativas, sobre o conteúdo deste documento são bem-vindas. Fizemos o melhor possível para garantir que as informações aqui contidas sejam confiáveis e acuradas. Por favor avise-nos caso sejam encontrados erros ou omissões.

Agradecemos às inúmeras pessoas que enviaram as suas sugestões e correções. A sua contribuição certamente tornou este documento muito melhor, do que se tivéssemos feito tudo sozinhos.

Por favor envie comentários, correções e dúvidas ao autor no email apresentado mais acima. Não me incomodo em responder dúvidas, porém por favor leia primeiramente a seção C.7 (Problemas).

C.1.3 Política de Distribuição

C.2 Introdução

Discos de inicialização do Linux são úteis em diversas situações, tais como:

- Testes de um novo kernel.

- Recuperação de falhas de disco – qualquer coisa que pode variar desde um setor de inicialização a uma quebra de cabeça de disco.
- Consertando um sistema defeituoso. Um pequeno erro como superusuário pode tornar o sistema sem condições de uso, e pode ser necessário inicializar o sistema a partir um disquete, para consertá-lo.
- Atualizar arquivos críticos do sistema, tais como o `libc.so`.

Há várias formas de se obter discos de inicialização (boot):

- Usar um de uma distribuição como Slackware, Red Hat, Conectiva Linux, o qual permitirá que ao menos o sistema seja inicializado.
- Usar um pacote desenvolvido para a criação de discos de emergência.
- Aprender o que é necessário para cada tipo de disco operar e construir um por conta própria.

Algumas pessoas escolhem a última opção, construindo os seus próprios discos de emergência. Desta forma, na ocorrência de algum problema, eles podem saber o que deve ser corrigido. Mais ainda, é uma grande maneira de aprender como o Linux funciona.

Este documento assume alguma familiaridade com os conceitos básicos de administração de sistemas Linux. Por exemplo, deve-se conhecer os conceitos de diretórios, sistemas de arquivos e disquetes. Deve-se saber ainda utilizar os comandos `mount` e tt/df/, bem como o significado dos arquivos `/etc/passwd` e `fstab` e como eles estão constituídos, assim como saber que a maioria dos comandos deste HOWTO devem ser executados como superusuário (root).

A construção de um disco de inicialização do nada pode ser bem complexa. Caso não se tenha lido os conteúdos do FAQ do Linux e documentos relacionados, como por exemplo, o HOWTO Instalação Linux e o Guia de Instalação Linux, não é aconselhável tentar-se a construção dos discos desta forma. Caso se necessite somente de um disco de emergência que funcione, será *muito* mais simples utilizar um preexistente. Veja o Apêndice C.11.1 (Discos de Inicialização Pré-Configurados), abaixo, a fim de verificar onde encontrá-los.

C.3 Discos de Inicialização e o Processo de Inicialização do Sistema

Um disco de inicialização é basicamente uma miniatura de um sistema completo, ou seja é um sistema Linux contido em um disquete. Ele deve executar muitas das funções que o sistema completo permite. Antes de tentar construir um, deve-se conhecer os conceitos básicos do processo de inicialização do sistema, apresentados a seguir, os quais são suficientes para o entendimento do restante deste documento. Alguns detalhes e opções foram omitidos por não serem significativos para o conteúdo deste documento.

C.3.1 O Processo de Inicialização

Todos os sistemas em microcomputadores começam o processo de inicialização executando código existente na ROM (especificamente no BIOS), para carregar o setor 0 do cilindro 0 do dispositivo de inicialização. O dispositivo de inicialização é normalmente a primeira unidade de disquete (denominada `A:` no DOS e `/dev/fd0` no Linux). O BIOS tenta então executar este setor. Em muitos discos inicializáveis, o setor 0, cilindro 0, pode conter ainda:

- código de um carregador de sistemas, como o LILO, o qual localiza o kernel do sistema escolhido, carrega e executa, de acordo com a opção definida.
- o início de um kernel de um sistema operacional, como por exemplo o Linux.

Caso o kernel do Linux tenha sido copiado fisicamente para um disquete, o primeiro setor do disco será o primeiro setor do kernel do Linux. O primeiro setor continuará o processo de inicialização, carregando o restante do kernel contido no dispositivo.

Uma vez que o kernel tenha sido completamente carregado, ele executa alguma inicialização básica de dispositivos. Ele tenta carregar e montar o **sistema de arquivos raiz** a partir de algum dispositivo. Um sistema de arquivos raiz é simplesmente um sistema de arquivos montado como "/". Deve-se indicar para o kernel a localização do sistema de arquivos raiz; caso não seja encontrada uma imagem inicializável naquele local, o sistema é paralisado.

Em algumas situações de início do sistema, freqüentemente na inicialização a partir de disquetes, o sistema de arquivos é montado em um **disco em memória**, o qual é acessado na memória RAM, como se fosse um disco físico. Há duas razões para o sistema ser carregado em discos em memória. Inicialmente, memória RAM é muitas vezes mais rápida que um disquete, tornando a operação do sistema mais rápida, e segundo, o kernel do sistema pode ser armazenado como um **sistema de arquivos compactado** em um disquete, e descompactado no disco em memória, permitindo que muitos mais arquivos sejam armazenados no disquete.

Uma vez que o sistema de arquivos raiz é montado, pode-se visualizar uma mensagem similar a:

```
VFS:  Raiz montado (sistema de arquivos ext2) somente para leitura.
```

Neste momento o sistema encontra o programa `init` no sistema de arquivos raiz (em `/bin` ou `/sbin`) e executa-o. `init` lê o arquivo de configuração `/etc/inittab`, procurando por uma linha denominada `sysinit`, e executa o programa especificado. O programa `sysinit` é normalmente denominado `/etc/rc` ou `/etc/init.d/boot`. Este programa é constituído de uma série de comandos de shell que configuram os serviços básicos do sistema, como por exemplo:

- Execução do comando `fsck` em todos os discos
- Carga dos módulos necessários ao kernel
- Inicialização da área de troca
- Inicialização da rede
- Montagem dos discos descritos em `fstab`.

Este programa normalmente aciona diversos outros, tornando o processo de inicialização modular. Por exemplo, na estrutura comum do SysVinit, o diretório `/etc/rc.d/` contém uma estrutura complexa de subdiretórios, cujos arquivos definem como iniciar e desligar a maior parte dos serviços do sistema. De qualquer forma, um programa sysinit em um disco de inicialização é normalmente muito simples.

Quando o programa sysinit termina, o controle retorna ao `init`, o qual entrará no *nível de execução padrão*, especificado em `inittab`, através da palavra chave `initdefault`. A linha de nível de execução normalmente especifica um programa como `getty`, o qual é responsável pelo gerenciamento das comunicações através da console e ttys. É o programa `getty` que apresenta a expressão familiar "`login:`". O programa `getty` por sua vez, chama o programa `login` para administrar o processo de validação e iniciar as sessões dos usuários.

C.3.2 Tipos de Discos

Após a revisão do processo básico de inicialização, podemos agora definir diversos tipos de discos envolvidos. Podemos classificar os discos em quatro tipos. A discussão aqui contida e através de todo o documento do uso do termo disco refere-se a disquetes, a menos que seja especificado o contrário, observando-se que na verdade, o conceito pode ser aplicado sem distinção a discos rígidos.

inicialização

Um disco contendo um kernel do sistema que pode ser inicializado. O disco pode ser usado para iniciar o kernel do sistema, o qual pode carregar o sistema de arquivos raiz a partir de outro disco. O kernel em um disco de inicialização pode receber informações sobre a localização do sistema de arquivos raiz. Freqüentemente um disco de inicialização carrega o sistema de arquivos raiz a partir de outro disquete, porém é possível configurar a carga a partir de um sistema de arquivos raiz residente em um disco rígido, por exemplo. Isso é comumente feito quando se está testando um novo kernel (na verdade "`make zdisk`" criará um disco de inicialização automaticamente a partir dos fontes do kernel).

raiz

Um disco com um sistema de arquivos raiz contém os arquivos necessários para a execução de um sistema Linux. Tal disco pode não conter necessariamente nem o kernel e tão pouco o carregador de sistemas .

Um disco raiz pode ser usado para executar o sistema independentemente de outros discos, uma vez que o kernel do sistema tenha sido inicializado. Normalmente o disco raiz é automaticamente copiado para um disco em memória, o que torna o acesso às suas informações muito mais rápido e libera a unidade de disquetes para outras atividades.

inicialização/raiz

Um disco pode conter tanto o kernel quanto um sistema de arquivos raiz. Em outras palavras, ele contém todo o necessário para inicializar e executar um sistema Linux, sem a necessidade de um disco rígido. A vantagem desse tipo de disco é que a solução torna-se compacta. Todo o necessário está em um único disco. Por outro lado, o gradual aumento de tamanho dos itens necessários ao processo significa o aumento da dificuldade de colocar-se tudo em um único disquete, mesmo com compactação.

utilitário

É um disco que contém um sistema de arquivos, mas que não será montado como um sistema raiz. É um disco de dados adicionais, e pode ser utilizado para a disponibilização de utilitários, caso o disco raiz não tenha mais espaço disponível.

Em geral, quando falamos de "construir um disco de inicialização" significa a criação das funções de carga do kernel e do sistema de arquivos raiz. Elas podem estar juntas (em um único disco de inicialização e raiz) ou separados (disco de inicialização e disco raiz). A abordagem mais flexível para discos de emergência é provavelmente usar disquetes separados, e um ou mais disquetes de utilitários para gerenciar o que não foi possível colocar nos primeiros.

C.4 Construíndo um Sistema de Arquivos Raiz

Criar um sistema de arquivos raiz envolve a seleção dos arquivos necessários para que o sistema possa ser executado. Nesta seção descreveremos como construir um *sistema de arquivos raiz compactado*. Uma opção menos usual é a construção de um sistema de arquivos não compactados em um disquete, que é montado diretamente como raiz; esta alternativa é descrita na seção C.8.2.

C.4.1 Visão Geral

Um sistema de arquivos raiz deve conter todo o necessário para suportar um sistema Linux completo. Para tanto, o disco deve incluir os requisitos mínimos de um sistema Linux:

- A estrutura básica do sistema de arquivos
- Conjunto mínimo de diretórios: `/dev, /proc, /bin, /etc, /lib, /usr, /tmp`
- Conjunto básico de utilitários: `sh, ls, cp, mv`, etc.
- Conjunto mínimo de arquivos de configuração: `rc, inittab, fstab`, etc...
- Dispositivos: `/dev/hd*, /dev/tty*, /dev/fd0`, etc...
- Biblioteca que disponibilize as funções básicas necessárias aos utilitários

Evidentemente, qualquer sistema somente torna-se útil quando permite a execução de algum programa, e um disquete raiz somente é útil quando permite que sejam executadas funções como:

- Verificar um sistema de arquivos em outro dispositivo, por exemplo para checar o sistema de arquivos raiz em um disco rígido será necessário carregar o sistema operacional a partir de outro dispositivo, o que pode ser feito com um sistema de arquivos raiz em disquete. Pode-se então executar `fsck` no dispositivo original que contém o sistema raiz enquanto ele não estiver montado.
- Restauração total ou parcial do dispositivo raiz original a partir de uma cópia de segurança usando arquivamento ou utilitários de compactação, tais como `cpio, tar, gzip` e `ftape`.

Descreveremos como construir um sistema de arquivos *compactado*, assim chamado porque é arquivado compactado no disco e é descompactado do disco para memória. Um sistema de arquivos compactado pode conter diversos arquivos (aproximadamente 2 megabytes) em um disquete padrão 1.440 Kb. Como o arquivo é muito maior que o disquete, não se pode construí-lo diretamente no dispositivo. Deve-se construí-lo em outro local qualquer, compactá-lo, e então copiá-lo para o disquete.

C.4.2 Criando o Sistema de Arquivos

Para construir-se um sistema de arquivos raiz, deve-se ter um dispositivo extra, grande o suficiente para conter todos os arquivos antes da compactação. Deve-se ter à mão um dispositivo capaz de conter pelo menos 4 Mb. Há diversas opções:

- Usar um **disco em memória** (DEVICE = `/dev/ram0`). Neste caso a memória é utilizada para simular um dispositivo de disco, sendo que deve ser grande o suficiente para conter todo o sistema de arquivos em seu tamanho adequado. Caso se use o LILO, deve-se verificar o arquivo de configuração (`/etc/lilo.conf`) e buscar uma linha similar a :

  ```
  RAMDISK_SIZE = nnn
  ```

  a qual determina quanto de memória RAM será alocada. O padrão é 4096K, que deverá ser suficiente. Não se deve tentar utilizar esse disco em memória para máquinas com menos de 8 Mb de RAM.

  Verifique a existência de um dispositivo como por exemplo `/dev/ram0,/dev/ram` ou `/dev/ramdisk`. Caso não exista, crie `/dev/ram0` com `mknod` (número principal 1, secundário 0).

- Caso se tenha uma partição de disco rígido grande o suficiente (diversos megabytes) disponível, esta certamente será uma boa solução. Caso se tenha memória RAM disponível, pode-se desligar as funções de troca e utilizar a partição de troca (swap).

- Usar uma **simulação de dispositivo**, a qual permite que um arquivo em disco seja tratado como um dispositivo. Usando-se uma simulação permite a criação de um arquivo de três megabytes no disco rígido e a construção do sistema de arquivos nele.

  Para fazer uso da simulação de dispositivos, deve-se utilizar os programas `mount` e `unmount` especialmente alterados para isso. Eles podem ser encontrados no diretório: `ftp://ftp.win.tue.nl/pub/linux/util/mount/`

  Caso não se tenha uma simulação de dispositivos (`/dev/loop0`, `/dev/loop1`, etc...) no sistema, pode-se criar uma através do comando “`mknod /dev/loop0 b 7 0`”. Uma vez instalados os binários especiais de `mount` e `umount`, deve-se criar um arquivo temporário em um disco rígido com capacidade suficiente (por exemplo, `/tmp/fsfile`). Pode-se, por exemplo, utilizar o comando:

  ```
  dd if=/dev/zero of=/tmp/fsfile bs=1k count=nnn
  ```

  para criar um arquivo com *nnn* blocos.

  Deve-se utilizar então o nome do arquivo no lugar do DISPOSITIVO a seguir. Ao utilizar o comando mount deve-se incluir a opção “`-o loop`” para definir uma simulação de dispositivos. Por exemplo:

  ```
  mount -o loop -t ext2 /tmp/fsfile /mnt
  ```

  irá montar `/tmp/fsfile` (através de uma simulação de dispositivo) no ponto de montagem `/mnt`. Através do comando `df` pode-se obter a confirmação disso.

Após a escolha de alguma dessas opções, deve-se preparar o DISPOSITIVO com o seguinte comando:

```
dd if=/dev/zero of=DISPOSITIVO bs=1k count=3000
```

Este comando inicializa com zeros o DISPOSITIVO. Este passo é importante pois o sistema de arquivos será compactado posteriormente e as partes não utilizadas e preenchidas com zeros, atingirão o máximo de compactação.

Após, pode-se criar o sistema de arquivos. O kernel do Linux reconhece dois tipos de sistemas de arquivos para discos raiz a serem automaticamente copiados para discos em memória. Há o minix e o ext2, sendo o segundo o mais indicado. No caso de utilização do ext2, pode ser útil o uso da opção `-i` para especificar um número maior de inodes que o padrão do sistema; `-i 2000` é o valor sugerido, garantindo-se assim que não faltarão inodes. Alternativamente, pode-se salvar inodes removendo-se os diversos arquivos `dev` desnecessários. `mke2fs` irá criar 360 inodes por padrão em um disquete 1.44. Cremos que 120 inodes são suficientes para um disco de emergência padrão, mas caso todos os dispositivos `/dev` sejam incluídos, então pode-se facilmente exceder os 360 disponíveis. Usar um sistema de arquivos raiz compactado permite um sistema de arquivos maior, provocando a utilização de um número de inodes maior que o padrão, porém pode ainda ser necessário reduzir o número de arquivos ou incrementar o número de inodes.

O comando necessário será algo similar a:

```
mke2fs -m 0 -i 2000 DISPOSITIVO
```

(Caso se esteja usando uma simulação de dispositivos, o arquivo em disco que se esteja utilizando deve ser informado no lugar do DISPOSITIVO. Neste caso `mke2fs` perguntará se realmente se deseja executar o comando e a resposta deve ser positiva.)

O comando `mke2fs` automaticamente detectará o espaço disponível e fará a configuração automaticamente. O parâmetro `-m 0` evita a alocação de espaço para o raiz, e adicionalmente provê mais espaço útil em disco.

A seguir deve-se montar o dispositivo:

```
mount -t ext2 DISPOSITIVO /mnt
```

(Deve-se criar um ponto de montagem `mnt` caso ele ainda não exista). Nas próximas seções, todos os nomes de diretórios são assumidos como relativos a `/mnt`.

C.4.3 Ocupando o Sistema de Arquivos

Segue um razoável número mínimo de diretórios no sistema de arquivos raiz:

- `/dev` Dispositivos, necessários para as operações de Leitura/Gravação
- `/proc` Diretório temporário requerido pelo sistema de arquivos proc
- `/etc` Arquivos de configuração do sistema
- `/sbin` Binários fundamentais do sistema
- `/bin` Binários básicos e considerados parte do sistema
- `/lib` Bibliotecas compartilhadas que provêem suporte à execução dos binários
- `/mnt` Um ponto de montagem para manutenção em outros discos
- `/usr` Utilitários adicionais e aplicações

(A estrutura de diretórios aqui apresentada é somente para uso no disquete raiz. Sistemas Linux têm uma política mais complexa e disciplinada, chamada Padrões de Sistemas de Arquivos, para determinar quais arquivos devem estar presentes e aonde.)

Três destes diretórios devem estar vazios no sistema de arquivos raiz, devendo somente serem criados com o comando `mkdir`. O diretório `/proc` é basicamente um ponto de referência onde o sistema de arquivos proc está localizado. Os diretórios `/mnt` e `/usr` são somente pontos de montagem para uso após a inicialização do sistema e a carga do sistema de arquivos raiz. Mais uma vez, estes diretórios somente precisam ser criados.

Os quatro diretórios remanescentes estão descritos nas seções a seguir:

`/dev`

Um diretório `/dev` contendo um arquivo especial com todos os dispositivos a serem utilizados pelo sistema é fundamental para o Linux. O diretório em si é um diretório comum e pode ser criado com o comando `mkdir` da forma usual. Os arquivos especiais de dispositivos devem ser criados de uma forma especial, utilizando-se o comando `mknod`.

Há um atalho, podendo-se copiar o conteúdo do diretório `/dev` e apagando-se o que for desnecessário. A única exigência é que a cópia seja efetuada com a utilização do parâmetro `-R`. Isso copiará o diretório sem copiar o conteúdo dos arquivos. *Esteja seguro de utilizar um R maiúsculo*. Caso seja utilizado `r` em formato minúsculo, provavelmente será copiado o conteúdo completo de todo o disco rígido – ou no mínimo, o que couber no disquete! De qualquer forma, é importante estar atento ao comando:

```
cp -dpR /dev /mnt
```

Assumindo-se que o disquete esteja montado em `/mnt`. A opção `dp` garante que ligações simbólicas serão copiadas como ligações, ao invés de usar um arquivo de destino, e que os atributos originais do arquivo serão preservados, assim como as informações sobre os donos.

Alternativamente, pode-se usar o programa `cpio` com a opção `-p`, uma vez que `cpio` lida com arquivos especiais corretamente, e não tentará copiar o seu conteúdo. Pode-se por exemplo, utilizar o seguinte comando:

```
cd /dev
find . -print | cpio -pmd /mnt/dev
```

o qual irá copiar todos os arquivos especiais de `/dev` para `/mnt/dev`. Na verdade, irá copiar todos os arquivos da árvore de diretórios iniciada em `/dev`, e criará todos os subdiretórios necessários na árvore de diretórios de destino.

Caso se deseje fazer da forma mais difícil, deve-se usar `ls -l` para mostrar os números major e minor dos dispositivos desejados, e criá-los no disquete através do comando `mknod`.

Uma vez que os dispositivos estejam copiados, é aconselhável verificar se todos os arquivos de dispositivos necessários foram copiados no disco de emergência. Por exemplo, `ftape` é utilizado por unidades de fitas, sendo necessário copiar todos eles, caso se pretenda acessar um dispositivo desse tipo a partir do disco de inicialização.

Note que um inode é necessário para cada tipo de arquivo especial de dispositivo, e inodes podem, às vezes, serem um recurso escasso, especialmente em disquetes com sistemas de arquivos configurados. Desta forma é indicada a remoção de qualquer arquivo especial de dispositivos em `/dev` que não seja necessário no sistema específico. Por exemplo, caso não se tenha discos SCSI, pode-se tranqüilamente remover todos os arquivos de dispositivos começados por `sd`. Similarmente, caso não se pretenda utilizar portas seriais, então todos os arquivos começados com `cua` também podem ser removidos.

Deve-se necessariamente ter os seguintes arquivos neste diretório: `console, kmem, mem, null, ram, tty1`.

`/etc`

Este diretório deve conter uma série de arquivos de configuração. Na maioria dos sistemas este pode estar dividido em três grupos:

1. Sempre requeridos, *por exemplo* `rc, fstab, passwd`
2. Podem ser requeridos, mas não há como assegurar
3. Arquivos desnecessários

Arquivos que não são essenciais podem ser identificados através do comando:

```
ls -ltru
```

Este comando gera uma lista em ordem inversa de último acesso, dependendo de que arquivos não são acessados ou utilizados e que, podem não estar presentes no disquete raiz.

Nos nossos disquetes raiz, temos um número de arquivos de configuração inferior a 15. Isso reduz o trabalho de lidar com um conjunto de três tipos de arquivos.

1. Os arquivos que devem ser configurados para um sistema:
   (a) `rc.d/*` início do sistema e definição dos programas de cada nível de execução
   (b) `fstab` lista dos sistemas de arquivos que devem ser montados
   (c) `inittab` parâmetros para o processo `init`, o primeiro que é executado em tempo de inicialização do sistema.
2. Estes devem ser customizados para a inicialização de um sistema:
   (a) `passwd` lista de usuários, diretórios pessoais, etc...
   (b) `grupo` grupos de usuários
   (c) `shadow` senha dos usuários. Eventualmente pode não existir.

   Caso segurança seja um item importante do sistema específico, `passwd` e `shadow` devem ser suprimidos, a fim de evitar a cópia de senhas de usuários para fora do sistema, e quando o sistema for inicializado através de disquetes, acessos indesejados serão rejeitados. De qualquer forma, há uma razão para *não* suprimir `passwd` e `group`. `tar` (e provavelmente outros programas de arquivamento) armazenam o usuário e o grupo junto com os dados dos arquivos. Caso estes arquivos sejam restaurados no disco rígido a partir de uma fita, os arquivos serão restaurados com seus nomes originais. Caso os nomes dos donos e grupos não existam em `passwd`/`group` durante a restauração, as identificações de usuários e grupos (UID e GID) não estarão corretas.

   Esteja certo de que o arquivo `passwd` contém ao menos o superusuário `root`. Caso se pretenda utilizar outros usuários para acessar o sistema, deve-se estar seguro da existência de seus diretórios pessoais e interpretadores de comando (shell).
3. Os demais: verificaremos mais adiante as informações sobre este tópico.

Além disso, deve-se somente configurar dois arquivos, e o que eles devem conter é surpreendentemente pequeno.

- `rc` deve conter

  ```
  #!/bin/sh
  /bin/mount -av
  /bin/hostname Conectiva
  ```

  Deve-se estar seguro de que os diretórios estão corretos. A execução de `hostname` não é obrigatória, somente dá um melhor acabamento ao trabalho.
- `fstab` deve conter, no mínimo:

  ```
  /dev/ram0     /         ext2    defaults
  /dev/fd0      /         ext2    defaults
  /proc         /proc     proc    defaults
  ```

  Pode-se copiar as entradas de um arquivo `fstab`, já existente, mas não se deve montar automaticamente qualquer partição do disco rígido; usando-se então o parâmetro `noauto`, pois o disco rígido pode estar danificado ou sem condições de uso no momento da inicialização do sistema.

O `inittab` deve ser alterado, de outra forma a linha `sysinit` executará o `rc` ou qualquer outro programa básico de inicialização que seja indicado. Ainda para assegurar-se de que usuários em portas seriais não poderão acessar o sistema, pode-se comentar todas as entradas em `getty` que incluam dispositivos `ttys` ou `ttyS` ao final da linha. Deve-se deixar as portas `tty` para poder-se acessar o sistema a partir da console.

Um arquivo `inittab` mínimo contém:

```
id:2:initdefault:
si::sysinit:/etc/rc
1:2345:respawn:/sbin/getty 9600 tty1
2:23:respawn:/sbin/getty 9600 tty2
```

O arquivo `inittab` define o que o sistema executará nos vários estados, inclusive no seu início, em modo multiusuário, etc... Um ponto no qual deve-se ter muito cuidado é o de checar se todos os comandos informados em `inittab` referem-se a programas presentes e se o diretório está corretamente indicado. Caso se coloque no disco de emergência os arquivos de comandos apresentados na Seção C.13 (Listas de exemplo do conteúdo do disco de inicialização) como um guia, e após se copie o `inittab` para o disco sem uma checagem cuidadosa, provavelmente ele falhará, e o problema terá origem na ausência de arquivos ou indicações erradas de diretórios.

Note que alguns comandos não podem ser movidos para qualquer outro lugar, porque alguns programas têm a sua localização dentro de seu código. Por exemplo em nosso sistema, `/etc/shutdown` tem a sua localização definida no fonte do comando `/etc/reboot`. Caso `reboot` seja movido para `/bin/reboot`, e após seja executado o comando `shutdown`, ele falhará, porque o arquivo `reboot` não pode ser localizado.

Para todo o restante, deve-se simplesmente copiar os arquivos texto no diretório `/etc`, mais os executáveis do mesmo diretório que não possam ser definidos como desnecessários. Como um guia, pode-se consultar os exemplos na Seção C.13 (Listas de exemplo do conteúdo do disco de inicialização). Provavelmente será suficiente copiar somente aqueles arquivos, porém sistemas podem ser muito diferentes, então não se pode estar seguro de que a lista apresentada seja suficiente. O único método de estar seguro é iniciar o sistema com `inittab` e verificar o que é solicitado.

Muitos sistemas utilizam um diretório `/etc/rc.d/` contendo shell scripts de diferentes níveis de execução. O mínimo é um simples programa `rc`, mas pode ser mais simples copiar o `inittab` e o diretório `/etc/rc.d` de um sistema já existente, e suprimir os scripts no diretório `rc.d` para remover os processamentos não relevantes do ambiente de sistema em disquete.

`/bin e /sbin`

O diretório `/bin` é um lugar adequado para utilitários extras necessários à execução de atividades básicas. Utilitários como `ls`, `mv`, `cat` e `dd`.Veja a Seção C.13 (Listagem de exemplo de conteúdo do disco de inicialização) para um exemplo da lista de arquivos que podem estar presentes nos diretórios `/bin` e `/sbin`. Ela não inclui nenhum utilitário requerido para restaurar cópias de segurança, tais como `cpio`, `tar` e `gzip`. Isso porque estes programas foram colocados em um disquete de utilitários em separado, visando economizar espaço no disquete de inicialização e raiz. Uma vez que o disquete de inicialização tenha sido carregado, ele é copiado para o disco em memória, deixando a unidade de disquetes livre para montar outro disquete, o disquete de utilitários. Normalmente montamos esse disquete como `/usr`.

A criação de um disquete de utilitários é descrito na Seção C.8.3 (Construindo um disquete de utilitários). É desejável manter uma cópia da mesma versão dos utilitários de cópias de segurança usados para gerar as cópias de segurança disponíveis, não perdendo-se tempo assim tentando-se instalar versões que não podem ler as cópias geradas.

Esteja seguro de incluir os seguintes programas: `init`, `getty` ou equivalente, `login`, `mount`, algum interpretador que possa executar os programas rc, e uma ligação de `sh` para o shell.

/lib

No diretório `/lib` deve-se colocar as bibliotecas compartilhadas e seus carregadores. Caso as bibliotecas necessárias não sejam encontradas no diretório `/lib`, o sistema não poderá ser iniciado. Com um pouco de sorte pode-se receber uma mensagem de erro dizendo a razão.

Praticamente todos os programas requerem no mínimo a biblioteca `libc`, `libc.so.N`, onde N é o número da versão corrente. Ao verificar o diretório `/lib`, `libc.so.5` é normalmente uma ligação simbólica para um arquivo com o número completo da versão.

```
% ls -l /lib/libc.so*
lrwxrwxrwx  1 root root      14 Nov  1 20:34 /lib/libc.so.5 -> libc.so.5.4.33*
-rwxr-xr-x  1 root root  573176 Jun 12 02:05 /lib/libc.so.5.4.33*
```

Neste caso, tem-se disponível o arquivo `libc.so.5.4.33`. Para encontrar outras bibliotecas deve-se verificar todos os binários necessários e checar as suas dependências com o comando `ldd` . Por exemplo:

```
% ldd /sbin/mke2fs
        libext2fs.so.2 => /lib/libext2fs.so.2
        libcom_err.so.2 => /lib/libcom_err.so.2
        libuuid.so.1 => /lib/libuuid.so.1
        libc.so.5 => /lib/libc.so.5
```

O arquivo apresentado na coluna da direita é necessário, tendo-se em mente que as bibliotecas listadas podem ser ligações simbólicas.

Em `/lib` deve-se ainda incluir um carregador de bibliotecas. O carregador será ou o `ld.so` (para bibliotecas a.out) ou `ld-linux.so` (para bibliotecas ELF). Caso não se esteja seguro do que será necessário, deve-se executar o comando `file` na biblioteca. Por exemplo:

```
% file /lib/libc.so.5.4.33 /lib/libc.so.4.7.2
  /lib/libc.so.4.7.2: Linux/i386 demand-paged executable (QMAGIC), stripped
  /lib/libc.so.5.4.33: ELF 32-bit LSB shared object, Intel 386, version 1, stripped
```

`QMAGIC` indica que `4.7.2` é para bibliotecas a.out, e `ELF` indica que `5.4.33` é para ELF.

Deve-se então copiar o(s) carregador(es) necessário(s) para o sistema de arquivos raiz em construção. Bibliotecas e carregadores devem ser checados *cuidadosamente* com os binários incluídos. Caso o kernel não possa carregar a biblioteca necessária, normalmente haverá um travamento sem mensagens de erro.

C.4.4 Módulos

Caso se tenha um kernel modular, deve-se considerar quais módulos devem ser carregados a partir do disco de inicialização após o início do sistema. Pode-se incluir os módulos `ftape` e `zftape` caso cópias de segurança tenham sido feitas em uma fita, módulos para dispositivos SCSI caso eles estejam presentes, e módulos para suporte a PPP ou SLIP caso se queira acesso à rede em uma emergência.

Estes módulos podem ser colocados em `/lib/modules`. Deve-se ainda incluir `insmod`, `rmmod` e `lsmod`.

Caso se deseje carregar os módulos automaticamente, pode-se incluir ainda `modprobe`, `depmod` e `swapout`. E caso se use o `kerneld`, deve-se incluir ainda o `/etc/conf.modules`.

A principal vantagem de utilizar módulos reside no fato de poder mover módulos não essenciais para um disco de utilitários e carregá-los quando necessário, usando menos espaço no disco raiz. Porém, caso seja necessário lidar com muitos dispositivos diferentes, uma abordagem mais adequada pode residir em construir um único kernel com diversos módulos integrados.

Note que para se ter um sistema de arquivos ext2 compactado, é obrigatória a existência de suporte a disco em memória e sistemas de arquivos ext2. Estes não podem ser disponibilizados como módulos.

C.4.5 Alguns detalhes finais

Alguns programas de sistema, como o `login`, apresentam mensagem de advertência caso o arquivo `/var/run/utmp` e o diretório `/var/log` não existam. Então:

```
mkdir -p /mnt/var/{log,run}
touch /mnt/var/run/utmp
```

Finalmente, após ter-se configurado todas as bibliotecas necessárias, deve-se executar o `ldconfig` para gerar novamente o `/etc/ld.so.cache` no sistema de arquivos raiz. O cache diz ao carregador onde encontrar as bibliotecas. Para reconstruir o `ld.so.cache`, execute os seguintes comandos:

```
chdir /mnt; chroot /mnt /sbin/ldconfig
```

O comando `chroot` é necessário porque `ldconfig` sempre gera o cache para o sistema de arquivos raiz.

C.4.6 Empacotando

Uma vez concluída a construção do sistema de arquivos raiz, ele deve ser desmontado, copiado para um arquivo e compactado:

```
umount /mnt
dd if=DISPOSITIVO bs=1k | gzip -v9 > saraiz.gz
```

Isso pode levar diversos minutos. Ao finalizar estará disponível um arquivo `saraiz.gz` que é o sistema de arquivos raiz compactado. Deve-se verificar se o arquivo cabe em um disquete. Caso não caiba, deve-se retornar aos passos anteriores e eliminar alguns arquivos. A Seção C.8.1 (Reduzindo o tamanho de um sistema de arquivos raiz) fornece algumas dicas sobre a redução de tamanho do sistema de arquivos raiz.

C.5 Escolhendo um kernel

Neste ponto tem-se disponível um sistema de arquivos raiz compactado. O próximo passo é construir um kernel ou selecionar um kernel. Em muitos casos é possível copiar o kernel atual e inicializar o sistema a partir do disquete. Porém em muitos casos poderá ser necessário construir um em separado.

Uma das razões é o tamanho. Caso se esteja construindo um único disquete de inicialização e raiz, o kernel será um dos maiores arquivos no disquete e será necessário reduzí-lo o máximo possível. Caso se esteja construindo dois discos (um de inicialização e um raiz), isso não será problema pois o kernel irá em um disquete separado.

Para reduzir seu tamanho, deve-se construir um kernel com o mínimo de facilidades necessárias ao suporte do sistema desejado. Isso significa, deixar de lado tudo que não seja absolutamente necessário. Suporte a rede é um dos prováveis candidatos, assim como suporte a unidades de disco e outros dispositivos desnecessários durante o início do sistema. Conforme descrito anteriormente, o kernel *deve* ter suporte a disco em memória e ext2 .

Após incluir somente o mínimo necessário de facilidades no kernel, deve-se verificar o que deve retornar. Provavelmente um dos usos mais comuns a um disquete de inicialização e raiz é a restauração de sistemas de arquivos raiz com problemas, e para que isso seja possível é necessário suporte no kernel do sistema. Por exemplo, caso as cópias de segurança tenham sido efetuadas em fita, utilizando Ftape para acessar uma unidade de fita, então será obrigatória a presença de suporte a dispositivos de fitas para efetuar uma restauração. Caso não esteja presente, poderá ser necessário reinstalar o Linux, copiar e reinstalar ftape, e então tentar ler as cópias de segurança.

O ponto aqui é, qualquer suporte a leitura e gravação que seja adicionado ao kernel para suportar cópias de segurança, deve também ser adicionado ao kernel de inicialização do sistema em disquete de emergência.

O procedimento para construir um kernel é descrito na documentação que o acompanha. É relativamente simples, podendo-se verificar o conteúdo de `/usr/src/linux`. Note que caso se tenha problemas construindo um novo kernel, então provavelmente não se deve tentar construir um disco de inicialização. Deve-se lembrar de compactar o kernel através do comando “`make zImage`”.

C.6 Colocando tudo junto: Construindo o(s) disco(s)

Neste ponto, tem-se um kernel e um sistema de arquivos raiz. Caso se esteja construindo um único disco de inicialização e raiz, deve-se verificar o seu tamanho e estar seguro de que eles caberão em um único disco. Caso se esteja construindo dois discos, deve-se verificar se o sistema de arquivos cabe em um único disquete.

Deve-se decidir pelo uso ou não do LILO no disquete de inicialização do kernel. A alternativa será copiar o kernel diretamente no disquete e iniciar o sistema sem o LILO. A vantagem de utilizar o LILO reside na possibilidade de passagens de parâmetros para o kernel, o que pode ser necessário para inicializar algum hardware (deve-se verificar o conteúdo do arquivo `/etc/lilo.conf`. Caso exista alguma linha do tipo “`append=...`”, provavelmente esta facilidade será necessária). A desvantagem de usar o LILO é que a construção do disco de inicialização torna-se mais complexa e mais espaço é necessário. Deve-se construir um pequeno sistema de arquivos em separado, o qual nós denominamos **sistema de arquivos do kernel**, para onde se pode transferir o kernel e algumas outras coisas necessárias para o LILO.

Caso se vá utilizar o LILO, siga adiante, caso contrário pode-se ir diretamente para a seção C.6.2 (Transferindo o kernel sem o LILO).

C.6.1 Transferindo o kernel com o LILO

O primeiro item que deve ser criado é um pequeno arquivo de configuração do LILO. Ele deve ser similar ao seguinte:

```
boot      =/dev/fd0
install   =/boot/boot.b
map       =/boot/map
read-write
backup    =/dev/null
compact
image     = KERNEL
```

```
label      = Disquete_Inic
root       =/dev/fd0
```

Para opter detalhes destes parâmetros, por favor verifique a documentação de usuário do LILO. Pode ser desejável ainda adicionar a linha `append=...` copiada do arquivo `/etc/lilo.conf` existente em disco.

Deve-se salvar o arquivo como `bdlilo.conf` e a seguir criar um pequeno sistema de arquivos, o qual será denominado **sistema de arquivos do kernel**, para distinguí-lo do sistema de arquivos raiz.

Inicialmente, deve-se verificar o tamanho que o sistema de arquivos terá. Verifique o tamanho do kernel em blocos (o tamanho é mostrado através do comando "`ls -l KERNEL`" dividido por 1024, arredondado para cima e acrescido de 50), referentes ao espaço necessário aos inodes e outros arquivos. Pode-se calcular o número exato ou simplesmente utilizar 50. Caso se esteja criando um conjunto com dois disquetes, pode-se superestimar o espaço usado pelo kernel. Denominaremos este número de `NÚCLEO_BLOCOS`.

Deve-se colocar um disquete no dispositivo (para simplificar assumiremos que será em `/dev/fd0`) e após criar um sistema de arquivos tipo ext2 para o kernel:

```
mke2fs -i 8192 -m 0 /dev/fd0 NÚCLEO_BLOCOS
```

O parâmetro "`-i 8192`" especifica que desejamos um inode para cada 8192 bytes. Após, deve-se montar o sistema de arquivos, remover o diretório `lost+found` e criar os diretórios `dev` e `boot` para o LILO:

```
mount /dev/fd0 /mnt
rm -rf /mnt/lost+found
mkdir /mnt/{boot,dev}
```

Após, deve-se criar os dispositivos `/dev/null` e `/dev/fd0`. Ao invés de procurar pelos números dos dispositivos, pode-se simplesmente copiá-lo do disco rígido utilizando-se `-R`:

```
cp -R /dev/{null,fd0} /mnt/dev
```

LILO necessita de uma cópia de seu carregador de inicialização, `boot.b`, o qual pode ser encontrado no disco rígido, normalmente no diretório `/boot` .

```
cp /boot/boot.b /mnt/boot
```

Finalmente, deve-se copiar os arquivos de configuração do LILO criado conforme a seção anterior, assim como o kernel. Ambos devem estar presentes no diretório raiz:

```
cp bdlilo.conf KERNEL /mnt
```

Todo o necessário para que o LILO possa ser executado está presente no sistema de arquivos do kernel. Deve-se então executar o LILO com o parâmetro `-r` para instalar o carregador de inicialização :

```
lilo -v -C bdlilo.conf -r /mnt
```

LILO deverá ser executado sem erros, após o qual o sistema de arquivos do kernel deve ter a seguinte aparência:

```
total 361
  1 -rw-r--r--   1 root     root          176 Jan 10 07:22 bdlilo.conf
  1 drwxr-xr-x   2 root     root         1024 Jan 10 07:23 boot/
  1 drwxr-xr-x   2 root     root         1024 Jan 10 07:22 dev/
358 -rw-r--r--   1 root     root       362707 Jan 10 07:23 vmlinuz
boot:
total 8
  4 -rw-r--r--   1 root     root         3708 Jan 10 07:22 boot.b
  4 -rw-------   1 root     root         3584 Jan 10 07:23 map
dev:
total 0
  0 brw-r-----   1 root     root       2,   0 Jan 10 07:22 fd0
  0 crw-r--r--   1 root     root       1,   3 Jan 10 07:22 null
```

Não há motivos para preocupação caso os tamanhos de arquivos sejam um pouco diferentes.

Pode-se ir agora para a seção C.6.3 (Configurando o disco em memória).

C.6.2 Transferindo o kernel sem o LILO

Caso *não* se esteja utilizando o LILO, o kernel pode ser transferido para o disco de inicialização com o comando `dd` :

```
% dd if=KERNEL of=/dev/fd0 bs=1k
353+1 records in
353+1 records out
```

Neste exemplo, `dd` gravou 353 registros completos e 1 parcialmente, concluindo-se que o kernel ocupou 354 blocos do disquete. Denominaremos este número como `NÚCLEO_BLOCOS` , o qual será utilizado na próxima seção.

Finalmente, deve-se configurar o disquete como o dispositivo raiz e dar-lhe permissões de leitura e gravação:

```
rdev /dev/fd0 /dev/fd0
rdev -R /dev/fd0 0
```

Deve-se ter o máximo cuidado ao usar o `-R` maiúsculo no comando `rdev` .

C.6.3 Configurando o disco em memória

Dentro da imagem do kernel está a **palavra (dois ou mais bytes) de configuração do disco em memória** que especifica onde o sistema de arquivos raiz deve ser encontrado, em conjunto com as suas opções. A palavra é definida em `/usr/src/linux/arch/i386/kernel/setup.c` e é interpretada da seguinte forma:

```
bits  0-10:     deslocamento para início da memória, em blocos de 1024 bytes
bits 11-13:     sem utilização
bit     14:     Indicador se o disco em memória deve ser carregado
bit     15:     Indicador de prompt antes da carga do sistema de arquivos raiz
```

Caso o bit 15 esteja configurado, no processo de inicialização será solicitada a inserção de um novo disquete na unidade. Isso é necessário no caso de conjuntos de dois disquetes de inicialização.

Há dois casos, dependendo da construção de um único disquete ou de um conjunto de dois disquetes (inicialização e raiz).

1. Caso se esteja construindo um único disco, o sistema de arquivos raiz será colocado exatamente após o kernel, sendo o deslocamento então igual ao primeiro bloco livre (o que deve ser igual a `NÚCLEO_BLOCOS`). O bit 14 será configurado com 1 e o bit 15 com zeros.
2. Caso se esteja construindo um conjunto de dois disquetes, o sistema de arquivos raiz começará no bloco zero e o deslocamento será igual a zero. Bit 14 será igual a 1 e o bit 15 também será igual a 1.

Após calcular cuidadosamente o valor da palavra de disco em memória, deve-se configurá-la com `rdev -r`. Esteja seguro de utilizar valores *decimais*. Caso se esteja utilizando o LILO, o argumento `rdev` deve ser igual ao *caminho do kernel*, por exemplo `/mnt/vmlinuz`; caso o kernel tenha sido copiado com `dd`, deve-se usar o nome do dispositivo de disquetes (*por exemplo,* `/dev/fd0`).

```
rdev -r NÚCLEO_OU_UNIDADE_ DE _DISQUETE
```

Caso se esteja utilizando o LILO, a unidade de disquetes deve ser desmontada agora.

C.6.4 Transferindo o sistema de arquivos raiz

O último passo é a transferência do sistema de arquivos raiz.

- Caso o sistema de arquivos raiz seja colocado no *mesmo* disco que o kernel, a transferência deve ser efetuada utilizando-se o comando `dd` com a opção `seek`, a qual especifica quantos blocos devem ser ignorados até que a gravação tenha início:

  ```
  dd if=rootfs.gz of=/dev/fd0 bs=1k seek=BLOCOS_NÚCLEO
  ```

- Caso o sistema de arquivos raiz seja colocado em um *segundo* disco, deve-se remover o primeiro disquete e colocar o segundo na unidade, transferindo-se o sistema de arquivos raiz:

  ```
  dd if=rootfs.gz of=/dev/fd0 bs=1k
  ```

Parabéns, serviço concluído!

DEVE-SE SEMPRE TESTAR UM DISCO DE INICIALIZAÇÃO ANTES DE GUARDÁ-LO, PARA USO EM UMA EMERGÊNCIA

C.7 Problemas

Na construção de discos de inicialização, as primeiras tentativas normalmente gerarão discos com problemas. A abordagem de construção de um disco raiz é montar seus componentes a partir de um sistema já existente e tentar construir um disquete baseado nele que viabilize a carga do sistema até o momento em que mensagens possam ser apresentadas na console. Após esse passo, cabe verificar as mensagens e os erros apresentados, e ir corrigindo um a um, de acordo com o apresentado no console. Caso o sistema simplesmente trave, sem maiores explicações, encontrar a causa será um pouco mais difícil. Para ter-se um sistema que possa chegar ao estágio de enviar mensagens para o console, são requeridos diversos componentes, que devem estar presentes e corretamente configurados. O procedimento recomendado na investigação de um problema onde o sistema não apresente a sua causa é o seguinte:

- Verificar se o disco raiz contém realmente os diretórios necessários. É comum copiar de níveis errados e ter algo como `/discoraiz/bin` ao invés de `/bin` no disquete.
- Verificar se existe `/lib/libc.so` com a mesma ligação que aparece no diretório `/lib` do disco rígido.
- Verificar se todas as ligações simbólicas no diretório `/dev` existem no sistema de arquivos raiz do disquete, onde aquelas ligações devem ser para dispositivos que estão inclusos no disco raiz. Em particular, ligações para `/dev/console` são fundamentais em diversos casos.
- Verificar se os arquivos `/dev/tty1, /dev/null, /dev/zero, /dev/mem, /dev/ram` e `/dev/kmem` foram incluídos.
- Verificar se a configuração do kernel suporta todos os recursos requeridos até o momento de acesso ao sistema e se estão adicionados ao kernel de forma residente e não como módulos. *Suporte a discos em memória e ext2 devem estar residentes.*
- Verificar se as configurações do dispositivo raiz e do kernel e o disco em memória estão corretas.

Alguns destes aspectos gerais são melhor detalhados a seguir:

1. Esteja certo de que `init` foi incluído como `/sbin/init` ou `/bin/init` e que tem permissão de execução.
2. Execute `ldd init` para checar as bibliotecas necessárias à execução do init. Normalmente é necessária somente `libc.so`. Esteja certo de que todas as bibliotecas e carregadores foram incluídos.
3. Esteja certo de utilizar o carregador correto para as bibliotecas — `ld.so` para a.out ou `ld-linux.so` para ELF.
4. Verificar se o arquivo `/etc/inittab` no sistema de arquivos do disquete de inicialização, aponta para o programa `getty` (ou algum programa similar a `getty`, como por exemplo `agetty`, `mgetty` ou `getty_ps`). Cheque duplamente esse arquivo comparando-o com o disponível no disco rígido. Verifique as páginas do manual do programa que está sendo utilizado para estar seguro que a configuração está correta. `inittab` é possivelmente a parte mais repleta de detalhes devido à sua sintaxe e o seu conteúdo depende do programa usado no sistema. A única forma de não correr riscos é ler as páginas de manual do `init` e `inittab` e verificar exatamente o que o sistema está fazendo ao ser iniciado. Esteja seguro que `/etc/inittab` tem uma entrada de inicialização no sistema. Deve haver um comando de execução do programa de inicialização do sistema.
5. Assim como realizado com `init`, execute `ldd` em `getty` para verificar as dependências, esteja certo de que as bibliotecas necessárias e carregadores estão presentes no sistema de arquivos raiz.
6. Esteja seguro de ter incluído um shell script (por exemplo, `bash` ou `ash`), capaz de executar todos os programas rc .
7. Caso se tenha um arquivo `/etc/ld.so.cache` no disco de emergência, refaça-o.

Caso `init` comece, mas seja obtida a seguinte mensagem:

```
Id xxx respawning too fast: disabled for 5 minutes
```

ela é oriunda do `init`, normalmente indicando que os programas `getty` ou `login` estão sendo encerrados imediatamente após o seu início. Verifique os executáveis `getty` e `login` e as bibliotecas necessárias. Esteja seguro de que as chamadas em `/etc/inittab` estão corretas. Caso mensagens estranhas apareçam a partir do `getty`, pode significar que a chamada em `/etc/inittab` está errada. As opções disponíveis em *getty* variam bastante; assim como diferentes versões do `agetty` podem ter incompatibilidades na sintaxe das chamadas.

Caso ao se tentar executar algum programa, tal como `df`, presente no disco de emergência, e obtiver-se mensagens como: `df: não encontrado`, deve-se verificar se: (1) o diretório que contém o comando está configurado na variável de ambiente PATH, e (2) todas as bibliotecas e carregadores necessários ao programa estão presentes.

C.8 Diversos

C.8.1 Reduzindo o tamanho do sistema de arquivos raiz

Algumas vezes o sistema de arquivos raiz é muito grande para caber em um disquete, mesmo após a sua compactação. Seguem algumas formas de reduzir seu tamanho, listadas em ordem decrescente de efetividade:

Aumentar a densidade do disco

Por padrão, disquetes são formatados em 1400K, mas formatos de maior densidade estão disponíveis. `fdformat` poderá formatar nos seguintes tamanhos: 1600, 1680, 1722, 1743, 1760, 1840, e 1920. Muitos dispositivos de 1440K suportarão 1722K, e é este que sempre usamos para disquetes de inicialização. A página de manual `fdformat` e o arquivo `/usr/src/linux/Documentation/devices.txt` fornecem maiores informações sobre o tema.

Alterar o interpretador de comandos

Alguns dos mais populares interpretadores de comando para Linux, como `bash` e `tcsh`, são grandes e requerem diversas bibliotecas. Alternativas mais leves existem, tais como `ash`, `lsh`, `kiss` e `smash`, os quais são muito menores e requerem menos (ou nenhuma) bibliotecas. Muitos dos interpretadores alternativos estão disponíveis em `<http://sunsite.unc.edu/pub/Linux/system/shells/>`. Esteja certo porém de que o interpretador seja capaz de executar todos os comandos e programas configurados nos arquivos `rc` incluídos no disco de inicialização.

Diminuir as bibliotecas e os binários

Muitas bibliotecas e binários estão tipicamente com todos os símbolos de depuração inclusos, o que aumenta seu tamanho. Executando-se '`file`' nestes arquivos , ele apresentará a mensagem '`not stripped`' caso essa afirmativa seja verdadeira. Ao copiar binários para o seu sistema de arquivos raiz, é aconselhável utilizar:

```
objcopy --strip-all FROM TO
```

Ao copiar bibliotecas deve ser usado:

```
objcopy --strip-debug FROM TO
```

Mover arquivos não críticos para um disco de utilitários

Caso alguns binários não sejam necessários durante a inicialização ou no acesso ao sistema, eles podem ser movidos para um disquete adicional. Veja a Seção C.8.3 (Construindo um disquete de utilitários) para maiores detalhes. Deve-se ainda considerar também a movimentação de módulos.

C.8.2 Sistemas de arquivos raiz não residentes em discos na memória

A Seção C.4 (Construindo um sistema de arquivos raiz) apresenta instruções sobre como construir um sistema de arquivos raiz compactado o qual é carregado em um disco em memória quando o sistema é iniciado. Este método tem diversas vantagens e é o mais comumente utilizado. De qualquer forma, alguns sistemas com pouca memória podem não suportar toda a RAM necessária para isso; e nestes casos o sistema de arquivos raiz é montado diretamente ao invés de ser copiado para um disco em memória.

Tais sistemas de arquivos são na verdade mais simples de serem construídos do que sistemas compactados, uma vez que eles podem ser construídos tanto em disquetes como em qualquer outro dispositivo, e não têm que ser compactados. Apresentaremos os procedimentos necessários e as suas diferenças em relação ao descrito até aqui. Caso seja esta a sua opção, deve-se ter em mente que se terá *muito menos espaço* disponível.

1. Calcular o espaço disponível para os arquivos do sistema de arquivos raiz.

   Caso se esteja construindo um único disco de inicialização e raiz, todos os blocos do kernel mais os do sistema de arquivos raiz devem caber em um disquete.

2. Utilizar o programa `mke2fs`, para criar um sistema de arquivos raiz em um disquete com o tamanho apropriado.
3. Incluir os arquivos do sistema de arquivos conforme descrito acima.
4. Ao finalizar, desmontar o sistema de arquivos e transferí-lo para um arquivo de disco, *sem compactação*.
5. Transferir o kernel para um disquete, conforme descrito anteriormente. Ao calcular a palavra do disco em memória, **configurar o bit 14 para zero**, para indicar que o sistema de arquivos raiz não será carregado para um disco em memória. Executar o comando `rdev` conforme descrito.
6. Transferir o sistema de arquivos raiz conforme descrito anteriormente.

Há diversos atalhos que podem ser executados. Caso se esteja construindo um conjunto de dois disquetes, pode-se construir o sistema de arquivos raiz diretamente no segundo disquete, não sendo necessário transferi-lo para o disco rígido e após para o disquete novamente. Ainda, caso se esteja construindo um único disquete de raiz e inicialização utilizando-se o LILO, pode-se construir um *único* sistema de arquivos no disco inteiro, contendo o kernel, arquivos do LILO e do raiz, e simplesmente executar o LILO como o último passo.

C.8.3 Construindo um disquete de utilitários

Construir um disquete de utilitários é bastante simples, basta simplesmente criar um sistema de arquivos em um disquete formatado e copiar os arquivos para ele. Para usá-lo com o disco de inicialização, deve-se montá-lo manualmente após o sistema ser carregado.

Conforme descrito anteriormente, o disquete de utilitários pode ser montado como `/usr`. Neste caso, os binários podem ser colocados no diretório `/bin` no disquete de utilitários, sendo que uma indicação a `/usr/bin` na variável de ambiente de caminho permitirá o acesso direto a eles. Bibliotecas adicionais podem ser colocadas no diretório `/lib` no disquete de utilitários.

Há vários pontos importantes para atentar-se ao se construir um disco de utilitários, a saber:

1. *Não* colocar binários ou bibliotecas que sejam críticos para o início do sistema, uma vez que o disquete não será montado antes da inicialização do sistema.
2. Não é possível acessar uma unidade de disquete e uma unidade de fita simultaneamente. Isso significa que caso se tenha uma unidade de fita, não será possível acessá-la enquanto o disco de utilitários estiver montado.
3. O acesso aos arquivos no disco de utilitários será lento.

Apêndice C.14 (Listas de exemplo do conteúdo do disco de utilitários) mostra um lista de exemplos de arquivos em um disquete de utilitários. Aqui estão colocadas algumas idéias que podem ser úteis: programas para examinar e manipular discos (`format, fdisk`) e sistemas de arquivos (`mke2fs, fsck, isofs.o`), um editor de textos leve (`elvis, jove`), utilitários de compactação e arquivamento (`gzip, tar, cpio, afio`), utilitários para fitas (`mt, tob, taper`), comunicação (`ppp.o, slip.o, minicom`) e dispositivos (`setserial, mknod`).

C.9 Como os profissionais fazem isso

É possível perceber que os discos de inicialização das maiores distribuições, tais como Conectiva Linux, Red Hat, Slackware ou Debian, parecem ser mais sofisticados do que o descrito neste documento. Discos de inicialização de distribuições profissionais são baseados nos mesmos princípios aqui descritos, mas empregam alguns detalhes adicionais, uma vez que devem atender a um número maior de sistemas e situações. Inicialmente, eles devem ser capazes de funcionar com uma grande variedade de hardwares, sendo necessário então interagir com o usuário e carregar diversos arquivos de dispositivos. Segundo, eles devem ser preparados para trabalhar com diversas opções de instalação, com diferentes níveis de automação. Finalmente, distribuições normalmente combinam funcionalidades de instalação e emergências.

Alguns discos de inicialização contêm uma funcionalidade chamada **initrd** (**disco em memória inicial**). Esta funcionalidade foi introduzida com a versão 2.0.X e provê grande flexibilidade, permitindo que o kernel seja carregado em duas fases. Quando o kernel é inicializado, ele inicialmente carrega uma imagem do disco em memória a partir do disquete de inicialização. O disco em memória inicial é um sistema de arquivos raiz contendo um programa que é executado antes que o raiz verdadeiro seja carregado. Este programa normalmente inspeciona o ambiente e solicita que o usuário selecione entre diversas opções de inicialização, tais como o real dispositivo que deve ser utilizado para a carga do sistema de arquivos raiz. Tipicamente ele carrega módulos adicionais não incluídos no kernel. Quando o programa inicial termina, o kernel carrega o raiz original e o processo de inicialização continua normalmente. Para maiores informações sobre o `initrd`, veja `/usr/src/linux/Documentation/initrd.txt` e `<ftp://elserv.ffm.fgan.de/pub/linux/loadlin-1.6/initrd-example.tgz>`.

As informações a seguir são um resumo dos discos de instalação de cada uma das distribuições, baseado na inspeção de seus sistemas de arquivos e dos códigos fontes. Não podemos garantir a exata acuidade das informações aqui descritas, ou se elas foram alteradas em novas versões.

Slackware (v.3.1) usa uma inicialização similar a um LILO descrito na seção C.6.1 (Transferindo o kernel com o LILO). O disco de inicialização do Slackware apresenta uma mensagem (“`Bem-vindo ao disco de inicialização do kernel do Slackware Linux!`”) usando o parâmetro `message` do LILO. Após, o usuário é instruído a entrar com os parâmetros de inicialização, se necessários. Após a inicialização, um sistema de arquivos raiz é carregado de um segundo disco. O usuário executa o programa `setup` que inicia a instalação. Ao invés de usar um kernel modular, Slackware provê uma série de diferentes kernels e depende do usuário a escolha do mais adequado aos seus requisitos de hardware.

RedHat (v.4.0) também utiliza a inicialização do LILO. Ele carrega um disco em memória compactado a partir do primeiro disco, o qual executa um programa `init` customizado. Este programa solicita os arquivos de controle de dispositivos e carrega-os de um disco suplementar quando necessário.

Debian (v.1.3) é provavelmente o mais sofisticado disco de instalação. Ele utiliza o carregador SYSLINUX para permitir várias opções de inicialização, usando então uma imagem do `initrd` para guiar o usuário pelo processo de instalação. Aparenta usar tanto um `init` quanto um interpretador customizados.

C.10 Lista das Perguntas Mais Frequentes (FAQ)

Q. Eu carrego meus discos de inicialização e raiz e nada acontece. O que posso fazer?

Veja a Seção C.7 (Problemas).

Q. Como o disco de inicialização da Conectiva/Red Hat/Slackware/Debian funcionam?

Veja a Seção C.9 (Como os profisisonais fazem isso).

Q. Como eu posso construir um disquete de inicialização com o arquivo de controle de dispositivos XYZ?

A forma mais simples é obter um kernel do Slackware em um site espelho mais próximo. Os kernels do Slackware são genéricos e tentam incluir arquivos de controle para o maior número de dispositivos possível. Então caso se tenha uma controladora IDE ou SCSI, as chances são grandes de se ter um arquivo de controle para elas em um kernel do Slackware. Vá para o diretório `a1` e selecione IDE ou SCSI dependendo do tipo de controladora que se tenha. Verifique o arquivo xxxxkern.cfg do kernel selecionado para ver os arquivos de controle que foram incluídos neste kernel. Caso o dispositivo que se deseje instalar esteja incluído na lista, então o kernel correspondente deve inicializar o seu computador. Transfira o arquivo xxxxkern.tgz e copie para o seu disquete de inicialização conforme descrito acima, na seção de construção de discos de inicialização.

Deve-se checar o dispositivo raiz no kernel usando-se o comando rdev:

```
rdev zImage
```

`rdev` mostrará o dispositivo raiz atual do kernel. Caso este não seja o mesmo desejado, deve-se usar o `rdev` para alterá-lo. Por exemplo, um kernel testado apontava para `/dev/sda2`, porém a partição raiz SCSI era `/dev/sda8`. Para usar um disquete raiz, deve-se utilizar o comando:

```
rdev zImage /dev/fd0
```

Caso se queira saber como configurar um disquete de inicialização do Slackware, sugere-se verificar o Guia de Instalação Linux ou obter uma distribuição Slackware. Veja a seção denominada "Referências".

Q. Como atualizar um disco de inicialização com um novo kernel?

Simplesmente copie o novo kernel em um disquete de inicialização utilizando o comando dd para um disquete sem sistema de arquivos, ou através do comando cp para um disco de inicialização e raiz. Verifique a seção denominada "inicialização" para detalhes sobre a criação de um disquete de inicialização. A descrição aplica-se perfeitamente à atualização do kernel em um disquete de inicialização.

Q. Como atualizar o disquete raiz com novos arquivos?

A maneira mais simples é copiar o sistema de arquivos do disco raiz de volta ao DISPOSITIVO utilizado, seção C.4.2 (Criando um sistema de arquivos). Após monte o sistema de arquivos e faça as alterações necessárias. Deve-se lembrar sempre onde o sistema de arquivos começa e quantos blocos ele ocupa:

```
dd if=/dev/fd0 bs=1k skip=INÍCIO_RAIZ  count=BLOCOS | gunzip > DISPOSITIVO
mount -t ext2 DISPOSITIVO /mnt
```

Após as alterações serem concluídas, proceda de acordo com a Seção C.4.6 (Empacotando) e transfira o sistema de arquivos raiz de volta para o disco. Não se deve retransferir o kernel ou recalcular a palavra do disco em memória caso não tenha havido alterações do ponto de início do novo sistema de arquivos raiz.

Q. Como remover o LILO para ser possível utilizar a inicialização DOS novamente?

No Linux pode-se executar:

```
/sbin/lilo -u
```

Pode-se ainda usar o comando `dd` para utilizar a cópia de segurança salva pelo LILO do setor de inicialização. Veja na documentação do LILO maiores informações.

No DOS ou Windows pode-se executar o seguinte comando DOS:

```
FDISK /MBR
```

MBR significa Registro Master de Inicialização (Master Boot Record), e após o comando acima o setor de inicialização recebe registros DOS sem alterar a tabela de partições. Alguns puristas não concordam com essa abordagem, mas mesmo o autor do LILO, Werner Almesberger, sugere isso, além de ser simples e funcional.

Q. Como inicializar o sistema se o kernel *e* o disco de inicialização foram perdidos?

Caso não se tenha um disco de inicialização extra, provavelmente o método mais simples será obter um kernel do Slackware para a sua controladora de discos (IDE ou SCSI) conforme descrito anteriormente em "Como construir um disco de inicialização com o arquivo de controle de dispositivos XYZ?". Pode-se iniciar o sistema com esse kernel e procurar reparar o que estiver danificado.

O kernel obtido pode não conter o tipo de disco e a partição que se deseje. Por exemplo, kernel genérico do Slackware para controladoras SCSI tem o dispositivo raiz configurado para `/dev/sda2`, e eventualmente a partição raiz pode estar em `/dev/sda8`. Neste caso o dispositivo raiz do kernel deve ser alterado.

Pode-se mudar os parâmetros de dispositivo raiz e disco em memória no kernel, mesmo que tudo o que tenha seja um kernel ou mesmo a partir de outro sistema operacional, como por exemplo DOS.

`rdev` altera o kernel através de alterações de valores de deslocamentos fixos do arquivo do kernel, então pode-se utilizar um editor em hexadecimal em qualquer sistema disponível, como por exemplo o Editor de Disco de Utilitários Norton sob DOS. Deve-se checar se é necessário mudar os valores do kernel nos seguintes deslocamentos:

```
HEX     DEC  DESCRIPTION
0x01F8  504  Byte inferior da palavra do disco em memória
0x01F9  505  Byte superior da palavra do disco em memória
0x01FC  508  Número menor do dispositivo raiz - vide abaixo
0X01FD  509  Número maior do dispositivo raiz - vide abaixo
```

A interpretação da palavra do disco em memória está descrita na seção C.6.3 (Configurando o disco em memória).

O número maior e menor do dispositivo devem ser configurados caso se deseje montar o sistema de arquivos raiz nele. Alguns valores importantes são:

```
DISPOSITIVO      MAJOR MINOR
/dev/fd0             2     0      primeira unidade de disquetes
/dev/hda1            3     1      partição 1 no primeiro disco IDE
/dev/sda1            8     1      partição 1 no primeiro disco SCSI
/dev/sda8            8     8      partição 8 no primeiro disco SCSI
```

Uma vez que estes valores estejam configurados, pode-se gravar o arquivo utilizando tanto o editor Norton, quanto um programa chamado `rawrite.exe`. Este programa é incluído em todas as distribuições, e é executado em ambientes DOS, sendo capaz de gravar os dados de forma direta, ao invés de gravar através do sistema de arquivos. Caso se utilize o Norton deve-se gravar o arquivo em um disco físico com início igual ao do disco de inicialização.

Q. Como fazer cópias adicionais dos disquetes de inicialização e raiz?

Como a mídia magnética pode deteriorar-se com o tempo, deve-se manter diversas cópias dos discos de emergência, para as situações em que o original não funcione.

A forma mais simples de fazer cópias de qualquer disquete, inclusive disquetes que podem ser inicializados e disquetes de utilitários, é através da utilização do comando `dd` para cópia do disquete original em um disco rígido e após o mesmo comando para copiar do disco rígido para os diversos disquetes. Note que não se deve montar os disquetes, porque o comando `dd` utiliza o acesso direto ao dispositivo.

Para copiar o disquete original, execute o comando:

```
dd if=NOME_DISPOSITIVO of=NOME_ARQUIVO
onde NOME_DISPOSITIVO é o nome da unidade de disquetes
e NOME_ARQUIVO é o nome do arquivo de saída (no disco rígido)
```

Omitindo-se o parâmetro `count` faz com que o comando copie todo o disquete (2880 blocos de alta densidade).

Para copiar o arquivo resultante para um novo disquete, deve-se inserir o disquete na unidade e executar o comando:

```
dd if=NOME_ARQUIVO of=NOME_DISPOSITIVO
```

Note que o exemplo acima assume que se tenha somente uma unidade de gravação de disquetes. Caso se tenha duas de mesmo tipo, pode-se copiar o disquete usando-se o comando:

```
dd if=/dev/fd0 of=/dev/fd1
```

Q. Como inicializar o sistema sem ter que digitar "ahaxxxx=nn,nn,nn" toda vez?

Quando um dispositivo não pode ser detectado automaticamente, é necessário fornecer ao kernel os parâmetros através do comando de dispositivos, como por exemplo:

```
aha152x=0x340,11,3,1
```

Este parâmetro pode ser fornecido pelo LILO de diversas formas:

- Informando o comando toda a vez que o sistema for inicializado via LILO. A opção menos aconselhável.
- Usando o parâmetro "`lock`" para armazenar a linha de comando como a linha padrão de comando, fazendo com que o LILO use estas opções toda a vez que o sistema for carregado.
- Utilizando o parâmetro `append=` no arquivo de configuração do LILO. Note que o parâmetro deve estar entre aspas.

Por exemplo, uma linha de comando usando os parâmetros acima poderia ser:

```
zImage  aha152x=0x340,11,3,1 root=/dev/sda1 lock
```

Este comando passa os parâmetros do dispositivo, configura o dispositivo raiz em `/dev/sda1` e salva todo o comando para reutilização futura.

Um exemplo de comando append:

```
\{append} = "aha152x=0x340,11,3,1"
```

Note que o parâmetro não deve estar entre aspas na linha de comando, mas é obrigatório que assim esteja no comando append.

Note ainda que para que o parâmetro seja ativado, o kernel deve conter o módulo para o dispositivo assinalado. Caso contrário, o comando não surtirá efeito algum e o kernel deverá ser reconstruído para incluir o módulo requerido. Para maiores detalhes sobre a reconstrução do kernel, mude o diretório para `/usr/src/linux` e veja o arquivo README, e leia o FAQ e o tutorial de instalação. Alternativamente pode-se obter um kernel genérico e instalá-lo.

É extremamente indicada a leitura da documentação LILO antes de se tentar instalá-lo. Usos indevidos do comando `BOOT` podem causar danos às partições.

Q. Durante a inicialização ocorreu o erro "`A: não pode executar B`". Por quê?

Há diversos casos em que nomes de programas encontram-se dentro do código de vários utilitários. Estes casos não ocorrem em toda parte, mas pode ser uma explicação de porquê um executável aparentemente pode não ser encontrado, apesar de estar presente no sistema. Pode-se descobrir se um determinado programa tem o nome de outro dentro de seu código usando-se o comando `strings` e conectando sua saída com o comando `grep`.

Exemplos conhecidos de localizações predefinidas:

- Shutdown em algumas versões tem o programa `/etc/reboot` predefinido, sendo que `reboot` deverá estar no diretório `/etc` .
- `init` pode causar alguns problemas, caso o kernel não consiga encontrá-lo em `init`.

Para corrigir estes problemas, deve-se ou mover os programas para o diretório correto, ou mudar os arquivos de configuração (por exemplo `inittab`) para apontarem para o diretório correto. Em caso de dúvidas, ponha os programas no mesmo diretório em que eles estavam no disco rígido, e utilize os mesmos `inittab` e `/etc/rc.d` da forma como eles estão presentes.

Q. Meu kernel tem suporte a disco em memória, mas aparece com 0 Kb de espaço.

Quando isso ocorre, uma mensagem do kernel aparecerá durante a inicialização:

```
Dispositivo de disco em memória inicializado:
16 discos em memória com tamanho de OK.
```

Isso se deve provavelmente à definição do tamanho em zero através dos parâmetros de kernel em tempo de inicialização. Algo como:

```
ramdisk=0
```

Isso foi incluído como exemplo de configuração do LILO em algumas distribuições antigas, e foi colocado para sobrepor-se a parâmetros anteriores de configuração do kernel. Caso essa linha esteja presente, deve ser retirada.

Note que ao se tentar utilizar um disco em memória com tamanho igual a zero, o comportamento do sistema é imprevisível, e pode resultar em travamentos do kernel.

C.11 Recursos e Endereços

Nesta seção, *vvv* é usado no lugar de versões dos nomes dos pacotes, a fim de evitar a referência a uma versão específica. Ao recuperar um pacote, deve-se procurar usar sempre a versão mais recente, a menos que haja alguma boa razão para fazer o contrário.

C.11.1 Discos de Inicialização Pré-Configurados

Há diversas fontes de discos de distribuições. *Por favor use um dos sites espelhos para reduzir o tráfego nestas máquinas.*

- *Discos de inicialização Slackware* <http://sunsite.unc.edu/pub/Linux/distributions/slackware/bootdsks.144/> e *Sites espelho Slackware* <http://sunsite.unc.edu/pub/Linux/distributions/slackware/MIRRORS.TXT>
- *Discos de inicialização Red Hat* <http://sunsite.unc.edu/pub/Linux/distributions/redhat/current/i386/images/> e *Sites espelho Red Hat* <http://www.redhat.com/ftp.html>
- *Discos de inicialização Debian* <ftp://ftp.debian.org/pub/debian/stable/disks-i386> e *Sites espelho Debian* <ftp://ftp.debian.org/debian/README.mirrors>

Em adição aos discos das distribuições, as seguintes imagens de discos de emergência estão disponíveis:

- tomsrtbt, de Tom Oehser, é um único disco de inicialização e raiz, baseado no kernel 2.0.33, com uma grande lista de funcionalidades e programas de suporte. Suporta IDE, SCSI, fita, placas de rede, PCMCIA e mais. Mais de 100 utilitários e ferramentas estão inclusas para correção e restauração de discos. O pacote ainda inclui programas para desmontagem e reconstrução de imagens, permitindo que novos itens possam ser incluídos. <http://www.toms.net/~toehser/rb/tomsrtbt-current.tar.gz>
  <http://sunsite.unc.edu/pub/Linux/system/recovery/>
- rescue02, de John Comyns, é um disco de emergência baseado no kernel 1.3.84, com suporte a IDE e Adaptec 1542 e NCR53C7,8xx. Usa binários ELF, porém contém comandos suficientes para ser utilizado em qualquer sistema. Há módulos que podem ser carregados após a inicialização do sistema para todas as placas SCSI. Provavelmente não funcionará em sistemas com 4 Mb de RAM, uma vez que utiliza um disco de 3 Mb. <http://sunsite.unc.edu/pub/Linux/system/recovery/rescue02.zip>
- resque_disk-2.0.22, de Sergei Viznyuk, é um disco de inicialização e raiz baseado no kernel 2.0.22 com suporte a IDE, muitas controladoras SCSI e ELF/OUT. Inclui ainda diversos módulos e utilitários para reparar e restaurar discos rígidos. <http://sunsite.unc.edu/pub/Linux/system/recovery/resque_disk-vvv.tar.gz>
- Imagens cramdisk baseado no kernel 2.0.33, disponível para máquinas com 4 e 8 Mb. Incluem emulador matemático e suporte à comunicação (PPP e programas de discagem, NE2000, 3C509), ou suporte a dispositivos ZIP paralelos. Estas imagens serão capazes de inicializar um 386 com 4 Mb de RAM. Suporte a MSDOS está incluso, podendo-se assim transmitir da Internet para uma partição DOS. <http://sunsite.unc.edu/pub/Linux/system/recovery/images/>

C.11.2 Discos de Emergência

Diversos pacotes de criação de discos de emergência estão disponíveis em sunsite.unc.edu. Com estes pacotes pode-se especificar um conjunto de arquivos para inclusão e o software automatiza (em vários níveis) a criação do disco de inicialização. Veja <http://sunsite.unc.edu/pub/Linux/system/recovery/!INDEX.html> para maiores informações. **Verifique as datas dos arquivos cuidadosamente** — alguns destes pacotes podem estar desatualizados há vários anos e podem não suportar a criação de sistemas de arquivos raiz compactados carregados em discos em memória. Até onde conhecemos Yard é o único pacote que poderá fazer isso.

C.11.3 Programas de Lote de Graham Chapman

Graham Chapman escreveu um conjunto de programas que podem ser bastante úteis como exemplos de criação de discos de inicialização. Em versões anteriores deste tutorial os programas apareciam como um apêndice, mas foram apagados deste documento e colocados em uma página web: <http://www.zeta.org.au/grahamc/linux.html>

Pode ser conveniente o uso destes programas, mas deve-se ler cuidadosamente as instruções – por exemplo ao especificar uma área de troca incorreta, pode-se apagar o sistema de arquivos raiz definitivamente. Esteja seguro de que esteja bem configurado antes de usar os programas.

C.11.4 LILO — O carregador Linux

Escrito por Werner Almesberger. Excelente carregador da inicialização do sistema, e a documentação inclui informações sobre o setor de inicialização e os estágios anteriores do processo de carga de um sistema.

Ftp em `<ftp://tsx-11.mit.edu/pub/linux/packages/lilo/lilo.vvv.tar.gz>`. Está também disponível no Sunsite e seus diversos sites espelhos.

C.11.5 Perguntas Mais Freqüentes e Como Fazer

Estão disponíveis em diversos locais. Pode-se verificar os grupos de notícias da Usenet em `news.answers` e `comp.os.linux.announce`.

Perguntas mais freqüentes em `<http://sunsite.unc.edu/pub/Linux/docs/faqs/linux-faq>` e Como Fazer em `<http://sunsite.unc.edu/pub/Linux/docs/HOWTO>`.

Mais documentação sobre o Linux pode ser encontrada em *Página do Projeto de Documentação Linux* `<http://sunsite.unc.edu/LDP/>`.

Caso o problema seja seríssimo pode-se enviar uma mensagem em inglês para `mail-server@rtfm.mit.edu` com a palavra "`help`" na mensagem.

C.11.6 Uso do Disco em Memória

Uma excelente descrição de como funciona o novo código de disco em memória pode ser encontrada com a documentação do kernel do Linux. Veja em `/usr/src/linux/Documentation/ramdisk.txt`. Foi escrito por Paul Gortmaker e inclui uma seção sobre a criação de discos em memória compactados.

C.11.7 O processo de inicialização do Linux

Para maiores detalhes sobre o processo de inicialização do Linux, seguem algumas indicações:

- O Guia de Administração do Sistema Linux contém uma seção sobre o processo. Veja em `<http://sunsite.unc.edu/LDP/LDP/sag-0.5/node68.html>`
- Visão Geral do LILO `<http://sunsite.unc.edu/pub/Linux/system/boot/lilo/lilo-t-20.ps.gz>` tem uma visão tecnicamente detalhada e definitiva sobre o processo de inicialização e como o kernel é carregado.
- O código fonte é o último guia. Abaixo seguem alguns arquivos do kernel relacionados com o processo de inicialização. Os fontes do Linux podem ser obtidos em `/usr/src/linux` em sistemas Linux; ou alternativamente com Shigio Yamaguchi (shigio@wafu.netgate.net) que tem um kernel em hipertexto em `<http://wafu.netgate.net/linux/>`.

  arch/i386/boot/bootsect.S,setup.S
  : Contém o código Assembler para o setor de inicialização.

  arch/i386/boot/compressed/misc.c
  : Contém o código para descompactar o kernel.

  arch/i386/kernel/
  : Diretório contendo o código de inicialização do kernel. `setup.c` contém a palavra de configuração do disco em memória.

drivers/block/rd.c

Contém o arquivo de controle do disco em memória. Os procedimentos `rd_load` e `rd_load_image` carregam os blocos de um dispositivo em um disco em memória. O procedimento `identify_ramdisk_image` determina o tipo do sistema de arquivos encontrado e se ele é compactado.

C.12 Códigos de Erros de Inicialização do LILO

Questões sobre esses códigos de erros, têm sido freqüentes na Usenet, sendo que incluímos aqui como um serviço público. Este sumário foi desenvolvido a partir da Documentação do Usuário LILO de Werner Almsberger, disponível em `<ftp://lrcftp.epfl.ch:/pub/linux/local/lilo/lilo.u.19.ps.gz>`.

Quando o LILO carrega a si próprio, ele apresenta a palavra "`LILO`". Cada letra é apresentada antes da execução de algum processo, assim sendo as letras podem ser um indicador do estágio atingido e da origem do problema.

Nenhuma parte do LILO foi carregada. LILO pode não estar instalado ou a partição no qual o setor de inicialização está localizado não está ativa.

L

O primeiro estágio do carregador foi iniciado e executado, mas não foi possível carregar o segundo estágio. Os códigos de erro com dois dígitos indicam o tipo de problema (Veja ainda "Códigos de Erros de Discos"). Esta condição indica normalmente a falha na mídia ou erro de geometria (por exemplo parâmetros de disco incorretos).

LI

O primeiro estágio foi capaz de carregar o segundo, mas falhou na sua execução. Isso pode ser causado por erro de geometria ou pela movimentação do /boot/boot.b sem a execução do instalador.

LIL

O segundo estágio conseguiu ser iniciado, mas não pode carregar a tabela de descritores do arquivo map. Isso normalmente é causado por falha na mídia ou erro de geometria.

LIL?

O segundo estágio do LILO foi carregado para um endereço incorreto. Isso é tipicamente causado por erros de geometria ou pela movimentação do arquivo `/boot/boot.b` sem a execução do instalador.

LIL-

A tabela de descritores está corrompida. Isso pode ser causado por erros de geometria ou pela movimentação do arquivo `/boot/boot.b` sem a execução do instalador.

LILO

Todas as partes do LILO foram carregadas.

Caso o BIOS apresente algum erro quando o LILO estiver tentando carregar uma imagem de inicialização, o respectivo código de erro é apresentado. Estes códigos variam de `0x00` até `0xbb`. Veja o Guia do Usuário LILO para uma maior explicação sobre este tema.

C.13 Listas de exemplo do conteúdo do disco de inicialização

Aqui está o conteúdo dos disquetes raiz e utilitários de Graham. Estas listas são apresentadas como um exemplo dos arquivos incluídos em um sistema funcional. Graham adicionou algumas notas explicativas que parecem muito úteis.

```
total 18
drwxr-xr-x   2 root     root         1024 Jul 29 21:16 bin/
drwxr-xr-x   2 root     root         9216 Jul 28 16:21 dev/
drwxr-xr-x   3 root     root         1024 Jul 29 20:25 etc/
drwxr-xr-x   2 root     root         1024 Jul 28 19:53 lib/
drwxr-xr-x   2 root     root         1024 Jul 24 22:47 mnt/
drwxr-xr-x   2 root     root         1024 Jul 24 22:47 proc/
drwxr-xr-x   2 root     root         1024 Jul 28 19:07 sbin/
drwxr-xr-x   2 root     root         1024 Jul 29 20:57 tmp/
drwxr-xr-x   4 root     root         1024 Jul 29 21:35 usr/
drwxr-xr-x   3 root     root         1024 Jul 28 19:52 var/

/bin:
total 713
-rwxr-xr-x   1 root     bin          7737 Jul 24 22:16 cat*
-rwxr-xr-x   1 root     bin          9232 Jul 24 22:48 chmod*
-rwxr-xr-x   1 root     bin          8156 Jul 24 22:48 chown*
-rwxr-xr-x   1 root     bin         19652 Jul 24 22:48 cp*
-rwxr-xr-x   1 root     root         8313 Jul 29 21:16 cut*
-rwxr-xr-x   1 root     bin         12136 Jul 24 22:48 dd*
-rwxr-xr-x   1 root     bin          9308 Jul 24 22:48 df*
-rwxr-xr-x   1 root     root         9036 Jul 29 20:24 dircolors*
-rwxr-xr-x   1 root     bin          9064 Jul 24 22:48 du*
-rwxr-x---   1 root     bin         69252 Jul 24 22:51 e2fsck*
-rwxr-xr-x   1 root     bin          5361 Jul 24 22:48 echo*
-rwxr-xr-x   1 root     bin          5696 Jul 24 22:16 hostname*
-rwxr-xr-x   1 root     bin          6596 Jul 24 22:49 kill*
-rwxr-xr-x   1 root     bin         10644 Jul 24 22:17 ln*
-rwxr-xr-x   1 root     bin         13508 Jul 24 22:17 login*
```

```
-rwxr-xr-x   1 root     bin         26976 Jul 24 22:17 ls*
-rwxr-xr-x   1 root     bin          7416 Jul 24 22:49 mkdir*
-rwxr-x---   1 root     bin         34596 Jul 24 22:51 mke2fs*
-rwxr-xr-x   1 root     bin          6712 Jul 24 22:49 mknod*
-rwxr-xr-x   1 root     bin         20304 Jul 24 22:17 more*
-rwxr-xr-x   1 root     bin         24704 Jul 24 22:17 mount*
-rwxr-xr-x   1 root     bin         12464 Jul 24 22:17 mv*
-rwxr-xr-x   1 root     bin         20829 Jul 24 22:50 ps*
-rwxr-xr-x   1 root     bin          9424 Jul 24 22:50 rm*
-rwxr-xr-x   1 root     bin          4344 Jul 24 22:50 rmdir*
-rwxr-xr-x   1 root     root       299649 Jul 27 14:12 sh*
-rwxr-xr-x   1 root     bin          9853 Jul 24 22:17 su*
-rwxr-xr-x   1 root     bin           380 Jul 27 14:12 sync*
-rwxr-xr-x   1 root     bin         13620 Jul 24 22:17 umount*
-rwxr-xr-x   1 root     root         5013 Jul 29 20:03 uname*

/dev:
total 0
lrwxrwxrwx   1 root     root           10 Jul 24 22:34 cdrom -> /dev/sbpcd
crw--w--w-   1 root     tty        4,   0 Jul 24 21:49 console
brw-rw----   1 root     floppy     2,   0 Apr 28  1995 fd0
lrwxrwxrwx   1 root     root            4 Jul 24 22:34 ftape -> rft0
crw-rw-rw-   1 root     sys       10,   2 Jul 18  1994 inportbm
crw-rw----   1 root     kmem       1,   2 Jul 28 16:21 kmem
crw-rw----   1 root     kmem       1,   1 Jul 18  1994 mem
lrwxrwxrwx   1 root     root            4 Jul 24 22:34 modem -> cua0
lrwxrwxrwx   1 root     root            4 Jul 24 22:34 mouse -> cua1
crw-rw-rw-   1 root     sys        1,   3 Jul 18  1994 null
brw-rw----   1 root     disk       1,   1 Jul 18  1994 ram
crw-rw----   1 root     disk      27,   0 Jul 18  1994 rft0
brw-rw----   1 root     disk      25,   0 Jul 19  1994 sbpcd
***  Foram incluídos arquivos de dispositivos para as partições SCSI em uso
***  Caso sejam utilizados discos IDE, deve-se usar /dev/hdxx .
brw-rw----   1 root     disk       8,   0 Apr 29  1995 sda
brw-rw----   1 root     disk       8,   6 Apr 29  1995 sda6
brw-rw----   1 root     disk       8,   7 Apr 29  1995 sda7
brw-rw----   1 root     disk       8,   8 Apr 29  1995 sda8
lrwxrwxrwx   1 root     root            7 Jul 28 12:56 systty -> console
*** esta ligação de systty para a console é obrigatória
crw-rw-rw-   1 root     tty        5,   0 Jul 18  1994 tty
crw--w--w-   1 root     tty        4,   0 Jul 18  1994 tty0
crw--w----   1 root     tty        4,   1 Jul 24 22:33 tty1
crw--w----   1 root     tty        4,   2 Jul 24 22:34 tty2
crw--w--w-   1 root     root       4,   3 Jul 24 21:49 tty3
crw--w--w-   1 root     root       4,   4 Jul 24 21:49 tty4
crw--w--w-   1 root     root       4,   5 Jul 24 21:49 tty5
crw--w--w-   1 root     root       4,   6 Jul 24 21:49 tty6
crw-rw-rw-   1 root     tty        4,   7 Jul 18  1994 tty7
crw-rw-rw-   1 root     tty        4,   8 Jul 18  1994 tty8
crw-rw-rw-   1 root     tty        4,   9 Jul 19  1994 tty9
crw-rw-rw-   1 root     sys        1,   5 Jul 18  1994 zero

/etc:
total 20
-rw-r--r--   1 root     root         2167 Jul 29 20:25 DIR_COLORS
```

```
-rw-r--r--   1 root     root           20 Jul 28 12:37 HOSTNAME
-rw-r--r--   1 root     root          109 Jul 24 22:57 fstab
-rw-r--r--   1 root     root          271 Jul 24 22:21 group
-rw-r--r--   1 root     root         2353 Jul 24 22:27 inittab
-rw-r--r--   1 root     root            0 Jul 29 21:02 issue
-rw-r--r--   1 root     root         2881 Jul 28 19:38 ld.so.cache
*** Diversos erros ocorrem na inicialização se ld.so.cache não está presente,mas
*** esteja seguro que  ldconfig está incluído e pode ser executado em rc.x para
*** atualizá-lo.
-rw-r--r--   1 root     root           12 Jul 24 22:22 motd
-rw-r--r--   1 root     root          606 Jul 28 19:25 passwd
-rw-r--r--   1 root     root         1065 Jul 24 22:21 profile
drwxr-xr-x   2 root     root         1024 Jul 29 21:01 rc.d/
-rw-r--r--   1 root     root           18 Jul 24 22:21 shells
-rw-r--r--   1 root     root          774 Jul 28 13:43 termcap
-rw-r--r--   1 root     root          126 Jul 28 13:44 ttys
-rw-r--r--   1 root     root            0 Jul 24 22:47 utmp

/etc/rc.d:
total 5
*** Não me importo muito com programas de encerramento do sistema -
*** tudo é executado em um disco em memória, não havendo muitas coisas
*** para finalizar.
-rwxr-xr-x   1 root     root         1158 Jul 24 22:23 rc.K*
-rwxr-xr-x   1 root     root         1151 Jul 28 19:08 rc.M*
-rwxr-xr-x   1 root     root          507 Jul 29 20:25 rc.S*

/lib:
total 588
*** Tenho um sistema ELF, tendo sido incluído o carregador loader ld-linux.so.
*** Caso se esteja ainda utilizando o  a.out, então deve-se incluir ld.so.
*** Deve-se usar o comando file para verificar quais  bibliotecas devem ser incluídas.
lrwxrwxrwx   1 root     root           17 Jul 24 23:36 ld-linux.so.1 -> ld-linux.so.1.7.3*
-rwxr-xr-x   1 root     root        20722 Aug 15  1995 ld-linux.so.1.7.3*
lrwxrwxrwx   1 root     root           13 Jul 24 23:36 libc.so.5 -> libc.so.5.0.9*
-rwxr-xr-x   1 root     root       562683 May 19  1995 libc.so.5.0.9*
***  Deve-se incluir   libtermcap
lrwxrwxrwx   1 root     root           19 Jul 28 19:53 libtermcap.so.2 -> libtermcap.so.2.0.0*
-rwxr-xr-x   1 root     root        11360 May 19  1995 libtermcap.so.2.0.0*

/mnt:
total 0

/proc:
total 0

/sbin:
total 191
***  Utilizo Slackware, o qual usa  agetty. Muitos sistemas usam  getty.
***  Verifique em /etc/inittab para ver qual o necessário. Note que  (a)getty e login
***  são necessários para que se possa fazer algo no sistema.
-rwxr-xr-x   1 root     bin         11309 Jul 24 22:54 agetty*
-rwxr-xr-x   1 root     bin          5204 Jul 24 22:19 halt*
***  Obrigatório na inicialização do sistema
-rwxr-xr-x   1 root     bin         20592 Jul 24 22:19 init*
```

```
-rwxr-xr-x   1 root     root        86020 Jul 28 19:07 ldconfig*
-rwxr-xr-x   1 root     bin          5329 Jul 27 14:10 mkswap*
-rwxr-xr-x   1 root     root         5204 Jul 24 22:20 reboot*
-rwxr-xr-x   1 root     root         6024 Jul 24 22:20 rdev*
-rwxr-xr-x   1 root     bin         12340 Jul 24 22:20 shutdown*
-rwxr-xr-x   1 root     root         5029 Jul 24 22:20 swapoff*
-rwxr-xr-x   1 root     bin          5029 Jul 24 22:20 swapon*
-rwxr-xr-x   1 root     root        20592 Jul 27 18:18 telinit*
-rwxr-xr-x   1 root     root         7077 Jul 24 22:20 update*

/tmp:
total 0

/usr:
total 2
drwxr-xr-x   2 root     root         1024 Jul 29 21:00 adm/
drwxr-xr-x   2 root     root         1024 Jul 29 21:16 lib/

/usr/adm:
total 0

/usr/lib:
total 0

/var:
total 1
***  Muitos problemas ocorreram antes da inclusão do  /etc/rc.S
***  para inicializar o /var/run/utmp, mas talvez  isso não seja necessário
*** em outros sistemas
drwxr-xr-x   2 root     root         1024 Jul 28 19:52 run/

/var/run:
total 0
```

C.14 Listas de exemplo do conteúdo do disco de utilitários

```
total 579
-rwxr-xr-x   1 root     root        42333 Jul 28 19:05 cpio*
-rwxr-xr-x   1 root     root       103560 Jul 29 21:31 elvis*
-rwxr-xr-x   1 root     root        56401 Jul 28 19:06 find*
-rwxr-xr-x   1 root     root        29536 Jul 28 19:04 fdisk*
-rw-r--r--   1 root     root       128254 Jul 28 19:03 ftape.o
-rwxr-xr-x   1 root     root        17564 Jul 25 03:21 ftmt*
-rwxr-xr-x   1 root     root        64161 Jul 29 20:47 grep*
-rwxr-xr-x   1 root     root        45309 Jul 29 20:48 gzip*
-rwxr-xr-x   1 root     root        23560 Jul 28 19:04 insmod*
-rwxr-xr-x   1 root     root          118 Jul 28 19:04 lsmod*
lrwxrwxrwx   1 root     root            5 Jul 28 19:04 mt -> mt-st*
-rwxr-xr-x   1 root     root         9573 Jul 28 19:03 mt-st*
lrwxrwxrwx   1 root     root            6 Jul 28 19:05 rmmod -> insmod*
-rwxr-xr-x   1 root     root       104085 Jul 28 19:05 tar*
lrwxrwxrwx   1 root     root            5 Jul 29 21:35 vi -> elvis*
```

Apêndice D

Como Fazer - Dicas Linux

D.1 Introdução

Bem-vindo ao **Como Fazer - Dicas Linux**, uma lista de dicas e otimizações que tornam o Linux mais agradável. Tudo o que temos até aqui são idéias minhas e dicas do antigo Como Fazer-Dicas (Porque jogá-las fora, certo?). Então caso você queira agregar mais alguns avisos e dicas, fique à vontade e escreva para mim. Assim podemos incluí-los na próxima edição do Como Fazer-Dicas.

Paul Anderson *Mantedor - Como Fazer - Dicas Linux*

`panderso@ebtech.net`

D.2 Dicas Curtas

D.2.1 Dica Útil do Syslog *Paul Anderson, Mantenedor - Como Fazer - Dicas Linux*

Edite o /etc/syslog.conf, e coloque a seguinte linha:

```
# Copia tudo em  tty8
*.*                                     /dev/tty8
```

Um detalhe: *LEMBRE-SE DE USAR TAB !* syslog não gosta de espaços...

D.2.2 Programa de visualização dos Como Fazer. *Didier Juges,* `dj@destin.nfds.net`

De um iniciante para outro, segue aqui um programa curto que facilita a visualização e pesquisa em documentos Como Fazer - HOWTO. Meus documentos estão em /usr/doc/faq/howto/ e estão compactadas com gzip. O nome dos arquivos são XXX-HOWTO.gz, sendo XXX o assunto. Eu criei o seguinte programa, chamado "howto" no diretório /usr/local/sbin:

```
#!/bin/sh
```

```
if [ "$1" = "" ]; then
    ls /usr/doc/faq/howto | less
else
    gunzip -c /usr/doc/faq/howto/$1-HOWTO.gz | less
fi
```

Quando chamado sem argumentos, mostra o conteúdo do diretório que contém os HOWTOs disponíveis. Então ao se informar a primeira parte do nome do arquivo (antes do hífen) como argumento, ele descompacta (mantendo o original intocado) e mostra o documento.

Por exemplo, para ver o documento Serial-HOWTO.gz , deve-se informar:

$ howto Serial

D.2.3 Há espaço livre disponível??? *Hans Zoebelein,* zocki@goldfish.cube.net

Segue aqui um pequeno programa que checa de tempos em tempos se há espaço em disco suficiente em qualquer dispositivo que esteja montado (discos, cdrom, disquetes,...).

Caso o espaço acabe, uma mensagem é apresentada a cada X segundos na tela e é enviado um email por dispositivo que esteja sem espaço.

```
#!/bin/sh
#
# $Id: check_hdspace,v 1.18 1996/12/11 22:33:29 root Exp root $
#
#
# Desde que erros misteriosos ocorreram durante a compilação
# quando arquivos encheram o diretório tmp
# eu escrevi este programa para ser avisado de que os discos estavam cheios.
#
# Caso este utilitário evite que o seu servidor exploda mande uma
# mensagem de agradecimento para zocki@goldfish.cube.net.
#
# Caso você realmente saiba como lidar com o sed, por favor desculpe-me!
#
# Atire e esqueça: ponha 'check_hdspace &' no rc.local.
# Cheque o espaço livre nos dispositivos a cada $SLEEPTIME segundos.
# Pode-se verificar inclusive disquetes e fitas.
# Caso o espaço livre esteja abaixo de $MINFREE (kb), será apresentada
# uma mensagem e enviado um email por cada dispositivo em $MAIL_TO_ME.
# Caso haja mais espaço livre do que o limite definido, um email pode
# também ser enviado.
#
# TODO: Diferentes $MINFREE para cada dispositivo.
# Liberar diretórios /*tmp de coisas antigas e inúteis caso não haja mais espaço # livre.


DEVICES='/dev/sda2 /dev/sda8 /dev/sda9'       # device; informe seus discos aqui
MINFREE=20480                                 # kb; abaixo daqui, avise-me
SLEEPTIME=10                                  # sec; tempo entre as checagens
MAIL_TO_ME='root@localhost'                   # fool; quem deve ser avisado
```

```
# ------ a partir daqui nenhuma alteração será necessária (eu espero :) ) ------

MINMB=0
ISFREE=0
MAILED=""
let MINMB=$MINFREE/1024         # sim, nós somos rígidos :)

while [ 1 ]; do
 DF="`/bin/df`"
  for DEVICE in $DEVICES ; do
    ISFREE=`echo $DF | sed s#.\*$DEVICE" "\*[0-9]\*""\*[0-9]\*" "\*## | sed s#" ".\*##`

            if [ $ISFREE -le $MINFREE ] ; then
                let ISMB=$ISFREE/1024
                echo  "WARNING: $DEVICE only $ISMB mb free." >&2
                #echo "more stuff here" >&2
                echo -e "\a\a\a\a"

            if [ -z  "`echo $MAILED | grep -w $DEVICE`" ] ; then
             echo "WARNING: $DEVICE only $ISMB mb free. (Trigger is set to $MINMB mb)" \
             | mail -s "WARNING: $DEVICE only $ISMB mb free!" $MAIL_TO_ME
             MAILEDH="$MAILED $DEVICE"
             MAILED=$MAILEDH
             # put further action here like cleaning
             # up */tmp dirs...
            fi
            elif [ -n  "`echo $MAILED | grep -w $DEVICE`" ] ; then
             # Remove mailed marker if enough disk space
             # again. So we are ready for new mailing action.
             MAILEDH="`echo $MAILED  | sed s#$DEVICE##`"
             MAILED=$MAILEDH
            fi

        done
        sleep $SLEEPTIME
done
```

D.2.4 Utilitário para limpar os arquivos de históricos *Paul Anderson, Mantenedor - Como Fazer - Dicas Linux >*

Caso você seja como eu, deve manter uma lista com centenas de inscritos, mais de cem mensagens por dia chegando pelo UUCP. Bem, o que fazer para lidar com históricos enormes? Instalar chklogs. Chklogs foi escrito por Emilio Grimaldo, `grimaldo@panama.iaehv.nl`, e a versão atual é a 1.8 disponível em ftp.iaehv.nl:/pub/users/grimaldo/chklogs-1.8.tar.gz. Ele é auto explicativo na instalação (obviamente você verificará o conteúdo do subdiretório info). Uma vez instalado, adicione uma entrada ao crontab como esta:

```
# Executar chklogs as 3:00AM diariamente.
00 21 * * *       /usr/local/sbin/chklogs -m
```

Enquanto você estiver lidando isso, avise ao autor quão bom seu software é. :)

D.2.5 Programa útil para limpar arquivos core. *Otto Hammersmith,* ohammers@cu-online.com

Crie um arquivo chamado rmcores (o autor chama isso de trata-cores) com o seguinte conteúdo:

```
#!/bin/sh
USAGE="$0 <directory> <message-file>"

if [ $# != 2 ] ; then
        echo $USAGE
        exit
fi

 echo Apagando...
find $1 -name core -atime 7 -print -type f -exec rm {} \;

echo email
for name in `find $1 -name core -exec ls -l {} \; | cut -c16-24`
do
        echo $name
        cat $2 | mail $name
done
```

E adicione uma tarefa ao cron para executar este utilitário periodicamente.

D.2.6 Movendo diretórios entre sistemas de arquivos. *Alan Cox,* A.Cox@swansea.ac.uk

Uma maneira rápida de mover uma árvore de diretórios e arquivos de um disco para outro.

```
(cd /origem/diretório && tar cf - . ) | (cd /destino/diretório && tar xvfp -)
```

[Mude de cd /origem/diretório; tar....etc... para prevenir possíveis danos ao diretório em caso de problemas. Agradecimentos a Jim Dennis, jim@starshine.org, por avisar-nos.]

D.2.7 Descobrindo os maiores diretórios. *Mick Ghazey,* mick@lowdown.com

Quer saber quais são os maiores diretórios em seu computador? Veja como descobrir.

```
du -S | sort -n
```

D.2.8 A Gazeta Linux

Congratulações a John Fisk, criador do Linux Gazette. Esta é uma excelente revista eletrônica, é **GRÁTIS!!!** Agora o que mais você poderia perguntar? Visite A Gazeta Linux em:

```
http://www.linuxgazette.com
```

Por oportuno, é bom saber que LG tem periodicidade mensal, e (2) John Fisk não mais a mantém e sim o pessoal da SSC.

D.2.9 Ponteiro para uma atualização do GNU Make 3.70 para mudar o comportamento do VPATH. *Ted Stern,* `stern@amath.washington.edu`

Eu não sei se muita gente tem esse problema, mas há uma funcionalidade da versão 3.70 do GNU make que eu não gosto. É que o VPATH age estranhamente caso se informe um caminho de nome absoluto. Há uma atualização muito robusta que conserta isso em Paul D. Smith <`psmith@wellfleet.com`>. Ele também envia mensagens e documentação após cada revisão do GNU make em "gnu.utils.bug". Geralmente, eu aplico as atualizações e compilo gmake em todos os sistemas nos quais trabalho.

D.2.10 Como evitar que o meu sistema faça a checagem de integridade a cada inicialização? *Dale Lutz,* `dal@wimsey.com`

Q: Como evitar que e2fsck cheque o sistema de arquivos de meu disco a cada vez que o sistema seja inicializado.

A: Ao reconstruir o kernel, o sistema de arquivos é marcado como ´sujo´ e então o disco será checado a cada inicialização do sistema. Para corrigir isso execute: rdev -R /zImage 1

Isso avisa ao kernel de que os sistemas de arquivos não apresentam nenhum problema.

Nota: caso se esteja usando LILO, então adicione `read-only` *à configuração do Linux no arquivo de configuração do LILO (normalmente /etc/lilo.conf)*

D.2.11 Como evitar a checagem dos sistemas de arquivos, causados por dispositivos ocupados durante a inicialização do sistema. *Jon Tombs,* `jon@gtex02.us.es`

Caso você tenha erros de dispositivo ocupado ao desligar o sistema e que deixam os sistemas de arquivos com indicação para verificação de integridade em tempo de inicialização, segue uma correção simples:

Em `/etc/rc.d/init.d/halt` ou `/etc/rc.d/rc.0`, adicione a linha

```
mount -o remount,ro /mount.dir
```

para todos os sistemas de arquivos montados, exceto /, antes de chamar o comando umount -a. Isto significa que, se por alguma razão, o desligamento falhar ao finalizar todos os processos e desmontar os discos, eles estarão limpos para reinicialização. Isso economiza bastante tempo durante a carga do sistema.

D.2.12 Como encontrar os maiores arquivos de um disco rígido.

Simon Amor, `simon@foobar.co.uk`

```
ls -l | sort +4n
```

Ou, para aqueles que realmente necessitam de espaço, segue uma alternativa que demora um pouco mas funciona perfeitamente:

```
cd /
ls -lR | sort +4n
```

D.2.13 Como imprimir páginas com margem para arquivamento. *Mike Dickey*, mdickey@thorplus.lib.purdue.edu

```
#!/bin/sh
# /usr/local/bin/print
# um formatador simples, que permite uma margem
# para encadernações
cat $1 | pr -t -o 5 -w 85 | lpr
```

D.2.14 Um meio de pesquisar em árvores de arquivos por uma expressão regular específica.*Raul Deluth Miller*, rockwell@nova.umd.edu

Eu chamo este programa ´forall´. Pode-se usar da seguinte forma:

```
forall /usr/include grep -i ioctl
forall /usr/man grep ioctl
```

Aqui está forall:

```
#!/bin/sh
if [ 1 = `expr 2 \> $#` ]
then
        echo Uso: $0 dir cmd [opcargs]
        exit 1
fi
dir=$1
shift
find $dir -type f -print | xargs "$@"
```

D.2.15 Um programa para limpeza após programas que criam arquivos de cópias de segurança e salvamento automático. *Barry Tolnas*, tolnas@nestor.engr.utk.edu

Segue uma dica simples em duas linhas com recursividade em uma hierarquia de diretórios removendo arquivos emacs de salvamento automático (#) e cópias de segurança (~), arquivos .o e arquivos .log do Tex. Ele ainda comprime arquivos .tex e arquivos README. Eu chamo isso de encolhimento no meu sistema.

```
#!/bin/sh
#ENCOLHER remove arquivos desnecessários e comprime arquivos .tex e README
#De Barry tolnas, tolnas@sun1.engr.utk.edu
#
echo encolhendo $PWD
find  $PWD \( -name \*~ -or -name \*.o -or -name \*.log -or -name \*\#\) -exec
rm -f {} \;
find $PWD \( -name \*.tex -or -name \*README\* -or -name \*readme\* \) -exec gzip -9 {} \;
```

D.2.16 Como encontrar os processos que estão utilizando mais memória. *Simon Amor*, simon@foobar.co.uk

```
ps -aux | sort +4n
```

-OU-

```
ps -aux | sort +5n
```

D.2.17 Configurando o vi para Programação C *Paul Anderson,* Mantenedor - Como Fazer - Dicas Linux

Eu programo bastante em C em meu tempo livre, e eu usei algum tempo para melhorar o vi para ser mais amigável com o C. Segue o meu arquivo .exrc:

```
set autoindent
set shiftwidth=4
set backspace=2
set ruler
```

Como isso funciona? Autoindent faz com que o vi automaticamente indente cada linha, saltando para direita a distância de ˆT em 4 espaços, backspace configura o valor da tecla de retorno e ruler mostra a régua com os números. Lembre-se que para ir a uma linha especifica, digamos número 20, use:

```
vi +20 myfile.c
```

D.2.18 Usando ctags para facilitar a programação.

Muitos programadores já têm ctags em seus computadores, mas não as usam. Podem ser muito úteis em diversas ocasiões. Suponha que se tenha uma função, em um dos diversos arquivos de fontes em um diretório, para um programa que está em desenvolvimento, e se deseja editar a função para sua atualização. Chamaremos esta função de foo(). Porém não lembramos onde está o fonte. É aqui que ctags começa a ser muito útil. Ao executar ctagas ele produz um arquivo chamado tags no diretório atual, o qual é uma lista de todas as funções, em quais arquivos eles estão presentes e onde elas estão naqueles arquivos. O arquivo de resultado se parece com algo como:

```
ActiveIconManager  iconmgr.c    /^void ActiveIconManager(active)$/
AddDefaultBindings add_window.c /^AddDefaultBindings ()$/
AddEndResize        resize.c     /^AddEndResize(tmp_win)$/
AddFuncButton       menus.c      /^Bool AddFuncButton (num, cont, mods, func, menu, item)$/
AddFuncKey          menus.c      /^Bool AddFuncKey (name, cont, mods, func, menu, win_name, action)$/
AddIconManager     iconmgr.c    /^WList *AddIconManager(tmp_win)$/
AddIconRegion      icons.c      /^AddIconRegion(geom, grav1, grav2, stepx, stepy)$/
AddStartResize     resize.c     /^AddStartResize(tmp_win, x, y, w, h)$/
AddToClientsList   workmgr.c    /^void AddToClientsList (workspace, client)$/
AddToList          list.c       /^AddToList(list_head, name, ptr)$/
```

Para editar, digamos AddEndResize() no vim, execute:

```
vim -t AddEndResize
```

Isto irá trazer o arquivo apropriado ao editor, com o cursor localizado no início da função.

D.2.19 O que faz com que o sendmail demore 5 minutos na inicialização do Red Hat? *Paul Anderson,* paul@geeky1.ebtech.net

Este é um problema muito comum, quase no ponto de tornar-se um FAQ. A Red Hat já deve ter corrigido este problema em sua distribuição, mas isso pode ser corrigido por você mesmo. Ao se verificar o conteúdo do arquivo /etc/hosts file, pode-se encontrar algo como:

```
127.0.0.1            localhost        suamáquina
```

Quando sendmail é iniciado, ele procura pelo nome da máquina (neste exemplo suamáquina). Após, ele verifica que o IP de suamáquina é 127.0.0.1, porém ele não gosta desta definição e procura novamente, continuando neste processo por um longo período até desistir e finalizar. Corrigir este problema é muito simples, bastando editar o arquivo /etc/hosts e alterá-lo para algo como:

```
127.0.0.1            localhost
10.56.142.1          yourbox
```

D.2.20 Como configurar o Red Hat para utilizar o comando ls em cores? *Paul Anderson,* paul@geeky1.ebtech.net

A distribuição da Red Hat vem com o comando ls em cores, porém ele não é configurado automaticamente na instalação. Vejam como fazê-lo.

Primeiro, digite eval ‘DIRCOLORS‘

Após, alias ls=’ls –color=auto’

E deve-se incluir o ’alias.....’ no arquivo /etc/bashrc

D.2.21 Como descobrir qual biblioteca em /usr/lib contém determinada função? *Pawel Veselow,* vps@unicorn.niimm.spb.su

O que fazer quando se está compilando um programa e não se sabe a biblioteca que está faltando? Todas as informações geradas pelo gcc são com nomes de funções. Veja aqui como um simples comando pode encontrar o que você está procurando:

```
for i in *; do echo $i:;nm $i|grep nome_função 2>/dev/null;done
```

Onde nome_função é o nome da função que se está procurando.

D.2.22 Eu compilei um pequeno programa de testes em C, mas ao executá-lo não aparece nenhuma informação de saída!

Provavelmente o programa foi compilado em um binário chamado test. Porém Linux tem um programa de mesmo nome, o qual testa se determinada condição é verdadeira, e nunca produz resultados na tela. Para testar seu programa, estando no diretório onde ele está localizado, digite: ./test

D.3 Dicas Detalhadas

D.3.1 Compartilhando partições de troca entre o Linux e o Windows. *Tony Acero,* ace3@midway.uchicago.edu

1. Formatar a partição como uma partição DOS, e criar o arquivos de troca Windows nela, mas sem rodar o Windows ainda. (deve-se manter o arquivo de troca completamente vazio por enquanto, assim ele pode ser compactado com eficiência).
2. Inicie o Linux e salve a partição em um arquivo. Por exemplo, se a partição é /dev/hda8:

```
dd if=/dev/hda8 of=/etc/dosswap
```

3. Compacte o arquivo de troca; uma vez que tudo que ele contém são zeros, a compressão terá um excelente resultado

```
gzip -9 /etc/dosswap
```

4. Adicione a seguinte linha ao arquivo /etc/rc para preparar e instalar a área de troca sob o Linux: *XXXXX é o número de blocos da partição de troca*

```
mkswap /dev/hda8 XXXXX
swapon -av
```

   Esteja seguro de adicionar uma linha de definição da área de troca no arquivo /etc/fstab

5. Caso o pacote init/reboot suporte /etc/brc ou /sbin/brc adicione o seguinte ao arquivo /etc/brc, ou faça manualmente ao iniciar o DOS | OS/2 para converter de volta a partição em uma versão do DOS/Windows:

```
swapoff -av
zcat /etc/dosswap.gz | dd of=/dev/hda8 bs=1k count=100
```

Note-se que este comando grava somente os primeiros 100 blocos da partição. Eu encontrei empiricamente este número como suficiente.

>> Quais os prós e contras de se utilizar essa sistemática?

Prós: pode-se economizar uma quantidade substancial de disco.

Contras: caso o passo 5 não seja realizado de forma automática, deve-se lembrar de executá-lo manualmente, e isso pode atrasar a inicialização em nanosegundos. :-)

D.3.2 Recuperação de arquivos apagados. *Michael Hamilton,* `michael@actrix.gen.nz`

Segue aqui uma dica, usada por mim algumas vezes.

Recuperação de arquivos textos pessoais.

Caso acidentalmente um arquivo texto tenha sido apagado, por exemplo, algum email ou os resultados da última noite de programação, há uma chance de recuperá-los. Caso o arquivo tenha sido gravado em disco, isto é se não foi gravado há mais de 30 segundos, seu conteúdo pode ainda estar na partição em disco.

Pode-se usar o comando grep para procurar o conteúdo do arquivo diretamente na partição em disco.

Por exemplo, recentemente, eu acidentalmente apaguei um email. Então imediatamente cessei qualquer atividade que pudesse modificar a partição: Neste caso, bastou suspender qualquer atividade de salvamento de arquivos ou compilações, etc... Em algumas situações eu na verdade desliguei o sistema e o inicializei em modo monousuário, sem montar os sistemas de arquivos.

Após eu utilizei o comando egrep na partição do disco: no meu caso uma mensagem eletrônica em /usr/local/home/walter/. Inicialmente pode-se ver que o diretório está localizado em /dev/hdb5, utilizando-se o comando df:

```
sputnik3:~ % df
  Filesystem         1024-blocks  Used Available Capacity Mounted on
  /dev/hda3              18621    9759     7901     55%   /
```

```
/dev/hdb3            308852  258443    34458     88%   /usr
/dev/hdb5            466896  407062    35720     92%   /usr/local

sputnik3:~ % su
Password:
[michael@sputnik3 michael]# egrep -50 'ftp.+COL' /dev/hdb5 > /tmp/x
```

Agora deve-se ser extremamente cuidadoso ao lidar com partições de discos, devendo-se verificar cuidadosamente o comando a ser aplicado antes de pressionar [Enter]. Neste caso procuro o email contendo a palavra ftp seguido de algum texto que contenha a palavra COL. A mensagem continha aproximadamente 20 linhas, tendo então usado -50 para recuperar todas as linhas no entorno da mensagem. No passado eu cheguei a usar -3000 para ter certeza de recuperar todas as linhas de algum código fonte. A saída do comando egrep foi direcionada para uma partição diferente do disco - prevenindo-se assim alguma gravação sobre a área que contém as informações desejadas.

Após, utilizei o comando strings para auxiliar na avaliação da saída anterior:

```
strings /tmp/x | less
```

Seguro o suficiente para garantir que o email estava lá.

Este método não é totalmente seguro, pois parte ou toda a área utilizada pelo arquivo anteriormente já pode ter sido utilizada.

Esta dica é provavelmente útil somente em sistemas monousuários. Em sistemas multiusuários com alto nível de atividade em disco, o espaço liberado pode já ter sido utilizado.

Em meu sistema doméstico esta dica pode ser utilizada com sucesso em pelo menos três ocasiões nos últimos anos - principalmente quando acidentalmente eu apaguei o resultado de dias de trabalho. E caso a tarefa na qual eu esteja trabalhando chegue a um ponto onde eu sinta um progresso significativo, é aconselhável copiá-lo em um ou mais disquetes, não sendo necessário usar esta dica com muita freqüência.

D.3.3 Como utilizar um indicador imutável. *Jim Dennis,* `jadestar@rahul.net`

Uso de Um Sistema Imutável

Imediatamente após a instalação e configuração de um sistema, vá aos diretórios /bin, /sbin/, /usr/bin, /usr/sbin e /usr/lib (e alguns outros prováveis candidatos) e utilize livremente o comando 'chattr +i'. Adicione também os arquivos do kernel no raiz. Após execute 'mkdir /etc/.dist/' e copie todo o conteúdo de /etc/ nele (eu normalmente faço isto em dois passos usando /tmp/etcdist.tar para evitar a recursividade) no diretório. (Opcionalmente pode-se simplesmente criar /etc/.dist.tar.gz) - e indicá-lo como imutável.

A razão para tudo isso é limitar eventuais danos que podem ser feitos pelo usuário que acessa o sistema como superusuário root. Assim não se apagará arquivos importantes com um redirecionamento ou com um comando acidental 'rm -fr' (apesar destes comandos ainda poderem causar um grande estrago no sistema - ao menos as bibliotecas e executáveis estarão a salvo).

Isso torna o sistema ainda mais seguro pois evita serviços de busca de falhas de segurança, tornando mais difícil ou praticamente impossível o seu uso (uma vez que muitos deles fundamentam-se na regravação de arquivos que executem ações com o programa SUID e tenham acesso a ambientes de trabalho).

O único incoveniente deste procedimento aparece ao se executar 'make install' em diversos binários do sistema. Caso se esqueça de executar o comando chattr -i nos arquivos a serem regravados e nos diretórios onde eles estão localizados, o comando make falhará. Basta executar o comando e executar novamente make. Pode-se ainda mover os binários antigos, bibliotecas, e tudo o mais para um diretório .old/ ou renomeá-los ou ainda aplicar o comando tar, etc...

D.3.4 Sugestão de onde colocar novos programas. *Jim Dennis,* `jadestar@rahul.net`

Todos os arquivos e comandos novos podem estar em /usr/local ou /usr/local/'nome_da_máquina'. Caso a sua distribuição deixe vazio o diretório /usr/local , então simplesmente crie os diretórios /usr/local/src, /usr/local/bin, etc... e utilize-os. Caso a sua distribuição coloque arquivos em /usr/local, então pode-se criar um diretório com o comando 'mkdir /usr/local/'hostname'' e atribuir a permissão de grupo +w para ele (eu ainda faço uso de SUID e SGID para garantir que cada membro do grupo w possa somente lidar com arquivos que a eles pertencem, e que assim todos os arquivos criados pertencerão ao grupo w).

Após, adquira o hábito de sempre colocar novos pacotes em /usr/local/src/.from/$NOME_DO_PACOTE/ (para arquivos .tar ou qualquer outro) e construa-os em /usr/local/src (ou .../$NOME_DA_MÁQUINA/src). Esteja seguro que ele seja instalado sob a hierarquia de diretórios local. Caso ele necessariamente deva ser instalado em /bin ou /usr/bin ou algum outro local, crie uma ligação simbólica da hierarquia local com cada elemento que deva ser colocado em outro local.

A razão para isto - mesmo que dê um pouco mais de trabalho - reside na capacidade de isolar o que deve ser copiado e restaurado ou reinstalado no caso de uma reinstalação completa a partir de uma mídia da distribuição (normalmente um CD). Usando um diretório /usr/local/.from pode-se ainda manter-se um histórico informal da origem dos fontes – o que auxilia na hora de buscar atualizações - o que pode ser crítico ao se monitorar a lista de questões de segurança.

Por exemplo, todos os sistemas que eu configurei no trabalho (quando eu era encarregado da administração do sistema) e que foram administrados por muitos contratados e outras pessoas, e tiveram um grande número de atualizações, mantiveram uma idéia precisa dos elementos que foram adicionados ao sistema após a instalação e configuração iniciais.

D.3.5 Convertendo os nomes de todos os arquivos de um diretório para letras minúsculas. *Justin Dossey,* `dossey@ou.edu`

Eu reparei em uma dificuldade desnecessária nos procedimentos recomendados na seção 2c das dicas, edição 12 da LG. Neste caso estou enviando uma alteração para facilitar o processo:

```
#!/bin/sh
       # lowerit
         # converte os nomes de todos os arquivos do diretório atual para letras
       # minúsculas. Funciona somente em arquivo - não muda nomes de diretórios
       # e solicitará confirmação antes de regravar um arquivo já existente
         for x in `ls`
           do
           if [ ! -f $x ]; then
             continue
             fi
           lc=`echo $x  | tr '[A-Z]' '[a-z]'`
           if [ $lc != $x ]; then
             mv -i $x $lc
```

```
        fi
        done
```

Bem, é um programa longo. Ao invés de usar tudo isso pode-se usar o seguinte comando:

```
for i in * ; do [ -f $i ] && mv -i $i ‘echo $i | tr ’[A-Z]’ ’[a-z]’‘;
done;
```

D.3.6 Encerrando os processos de um usuário. *Justin Dossey,* `dossey@ou.edu`

Na próxima dica pode-se encerrar todo os processos de um determinado usuário com o seguinte comando.

```
kill -9 ‘ps -aux |grep ^<nome_usuário> |tr -s " " |cut -d " " -f2‘
```

Por exemplo, caso o usuário se chame paloma

```
kill -9 ‘ps -aux |grep ^paloma |tr -s " " |cut -d " " -f2‘
```

D.3.7 Senha de superusuário perdida.

Agora com cuidado, vamos falar de senhas do superusuário que foram esquecidas ou perdidas.

A solução dada na Linux Gazette é a mais universal, mas não a mais simples. Tanto com o LILO como com o loadlin, pode-se incluir o parâmetro "single" para se iniciar diretamente no ambiente de trabalho padrão, sem a necessidade de informar senhas. A partir daqui, pode-se mudar ou remover quaisquer senhas, antes de digitar "init 3" para iniciar o sistema em modo multiusuário. Justin Dossey

D.3.8 Como atualizar o Sendmail *Paul Anderson,* `paul@geeky1.ebtech.net`

Começaremos do princípio, dos fontes do programa. Inicialmente deve-se obter os fontes do sendmail. Eu estou utilizando a versão 8.9.0, a qual é, no momento em que escrevo este artigo, o que há de mais recente. Eu a obtive em ftp.sendmail.org:/pub/sendmail/sendmail.8.9.0.tar.gz

Tem o tamanho de aproximadamente 1 Mb e considerando que se esteja executando a versão 8.7.6, eu creio que o esforço vale a pena. Caso funcione, certamente você saberá prontamente, de outra forma não há como receber as novas versões dos HOWTOs :-)

Agora que já se tem os fontes, deve-se descomprimí-los. Pode-se criar um diretório chamado `sendmail-8.9.0` dentro do diretório atual. Vá para o diretório (com o comando cd), e leia os arquivos README e RELEASE_NOTES (e esteja ciente das atualizações que foram feitas). Agora pode-se ir para o diretório src. Aqui é onde a maior parte do trabalho será executada.

Nota: Sendmail é um pequeno, poderoso e bem escrito programa. O binário é compilado em menos de 5 minutos em uma máquina 5x86 133 com 32 Mb de RAM! A compilação completa e a instalação (configuração) levam menos de 15 minutos.

Eu normalmente não executo o BIND em meu sistema, então encontrei as seguintes linhas:

```
# ifndef NAMED_BIND
#  define NAMED_BIND    1       /* usando Servidor de Domínios Berkeley */
# endif
```

e mudando 1 para 0:

```
# ifndef NAMED_BIND
#  define NAMED_BIND    0       /* usando Servidor de Domínios Berkeley */
# endif
```

No Debian 1.3.1, db.h é por padrão instalado em /usr/include/db, ao invés de /usr/include, onde sendmail espera encontrá-lo. Vá para os diretórios src, mailstats, makemap, praliases, rmail e smrsh e execute o seguinte comando:

```
./Build -I/usr/include/db
```

Uma vez feito isso, execute cd.. e digite make install. Voilá! Sendmail versão 8.9.0 deve estar instalado. Isso claro, assumindo que a configuração original já foi realizada. Para que tudo funcione corretamente, uma vez que eu hospedo listas de mensagens usando majordomo, é necessário adicionar o seguinte no início de /etc/sendmail.cf:

```
O DontBlameSendmail=forwardfileinunsafedirpath, forwardfileinunsafedirpathsafe
```

Sendmail 8.9.0 é desagradável sobre permissões de arquivos e diretórios, e irá reclamar sobre arquivos e diretórios em aliases ou .forward que tenham permissões de escrita para todos os usuários ou para o grupo. Uma vez que não é uma boa idéia desabilitar estes avisos, como sou o único que utiliza o sistema, sinto-me à vontade para fazê-lo. YMMV.

D.3.9 Algumas dicas para novos administradores de sistemas. *Jim Dennis,* jadestar@rahul.net

Criar e manter um arquivo /LEIAME.‘nome_máquina‘ e/ou um /etc/LEIAME.‘nome_máquina‘ *[Ou possivelmente /usr/local/etc/LEIAME.‘nome_ máquina‘ -Maint.]*

Desde o primeiro dia da administração do sistema, tome notas neste arquivo como um histórico on-line. Pode-se incluir a linha "vi /LEIAME.$(nome_máquina)" no arquivo /bash_logout do superusuário. Uma outra forma de fazer isso é criar um programa su ou sudo que faça mais ou menos o seguinte:

```
function exit \
        { unset exit; exit; \
          cat ~/tmp/session.$(date +%y%m%d) \
          >> /LEIAME.$(nome_máquina) && \
          vi /LEIAME.$(nome_máquina)
          }
script -a ~/tmp/session.$(date +%y%m%d)
/bin/su.org -
```

(use o comando typescript para criar um histórico da sessão e criar uma função para automatizar a inclusão e atualização do histórico).

Eu admito que não utilizei essa política de automação, confiando sempre na minha disciplina. De qualquer forma a idéia está formatada.

Minha última sugestão é manter o caminho para o superusuário com o seguinte formato 'PATH= /bin'.

Somente isso e nada mais. Assim tudo o que o superusuário fizer pode ser provido por uma ligação simbólica com /bin ou por um nome alternativo ou por uma função de ambiente de trabalho, ou ainda pela digitação explícita do caminho.

Isso torna qualquer usuário acessando o sistema como superusuário ciente de que binários estão sendo utilizados. O administrador inteligente de um sistema multiusuário irá periodicamente observar os arquivos /bin e /.*history procurando padrões ou mensagens freqüentes.

O administrador realmente interessado irá buscar seqüências que podem ser automatizadas, colocar checagens em pontos necessários, e verificar quais privilégios do "superusuário" podem ser revistos (uso de editores, MTA e outros programas interativos com funcionalidades elaboradas que podem agregar informações em arquivos de dados - como o famoso vi ./.exrc e emacs ./.emacs e mesmo o $EXINIT e as macros header/footer). Naturalmente estes comandos podem ser executados da seguinte forma:

```
cp $data $algum_diretório_usuário/tmp
su -c $comando_origem $mudanças
cp $algum_diretório_usuário/tmp $data
```

(...onde os dados variam de comando para comando).

Muitas destas precauções podem ser consideradas exageradas em estações de trabalho domésticas ou de usuários individuais, mas são altamente recomendadas como política de administração de sistemas multiusuários, particularmente um sistema exposto ao público.

D.3.10 Como configurar o seletor de servidor do xdm. *Arrigo Triulzi,* `a.triulzi@ic.ac.uk`

1. Edite o arquivo que inicializa o xdm, (possivelmente /etc/rc/rc.6 ou /etc/rc.local) e que contenha as seguintes linhas na seção de início do xdm.

   ```
   /usr/bin/X11/xdm
   exec /usr/bin/X11/X -indirect servidor
   ```

2. Edite o arquivo /usr/lib/X11/xdm/Xservers, comentando a linha que inicia o servidor na máquina local (por exemplo starting 0:)
3. Ao reinicializar o computador, você estará em casa e além.

Eu inclui isso quando eu estava desesperadamente tentando configurar minha própria subrede a partir de uma máquina e levei mais de uma semana para corrigir todos os problemas.

Adicional: com o velho SLS (1.1.1) por alguma razão pode-se utilizar o parâmetro -nodaemon após a linha do xdm - porém isso **NÃO** funciona para versões mais recentes.

Apêndice E

Comandos e Programas Relacionados

Quando todas as tentativas falharem siga as instruções

Este apêndice contém diversas páginas de manual relacionadas no decorrer deste livro, bem como comandos úteis para o administrador de sistemas.

E.1 bdflush

NOME

bdflush - servidor responsável pela gravação em disco de dados contidos em buffers.

SINOPSE

bdflush [opc]

DESCRIÇÃO

O **bdflush** é usado para iniciar o servidor do kernel que descarrega o conteúdo de buffers com dados no disco rígido. O trabalho atual de descarga é uma função do kernel, sendo que o bdflush na verdade inicia um novo processo que executa esta função.

O **bdflush** na verdade aciona um segundo servidor, e este age como mais um processo tradicional de atualização de dados, exceto pelo fato de que os buffers não são considerados prontos para gravação até que o tempo determinado tenha transcorrido. O relógio inicia a marcação de tempo quando o bit de conteúdo é inicializado, e o conteúdo do buffer não será transferido até que um determinado intervalo de tempo transcorra. O intervalo é distinto para buffers de dados e metadados (tais como diretórios, mapas de bits, blocos de indireção, etc...), e as configurações atuais são mostradas quando se executa o programa com argumentos através da linha de comando. O padrão é de 30 segundos para buffers de dados e 5 segundos para metadados.

Os dois servidores são normalmente iniciados com um comando contido no /etc/rc:

/sbin/update

Note que é necessário ter-se dois servidores sendo executados, porque cada um tem um propósito diferente, e que devem estar sendo executados antes que qualquer tarefa maior de E/S seja executada. Na verdade o update deve ser executado antes que qualquer sistema de arquivos seja montado para leitura ou gravação e antes da execução do programa fsck nestes sistemas de arquivos.

Quando o bdflush é chamado por um usuário sem privilégios de superusuário, ele chama as funções flush() e sync() e após é finalizado. Não é necessário ter-se 20 servidores update sendo executados simultaneamente.

COMANDO-OPÇÕES

-d Mostra os parâmetros do kernel. Com esta opção o programa não é colocado em execução.

-h Apresenta as informações de uso (ajuda).

-s Caso o bdflush retorne ao comportamento padrão, este parâmetro provoca a execução do comando sync com a periodicidade aqui definida. (Em segundos) Padrão:30.

-f Chama a descarga de dados com a freqüência aqui estipulada. (em segundos). Padrão:5

-0 Máxima fração da lista LRU que será examinada na verificação de buffers com conteúdo.

-1 Número máximo de blocos com conteúdo que serão gravados cada vez que o bdflush for ativado.

-2 Número de buffers sem conteúdo que devem ser carregados em uma lista de buffers livres pela função refill_freelist.

-3 Blocos com conteúdo necessários para ativar bdflush em refill_freelist.

-4 Porcentagem do cache a ser pesquisada por área livres.

-5 Tempo de espera para gravação dos dados residentes em buffers.

-6 Tempo de espera para gravação dos metadados residentes em buffers.

-7 Constante da média de tempo de carga de buffers.

-8 Proporção LAV (usada para determinar o limite de eliminação de buffers).

AUTOR

bdflush foi escrito como um pequeno utilitário por Eric Youngdale <ericy@gnu.ai.mit.edu>. O principal objetivo foi incrementar a performance do kernel tornando a descarga de dados dos buffers mais inteligente e adicionando suporte a conjuntos de buffers. Sinta-se à vontade para melhorá-lo. Diversos aprimoramentos foram realizados por Phil Bostley <bostley@cs.colorado.edu> e Daniel Quinlan <quinlan@yggdrasil.com>.

PROBLEMAS

Caso haja algum, possivelmente estarão no código do kernel.

E.2 cfdisk

NOME

cfdisk - manipulador da tabela de partições de discos baseada em curses.

SINOPSE

cfdisk [-avz] [-c *cilindros* **] [-h** *cabeças* **] [-s** *setores-por-trilha* **] [-P** *opt* **] [** *dispositivo* **]**

DESCRIÇÃO

cfdisk é um programa baseado em curses para particionamento de discos rígidos. O *dispositivo* pode ser qualquer um dos seguintes:

/dev/hda [padrão]

/dev/hdb

/dev/sda

/dev/sdb

/dev/sdc

/dev/sdd

O **cfdisk** inicialmente tenta ler a geometria do disco rígido. Caso ele falhe, uma mensagem de erro será apresentada e o **cfdisk** finalizará. Isso somente deverá acontecer ao se particionar uma unidade SCSI em um adaptador sem BIOS. Para corrigir este problema, pode-se definir os *cilindros*, *cabeças* e *setores-por-trilha* na linha de comando. A seguir o **cfdisk** tenta ler a tabela atual de partições do disco rígido. Caso não seja possível descobrir a tabela de arquivos, um erro é apresentado e o programa terminará. Isso pode ser causado por informações incorretas sobre a geometria do disco, e pode ser sobreposto pela linha de comando. Outra forma de lidar com isso é utilizar a opção **-z.** Isso fará com que a tabela de partições em disco seja ignorada.

A tela principal é composta por quatro seções, de cima para baixo: cabeçalho, partições, linha de comando e linha de avisos. O cabeçalho contém o nome do programa e a versão, seguidos do disco rígido e sua geometria. A seção de partições sempre apresenta a tabela de partições atual. A linha de comando é o local onde os comandos são informados. Os comandos disponíveis estão normalmente entre chaves. A linha de avisos está normalmente vazia, exceto quando há alguma informação importante a ser apresentada. A partição corrente é destacada com vídeo reverso (ou uma flecha se a opção **-a** é informada). Todos os comandos específicos de partições aplicam-se à partição atual.

O formato da tabela de partições na seção partições é o seguinte, da esquerda para a direita: Nome, Indicadores, Tipo da Partição, Tipo do Sistema de Arquivos e Tamanho. O nome é o nome do dispositivo da partição. Os indicadores podem ser *Boot*, o qual designa uma partição inicializável ou *NC*, que significa "Não Compatível com DOS ou OS/2". DOS, OS/2 e possivelmente outros sistemas operacionais requerem que o primeiro setor da primeira partição do disco e todas as partições lógicas comecem na segunda cabeça. Isso desperdiça do segundo ao último setor da primeira trilha (o primeiro setor é utilizado pela própria tabela de partições). O **cfdisk** permite recuperar estes setores "perdidos" com o comando maximizar (**m**). *Nota:* O **fdisk(8)** e algumas versões recentes do DOS já criam todas as partições com o número máximo de setores já maximizados. Para mais informações, veja o comando maximizar abaixo. O tipo de partição pode ser

Primária ou *Lógica*. Para espaço não alocado na unidade, o tipo de partição pode ser *Pri/Log*, ou vazia (se o espaço não puder ser utilizado). A seção de tipo do sistema de arquivos mostra o nome do sistema de arquivos usado na partição, caso conhecido. Caso seja desconhecido, então a expressão *Unknown* e os valores do tipo do sistema de arquivos em hexadecimais serão apresentados. Um caso especial ocorre quando há setores da unidade de disco que não podem ser usadas (porque todas as partições primárias estão em uso). Quando isso ocorre, o tipo do sistema de arquivos apresentado é igual a *Unusable* (não pode ser utilizado). O campo tamanho apresenta o tamanho da partição em megabytes (por padrão). Pode ainda apresentar o tamanho em setores e cilindros (veja o comando de mudança de unidades abaixo). Caso asteriscos (*) apareçam após o tamanho, isso significa que a partição não está alinhada nos limites do cilindro.

AVISO DOS 6.x

O comando FORMAT do DOS 6.x procura por alguma informação no primeiro setor da área de dados da partição, e trata esta informação como mais confiável do que a informação contida na tabela de partições. O comando FORMAT do DOS espera que o FDISK do DOS limpe os primeiros 512 bytes da área de dados de uma partição toda vez que uma mudança de tamanho ocorrer. O FORMAT do DOS sempre procurará por informações adicionais mesmo se o indicador /U for informado - consideramos isso como um erro do FORMAT DOS e do FDISK do DOS.

A questão reside no fato de, ao se utilizar o cfdisk ou fdisk para mudar o tamanho de uma partição DOS em uma tabela, deve-se então executar o comando **dd** para zerar os primeiros 512 bytes daquela partição antes de usar o comando FORMAT do DOS para formatá-la. Por exemplo, caso se esteja utilizando o cfdisk para criar uma partição DOS na tabela de partições de /dev/hda1, então (após finalizar o fdisk ou o cfdisk e reinicializar o Linux, sendo que então a tabela será válida) utilizar o comando "dd if=/dev/zero of=/dev/hda1 bs=512 count=1" para zerar os primeiros 512 bytes da partição. **SEJA EXTREMAMENTE CUIDADOSO** ao usar o comando **dd** , uma vez que um pequeno erro de datilografia pode inutilizar os dados no disco rígido.

Para melhores resultados, deve-se procurar utilizar um programa de manipulação específico para o sistema operacional que será utilizado na partição, Por exemplo, deve-se usar o FDISK do DOS para criar partições DOS e o fdisk ou cfdisk do Linux para criar partições Linux.

COMANDOS

Comandos do **cfdisk** podem ser realizados pressionando-se as teclas desejadas (não é necessário pressionar [`Enter`] após o comando). Segue uma lista dos comandos disponíveis:

b indica que a partição atual é inicializável. Isso permite selecionar qual partição primária será inicializável no disco rígido.

d Apaga a partição atual. Este comando converterá a partição atual em espaço livre e a adicionará imediatamente a qualquer espaço livre contíguo à partição. Uma partição já assinalada como espaço livre ou não utilizável não pode ser apagada.

g altera a geometria do disco (cilindros, cabeças ou setores por trilha). **ATENÇÃO:** Esta opção deve ser usada somente por pessoas que saibam o que estão fazendo. Uma opção através da linha de comando está também disponível para alterar a geometria do disco. Na linha de comando pode-se escolher a alteração de: cilindros (**c**), cabeças (**h**), e setores por trilha (**s**). O valor padrão será apresentado na linha de comandos e para aceitá-lo basta simplesmente pressionar [`Enter`], ou pode-se sair sem efetivar as alterações pressionando-se [`ESC`]. Caso se deseje alterar o valor padrão, simplesmente informe os novos valores e pressione [`Enter`]. As alterações dos parâmetros do disco não terão efeito até que se retorne à tela principal (pressionando-se [`Enter`] ou [`ESC`] na linha de comando de alteração de geometria de disco). Caso a mudança de geometria altere o disco para um tamanho maior, os setores adicionais serão incluídos ao final do disco como espaço livre. Caso o disco fique menor, as partições que estejam além do novo último setor serão apagadas e a última partição do disco (ou o espaço livre ao final do disco) terá o seu final no novo último setor.

h Imprime a tela de ajuda.

m Maximiza o uso de disco da partição atual. Este comando irá recuperar espaço não utilizado entre a tabela de partições e o início da partição, porém tornando-a incompatível com DOS, OS/2 e possivelmente com outros sistemas operacionais. O padrão na criação de uma partição é que ela seja compatível com outros sistemas operacionais como DOS, OS/2, etc...

n Criar novas partições a partir do espaço livre disponível. Caso o tipo de partição seja *Primária* ou *Lógica*, uma partição daquele tipo será criada, mas se for do tipo *Pri/Log*, será solicitada a informação do tipo que se deseja criar. Esteja ciente que (1) podem ser criadas somente quatro partições primárias e (2) pode existir somente uma partição estendida, a qual contém todos os dispositivos lógicos, estes devem ser contíguos (sem nenhuma partição primária entre eles). O **cfdisk** solicitará após, a informação de tamanho da partição que se deseja criar. O tamanho padrão é igual ao tamanho total do espaço livre disponível, em megabytes. Pode-se tanto pressionar `Enter` para aceitar o tamanho padrão ou informar um tamanho diferente na linha de comandos. O **cfdisk** aceita tamanhos em diferentes formatos, tais como megabytes (**M**) [padrão], kilobytes (**K**), cilindros (**C**) e setores (**S**) através da informação do número desejado seguido por (**M**, **K**, **C** ou **S**). Caso a partição tenha sido criada com o espaço livre disponível, após a sua criação o sistema retornará para a linha de comandos. De outra forma, a partição poderá ser criada no início ou no fim da área de espaço livre, e o **cfdisk** perguntará onde a partição deverá ser colocada. Após a partição ser criada o **cfdisk** automaticamente ajustará os tipos das outras partições caso todas as partições primárias sejam usadas.

p Lista a tabela de partições na tela ou em um arquivo. Há diversos formatos diferentes que podem ser escolhidos:

r Formato de dados brutos (exatamente o que deveria ser gravado em disco)

s Tabela de partições em ordem de setores

t

Tabela de partições em formato bruto O *formato* bruto de dados listará os setores que podem ser gravados em disco, se um comando **w**(gravar) for selecionado. Primeiro, a tabela da partição primária é listada, seguida das tabelas de partições associadas com cada partição lógica. Os dados são apresentados em formato hexadecimal byte a byte, com 16 bytes por linha.

A *tabela* de partições apresentada em ordem de setor apresentará a tabela de partições ordenada por número do setor. Os campos da esquerda para a direita são: número da partição, tipo de partição, primeiro setor, último setor, deslocamento em relação ao primeiro setor da partição para início da gravação de dados, tamanho da partição, tipo do sistema de arquivos (com valores hexadecimais entre parênteses). Adicionalmente às partições primárias e lógicas, são apresentados os espaços livre e não utilizáveis e a partição estendida é apresentada antes da primeira partição lógica.

Se uma partição não começa ou termina nos limites de um cilindro ou se o tamanho da partição não é divisível pelo tamanho do cilindro, um asterisco (*****) é apresentado após o número do setor não alinhado. Isso normalmente indica que a partição foi criada por outro sistema operacional que/ou não alinha partições nos limites do cilindro ou usa diferentes informações de geometria de disco. Caso você saiba a geometria utilizada por outro sistema operacional, pode-se informá-la com o comando de alteração de geometria. (**g**).

Para a primeira partição do disco e para todas as partições lógicas, se o deslocamento do início da partição não for igual ao número de setores por trilha (isto é, os dados não começam na primeira cabeça), um sinal (#) é apresentado após o deslocamento. Para as partições remanescentes, caso o deslocamento seja diferente de zero, um sinal também será listado após o deslocamento. Isto corresponde ao indicador *NC* na seção partições na tela principal.

A *tabela* de partições em formato bruto listará a tabela de partições em ordem de número da partição. Ela deixará de fora todo o espaço livre ou não utilizável. Os campos, da esquerda para a direita, são: número da partição, indicadores (em hexadecimal), cabeça de início, setor e cilindro, setor de início na partição e o número de setores na partição. A informação na tabela pode ser diretamente convertida em *formato de dados brutos.*

As entradas na tabela de partições têm somente 10 bits disponíveis para representar os cilindros iniciais e finais. Mas, quando o número do setor de início está em um cilindro maior que 1023, o valor máximo da cabeça de início, setor e cilindro são listados. Este método é usado pelo OS/2, e corrige os problemas associados com a regravação da tabela de partições feita pelo fdisk do OS/2. Uma vez que Linux e OS/2 usam contadores absolutos dos setores, os valores de cabeças de início e fim, setores e cilindros não são usados.

q Finalizar o programa. Este procedimento finalizará o programa sem a gravação de qualquer dado em disco.

t Muda o tipo do sistema de arquivos. Por padrão, novas partições são criadas como partições *Linux* "," porém como o **cfdisk** pode criar partições para outros sistemas operacionais, mudar tipos de partições permite que se informe valores hexadecimais dos sistemas de arquivos desejados. Uma lista dos tipos de sistemas de arquivos disponíveis é apresentada. Pode-se informar o tipo do sistema de arquivos na linha de comandos ou aceitar o tipo de sistema de arquivos padrão: [*Linux*].

u Muda as unidades de apresentação do tamanho da partição. Ele rotacionará entre megabytes, setores e cilindros.

W Grava a tabela de partições em disco (deve ser informada em maiúsculas W). Uma vez que isso pode destruir os dados em disco, deve-se confirmar ou negar a gravação informando 'yes' ou 'no'. Caso se informe 'yes', o **cfdisk** gravará a tabela de partições em disco e avisará ao kernel para reler a tabela de partições a partir do disco rígido. A nova leitura da tabela de partições funciona na maioria dos casos, mas eventualmente pode não funcionar corretamente. Não entre em pânico, simplesmente reinicialize o sistema. Em todos os casos é recomendada sempre a reinicialização do sistema como medida de segurança.

Setas Move o cursor para as partições anterior ou posterior. Caso haja mais partições do que as que podem ser apresentadas na tela, pode-se mostrar o próximo conjunto de partições movendo-se para baixo na última partição mostrada na tela.

CTRL-L Redesenha a tela. Caso algo não funcione bem e não se possa ler nada, pode-se atualizar a tela a partir da linha de comandos da tela principal.

? Imprime a tela de ajuda.

Todos os comandos podem ser informados tanto em maiúsculas como em minúsculas (exceto o comando **W**(gravar)). Quando em um submenu ou em uma linha de comandos onde se deve informar um nome de arquivo, pode-se pressionar [ESC] para retornar para a linha de comandos da tela principal.

OPÇÕES

-a Usa um cursor em formato de seta ao invés de vídeo reverso para salientar a partição atual.

-v Lista a versão atual e os direitos autorais.

-z Inicia com uma tabela de partições zerada. Esta opção é útil quando se deseja reparticionar todo o disco. *Nota:* esta opção não zera realmente a tabela de partições no disco; na verdade simplesmente inicia o programa sem ler a partição atual.

-c *cilindros*

-h *cabeças*

-s *set/tril* Sobrepõe o número de cilindros, cabeças e setores por trilha lidos a partir do BIOS. Caso seu BIOS ou adaptador não forneça estas informações ou caso ela disponibilize informações incorretas, utilize estas opções para configurar os valores da geometria do disco.

-P *opc* Lista a tabela de partições nos formatos especificados. *opc* pode ser um ou mais de "r", "s" ou "t". Veja o comando **p (listar)**. Veja acima para maiores informações sobre os formatos de listagem.

VEJA TAMBÉM

fdisk(8)

PROBLEMAS

A versão atual não suporta múltiplos discos (para adição futura).

AUTOR

Kevin E. Martin (martin@cs.unc.edu)

E.3 chroot

NOME

chroot - executa comandos ou shell interativo com diretório raiz especial.

SINOPSE

chroot [–help] [–version] diretório [comando...]

DESCRIÇÃO

Este documento não é mais mantido e pode conter informações imprecisas ou desatualizadas. A documentação em Textinfo é a fonte oficial.

Este manual documenta a versão GNU do **chroot**. O **chroot** executa uma alteração do diretório raiz durante a execução de um determinado comando para o diretório informado. Se nenhum comando for informado **chroot** executa um shell interativo. A variável de ambiente 'SHELL' especifica o programa a ser utilizado. O padrão é /bin/sh.

OPÇÕES

–help Imprime uma mensagem sobre o uso deste comando na saída padrão e conclui a execução do programa.

–version Imprime as informações de versão na saída padrão e conclui a execução do programa.

NOTAS

Em muitos sistemas somente o superusuário pode mudar o diretório raiz.

E.4 cron

NOME

cron - servidor que executa comandos agendados (Vixie Cron)

SINOPSE

cron

DESCRIÇÃO

O *cron* deve ser iniciado a partir do /etc/rc ou /etc/rc.local. Ele retornará imediatamente, não sendo necessário então iniciá-lo com '&'.

O *cron* pesquisa o diretório /var/spool/cron por arquivos crontab, os quais têm o nome do usuário, conforme aparecem no /etc/passwd, como sufixo do arquivo; os arquivos crontabs encontrados são carregados em memória. *Cron* pesquisa ainda no /etc/crontab o qual usa um formato diferente (veja *crontab(5)). Cron* é iniciado a cada minuto, examinando todos os crontabs armazenados, verificando em cada comando, se ele deve ser executado naquele minuto. Ao executar os comandos, qualquer saída será enviada via email para o dono do arquivo crontab (ou para o usuário definido na variável de ambiente MAILTO em crontab, caso exista).

Adicionalmente, *cron* verifica a cada minuto se houve alguma mudança na data de seu diretório de serviços (ou na data de */etc/crontab),* e em caso positivo o *cron* irá examinar a data de todos os arquivos crontab e recarregar aqueles que foram alterados. Assim, o *cron* não necessita ser reinicializado toda vez que um arquivo crontab for alterado. Note que um comando de atualização de um arquivo *Crontab(1)* atualiza também a data dos diretórios de serviços.

VEJA TAMBÉM

crontab(1), crontab(5)

AUTOR

Paul Vixie <paul@vix.com>

E.5 crontab

NOME

crontab - mantém arquivos crontab para usuários individuais (V3)

SINOPSE

crontab [-u usuário] arquivo

crontab [-u usuário] { -l | -r | -e }

DESCRIÇÃO

Crontab é o programa usado para instalar, desinstalar ou listar as tabelas usadas para orientar o servidor *cron(8)* no Vixie Cron. Cada usuário pode ter seu próprio crontab, e, apesar destes arquivos estarem no /var, eles não devem ser editados diretamente.

Caso o arquivo *allow* exista, então o usuário deve estar incluído neste arquivo para permitir o uso deste comando. Caso o arquivo *allow* não existe, mas exista o arquivo *deny* , então o usuário **não** pode estar listado no arquivo *deny* para que se possa usar este comando. Caso nenhum destes arquivos exista, então, dependendo dos parâmetros de configuração da máquina, somente o superusuário terá permissão de usar este comando, ou todos os usuários poderão usá-lo.

Caso a opção *-u* seja informada, ela indicará o nome do usuário cujo arquivo crontab será tratado. Caso esta opção não seja dada, *crontab* examina o crontab do usuário que esteja executando o comando. Note que o comando *su(8)* pode confundir o *crontab* e caso se esteja executando o *crontab* desta forma, é recomendado sempre usar a opção *-u* .

A primeira forma de uso deste comando é a instalação de um novo crontab a partir de um arquivo já existente ou da entrada padrão se o caracter "-" for informado.

A opção *-l* faz com que o crontab seja listado na saída padrão.

A opção *-r* remove o crontab atual.

A opção *-e* é usada para editar o crontab atual usando o editor especificado nas variáveis de ambiente VISUAL ou EDITOR. Após a finalização da edição, o arquivo crontab modificado terá efeito imediato.

VEJA TAMBÉM

crontab(5), cron(8)

ARQUIVOS

/etc/cron.allow

/etc/cron.deny

PADRÕES

O comando *crontab* tem conformidade com IEEE Std1003.2-1992 ("POSIX"). Esta nova sintaxe de comando difere das versões anteriores do Vixie Cron, assim como da sintaxe clássica de SVR3.

DIAGNÓSTICOS

Uma mensagem explicativa aparece caso o programa seja executado de forma equivocada.

AUTOR

Paul Vixie <paul@vix.com>

E.6 crontab

NOME

crontab - tabelas para controle do cron

DESCRIÇÃO

Um arquivo *crontab* contém instruções para o daemon *cron(8)*, na forma: "execute este comando nesta hora e nesta data". Cada usuário tem seu próprio crontab, e comandos em um dado crontab são executados como se fosse pelo usuário que possui o crontab. Uucp e News normalmente têm seus próprios crontabs, eliminando a necessidade de executar *su(1)* explicitamente como parte de um comando do cron.

Linhas em branco, espaços iniciais e tabs são ignorados. Linhas cujo primeiro caractere não-branco for um '#' são comentários, e são ignoradas. Note que comentários não podem estar na mesma linha que comandos do cron, já que serão considerados parte do comando. Analogamente, comentários não podem estar na mesma linha que definições de variáveis de ambiente.

Uma linha ativa em um crontab é ou uma definição de ambiente ou um comando do cron. Uma definição de ambiente é da forma

nome = valor

onde os espaços em torno do sinal de igual (=) são opcionais, e quaisquer espaços não iniciais em *valor* são parte do valor atribuído a *nome*. A string *valor* pode ser colocada entre aspas (simples ou duplas, mas correspondentes) para preservar espaços iniciais ou finais.

Várias variáveis de ambiente são definidas automaticamente pelo daemon *cron(8)*. SHELL é definido como /bin/sh, LOGNAME e HOME são definidos a partir da linha de /etc/passwd referente ao dono do crontab. HOME e SHELL podem ser sobrescritos; LOGNAME não.

(Outra observação: a variável LOGNAME às vezes é chamada USER em sistemas BSD... nesses sistemas, USER será definido também.)

Além de LOGNAME, HOME, e SHELL, *cron(8)* olha em MAILTO se tiver algum motivo para enviar mail como resultado da execução de comandos neste crontab. Se MAILTO for definida (e não vazia), o mail é enviado para o usuário designado. Se MAILTO for definida mas vazia (MAILTO=), não é enviado nenhum mail. Caso contrário, o mail é enviado ao dono do crontab. Esta opção é útil se você decidir que o mailer será /bin/mail ao invés de /usr/lib/sendmail quando instalar cron – /bin/mail não faz aliasing, e UUCP normalmente não lê seu mail.

O formato de um comando do cron é em grande parte o padrão V7, com várias extensões upward-compatible. Cada linha tem cinco campos de hora e data, seguidos por um nome de usuário se este arquivo crontab for o do sistema, seguidos por um comando. Comandos são executados pelo *cron(8)* quando os campos minuto, hora, e mês correspondem à hora atual, *e* quando pelo menos um dos campos de dia (dia do mês, ou dia da semana) correspondem ao dia atual (veja "Observação" abaixo). O *cron(8)* examina as entradas cron a cada minuto. Os campos de hora e data são:

Tabela E.1: Campos de data e hora do cron

campo	valores permitidos
minuto	0-59
hora	0-23
dia do mês	0-31
mês	0-12 (ou nomes, veja abaixo)
dia da semana	0-7 (0 ou 7 é domingo, ou use nomes)

Um campo pode ser um asterisco (*), que sempre significa "primeiro-último".

Pode-se usar intervalos de números. Um intervalo é um par de números separados por um hífen. O intervalo especificado é inclusivo. Por exemplo, 8-11 em uma entrada "horas" especifica execução às 8, 9, 10 e 11 horas.

Pode-se usar listas. Uma lista é um conjunto de números (ou intervalos) separados por vírgulas. Exemplos: "1,2,5,9", "0-4,8-12".

Valores de passo podem ser usados em conjunto com intervalos. Um "/<número>" imediatamente após um intervalo especifica um passo no valor do número através do intervalo. Por exemplo, "0-23/2" pode ser usado no campo das horas para especificar que o comando deve ser executado a cada duas horas (a alternativa no padrão V7 seria "0,2,4,6,8,10,12,14,16,18,20,22"). Passos também são permitidos depois de um asterisco. Logo, se você quer dizer "a cada duas horas", simplesmente use "*/2".

Nomes (em inglês) também podem ser usados nos campos "mês" e "dia da semana". Use as primeiras três letras do dia ou mês desejado (em maiúsculas ou minúsculas, não importa). Intervalos ou listas de nomes não são permitidos.

O campo "sexto" (o resto da linha) especifica o comando a ser executado. Toda a parte da linha correspondente ao comando, até um fim-de-linha ou um caractere '%', será executada por /bin/sh ou pelo shell especificado na variável SHELL do arquivo cron. Símbolos de porcentagem (%) no comando, a menos que precedidos por uma barra invertida (\), são transformados em caracteres fim-de-linha, e todos os dados após o primeiro '%' são enviados ao comando como entrada padrão.

Observação: o dia de execução de um comando pode ser especificado por dois campos – dia do mês e dia da semana. Se ambos os campos são restritos (i.e., diferentes de '*'), o comando será executado quando *qualquer dos dois* campos corresponder à data atual. Por exemplo,

"30 4 1,15 * 5" faria com que o comando fosse executado às 4:30 nos dias 1 e 15 de cada mês, e em todas as sextas-feiras.

EXEMPLO DE ARQUIVO CRON

usa /bin/sh para executar comandos, independentemente do que /etc/passwd diz.

SHELL=/bin/sh

envia (por mail) toda saída para 'paul', independentemente de quem

é o dono do crontab.

MAILTO=paul

#

executa a 0h05, todo dia.

5 0 * * * $HOME/bin/daily.job >> $HOME/tmp/out 2>&1

executa às 14h15 no dia primeiro de cada mês – a saída é enviada para paul

15 14 1 * * $HOME/bin/monthly

executa às 22h00 em dias de semana.

0 22 * * 1-5 mail -s "São 22 horas" joe%Joe,%%Where are your kids?%

23 0-23/2 * * * echo "executa a 0h23, 2h23, 4h23 ..., todo dia"

5 4 * * sun echo "executa às 4h05 todo domingo"

VEJA TAMBÉM

cron(8), crontab(1)

EXTENSÕES

Na especificação do dia da semana, tanto dia 0 como dia 7 são considerados domingo. BSD e ATT parecem discordar nesse ponto.

Listas e intervalos podem coexistir no mesmo campo. "1-3,7-9" seria rejeitado pelo cron da ATT ou do BSD – eles só aceitam "1-3" ou "7,8,9".

Intervalos podem incluir "passos", logo "1-9/2" é o mesmo que "1,3,5,7,9".

Meses ou dias da semana podem ser especificados pelo nome (em inglês).

Variáveis de ambiente podem ser definidas no crontab. No BSD ou ATT, o ambiente dado aos processos filhos é basicamente o de /etc/rc.

A saída dos comandos pode ser enviada por mail ao dono do crontab (BSD não faz isso), pode ser enviada a outra pessoa que não o dono do crontab (SysV não faz isso), e este recurso pode ser desligado de modo que nenhum mail seja enviado (SysV não faz isso tampouco).

AUTOR

Paul Vixie <paul@vix.com>

E.7 debugfs

NOME

debugfs - um depurador do sistema de arquivos ext2

SINOPSE

debugfs [**-w**] [**-f** arquivo-de-comandos] [**-R** solicitação] [**-V**] [dispositivo]

DESCRIÇÃO

O programa **debugfs** é um depurador do sistema de arquivos. Ele pode ser usado para examinar e mudar o estado de um sistema de arquivos ext2.

dispositivo é o arquivo especial correspondente ao dispositivo contendo o sistema de arquivos ext2 (ex. /dev/hdXX).

OPÇÕES

-w Especifica que o sistema de arquivos deve ser aberto em modo de leitura e gravação. Sem esta opção, o sistema de arquivos é aberto em modo somente de leitura.

-f arquivo Faz com que **debugfs** leia os comandos em *arquivo*, e execute-os. Quando **debugfs** terminar de executar os comandos, ele irá terminar.

-R solicitação Faz com que **debugfs** execute o único comando *solicitação*, e então terminar.

-V Exibe o número de versão de **debugfs** e encerra sua execução.

COMANDOS

debugfs é um depurador interativo. Ele entende vários comandos.

cat arquivo Mostra o conteúdo do inode *arquivo* na saída padrão.

cd dir Muda o diretório de trabalho para *dir*.

chroot dir Muda o diretório raiz para *dir*.

close Fecha o sistema de arquivos atualmente aberto.

clri arquivo Limpa o conteúdo do inode *arquivo*.

dump -p arquivo arquivo-de-saída Descarrega o conteúdo do inode *arquivo* para o arquivo *arquivo-de-saída*. Se a opção *-p* for dada, então define informações sobre o dono, grupo e permissões de *arquivo-de-saída* para que casem com as de *arquivo*.

expand_ dir arquivo Expande o diretório *arquivo.*

find_ free_ block [início] Acha o primeiro bloco livre, começando por *início* e o aloca.

find_ free_ inode [arquivo [modo]] Acha um inode livre e o aloca. Se presente, *arquivo* especifica o número do inode do diretório cujo inode será localizado. O segundo parâmetro opcional *modo* especifica as permissões do novo inode. (Se o bit de diretório for definido no modo, a rotina de alocação irá funcionar diferentemente).

freeb bloco Marca o bloco número *bloco* como não alocado.

freei arquivo Libera o inode especificado por *arquivo.*

help Exibe uma lista de comandos entendidos por **debugfs(8)**.

icheck bloco ... Exibe uma listagem dos inodes que usam os blocos especificados na linha de comando.

initialize dispositivo tamanho-dos-blocos Cria um sistema de arquivos ext2 no *dispositivo* com o tamanho dos blocos *tamanho-dos-blocos.* Note que isto não inicia completamente todas as estruturas de dados; para isso, use o programa **mke2fs(8)**. Isto apenas faz uma chamada à biblioteca de baixo nível, que define o superbloco e os descritores de blocos.

kill_ file arquivo Desaloca o inode *arquivo* e os seus blocos. Note que isto não remove nenhuma entrada de diretório (se existir) neste inode. Veja o comando *rm* se você deseja desvincular um arquivo.

ln arquivo destino Cria um vínculo de nome *destino* que é um vínculo para *arquivo.* Note que isto não ajusta a contagem de referências do inode.

ls -l arquivo Exibe uma listagem dos arquivos no diretório *arquivo.*

modify_ inode arquivo Modifica o conteúdo da estrutura do inode *arquivo.*

mkdir arquivo Cria um diretório de nome *arquivo.*

mknod arquivo [p|[[c|b majoritário minoritário]]] Cria um arquivo especial de dispositivo (um pipe nomeado, ou dispositivo de caractere ou bloco). Se um dispositivo de caractere ou bloco for criado, os números de dispositivo *majoritário* e *minoritário* devem ser especificados.

ncheck num-inode ... Recebe a lista requisitada de números de inodes, e exibe uma listagem de caminhos para aqueles inodes.

open -w dispositivo Abre um sistema de arquivos para edição.

pwd Exibe o diretório de trabalho atual.

quit Encerra o **debugfs**.

rm caminho Desvincula o *caminho.* Isto faz com que o inode apontado por *caminho* fique sem nenhuma referência, desalocando o arquivo. Este comando funciona como a chamada do sistema unlink().

rmdir arquivo Remove o diretório de nome *arquivo.* Esta função ainda não está implementada.

setb bloco Marca o bloco número *bloco* como alocado.

seti arquivo Marca o inode *arquivo* como em uso no mapa de inodes.

show_ super_ stats Exibe o conteúdo do superbloco.

stat arquivo Exibe o conteúdo da estrutura do inode *arquivo.*

testb bloco Testa se o bloco número *bloco* está marcado como alocado no mapa de blocos.

testi arquivo Testa se o inode *arquivo* está marcado como alocado no mapa de inodes.

unlink caminho Remove o link especificado por *caminho* para um inode. Note que isto não ajusta a contagem de referências do inode.

write arquivo-origem arquivo-de-saída Cria um arquivo no sistema de arquivos de nome *arquivo-de-saída*, e copia o conteúdo de *arquivo-origem* para o arquivo de destino.

ESPECIFICANDO ARQUIVOS

Muitos comandos do **debugfs** recebem um *arquivo* como um parâmetro para especificar um inode (oposto a *caminho*) no sistema de arquivos que está atualmente aberto por debugfs. O argumento *arquivo* pode ser especificado de duas formas. A primeira forma é um número de inode cercado por sinais de menor e maior, ex: <*2*>. A segunda forma é um caminho; se o caminho for prefixado por uma barra ('/'), então é interpretado como sendo relativo à raiz do sistema de arquivos que está atualmente aberto por debugfs. Se não, o caminho é interpretado relativo ao diretório de trabalho atual como mantido por debugfs. Este pode ser modificado usando o comando do debugfs *cd*.

Nota de tradução: Note que *arquivo* pode também ser um diretório, já que para o sistema de arquivos um diretório é apenas um tipo diferente de arquivo.

AUTOR

debugfs foi escrito por Theodore Ts'o <tytso@mit.edu>.

VEJA TAMBÉM

dumpe2fs(8), **e2fsck**(8), **mke2fs**(8)

E.8 dumpe2fs

NOME

dumpe2fs - mostra informações a respeito do sistema de arquivos.

SINOPSE

dumpe2fs [**-b**] [**-V**] dispositivo

DESCRIÇÃO

dumpe2fs exibe informações sobre o super bloco e grupos de blocos para o sistema de arquivos presente no *dispositivo*.

dumpe2f é semelhante ao programa da Berkeley **dumpfs** para o BSD Fast File System.

OPÇÕES

-b exibe os blocos que estão reservados como ruins no sistema de arquivos.

-V exibe número da versão de dumpe2fs e encerra a execução.

FALHAS

Você precisa conhecer a estrutura física do sistema de arquivos e entender a saída.

AUTOR

dumpe2fs foi escrito por Remy Card <card@masi.ibp.fr>, o desenvolvedor e mantenedor do sistema de arquivos ext2.

DISPONIBILIDADE

dumpe2fs está disponível através de ftp anônimo em ftp.ibp.fr e tsx-11.mit.edu no diretório /pub/linux/packages/ext2fs.

VEJA TAMBÉM

e2fsck(8), **mke2fs**(8), **tune2fs**(8)

E.9 e2fsck

NOME

e2fsck - verifica um sistema de arquivos Linux second extended

SINOPSE

e2fsck [**-pacnyrdfvstFSV**] [**-b** *superbloco*] [**-B** *tamanho-dos-blocos*] [**-l**|**-L** *arquivo-de-blocos-ruins*] [**-C** *da*] *dispositivo*

DESCRIÇÃO

e2fsck é usado para verificar um sistema de arquivos Linux second extended.

dispositivo é o arquivo especial correspondente ao dispositivo (ex. /dev/hdXX).

OPÇÕES

-a Esta opção faz o mesmo que a opção *-p*. Ela é fornecida somente para compatibilidade reversa. Sugere-se que as pessoas usem a opção *-p* sempre que possível.

-b superbloco — Ao invés de usar o superbloco normal, usa o superbloco alternativo especificado por *superbloco*.

-B tamanho-dos-blocos — Normalmente, e2fsck vai procurar pelo superbloco em vários tamanhos de blocos diferentes numa tentativa de descobrir o tamanho apropriado. Esta busca pode ser enganada em alguns casos. Esta opção força o e2fsck a tentar localizar o superbloco somente em determinado tamanho de bloco. Se o superbloco não for encontrado, e2fsck terminará com um erro fatal.

-c — Esta opção fará e2fsck executar o programa **badblocks(8)** para localizar quaisquer blocos que estejam ruins no sistema de arquivos, e marcá-los como tal adicionando-os ao inode de blocos ruins.

-C — Esta opção faz com que **e2fsck** escreva informações sobre o seu andamento no descritor de arquivo especificado, para que o progresso de verificação do sistema de arquivos possa ser monitorado. Esta opção é tipicamente utilizada por programas que estão executando **e2fsck**, mas -C 0 é um caso especial que mostrará um caractere rodando que pode ser útil para usuários que querem alguma coisa para observar enquanto **e2fsck** cuida de seus negócios.

-d — Exibe informação de depuração (inútil a não ser que você esteja depurando **e2fsck**).

-f — Força a verificação mesmo que o sistema pareça limpo.

-F — Descarrega o buffer do dispositivo do sistema de arquivos antes de executar. Útil apenas para realizar medições de tempo do e2fsck.

-l nome-de-arquivo — Adiciona os blocos listados no arquivo especificado por *nome-de-arquivo* à lista de blocos ruins.

-L nome-de-arquivo — Define a lista de blocos ruins como a lista de blocos especificada por *nome-de-arquivo*. (Esta opção faz o mesmo que a opção *-l*, exceto que a lista de blocos ruins é apagada antes que os blocos listados no arquivo sejam adicionados à lista de blocos ruins.)

-n — Abre o sistema de arquivos somente para leitura, e assume uma resposta negativa (no) para todas as perguntas. Permite que **e2fsck** seja executado não-interativamente. (Nota: se as opções *-c*, *-l*, ou *-L* forem especificadas além da opção *-n*, então o sistema de arquivos será aberto para leitura e gravação, para permitir que a lista de blocos ruins seja atualizada. Entretanto, nenhuma outra modificação será feita no sistema de arquivos).

-p — Automaticamente repara o sistema de arquivos sem nenhuma pergunta.

-r — Esta opção não faz nada; é fornecida apenas para compatibilidade reversa.

-s — Esta opção irá trocar bytes do sistema de arquivos para que ele use a ordem de bytes normalizada, padrão (que é i386 ou little endian). Se o sistema de arquivos já estiver com a ordem de bytes padrão, e2fsck não realizará nenhuma ação.

-S — Esta opção irá trocar bytes do sistema, mesmo que ele já esteja na ordem padrão.

-t — Exibe estatísticas de tempo para **e2fsck**. Se esta opção for usada duas vezes, estatísticas adicionais serão impressas passo a passo.

-v — Modo detalhado.

-V — Exibe informações sobre a versão e encerra a execução.

-y — Assume uma resposta afirmativa (yes) para todas as perguntas; permite que **e2fsck** seja usado não-interativamente.

CÓDIGO DE SAÍDA

O código de saída retornado por **e2fsck** é a soma das seguintes condições:

0	Nenhum erro.
1	Erros do sistema de arquivos foram corrigidos.
2	Erros do sistema de arquivos corrigido, o sistema deve ser reiniciado se o sistema de arquivos estava montado.
4	Erros do sistema de arquivo não foram corrigidos.
8	Erro operacional.
16	Erro de uso ou de sintaxe.
128	Erro de biblioteca compartilhada.

RELATANDO FALHAS

Qualquer programa terá falhas. Se você conseguir achar um sistema de arquivos que faça **e2fsck** travar, ou que **e2fsck** seja incapaz de reparar, por favor relate ao autor.

Por favor inclua o máximo de informações possíveis em seu relatório de falhas. De preferência, inclua uma transcrição completa da execução de **e2fsck**, assim eu posso ver exatamente quais mensagens de erro foram exibidas. Se você tem um sistema de arquivos gravável onde a transcrição possa ser armazenada, o programa **script**(1) é uma forma prática de gravar a saída de dados em um arquivo.

Também é útil enviar a saída do programa **dumpe2fs**(8). Se um inode ou inodes específicos parecem estar dando problemas para **e2fsck**, tente executar o comando **debugfs**(8) e enviar a saída do comando *stat* no(s) inode(s) relevantes. Se o inode for um diretório, o comando *dump* do debugfs permitirá que você extraia o conteúdo do diretório do inode, que pode ser enviado para mim depois de executado através do programa **uuencode**(1).

Sempre inclua o texto de versão completo que **e2fsck** exibe quando é executado, para que eu saiba que versão você está executando.

AUTOR

Esta versão de **e2fsck** foi escrita por Theodore Ts'o <tytso@mit.edu>.

VEJA TAMBÉM

mke2fs(8), **tune2fs**(8), **dumpe2fs**(8), **debugfs**(8)

E.10 fdformat

NOME

fdformat - formatação de baixo nível em disquetes

SINOPSE

fdformat [-n] dispositivo

DESCRIÇÃO

fdformat executa uma formatação de baixo nível em um disquete. *dispositivo* é normalmente um dos seguintes (para dispositivos de disquetes, major = 2, e o minor é mostrado somente para propósitos de informação):

/dev/fd0d360 (minor = 4)

/dev/fd0h1200 (minor = 8)

/dev/fd0D360 (minor = 12)

/dev/fd0H360 (minor = 12)

/dev/fd0D720 (minor = 16)

/dev/fd0H720 (minor = 16)

/dev/fd0h360 (minor = 20)

/dev/fd0h720 (minor = 24)

/dev/fd0H1440 (minor = 28)

/dev/fd1d360 (minor = 5)

/dev/fd1h1200 (minor = 9)

/dev/fd1D360 (minor = 13)

/dev/fd1H360 (minor = 13)

/dev/fd1D720 (minor = 17)

/dev/fd1H720 (minor = 17)

/dev/fd1h360 (minor = 21)

/dev/fd1h720 (minor = 25)

/dev/fd1H1440 (minor = 29)

Os dispositivos de disquetes genéricos, /dev/fd0 e /dev/fd1, não funcionarão com **fdformat** quando um formato não padrão estiver sendo usado, ou o formato não seja autodetectado. Neste caso use **setfdprm(8)** para carregar os parâmetros de disco.

OPÇÕES

-n Não verificar. Esta opção desabilitará a verificação que é realizada após a formatação.

VEJA TAMBÉM

fd(4), **setfdprm**(8), **mkfs**(8), **emkfs**(8)

AUTOR

Werner Almesberger (almesber@nessie.cs.id.ethz.ch)

E.11 fdisk

NOME

fdisk - Manipulador da tabela de partições para o Linux

SINOPSE

fdisk [-b] [-u] [*dispositivo***]**

fdisk -l [-b] [-u] [*dispositivo ...***]**

fdisk -s *partição ...*

fdisk -v

DESCRIÇÃO

Discos rígidos podem ser divididos em um ou mais discos lógicos chamados de *partições*. Esta divisão é descrita na *tabela de partições* encontrada no setor 0 do disco.

No mundo BSD as pessoas falam sobre 'disk slices' e 'disklabel'.

O Linux precisa de pelo menos uma partição, para o seu sistema de arquivos raiz. Ele pode usar arquivos de swap ou partições de swap, mas as partições são mais eficientes. Normalmente usa-se uma segunda partição no Linux dedicada como swap. Em equipamentos compatíveis com o padrão Intel, o BIOS que inicializa o sistema normalmente pode acessar somente os primeiros 1024 cilindros do disco. Por esta razão, muitas pessoas com discos grandes criam uma terceira partição, com somente alguns Mb, normalmente montada no */boot*, para guardar a imagem do kernel e alguns arquivos auxiliares necessários em tempo de inicialização do sistema. Assim, tem-se a certeza que tudo será acessível pelo BIOS. Podem ainda existir razões de segurança, facilidade de administração e geração de cópias de segurança, ou teste, para usar um número maior de partições no sistema.

O **fdisk** (na primeira forma de execução) é um programa sob a forma de menus para a criação e manipulação de tabelas de partição. Ele conhece tabelas de partições DOS e BSD ou disklabels da SUN.

O *dispositivo* é normalmente um dos seguintes:

/dev/hda

/dev/hdb

/dev/sda

/dev/sdb

(/dev/hd[a-h] para discos IDE, /dev/sd[a-p] para discos SCSI, /dev/ed[a-d] para discos ESDI, /dev/xd[ab] para discos XT). Um nome de dispositivo refere-se ao disco inteiro.

A *partição* é um *dispositivo* seguido por um número de partição. Por exemplo, **/dev/hda1** é a primeira partição do primeiro disco IDE no sistema. Discos IDE podem ter até 63 partições, discos SCSI até 15. Veja também */usr/src/linux/Documentation/devices.txt*.

Um disklabel BSD/SUN pode descrever 8 partições, sendo que a terceira deve conter todo o disco. Não inicie uma partição que usa seu primeiro setor (como uma partição swap) no cilindro 0, pois esta irá destruir o disklabel.

Uma tabela de partição do DOS pode descrever um número ilimitado de partições. No setor 0 há espaço para a descrição de 4 partições (chamadas 'primárias'). Uma delas pode ser uma partição estendida; esta guarda as partições lógicas, com os descritores localizados em uma lista ligada de setores, cada um precedendo a partição lógica correspondente. As quatro partições primárias, presentes ou não, possuem números de 1 a 4. Partições lógicas iniciam no número 5.

Em uma tabela de partição do DOS, o deslocamento inicial e o tamanho de cada partição é armazenada de duas maneiras: como um número absoluto de setores (dados em 32 bits) e como uma tripla cilindros/cabeças/setores (dados em 10+8+6 bits). A primeira funciona até 2 TB, utilizando setores de 512-bytes. A segunda tem dois problemas diferentes. Primeiro, os campos de C/C/S podem ser preenchidos somente quando o número de cabeças e o número de setores por trilha são conhecidos. Depois, mesmo que se conheça estes números, os 24 bits que estão disponíveis não são suficientes. O DOS usa somente a trinca C/C/S, o Windows ambas e o Linux nunca as usa.

Se possível, o **fdisk** obterá a geometria do disco automaticamente. Esta não é necessariamente a geometria física do disco (ainda, discos novos não têm algo como uma geometria física, certamente nada que pode ser descrito da forma simplista cilindro/cabeças/setores) mas é a geometria do disco que o MS-DOS usa para a tabela de partições.

Normalmente tudo funciona bem por padrão, e não existem problemas caso o Linux seja o único sistema operacional no disco. Contudo, caso o disco seja dividido com outro sistema operacional, é uma boa idéia deixar o fdisk do outro sistema criar pelo menos uma partição. Quando o Linux inicializa, ele acessa a tabela de partição e tenta deduzir qual geometria é adequada para a cooperação com outros sistemas.

Sempre que a tabela de partições é mostrada, uma checagem de consistência é feita nas entradas da tabela. Esta checagem verifica que o início físico e lógico, bem como os pontos finais são idênticos, e também que a partição inicia e termina nos limites de um cilindro (exceto para a primeira partição).

Algumas versões do MS-DOS criam a primeira partição fora dos limites de um cilindro, ficando no segundo setor do primeiro cilindro. Partições começando no primeiro cilindro não iniciam nos limites do cilindro, mas normalmente não causam problemas a menos que você use o OS/2 em sua máquina.

Um sync() e um BLKRRPART ioctl() (releitura da tabela de partição do disco) são executados antes da saída quando a tabela de partições foi atualizada. A tempos atrás era necessário reinicializar o sistema depois do uso do fdisk. Eu não acho que isto seja mais necessário - de fato, reinicializar o sistema muito rapidamente pode causar perda de dados ainda não escritos. Note que o kernel e o hardware do disco podem armazenar dados (buffers) antes de gravá-los.

DOS 6.x ALERTA

O comando FORMAT do DOS 6.x procura por informações no primeiro setor da área de dados da partição, e utiliza esta informação ao invés da que está na tabela de partições. O format do DOS espera que o fdisk (do DOS) limpe os primeiros 512 bytes da área de dados da partição caso qualquer mudança de tamanho ocorra. O format do DOS irá usar esta informação adicional mesmo que o indicador /U for usado – nós consideramos isto um erro (bug) no format e no fdisk do DOS.

A questão é que se você usar o cfdisk ou o fdisk para trocar o tamanho de uma partição DOS, então deverá também usar o **dd** para zerar os primeiros 512 bytes desta partição antes de usar o FORMAT do DOS para formatá-la. Por exemplo, caso você esteja usando o cfdisk para criar uma partição DOS para o /dev/hda1, então (depois de sair do fdisk ou do cfdisk e reinicializar o Linux para ter certeza que a informação na tabela de partições é válida) você deverá usar o comando "dd if=/dev/zero of=/dev/hda1 bs=512 count=1" para zerar os primeiros 512 bytes desta partição.

SEJA EXTREMAMENTE CUIDADOSO caso você use o comando **dd,** pois o menor descuido pode fazer com que todos os seus dados sejam perdidos.

Para melhores resultados, você deve sempre usar o programa particionador que acompanha cada sistema operacional. Por exemplo, crie partições DOS com o DOS FDISK e partições Linux com o fdisk ou o cfdisk do Linux.

OPÇÕES

-v Mostra o número da versão do **fdisk** e sai.

-l Lista a tabela de partições para os seguintes dispositivos **/dev/hd[a-d]**, **/dev/sd[a-h]**, **/dev/ed[a-d]**, e sai.

-b Na listagem de tabelas de partições, também imprime uma coluna de início ('Begin') como versões antigas do fdisk faziam por padrão. (Nota: os valores nesta coluna, quando dados em cilindros, não podem ser maiores que 1023. Não há nada de errado caso o 'Begin' e o 'Start' sejam diferentes, pelo menos nada que o Linux se importe.)

-u Na listagem de tabelas de partições, mostra os tamanhos em setores ao invés de cilindros.

-s *partição* O *tamanho* da partição (em blocos) é mostrado na saída padrão. Este valor é normalmente usado como um argumento para o programa **mkfs(8)** para especificar o tamanho da partição que irá ser formatada. (Versões antigas do fdisk somente farão isso caso a identificação da partição for maior que 10, na tentativa de recusar partições DOS; este teste foi retirado.) Note que o **sfdisk -s** mostra diferentes (na verdade, corretas) respostas. A razão para a diferença é que o kernel e o **fdisk** não necessitam ter o mesmo padrão sobre numeração de partições (isto é, no caso de você ter partições BSD), e podem ter idéias diferentes sobre o tamanho de um partição estendida.

BUGS

Existem diversos *fdisk por aí. Cada um tem seus prós e contras. Teste-os nesta ordem **cfdisk**, **fdisk**, **sfdisk**.

E.12 fsck

NOME

fsck - verifica e repara um sistema de arquivos.

SINOPSE

fsck [**-AVRTNP**] [**-s**] [**-t** *tipo-de-sistema*] [**opções-do-sistema**] *sistema-de-arquivos* [...]

DESCRIÇÃO

fsck é usado para verificar e opcionalmente reparar um sistema de arquivos do Linux. *sistema-de-arquivos* pode ser o nome do dispositivo (ex. /dev/hda1, /dev/sdb2) ou o ponto de montagem para o sistema de arquivos (ex. /, /usr, /home). Se forem fornecidos diversos sistemas de arquivos em diferentes discos físicos para verificação, **fsck** tentará verificá-los em paralelo. Isto reduz o tempo total de verificação de todos os sistemas de arquivos, já que fsck tira vantagem do paralelismo de múltiplos discos.

O código de saída retornado por **fsck** é a soma das seguintes condições:

0	Nenhum erro.
1	Erros do sistema de arquivos corrigidos.
2	O sistema deve ser reiniciado.
4	Erros do sistema de arquivos não corrigidos.
8	Erro operacional.
16	Erro de uso ou de sintaxe.
128	Erro de biblioteca compartilhada.

O código de saída retornado quando todos os sistemas de arquivos são checados usando a opção **-A** é um OU bit a bit dos códigos de saída de cada sistema de arquivos verificado.

Na verdade, **fsck** é simplesmente um intermediário para os vários verificadores de sistemas de arquivos (**fsck** *.tipo-de-sistema*) disponíveis no Linux. O executável do verificador específico do sistema de arquivos é procurado primeiro em /sbin, então em /etc/fs, e finalmente nos diretórios listados na variável de ambiente PATH. Por favor veja a página de manual do verificador de sistemas de arquivos específico para mais detalhes.

OPÇÕES

-A Analisa o arquivo */etc/fstab* e tenta verificar todos os arquivos listados de uma vez. Esta opção é normalmente usada em um arquivo de inicialização em */etc/rc* ao invés de múltiplos comandos para verificar um único sistema de arquivos.

-R Quando verificando todos os sistemas de arquivos com a opção **-A**, pula o sistema de arquivos raiz (neste caso ele já estaria montado para leitura e gravação).

-T Não mostra o título no início.

-N Não executa, apenas mostra o que seria feito.

-P Quando a opção **-A** é definida, verifica o sistema de arquivos raiz em paralelo com os outros sistemas de arquivos. Esta não é a coisa mais segura no mundo a se fazer. Já que se duvida da integridade do sistema de arquivos raiz, coisas como o executável do e2fsck podem estar danificadas! Esta opção é fornecida principalmente para aqueles administradores de sistema que não querem reparticionar o sistema de arquivos raiz para que seja pequeno e compacto (o que é de fato a solução correta).

-s Serializa as operações do fsck. Esta é uma boa idéia se você está verificando múltiplos sistemas de arquivos e os verificadores estão em modo interativo. (Nota: **e2fsck** é executado no modo interativo por padrão. Para executá-lo em modo não-interativo, você deve especificar as opções **-p** ou **-a** se você deseja que os erros sejam corrigidos automaticamente, ou a opção **-n** caso contrário).

-V Produz uma saída detalhada, incluindo todos os comandos específicos do sistema de arquivos que forem executados.

-t *tipo-de-sistema* Especifica o tipo de sistema de arquivos a ser verificado. Quando a opção **-A** estiver especificada, somente os sistemas de arquivos que casarem com *tipo-de-sistema* serão verificados. Se *tipo-de-sistema* estiver prefixado com **no**, somente os sistemas de arquivos que não casarem com *tipo-de-sistema* serão verificados.

Normalmente, o tipo do sistema de arquivos é deduzido procurando por ele no arquivo */etc/fstab* e usando a entrada correspondente. Se o tipo não puder ser deduzido, então **fsck** usará o tipo especificado pela opção **-t**, se ela especificar um único sistema de arquivos. Se o tipo não estiver disponível, então o *tipo-de-sistema* predefinido (atualmente ext2) é usado.

opções-do-sistema Quaisquer opções que não sejam entendidas por **fsck**, ou que seguem a opção – são tratadas como opções específicas do sistema de arquivos e serão passadas ao verificador específico.

Atualmente, um padrão para as opções específicas do sistema de arquivos está em fluxo. Apesar de não haver nenhuma garantia, as seguintes opções são suportadas pela maioria dos verificadores de sistemas de arquivos:

-a Automaticamente repara o sistema de arquivos sem nenhuma pergunta (use esta opção com cautela). Note que **e2fsck** suporta **-a** somente para compatibilidade reversa. Esta opção é mapeada para a opção **-p** que é mais segura de usar, diferente da opção **-a** que a maioria dos verificadores de sistema de arquivos suporta.

-r Interativamente repara o sistema de arquivos (pergunta por confirmações). Nota: é geralmente uma má idéia usar esta opção se múltiplos **fsck**'s estão sendo executados em paralelo. Note também que este é o comportamento padrão para **e2fsck**. Ele suporta esta opção somente por razões de compatibilidade reversa.

AUTOR

Theodore Ts'o (tytso@mit.edu)

Esta página de manual foi descaradamente adaptada do programa intermediário genérico fsck de David Engel e Fred van Kempen, que por sua vez foi descaradamente adaptada da versão de Remy Card para o sistema de arquivos ext2.

ARQUIVOS

/etc/fstab.

VEJA TAMBÉM

fstab(5), **mkfs**(8), **fsck.minix**(8), **fsck.ext2**(8) ou **e2fsck**(8), **fsck.xiafs**(8).

E.13 fstab

NOME

fstab - informações estáticas sobre os sistemas de arquivos.

SINOPSE

#include <fstab.h>

DESCRIÇÃO

O arquivo **fstab** contém a descrição dos vários sistemas de arquivos. **fstab** é lido somente pelos programas, mas não é gravado; é uma tarefa do administrador do sistema criar e manter este arquivo. Cada sistema de arquivos é descrito em uma linha em separado; campos em cada linha são separados por tabulações ou espaços. A ordem dos registros no **fstab** é importante porque o **fsck**(8), **mount**(8), e **umount**(8) interagem seqüencialmente através do **fstab** para executar suas tarefas.

O primeiro campo, (*fs_ spec*), descreve um dispositivo especial de blocos ou um sistema de arquivos remoto a ser montado.

O segundo campo, (*fs_file*), descreve o ponto de montagem para o sistema de arquivos. Para partições da área de swap, este campo deve ser especificado como "none".

O terceiro campo, (*fs_ vfstype*), descreve o tipo do sistema de arquivos. O sistema atualmente suporta estes tipos de sistemas de arquivos (e possivelmente outras listas em */proc/filesystems*):

minix um sistema de arquivos local, suporta nomes de arquivos com 14 ou 30 caracteres.

ext um sistema de arquivos local com nomes de arquivos longos e inodes maiores. Este sistema de arquivos foi substituído pelo *ext2* , e não deve mais ser utilizado.

ext2 um sistema de arquivos local, com nomes de arquivos longos, inodes maiores e muitas outras funcionalidades.

xiafs um sistema de arquivos local com nomes de arquivos longos, inodes maiores e outras funcionalidades.

msdos um sistema de arquivos local para partições MS-DOS.

hpfs um sistema de arquivos local para partições HPFS.

iso9660 um sistema de arquivos local usado para dispositivos CDROM.

nfs um sistema de arquivos para montagem de partições a partir de sistemas remotos.

swap uma partição de disco usada como memória auxiliar.

Caso *vfs_fstype* seja especificado como “ignore” então a entrada será ignorada. Isso é útil para mostrar as partições de disco que estão sem uso.

O quarto campo, (*fs_mntops*), descreve as opções de montagem associadas com o sistema de arquivos. É formatado com uma vírgula separando uma lista de opções. Contém no mínimo o tipo da montagem mais qualquer opção adicional apropriada para o sistema de arquivos. Para verificar as opções disponíveis para sistemas de arquivos não nfs, veja **mount(8)**. Para documentação específica ao nfs veja **nfs(5)**. Comum a todos os tipos de sistemas de arquivos são as opções “noauto”(não montar quando "mount -a" for executado, por exemplo durante a inicialização do sistema), e “user” (permite que um usuário monte o sistema). Para maiores detalhes veja **mount(8)**.

O quinto campo, (*fs_freq*), é usado nos sistemas de arquivos pelo comando **dump(8)** para determinar quais sistemas de arquivos necessitam ser copiados. Caso o quinto campo não esteja presente, um valor zero é retornado e **dump** assumirá que o sistema de arquivos não necessita ser copiado.

O sexto campo, (*fs_passno*), é usado pelo programa **fsck(8)** para determinar a ordem em que os sistemas de arquivos são checados durante a reinicialização do sistema. O sistema de arquivos raiz deve ser especificado com um *fs_passno* de 1, e outros sistemas de arquivos devem ter *fs_passno* de 2. Sistemas de arquivos em um mesmo dispositivo serão checados seqüencialmente, mas sistemas de arquivos em diferentes dispositivos serão checados ao mesmo tempo para utilizar o paralelismo disponível com o hardware. Caso o sexto campo não esteja presente ou seja igual a zero, um valor igual a zero é retornado e **fsck** assumirá que o sistema de arquivos não necessita ser checado.

A maneira correta de ler os registros do **fstab** é usar as rotinas **getmntent(3)**.

ARQUIVOS

/etc/fstab O arquivo **fstab** reside em */etc*.

PROBLEMAS

A documentação em **mount(8)** é freqüentemente mais atualizada.

VEJA TAMBÉM

getmntent(3), **mount(8)**, **swapon(8)**, **nfs(5)**

HISTÓRIA

O formato do arquivo **fstab** apareceu em 4.0BSD.

E.14 inittab

NOME

inittab - formato do arquivo inittab usado pelo processo init compatível com o system V

DESCRIÇÃO

O arquivo **inittab** descreve quais processos são iniciados na inicialização e durante a operação normal (por exemplo /etc/rc.d/init.d/boot, /etc/rc.d/rc, gettys...). **Init(8)** distingue múltiplos *níveis de execução*, cada um dos quais com o seu próprio conjunto de processos a serem iniciados. Níveis de execução válidos estão entre **0** **-6** além de **A** , **B** , e **C** para entradas **ondemand** . Uma entrada no arquivo **inittab** tem o seguinte formato:

identificação :*níveis de execução* :*ação* :*processo*

Linhas começando com '#' são ignoradas.

identificação é uma seqüência única de 1 a 4 caracteres os quais identificam uma entrada no **inittab** (para versões do sysvinit compiladas com bibliotecas < 5.2.18 ou bibliotecas a.out o limite é de 2 caracteres).

Nota: para o getty ou outros programas de acessos, o campo *identificação* deve ter o sufixo tty do tty correspondente, por exemplo **1** para **tty1**. De outra forma o programa de acesso ao sistema poderá não funcionar corretamente.

níveis listas dos níveis de execução para os quais a ação especificada deverá ser ativada.

ação descreve qual ação será executada.

processo especifica o processo a ser executado. Caso o campo processo comece com o caracter '+', **init** não criará registros em utmp e wtmp para aqueles processos. Isto é necessário para gettys que insistem em ter a sua própria manutenção de utmp/wtmp. Este é também um problema histórico.

O campo *níveis de execução* pode conter múltiplos caracteres para diferentes níveis. Por exemplo, **123** especifica que o processo deve ser executado no níveis de execução 1, 2 e 3. O campo *níveis de execução* para entradas **ondemand** devem conter um **A**, **B**, ou **C**. O campo *níveis de execução* das entradas **sysinit** , **boot**, e **bootwait** é ignorado.

Quando um nível de execução de sistema é alterado, e processos que sejam do novo nível de execução estejam sendo executados, eles serão primeiramente finalizados, inicialmente com o comando SIGTERM, e após com SIGKILL.

Ações válidas para o campo *ação* são:

respawn O processo será reinicializado sempre que finalizar (por exemplo getty).

wait O processo será iniciado uma vez, quando o nível de execução foi informado e **init** irá esperar pela sua finalização.

once O processo será executado uma única vez, quando o nível de execução for iniciado.

boot O processo será executado durante a inicialização do sistema. O campo *nível de execução* é ignorado.

bootwait O processo será executado durante a inicialização do sistema, e **init** esperará pela sua conclusão (por exemplo /etc/rc). O campo *níveis de execução* será ignorado.

off Não faz nada.

ondemand Um processo definido no nível de execução **ondemand** (0nline) será executado sempre que o nível de execução **ondemand** for acionado. De qualquer forma nenhuma alteração de nível de execução será efetivada (níveis de execução **ondemand** são 'a', 'b', e 'c').

initdefault Uma entrada **initdefault** especifica o nível de execução padrão do sistema após a sua inicialização. Caso não exista, **init** irá perguntar na console qual o nível de execução a ser utilizado. O campo *processo* é ignorado.

sysinit o processo será executado durante a inicialização do sistema. Será executado antes das entradas **boot** ou **bootwait** . O campo *níveis de execução* será ignorado.

powerwait O processo será executado quando **init** receber um sinal SIGPWR, indicando que há algo errado com a alimentação elétrica. **Init** irá esperar pela finalização do processo antes de continuar.

powerfail Assim com em **powerwait** , exceto que **init** não aguarda a finalização do processo.

powerokwait O processo será executado quando **init** receber um sinal SIGPWR, alertando que há um arquivo chamado **/etc/powerstatus** contendo a palavra **OK**. Isto significa que a alimentação elétrica retornou novamente.

ctrlaltdel O processo será executado quando o **init** receber o sinal SIGINT. Isso significa que alguém na console pressionou a combinação de teclas **CTRL-ALT-DEL** . Tipicamente alguém deseja desligar o sistema ou executar algum tipo de **shutdown** para entrar em modo monousuário ou para reinicializar a máquina.

kbrequest O processo será executado quando **init** receber um sinal do gerenciador de teclado de que uma combinação especial de teclas foi pressionada.

A documentação desta função não está completa ainda; mais informações podem ser encontradas nos pacotes kbd-x.xx. Basicamente deve-se mapear uma combinação de teclado que execute uma ação de "sinal de Teclado". Por exemplo para mapear a tecla seta para cima para este propósito, use o seguinte em seus arquivos de mapeamento de teclado:

alt keycode 103 = KeyboardSignal

EXEMPLOS

Este é um exemplo de um inittab que lembra a inittab antiga do Linux:

inittab para Linux

id:1:initdefault:

rc::bootwait:/etc/rc

1:1:respawn:/etc/getty 9600 tty1

2:1:respawn:/etc/getty 9600 tty2

3:1:respawn:/etc/getty 9600 tty3

4:1:respawn:/etc/getty 9600 tty4

O arquivo inittab executa **/etc/rc** durante a inicialização do sistema e inicia os gettys em tty1-tty4.

Um **inittab** mais elaborado com diferentes níveis de execução (veja os comentários):

Nível de execução inicial

id:2:initdefault:

Inicialização do sistema antes de qualquer coisa.

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

Nível de execução 0,6 é desligar e reiniciar, 1 é o

modo de manutenção.

l0:0:wait:/etc/rc.d/rc.halt

l1:1:wait:/etc/rc.d/rc.single

l2:2345:wait:/etc/rc.d/rc.multi

l6:6:wait:/etc/rc.d/rc.reboot

#O que fazer com a "saudação dos 3 dedos".

ca::ctrlaltdel:/sbin/shutdown -t5 -rf now

Níveis 2&3: getty na console, nível 3 getty também na

porta do modem.

1:23:respawn:/sbin/getty tty1 VC linux

2:23:respawn:/sbin/getty tty2 VC linux

3:23:respawn:/sbin/getty tty3 VC linux

4:23:respawn:/sbin/getty tty4 VC linux

S2:3:respawn:/sbin/uugetty ttyS2 M19200

ARQUIVOS

/etc/inittab

AUTOR

Init foi escrito por Miquel van Smoorenburg (miquels@cistron.nl). Esta página de manual foi escrita por Sebastian Lederer (lederer@francium.informatik.uni-bonn.de) e modificada por Michael Haardt (u31b3hs@pool.informatik.rwth-aachen.de).

VEJA AINDA

init(8), **telinit(8)**

E.15 kill

NOME

kill - finaliza um processo

SINOPSE

kill [-s sinal | -p] **[-a]** pid ...

kill -l [sinal]

DESCRIÇÃO

kill envia um sinal específico para um determinado processo. Caso nenhum sinal seja especificado, o sinal TERM é enviado. Este sinal irá finalizar processos que esperam este tipo de mensagem. Para outros processos, pode ser necessário usar o sinal KILL (9), uma vez que este sinal não pode ser ignorado.

Interpretadores de comandos modernos tem uma função kill pré-construída.

OPÇÕES

pid ... Especifica a lista de processos para os quais **kill** deve sinalizar. Cada *pid* pode ser um entre quatro opções. Um *nome de processo* no qual o processo nomeado receberá o sinal. *n* onde *n* é maior que 0. O processo com o pid (número de identificação) *n* receberá o sinal. *-1* onde todos os processos de MAX_INT a 2 receberão o sinal, se permitido pelo dono do processo. *-n* onde *n* é maior que 1, e todos os processos do grupo *n* receberão o sinal. Caso um sinal negativo seja informado, o sinal *obrigatoriamente* deve ser especificado antes, de outra forma será interpretado como o sinal a ser enviado.

-s Especifica o sinal a ser enviado. O sinal pode ser informado como um dígito ou como um número.

-p Especifica que **kill** pode somente listar a identificação do processo *(pid)* do processo nomeado, e não deve enviar-lhe um sinal.

-l Lista uma relação dos nomes de sinais. Eles podem ser encontrados em */usr/include/linux/signal.h*

VEJA TAMBÉM

bash(1), **tcsh**(1), **kill**(2), **sigvec**(2)

AUTOR

Incorporado a partir do BSD 4.4. A habilidade de converter nomes de processos para suas identificações foi desenvolvida por Salvatore Valente <svalente@mit.edu>.

E.16 lilo.conf

NOME

lilo.conf - arquivo de configuração do lilo

DESCRIÇÃO

Este arquivo, por padrão */etc/lilo.conf*, é lido pelo carregador de inicialização lilo (veja lilo(8)).

Ele pode se parecer como:

```
boot = /dev/hda
delay = 40
compact
vga = normal
root = /dev/hda1
read-only

image = /zImage-1.5.99
    label = try

image = /zImage-1.0.9
    label = 1.0.9

image = /tamu/vmlinuz
  label = tamu
  root = /dev/hdb2
  vga = ask

other = /dev/hda3
  label = dos
  table = /dev/hda
```

Este arquivo de configuração especifica que o lilo usa o Registro Mestre de Inicialização - MBR em /dev/hda. (para uma discussão sobre as várias formas de usar o lilo, e a interação com outros sistemas operacionais veja user.tex na documentação do lilo).

Ao inicializar, o carregador irá aguardar por quatro segundos (40 décimos de segundos) para que [shift] seja pressionado. Caso isso não ocorra, o primeiro kernel mencionado (/zImage-1.5.99, o qual provavelmente foi instalado há pouco) será iniciado. Caso [shift] seja pressionado, o carregador perguntará qual imagem deverá ser usada. Para verificar as alternativas, basta pressionar [TAB] (ou [?], caso se tenha um teclado padrão US), e será apresentada uma lista das imagens disponíveis. Tem-se então a alternativa de carregar um kernel novo, ou um antigo e confiável, ou um de outro sistema de arquivos (caso algo tenha ocorrido com o sistema de arquivos raiz usual). Podem haver até 16 imagens mencionadas em lilo.conf.

Como se pode ver acima, um arquivo de configuração começa com um número de opções globais (as primeiras 6 linhas no exemplo), seguidas das descrições das opções das várias imagens. Uma opção em uma descrição de uma imagem irá se sobrepor à uma definição global.

OPÇÕES GLOBAIS

Há diversas opções possíveis. As apresentadas a seguir são literalmente as mesmas listadas em user.tex (ligeiramente abreviadas).

backup=*arquivo* Copiar o setor de inicialização original para *arquivo* (o qual pode também ser um dispositivo, por exemplo */dev/null*) ao invés de */boot/boot.NNNN.*

boot=*dispositivo* Define o nome do dispositivo (por exemplo uma partição de disco rígido) que contenha um setor de inicialização. Caso a palavra chave seja omitida, o setor de inicialização é lido do (e provavelmente gravado em) dispositivo montado como raiz.

compact Tenta combinar requisições de leitura de setores adjacentes em uma única solicitação de leitura. Isso reduz drasticamente o tempo de carga e mantém o mapa menor. Usar esta opção é bastante recomendável especialmente quando se inicializa a partir de um disquete.

default=*nome* Usa a imagem especificada como a imagem de inicialização padrão. Caso 'default' seja omitido, a imagem listada em primeiro lugar no arquivo de configuração é usada.

delay=*tsecs* Especifica o número de décimos de segundo que o carregador deve esperar antes de carregar a primeira imagem. Isso é útil em sistemas que inicializam imediatamente de um disco rígido após a habilitação do teclado. O carregador não tem que esperar pelo período de tempo caso 'delay' seja omitido ou configurado como zero.

disk=*dispositivo* Define parâmetros não padronizados para o disco especificado. Veja a seção "Geometria de Disco " em user.tex para detalhes.

disktab=*disktab-arq* Especifica o nome da tabela de parâmetros de disco. O instalador do mapa procura por */etc/disktab* caso 'disktab' seja omitido. O uso deste parâmetro é desencorajado.

fix-table Permite ao lilo ajustar os endereços nas tabelas de partições. Cada partição contém um endereço constituído por setor/cabeça/cilindro e um linear do primeiro e do último setor da partição. Caso a partição não esteja alinhada com as trilhas e se certos sistemas operacionais estão usando o mesmo disco (PC/MS-DOS ou OS/2), eles podem ter alterado o endereço. O lilo pode guardar seu setor de inicialização somente em partições onde ambos endereços correspondam. O lilo ajustará os endereços caso esta opção esteja configurada.

ATENÇÃO: isso não garante que outros sistemas operacionais tentem mudar os endereços posteriormente. É possível ainda que esta opção cause efeitos colaterais imprevisíveis. A forma correta de ajustar este problema é reparticionar o disco com um programa que faça o alinhamento. Ainda em alguns discos (grandes discos EIDE com conversão de endereços habilitada), sob certas circunstâncias, pode ser inevitável ter certos conflitos na tabela de partições.

force-backup=*arquivo* Assim como em 'backup', regrava uma cópia antiga de arquivo caso exista.

ignore-table Diz ao lilo para ignorar tabelas com partições corrompidas.

install=*setor-inic* Instala o arquivo especificado no novo setor de inicialização. Caso 'install' seja omitida, */boot/boot.b* é usado como padrão.

linear Gera endereços lineares ao invés de endereços no formato setor/cabeça/cilindro. Endereços lineares são convertidos em tempo de execução e não dependem da geometria do disco. Note que os discos de inicialização podem não ser portáveis caso a opção 'linear' seja usada, porque os serviços do BIOS que determinam a geometria do disco não funcionam adequadamente em disquetes. Ao usar 'linear' com discos grandes, */sbin/lilo* pode gerar referências a área inacessíveis do disco, uma vez que os endereços setor/cabeça/cilindro não são conhecidas antes da inicialização do sistema.

lock Habilita a gravação das linhas de comandos de inicialização como padrão das próximas inicializações do sistema. Desta forma, o lilo guarda as opções até que sejam manualmente sobrepostas.

map=*arquivo-mapa* Especifica a localização do arquivo mapa. Caso 'map' seja omitido, o arquivo */boot/map* será usado.

message=*arquivo* Especifica um arquivo contendo uma mensagem a ser apresentada antes da linha de inicialização. Nenhuma mensagem é apresentada enquanto o programa estiver esperando pela tecla [`shift`] após ser apresentada a mensagem LILO. Na mensagem o caracter FF([Ctrl L]) limpa a tela. O tamanho do arquivo de mensagem está limitado a 65535 bytes. O arquivo map tem que ser reconstruído se o arquivo de mensagem for alterado ou movimentado para outra localização.

nowarn Desabilita avisos sobre possíveis perigos futuros.

optional Um parâmetro 'optional' (veja abaixo) pode ser aplicado para cada uma das imagens.

password=*senha* Uma opção 'password=...' (veja abaixo) pode ser aplicado para cada uma das imagens.

prompt Força a apresentação da mensagem de acesso ao sistema, e fica aguardando que alguma tecla seja pressionada. Reinicializações desatendidas são impossíveis caso 'prompt' esteja configurado e 'timeout' não.

restricted Uma opção 'restricted' (veja abaixo) pode ser aplicado para cada uma das imagens.

serial=*parâmetros* Habilita o controle a partir de uma linha serial. A porta serial especificada é inicializada e o carregador de inicialização aceita entradas a partir daí e do teclado. Enviando um caracter de quebra em uma linha serial corresponde a pressionar a tecla [`shift`] no console, para se obter a atenção do carregador. Todas as imagens de inicialização devem ser protegidas por senha caso a linha serial seja menos segura que o console, por exemplo se a linha está conectada a um modem. Os parâmetros tem a seguinte sintaxe:

<porta>[,<bps>[<paridade>[<bits>]]]

<porta>: o número da porta serial. 0 corresponde a COM1 ou /dev/ttyS0, etc. Todas as quatro portas podem ser usadas (caso presentes).

<bps>: taxa de transferência, em baud, da porta serial. As seguintes velocidades são suportadas: 110, 150, 300, 600, 1200, 2400, 4800 e 9600 bps. Padrão é 2400 bps.

<paridade>: paridade usada na linha serial. O carregador ignora as paridades de entrada e ignora pelo oitavo bit. Os seguintes (maiúscula ou minúscula) caracteres são usados para descrever a paridade: n para nenhuma, "e" para par e "o" para paridade ímpar.

<bits>: o número de bits de um caracter. Somente 7 e 8 bits são suportados. O padrão é 8 bits para paridade nenhuma, 7 se a paridade é par ou ímpar.

Caso serial esteja configurado, o valor de 'delay' é automaticamente elevado para 20.

Exemplo: serial=0,2400n8 inicializa COM1 com os parâmetros padrões.

timeout=*tsecs* Configura o tempo de espera (em décimos de segundos) para entrada de teclado. Caso nenhuma tecla seja pressionada no tempo especificado, a primeira imagem é automaticamente carregada. Similarmente, a entrada da senha é abortada se o usuário levar muito tempo. O padrão de espera é infinito.

verbose=*nível* Ativa a geração de relatórios de progresso. Números maiores fornecem mais mensagens de saída. Caso -v seja adicionalmente especificada na linha de comando do lilo, o nível é incrementado de acordo. O nível máximo é igual a 5.

Adicionalmente, os parâmetros de configuração **append**, **ramdisk**, **read-only**, **read-write**, **root** e **vga** podem ser configurados na seção de opções globais. Estas são usadas como padrões, caso não sejam especificadas na seção de configuração das respectivas imagens do kernel.

SEÇÃO POR IMAGEM

Uma seção por imagem começa ou com uma linha

image= *caminho*

(para indicar o arquivo ou dispositivo contendo a imagem de inicialização do kernel do Linux), ou a linha

other= *caminho*

para indicar um sistema arbitrário a ser iniciado.

Caso uma linha **image** especifique a inicialização a partir de um dispositivo, então há que se indicar a faixa de setores a serem mapeados usando-se

range=*início-fim*

No último caso (carregando outro sistema) há três opções:

loader=*chain-loader* Especifica o carregador a ser usado. Por padrão, */boot/chain.b* é usado. O carregador deve ser especificado caso a inicialização de um dispositivo seja outro que não o primeiro disco ou disquete.

table=*dispositivo* Isso especifica o dispositivo que contém a tabela de partição. O carregador de inicialização não passará informações da partição para o sistema operacional caso esta variável seja omitida. (alguns sistemas operacionais têm outras formas de determinar de qual partição eles foram carregados. Por exemplo MS-DOS usualmente armazena a geometria do disco de inicialização ou partição no seu setor de inicialização). Note que /sbin/lilo deve ser reexecutado se a tabela de partições mapeadas for modificada.

unsafe Não acessar o setor de inicialização em tempo de criação do mapa. Isso desabilita algumas checagens, incluindo a checagem da tabela de partição. Caso o setor de inicialização esteja em um disquete de formato fixo, usando UNSAFE evita a necessidade de por-se um disco no drive ao executar o mapa instalador. 'unsafe' e 'table' são mutuamente incompatíveis.

Em ambos os casos as seguintes opções são aplicadas:

label=*nome* O carregador usa o nome do arquivo sem seu caminho, em cada imagem especificada, para identificá-lo. Um nome diferente pode ser usado para configurar a variável 'label'.

alias=*nome* Um segundo nome para a mesma entrada pode ser usado, criando-se um nome alternativo.

lock (Veja acima.)

optional Omite a imagem, caso não esteja disponível em tempo de criação. Isso é útil para especificar os testes de kernel que nem sempre estão presentes.

password=*senha* Protege a imagem através de senha.

restricted Uma senha é requerida somente para inicializar a imagem caso os parâmetros sejam especificados na linha de comando (por ex. single).

OPÇÕES DO NÚCLEO

Caso a imagem carregada seja um kernel do Linux, então podem ser passados parâmetros na linha de comando para este kernel.

append=*string* Anexa as opções especificadas à linha de comando passadas para o kernel. Isto é tipicamente usado para especificar parâmetros de hardware que não podem ser detectados automaticamente ou cujos testes podem ser perigosos. Por exemplo:

append = "hd=64,32,202"

literal=*string* Como ‘append’, porém remove todas as outras opções (por exemplo, configurando o dispositivo raiz). Uma vez que opções vitais podem ser removidas inadvertidamente com ‘literal’, esta opção não deve ser configurada na seção de opções globais.

ramdisk=*tamanho* Especifica o tamanho do disco de RAM opcional. Um valor igual a zero indica que nenhum disco em memória deve ser usado. Caso esta variável seja omitida, um tamanho para disco em memória configurado na imagem de inicialização será usado.

read-only Especifica que o sistema de arquivos raiz deve ser montado com permissões somente de leitura. Tipicamente o procedimento de início do sistema remonta o sistema raiz com permissões de leitura e gravação posteriormente (por exemplo após executar o comando fsck).

read-write Especifica que o sistema de arquivos raiz deve ser montado com permissões de leitura e gravação.

root=*dispositivo* Isso especifica o dispositivo que deve ser montado como raiz. Caso o nome especial **current** seja usado, o dispositivo raiz é configurado para o dispositivo no qual o sistema de arquivos raiz está atualmente montado. Caso o raiz tenha sido alterado com -r o respectivo dispositivo é usado. Caso a variável ‘root’ seja omitida, o dispositivo raiz configurado na imagem do kernel será utilizado. (Pode ser definido em tempo de compilação, através da variável ROOT_DEV e pode ser posteriormente alterada com o programa rdev(8).)

vga=*modo* Especifica que o modo VGA texto deve ser usado na inicialização do sistema. Os seguintes valores são reconhecidos (não há diferenças entre maiúsculas e minúsculas):

normal: seleciona modo texto normal 80x25.

extended (ou **ext**): seleciona modo texto 80x50.

ask: para e pergunta ao usuário (em tempo de inicialização).

<number>: usa o modo texto correspondente. Uma lista dos modos disponíveis pode ser obtida iniciando-se o sistema com vga=ask e pressionando [Enter].

Caso esta variável seja omitida, a configuração VGA contida na imagem do kernel é usada. (Pode ser definido em tempo de compilação, através da variável SVGA_MODE e posteriormente alterada com o programa rdev(8)).

VEJA TAMBÉM

lilo(8), rdev(8).

A distribuição do lilo é acompanhada por uma documentação extensa da qual os dados acima foram retirados.

E.17 lilo

NOME

lilo - instala o carregador de inicialização

SINOPSE

Função principal:

/sbin/lilo - instala o carregador de inicialização

Usos auxiliares:

/sbin/lilo -q - mapa de pesquisa

/sbin/lilo -R - define a nova linha de comando a ser usada na próxima reinicialização

/sbin/lilo -I - solicita o caminho do kernel atual

/sbin/lilo {-u|-U} - desinstala lilo

DESCRIÇÃO

lilo instala um carregador de inicialização que será ativado da próxima vez que o sistema for iniciado. Ele tem diversas opções:

-v Aumenta o número de mensagens. Fornecendo-se uma ou mais opções **-v** fará com que o lilo apresente maior número de mensagens.

-q Lista os arquivos atualmente mapeados. **lilo** mantém um arquivo, por padrão */boot/map*, contendo o nome e a localização do kernel para inicialização. Esta opção irá listar seus nomes.

-m *mapa* Usa um arquivo específico de mapa de arquivos ao invés do padrão.

-C *arquivo de configuração* **lilo** lê as instruções sobre quais arquivos mapear a partir do arquivo de configuração, por padrão */etc/lilo.conf*. Esta opção pode ser usada para especificar um arquivo de configuração diferente do padrão.

-d *retardo* Caso se tenha diversos kernels especificados, e pressionando-se shift durante a inicialização do sistema, o carregador irá apresentar uma lista de opções de qual kernel ou sistema operacional deve ser usado. Após um período de espera o primeiro kernel da lista será usado. Esta opção especifica o tempo de espera em décimos de segundos.

-D *rótulo* Usa o kernel informado na etiqueta especificada, ao invés do primeiro da lista, como o kernel padrão para iniciar.

-r *diretório-raiz* Antes de qualquer coisa, executar o comando *chroot* para o diretório indicado. Usado para reparar uma configuração a partir de um disquete de inicialização.

-t Teste somente. Não grava realmente um novo setor no arquivo de mapas. Usado com **-v** ajuda a descobrir o que **lilo** irá fazer.

-c Habilita a compactação do mapa. Isso irá misturá-lo com as requisições de leitura de setores adjacentes. Aumenta a velocidade de inicialização, especialmente a partir de disquetes.

-f *arquivo* Especifica o arquivo de geometria do disco (o padrão é */etc/disktab*.)

-i *setor-de-inicialização* Especifica um arquivo a ser usado como o novo setor de inicialização (o padrão é */boot/boot.b*).

-l Gera endereços de setor lineares ao invés de usar endereços setor/cabeça/cilindro.

-P *{fix|ignore}* Corrige (ou ignora) tabelas de partições corrompidas, isto é, tabelas de partições com endereços linear e setor/cabeça/cilindro que não correspondam.

-s *arquivo-a-salvar* Quando **lilo** regrava um setor de inicialização, ele preserva o conteúdo antigo em um arquivo, por padrão */boot/boot.NNNN* onde NNNN depende do dispositivo. Esta opção especifica um arquivo alternativo para o setor de inicialização (ou junto com a opção **-u**, especifica de onde restaurar o setor).

-S *arquivo-a-salvar* Normalmente, o **lilo** não regravará em um arquivo já existente. Esta opção diz que a regravação está autorizada.

-u *nome-dispositivo* Desinstala o **lilo**, copiando o setor de inicialização salvo de volta. A data do arquivo é verificada.

-U *nome-dispositivo* Idem, mas não verifica a data.

-R *linha de comando* Esta opção configura o comando padrão do carregador na próxima vez em que ele for executado. O lilo irá apagar esta linha, ou seja este comando é executado somente uma única vez. É tipicamente usado em programas de reinicialização do sistema, imediatamente antes de executar 'shutdown -r'.

-I *rótulo* A identificação do kernel em execução pode ser encontrado na variável de ambiente BOOT_IMAGE após o início do sistema. Este comando irá imprimir o nome do caminho correspondente na saída padrão.

-V Lista a versão do programa.

As opções de linha de comando correspondem a palavras chaves no arquivo de configuração listadas a seguir.

```
-b bootdev   boot=bootdev
-c   compact
-d dsec   delay=dsec
-D label  default=label
-i bootsector install=bootsector
-f file   disktab=file
-l   linear
-m mapfile   map=mapfile
-P fix   fix-table
-P ignore  ignore-table
-s file   backup=file
-S file   force-backup=file
-v   verbose=level
```

VEJA TAMBÉM

lilo.conf(5).

A distribuição do lilo vem acompanhada de uma extensa documentação.

AUTOR

Werner Almesberger (almesber@bernina.ethz.ch).

E.18 login

NOME

login - acesso ao sistema

SINOPSE

login [nome **]**

login -p

login -h nome da máquina

login -f nome

DESCRIÇÃO

login é usado quando se obtém acesso ao sistema. Ele pode ser usado para alterar de um para outro usuário a qualquer tempo (muitos shells modernos têm suporte a esta funcionalidade pré-construída).

Caso nenhum argumento seja passado ele solicita o nome do usuário.

Caso o usuário *não* seja o superusuário e se o */etc/nologin* existir, o conteúdo deste arquivo será listado na tela e o login será terminado. Isso é tipicamente usado para prevenir acessos quando o sistema está sendo desligado.

Caso restrições especiais de acesso sejam especificadas para o usuário em */etc/usertty*, estas devem ser atendidas ou a tentativa de acesso será negada e um registro será incluído no **syslog**. Veja a seção "Restrições Especiais de Acesso".

Caso o usuário seja o superusuário, então o acesso deve ocorrer em um terminal listado no */etc/securetty*. Falhas serão listadas através do **syslog**.

Após a checagem destas condições, a senha será solicitada e checada (caso uma senha seja exigida para este usuário). Dez tentativas serão permitidas antes que **login** morra, porém após as primeiras três, a resposta começa a ficar bastante lenta. Falhas de acesso são reportadas através do **syslog**. Esta facilidade é usada ainda para reportar acessos do superusuário bem sucedidos.

Caso o arquivo *.hushlogin* exista, então um acesso sem mensagens é executado (isso desabilita a checagem de emails e a apresentação da mensagem do último acesso e a mensagem do dia). De outra forma, caso */var/log/lastlog* exista, o último acesso é listado (e o acesso atual é registrado).

Tarefas administrativas aleatórias, como a configuração de UID e GID do terminal são executadas. A variável de ambiente TERM é preservada, caso exista (outras variáveis de ambiente são preservadas se a opção **-p** é usada). Então as variáveis de ambiente HOME, PATH, SHELL, TERM, MAIL, e LOGNAME são configuradas. PATH padrões como */usr/local/bin:/bin:/usr/bin:.* para usuários normais, e

/sbin:/bin:/usr/sbin:/usr/bin para o root. Por fim, se não é um acesso "quieto", a mensagem do dia é listada e um arquivo com o nome do usuário em */var/spool/mail* será checado, e uma mensagem será apresentada caso o arquivo tenha um tamanho diferente de zeros.

Então o interpretador de comandos do usuário é iniciado. Caso nenhum interpretador tenha sido especificado para o usuário no **/etc/passwd**, então **/bin/sh** será usado. Caso não haja nenhum diretório especificado no */etc/passwd*, então o / é usado (o diretório pessoal é checado para o arquivo *.hushlogin,* descrito acima).

OPÇÕES

-p Usado por **getty(8)** para dizer ao **login** não destruir o ambiente.

-f Usado para evitar uma segunda autenticação de acesso. Isso **não** funciona para o superusuário, e não parece funcionar muito bem sob Linux.

-h Usado por outros servidores (por exemplo **telnetd(8)**) para passar o nome para o servidor remoto para **acessos** que podem ser colocados em utmp e wtmp. Somente o superusuário pode usar esta opção.

RESTRIÇÕES ESPECIAIS DE ACESSO

O arquivo */etc/securetty* lista os nomes dos ttys onde o superusuário tem permissão de acesso. Um nome de um dispositivo terminal sem o prefixo /dev/ deve ser especificado em cada linha. Caso o arquivo não exista, o superusuário não poderá acessar o sistema em qualquer terminal.

O arquivo */etc/usertty* especifica restrições de acessos adicionais para usuários específicos. Caso este arquivo não exista, nenhuma restrição adicional será aplicada. O arquivo consiste de uma seqüência de seções, com três tipos possíveis: CLASSES, GRUPOS e USUÁRIOS. Uma seção de CLASSES define classes de padrões de tty e nomes de máquinas. Uma seção GRUPO define ttys e nomes de máquinas permitidas em grupos e uma seção de USUÁRIOS define ttys e nomes de máquinas para usuários.

Cada linha neste arquivo não pode ser maior que 255 caracteres. Comentários começam com o caracter # e se estendem até o final da linha.

Seção CLASSES

Uma seção de CLASSE começa com a palavra CLASSES no início da linha em maiúsculas. Cada linha seguinte até o início de uma nova seção ou o fim do arquivo consistem de seqüências de palavras separadas por tabulações ou espaços. Cada linha define uma classe de ttys e nomes de máquinas.

A palavra no início da linha torna-se uma definição de um nome coletivo para os ttys e nomes de máquinas especificados no restante da linha. O nome coletivo pode ser usado em quaisquer seções subseqüentes da seção GRUPOS ou USUÁRIOS. Nenhuma classe pode ocorrer como uma parte da definição de outra classe para evitar problemas com classes recursivas.

Um exemplo da CLASSES:

CLASSES

classe1 tty1 tty2

classe2 tty3 @.dominio.com.br

Isso define as classes *classe1* e *classe2* e seus correspondentes lados direitos.

Seção GRUPOS

Uma seção GRUPOS define os ttys e máquinas autorizados em uma base de grupos. Caso o usuário seja membro de um grupo Unix de acordo com o */etc/passwd* e */etc/group* e tal grupo seja mencionado na seção */etc/usertty* então o usuário obterá os mesmos acessos que o seu grupo.

Uma seção de GRUPOS inicia com a palavra GROUPS em maiúsculas no início da linha, e cada linha subseqüente é uma seqüência de palavras separadas por espaços ou tabulações. A primeira palavra na linha é o nome do grupo e o restante da linha especifica os ttys e nomes de máquinas onde os membros do grupo têm o acesso permitido. Estas especificações podem envolver o uso de classes definidas em seções de CLASSES anteriores.

Exemplo.

GROUPS

sys tty1 @.dominio.edu.br

stud classe1 tty4

Este exemplo especifica que os membros do grupo *sys* podem acessar o sistema no tty1 e a partir das máquinas do domínio dominio.edu.br. Usuários do grupo *stud* podem acessar o sistema a partir de máquinas/ttys definidos na classe1 ou a partir da tty4.

Seção USUÁRIOS

Uma seção USUÁRIOS começa com a palavra USERS em maiúsculas no início da linha, e cada linha subseqüente é uma seqüência de palavras separadas por espaços ou tabulações. A primeira palavra na linha é o nome do usuário e o restante da linha especifica as ttys e nomes de máquinas onde os membros do grupo têm o acesso permitido. Estas especificações podem envolver o uso de classes definidas em seções de CLASSES anteriores. Caso nenhum cabeçalho seja especificado no início do arquivo, a primeira seção padroniza a seção USERS.

Um exemplo da seção USERS:

USERS

kiyumi tty1 @130.225.16.0/255.255.255.0

paulo tty3 classe2

Isso permite que o usuário kiyumi somente acesse o sistema a partir do terminal tty1 e a partir dos endereços IP na faixa de 130.225.16.0 - 130.225.16.255, e o usuário paulo tem permissão de acesso a partir de tty3 e tudo o mais que estiver especificado em classe2.

Haverá uma linha na seção USUÁRIOS começando com um nome de usuário com *. Esta é a regra padrão e será aplicada a qualquer usuário que não se enquadre em outra linha.

Caso um usuário se enquadre tanto em USERS e GROUPS, então o usuário terá permissões de acesso iguais à união de ttys e máquinas mencionados nestas especificações.

Origens

Os padrões de tty e nome da máquina usados na especificação das classes, grupo e usuários são chamados origens. Uma origem pode conter os seguintes formatos:

O nome de um dispositivo tty sem o prefixo /dev/, por exemplo tty1 ou ttyS0.

O padrão @nome_da_máquina_local, significando que o usuário tem permissão de executar telnet/rlogin de uma máquina local para a mesma máquina. Isso permite ao usuário executar por exemplo: xterm -e /bin/login.

Um sufixo de nome de domínio tal como @.algum.dom.br, significando que o usuário tem permissão de executar telnet/rlogin de qualquer máquina do domínio de sufixo àlgum.dom.br

Uma faixa de endereços Ipv4, escritos no formato @x.x.x.x/y.y.y.y onde x.x.x.x é o endereço IP na notação decimal, quádrupla, separada por pontos, e y.y.y.y é uma máscara de bits com a mesma notação especificando quais bits do endereço devem ser comparados com o endereço IP da máquina remota. Por exemplo @130.225.16.0/255.255.254.0 significa que o usuário pode executar rlogin/telnet de qualquer máquina cujo IP tenha endereço na faixa 130.225.16.0-130.225.17.255.

Quaisquer das origens acima podem ter um prefixo com uma especificação de tempos como por exemplo:

timespec ::= '[' <dia-ou-hora> [':' < dia-ou-hora >]* ']'

day ::= 'mon' | 'tue' | 'wed' | 'thu' | 'fri' | 'sat' | 'sun'

hour ::= '0' | '1' | ... | '23'

hourspec ::= <hora> | <hora> '-' <hora>

day-or-hour ::= <dia> | <hora_esper>

Por exemplo, a origem [mon:tue:wed:thu:fri:8-17]tty3 significa que o acesso é permitido de Segunda a Sexta-feira entre 8:00 e 17:59 no tty3. Isso também mostra uma faixa de horário a-b que inclui todos os momentos entre a:00 e b:59. Uma especificação única de hora (como 10) significa o tempo decorrido entre 10:00 e 10:59.

Nenhuma especificação de prefixo de tempo para um tty ou máquina significa que o acesso daquela origem é permitida a qualquer tempo. Caso seja informado um prefixo de tempo esteja seguro de especificar um conjunto de dias e uma ou mais faixas de horas. Uma especificação não pode incluir qualquer espaço em branco.

Caso nenhuma regra padrão seja informada então usuários que não coincidam com qualquer linha */etc/usertty* têm permissão de acesso de qualquer local no comportamento padrão.

ARQUIVOS

/var/run/utmp /var/log/wtmp /var/log/lastlog /usr/spool/mail/ /etc/motd /etc/passwd /etc/nologin /etc/usertty .hushlogin*

VEJA TAMBÉM

init(8), **getty**(8), **mail**(1), **passwd**(1), **passwd**(5), **environ**(7), **shutdown**(8)

BUGS

A opção não documentada de BSD **-r** não é suportada. Isso pode ser requerido por alguns programas **rlogind(8).**

AUTOR

Derivado do login BSD 5.40 (5/9/89) por Michael Glad (glad@daimi.dk) do HP-UX

Portado para o Linux 0.12: Peter Orbaek (poe@daimi.aau.dk)

E.19 mke2fs

NOME

mke2fs - cria um sistema de arquivos do tipo Linux second extended

SINOPSE

mke2fs [**-c** | **-l** *nome-do-arquivo*] [**-b** *tamanho-dos-blocos*] [**-f** *tamanho-dos-fragmentos*] [**-i** *bytes-por-inode*] [**-m** *porcentagem-de-blocos-reservados*] [**-o** *so-do-criador*] [**-q**] [**-r revisão**] [**-R** *opções-de-raid*] [**-s** *marca-super-esparso*] [**-v**] [**-F**] [**-L** *rótulo-do-volume*] [**-M** *diretório-da-última-montagem*] [**-S**] [**-V**] *dispositivo* [*número-de-blocos*]

DESCRIÇÃO

mke2fs é usado para criar um sistema de arquivos do tipo Linux second extended em um dispositivo (normalmente uma partição de disco).

dispositivo é o arquivo especial correspondente ao dispositivo (ex. /dev/hdXX).

número-de-blocos é o número de blocos no dispositivo. Se omitido, **mke2fs** automaticamente descobre o tamanho do sistema de arquivos.

OPÇÕES

-b tamanho-dos-blocos Especifica o tamanho dos blocos em bytes.

-c Verifica se o dispositivo tem blocos ruins antes de criar o sistema de arquivos, usando um teste rápido somente de leitura.

-f tamanho-dos-fragmentos Especifica o tamanho dos fragmentos em bytes.

-i bytes-por-inode Especifica a relação bytes/inode. **mke2fs** cria um inode para cada *bytes-por-inode* bytes de espaço no disco. Este valor é predefinido como 4096 bytes. *bytes-por-inode* deve ser sempre um múltiplo de 1024 e não menos que isso.

-l nome-do-arquivo Lê a lista de blocos ruins a partir de *nome-do-arquivo*.

-m porcentagem-de-blocos-reservados Especifica a porcentagem de blocos reservados para o superusuário. Este valor é predefinido como 5%.

-o Manualmente substitui o valor predefinido do campo "so do criador" do sistema de arquivos. Normalmente o campo so do criador é ajustado para um valor predefinido igual ao sistema operacional nativo do executável do **mke2fs**.

-q Execução silenciosa. Útil se **mke2fs** for executado em um script.

-s marca-super-esparso Se *marca-super-esparso* for 1, então a marca "superbloco esparso" é ativada. Se 0, então a marca "superbloco esparso" é desativada. (Atualmente, a marca "superbloco esparso" é predefinida como desativada). **Aviso:** O kernel 2.0 do Linux não suporta corretamente esta característica, nem todos os kernels 2.1 do Linux; por favor não use isto a não ser que você saiba o que está fazendo!

-v Execução detalhada.

-F Força **mke2fs** a ser executado, mesmo que o dispositivo especificado não seja um dispositivo especial de bloco.

-L Define o rótulo do volume para o sistema de arquivos.

-M Define o último ponto de montagem para o sistema de arquivos. Isto pode ser útil para utilidades que usam o último ponto de montagem para determinar onde o sistema de arquivos deve ser montado.

-r revisão Define a revisão do sistema de arquivos para o novo sistema de arquivos. Note que kernels 1.2 suportam apenas sistemas de arquivos de revisão 0.

-R opções-de-raid Ajusta as opções relacionadas com raid no sistema de arquivos. Opções de raid são normalmente separadas, e podem receber parâmetros usando o sinal de igual ('='). Atualmente a única opção suportada é *stride* que recebe como seu parâmetro o número de blocos numa faixa RAID.

-S Grava apenas os descritores de superbloco e grupo. Isto é útil se todo o superbloco e cópias de segurança do superbloco estão danificadas, e um último método de recuperação é desejado. Isto faz com que **mke2fs** reinicie o superbloco e os descritores de grupo, sem tocar na tabela de inode e nos mapas de bit de blocos e inodes. O programa **e2fsck** deve ser executado imediatamente após o uso desta opção, e não há garantia que quaisquer dados serão recuperados.

-V Exibe a versão de **mke2fs** e encerra a execução.

AUTOR

Esta versão de **mke2fs** foi escrita por Theodore Ts'o <tytso@mit.edu>.

FALHAS

mke2fs aceita a opção -f mas atualmente a ignora porque o sistema de arquivos second extended ainda não suporta fragmentos.

Podem haver outros problemas. Por favor, relate-os ao autor.

DISPONIBILIDADE

mke2fs está disponível através de ftp anônimo em ftp.ibp.fr e em tsx-11.mit.edu no diretório /pub/linux/packages/ext2fs.

VEJA TAMBÉM

dumpe2fs(8), **e2fsck**(8), **tune2fs**(8)

E.20 mkswap

NOME

mkswap - configura uma área de troca (swap) do Linux

SINOPSE

mkswap [-c]

DESCRIÇÃO

mkswap configura uma área de troca em um dispositivo ou em um arquivo.

O *dispositivo* tem normalmente o seguinte formato:

/dev/hda[1-8]

/dev/hdb[1-8]

/dev/sda[1-8]

/dev/sdb[1-8]

O *tamanho_em_blocos* é o tamanho desejado para o sistema de arquivos, em blocos. Esta informação é determinada automaticamente por **mkswap** caso não seja declarada. Contadores de blocos são arredondados para baixo, então o tamanho total é um inteiro, múltiplo do tamanho de página da máquina. Somente um valor na faixa MINCOUNT..MAXCOUNT são permitidos. Caso o número exceda o campo MAXCOUNT, ele é truncado para aquele valor e uma mensagem de aviso será gerada.

Os valores MINCOUNT e MAXCOUNT para uma área de troca são:

MINCOUNT = 10 * TAMANHO_PÁGINA / 1024

MAXCOUNT = (TAMANHO_PÁGINA - 10) * 8 * TAMANHO_PÁGINA / 1024

Por exemplo, em uma máquina com 4 Kbytes (em sistemas x86), temos:

MINCOUNT = 10 * 4096 / 1024 = 40

MAXCOUNT = (4096 - 10) * 8 * 4096 / 1024 = 130752

Como cada bloco tem o tamanho de 1 Kb, a área de troca neste exemplo pode ter um tamanho qualquer entre 40kB até 127.6875MB.

Caso não se saiba o tamanho da página da máquina, pode-se obter esta informação através do comando "cat /proc/cpuinfo".

A razão para o limite em MAXCOUNT reside no fato de que uma única página é usada para manter o mapa de bits no início da área de troca, onde cada bit representa uma única página. A razão para a subtração de 10, é o tamanho da assinatura do arquivo da área de troca ("SWAP-SPACE").

Para configurar um arquivo de troca, é necessário criar o arquivo antes, executando-se **mkswap .** Uma seqüência de comandos similar aos seguintes é suficiente:

dd if=/dev/zero of=arquivo_troca bs=1024 count=8192

mkswap arquivo_troca 8192

sync

swapon arquivo_troca

Note que um arquivo de troca não contém quaisquer espaços vagos (então usar-se **cp(1)** para criá-lo, não é recomendado).

OPÇÃO

-c Verifica se o dispositivo contém blocos ruins antes de criar o sistema de arquivos. Caso algum seja encontrado, seu número é apresentado. Esta opção tem sentido para ser utilizada somente com partições de troca, e **não** deve ser usada para arquivos normais. Para assegurar-se que arquivos normais não contenham blocos ruins, a partição que os contém deve ser criada com a opção **mkfs -c**.

VEJA TAMBÉM

fsck(8), **mkfs(8)**, **fdisk(8)**

AUTOR

Linus Torvalds (torvalds@cs.helsinki.fi)

E.21 mount

NOME

mount - monta um sistema de arquivos

SINOPSE

mount [-hV]

mount -a [-fFnrsvw] [-t *vfstype***]**

mount [-fnrsvw] [-o *opções* **[,...]]** *dispositivo* | *dir*

mount [-fnrsvw] [-t *vfstype***] [-o** *opções***]** *dispositivo dir*

DESCRIÇÃO

Todos os arquivos acessíveis em um sistema Unix estão organizados em uma grande árvore, a hierarquia de arquivos, iniciada pelo raiz simbolizado como /. Estes arquivos podem estar distribuídos por diversos dispositivos. O comando **mount** destina-se a incluir o sistema de arquivos encontrado em algum dispositivo à grande árvore de arquivos. Enquanto que o comando **umount(8)** executará a ação inversa.

O formato padrão do comando **mount** é

mount -t *tipo dispositivo dir.* Isso indica ao kernel para incluir o sistema de arquivos encontrado em *dispositivo* (o qual é do tipo *tipo*) nominando-o como diretório *dir.* O conteúdo anterior (se houver) e o dono e o modo de *dir* estarão invisíveis, enquanto o sistema de arquivos estiver montado, sendo que o caminho *dir* irá referenciar-se ao raiz do sistema de arquivos do *dispositivo.*

Três formas de acionar mount sem que ele execute nenhuma ação desta ordem:

mount -h imprime uma mensagem de ajuda.

mount -V imprime a versão do programa; e

mount [-t *tipo***]** lista todos os sistemas de arquivos montados (do tipo *tipo*) - veja abaixo.

O sistema de arquivos *proc* não é associado com nenhum dispositivo em especial, e quando montado, uma palavra arbitrária, como por exemplo *proc* pode ser usada ao invés da especificação de dispositivo. (A opção usual *none* é menos indicada: a mensagem de erro 'none busy' a partir do comando **umount** pode ser confusa.)

Muitos dispositivos são indicados pelo nome do arquivo (de um dispositivo especial de blocos), tais como */dev/sda1*, mas há outras possibilidades. Por exemplo, no caso de uma montagem NFS, *dispositivo* pode se parecer com algo como *knuth.cwi.nl:/dir.*

O arquivo */etc/fstab* (veja **fstab(5)**), pode conter linhas descrevendo quais dispositivos são usualmente montados, e com quais opções. O arquivo é usado de três formas:

(i) O comando **mount -a [-t** *tipo***]** (normalmente definindo um programa de inicialização) faz com que todos os sistemas de arquivos indicados em *fstab* (de tipo apropriado) sejam montados conforme indicado, exceto para aqueles cujas linhas contenham a palavra chave **noauto .** Adicionado a opção **-F** fará com que mount gere novas instâncias do programa e monte os diversos sistemas de arquivos simultaneamente.

(ii) Quando estiver montando um sistema de arquivos mencionado em *fstab*, é suficiente fornecer somente o dispositivo, ou somente o ponto de montagem.

(iii) Normalmente somente o superusuário pode montar os sistemas de arquivos. De qualquer forma, quando *fstab* contém a opção **user** na linha, então qualquer um poderá montar o sistema correspondente.

Por exemplo, através da linha **/dev/cdrom /cd iso9660 ro,user,noauto,unhide** qualquer usuário poderá montar o sistema de arquivos iso9660 encontrado em seu cdrom, através do comando **mount /dev/cdrom** ou **mount /cd**.

Para maiores detalhes, veja **fstab(5)**.

Os programas **mount** e **umount** mantêm uma lista dos sistemas de arquivos montados atualmente no arquivo */etc/mtab*. Caso nenhum arquivo seja informado em **mount**, esta lista será apresentada. Quando o sistema de arquivos *proc* é montado (digamos em */proc*), os arquivos */etc/mtab* e */proc/mounts* têm conteúdo muito parecidos. O */proc/mounts* normalmente tem mais informações, tais como opções de montagem usadas, mas estas não estão necessariamente atualizadas (por exemplo a opção **-n** abaixo). É possível substituir */etc/mtab* por um link simbólico para */proc/mounts*, mas algumas informações podem ser perdidas desta forma, e algum uso particular como uma simulação de dispositivo não será aconselhável.

OPÇÕES

O conjunto completo de opções usado ao se iniciar o comando **mount** é determinado inicialmente pela busca das opções para o sistema de arquivos na tabela *fstab*, após será aplicada qualquer opção especificada pelo argumento **-o**, e finalmente será executada a opção **-r** or **-w**, quando presente.

Opções disponíveis para o comando **mount :**

-V Versão de saída.

-h Imprime a mensagem de ajuda.

-v Modo de mensagens ativas.

-a Monta todos os sistemas de arquivos (ou aqueles com os tipos mencionados) descritos em *fstab*.

-F (usado em conjunto com **-a**.) gera uma nova instância do comando mount para cada dispositivo. Assim sendo, a montagem em diferentes dispositivos ou diferentes servidores NFS ocorrerá em paralelo. A grande vantagem será a velocidade; porém os tempos de espera do NFS correrão em paralelo. Uma desvantagem nesta montagem será a sua ordem indefinida. Ou seja, não se pode usar esta opção caso se deseje montar tanto */usr* quanto */usr/spool*.

-f faz com que tudo seja executado exceto a montagem efetiva em si; apesar de não ser tão óbvia, esta opção permite que falsas montagens sejam realizadas, e é útil quando em conjunto com **-v** permite determinar o que o comando **mount** está tentando fazer. Pode ainda ser usado para adicionar entradas para dispositivos que foram montados anteriormente com a opção -n.

-n Montagem sem gravação de */etc/mtab*. Isso é necessário por exemplo quando o sistema de arquivos */etc* está somente com permissões de leitura.

-s Tolera o uso de opções "sujas" de montagem para sistemas de arquivos em vez de falhar. Esta opção irá ignorar as opções não suportadas pelo tipo do sistema de arquivos. Nem todos os sistemas de arquivos suportam esta opção, a qual é disponibilizada para suportar a montagem automática do Linux (automounter).

-r Monta o sistema de arquivos somente com permissões de leitura. Um sinônimo é **-o ro**.

-w Monta o sistema de arquivos com permissões de leitura e gravação. Este é o padrão. É um sinônimo de **-o rw**.

-t *vsftipe* O argumento seguinte a **-t** é usado para indicar o tipo do sistema de arquivos. Os tipos atualmente suportados são: *linux/fs/filesystems.c*: *minix*, *ext*, *ext2*, *xiafs*, *hpfs*, *msdos*, *umsdos*, *vfat*, *proc*, *nfs*, *iso9660*, *smbfs*, *ncpfs*, *affs*, *ufs*, *romfs*, *sysv*, *xenix*, *coherent*. Note que os últimos três são equivalentes e que *xenix* e *coherent* serão descontinuados em algum momento no futuro. Sugere-se o uso de *sysv* em seu lugar. Desde o kernel 2.1.21 os tipos *ext* e *xiafs* foram descontinuados.

O tipo *iso9660* é o padrão. Se nenhuma opção **-t** for apresentada, ou se o tipo **auto** for especificado, o superbloco será testado para verificação do tipo do sistema de arquivos (*minix*, *ext*, *ext2*, *xiafs*, *iso9660*, *romfs* são suportados). Caso este teste falhe e */proc/filesystems* exista, então todos os sistemas de arquivos listados serão testados, exceto aqueles que estejam marcados como "nodev" (por exemplo *proc* e *nfs*). Note que o tipo **auto** pode ser útil para unidades de disquetes montadas pelos usuários. Porém atente que o teste usa um método heurístico (a presença de um número mágico) e pode reconhecer de forma equivocada o tipo do sistema de arquivos).

Mais que um tipo pode ser especificado com uma vírgula como separador. A lista dos tipos de sistema de arquivos pode ser precedida pela palavra **no** para especificar tipos de sistemas que não devem ser utilizados nos testes. (Isso pode não ter sentido com a opção **-a** option.)

Por exemplo, o comando **mount -a -t nomsdos,ext** monta todos os sistemas de arquivos, exceto os de tipo *msdos* e *ext*.

-o Opções são especificadas com um indicador **-o** seguido por vírgula como separador. Algumas dessas opções são úteis somente quando aparecem no arquivo */etc/fstab* . As opções a seguir aplicam-se a qualquer sistema de arquivos que esteja sendo montado.

async Todas as operações de E/S no sistema de arquivos devem ser realizadas assincronamente.

atime Atualiza a data de acesso ao inode do sistema de arquivos para cada acesso que seja realizado. Esta é a opção padrão.

auto Pode ser montado com a opção **-a .**

defaults Usa as opções padrão: **rw**, **suid**, **dev**, **exec**, **auto**, **nouser**, e **async.**

dev Interpreta dispositivos especiais de blocos ou caracter no sistema de arquivos.

exec Permite a execução de binários.

noatime Não atualiza a data de acesso no inode deste sistema de arquivos (por exemplo para agilizar o acesso aos serviços de news para aumentar a velocidade de novos servidores).

noauto O arquivo somente pode ser montado explicitamente (ou seja, a opção **-a** não montará o sistema de arquivos).

nodev Dispositivos especiais de blocos ou caracter não devem ser interpretados no sistema de arquivos.

noexec Não permite a execução de qualquer binário no sistema de arquivos montado. Esta opção pode ser útil para um servidor que tem sistemas de arquivos com binários para diferentes arquitetura que não a sua própria.

nosuid Não permite o uso dos bits de configuração de identificação de usuário ou de grupo.

nouser Proíbe que um usuário comum (qualquer diferente do superusuário root) monte o sistema de arquivos. Este é o padrão.

remount Tenta remontar um sistema de arquivos já montado. Isso é comumente usado para mudar indicadores de montagem para sistemas de arquivos, especialmente para tornar sistemas de arquivos montados como somente leitura para leitura/escrita.

ro Monta o sistema de arquivos somente para leitura.

rw Monta o sistema de arquivos com permissão de leitura e gravação.

suid Permite o uso dos bits de configuração de identificação do usuário e do grupo.

sync Todas as operações de E/S do sistema de arquivos devem ser realizadas sincronamente.

user Permite que um usuário normal possa montar o sistema de arquivos. Esta opção tem implicações com as opções **noexec**, **nosuid**, e **nodev** (a menos que seja sobreposta pelas opções subseqüentes, como as opções de linha **user,exec,dev,suid**).

SISTEMAS DE ARQUIVOS COM PARÂMETROS ESPECÍFICOS

As seguintes opções aplicam-se somente a certos sistemas de arquivos. Estão ordenadas pelo tipo de sistema. Todos seguem o parâmetro **-o .**

Opções de montagem para affs

uid= *valor* e **gid=** *valor* Configura o dono e o grupo do superusuário do sistema de arquivos (padrão: uid=gid=0, mas com a opção **uid** ou **gid** sem um valor específico, são utilizados os uid e gid do processo em execução).

setuid= *valor* e **setgid=** *valor* Configura o dono e o grupo de todos os arquivos.

mode=*valor* Configura o valor de todos os arquivos para *& 0777* independentes das permissões originais. Adiciona permissões de pesquisa ao diretórios que têm permissão de leitura. O valor é dado em formato octal.

protect Não permite mudança aos bits de proteção do sistema de arquivos.

usemp Configura o uid e gid do raiz do sistema de arquivos para o uid e gid do ponto de montagem no primeiro sysn ou umount, e após descarta esta opção. Estranho...

verbose Imprime uma mensagem inicial para cada montagem bem sucedida.

prefix=*string* Prefixo usado antes do nome de volume ao se seguir um link.

volume=*string* Prefixo (de tamanho máximo de 30 caracteres) usando antes de '/' ao se seguir um link simbólico.

reserved=*valor* (Padrão: 2) Número de blocos sem uso no início do dispositivo.

root=*valor* Fornece a localização do bloco raiz de forma explícita.

bs=*valor* Fornece o tamanho do bloco. Valores permitidos são 512, 1024, 2048, 4096.

grpquota / **noquota** / **quota** / **usrquota** Estas opções são aceitas mas ignoradas.

Opções de montagem para coherent

Nenhuma.

Opções de montagem para ext

Nenhuma. Note que o sistema de arquivos 'ext' está obsoleto, portanto procure não utilizá-lo. Desde a versão 2.1.21 do Linux, o extfs não faz mais parte dos fontes do kernel.

Pontos de montagem para ext2

O sistema de arquivos 'ext2' é o sistema de arquivos padrão do Linux. A partir da versão 2.0.4 do kernel pode ser montado com a opção de montagem aleatória.

bsddf / **minixdf** Configura o comportamento da chamada ao sistema *statfs* . O padrão de **minixdf** é retornar no campo *f_ blocks* o total do número de blocos do sistema de arquivos, enquanto o **bsddf** (que é o padrão) subtrai os blocos adicionais usados pelo sistema de arquivos ext2 e não disponíveis para armazenamento.

Por exemplo % mount /k -o minixdf; df /k; umount /k

Filesystem 1024-blocks Used Available Capacity Mounted on

/dev/sda6 2630655 86954 2412169 3% /k

% mount /k -o bsddf; df /k; umount /k

Filesystem 1024-blocks Used Available Capacity Mounted on

/dev/sda6 2543714 13 2412169 0% /k

(Note que este exemplo mostra que pode-se adicionar outros parâmetros aos informados em */etc/fstab*.)

check / **check=normal** / **check=strict** Configura o nível de checagem. Quando no mínimo uma destas opções estiver ativa (e **check=normal** estiver configurado por padrão) os inodes e mapas de blocos são checados durante a montagem (o que leva meio minuto em um disco não muito grande). Com a opção de checagem estrita, a liberação de blocos verifica se o bloco a ser liberado está na área de dados.

check=none / **nocheck** Nenhuma checagem é feita.

debug Modo de depuração efetuada em cada (re)montagem.

errors=continue / **errors=remount-ro** / **errors=panic** Define o comportamento quando um erro é encontrado (ou ignora os erros e somente marca o sistema de arquivos e continua, ou remonta o sistema de arquivos com permissões somente de leitura, ou trava e desliga todo o sistema). O padrão é configurar o superbloco do sistema de arquivos com a indicação de erro, porém isso pode ser alterado usando-se **tune2fs(8)**.

grpid ou **bsdgroups** / **nogrpid** ou **sysvgroups** Estas opções definem qual a identificação de grupo que arquivos recém-criados receberão. Quando **grpid** estiver configurado, o arquivo receberá o id de grupo do diretório aonde estiver sendo criado; de outro lado (o padrão) ele assume o gid do sistema de arquivos do processo atual, a menos que o diretório tenha o bit setgid ativo, quando então ele receberá o gid do diretório pai, e recebe ainda o bit setgid ativo caso seja um diretório.

resgid= *n* e **resuid=** *n* O sistema de arquivos ext2 reserva um certo percentual do espaço disponível (por padrão 5%, veja **mke2fs(8)** e **tune2fs(8)**). Estas opções determinam quem pode usar os blocos reservados. Independente de quem tenha especificado o uid ou a quem pertença o grupo especificado.

sb=*n* Ao invés do bloco 1, use o bloco *n* como o superbloco. Isso pode ser útil quando o sistema de arquivos estiver danificado. Normalmente, cópias do superbloco são encontradas a cada 8192 blocos: no bloco 1, 8193, 16385,... (Logo pode-se ter centenas ou mesmo milhares de cópias do superbloco em um grande sistema de arquivos. Desconhecemos as opções do mke2fs que inibam a geração de tantas cópias).

grpquota / **noquota** / **quota** / **usrquota** Estas opções são aceitas mas ignoradas.

Opções de montagem para fat

(Nota: *fat* não é um sistema de arquivos, mas uma parte comum dos sistemas de arquivos *msdos*, *umsdos* e *vfat*.)

blocksize=512 / blocksize=1024 Define o tamanho do bloco (padrão é 512).

uid= *valor* e **gid=** *valor* Configura o dono e o grupo de todos os arquivos (padrão é o uid e gid do processo atual).

umask=*valor* Seta o parâmetro umask (a máscara de bits com as permissões que **não** estão presentes). O padrão é a máscara do processo atual. O valor é dado em formato octal.

check=*valor* Três diferentes níveis podem ser definidos:

r[elaxed] Maiúsculas e minúsculas são aceitas e equivalentes, parte de nomes longos são truncados (exemplo *nomemuitolongo.foobar* torna-se *nomemuit.foo*), espaços no início ou no interior do nome são aceitos no nome e na extensão.

n[ormal] Como "relaxed", porém diversos caracteres especiais (*, ?, <, espaços, etc...) são rejeitados. Este é o padrão.

s[trict] Como "normal", mas nomes não podem conter nomes longos e caracteres especiais que algumas vezes são utilizados no Linux, mas não são aceitos pelo MS-DOS (+, =, espaços, etc...).

conv=b[inary] / conv=t[ext] / conv=a[uto] O sistema de arquivos *fat* pode executar a conversão de CRLF<–>NL (formato texto MS-DOS para formato texto UNIX) no kernel. Os seguintes modos de conversão estão disponíveis:

binary nenhuma conversão é executada. Este é o padrão.

text Conversão de CRLF<–>NL é executada em todos os arquivos.

auto Conversão de CRLF<–>NL é executada em todos os arquivos que não têm uma extensão binária conhecida. A lista destas extensões pode ser encontrada no começo de *fs/fat/misc.c* (na 2.0, a lista é: exe, com, bin, app, sys, drv, ovl, ovr, obj, lib, dll, pif, arc, zip, lha, lzh, zoo, tar, z, arj, tz, taz, tzp, tpz, gz, tgz, deb, gif, bmp, tif, gl, jpg, pcx, tfm, vf, gf, pk, pxl, dvi).

Programas que executam lseeks calculado podem não funcionar após a conversão. Muitas pessoas têm seus dados perdidos após a conversão, portanto deve ser usada com extremo cuidado.

Para sistemas de arquivos montados em modo binário, uma ferramenta de conversão (de/para DOS) está disponível.

debug Aciona o indicador de *validação*. A versão e uma lista de parâmetros do sistema será apresentada (estes dados serão listados se os parâmetros aparentemente estiverem inconsistentes).

fat=12 / fat=16 Especifica uma fat de 12 bits ou de 16 bits. Isso sobrepõe a rotina de detecção automática da FAT. Deve ser usado com cuidado!

quiet Aciona o modo *sem* mensagens do sistema. Tentativas de uso do chown ou chmod não retorna erros mesmo que falhem. Deve ser usado com cuidado.

sys_immutable, showexec, dots, nodots, dotsOK=[yes|no] Diversas tentativas de forçar as convenções Unix ou DOS em um sistema de arquivos FAT.

Opções de montagem em hpfs

uid= *valor* e **gid=** *valor* Ajusta o dono e o grupo de todos os arquivos (padrão: o uid e gid do processo corrente).

umask=*valor* Ajusta a máscara de permissões que **não** estão presentes. O padrão é a umask do processo atual. O valor é fornecido em formato octal.

case=lower / **case=asis** Converte todos os nomes de arquivos para minúsculas ou deixa como estão (Padrão: **case=lower**).

conv=binary / **conv=text** / **conv=auto** Para **conv=text**, apagará alguns retornos de linha (em particular todos os seguidos por nova linha), ao ler um arquivo. Para **conv=auto**, escolha mais ou menos aleatória entre **conv=binary** e **conv=text**. Para **conv=binary**, somente lê o que esteja no arquivo. Este é o padrão.

nocheck Não cancela a montagem caso certas checagens de consistência falhem.

Opções de montagem para iso9660

Nomes de arquivos *iso9660* aparecem no formato 8.3 (ou seja como no DOS), e adicionalmente todos os caracteres estão em maiúsculas. Não há campo de dono, proteção, número de links, provisão para dispositivos de blocos ou caracter, etc... Rock Ridge é uma extensão ao iso9660 que provê todas as facilidades disponíveis no Linux. Basicamente há extensões para cada registro de diretório que provê todas as informações adicionais, e quando Rock Ridge está em uso, o sistema de arquivos não pode ser diferenciado de um sistema de arquivos Unix normal (exceto pelo fato de ter somente permissões de leitura).

norock Inibe o uso de extensões Rock Ridge, mesmo que disponíveis. Cf. **map**.

check=r[elaxed] / **check=s[trict]** Com **check=relaxed**, um nome de arquivos é primeiramente convertido para minúsculas antes da execução da sua verificação. Isto não tem sentido sem **norock** e **map=normal** (Padrão: **check=strict**).

uid= *valor* e **gid=** *valor* Fornece a todos os arquivos no sistema de arquivos o usuário e grupo indicados, possivelmente sobrepondo a informação encontrada nas extensões Rock Ridge. (Padrão: **uid=0,gid=0**.)

map=n[ormal] / **map=o[ff]** Para volumes que não estejam no formato Rock Ridge, traduz nomes normais de maiúsculas para minúsculas ASCII, despreza um finalizador ';1', e converte ';' em '.'. Com **map=off** nenhuma tradução de nomes é realizada. Veja **norock**. (padrão: **map=normal**.)

mode=*valor* Para volumes sem a extensão Rock Ridge, dá a todos os arquivos o modo indicado. (Padrão: lê a permissão para todos). Desde o Linux 2.1.37 não há mais necessidade de especificar o modo em decimais. (Octal é indicado pela precedência de um zero).

unhide Mostra arquivos associados e escondidos.

block=[512|1024|2048] Define o tamanho do bloco para o volume indicado. (Padrão: **block=1024**.)

conv=a[uto] / **conv=b[inary]** / **conv=m[text]** / **conv=t[ext]** (Padrão: **conv=binary**.) Desde o Linux 1.3.54 esta opção não tem mais efeito.

cruft Caso o maior byte do tamanho do arquivo contenha dados inúteis, configure esta opção para ignorar os bits de alta ordem do tamanho do arquivo. Isso implica que um arquivo não pode ser maior que 16 Mb. A opção 'cruft' é configurada automaticamente para o CDROM inteiro, caso ele tenha um tamanho pouco usual (negativo ou mais de 800 Mb). Também configura os números de seqüências com outros valores diferentes de 0 ou 1.

Opções de montagem para minix

Nenhuma.

Opções de montagem para MS-DOS

Caso o sistema de arquivos *MS-DOS* detecte uma inconsistência, ele reporta um erro e configura o sistema de arquivos para modo de leitura somente. O sistema de arquivos pode tornar-se gravável novamente através de uma remontagem.

Opções de montagem para ncp

Assim como *nfs*, a implementação do ncp espera um argumento binário (um *struct ncp_mount_data*) para a chamada da montagem de sistema. Este argumento é construído pelo **ncpmount(8)** e a versão atual de **mount** (2.6h) não reconhece nada sobre ncp.

Opções de montagem para nfs

Ao invés da opção de texto, passado pelo kernel, o sistema de arquivos *nfs* espera um argumento binário do tipo *struct nfs_mount_data*. O programa **mount** por si só passa as seguintes opções no formato 'tag=valor', e coloca-as na estrutura mencionada: **rsize**=*n*, **wsize**=*n*, **timeo**=*n*, **retrans**=*n*, **acregmin**=*n*, **acregmax**=*n*, **acdirmin**=*n*, **acdirmax**=*n*, **actimeo**=*n*, **retry**=*n*, **port**=*n*, **mountport**=*n*, **mounthost**=*name*, **mountprog**=*n*, **mountvers**=*n*, **nfsprog**=*n*, **nfsvers**=*n*, **namlen**=*n*. A opção **addr**=*n* é aceita mas ignorada. Ainda as seguintes opções booleanas são aceitas: **bg**, **fg**, **soft**, **hard**, **intr**, **posix**, **cto**, **ac**, **tcp**, **udp**, **lock**. Para detalhes veja **nfs(5)**.

Opções especialmente úteis incluem

rsize=8192,wsize=8192 Isto fará com que a conexão NFS seja muito mais rápida do que o buffer padrão de 1024 bytes.

hard O programa acessando um arquivo em um sistema de arquivos NFS montado irá travar quando o servidor estiver inativo. O processo não pode ser interrompido ou desativado a menos que seja especificado **intr**. Quando um servidor NFS estiver ativo novamente o programa irá continuar normalmente do ponto onde estava. Isso provavelmente é o procedimento desejado.

soft Esta opção permite que o kernel finalize a espera pelo servidor NFS caso ele não responda por alguma razão. O tempo de espera pode ser especificado da seguinte forma **timeo=time**. Esta opção pode ser útil caso o servidor NFS eventualmente não responda ou seja reinicializado quando alguns processos tentem acessar algum arquivo a partir do servidor. Normalmente isto somente causa alguns problemas.

nolock Não utiliza o travamento e reserva de recursos. Não inicializa lockd.

Opções de montagem para proc

uid= *valor* e **gid=** *valor* Estas opções são reconhecidas, mas não têm efeito até onde sabemos.

Opções de montagem para romfs

Nenhuma.

Opções de montagem para smbfs

Assim como em *nfs*, a implementação *smb* espera por um argumento binário (uma *estrutura smb_mount_data*) para montar a chamada ao sistema. Este argumento é construído por **smbmount(8)** e a versão atual de **mount** (2.6c) não reconhece nada acerca de SMB.

Opções de montagem para sysv

Nenhuma.

Opções de montagem para ufs

Nenhuma.

Opções de montagem para umsdos

Veja as opções de montagem para MS-DOS. A opção **dotsOK** é explicitamente encerrada por *umsdos*.

Opções de montagem para vfat

Antes de qualquer coisa, as opções de montagem para *fat* são reconhecidas. A opção **dotsOK** é explicitamente encerrada por *vfat*. Adicionalmente há:

uni_xlate Traduz caracteres não reconhecidos por Unicode para uma seqüência especial de fuga. Isso permite copiar e restaurar arquivos que são criados com qualquer caracter Unicode. Sem esta opção, um '?' é usado quando uma conversão não for possível. O caracter de fuga é ':' porque ele é ilegal em sistemas de arquivos vfat. A seqüência de fuga que é usada, onde u é igual a um caracter Unicode, é igual a: ':', (u & 0x3f), ((u>>6) & 0x3f), (u>>12).

posix Permite dois arquivos cujos nomes se diferenciem somente por maiúsculas ou minúsculas.

nonumtail Inicialmente tenta construir um nome menor sem uma seqüência numérica, antes de tentar *nome num.ext*.

Opções de montagem para xenix

Nenhuma.

Opções de montagem para xiafs

Nenhuma. Apesar de não haver nada de errado com xiafs, não é muito usado, e não tem manutenção. Provavelmente não se poderá utilizá-lo. Desde a versão 2.1.21 xiafs não faz parte dos fontes do kernel.

O DISPOSITIVO LOOP

Um tipo adicional é a montagem via um dispositivo chamado loop. Por exemplo, o comando **mount /tmp/fdimage /mnt -t msdos -o loop=/dev/loop3,blocksize=1024** irá configurar o dispositivo loop */dev/loop3* para corresponder ao arquivo */tmp/fdimage*, e então montará o dispositivo em */mnt*. Este tipo de montagem reconhece três opções, denominadas **loop**, **offset** e **encryption**, que são realmente opções para **losetup(8)**. Caso um dispositivo loop não seja mencionado (mas somente a opção '**-o loop** ' seja informada), então **mount** tentará encontrar algum dispositivo simulado e usá-lo.

ARQUIVOS

/etc/fstab tabela de sistemas de arquivos

/etc/mtab tabela de sistemas de arquivos montados

/etc/mtab arquivo de lock

/etc/mtab.tmp arquivo temporário

VEJA TAMBÉM

mount(2), **umount(2)**, **fstab(5)**, **umount(8)**, **swapon(8)**, **nfs(5)**, **mountd(8)**, **nfsd(8)**, **mke2fs(8)**, **tune2fs(8)**, **losetup(8)**

PROBLEMAS

É possível que um sistema de arquivos corrompido cause erros fatais no programa.

Alguns sistemas de arquivos Linux não suportam a opção **-o sync** (o ext2fs *suporta* atualizações síncronas (do modo com que o BSD faz) quando montado com a opção **sync .**)

A opção **-o remount** pode não ser capaz de mudar os parâmetros de montagem (todos os parâmetros específicos do *ext2fs* , exceto **sb**, podem ser alterados através de um comando remount, mas não se pode mudar o **gid** ou **umask** para *fatfs*).

HISTÓRIA

O comando **mount** apareceu na Versão 6 do AT&T UNIX.

E.22 mountd

NOME

mountd - servidor de sistemas de arquivos em rede - NFS

SINOPSE

/usr/sbin/rpc.mountd [-f exports-file] [-d facilidade] [-P porta] [-Dhnprv] [–debug] [–exports-file=arquivo] [–help] [–allow-non-root] [–re-export] [–version]

DESCRIÇÃO

O programa *mountd* é um servidor de sistemas de arquivos em rede. Ao receber uma solicitação de montagem a partir de um cliente NFS, ele verifica a solicitação contra uma lista de arquivos exportados descritos em */etc/exports*. Caso o cliente tenha permissão para montar o sistema de arquivos, *mountd* cria um manipulador de arquivos para o diretório selecionado, e adiciona uma entrada em */etc/rmtab*. Após receber uma solicitação de desmontar o sistema de arquivos, ele remove a entrada do cliente de *rmtab*. Note que um cliente pode estar apto a manipular um arquivo após a solicitação de desmontagem do sistema de arquivos (por exemplo, caso o cliente monte o mesmo sistema de arquivos remoto em dois diferentes pontos de montagem). Similarmente caso um cliente reinicialize o sistema sem notificar *mountd*, uma entrada permanecerá em *rmtab*.

Executado a partir do inetd

mountd pode ser inicializado a partir do *inetd* assim como no momento da inicialização do sistema adicionando-se as duas linhas seguintes ao */etc/inetd.conf*:

mount/1-2 dgram rpc/udp wait root /usr/sbin/rpc.mountd rpc.mountd

mount/1-2 stream rpc/tcp wait root /usr/sbin/rpc.mountd rpc.mountd

Quando executado a partir do *inetd*, *mountd* irá terminar após um certo período de inatividade.

OPÇÕES

-f ou **–exports-file** Esta opção especifica o arquivo de exports, listando os clientes que aquele servidor está preparado para atender e os parâmetros a serem aplicados em cada montagem (veja exports(5)). Por padrão exports são lidos de */etc/exports*.

-d ou **–debug** Registra cada transação em modo de apresentação de mensagens na saída padrão de erros. Facilidades de log válidas são *call* para registro de todas as solicitações recebidas, *auth* para autenticação de clientes, *fhcache* para operação de manipulador de arquivos cache, e *rmtab* para manipulação do */etc/rmtab*. Por padrão, a saída de registros de atividade é enviada para syslogd a menos que o servidor esteja sendo executado em primeiro plano.

-F ou **–foreground** A menos que a opção normal esteja em operação, *mountd* não irá liberar o terminal quando esta opção for informada. Quando mensagens de erro forem geradas, serão enviadas para a saída padrão de erros.

-h ou **–help** Disponibiliza um pequeno resumo de ajuda.

-n ou **–allow-non-root** Permite que solicitações de montagem sejam atendidas mesmo que não sejam originadas de portas IP reservadas. Algumas implementações mais antigas do NFS cliente necessitam disso. Algumas implementações mais recentes do NFS cliente não se baseiam na checagem de portas reservadas.

-P portnum ou **–port portnum** Faz com que *mountd* ouça a porta **portnum** ao invés de alguma porta aleatória. Por padrão, *mountd* ouvirá a porta mount/udp especificada em */etc/services*, ou, caso não seja definida, em alguma porta arbitrária abaixo de 1024.

-p ou **–promiscuous** Coloca o servidor em modo promíscuo onde ele pode servir a qualquer máquina da rede.

-r ou **–re-export** Permite que sistemas de arquivos SMB ou NFS sejam exportados. Isso pode ser usado para tornar a máquina um multiplicador NFS/SMB. Atenção deve ser dispensada ao reexportar montagens de dispositivos simulados porque a reentrada em um ponto de montagem poderá resultar em uma operação sem fim entre o cliente do sistema de arquivos e o servidor.

-v ou **–version** Informa o número da versão atual do programa.

PROBLEMAS

A informação em */etc/rmtab* não está sempre muito atualizada.

SINAIS

Ao receber um sinal SIGHUP, *mountd* irá reler o arquivo *exports* . Note que para que alterações no arquivo exports tenham efeito, deve-se enviar um SIGHUP para o *nfsd* também.

ARQUIVOS

/etc/exports

/etc/rmtab

VEJA TAMBÉM

exports(5), nfsd(8), ugidd(8C), showmount(8).

E.23 mtools

NOME

mtools - utilitários para acessar discos DOS no Unix.

INTRODUÇÃO

O mtools é uma coleção de ferramentas de domínio público que permite aos sistemas Unix manipular arquivos MS-DOS: leitura, gravação e movimentação de arquivos em sistemas de arquivos MS-DOS (tipicamente disquetes). Sempre que possível o comando tenta simular o comando equivalente no MS-DOS. Por outro lado, as restrições do MS-DOS não são implementadas. Por exemplo, é possível mover ou renomear subdiretórios.

Estas ferramentas são suficientes para fornecer todo o acesso necessário a sistemas de arquivos MS-DOS. Por exemplo, comandos como mdir a: funcionam na unidade de disquetes a: sem qualquer montagem ou inicialização prévia (assumindo que o padrão /etc/mtools.conf está configurado). Com o mtools, pode-se mudar os disquetes sem a necessidade de montá-los e desmontá-los.

ONDE OBTER O MTOOLS

O mtools pode ser encontrado nos seguintes endereços (e nos seus sites espelhos): http://mtools.ltnb.lu/mtools-3.9.1.tar.gz

ftp://www.tux.org/pub/knaff/mtools/mtools-3.9.1.tar.gz

ftp://sunsite.unc.edu/pub/Linux/utils/disk-management/mtools-3.9.1.tar.gz

Antes de reportar um erro, esteja certo de que ele já não foi corrigido nas atualizações Alpha, que podem ser encontradas em: http://www/poboxes.com/Alain.Knaff/mtools

ftp://www.tux.org/pub/knaff/mtools

As atualizações e acertos são denominados mtools- *versão* - *ddmm* .taz , onde versão significa o programa base, *dd* o dia e *mm* o mês. Devido a questões de espaço de armazenamento, normalmente estão disponíveis somente a última versão de cada ferramenta.

Há uma lista de discussão sobre o tema em mtools@tux.org . Por favor, envie todos os comunicados de erros para esta lista. Pode-se subscrevê-la enviando uma mensagem com 'subscribe mtools@linux.wauug.org' como conteúdo, destinada a majordomo@tux.org. Anúncios de novas versões são também enviados para a lista, em adição aos realizados nas lista de discussão Linux. Um histórico das mensagens está arquivado em http://www.tux.org/hypermail/mtools/latest.

FUNCIONALIDADES COMUNS A TODOS OS COMANDOS

OPÇÕES E NOMES DE ARQUIVOS

Os nomes de arquivos no MS-DOS são compostos por uma letra representando o dispositivo seguido por dois pontos, um subdiretório e o nome do arquivo. Somente o nome do arquivo é obrigatório, os demais componentes são opcionais. Nomes de arquivos sem uma letra de dispositivo referenciam arquivos Unix. Nomes de subdiretórios podem utilizar os separadores:' / ' ou ' \ '. O uso do separador ' \ ' ou caracteres de generalização devem estar entre apóstrofos a fim de evitar confusões com o ambiente de trabalho. De qualquer forma, caracteres de generalização não devem estar entre apóstrofos quando se deseje que o ambiente de trabalho **expanda** o nome do arquivo.

As rotinas de expressões regulares de generalização seguem as regras do estilo Unix. Por exemplo, ' * ' equivale à expressão DOS ' *.* '. Os bits de atributos dos arquivos (escondido, somente leitura e atributos do sistema) são ignorados durante os testes de coincidência de nomes.

Todas as opções usam o sinal - (menos), como seu primeiro caracter, diferentemente de / , utilizado no MS-DOS.

Muitos comandos do mtools permitem como parâmetros, múltiplos nomes de arquivos, que não conferem com as convenções MS-DOS, mas que são certamente mais amigáveis.

Muitos comandos do mtools permitem opções que os instruam em como lidar com nomes de arquivos distintos. Veja a seção Nomes distintos, para maiores detalhes. Todos os comandos aceitam os indicadores -V , o qual apresenta a versão do comando, e muitos aceitam o indicador -v , que aciona o modo ativo de mensagens. Neste modo, os comandos listam os nomes dos arquivos MS-DOS sobre os quais eles estão atuando, a menos que o contrário seja indicado. Veja a seção Comandos para uma descrição das opções específicas de cada comando.

LETRAS DE DISPOSITIVOS

O significado das letras de dispositivos depende da arquitetura de destino. Na maioria delas, o dispositivo A significa a primeira unidade de disquetes, dispositivo B, a segunda unidade de disquetes (se disponível), dispositivo J é um equipamento Jaz, e dispositivo Z significa uma unidade Zip. Nos sistemas onde o nome de dispositivo é derivado de identificações SCSI, a unidade Jaz é assumida como sendo dispositivo SCSI 4 e o Zip como sendo SCSI 5 (padrões de fábrica). No Linux, ambos os dispositivos são assumidos como estando no segundo dispositivo SCSI (/dev/sdb). As configurações padrão podem ser alteradas através do arquivo de configuração (Veja a seção Configuração).

DIRETÓRIO ATUAL DE TRABALHO

O comando mcd (ifmcdis) é usado para estabelecer o diretório atual de trabalho (relativo ao sistema de arquivos MS-DOS), sendo que o padrão assumido é A:/ . Diferentemente do MS-DOS, há somente um diretório de trabalho para todos os dispositivos e não um por dispositivo.

ESTILO VFAT - ARQUIVOS COM NOMES LONGOS

Esta versão do mtools suporta arquivos com nomes longos. Caso um nome de arquivo Unix seja muito longo para ser armazenado no formato DOS, ele é armazenado como um arquivo de nome longo VFAT, e um nome pequeno de acompanhamento é gerado. Este nome curto é o apresentado ao se examinar o disco com uma versão anterior a 7.0 do DOS. A tabela E.2 apresenta alguns exemplos de nomes curtos:

Nome Longo	*Nome MS-DOS*	*Razão da Mudança*
istoeumteste	`ISTOEU~1`	nome de arquivo muito longo
joao.silva	JOAO.SIL	extensão muito longa
prn.txt	`PRN~1.TXT`	PRN é um nome de dispositivo
.abc	`ABC~1`	nome de arquivo nulo
istoéum.tst	ISTO_UM.TST	caractere ilegal

Tabela E.2: Conversão de nomes de arquivos

Como se pode perceber, as seguintes transformações decorrem da derivação para nomes curtos:

* Caracteres ilegais são substituídos pelo traço (_). Os caracteres ilegais em nomes de arquivos são: ;+=[]',\"*\\<>/?:| .
* Pontos extras, que não podem ser interpretados como o nome principal ou a extensão, são removidos.
* Um número n é gerado,
* O nome é abreviado para caber no formato 8+3

O nome de arquivo inicial no estilo Unix (curto ou longo) é também chamado de nome *primário* , e o nome curto derivado é chamado de *secundário* .

Exemplo: mcopy /etc/motd a:umnomebemcomprido

O mtools cria uma entrada VFAT para umnomebemcomprido e usa UMNOMEBE como nome curto; umnomebemcomprido é o nome primário e UMNOMEBE é o secundário. mcopy /etc/motd a:motd

Motd atende às limitações do nome de arquivos DOS. Então, um nome derivado não é gerado. Neste caso, motd é o nome primário e não há nome secundário.

De forma direta: o nome primário é o nome longo, caso ele exista, ou o nome curto caso não haja nome longo.

Apesar de VFAT ser muito mais flexível que FAT, há ainda alguns nomes que não são aceitáveis, mesmo na VFAT. Há ainda alguns caracteres ilegais, como (\"*\\<>/?:|), e nomes de dispositivos que são reservados. Veja a tabela E.3.

Nome Unix	*Nome Longo*	*Razão da Mudança*
prn	prn-1	PRN é um nome de dispositivo
ab:c	ab_c-1	caractere ilegal

Tabela E.3: Transformações de nomes de arquivos

Como se pode ver, as seguintes transformações ocorrem caso um nome longo seja ilegal:

*	Caracteres ilegais são substituídos pelo traço (_).
*	Um número - *n* é gerado.

NOMES DISTINTOS

Ao gravar um arquivo em disco, seu nome longo ou nome curto pode colidir com um arquivo ou diretório já existente. Isso pode ocorrer com todos os comandos que criem novas entradas de diretório, como por exemplo: mcopy , mmd , mren , mmove , mwrite e mread . Quando ocorre uma colisão, o mtools pergunta ao usuário o que deve ser feito. E algumas opções são ofertadas:

overwrite	Regrava o arquivo existente. Não é possível sobrescrever um diretório com um arquivo.
rename	Renomeia o arquivo recém-criado. O novo nome do arquivo será solicitado.
autorename	Renomeia o arquivo recém-criado. Será criado um nome automaticamente, sem qualquer solicitação ao usuário.
skip	Desiste do arquivo corrente e vai para o próximo (se houver).

Para escolher uma dessas opções, deve-se digitar a primeira letra da opção na linha de comando. Caso se use uma letra minúscula, a ação será aplicada exclusivamente ao arquivo, caso se use uma letra maiúscula, a ação será aplicada a todos os arquivos, e não haverá uma nova pergunta.

Pode-se ainda escolher entre as seguintes opções na linha de comando, ao se utilizar o mtools:

-o	Regrava os nomes arquivos por padrão.
-O	Regrava nomes secundários por padrão.
-r	Muda o nome primário por padrão.
-R	Muda o nome secundário por padrão.
-a	Muda automaticamente o nome primário por padrão.
-A	Muda automaticamente o nome secundário por padrão.
-s	Ignora o nome primário por padrão.
-S	Ignora o nome secundário por padrão.
-m	Solicita ao usuário a decisão do que fazer com o nome primário.

-M Solicita ao usuário a decisão do que fazer com o nome secundário.

Por padrão, é solicitado ao usuário, a decisão sobre o que fazer com o nome primário e o nome secundário é automaticamente alterado, caso haja colisão de nomes.

Caso a colisão de nomes ocorra em um diretório Unix, o mtools somente pergunta se deve sobrescrever ou ignorar o arquivo.

SENSIBILIDADE A MAIÚSCULAS E MINÚSCULAS NO SISTEMA DE ARQUIVOS VFAT

O sistema de arquivos VFAT é capaz de lembrar as maiúsculas e minúsculas de cada sistema de arquivos. Ainda assim, nomes de arquivos que diferenciem-se somente por esta diferença, não poderão coexistir no mesmo diretório. Por exemplo, caso se armazene um arquivo chamado ArquivoNomeLongo em um sistema de arquivos VFAT, o mdir mostrará este arquivo como ArquivoNomeLongo e não Arquivonomelongo. Porém ao se tentar acrescentar o arquivo ArquivoNomelongo, esse não será aceito, porque maiúsculas e minúsculas são ignoradas na geração de nomes curtos.

O sistema de arquivos VFAT permite o armazenamento de maiúsculas e minúsculas no byte de atributo, caso todas as letras do nome principal e da extensão tenham o mesmo formato. O mtools usa esta informação ao mostrar os arquivos, e também para gerar um nome de arquivo Unix ao copiar (com o mcopy) um arquivo para um diretório Unix. Isso pode ter resultados inesperados quando aplicado a arquivos gravados com uma versão anterior a DOS 7.0. Na verdade o antigo estilo de nomes de arquivos mapeia todo o nome do arquivo para maiúsculas. Isso é diferente de usar versões antigas do mtools para gerar nomes de arquivos Unix em minúsculas.

FORMATOS DE ALTA CAPACIDADE

O mtools suporta um número de formatos que permitem armazenar mais dados em um disco do que o usual. Dada a diferença entre as habilidades de cada sistema, estes formatos podem não ser suportados em todos eles. O mtools reconhece esses formatos de forma transparente, quando suportados.

Para formatar estes discos, deve-se usar uma ferramenta específica do sistema operacional. Para Linux, ferramentas para disquetes podem ser encontradas no pacote fdutils nas seguintes localizações: ftp://www.tux.org/pub/knaff/fdutils/.

ftp://sunsite.unc.edu/pub/Linux/utils/disk-management/fdutils-*

Veja ainda as páginas de manual incluídas nesse pacote para maiores informações. Use superformat para formatar em qualquer padrão exceto XDF, para o qual deve ser usado xdfcopy .

MAIS SETORES

O método mais antigo de se colocar mais dados em um disco é o de utilizar mais setores e mais trilhas. Apesar do formato padrão de disquetes utilizar 80 trilhas e 18 setores (em um disco de 3 1/2 polegadas de alta densidade), é possível utilizar-se até 83 trilhas (na maioria dos dispositivos) e até 21 setores. Este método permite a armazenagem de até 1.743 Kb em disquetes de alta densidade de 3 e 1/2 polegadas. Discos com 21 setores são duas vezes mais lentos que o padrão de 18 setores, porque os setores são armazenados tão próximos que é necessário criar um espaço entre eles. Este problema não existe para formatos com 20 setores.

Estes formatos são suportados por numerosos utilitários shareware DOS, como por exemplo fdformat e vgacopy. Em seu infinito afã, Bill Gate$ acredita ter inventado isso, e chamado de discos DMF, ou discos formatados Windows. Mas na realidade, eles já existem há anos! O mtools suporta estes formatos no Linux, no SunOs e ainda em DELL Unix PC.

SETORES MAIORES

Ao se utilizar setores maiores é possível ir-se além da capacidade que pode ser obtida com setores padrões de 512 bytes. Isso se deve ao cabeçalho do setor. Este tem o mesmo tamanho, independentemente de quantos bytes estejam no setor. Ou seja, obtém-se algum espaço usando-se *menos*, mas maiores setores. Por exemplo, um setor de 4 Kb usa somente um espaço de cabeçalho, onde 8 setores de 512 bytes utilizariam oito espaços, para a mesma quantidade de espaço útil.

Este método permite armazenar-se até 1.992 Kb em um disquete de alta densidade de 3 1/2 polegadas.

O mtools suporta estes formatos somente em Linux.

2M

O formato 2m foi originalmente criado por Ciriaco Garcia de Celis. Ele também usa setores maiores que o normal para armazenar mais dados no disquete. De qualquer forma, ele usa o formato padrão (18 setores de 512 bytes cada) no primeiro cilindro, a fim de tornar estes discos mais simples de serem manuseados no DOS. Adicionalmente este método permite a utilização de formatos padrões de setor de inicialização, o qual contém a descrição de como o resto do disco deve ser lido.

Por outro lado, esta opção de formato do primeiro cilindro pode conter menos dados que as outras. Infelizmente, o DOS pode manusear somente discos onde cada trilha contenha a mesma quantidade de dados. Isso é resolvido pelo formato 2M ao esconder o fato de que a primeira trilha contém menos dados, usando uma *FAT sombra* . (Normalmente, o DOS armazena a FAT em duas cópias idênticas, para segurança adicional. O XDF armazena somente uma cópia, e diz ao DOS ter armazenado duas. Porém os dados que seriam obtidos na segunda cópia da FAT são salvos). Isso significa que **nunca se poderá usar um disco 2M para armazenar algo diferente de um sistema de arquivos DOS** .

O mtools suporta este formato somente no Linux.

XDF

XDF é um formato de alta capacidade usado pelo OS/2. Ele pode armazenar até 1.840 Kb por disco. Isso está abaixo dos melhores formatos 2M, mas a sua grande vantagem é a velocidade: 600 ms por trilha, o que é mais rápido que o formato de 21 setores por trilha, e quase mais rápido ainda que o formato padrão de 18 setores. Para poder acessar estes discos, assegure-se que o mtools foi compilado com suporte para XDF, e configure a variável use_xdf no arquivo de configuração. Veja a seção Compilando mtools e variáveis diversas, para maiores detalhes. O acesso rápido XDF está disponível somente para kernels do Linux posteriores ao 1.1.34.

O mtools suporta este formato somente em Linux.

CÓDIGOS DE SAÍDA

Todos os comandos do mtools retornam 0 ao finalizarem normalmente, 1 para falhas fatais e 2 para falhas parciais. Todos executam algumas checagens de integridade antes de prosseguirem, para estarem seguros de que o disco está no formato MS-DOS ou ext2 ou minix. Estas verificações podem rejeitar parcialmente discos corrompidos, os quais eventualmente ainda podem ser lidos. Para evitar estas checagens configure a variável MTOOLS_SKIP_CHECK ou a variável do arquivo de configuração correspondente.

PROBLEMAS

Um efeito colateral de não testar o dispositivo adequadamente (quando múltiplas capacidades de discos são suportadas), reside em mensagens de erros ocasionais do programa de controle dos dispositivos. Essas podem ser ignoradas com segurança.

A checagem dos códigos da FAT conflita em discos formatados com 1.72 Mb e com Mtools pré 2.0.7. Configure a variável MTOOLS_FAT_COMPATIBILITY (ou a variável correspondente no arquivo de configuração, variáveis globais) para evitar a verificação.

COMO CONFIGURAR O MTOOLS PARA O SEU AMBIENTE

DESCRIÇÃO

Esta seção explica a sintaxe dos arquivos de configuração do mtools. Esses arquivos são chamados /usr/local/etc/mtools.conf e /.mtoolsrc. Caso a variável de ambiente MTOOLSRC esteja configurada, o seu conteúdo será usado como o nome do terceiro arquivo de configuração. Esses arquivos de configuração descrevem os seguintes itens:

* Indicadores e variáveis de configuração global
* Indicadores e variáveis por dispositivo
* Tabelas de tradução de caracteres

LOCALIZAÇÃO DOS ARQUIVOS DE CONFIGURAÇÃO

/usr/local/etc/mtools.conf é o arquivo de configuração de todo o sistema, e /.mtoolsrc é o arquivo de configuração privativa do usuário.

Em alguns sistemas, o arquivo de configuração de todo o sistema é denominado /etc/defaults/mtools.conf.

SINTAXE DO ARQUIVO DE CONFIGURAÇÃO GERAL

O arquivo de configuração é construído por seções. Cada seção inicia com uma palavra chave de identificação, seguida por dois pontos. A seguir as variáveis e indicadores são definidos. As variáveis são inicializadas da seguinte forma: nome=valor.

Indicadores são palavras chaves sem o sinal de igualdade e os valores são intrínsecos à sua presença. Uma seção termina no final do arquivo ou onde a seção seguinte começa.

As linhas que começam com o sinal de número (#) são consideradas comentários. Caracteres de nova linha são equivalentes a espaço em branco (exceto quando finalizarem um comentário). O arquivo de configuração é insensível a maiúsculas e minúsculas, exceto para itens entre apóstrofos (como por exemplo nomes de arquivos).

VALORES PADRÕES

Para muitas plataformas, o mtools contém um número razoável de padrões pré-construídos internamente para dispositivos de disquetes. Normalmente isso não será motivo de preocupação para o usuário, caso se deseje somente acessar unidades de disquetes. Por outro lado, arquivos de configuração são necessários caso se deseje utilizar o mtools para acessar partições de disco rígido e arquivos imagens do DOSEMU.

VARIÁVEIS GLOBAIS

Variáveis globais devem ser configuradas com 1 ou 0.

As seguintes variáveis globais são reconhecidas:

MTOOLS_SKIP_CHECK Caso esteja configurada com 1, o mtools não executa muitas das checagens de integridade. Isso é necessário para ler alguns discos Atari que foram desenvolvidos com ROM mais recentes e que não serão reconhecidos de outra forma.

MTOOLS_FAT_COMPATIBILITY Caso esteja configurado com 1, o mtools não executa a checagem de tamanho da FAT. Alguns discos tem uma FAT maior que a necessária e neste caso serão rejeitadas caso esta opção não esteja configurada.

MTOOLS_LOWER_CASE Caso esteja configurada com 1, o mtools lista todos os nomes curtos de arquivo que estejam em maiúsculas como minúsculas. Esta funcionalidade está disponível para permitir um comportamento consistente com versões mais antigas do mtools que não interpretam bits de maiúsculas e minúsculas.

MTOOLS_NO_VFAT Caso esteja configurada com 1, o mtools gera entradas na VFAT para nomes de arquivos que têm nomes mistos, com caracteres maiúsculos e minúsculos, mas sempre com caracteres legais no DOS. Isso é útil ao se trabalhar com versões DOS que não conseguem lidar com nomes longos de VFAT, como por exemplo o FreeDos.

MTOOLS_DOTTED_DIR No diretório, lista o nome curto como um ponto ao invés de espaços separando o nome base da extensão.

MTOOLS_NAME_NUMERIC_TAIL Caso esteja configurado com 1 (padrão), gera sufixos numéricos para nomes longos (1). Caso esteja configurado com 0, somente gera sufixos numéricos se uma colisão de nomes ocorrer.

MTOOLS_TWENTY_FOUR_HOUR_CLOCK Caso seja igual a 1, usa a notação européia para o horário (24 horas) aos invés da notação Americana/Inglesa (12 horas am/pm).

Exemplo: Inserindo a seguinte linha no arquivo de configuração, instrui o mtools a evitar checagens de integridade: MTOOLS_SKIP_CHECK=1

Variáveis globais podem ainda ser configuradas através do ambiente: export MTOOLS_SKIP_CHECK=1

Variáveis de texto globais podem ser configuradas para qualquer valor:

MTOOLS_DATE_STRING O formato usado para a impressão das datas nos arquivos. Por padrão, igual a dd-mm-aaaa.

INDICADORES E VARIÁVEIS POR DISPOSITIVO

INFORMAÇÃO GERAL

Os indicadores e variáveis por dispositivo são descritos em uma seção de dispositivo, que começa com: dispositivo "*letra_do_dispositivo* " :

Após seguem os pares e indicadores de valores de variáveis.

Este é um exemplo de descrição de dispositivo: drive a:

file="/dev/fd0" use_xdf=1

CONFIGURAÇÃO DE GEOMETRIA DE DISCO

Informações de geometria descrevem as características físicas dos discos. Elas têm três propósitos:

formatação
: As informações sobre a geometria são gravadas no setor de inicialização de discos novos. Porém, pode-se informar este parâmetro na linha de comando. Veja a seção mformat para maiores detalhes.

filtros
: Em alguns Unix há alguns arquivos de dispositivos que suportam somente uma geometria física. Nestes casos será necessário usar um ou outro arquivo de controle de dispositivo para controlar discos de baixa ou alta densidade. A geometria é comparada com a informação contida no setor de inicialização para garantir que o arquivo de controle é capaz de ler corretamente o disco. Caso a geometria seja diferente, a entrada de dispositivo falhará e a próxima entrada no disco para o mesmo dispositivo (de mesma letra) será utilizada. Veja a seção de descrições múltiplas para maiores detalhes, no fornecimento de diversos parâmetros para o mesmo dispositivo.

  Caso nenhuma informação de geometria seja fornecida no arquivo de configuração, todos os discos serão aceitos. No Linux (e no Sparc) existem arquivos de controle que suportam configuração de geometria, como por exemplo (/dev/fd0, /dev/fd1 etc), onde os filtros não são necessários. (O mtools ainda faz filtragem de arquivos imagens de discos em Linux: isso é executado somente com o propósito de testes. Desconheço qualquer Unix que necessite de filtros atualmente).

  Se você não precisa de filtros, mas mesmo assim quer uma geometria padrão para a formatação, pode-se desativar os filtros usando o indicador mformat_only .

geometria inicial
: A informação de geometria (caso disponível) pode ser usada para configurar a geometria inicial de arquivos configuráveis de dispositivos. Ela é usada para ler o setor de inicialização, o qual contém a geometria real do disco. Caso nenhuma configuração seja fornecida, nenhuma configuração inicial será realizada. No Linux, isso não é realmente necessário, já que os arquivos configuráveis de dispositivos são capazes de autodetectar o tipo do disco (os formatos mais comuns) para ler o setor de inicialização.

Informações incorretas de geometria podem produzir erros muito estranhos. Por isso, recomendo a adição do indicador mformat_only na descrição do seu dispositivo, a menos que os filtros ou a geometria inicial sejam realmente necessários.

As seguintes variáveis relacionadas com a geometria são aceitas:

cilindros

trilhas	O número de cilindros. (cilindros é a forma preferencial, já que o formato trilhas é considerado obsoleto).
cabeças	O número de cabeças (faces).
setores	O número de setores por trilha.

Exemplo: a seguinte seção de dispositivo descreve um dispositivo de 1.44 Mb:

drive a:

file="/dev/fd0H1440"

fat_bits=12

cylinders=80 heads=2 sectors=18

As seguintes descrições de geometria e seus formatos resumidos estão disponíveis:

1.44m	discos de alta densidade de 3 1/2 polegadas. Equivale a fat_bits=12 cylinders=80 heads=2 sectors=18 .
1.2m	discos de alta densidade de 5 1/4 polegadas. Equivale a fat_bits=12 cylinders=80 heads=2 sectors=15 .
720k	discos de dupla densidade de 3 1/2 polegadas. Equivale a fat_bits=12 cylinders=80 heads=2 sectors=9 .
360k	discos de dupla densidade de 5 1/4 polegadas. Equivale a fat_bits=12 cylinders=40 heads=2 sectors=9 .

O formato resumido pode ser complementado ou sobreposto. Por exemplo, 360k sectors=8 descreve um disco de 320 Kb e é equivalente a: fat_bits=12 cylinders=40 heads=2 sectors=8.

OUTROS INDICADORES

Adicionalmente os seguintes indicadores estão disponíveis:

sync	Todas as operações de entrada e saída serão realizadas sincronamente.
nodelay	O dispositivo ou arquivo é aberto com o indicador O_NDELAY. Isso é necessário em algumas arquiteturas não Linux.
exclusive	O dispositivo ou arquivo é aberto com o indicador O_EXCL. No Linux, isso assegura acesso exclusivo ao dispositivo de disquetes. Em outras arquiteturas e para arquivos comuns, não tem qualquer efeito.

VARIÁVEIS DE USO GERAL

As seguintes variáveis de uso geral para dispositivos estão disponíveis. Dependendo do tipo, estas variáveis podem ser configuradas para um texto (arquivo, precmd) ou um número (demais variáveis):

file	O nome do arquivo ou dispositivo que contém a imagem de disco. Este é um indicador obrigatório e o nome do arquivo deve estar entre apóstrofos.

use_xdf — Caso esteja com um valor diferente de zeros, faz com que o disco seja acessado como um disquete XDF, que é um formato de alta capacidade usado pelo OS/2. Este indicador é por padrão desligado. Veja a seção XDF para maiores detalhes.

partition — Indica ao mtools para tratar o dispositivo como uma unidade particionada, e para usar a partição informada. Somente partições primárias podem ser acessadas usando-se este método, elas são numeradas de 1 a 4. Para partições lógicas, usa-se a variável de formato geral offset. A variável partition visa as mídias removíveis como Syquests, ZIP drives, e discos ótico magnéticos. Apesar do DOS enxergar dispositivos Syquests e discos ótico magnéticos como disquetes gigantesis os quais não são particionados, OS/2 e Windows NT os tratam como discos rígidos, ou seja como dispositivos particionados. O indicador partition é útil ainda em imagens de discos DOSEMU. Não é recomendado para discos rígidos com acesso direto a partições disponíveis através de montagem.

SCSI — Quando configurado com 1, esta opção indica ao mtools usar o acesso direto a entradas e saídas SCSI ao invés de utilizar as chamadas de leituras e gravações no acesso ao dispositivo. Atualmente é suportada pelas plataformas HP/UX, Solaris e SunOs. Isso é necessário, porque em algumas arquiteturas, como SunOs ou Solaris, a mídia não pode ser acessada usando-se as chamadas de sistema read e write, uma vez que o SO espera que elas contenham uma etiqueta de disco específica da SUM.

Como o acesso direto a SCSI sempre usa todo o dispositivo, é necessário especificar o indicador "partition" em conjunto.

Em algumas arquiteturas, como Solaris, o mtools necessita de privilégios de superusuário para ser capaz de executar a opção scsi=1 . Caso o mtools tenha sido instalado com a identificação de superusuário no Solaris e tente acessar um dispositivo Zip/Jaz, ele usará os privilégios de superusuário para abrir o dispositivo e utilizar as chamadas de entrada e saída SCSI. Mais comumente, privilégios de superusuário são usados somente em dispositivos descritos no arquivo de configuração de todo o sistema tais como /usr/local/etc/mtools.conf, e menos naqueles descritos em /.mtoolsrc ou $MTOOLSRC.

privileged — Quando configurado para 1, instrui o mtools a usar seus privilégios de identificação de usuário e grupo para abrir o dispositivo informado. Esta opção é válida para dispositivos descritos nos arquivos de configuração de todo o sistema (tais como /usr/local/etc/mtools.conf , e não em /.mtoolsrc ou $MTOOLSRC). Obviamente esta opção poderá não ser utilizada se mtools não está instalado com as opções setuid ou setgid. Esta opção tem implicação com 'scsi=1', mas somente para arquivos de configuração de todo o sistema. Esta opção pode ser inicializada com 0, para indicar que o mtools não deve usar seus privilégios em um determinado dispositivo mesmo se scsi=1 estiver configurado.

O mtools somente necessita ser instalado com a opção setuid se forem usadas as variáveis privileged ou SCSI. Caso contrário ele também funcionará perfeitamente sem estes privilégios.

vold — Instrui o mtools a interpretar o nome do dispositivo como um identificador de volume ao invés de como um nome de arquivo. O identificador vold é traduzido para um nome de arquivo real utilizando-se as funções media_findname() e media_oldaliases() da biblioteca volmgt . Este indicador somente estará disponível se a opção –enable-new-vold for configurada antes da compilação.

nolock — Instrui o mtools a não usar reserva de arquivos neste dispositivo. Isso é necessário em sistemas com semânticas de reserva de arquivos que contenham problemas. De qualquer forma, habilitar esta opção torna a operação menos segura em casos onde diversos usuários possam acessar o dispositivo simultaneamente.

offset — Descreve onde, no arquivo, o sistema de arquivos MS-DOS começa. Isso é útil para partições lógicas em imagens de disco rígido do DOSEMU ou para discos em memória Atari. Por padrão é igual a zero, significando que o sistema de arquivos começa no início do arquivo.

fat_bits — O número de FAT bits. Pode ser igual a 12 ou 16. Raramente é utilizado e geralmente deduzido das informações do setor de inicialização. Por outro lado, descrever erroneamente este parâmetro pode ser desastroso, caso um valor inválido seja informado. Deve-se usar somente se o mtools obtiver um número inválido na auto detecção, ou caso se deseje formatar o disco com um número estranho de FAT bits.

precmd — Em algumas variantes do Solaris é necessário usar 'volcheck -v' antes de abrir um Dispositivo de disquetes, para que o sistema seja notificado de que há um disquete no dispositivo. precmd="volcheck -v" na cláusula de dispositivo consegue obter o comportamento desejado.

Somente a variável file é obrigatória, os outros parâmetros não necessitam ser utilizados. Nestes casos os valores padrões ou auto detectados são utilizados.

FORNECENDO MÚLTIPLAS CONFIGURAÇÕES PARA UM DISPOSITIVO

É possível fornecer múltiplas descrições para um dispositivo, sendo que elas serão testadas na ordem que forem definidas, até que seja encontrada uma adequada ao dispositivo. Descrições podem falhar por várias razões:

1. porque a geometria não é adequada,
2. porque não há nenhum disco na unidade,
3. ou devido a outros problemas.

Múltiplas definições são úteis ao se usar dispositivos físicos que suportam somente uma geometria de disco. Exemplo:

drive a: file="/dev/fd0H1440" 1.44m   drive a: file="/dev/fd0H720" 720k

Esta configuração indica que o mtools deve usar /dev/fd0H1440 para discos de 1.44m (alta densidade) e /dev/fd0H720 para discos de 720k (dupla densidade). No Linux, esta funcionalidade não é realmente necessária, porque o dispositivo /dev/fd0 é capaz de administrar qualquer geometria.

Pode-se ainda usar descrições múltiplas para acessar ambos os dispositivos físicos através de somente uma letra de dispositivo:

drive z: file="/dev/fd0"

drive z: file="/dev/fd1"

Com esta descrição, mdir z: acessa o primeiro dispositivo físico que contiver um disco. Caso o primeiro esteja vazio, então o mtools testará o segundo.

Ao usar arquivos de múltiplas configurações, as descrições de dispositivos nos arquivos informados sobrepõem-se às definições usadas anteriormente para os mesmos dispositivos. Para evitar isso, pode-se usar drive+ ou +drive no lugar de drive . A primeira adiciona uma descrição ao fim da lista (por exemplo: última tentativa), e a outra no início da lista.

TABELAS DE TRADUÇÃO DE CONJUNTOS DE CARACTERES

Caso você viva nos Estados Unidos, Europa Oriental ou Austrália você não precisa ler esta seção.

POR QUE AS TABELAS DE TRADUÇÕES SÃO NECESSÁRIAS?

O DOS utiliza um mapa de códigos de caracteres diferente do Unix. Caracteres com 7 bits têm o mesmo significado, porém os conjuntos de 8 bits são afetados. Para tornar as coisas um pouco mais confusas, há diversas tabelas de traduções dependendo do país onde se esteja. A aparência dos caracteres é definida usando-se páginas de código, que não são as mesmas em todos os países. Por exemplo, algumas páginas não contêm caracteres acentuados maiúsculos. Por outro lado, algumas páginas de códigos contêm caracteres que não existem no Unix, como certos símbolos desenhados ou consoantes acentuadas, usados em alguns países do leste europeu. Isso afeta em dois aspectos os nomes de arquivos:

Caracteres Maiúsculos — Em nomes curtos, somente caracteres maiúsculos são permitidos. Isso também vale para caracteres acentuados. Por exemplo, se uma página de código não permite caracteres acentuados maiúsculos, os acentuados minúsculos serão transformados em maiúsculos sem acentos.

Nomes Longos de Arquivos — A Microsoft finalmente começou a utilizar um mapa mais padronizado para nomes de arquivos longos. Eles utilizam Unicode, que é basicamente uma versão 32 bits do ASCII. Os primeiros 256 caracteres são idênticos ao ASCII do Unix. A página de código afeta ainda a correspondência entre códigos usados em nomes longos e em nomes curtos.

O mtools considera os nomes de arquivos informados na linha de comando como tendo um mapeamento Unix, e converte os caracteres para obter nomes curtos. Por padrão, a página de código 850 é usada para mapeamento maiúsculo/minúsculo Suiço. Eu escolhi esta página de código, porque é um conjunto de caracteres com mais coincidências com os do Unix. Adicionalmente, esta página suporta a maioria dos caracteres em uso nos Estados Unidos, Austrália e Europa Ocidental. De qualquer forma, é possível escolher um mapeamento diferente. Há dois métodos: através da variável country ou explicitando as tabelas.

CONFIGURAÇÃO USANDO A VARIÁVEL COUNTRY

A variável COUNTRY é recomendada para pessoas que têm acesso a arquivos de sistemas MS-DOS e à documentação. Caso você não tenha acesso, sugerimos o uso de tabelas explícitas.

Sintaxe:

COUNTRY=" *País* [,[*página_ código*], *arquivo_ país*]"

Isso indica ao mtools para usar a tabela de tradução Unix para DOS, que confere com a *página_ código* , uma tabela de minúsculas para maiúsculas para o *País* e para usar o *arquivo_ país* para obter a tabela. O código country é comumente o prefixo telefônico do País. Referências podem ser encontradas na ajuda do DOS. Os parâmetros *página_ código* e o *arquivo país* são opcionais. Por favor não digite os colchetes, eles somente indicam que os parâmetros são opcionais. O arquivo *arquivo_ país* é fornecido com o MS-DOS, e normalmente é chamado COUNTRY.SYS, e armazenado no diretório C:\DOS. Em muitos casos ele não será necessário, e as tabelas mais comuns estão compiladas com o mtools. Então não se preocupe caso você use uma máquina que contenha somente o Unix e ela não tenha esse arquivo.

Se a *página_ código* não for fornecida, um padrão por País é usado. Caso o *arquivo_ país* não seja informado, padrões compilados serão usados para a tabela de minúsculas para maiúsculas. Isso é útil para outros Unix, distintos do Linux, que necessitem do arquivo COUNTRY.SYS disponível online.

A tabela de tradução Unix para DOS não está contida no arquivo COUNTRY.SYS, então o mtools sempre usa padrões compilados para isso. Portanto, somente algumas páginas de código são suportadas. Se a página de código de sua preferência está faltando, ou se você sabe o nome do arquivo no Windows 95 que contém esta página, mande uma mensagem para Alain.Knaff@poboxes.com.

A variável COUNTRY pode ser configurada usando-se os recursos de ambiente.

CONFIGURAÇÃO USANDO TABELAS EXPLÍCITAS

Tabelas de traduções podem ser descritas no arquivo de configuração. Duas tabelas são necessárias: primeiramente a de DOS para Unix e depois a tabela de minúsculas para maiúsculas. Uma tabela DOS para Unix começa com a palavra chave tounix, seguida por dois pontos e 128 números hexadecimais. Uma tabela de minúsculas para maiúsculas começa com a palavra chave fucase, seguida por dois pontos e 128 números hexadecimais.

A tabela mostra somente as traduções para caracteres cujos códigos sejam maiores que 128, porque a tradução para códigos menores é muito simples.

Exemplo:

tounix:

0xc7 0xfc 0xe9 0xe2 0xe4 0xe0 0xe5 0xe7

0xea 0xeb 0xe8 0xef 0xee 0xec 0xc4 0xc5

0xc9 0xe6 0xc6 0xf4 0xf6 0xf2 0xfb 0xf9

0xff 0xd6 0xdc 0xf8 0xa3 0xd8 0xd7 0x5f

0xe1 0xed 0xf3 0xfa 0xf1 0xd1 0xaa 0xba

0xbf 0xae 0xac 0xbd 0xbc 0xa1 0xab 0xbb

0x5f 0x5f 0x5f 0x5f 0x5f 0xc1 0xc2 0xc0

0xa9 0x5f 0x5f 0x5f 0x5f 0xa2 0xa5 0xac

0x5f 0x5f 0x5f 0x5f 0x5f 0x5f 0xe3 0xc3

0x5f 0x5f 0x5f 0x5f 0x5f 0x5f 0x5f 0xa4

0xf0 0xd0 0xc9 0xcb 0xc8 0x69 0xcd 0xce

0xcf 0x5f 0x5f 0x5f 0x5f 0x7c 0x49 0x5f

0xd3 0xdf 0xd4 0xd2 0xf5 0xd5 0xb5 0xfe

0xde 0xda 0xd9 0xfd 0xdd 0xde 0xaf 0xb4

0xad 0xb1 0x5f 0xbe 0xb6 0xa7 0xf7 0xb8

0xb0 0xa8 0xb7 0xb9 0xb3 0xb2 0x5f 0x5f

fucase:

0x80 0x9a 0x90 0xb6 0x8e 0xb7 0x8f 0x80

0xd2 0xd3 0xd4 0xd8 0xd7 0xde 0x8e 0x8f

0x90 0x92 0x92 0xe2 0x99 0xe3 0xea 0xeb

0x59 0x99 0x9a 0x9d 0x9c 0x9d 0x9e 0x9f

0xb5 0xd6 0xe0 0xe9 0xa5 0xa5 0xa6 0xa7

0xa8 0xa9 0xaa 0xab 0xac 0xad 0xae 0xaf

0xb0 0xb1 0xb2 0xb3 0xb4 0xb5 0xb6 0xb7

0xb8 0xb9 0xba 0xbb 0xbc 0xbd 0xbe 0xbf

0xc0 0xc1 0xc2 0xc3 0xc4 0xc5 0xc7 0xc7

0xc8 0xc9 0xca 0xcb 0xcc 0xcd 0xce 0xcf

0xd1 0xd1 0xd2 0xd3 0xd4 0x49 0xd6 0xd7

0xd8 0xd9 0xda 0xdb 0xdc 0xdd 0xde 0xdf

0xe0 0xe1 0xe2 0xe3 0xe5 0xe5 0xe6 0xe8

0xe8 0xe9 0xea 0xeb 0xed 0xed 0xee 0xef

0xf0 0xf1 0xf2 0xf3 0xf4 0xf5 0xf6 0xf7

0xf8 0xf9 0xfa 0xfb 0xfc 0xfd 0xfe 0xff

A primeira tabela traduz códigos de caracteres DOS para caracteres Unix. Por exemplo, o caracter DOS de número 129. Este é um u com dois pontos (trema) em cima. Para traduzir para Unix, buscamos o caracter número 1 na primeira tabela (1 = 129 - 128), que é igual a 0xfc. (cuidado, a numeração começa em zero). A segunda tabela converte minúsculas em maiúsculas. O mesmo exemplo de u com trema aponta para o caracter 0x9a, que é um u maiúsculo com tremas no DOS.

UNICODE - CARACTERES MAIORES QUE 256

Caso o nome de um arquivo do MS-DOS contenha um caracter Unicode maior que 256, eles serão traduzidos como sublinhado (_) ou para caracteres aproximados na aparência visual. Por exemplo, consoantes acentuadas são traduzidas em maiúsculas sem acentos. Esta tradução é usada no mdir e para os nomes de arquivos Unix gerados pelo mcopy. O Linux suporta Unicode, mas infelizmente poucas aplicações suportam estes códigos, o que dificilmente gera algum problema para o mtools. Mais importante ainda, o xterm não pode listar Unicode ainda. Caso haja demanda, pode-se incluir suporte ao Unicode nos nomes de arquivos Unix.

Cuidado: ao apagar arquivos com o mtools, o símbolo de sublinhado confere com todos os caracteres que não podem ser representados no Unix. Seja cuidadoso com o mdel.

LOCALIZAÇÃO DE ARQUIVOS DE CONFIGURAÇÃO

Os arquivos de configuração são utilizados na seguinte ordem:

1. padrões pré-compilados

2. /usr/local/etc/mtools.conf

3. /etc/mtools Esta é uma questão de compatibilidade com versões anteriores, e é utilizada somente se mtools.conf não existir.

4. /.mtoolsrc.

5. $MTOOLSRC (arquivo apontado para a variável de ambiente MTOOLSRC.

Opções descritas em arquivos posteriores sobrepõem-se às descritas em arquivos anteriores. Dispositivos definidos em arquivos anteriores persistem se não forem redefinidos em arquivos posteriores. Por exemplo dispositivos A e B podem ser definidos em /usr/local/etc/mtools.conf e os dispositivos C e D podem ser definidos em /.mtoolsrc. De qualquer forma, se /.mtoolsrc também definir A, esta nova descrição alterará a descrição de A em /usr/local/etc/mtools.conf ao invés de adicionar-se. Caso se deseje adicionar uma nova descrição a um dispositivo já descrito em um arquivo anterior, pode-se usar as palavras chaves +drive ou drive+ .

COMPATIBILIDADE COM ARQUIVOS ANTIGOS DE CONFIGURAÇÃO

A sintaxe descrita aqui é nova e vale para a versão mtools-3.0 . A sintaxe antiga ainda é suportada. Cada linha começando com letra sozinha é considerada uma descrição de dispositivo usando o antigo estilo. O estilo antigo e o novo podem ser misturados no mesmo arquivo de configuração, buscando tornar a atualização mais simples. O suporte ao formato antigo será descontinuado em algum ponto do futuro, então, de modo a desencorajar o seu uso, omito sua descrição aqui.

LISTA DE COMANDOS

Esta seção descreve os comandos disponíveis no mtools e os parâmetros de linha de comando aceitos por cada um deles. Opções que são comuns a todos os comandos do mtools não estão descritos aqui.

MATTRIB

O Mattrib é utilizado para mudar os indicadores de atributos de um arquivo MS-DOS, e obedece à seguinte sintaxe:

mattrib [-a|+a] [-h|+h] [-r|+r] [-s|+s] *arquivo_ MS-DOS* [*arquivos_ MS-DOS* ...]

Ele adiciona indicadores de atributos a um arquivo MS-DOS (com o operador ‘ + ’) ou remove (através do operador ‘ - ’).

São suportados os seguintes bits de atributo:

a — Bit de arquivamento. Usado por alguns programas de cópia de segurança para indicar um arquivo novo.

r — Bit de permissão somente para leitura. Indica que é um arquivo somente para leitura. Arquivos com este bit configurado não podem ser apagados com o comando DEL, nem modificados.

s — Bit de sistema. Usado pelo MS-DOS para indicar um arquivo do sistema operacional.

h — Bit de ocultamento. Usado para tornar o arquivo oculto para o comando DIR .

O Mattrib suporta os seguintes indicadores na linha de comando:

/	Recursivo. Lista recursivamente os atributos dos arquivos nos subdiretórios.
X	Conciso. Imprime os atributos sem espaços em branco. Se "/" não for usado, ou o *arquivo_ MS-DOS* não contém um caracter de mascaramento, e há apenas um parâmetro de arquivo do MS-DOS na linha de comando, somente os atributos são mostrados, e não o nome do arquivo. Estas situações são convenientes para um script (roteiro).

MBADBLOCKS

O comando mbadblocks é usado para pesquisar um disquete MS-DOS e marcar os blocos ruins não usados como não utilizáveis. Ele obedece a seguinte sintaxe:

mbadblocks *dispositivo* :

Ele pesquisa um disquete MS-DOS, procurando por blocos defeituosos sem uso, os quais são marcados como tais na FAT. Este comando pode ser usado após o comando mformat . Não é indicado para recuperação de discos com muitos erros.

PROBLEMAS

Este comando deveria (mas ainda não faz) tentar recuperar blocos defeituosos que estejam em uso através de leituras repetidas, e então marcá-los como defeituosos.

MCD

O comando mcd é usado para mudar o diretório de trabalho do mtools em um disco MS-DOS. Utiliza a seguinte sintaxe:

mcd [*diretório_ MS-DOS*]

Sem argumentos, mcd indica o diretório de trabalho e o dispositivo atual. De outra forma, ele muda o dispositivo e o diretório de trabalho relativo para um sistema de arquivos MS-DOS.

A variável de ambiente MD pode ser usada para localizar o arquivo onde o dispositivo e o diretório de trabalho atual estão armazenados. O padrão é $HOME/.mcwd. Informações neste arquivo com data anterior a 6 horas do horário atual não serão consideradas.

Mcd retorna 0 caso termine com sucesso e 1 em caso de falha.

Diferentemente de versões MS-DOS de cd , mcd pode ser usado para mudanças para outro dispositivo. É aconselhável remover arquivos antigos .mcwd na saída do sistema.

MCOPY

O comando mcopy é usado para copiar arquivos MS-DOS de e para o Unix. Usa a seguinte sintaxe:

mcopy [-tnvmoOsSrRA] *arquivo_ origem* *arquivo_ destino*

mcopy [-tnvmoOsSrRA] *arquivo_ origem* [*arquivos_ origem ...*] *diretório_ destino* mcopy [-tnvm] *arquivo_ origem_ MS-DOS*

Ele copia o arquivo especificado para o destino especificado, ou, copia múltiplos arquivos para o diretório especificado. A origem e o destino podem ser arquivos do MS-DOS ou do Unix.

O uso de uma letra de dispositivo nos arquivos MS-DOS, por exemplo 'a:', determina a direção da transferência. Uma designação não informada implica em um arquivo Unix cujo caminho começa no diretório atual. Caso uma letra de dispositivo seja especificada com nenhum arquivo informado (por exemplo mcopy a: .), todos os arquivos daquele dispositivo serão copiados.

Caso somente um único parâmetro de origem MS-DOS seja informado (por exemplo "mcopy a:conec.exe"), o diretório atual (' . ') é assumido como destino.

Um nome de arquivo igual a ' - ' significa a entrada ou saída padrão, dependendo de sua posição na linha de comando.

Mcopy aceita as seguintes opções na linha de comando:

b	Modo batch. Otimizada para grandes cópias recursivas, porém é menos segura se uma colisão acontece durante a cópia.
/	Cópia recursiva. Copia também os diretórios e seus conteúdos.
p	Preserva os atributos dos arquivos copiados.
Q	Ao copiar múltiplos arquivos, sinaliza assim que alguma cópia falhar (por exemplo devido à falta de espaço de armazenamento no disco de destino).
t	Transferência de arquivos texto. O mcopy converte retorno de linha e nova linha para somente nova linha.
n	Não solicita confirmação ao regravar arquivos Unix. O mcopy não avisa ao usuário quando estiver regravando um arquivo Unix existente. Para inibir o aviso para arquivos DOS, deve-se usar -o .
m	Preserva a data de modificação do arquivo. Caso o arquivo de destino já exista, a opção -n é afetada, pois o mcopy perguntará se deve sobrescrever o arquivo ou renomeá-lo para um novo arquivo.

PROBLEMAS

Diferentemente do MS-DOS, o operador '+' (anexar) não é suportado. Há porém, uma forma de produzir o mesmo efeito: mtype a:arq1 a:arq2 a:arq3 >arq_unix

mtype a:arq1 a:arq2 a:arq3 | mcopy - a:arq_dos

MDEL

O comando mdel é usado para apagar arquivos MS-DOS. Sua sintaxe é:

mdel [-v] *arquivo_ MS-DOS* [*arquivos_ MS-DOS* ...]

Ele remove arquivos de sistemas de arquivos MS-DOS.

É solicitada uma confirmação antes de remover um arquivo com permissões somente de leitura.

MDELTREE

O comando mdeltree é usado para apagar diretórios e arquivos MS-DOS. Sua sintaxe é:

mdeltree [-v] *diretório_ MS-DOS* [*diretórios_ MS-DOS* ...]

Ele remove um diretório e todos os arquivos e subdiretórios que ele contenha em um sistema de arquivos MS-DOS. Um erro ocorre caso o diretório a ser removido não exista.

MDIR

O comando mdir é usado para listar um diretório MS-DOS. Sua sintaxe é:

mdir [-w] *diretório_ MS-DOS*

mdir [-f] [-w] [-a] *arquivo_ MS-DOS* [*arquivos_ MS-DOS* ...]

Ele lista o conteúdo de um diretório MS-DOS.

São suportadas as seguintes opções nas linhas de comando:

/	Recursiva, similar ao -s no DOS.
w	Opção ampla. Com esta opção, o mdir lista os nomes de arquivos através da página sem mostrar os tamanhos e datas de criação.
a	Lista também os arquivos ocultos.
f	Rápido. Não tenta descobrir o espaço livre. Em discos grandes, descobrir o espaço livre em disco pode tomar um tempo considerável, uma vez que toda a FAT é lida e pesquisada. O indicador -f não executa este passo. Este indicador não é necessário para sistemas de arquivos com FAT32, os quais armazenam os tamanhos explicitamente.

Um erro ocorrerá se um componente do caminho não for um diretório.

MDU

O Mdu é usado para listar o espaço ocupado por um diretório, seus subdiretórios e arquivos. É similar ao comando du no Unix. A unidade usada é um dispositivo. Use o comando minfo para descobrir o tamanho do dispositivo.

mdu [-a] [*arquivos_ MS-DOS* ...]

a	Todos arquivos. Lista também o espaço ocupado por arquivos individuais.
s	Lista apenas o espaço total, não informa os detalhes de cada subdiretório.

MFORMAT

O comando mformat é usado para adicionar um sistema de arquivos MS-DOS para um disquete com formatação de baixo nível. Sua sintaxe é:

mformat [-t *cilindros*] [-h *cabeças*] [-s *setores*] [-l *nome_do_volume*] [-F] [-I *versão_do_sa*] [-S *código_do_tamanho*] [-2 *setores_na_trilha_0*] [-M *tamanho_do_setor_do_programa*] [-a] [-X] [-C] [-H *setores_ocultos*] [-r *setores_raiz*] [-B *setor_de_inicialização*] [-k] *unidade:*

Este comando adiciona um sistema de arquivos MS-DOS mínimo (setor de inicialização, FAT e diretório raiz) num disquete que já foi formatado em baixo nível pelo Unix.

As seguintes opções são suportadas: (S, 2, 1 e M podem ser usadas caso a cópia do mtools utilizada tenha sido compilada sem a opção USE_2M).

t
: O número de cilindros.

h
: O número de cabeças (lados).

s
: O número de setores por trilha. Caso a opção 2M seja informada, um setor de 512 bytes é usado em trilhas genéricas (menos na cabeça 0 trilha 0). Caso a opção 2M não seja informada, fornece o número de setores físicos por trilha (que devem ser maiores que 512 bytes).

l
: Uma etiqueta opcional para o volume.

S
: O código de tamanho, ou seja o tamanho do setor é igual a 2 ^ (código de tamanho + 7).

2
: Formato 2M. O parâmetro para esta opção descreve o número de setores na trilha zero, cabeça zero. Esta opção é recomendada para setores maiores que o normal.

1
: Não utiliza o formato 2M, mesmo que a geometria atual do disco seja 2M.

M
: Tamanho do setor. Este parâmetro descreve o tamanho do setor em bytes usado pelo sistema de arquivos MS-DOS. Por padrão é o tamanho do setor físico.

a
: Caso esta opção seja informada, um número de série no estilo Atari é gerado. O Atari guarda o número de série na etiqueta OEM.

X
: Formata o disco como um XDF. Veja a seção XDF para maiores detalhes. O disco tem inicialmente que ser formatado com o utilitário xdfcopy do pacote fdutils.

C
: Cria um arquivo imagem de disco para instalar o sistema de arquivos MS-DOS. Obviamente isso não tem aplicação em dispositivos físicos como disquetes ou partições de disco.

H
: Número de setores escondidos. Este parâmetro é útil para formatar partições de disco rígidos, que não estão alinhadas nos limites dos cilindros (ou seja a primeira cabeça na primeira trilha não pertence à partição, mas contém uma tabela de partições). Neste caso o número de setores escondidos é em geral o número de setores por cilindro. Esta facilidade não está completamente testada.

n
: Número de série.

F
: Formata a partição como FAT32 (experimental).

I
: Configura a identificação da versão_do_sa (versão do sistema de arquivos) ao formatar um dispositivo com FAT32. Para verificar a identificação, execute minfo em um dispositivo com uma FAT32 existente, e mande-me um email para que eu possa incluir isto nas versões futuras do Mtools.

c
: Configura o tamanho de dispositivos (por setor). Caso o tamanho dos dispositivos possa gerar uma FAT maior que o seu número de bits, o mtools automaticamente incrementa o número de dispositivos, até que a FAT seja pequena o suficiente.

r
: Configura o tamanho do diretório raiz (por setores). Somente aplicável a FATs de 12 e 16 bits.

B	Usa o setor de inicialização arquivado em um arquivo ou dispositivo informado, ao invés de usar o seu próprio. Somente o campo de geometria é atualizado para conferir com os parâmetros do disco de destino.
k	Mantém o setor de inicialização existente tanto quanto for possível. Somente o campo geometria é atualizado para conferir com os parâmetros do disco de destino.

Para formatar um disquete com uma densidade diferente da padrão, deve-se fornecer (no mínimo) os parâmetros de linha de comando que informam as diferenças em relação ao padrão.

O mformat retorna 0 quando finalizado com sucesso, e 1 quando houver alguma falha.

Este comando não grava informações de blocos ruins na FAT. Para tanto deve-se usar o comando mkmanifest .

MKMANIFEST

O comando mkmanifest é usado para criar um ambiente de trabalho (lista de empacotamento) para restaurar nomes de arquivos Unix. Sua sintaxe é:

mkmanifest [*arquivos*]

Ele cria um ambiente de trabalho que auxilia na restauração de nomes de arquivos Unix que foram alterados pelas restrições de nomes do MS-DOS, cujas restrições são: nomes limitados a 8 caracteres, extensões com 3 caracteres, somente maiúsculas, sem nomes de dispositivos, e sem caracteres ilegais.

Esse comando é compatível com os métodos usados em pcomm, arc, e mtools para mudar nomes de arquivos válidos no Unix, adequando-os às limitações do MS-DOS. Este comando somente é útil se o sistema de destino que lerá o disquete suportar nomes longos VFAT.

EXEMPLO

Caso se deseje copiar os seguintes arquivos Unix para um disquete MS-DOS (usando o comando mcopy).

very_long_name

2.many.dots

illegal:

good.c

prn.dev

Capital

Mcopy converte os nomes para:

very_lon

2xmany.dot

illegalx

good.c

xprn.dev

capital

O comando: mkmanifest very_long_name 2.many.dots illegal: good.c prn.dev Capital >manifest

pode produzir o seguinte resultado: mv very_lon very_long_name

mv 2xmany.dot 2.many.dots

mv illegalx illegal:

mv xprn.dev prn.dev

mv capital Capital

Note que "good.c" não requer qualquer conversão, então ele não é listado na saída.

Supondo-se que estes arquivos foram copiados de um disquete para outro sistema Unix, e se deseja retornar os arquivos para o seu formato original. Caso o arquivo "manifest" (a saída capturada acima) fosse enviado com os arquivos, ele poderia ser usado para converter os nomes de arquivos.

PROBLEMAS

Os nomes curtos gerados por mkmanifest seguem as convenções antigas (do Mtools -2.0.7) e não as do Windows 95 e Mtools-3.0.

MINFO

O comando minfo lista os parâmetros de um sistema de arquivos MS-DOS, tais como número de setores, cabeças e cilindros. Imprime ainda uma linha com o comando mformat, a qual pode ser usada para criar um sistema de arquivos DOS similar em outra mídia. Esta funcionalidade porém não funciona com mídias 2m ou Xdf, e com sistema de arquivos DOS 1.0

minfo *dispositivo* :

Este comando suporta a seguinte opção:

v	Imprime uma listagem em hexadecimal do setor de inicialização, em adição às demais informações.

MLABEL

O comando mlabel adiciona uma etiqueta de volume ao disco. Sua sintaxe é:

mlabel [-vcs] *dispositivo* :[*novo_nome*]

Ele lista a identificação do volume atual, caso esteja presente. Caso *novo_nome* não seja informado, e nem c ou s estejam configurados, será solicitado ao usuário a nova identificação do volume. Para apagar a identificação atual, basta pressionar Enter quando for solicitada a informação.

Deve-se ter cuidado ao se criar uma identificação de volume MS-DOS. Caso uma etiqueta inválida seja informada, o mlabel efetuará a alteração (e a nova etiqueta será apresentada caso o modo ativo de mensagens esteja configurado). O comando retorna 0 em caso de sucesso e 1 em caso de falha.

São suportadas as seguintes opções:

c	Apaga uma identificação existente, sem solicitar confirmação ao usuário.
s	Lista a identificação existente, sem solicitar confirmação ao usuário.

MMD

O comando mmd é usado para a criação de um subdiretório MS-DOS. Sua sintaxe é:

mmd [-voOsSrRA] *diretório_MS-DOS* [*diretórios_MS-DOS ...*]

Este comando cria um novo diretório em um sistema de arquivos MS-DOS. Caso o diretório já exista ocorrerá um erro.

MMOUNT

O comando mmount é usado para montar um disco MS-DOS. Está disponível somente no Linux, e somente tem utilidade se o kernel do SO permitir a configuração da geometria de disco. Sua sintaxe é:

mmount *dispositivo_MS-DOS* [*argumentos_do_mount*]

Esse comando lê o setor de inicialização de um disco MS-DOS, configura a geometria do disco MS-DOS e finalmente monta o dispositivo, passando os argumentos_do_mount para o comando mount. Caso nenhum argumento seja especificado, o nome do dispositivo é usado. Se o disco estiver protegido contra gravação, é montado automaticamente com permissões de somente leitura.

MMOVE

O comando mmove é usado para mover ou renomear um arquivo ou subdiretório MS-DOS existente.

mmove [-voOsSrRA] *arquivo_origem* *arquivo_destino* mmove [-voOsSrRA] *arquivo_origem* [*arquivo_origem ...*] *diretório_destino* Esse comando move ou renomeia um arquivo ou diretório MS-DOS existente. Diferentemente da versão MS-DOS de MOVE, ele é capaz de mover diretórios.

MPARTITION

O comando mpartition é usado para criar sistemas de arquivos MS-DOS como partições. Este comando objetiva ser usado em sistemas não Linux, isto é, em sistemas onde não estejam disponíveis fdisk e acesso fácil a dispositivos SCSI. Este comando funciona somente em dispositivos onde a tabela de partições está configurada.

mpartition -p *dispositivo* , mpartition -r *dispositivo* , mpartition -I *dispositivo* , mpartition -a *dispositivo* , mpartition -d *dispositivo* , mpartition -c [-s *setores*] [-h *cabeças*], [-t *cilindros*] [-v [-T *tipo*] [-b e *começo*] [-l tamanho] [-f]

São suportadas as seguintes opções:

p	Lista uma linha de comando para recriar a partição no dispositivo. Nada é apresentado se a partição do dispositivo não for definida, ou alguma inconsistência for detectada. Caso o modo de mensagens ativa (-v) esteja configurado, imprime a tabela de partições.

r	Remove a partição descrita em *dispositivo* .
I	Inicializa a tabela de partições e remove todas as partições.
c	Cria a partição descrita em *dispositivo* .
a	Ativa a partição, ou seja, torna-a inicializável. Somente uma partição pode ser inicializável de cada vez.
d	Desativa a partição, ou seja, torna-a não inicializável.

Caso nenhuma opção seja informada, as configurações atuais são listadas.

Para a criação de partições, as seguintes opções estão disponíveis:

s *setores*	Número de setores por trilha da partição (a qual é também o número de setores por trilha de todo o dispositivo).
h *cabeças*	O número de cabeças da partição (a qual é também o número de cabeças para todo o disco). Por padrão, a informação de geometria (número de setores e cabeças) é obtida através de outras entradas existentes na tabela de partições ou deduzida a partir do tamanho.
t *cilindros*	O número de cilindros da partição (e não o número de cilindros de todo o dispositivo).
b *começo*	Deslocamento inicial da partição, em setores. Caso não seja especificado, mpartition iniciará a partição no início do disco (partição número 1) ou imediatamente após o final da partição anterior.
l *tamanho*	O tamanho da partição, em setores. Caso o final não seja dado, mpartition deduz o tamanho a partir dos setores, cabeças e cilindros. Caso estes não sejam fornecidos, ele dará à partição o maior tamanho possível, considerando o tamanho do disco e o início da próxima partição.

As seguintes opções estão disponíveis para todas as operações que modifiquem a tabela de partições:

f	Normalmente, antes de gravar qualquer alteração na partição, mpartition executa certas checagens de consistência, como sobreposição e alinhamento adequado das partições. Caso alguma destas checagens falhar, a tabela de partições não será alterada. A opção -f permite que estas checagens sejam evitadas.

As seguintes opções estão disponíveis para todas as operações:

v	Junto com -p , lista a posição atual da tabela de partições (sem qualquer alteração), ou apresenta a posição após a execução de alguma alteração.
vv	Caso o modo de mensagens ativas seja ativado duas vezes, mpartition apresentará uma imagem em formato hexadecimal da tabela de partições, durante o processo de leitura e gravação da tabela.

MRD

O comando mrd é usado para remover um subdiretório MS-DOS. Sua sintaxe é:

mrd [-v] *diretório_MS-DOS* [*diretórios_MS-DOS* ...]

Ele remove um diretório de um sistema de arquivos MS-DOS. Um erro ocorrerá caso o diretório informado não exista ou esteja vazio.

MREN

O comando mren é usado para renomear ou mover um arquivo ou subdiretório MS-DOS existente. Sua sintaxe é:

mren [-voOsSrRA] *arquivo_ origem* *arquivo_ destino*

Ele renomeia um arquivo em um sistema de arquivos MS-DOS.

No modo de mensagens ativas, o Mren lista o novo nome do arquivo, caso o nome fornecido seja inválido.

Caso a primeira sintaxe seja usada (somente um arquivo fonte), e o nome de destino não contenha quaisquer barras ou vírgulas, o arquivo (ou subdiretório) é renomeado no mesmo diretório, ao invés de ser movido para o diretório atual, o diretório definido por mcd, assim como poderia ser no uso do mmove. Diferentemente da versão MS-DOS do comando REN, mren pode ser usado para renomear diretórios.

MSHOWFAT

O comando mshowfat é usado para mostrar a entradas de FAT para um arquivo. Sua sintaxe é:

$ mshowfat arquivos

MTOOLSTEST

O comando mtoolstest é usado para testar os arquivos de configuração dos arquivos do mtools. Para acioná-lo basta digitar mtoolstest sem qualquer argumento. O mtoolstest lê os arquivos de configuração do mtools, e lista as configurações na saída padrão. A saída pode ser usada como um arquivo de configuração por si só (apesar de provavelmente querer-se retirar as cláusulas redundantes). Pode-se utilizar este programa para converter antigos arquivos de configuração para formatos atualizados.

MTYPE

O comando mtype é usado para listar o conteúdo de arquivos MS-DOS. Sua sintaxe é:

mtype [-ts] *arquivo_ MS-DOS* [*arquivos_ MS-DOS ...*]

Os arquivos do MS-DOS são listados na tela.

Adicionalmente às opções padrão, o Mtype permite as seguintes opções na linha de comando:

t	Visualização de arquivo texto. O Mtype converte retornos de linha e alimentação de linha em alimentação de linha somente.
s	É retirado o bit de mais alta ordem dos dados.

Esse comando pode ser usado para estabelecer o dispositivo e o diretório atual de trabalho (relativo ao MS-DOS). De outra forma, o padrão será: A:/ .

O mtype retorna 0 ao finalizar com sucesso, 1 em algum erro fatal e 2 em caso de falha parcial.

Diferentemente da versão MS-DOS de TYPE , mtype permite múltiplos argumentos.

MZIP

O comando mzip é usado para permitir comandos específicos em discos ZIP em sistemas Solaris ou HPUX. Sua sintaxe é:

mzip [-epqrwx]

Esse comando permite as seguintes opções na linha de comando:

e	Ejeta o disco.
f	Força a ejeção mesmo que o disco esteja montado (deve ser informado em conjunto com -e).
r	Protege o disco contra gravações.
w	Remove a proteção à gravação.
p	Senha de proteção à gravação.
x	Senha de proteção.
u	Remove a proteção temporariamente até que o disco seja ejetado. O disco começa com permissão para escrita, e quando for ejetado volta ao estado anterior.
q	Solicita informações.

Para remover a senha, configurada em um dos modos sem senhas -r ou -w : o mzip solicitará a senha e após permitirá o acesso ao disco. Caso a senha tenha sido esquecida, a saída será uma formatação de baixo nível do disco (usando a configuração do BIOS do adaptador SCSI).

As ferramentas Zip comercializadas com o dispositivo também disponibilizam funções de proteção através de senhas. No MS-DOS ou no MAC, esta senha é automaticamente removida uma vez que o Ziptools seja instalado. A partir de vários artigos postados na Usenet, aprendi que a senha para o disco de ferramentas é APlaceForYourStuff . O Mzip conhece esta senha e tenta como primeira alternativa, através do comando mzip -w z: , o qual destrava o disco de ferramentas, e formata em um formato especial, utilizável tanto em PCs como em Macs. No PC, os arquivos do Mac aparecem como arquivos ocultos chamados & partishn.mac. Caso eles sejam apagados pode-se obter aproximadamente 50 Mb de espaço utilizado pelo sistema de arquivos Mac.

XCOPY

O programa xcopy é usado para se copiar recursivamente um diretório para outro. Sua sintaxe é:

xcopy *diretório_origem* *diretório_destino*

Caso o diretório de destino não exista, ele é criado. Caso ele exista, os arquivos do diretório de origem são copiados diretamente nele, e nenhum subdiretório chamado *diretório_origem* é criado, como em cp -rf .

PROBLEMAS

Este comando é bastante problemático. Uma implementação mais adequada deve provocar um novo trabalho de partes significativas do mtools, mas isso não é possível no momento. O maior problema desta implementação é que ela é ineficiente em algumas arquiteturas (diversas chamadas sucessivas ao mtools, o que pode superlotar o cache do programa).

INDICADORES DE COMPILAÇÃO

Para compilar o mtools, inicialmente deve ser acionado ./configure antes de make. Em adição ao padrão de indicadores autoconfigure, há dois indicadores específicos para arquiteturas distintas:

./configure –enable-xdf

./configure –disable-xdf Habilita o suporte a discos XDF. Por padrão estará ativo. Veja a seção XDF para maiores detalhes.

./configure –enable-vold

./configure –disable-vold Habilita o suporte a vold em plataformas Solaris. Quando utilizado em conjunto com vold, o mtools pode usar diferentes arquivos de dispositivos para o acesso direto.

./configure –enable-new-vold

./configure –disable-new-vold Habilita o novo suporte a vold para Solaris. Esta opção deverá trabalhar mais tranqüilamente do que o antigo suporte.

VEJA TAMBÉM

mattrib, mbadblocks, mcd, mcopy, mdel, mdeltree, mdir, mdu, mformat, mkmanifest, mlabel, mmd, mmount, mmove, mrd, mread, mren, mtoolstest, mtype e **mwrite.**

E.24 nfs

NOME

nfs - formato de fstab e opções

SINOPSE

/etc/fstab

DESCRIÇÃO

O arquivo *fstab* contém informações sobre os sistemas de arquivos que devem ser montados, onde e com quais opções. Para montagens NFS, contém o nome do servidor e o diretório exportado pelo servidor a ser montado, o diretório local que será o ponto de montagem e as opções específicas do NFS que controlam a forma como o arquivo será montado.

Segue um exemplo de um arquivo */etc/fstab* com uma montagem NFS.

server:/usr/local/pub /pub nfs rsize=8192,wsize=8192,timeo=14,intr

OPÇÕES

rsize=n O número de bytes que NFS usará ao ler arquivos de um servidor NFS. O valor padrão depende do kernel e normalmente é de 1.024 bytes (ainda que a velocidade de acesso cresça substancialmente ao se informar *rsize=8192).*

wsize=n O número de bytes que NFS usará ao gravar arquivos em um servidor NFS. O valor padrão depende do kernel e normalmente é de 1.024 bytes (ainda que a velocidade de acesso cresça substancialmente ao se informar *wsize=8192).*

timeo=n O número de décimos de segundo antes de enviar a primeira retransmissão após findo o tempo de espera de uma RPC. O valor padrão é de 7 décimos de segundo. Após a primeira espera, o tempo é dobrado após cada espera sem respostas, até um máximo de 60 segundos ou um número máximo de retransmissões ser atingido. Então, caso o sistema de arquivos esteja montado com a opção hard, cada novo tempo de espera começa com o dobro do tempo da anterior, novamente dobrando a cada retransmissão. O tempo máximo de espera é sempre de 60 segundos. Uma melhor performance pode ser atingida ao se incrementar o tempo de espera, quando se está montando sistemas sobre uma rede com muito tráfego, utilizando-se servidores lentos ou usando o sistema através de diversos roteadores e gateways.

retrans=n O número de tempo limite e retransmissões que devem ocorrer antes que um alarme de tempo de resposta seja acionado. O padrão é de 3 ocorrências. Quando um alarme de tempo de espera maior ocorre, a operação é interrompida ou uma mensagem de "servidor não está respondendo" é apresentada na console.

acregmin=n O tempo mínimo em segundos que os atributos de um arquivo normal devem estar em memória cache antes de solicitar novas informações para o servidor. O padrão é de 3 segundos.

acregmax=n O tempo máximo em segundos que os atributos de um arquivo normal devem estar em memória cache antes de solicitar novas informações para o servidor. O padrão é de 60 segundos.

acdirmin=n O tempo mínimo em segundos que os atributos de um diretório devem estar em memória cache antes de solicitar novas informações para o servidor. O padrão é de 30 segundos.

acdirmax=n O tempo máximo em segundos que os atributos de um diretório devem estar em memória cache antes de solicitar novas informações para o servidor. O padrão é de 60 segundos.

actimeo=n Utilizando-se actimeo, os parâmetros *acregmin, acregmax, acdirmin,* e *acdirmax* recebem o mesmo valor. Não há valor padrão.

retry=n O número de minutos na tentativa de executar operações de montagem NFS em primeiro ou segundo plano antes de desistir definitivamente. O valor padrão é de 10.000 minutos, o que é quase uma semana.

namlen=n Quando um servidor NFS não suporta a versão 2 do protocolo de montagem RPC, esta opção pode ser usada para especificar o tamanho máximo do nome de arquivos que é suportado pelo sistema de arquivos remoto. Esta opção é usada para suportar as funções pathconf do POSIX. O padrão é de 255 caracteres.

port=n O número da porta para conexão no servidor NFS. Caso esta porta seja igual a 0 (o padrão), então será perguntado ao programa mapeador de portas do servidor, qual o número a ser usado. Caso o servidor NFS não esteja registrado no programa mapeador, a porta padrão

NFS 2039 será usada.

mountport=n O número da porta de **mountd** .

mounthost=nome O nome do servidor executando **mountd** .

mountprog=n Número de programa RPC alternativo para contatar o servidor mount no servidor remoto. Esta opção é útil para servidores que podem rodar múltiplos servidores NFS. O valor padrão é 100.005, o qual é o padrão para o número do servidor mount.

mountvers=n Versão alternativa do RPC usado para contatar o servidor mount no servidor remoto. Esta opção é útil para servidores que podem executar múltiplos servidores NFS. O valor padrão é versão 1.

nfsprog=n Número alternativo do programa RPC usado para contatar o servidor NFS no servidor remoto. Esta opção é útil para servidores que podem executar múltiplos servidores NFS. O valor padrão é 100.003 para o número do servidor NFS.

nfsvers=n Versão alternativa do RPC usado para contatar o servidor NFS no servidor remoto. Esta opção é útil para servidores que podem executar múltiplos servidores NFS. O valor padrão é versão 2.

bg Caso a primeira tentativa de montagem NFS não ocorra dentro do tempo de espera definido, tenta a montagem em segundo plano. Após a transferência para segundo plano da operação de montagem, todas as tentativas subseqüentes no mesmo servidor NFS serão transferidas para segundo plano automaticamente, sem a primeira tentativa de montagem em primeiro plano. Um ponto de montagem não encontrado é tratado como a ultrapassagem do tempo de espera, para permitir montagens NFS encadeadas.

fg Caso a primeira tentativa de montagem ultrapasse o tempo de espera, tenta novamente a montagem, porém em primeiro plano. Isso complementa a opção *bg,* e o comportamento padrão.

soft Caso uma operação NFS ultrapasse o tempo de espera, então relata um erro de E/S para o programa que a acionou. O padrão é continuar tentando a operação indefinidamente.

hard Caso uma operação NFS ultrapasse o tempo de espera, então apresenta a mensagem "servidor não responde" na console e continua indefinidamente. Este é o padrão.

intr Se uma operação NFS ultrapassar o tempo de espera e estiver montada com a opção hard, permite o envio de sinais de interrupção da operação e provoca um retorno EINTR para o programa de origem. O padrão é não permitir que as operações sejam interrompidas.

posix Monta o sistema de arquivos usando a semântica POSIX. Isso permite que um sistema de arquivos NFS suporte adequadamente o comando POSIX pathconf através da solicitação de informações ao servidor sobre o tamanho máximo de um nome de arquivo. Para fazer isso, o servidor remoto deve suportar a versão 2 do protocolo de montagem RPC. Muitos servidores NFS suportam somente a versão 1.

nocto Suprime a recuperação de novos atributos na criação de um arquivo.

noac Desabilita inteiramente o cache de atributos. Esta forma de trabalho penaliza a performance de um servidor, mas permite que dois diferentes clientes NFS tenham resultados razoáveis ao utilizar ativamente um sistema de arquivos comum para gravação no servidor.

tcp Monta o sistema de arquivos usando o protocolo TCP ao invés do protocolo padrão UDP. Muitos servidores NFS suportam somente UDP.

udp Monta o sistema de arquivos NFS usando o protocolo UDP. Este é o padrão.

Todos as opções que não sejam valores, têm o seu antônimo correspondente. Por exemplo, nointr significa não permitir a interrupção de operações com arquivos.

ARQUIVOS

/etc/fstab

VEJA TAMBÉM

fstab(5), **mount**(8), **umount**(8), **exports**(5)

AUTOR

"Rick Sladkey" <jrs@world.std.com>

PROBLEMAS

As opções posix e nocto são analisadas pelo mount, porém são ignoradas silenciosamente.

As opções tcp e namlen são implementadas mas não são suportadas pela versão atual do kernel do Linux.

O comando umount deveria notificar ao servidor quando um sistema de arquivos NFS é desmontado.

E.25 nologin

NOME

nologin - evita que usuários não-root consigam se conectar ("logar") no sistema

DESCRIÇÃO

Se o arquivo **/etc/nologin** existir, **login(1)** permite acesso apenas ao root. Se outro usuário tentar o login, o conteúdo desse arquivo será exibido e o login será recusado.

ARQUIVOS

/etc/nologin

VEJA TAMBÉM

login(1), **shutdown**(8)

E.26 passwd

NOME

passwd - atualiza a senha de um usuário

SINOPSE

passwd [-u] [nome_do_usuário]

DESCRIÇÃO

O passwd é usado para alterar a senha de autenticação de um usuário.

Somente o superusuário (root) pode atualizar a senha de outro usuário, fornecendo o **nome_do_usuário**. A opção **-u**, é usada para indicar que a atualização somente pode ser efetuada para senhas expiradas, mantendo-se a senha atual até a data de sua expiração.

O passwd é configurado para trabalhar através da **API Linux-PAM.** Essencialmente, ele se auto inicializa como um serviço "passwd" com o *Linux-PAM* e utiliza módulos *password* para autenticar e atualizar as senhas dos usuários.

Uma entrada simples no arquivo de configuração do *Linux-PAM* para este serviço, pode ser:

#

a entrada do serviço passwd provoca a checagem da senha

proposta antes de atualizá-la.

#

passwd password requisite pam_cracklib.so retry=3

passwd password required pam_pwdb.so use_authtok

#

Note que nenhum outro tipo de módulo é requerido para que esta aplicação funcione corretamente.

CÓDIGOS DE RETORNO

Em caso de término normal do comando **passwd** este retornará o código 0 (zero). Um código de saída igual a 1 indica a ocorrência de erro. Mensagens de erro são descritas na saída padrão de erros.

CONFORMIDADE COM

Linux-PAM (Módulos de Autenticação Plugáveis para o Linux).

ARQUIVOS

/etc/pam.conf - o arquivo de configuração do **Linux-PAM**

PROBLEMAS

Nenhum conhecido.

VEJA TAMBÉM

pam(8), e **pam_chauthok(2)**.

Para maiores informações sobre como configurar esta aplicação com o **Linux-PAM**, veja o **Linux-PAM Guia de Administração do Sistema** em

<http://parc.power.net/morgan/Linux-PAM/index.html>

E.27 passwd

NOME

passwd - O arquivo de senhas

DESCRIÇÃO

passwd contém várias informações para cada conta de usuário. Inclui:

Nome de login

Senha criptografada opcional

ID numérico do usuário

ID numérico do grupo

Nome do usuário ou campo de comentário

Diretório home do usuário

Interpretador de comandos do usuário

O campo da senha pode não estar preenchido se o shadow de senhas estiver habilitado. Se o shadow estiver sendo usado, a senha criptografada encontra-se em */etc/shadow*. A senha criptografada consiste de 13 caracteres de um alfabeto de 64 que inclui a-z, A-Z, 0-9, . e /. Consulte **crypt (3)** para detalhes sobre a interpretação dessa string.

Uma string opcional de idade de senha pode seguir a senha criptografada, separada por uma vírgula, com o mesmo alfabeto da senha. O primeiro caractere dá o número de semanas em que a senha permanece válida. O segundo caractere dá o número de semanas que devem passar antes que o usuário possa mudar a senha. Os últimos dois caracteres dão o número de semanas passadas desde janeiro de 1970 até quando a senha foi alterada pela última vez. Quando o número de semanas em que a senha é válida passa, o usuário deve fornecer uma nova senha.

O campo de comentário é usado por vários utilitários do sistema, como **finger (1)**. Três valores adicionais podem estar presentes no campo de comentário:

pri= - define o valor inicial de nice

umask= - define o valor inicial de umask

ulimit= - define o valor inicial de ulimit

Estes campos são separados uns dos outros, e de quaisquer outros comentários, por vírgulas.

O diretório home fornece o nome do diretório de trabalho inicial. **login** usa esta informação para definir o valor da variável de ambiente **HOME** .

O campo interpretador de comando fornece o nome do interpretador (shell) a ser usado pelo usuário, ou o nome do programa inicial a ser executado. **Login** usa esta informação para definir o valor da variável de ambiente **SHELL** . Se este campo estiver vazio, o valor default é **/bin/sh** .

ARQUIVOS

/etc/passwd - informação sobre contas de usuários

VEJA TAMBÉM

login(1), **passwd(1)**, **su(1)**, **sulogin(8)**, **shadow(5)**, **pwconv(8)**, **pwunconv(8)**

E.28 proc

NOME

proc - pseudo sistema de arquivos de informações de processos.

DESCRIÇÃO

/proc é um pseudo sistema de arquivos usado como uma interface para as estruturas de dados do kernel, assim como para leitura e interpretação de /dev/kmem. Muitos dos arquivos fornecem somente permissões de leitura, mas alguns permitem que variáveis do kernel seja alteradas.

Apresentamos a seguir uma rápida descrição da hierarquia do /proc.

[número] Há um subdiretório numérico para cada processo que esteja sendo executado; o subdiretório tem o nome da identificação do processo (PID). Cada um contém os seguintes pseudo arquivos e diretórios:

cmdline Contém a linha de comando completa para o processo, a menos que todo o processo tenha sido transferido para a área de troca (swap), ou seja um processo zumbi. Nestes casos o arquivo estará vazio; isto é um arquivo que retornará 0 caracteres. Este arquivo é terminado com o caracter nulo, e não com nova linha.

cwd É o link do diretório atual de trabalho do processo. Para encontrar o cwd do processo 20, por exemplo, deve-se:

cd /proc/20/cwd; /bin/pwd

Note que o comando pwd está freqüentemente incorporado no interpretador de comandos e pode não funcionar exatamente desta forma neste contexto.

environ Este arquivo contém o ambiente do processo. As entradas são separadas por caracteres nulos, e deve haver um caracter nulo ao final do arquivo. Para listar o ambiente do processo 1, deve-se:

(cat /proc/1/environ; echo) | tr "\000" "\n"

(caso alguém queira saber porque fazer isso, veja o comando *lilo(8)*.)

exe um ponteiro para o binário que está sendo executado e aparece como uma ligação simbólica. *readlink(2)* no arquivo especial exe retorna o seguinte:

[dispositivo]:inode

Por exemplo, [0301]:1502 pode ser o inode 1502 no dispositivo com identificação primária 03 (major) (IDE, MFM, etc...) e secundária 01 (minor) (primeira partição do primeiro dispositivo).

Ainda, a ligação simbólica pode ser referenciada normalmente, ou seja ao tentar-se abrir "exe" , na verdade será aberto o executável. Pode ainda executar o comando */proc/[número]/exe* para executar uma cópia do mesmo processo como [número].

find(1) com a opção -inum pode ser usado para localizar um arquivo.

fd Este é um subdiretório contendo uma entrada para cada arquivo aberto pelo processo, nomeado pelos seus descritores e que tenham uma ligação simbólica com o arquivo real (como nas entradas em exe). Zero é a entrada padrão, 1 a saída padrão e 2 a saída padrão de erros, etc...

Programas que utilizarão nomes de arquivos, mas não a partir da entrada padrão, e que gravam arquivos, mas não através da saída padrão, podem ser depurados através do seguinte comando (assumindo-se -i como o indicador do arquivo de entrada e -o como o indicador do arquivo de saída:

CWfoobar -i /proc/self/fd/0 -o /proc/self/fd/1 ...

tendo-se então um filtro de trabalho. Note que isso não irá funcionar para programas que fazem buscas em seus arquivos, pois os arquivos no diretório fd não podem ser pesquisados.

/proc/self/fd/N é aproximadamente o mesmo que /dev/fd/N em alguns sistemas UNIX e similares a UNIX. Diversos scripts MAKEDEV do Linux ligam simbolicamente /dev/fd para /proc/self/fd, na verdade.

maps Um arquivo contendo o mapa atual de regiões da memória e suas permissões de acesso.

O formato é mostrado na tabela E.4.

onde endereço é o endereço do espaço de memória que o processo ocupa, e perms é o conjunto de permissões:

endereço	*perms*	*desloc.*	*disp*	*inode*
00000000-0002f000	r-x–	00000400	03:03	1401
0002f000-00032000	rwx-p	0002f400	03:03	1401
00032000-0005b000	rwx-p	00000000	00:00	0
60000000-60098000	rwx-p	00000400	03:03	215
60098000-600c7000	rwx-p	00000000	00:00	0
bfffa000-c0000000	rwx-p	00000000	00:00	0

Tabela E.4: Mapa das regiões de memória e seus acessos

r = leitura

w = gravação

x = execução

s = compartilhada

p = privada (copia da gravação)

deslocamento é o deslocamento no arquivo, disp é o dispositivo (primária:secundária)(major:minor), e inode refere-se ao inode do dispositivo. Zero indica que o inode está associado à uma região da memória.

mem Este não é igual ao dispositivo mem (1,1), apesar de ter o mesmo número de dispositivos. O dispositivo /dev/mem é a memória física antes da conversão de endereços, mas o arquivo mem aqui descrito é a memória acessada pelo processo. Ela não pode ser mapeada por *mmap(2)* concorrentemente, e não poderá até que *mmap(2)* seja adicionada ao kernel (o que pode ocorrer em breve).

mmap Diretório dos mapas gerados por *mmap(2)* os quais são ligações simbólicas como exe, fd/*, etc. Note que estes mapas incluem um subconjunto destas informações, então /proc/*/mmap podem ser considerados obsoletos.

"0" é normalmente libc.so.4.

/proc//mmap* foi removido do kernel do Linux na versão 1.1.40 (e realmente **estava** obsoleto)

root Unix e Linux suportam a idéia de um raiz de sistema de arquivos por processo, definidos pela chamada ao sistema *chroot(2)* . Root aponta o raiz do sistema de arquivos, e comporta-se como exe, fd/*, etc...

stat Informações sobre o status do processo. Isso é fornecido por *ps(1)* .

Os campos, em ordem, com as suas propriedades específicas em *scanf(3)* são:

pid %d Identificação do processo.

comm %s O nome do arquivo do executável entre parênteses. É visível mesmo que o processo esteja na área de troca.

state %c Um caracter da cadeia "RSDZT" onde R é em execução, S é dormindo em uma espera por interrupção, D aguardando em uma espera que não pode ser interrompida ou em área de troca, Z é um zumbi e T significa paralisado (em um sinal) ou rastreado.

ppid %d O PID do processo pai.

pgrp %d — O ID do grupo do processo.

session %d — O ID da sessão do processo.

tty %d — O tty que o processo usa.

tpgid %d — O ID do grupo do processo que atualmente detém o tty no qual o processo está conectado.

flags %u — Os indicadores do processo. Atualmente, cada indicador tem o bit matemático configurado, porque crt0.s verifica a emulação de co-processador matemático, e isso não é incluído na saída. Isso é provavelmente um erro, e nem todos os processos são compiladores C. O bit matemático é um decimal 4 e o bit de rastreamento é um decimal 10.

minflt %u — O número de pequenos erros do processo, aqueles que não requerem a carga de páginas de memória a partir do disco.

cminflt %u — O número de erros menores do processo e de seus processos filhos.

majflt %u — O número de erros maiores do processo, aqueles que requerem a carga de páginas de memória a partir do disco.

cmajflt %u — O número de erros maiores do processo e de seus processos filhos.

utime %d — O número de ciclos do processador que o processo tem previsto em modo usuário.

stime %d — O número de ciclos do processador que o processo tem previsto em modo kernel.

cutime %d — O número de ciclos do processador que o processo e seus filhos têm previstos em modo usuário.

cstime %d — O número de ciclos do processador que o processo e seus filhos têm previstos em modo kernel.

counter %d — O número máximo de ciclos do processador do próximo período de processamento destinado ao processo ou o tempo restante no período atual, caso o processo esteja ocupando o processador.

priority %d — O valor padrão acrescido de 15. O valor nunca é negativo no kernel.

timeout %u — O tempo em ciclos do processador do próximo período de espera.

itrealvalue %u — O tempo (em ciclos do processador) antes que o próximo SIGALRM seja enviado para o processo relativo a um intervalo de tempo.

starttime %d — Tempo, em ciclos do processador, que o processo iniciou após o sistema ser iniciado.

vsize %u — Tamanho da memória virtual.

rss %u — Tamanho do conjunto residente: número de páginas que o processo tem na memória real, menos 3 para uso administrativo. Estas são as páginas que contêm texto, dados ou espaço da pilha, não incluindo páginas que foram carregadas de acordo com a demanda ou que foram para a área de troca.

rlim %u — Limite em bytes do rss do processo (normalmente 2,147,483,647).

startcode %u — O endereço acima do qual o texto do programa deve ser executado.

endcode %u — O endereço abaixo do qual o texto do programa deve ser executado.

startstack %u — O endereço de início da pilha.

kstkesp %u — O valor atual de esp (ponteiro da pilha com 32 bits), conforme encontrado na pilha de páginas do kernel para o processo.

kstkeip %u EIP atual (ponteiro da instrução com 32 bits).

signal %d O mapa de bits dos sinais pendentes (normalmente zero).

blocked %d O mapa de bits dos sinais bloqueados (normalmente 0, 2 para ambientes de trabalho).

sigignore %d O mapa de bits dos sinais ignorados.

sigcatch %d O mapa de bits de sinais recebidos.

wchan %u Este é o canal no qual o processo fica esperando. Este é o endereço da chamada ao sistema, e pode ser analisada em uma lista de nomes, caso se necessite de um nome textual (caso se tenha um /etc/psdatabase atualizado, então tente *ps -l* para ver o campo WCHAN em ação).

cpuinfo Esta é uma coleção de itens dependentes da CPU e da arquitetura do sistema, sendo que cada uma destas tem uma lista diferente. As únicas duas entradas comuns são *cpu* a qual é a CPU atual em uso e *BogoMIPS* uma constante do sistema que é calculada durante a inicialização do sistema.

devices Lista dos números primários (majors) e grupos de dispositivos. Isso pode ser usado pelos scripts MAKEDEV para checagem de consistência com o kernel.

dma Lista dos canais DMA *ISA* (acesso direto à memória) registrados em uso.

filesystems Lista dos sistemas de arquivos que foram compilados com o kernel. Pode ser usado por *mount(1)* para pesquisar através de diferentes sistemas de arquivos quando nenhum é especificado.

interrupts É usado para gravar o número de interrupções por cada IRQ nas arquiteturas i386. Muito simples de ler-se, feito em formato ASCII.

ioports Lista das portas de Entrada-Saída registradas que estão em uso.

kcore Este arquivo representa a memória física do sistema e está armazenada no formato de arquivo core. Com este pseudo arquivo, e o binário do kernel com as funções de mensagens incorporadas (/usr/src/linux/tools/zSystem), pode-se usar o GDB para examinar o estado atual de qualquer estrutura de dados do kernel.

O tamanho total do arquivo é o tamanho da memória física (RAM) mais 4 Kb.

kmsg Este arquivo pode ser usado ao invés da chamada ao sistema *syslog(2)* para registrar mensagens do kernel. Um processo deve ter privilégios de superusuário para ler este arquivo e somente um processo pode fazer isso. Esse arquivo não deve ser lido se um processo syslog está sendo executado o qual usa a chamada ao sistema *syslog(2)* para registrar as mensagens do kernel.

Informações destes arquivos são recuperadas com o programa *dmesg(8)* .

ksyms Contém as definições dos símbolos exportados pelo kernel usados pelas ferramentas de *módulos(X)* para dinamicamente ligar e vincular módulos carregáveis.

loadavg A média de carga do sistema fornecida pela média do número de serviços na fila de execução há mais de 1, 5 e 15 minutos. É o mesmo que a média dada pelo programa *uptime(1)* e outros.

malloc Este arquivo somente está presente se CONFIGDEBUGMALLOC for definido durante a compilação.

meminfo É usada pelo comando *free(1)* para informar a quantidade de memória livre e utilizada (tanto a memória física como a de troca) assim como a memória compartilhada e os buffers usados pelo kernel. Tem o mesmo formato que o comando *free(1)* , exceto pelo fato de estar em bytes ao invés de Kb.

modules Uma lista dos módulos carregados pelo sistema.

net Vários pseudo arquivos, que fornecem o status de alguma parte da camada de rede. Estes arquivos contêm estruturas em formato ASCII e podem ser lidas por exemplo pelo cat. De qualquer forma, as ferramentas do *netstat(8)* possibilitam um acesso muito mais adequado a estes arquivos.

arp Ele contém uma imagem em formato ASCII da tabela ARP do kernel usada na resolução de endereços. Irá apresentar dinamicamente as entradas ARP pré-programadas e recebidas dinamicamente.

O formato é mostrado na tabela E.5

IP address	*HW type*	*Flags*	*HW address*
10.11.100.129	0x1	0x6	00:20:8A:00:0C:5A
10.11.100.5	0x1	0x2	00:C0:EA:00:00:4E
44.131.10.6	0x3	0x2	GW4PTS

Tabela E.5: Imagem da tabela ARP do kernel

Onde 'IP address' é o endereço Ipv4 da máquina, o 'HW type' é o tipo de hardware no endereço conforme a RFC 826. Os indicadores são internos à estrutura ARP(conforme definido em /usr/include/linux/if_arp.h) e o 'HW address' é o mapeamento da camada física para cada endereço IP se conhecido.

dev Os pseudo arquivos dev contêm informações sobre a situação dos dispositivos de rede. Ele dá o número de pacotes recebidos e enviados, o número de erros e colisões e outras estatísticas básicas. Eles são usados pelo programa *ifconfig(8)* para apresentar relatórios do status do dispositivo. Inter-| Receive | Transmit

face |packets errs drop fifo frame|packets errs drop fifo colls carrier

lo: 0 0 0 0 0 2353 0 0 0 0 0

eth0: 644324 1 0 0 1 563770 0 0 0 581 0

ipx Nenhuma informação.

ipx_route Nenhuma informação.

rarp Este arquivo usa o mesmo formato do arquivo *arp* e contém a base de dados de mapeamento reverso usado para prover os serviços de pesquisa de endereços reversos do *rarp(8)* . Caso RARP não esteja configurado no kernel este arquivo não estará presente.

raw Mantém uma imagem RAW (crua) da tabela de conexões. Muita desta informação não tem outra finalidade senão a depuração. O valor 'sl' é a área do kernel para a conexão, e 'local address' é o endereço local e o par de números de protocolo. "St" é o status interno da conexão. "tx_queue" e "rx_queue" são as filas de dados de entrada e saída em termos de uso de memória do kernel. Os campos "tr", "tm->when" e "rexmits" não são usados por RAW. O campo uid contém a identificação do criador da conexão.

route Nenhuma informação, mas parece similar ao *route(8)*.

snmp Este arquivo contém dados em formato ASCII necessários para o gerenciamento de IP, ICMP, TCP e UDP por um agente snmp.

tcp Mantém uma imagem da tabela de conexões TCP. Muitas informações são utilizadas exclusivamente para depuração. O valor 'sl' é a área do kernel para a conexão, e 'local address' é o endereço local e o par de números de protocolo. "St" é o status interno da conexão. "tx_queue" e "rx_queue" são as filas de entrada de dados e de saída em termos de uso de memória do kernel. Os campos "tr", "tm->when" e "rexmits" não são usados por RAW. O campo uid contém a identificação do criador da conexão.

udp Mantém uma imagem da tabela de conexões UDP. Muitas informações são utilizadas exclusivamente para depuração. O valor 'sl' é a área do kernel para a conexão, e 'local address' é o endereço local e o par de números de protocolo. "St" é o status interno da conexão. "tx_queue" e "rx_queue" são as filas de dados de entrada e saída em termos de uso de memória do kernel. Os campos "tr", "tm->when" e "rexmits" não são usados por RAW. O campo uid contém a identificação do criador da conexão.

O formato é mostrado a seguir:

```
sl  local_address rem_address   st tx_queue rx_queue tr rexmits  tm->when uid
 1: 01642C89:0201 0C642C89:03FF 01 00000000:00000001 01:000071BA 00000000 0
 1: 00000000:0801 00000000:0000 0A 00000000:00000000 00:00000000 6F000100 0
 1: 00000000:0201 00000000:0000 0A 00000000:00000000 00:00000000 00000000 0
```

unix Lista de conexões com domínios Unix presentes no sistema e seus status. O formato é mostrado a seguir:

```
Num RefCount Protocol Flags    Type St Path
 0: 00000002 00000000 00000000 0001 03
 1: 00000001 00000000 00010000 0001 01 /dev/printer
```

Onde 'Num' é a área do kernel, 'RefCount' é o número de usuários da conexão, 'Protocol' é atualmente sempre zero, 'Flags' representam os indicadores internos do kernel com o status da conexão. Tipo é sempre igual a 1 (datagramas de conexões a domínios Unix ainda não são suportadas). 'St' é o estado interno da conexão e Path é o caminho (caso exista) da conexão.

pci Lista de todos os dispositivos PCI encontrados pelo kernel durante sua inicialização e configuração.

scsi Um diretório com pseudo arquivo scsi de nível médio, e vários diretórios de arquivos de controle de baixo nível para dispositivos SCSI. Contém um arquivo para cada dispositivo SCSI do sistema, cada um com o status de alguma parte do subsistema de E/S SCSI. Estes arquivos contêm estruturas ASCII que podem ser lidas pelo comando cat. Pode-se ainda gravar alguns arquivos para reconfigurar o subsistema ou ativar ou desativar algumas funcionalidades.

scsi Uma lista de todos os dispositivos SCSI conhecidos pelo kernel. A lista é similar a uma apresentada durante a inicialização do sistema. SCSI atualmente suporta somente o comando *singledevice* que permite ao superusuário adicionar dispositivos, sem desligar o sistema, à lista de dispositivos conhecidos.

Um comando **echo 'scsi singledevice 1 0 5 0' > /proc/scsi/scsi** provocará que scsi1 pesquise no canal SCSI 0 por um dispositivo de ID 5 LUN 0. Caso haja algum neste endereço ou o endereço seja inválido, será retornado um erro.

drivername *drivername* pode atualmente ser: NCR53c7xx, aha152x, aha1542, aha1740, aic7xxx, buslogic, eata_dma, eata_pio, fdomain, in2000, pas16, qlogic, scsi_debug, seagate, t128, u15-24f, ultrastore ou wd7000. Estes diretórios mostram todos os arquivos de controle que registraram pelo menos um HBA SCSI. Cada diretório contém um arquivo registrado por dispositivo. Cada arquivo é nomeado após a indicação do número dado pela inicialização.

Estes arquivos normalmente contêm a configuração do dispositivo e do arquivo de controle, estatísticas, etc... A gravação nestes arquivos permite a execução de diferentes tarefas. Por exemplo com os comandos de superusuário *latency* e *nolatency* pode-se ligar ou desligar o comando de medição de latência no arquivo de controle eata_dma. Com os comandos *lockup* e *unlock* pode-se controlar as pesquisas de controle de barramento simuladas pelo arquivo de controle de dispositivo scsi_debug .

self Este diretório referencia-se ao processo de acesso ao sistema de arquivos /proc, e é idêntico ao diretório /proc nomeado pela identificação do mesmo processo.

stat estatísticas do kernel e do sistema

cpu 3357 0 4313 1362393 O tempo dos ciclos do processador (em centésimos de segundo) que o sistema despende em modo usuário, modo usuário de baixa prioridade, modo sistema e tarefas disponíveis, respectivamente. O último valor deve ser 100 vezes a segunda entrada no pseudo arquivo uptime.

disk 0 0 0 0 As entradas para quatro discos não estão implementadas ainda. Não estamos seguros sequer que serão, uma vez que as estatísticas do kernel em outras máquinas normalmente monitora tanto a taxa de transferência quanto E/S por segundo e este somente permite um campo por dispositivo.

page 5741 1808 O número de páginas que entraram no sistema e o número de páginas que sairam (do disco).

swap 1 0 O número de páginas de troca que foram recebidas e enviadas de/para a área de troca.

intr 1462898 O número de interrupções recebidas a partir da inicialização do sistema.

ctxt 115315 O número de mudanças de contexto que o sistema realizou.

btime 769041601 Tempo de inicialização, em segundos desde 1 de Janeiro de 1970.

sys Este diretório, presente desde a versão 1.3.57, contém um número de arquivos e subdiretórios correspondente às variáveis do kernel. Estas variáveis podem ser lidas e algumas vezes modificadas usando-se o sistema de arquivos *proc* , e usando a chamada ao sistema *sysctl(2).* Atualmente estão presentes os subdiretórios *kernel*, *net*, *vm* e cada um contém diversos arquivos e subdiretórios.

kernel Contém os arquivos *domainname*, *file-max*, *file-nr*, *hostname*, *inode-max*, *inode-nr*, *osrelease*, *ostype*, *panic*, *real-root-dev*, *securelevel*, *version.*

O arquivo somente para leitura *file-nr* fornece o número de arquivos atualmente abertos.

O arquivo *file-max* fornece o número máximo de arquivos abertos que o kernel pode administrar. Caso 1024 não seja suficiente, pode-se tentar o comando

echo 4096 > /proc/sys/kernel/file-max

Similarmente, os arquivos *inode-nr* e *inode-max* indicam o número atual e o número máximo de inodes.

Os arquivos *ostype*, *osrelease*, e *version* fornecem informações retiradas de */proc/version*.

O arquivo *panic* fornece acesso para leitura e gravação da variável do kernel *panic_timeout*. Caso seja igual a zero, o kernel irá testar esta variável sucessivamente; caso seja diferente de zero indica que o kernel deve se auto-reinicializar após o número de segundos indicado.

O arquivo *securelevel* parece sem significado no momento - o superusuário tem todos os recursos do sistema.

uptime Este arquivo contém dois números: o tempo de atividade do sistema em segundos e o tempo gasto com o processamento de processos em segundos.

version Identifica a versão do kernel que está sendo executada. Por exemplo: Linux versão 2.037 (quinlan@phaze) #1 Dom Nov 19 01:51:54 EDT 1998.

VEJA TAMBÉM

cat(1) **find(1)**, **free(1)**, **mount(1)**, **ps(1)**, **tr(1)**, **uptime(1)**, **readlink(2)**, **mmap(2)**, **chroot(2)**, **syslog(2)**, **hier(7)**, **arp(8)**, **dmesg(8)**, **netstat(8)**, **route(8)**, **ifconfig(8)**, **procinfo(8)** e muito mais

EM CONFORMIDADE COM

Este texto está em razoável conformidade com o kernel 1.3.11. Por favor atualize caso necessário.

DICAS

Note que muitas cadeias de caracteres (por exemplo o ambiente e a linha de comando) estão no formato interno, com subcampos separados por bytes contendo o caracter nulo. Pode-se tornar as informações mais claras caso se utilize *od -c* ou *tr "\000" "\n"* para acessá-las.

Esta página de manual não é completa e possivelmente contenha alguns erros, e precisa ser atualizada freqüentemente.

PROBLEMAS

O sistema de arquivos */proc* pode gerar problemas de segurança em processos executados com **chroot(2)**. Por exemplo, se */proc* é montado na hierarquia **chroot,** um **chdir(2)** para */proc/1/root* retornará para o raiz original do sistema de arquivos. Isso pode ser considerada uma facilidade ao invés de um erro, uma vez que o Linux não suporta a chamada **fchroot(2)**.

E.29 rdev

NOME

rdev - pesquisa ou configura a imagem do dispositivo raiz, dispositivo de troca, tamanho do disco em memória ou do modo de vídeo.

SINOPSE

rdev [-rsvh] [-o deslocamento **] [** imagem **[** valor **[** deslocamento **]]] rdev [-o** deslocamento **] [** imagem **[** dispositivo_raiz **[** deslocamento **]]] swapdev [-o** deslocamento **] [** imagem **[** dispositivo_troca **[** deslocamento **]]] ramsize [-o** deslocamento **] [** imagem **[** tamanho **[** deslocamento **]]] vidmode [-o** deslocamento **] [** imagem **[** modo **[** deslocamento **]]] rootflags [-o** deslocamento **] [** imagem **[** indicadores **[** deslocamento **]]]**

DESCRIÇÃO

Sem argumentos, **rdev** lista a linha em */etc/mtab* referente ao sistema de arquivos raiz atual. Sem argumentos **swapdev**, **ramsize**, **vidmode**, e **rootflags** apresentam informações de ajuda.

Em uma imagem inicializável do kernel do Linux, há diversos conjuntos de bytes que especificam o dispositivo raiz, o modo de vídeo, o tamanho do disco em memória e a área de troca. Estes conjuntos, por padrão, começam no deslocamento decimal 504 na imagem do kernel:

498 Indicadores raiz

(500 e 502 Reservados)

504 Tamanho do disco em memória

506 Modo VGA

508 Dispositivo raiz

(510 assinatura de inicialização)

rdev altera estes valores. Tipicamente os valores para o parâmetro imagem, o qual é um arquivo com um kernel inicializável do Linux, são os seguintes: /vmlinux

/vmlinux.test

/vmunix

/vmunix.test

/dev/fd0

/dev/fd1

Ao se utilizar os comandos **rdev**, ou **swapdev**, os parâmetros *dispositivo_raiz* ou *dispositivo_troca* são os seguintes:

/dev/hda[1-8]

/dev/hdb[1-8]

/dev/sda[1-8]

/dev/sdb[1-8]

Para o comando **ramsize** o parâmetro **tamanho** especifica o tamanho do disco em memória em Kb.

Para o comando **rootflags,** o parâmetro **indicador** contém informações extras usadas na montagem do raiz. Atualmente o único efeito do parâmetro indicador é forçar o kernel a montar o sistema de arquivos raiz com permissões somente de leitura, caso **flags** seja diferente de zeros.

Para o comando **vidmode** o parâmetro **modo** especifica o modo de vídeo:

-3 = Perguntar

-2 = VGA estendido

-1 = VGA normal

0 = como se 0 tivesse sido teclado no prompt

1 = como se 1 tivesse sido teclado no prompt

2 = como se 2 tivesse sido teclado no prompt

n = como se n tivesse sido teclado no prompt

Caso *valor* não seja especificado, a *imagem* será examinada para determinar as configurações atuais.

OPÇÕES

-s Faz com que **rdev** aja como **swapdev**.

-r Faz com que **rdev** aja como **ramsize**.

-R Faz com que **rdev** aja como **rootflags**.

-v Faz com que **rdev** aja como **vidmode**.

-h Apresenta uma mensagem de ajuda.

PROBLEMAS

Por razões históricas, há dois métodos alternativos de especificação do deslocamento.

A interface do usuário é pobre, não intuitiva e pode provavelmente ser reescrita integralmente, permitindo parâmetros para utilização de múltiplas imagens do kernel, para alteração ou exame, em um único comando.

Caso LILO seja usado, **rdev** não é necessário para definir o dispositivo raiz e o modo VGA, já que estes parâmetros que **rdev** modifica, podem ser solicitados pelo LILO durante a inicialização. Porém **rdev** é necessário ainda para configurar o tamanho do disco em memória. Usuários são encorajados a procurar maiores informações na documentação do LILO e a usá-lo para inicializar seus sistemas.

AUTORES

Original por Werner Almesberger (almesber@nessie.cs.id.ethz.ch)

Modificado por Peter MacDonald (pmacdona@sanjuan.UVic.CA)

Suporte a rootflags adicionado por Stephen Tweedie (sct@dcs.ed.ac.uk)

E.30 rpm

NOME

rpm - Gerenciador de Pacotes Red Hat

SINOPSE

rpm [opções]

AVISO

Por favor não use rpm diretamente em outra distribuição que não utilize a tecnologia RPM, a não ser que você saiba exatamente o que está fazendo. Isto pode quebrar o banco de dados de sua distribuição. Se você usa a DEBIAN, instale o pacote **alien** e use-o para converter pacotes do Red Hat.

DESCRIÇÃO

rpm é um poderoso *gerenciador de pacotes*, que pode ser utilizado para construir, instalar, consultar, verificar, atualizar e desinstalar pacotes de software. Um *pacote* consiste em um **arquivo** contendo arquivos e informações sobre o pacote, incluindo nome, versão e descrição.

Há dez modos básicos de operação, e cada um aceita um conjunto diferente de opções. Elas são *Instalação*, *Consulta*, *Verificação*, *Validação de Assinatura*, *Desinstalação*, *Construção*, *Reconstrução do Banco de Dados*, *Ajustar Permissões*, *Ajustar Donos e Grupos* e *Exibir Configuração.*

Modo de Instalação:

rpm -i [opções-de-instalação] <arquivo_pacote>+

Modo de Consulta:

rpm -q [opções-de-consulta]

Modo de Verificação:

rpm -V|-y|–verify [opções-de-verificação]

Modo de Validação de Assinatura:

rpm –checksig <arquivo_pacote>+

Modo de Desinstalação:

rpm -e <nome_do_pacote>+

Modo de Construção:

rpm -[b|t]O [opções-de-construção] <spec_do_pacote>+

Reconstruir Banco de Dados:

rpm –rebuilddb

Ajustar Permissões:

rpm –setperms [especificadores-de-consulta-de-pacote]

Ajustar Donos e Grupos:

rpm –setugids [especificadores-de-consulta-de-pacote]

Exibir Configuração:

rpm –showrc

Um + após um parâmetro indica que ele pode aparecer mais de uma vez (múltiplos pacotes).

OPÇÕES GERAIS

Estas opções podem ser usadas em todos os modos de operação.

-vv Exibe um monte de informações bizarras de depuração.

–keep-temps Não remove arquivos temporários (/tmp/rpm-*). Serve principalmente para depurar o rpm.

–quiet Exibe o mínimo possível - normalmente apenas mensagens de erro serão mostradas.

–help Exibe uma mensagem de utilização mais longa que o normal.

–version Exibe uma única linha contendo o número da versão do rpm sendo usado.

–rcfile <arquivo> Usa **<arquivo>** ao invés de **/etc/rpmrc** e **$HOME.rpmrc** como arquivo de configurações.

–root <*dir*> Usa o sistema com raiz em <*dir*> para todas as operações. Note que o banco de dados será lido ou modificado sob <*dir*> e que os scripts de pré e pós instalação ou desinstalação serão executados após um chroot() para <*dir*> .

–dbpath <caminho> Usa o banco de dados do RPM em <caminho>.

–ftpproxy <máquina> Usa <máquina> como um proxy de FTP. Veja **OPÇÕES DE FTP** .

–ftpport <porta> Usa <porta> como a porta de FTP. Veja **OPÇÕES DE FTP** .

OPÇÕES DE INSTALAÇÃO E ATUALIZAÇÃO

A forma geral de um comando de instalação é

rpm -i [opções-de-instalação] <arquivo_pacote>+

Isto instala um novo pacote. A forma geral de um comando de atualização é

rpm -U [opções-de-instalação] <arquivo_pacote>+

Isto instala ou atualiza o pacote atualmente instalado para a versão do novo RPM. Isso é o mesmo que instalar, exceto que todas as versões anteriores dos pacotes serão removidas do sistema após a atualização.

O <arquivo_pacote> pode ser especificado no estilo de uma URL de ftp, no caso em que o pacote será transferido antes de instalado. Veja **OPÇÕES DE FTP** para mais informações ao suporte a ftp embutido no rpm.

–force O mesmo que usar –replacepkgs, –replacefiles e –oldpackage.

-h, –hash Exibe 50 caracteres # (hash) à medida que o **arquivo** é desempacotado. Use em conjunto com **-v** para uma exibição interessante.

–oldpackage Permite que uma atualização substitua um pacote por uma versão anterior.

–percent Exibe porcentagens à medida que os **arquivos** são desempacotados. Isto é para tornar o rpm facilmente executável a partir de outras ferramentas.

–replacefiles Instala os pacotes mesmo que eles substituam arquivos de outros pacotes, já instalados.

–replacepkgs Instala os pacotes mesmo que alguns deles já estejam instalados no sistema.

–allfiles Instala ou atualiza todos os arquivos do pacote que estão faltando, independente deles existirem ou não.

–nodeps Não verifica as dependências antes de instalar ou atualizar um pacote.

–noscripts Não executa os scripts de pré ou pós instalação.

–excludedocs Não instala nenhum arquivo marcado como documentação (o que inclui as páginas de manual e documentos texinfo).

–includedocs Instala os arquivos de documentação. Isto só é necessário se *excludedocs: 1* está especificado em um arquivo de configuração do rpm.

–test Não instala o pacote, apenas verifica e avisa sobre possíveis conflitos.

–prefix <caminho> Isto determina o prefixo de instalação como <caminho> para pacotes relocalizáveis (pacotes que não precisam ser instalados apenas em um certo caminho).

–ignorearch Isto permite a instalação ou atualização de pacotes mesmo que as arquiteturas do RPM binário e da máquina não coincidam.

–ignoreos Isto permite a instalação ou atualização mesmo que os sistemas operacionais do RPM binário e da máquina não coincidam.

OPÇÕES DE CONSULTA

A forma geral de um comando de consulta é

rpm -q [opções-de-consulta]

Você pode especificar o formato em que as informações sobre os pacotes serão exibidas. Para tal, use a opção **–queryformat**, seguida de uma frase especificando o formato.

Formatos de consulta são versões modificadas da formatação padrão da função **printf()** . O formato é feito de frases estáticas (que podem incluir as seqüências de escape do C como nova linha, tabulação e outros caracteres especiais) e formatadores de tipo do **printf()** . Já que o **rpm** já sabe qual tipo exibir, o especificador de tipo pode ser omitido, e substituído pelo nome do cabeçalho a ser exibido, cercado por **{}** . A parte **RPMTAG_** do cabeçalho pode ser omitida.

Saídas alternativas podem ser requisitadas acrescentando ***:tipo*** ao nome do cabeçalho. Atualmente, os seguintes tipos são suportados: **octal** , **date** , **shescape** , **perms** , **fflags** , e **depflags** .

Por exemplo, para exibir somente o nome dos pacotes consultados, você pode usar **%{NAME}** como frase de formato. Para exibir os pacotes e a distribuição, formatados em duas colunas, você pode usar **%-30{NAME}{DISTRIBUTION}** .

O **rpm** exibirá uma lista de formatos sobre os quais ele sabe quando for chamado com o parâmetro **–querytags** .

Há dois subconjuntos de opções de consulta: seleção de pacotes e seleção de informações.

Opções de seleção de pacotes:

<nome_do_pacote> Consulta o pacote instalado de nome ***<nome_do_pacote>*** .

-a Consulta todos os pacotes instalados.

–whatrequires *<capacidade>* Consulta todos os pacotes que requerem *<capacidade>* para funcionar corretamente.

–whatprovides *<virtual>* Consulta todos os pacotes que fornecem a capacidade *<virtual>* .

-f *<arquivo>* Consulta o pacote do qual *<arquivo>* faz parte.

-p *<arquivo_pacote>* Consulta um arquivo de pacote (desinstalado) de nome *<arquivo_pacote>* . O <arquivo_pacote> pode ser especificado como uma URL no estilo ftp, caso no qual o cabeçalho do pacote será transferido e consultado. Veja **OPÇÕES DE FTP** para mais informações sobre o suporte a ftp embutido no rpm.

Opções de seleção de informações:

-i Exibe informações sobre o pacote, incluindo nome, versão e descrição. Utiliza a opção –**queryformat** se ela foi especificada.

-R Lista os pacotes dos quais este depende (o mesmo que –**requires**).

–**provides** Lista as capacidades que este pacote fornece.

–**changelog** Exibe informações sobre as mudanças neste pacote.

-l Lista os arquivos contidos no pacote.

-s Exibe os *estados* dos arquivos no pacote (implica **-l**). O estado de cada arquivo é *normal* , *não instalado* (uninstalled), ou *substituído* (replaced).

-d Lista apenas os arquivos de documentação (implica **-l**).

-c Lista apenas os arquivos de configuração (implica **-l**).

–**scripts** Lista os scripts de shell do pacote, que são usados no processo de instalação e desinstalação, se existirem.

–**dump** Exibe informações dos arquivos como segue: caminho tamanho mtime md5sum modo dono grupo isconfig isdoc rdev symlink. Isto deve ser usado com uma das opções **-l** , **-c** , **-d** . mtime indica o dia e hora de criação do arquivo, isdoc indica se o arquivo é documentação e isconfig indica se é um arquivo de configuração.

OPÇÕES DE VERIFICAÇÃO

A forma geral de um comando de verificação é

rpm -V|-y|–verify [opções-de-verificação]

Verificar um pacote compara informações sobre os arquivos do pacote instalados e as informações sobre eles, tiradas do pacote original e armazenadas no banco de dados do rpm. Entre outras coisas, verificar compara o tamanho, a soma MD5, permissões, tipo, dono e grupo de cada arquivo. Quaisquer discrepâncias serão exibidas. As opções de especificação de pacotes são as mesmas que para consulta de pacotes.

Arquivos que não foram instalados do pacote, por exemplo, arquivos de documentações que foram excluídos usando a opção "–**excludedocs** ", serão ignorados silenciosamente.

O formato da saída é uma cadeia de 8 caracteres, um possível "**c** " denotando um arquivo de configuração, e então o nome do arquivo. Cada um dos oito caracteres denota o resultado da comparação de um atributo do arquivo com o valor registrado no banco de dados do RPM. Um simples "**.** " indica um teste bem sucedido. Os seguintes caracteres denotam falha em algum teste:

5 Soma MD5

S Tamanho do arquivo

L Vínculo simbólico (link)

T Mtime

D Dispositivo

U Usuário

G Grupo

M Modo (inclui permissões e tipo de arquivo)

VALIDAÇÃO DE ASSINATURA

A forma geral de um comando de validação de assinatura é

rpm –checksig <arquivo_pacote>+

Isto valida a assinatura PGP do pacote para garantir a integridade e origem do pacote. A configuração do PGP é lida a partir de /etc/rpmrc. Veja a seção ASSINATURAS PGP para detalhes.

OPÇÕES DE DESINSTALAÇÃO

A forma geral de um comando de desinstalação é

rpm -e <nome_do_pacote>+

–allmatches Remove todas as versões do pacote que casarem com *<nome_do_pacote>* . Normalmente um erro é exibido se *<nome_do_pacote>* casar com múltiplos pacotes.

–noscripts Não executa os scripts de pré e pós desinstalação.

–nodeps Não verifica se dependências serão quebradas antes de desinstalar o pacote.

–test Não desinstala nada, apenas simula todos os movimentos (O que conta é a emoção!). opções **-vv** .

OPÇÕES DE CONSTRUÇÃO

A forma geral de um comando de construção é

rpm -[b|t]*O* [opções-de-construção] <spec_do_pacote>+

O parâmetro usado é **-b** se um arquivo spec (especificações) está sendo usado para construir o pacote e **-t** se o rpm deve procurar dentro de um arquivo tar gzipado, .tar.gz (ou compactado com *compress* , .tar.Z) pelo arquivo de especificações a utilizar. Depois do primeiro parâmetro, o próximo argumento (*O*) especifica os estágios de construção que devem ser feitos, sendo um entre as seguintes possibilidades:

-bp Executa o estágio "%prep" do arquivo de especificações. Normalmente isso envolve a descompactação do código fonte e a aplicação de qualquer patch (modificações).

-bl Faz uma "verificação de lista". A seção "%files" do arquivo de especificações é expandida, e testes são feitos para garantir que os arquivos existam.

-bc Executa o estágio "%build" do arquivo de especificações (depois de passar pelo estágio prep). Isto geralmente envolve algo equivalente a um "make".

-bi Executa o estágio "%install" do arquivo de especificações (depois de passar pelos estágios prep e build). Isto geralmente envolve algo equivalente a um "make install".

-bb Constrói um pacote binário (depois de passar pelos estágios prep, build e install).

-ba Constrói os pacotes binário e fonte (depois de passar pelos estágios prep, build e install).

As seguintes opções também podem ser utilizadas:

–short-circuit Pula direto para o estágio especificado (isto é, pula todos os estágios precedendo o estágio especificado). Válida somente com **-bc** e **-bi** .

–timecheck Define o tempo de "verificação de tempo" (0 para desativar). Este valor também pode ser definido no arquivo de configuração (rpmrc) com "timecheck:". O valor da verificação de tempo expressa, em segundos, a idade máxima de um arquivo sendo empacotado. Avisos serão impressos sobre todos os arquivos mais velhos que a verificação de tempo.

–clean Remove a árvore de construção depois que os pacotes são feitos.

–rmsource Remove os arquivos fonte e o arquivo de especificações depois de construir o pacote (também pode ser usado sozinho, ex. "**rpm –rmsource bla.spec** ").

–test Não executa nenhum estágio de construção (build). Útil pra testar arquivos de especificação.

–sign Inclui uma assinatura PGP no pacote. Esta assinatura pode ser utilizada para verificar a integridade e a origem do pacote. Veja a seção ASSINATURAS PGP para os detalhes do /etc/rpmrc.

OPÇÕES DE RECONSTRUÇÃO E RECOMPILAÇÃO

Estas são duas outras formas de chamar o rpm:

rpm –recompile <arquivo_de_pacote_fonte>+

rpm –rebuild <arquivo_de_pacote_fonte>+

Quando chamado desta forma, o rpm instala o pacote fonte nomeado, e então o prepara, compila, e instala. Além disso, –**rebuild** constrói um novo pacote binário. Quando a construção terminar, o diretório usado para isso é removido (como em –**clean**) e os fontes e o arquivo de especificações são removidos.

ASSINANDO UM RPM EXISTENTE

rpm –resign <arquivo_de_pacote_binário>+

Esta opção gera e insere uma nova assinatura para os pacotes listados. Quaisquer assinaturas existentes são removidas.

ASSINATURAS PGP

Para utilizar as opções de assinatura do RPM você deve ser capaz de executar o PGP (ele deve estar instalado, e no seu caminho de procura), e ele deve ser capaz de encontrar um chaveiro (key ring) com as chaves públicas do RPM nele. Normalmente, o RPM usa os padrões do PGP para localizar os chaveiros (obedecendo o PGPPATH). Se seus chaveiros não estão localizados onde o PGP espera, sua localização deve ser especificada no arquivo /etc/rpmrc.

pgp_path Substitui /usr/lib/rpm. Deve conter os seus chaveiros.

Se você quer ser capaz de assinar os pacotes que você mesmo cria, você também vai precisar criar seu próprio par de chaves pública e privada (veja o manual do PGP). Além das entradas no /etc/rpmrc citadas acima, você deve adicionar as seguintes:

signature O tipo de assinatura. Por enquanto apenas pgp é suportado.

pgp_name O nome do "usuário" com cuja chave você deseja assinar seus pacotes.

Quando construir pacotes você então adiciona a opção –sign à linha de comando. Você será perguntado sobre a sua frase secreta (pass phrase), e o pacote será construído e assinado.

OPÇÕES DE RECONSTRUÇÃO DO BANCO DE DADOS

A forma geral de um comando de reconstrução do banco de dados é

rpm –rebuilddb

As únicas opções que este modo suporta são **–dbpath** and **–root** .

EXIBIR CONFIGURAÇÃO

Executar

rpm –showrc

exibe os valores que o RPM irá utilizar para todas as opções que podem estar definidas em seus arquivos de configuração.

OPÇÕES DE FTP

O RPM inclui um programa de FTP simples para facilitar a instalação e consulta de pacotes que estão disponíveis na Internet. Arquivos de pacotes para instalação, atualização ou consulta podem ser especificados como uma URL no estilo ftp:

ftp://<usuário>:<senha>@máquina/caminho/para/o/pacote.rpm

Se a parte **:senha** for omitida, a senha será perguntada (uma vez para cada par usuário/máquina). Se o usuário e a senha forem omitidos, ftp anônimo será usado. Em todos os casos, transferência passiva (PASV) será usada.

O RPM permite que as seguintes opções sejam usadas com URLs de ftp:

–ftpproxy <máquina> A máquina <máquina> será usada como servidor proxy para todas as transferências, permitindo que usuários utilizem o ftp através de firewalls que utilizem sistemas de proxy. Esta opção também pode ser especificada em um arquivo de configuração.

–ftpport <porta> Especifica o número da porta TCP a utilizar para a conexão ftp ao invés da porta padrão (21). Esta opção também pode ser especificada em um arquivo de configuração.

ARQUIVOS

/etc/rpmrc

/.rpmrc

/var/lib/rpm/packages

/var/lib/rpm/pathidx

/var/lib/rpm/nameidx

/tmp/rpm*

VEJA TAMBÉM

glint(8), **rpm2cpio**(8),

http://www.redhat.com/rpm

http://www.rpm.org

http://lie-br.conectiva.com.br

AUTORES

Marc Ewing <marc@redhat.com>

Erik Troan <ewt@redhat.com>

E.31 securetty

NOME

securetty - arquivo que lista os terminais (ttys) nos quais o superusuário (root) pode acessar o sistema.

DESCRIÇÃO

/etc/securetty é usado por **login(1)**; o arquivo contém as linhas com os nomes de dispositivos terminais (um por linha, sem a especificação de */dev/*) através dos quais o superusuário pode acessar o sistema.

ARQUIVOS

/etc/securetty

VEJA TAMBÉM

login(1)

E.32 setfdprm

NOME

setfdprm - configura os parâmetros de disquetes a partir dos dados fornecidos pelo usuário

SINOPSE

setfdprm [-p] dispositivo

setfdprm [-p] dispositivo tamanho setores cabeças stretch gap rate spec1 fmt_gap

setfdprm [-c] dispositivo

setfdprm [-y] dispositivo

setfdprm [-n] dispositivo

DESCRIÇÃO

setfdprm é um utilitário que pode ser utilizado para a carga de parâmetros de discos nos arquivos dos dispositivos de autodetecção de unidades de disquetes, para eliminar parâmetros antigos e para habilitar ou desabilitar mensagens de diagnósticos.

Sem opções, **setfdprm** carrega o *dispositivo* (normalmente */dev/fd0* ou */dev/fd1*) com um novo parâmetro de configuração, através da entrada do *nome* encontrado em */etc/fdprm* (normalmente chamada 360/360, etc...). Estes parâmetros são mantidos até que a mídia seja mudada.

-p*dispositivo* Carrega permanentemente um novo parâmetro para a autoconfiguração da unidade de disquetes cujo *nome* esteja em */etc/fdprm*. Alternativamente, os parâmetros podem ser informados diretamente na linha de comando.

-c *dispositivo* Elimina os parâmetros configurados para a unidade de disquetes especificada.

-y *dispositivo* Habilita as mensagens de detecção de formatos para a unidade de disquetes especificada.

-n *dispositivo* Desabilita as mensagens de detecção de formatos para a unidade de disquetes especificada.

PROBLEMAS

Esta documentação ainda não está completa.

ARQUIVOS

/etc/fdprm

AUTOR

Werner Almesberger (almesber@nessie.cs.id.ethz.ch)

E.33 shadow

NOME

shadow - arquivo de senhas criptografadas

DESCRIÇÃO

shadow contém as informações de senhas criptografadas das contas dos usuários e opcionalmente a informação de idade da senha. Contém:

Nome de acesso.

Senha criptografada.

Dias decorridos entre 1 de janeiro de 1970 e a última alteração da senha.

Número de dias até que a senha deva ser alterada.

Número de dias após o qual a senha deve ser alterada.

Número de dias antes da expiração da senha no qual o usuário será avisado.

Número de dias após a expiração da senha que a conta deve ser desabilitada.

Dias decorridos entre 1 de janeiro de 1970 e a data em que a conta foi desabilitada.

Campo reservado.

O campo senha deve ser preenchido. A senha criptografada consiste de 13 a 24 caracteres entre os 64 caracteres alfabéticos - de a até z e de A até Z, além de 0 a 9, . e /. Verifique em **crypt(3)** para maiores detalhes de como esta cadeia de caracteres é interpretada.

A data da última mudança da senha é dada pelo número de dias desde 1 de janeiro de 1970. A senha não pode ser alterada novamente até que um determinado número de dias tenha se passado, e deve ser alterada após um número máximo de dias. Se o número mínimo de dias for maior que o número máximo, a senha não pode ser alterada pelo usuário.

Uma conta é considerada inativa e desabilitada se a senha não foi alterada dentro de um determinado número de dias após a expiração da senha. Uma conta poderá ser desabilitada ainda no dia especificado, independentemente de qualquer informação de expiração da senha.

Esta informação sobrepõe-se a qualquer senha ou idade de senha presente no arquivo.

Este arquivo não poderá ser acessado por usuários comuns, caso deseje manter a segurança das senhas.

ARQUIVOS

/etc/passwd - informações das contas de usuários

/etc/shadow - senhas de usuários criptografadas

VEJA TAMBÉM

chage(1), **login(1)**, **passwd(1)**, **su(1)**, **sulogin(8)**, **shadow(3)**, **passwd(5)**, **pwconv(8)**, **pwunconv(8)**

AUTOR

Julianne Frances Haugh (jfh@tab.com)

E.34 shells

NOME

shells - rotas de shells de login válidos

DESCRIÇÃO

/etc/shells é um arquivo texto que contém os nomes-de-caminho completos dos shells de login válidos. Esse arquivo é consultado por **chsh(1)** e fica disponível para consulta por outros programas.

EXEMPLOS

/etc/shells pode conter, por exemplo, os seguintes caminhos:

/bin/bash

/bin/sh

/bin/csh

ARQUIVOS

/etc/shells

VEJA TAMBÉM

chsh(1)

E.35 shutdown

NOME

shutdown - desliga o sistema

SINOPSE

/sbin/shutdown [**-t** *sec*] [**-rkhncfF**] *tempo* [*msg-de-aviso*]

DESCRIÇÃO

shutdown desliga o sistema de uma forma segura. Todos os usuários que estiverem acessando o sistema serão notificados que o sistema está prestes a ser desligado, e o programa **login (1)** é bloqueado. É possível desligar o sistema imediatamente ou após um determinado intervalo de tempo. Todos os processos são inicialmente notificados que o sistema será desligado através do sinal SIGTERM. Isso permite que programas, como por exemplo o **vi (1)** tenham tempo de salvar arquivos que estejam sendo editados, mensagens e programas de processamento de notícias sejam finalizados normalmente, etc... **shutdown** executa esta atividade através da sinalização ao processo init, solicitando a mudança de nível de execução. O nível de execução **0** é usado para desligar o sistema, nível de execução **6** é usado para reinicializar o sistema em um estado onde as tarefas administrativas podem ser executadas; este é o padrão se os indicadores *-h* ou *-r* são informados em conjunto com o programa **shutdown** . Para verificar quais ações são tomadas ao se desligar ou reinicializar o sistema para estes níveis de execução verifique o arquivo */etc/inittab* .

OPÇÕES

-t *seg*	Indica a **init (8)** para aguardar *seg* segundos entre o processo de envio de sinal aviso e o sinal de finalização, antes de mudar de nível de execução.
-k	Não desliga realmente o sistema; somente envia mensagens de aviso a todos os usuários.
-r	Reinicializar após o desligamento do sistema.
-h	Desliga o sistema após a execução do comando.
-n	[ATENÇÃO] Não aciona **init (8)** para desligar o sistema, mas faz o encerramento por si só. O uso desta opção é desencorajada, e seus resultados são imprevisíveis.
-f	Não executa fsck na reinicialização do sistema.
-F	Força a execução do fsck na reinicialização do sistema.
-c	Cancela a execução de um programa shutdown. Com esta opção obviamente não é possível especificar o argumento **tempo** , mas pode-se informar uma mensagem explicativa na linha de comando, a qual será enviada para todos os usuários.
tempo	Quanto o sistema deverá ser executado.
msg-de-aviso	Mensagem a ser enviada a todos os usuários.

O argumento *tempo* pode ter diferentes formatos. Primeiro, ele pode ser informado em um formato absoluto no formato *hh:mm* , na qual *hh* é a hora (com 1 ou 2 dígitos) e *mm* são os minutos da hora (com dois dígitos). O segundo formato tem o formato $+\ m$, no qual m é o número de minutos a serem aguardados. A palavra **now** é um nome alternativo para **+0** .

O indicador **-f** significa 'reinicialização rápida'. Isso somente cria um arquivo de avisos */fastboot* que pode ser testado pelo sistema quando ele reinicializa. O programa rc de inicialização testa a presença deste arquivo, e decide não executar o programa **fsck (1)** desde que o sistema tenha sido desligado adequadamente. Após isso o sistema remove o arquivo */fastboot* .

Os indicadores **-F** significam 'forçar fsck'. Isso somente cria um arquivo de aviso */forcefsck* o qual pode ser testado pelo sistema na sua reinicialização. O programa rc de inicialização testa a presença do arquivo, e decide executar **fsck (1)** com um indicador especial de 'forçado', fazendo com que inclusive sistemas de arquivos desmontados sejam checados. Após a finalização do processo o sistema remove o arquivo */forcefsck* .

O indicador **-n** faz com que o programa **shutdown** não chame **init**, mas finalize todos os processos que estejam sendo executados. **shutdown** inibirá então a quota, a contabilização, a área de troca (swap) e todos os sistemas de arquivos desmontados.

CONTROLE DE ACESSO

shutdown pode ser chamado a partir do programa **init (8)** quando as teclas mágicas **CTRL-ALT-DEL** são pressionadas, através da criação de uma entrada apropriada no arquivo */etc/inittab*. Isso significa que qualquer um que tenha acesso ao teclado pode desligar o sistema. Para prevenir isso, **shutdown** pode verificar se um usuário autorizado está acessando o sistema através de uma console virtual. Caso **shutdown** seja acionado por **init (8)**, ele verifica se o arquivo */etc/shutdown.allow* está presente. Ele então compara o nome de acesso com a lista de pessoas que estão conectadas ao sistema através de uma console virtual (através de */var/run/utmp*). Somente se alguns dos usuários autorizados **ou o superusuário** estiverem acessando o sistema, o sistema será desligado. De outra forma será apresentada a mensagem

Português: **shutdown: nenhum usuário autorizado está acessando o sistema**

Inglês: **shutdown: no authorized users logged in**

na console do sistema. O formato do arquivo */etc/shutdown.allow* é de um usuário por linha. Linhas vazias e linhas comentadas (com o prefixo #) são permitidos. Atualmente há um limite de 32 usuários neste arquivo.

ARQUIVOS

/fastboot

/etc/inittab

/etc/init.d/halt

/etc/init.d/reboot

/etc/shutdown.allow

PROBLEMAS

Não se trata exatamente de um problema, mas muitos usuários esquecem de fornecer o argumento *tempo* e ficam em dúvida quando uma mensagem de erro de **shutdown** é apresentada. O argumento *tempo* é obrigatório; em 90 por cento dos casos ele pode ser informado através da palavra **now** - agora.

AUTOR

Miquel van Smoorenburg, miquels@cistron.nl

VEJA TAMBÉM

fsck(8), **init**(1), **halt**(8), **reboot**(8)

E.36 su

NOME

su - executa um interpretador de comandos com substituição de usuário e grupo.

SINOPSE

su [-flmp] [-c comando] [-s interpretador] [–login] [–fast] [–preserve-environment] [–command=comando] [–shell=interpretador] [-] [–help] [–version] [usuário [arg...]]

DESCRIÇÃO

Esta documentação não é mais mantida e pode estar desatualizada ou incompleta. A documentação Textinfo é agora a fonte indicada.

Esta página de manual documenta a versão GNU do comando **su**. **su** permite que um usuário torne-se outro temporariamente. Ele executa um interpretador com a identificação real e efetiva de usuário, identificação de grupo e grupos suplementares do usuário. Caso usuário não seja informado, o padrão é o superusuário (root). O interpretador executado é o especificado para o usuário no arquivo passwd, ou /bin/sh caso nenhum seja especificado. Caso USUÁRIO tenha senha, **su** solicita a senha a menos que se tenha a identificação real de usuário igual a 0 (superusuário).

Por padrão, **su** não altera o diretório atual. Ele configura as variáveis de ambiente 'HOME' e 'SHELL' a partir da entrada em passwd para usuário, e caso usuário não seja o superusuário, configura 'USER' e 'LOGNAME' para usuário. Por padrão, o interpretador não é um interpretador de acesso tradicional.

Caso um ou mais ARGs sejam fornecidos, eles são passados como argumentos adicionais para o interpretador.

su não administra /bin/sh ou outros interpretadores de forma especial (configurando argv[0] para -su", passando -c somente para certos interpretadores, etc.).

Em sistemas que tenham syslog, **su** pode ser compilado para comunicar falhas e opcionalmente comandos **su** bem sucedidos.

OPÇÕES

-c COMMAND, –command=COMANDO Envia COMANDO, uma linha de comando simples para ser executada, através da opção *-c* ao invés de iniciar um interpretador interativo.

-f, –fast Envia a opção *-f* para o interpretador de trabalho. Isso provavelmente faz sentido somente com **csh** e **tcsh**, para os quais a opção *-f* evita a leitura dos arquivos de inicialização (.cshrc). Com interpretadores similares a Bourne, a opção *-f* desabilita a expansão de padrões de nomes de arquivos, o que normalmente não é algo desejável.

–help Lista uma mensagem de ajuda na saída padrão e finaliza.

-, -l, –login Torna o interpretador um interpretador de acesso. Isso significa limpar todas as variáveis de interpretador, exceto 'TERM', 'HOME', e 'SHELL' (as quais são descritas acima), e 'USER' e 'LOGNAME' (as quais são configuradas, mesmo para o superusuário, conforme descrito acima), e configura 'PATH' para um valor predefinido no programa. Altera a localização para o diretório pessoal do usuário. Anexando-se "-" ao nome do interpretador, provoca a leitura do arquivo de acesso ao sistema.

-m, -p, –preserve-environment Não altera as variáveis de ambiente 'HOME', 'USER', 'LOGNAME', ou 'SHELL'. Executa o interpretador fornecido na variável de ambiente 'SHELL' ao invés do interpretador definido para o usuário no /etc/passwd, a menos que o usuário que execute **su** não seja o superusuário e o interpretador do usuário seja restrito. Um interpretador restrito é um não listado no arquivo /etc/shells, ou em uma lista compilada caso o arquivo não exista. Partes destas opções podem ser sobrepostas pelo parâmetros *–login* ou *–shell*.

-s, –shell interpretador Executar INTERPRETADOR ao invés do interpretador definido em /etc/passwd, a menos que o usuário que esteja executando **su** não seja o superusuário e o interpretador do usuário não seja restrito.

–version Lista a versão do programa na saída padrão e termina normalmente.

E.37 sulogin

NOME

sulogin - acesso em modo monousuário

SINOPSE

sulogin [**-t timeout**] [**tty-device**]

DESCRIÇÃO

sulogin é chamado pelo processo **init(8)** quando o sistema entra em modo monousuário (isso é feito através de uma entrada no *inittab(5)*). **Init** também tenta executar *sulogin* quando é passado o indicador **-b** de um monitor de inicialização (por exemplo: LILO).

Ao usuário é mostrado:

português

Informe a senha do root para manutenção do sistema (ou digite Control-D para inicialização normal):

inglês

Give root password for system maintenance (or type Control-D for normal startup):

sulogin irá conectar o terminal atual, ou o dispositivo opcional informado na linha de comando (tipicamente **/dev/console**).

Após o usuário sair do ambiente monousuário ou ao pressionar control-d na linha de comando, o sistema irá inicializar o sistema no nível de execução padrão.

VARIÁVEIS DE AMBIENTE

sulogin procura pela variável de ambiente **SUSHELL** ou **sushell** para determinar qual interpretador deve ser iniciado. Caso a variável de ambiente não esteja configurada, o comando tentará executar o interpretador do superusuário definido no **/etc/passwd** . Caso isso falhe, o sistema executará o **/bin/sh** .

Isso é muito útil em conjunto com a opção **-b** do init. Para inicializar o sistema em modo monousuário, com o sistema de arquivos raiz montado para gravação e leitura, usando um interpretador especial estaticamente ligado (este exemplo é válido para a linha de comandos de inicialização do LILO).

boot: linux -b rw sushell=/sbin/sash

ARQUIVOS

sulogin examina os arquivos abaixo para encontrar a senha do superusuário. Caso eles estejam danificados, ou não existam, ele usará um método de diminuição de exigências, até chegar a um ambiente de linha de comando, sem perguntar pela senha do superusuário casos eles estejam irremediavelmente danificados.

/etc/passwd,

/etc/shadow (se presente)

AUTOR

Miquel van Smoorenburg <miquels@cistron.nl>

VEJA TAMBÉM

init(8), inittab(5).

E.38 swapon

NOME

swapon, swapoff - habilita e desabilita dispositivos e arquivos para paginação e troca

SINOPSE

/sbin/swapon [-h -V]

/sbin/swapon -a [-v]

/sbin/swapon [-v] [-p *prioridade***]** *arquivo_especial* **...**

/sbin/swapon [-s]

/sbin/swapoff [-h -V]

/sbin/swapoff -a

/sbin/swapoff *arquivo_especial* **...**

DESCRIÇÃO

Swapon é usado para especificar dispositivos nos quais paginações e trocas serão efetuadas. Chamadas a **swapon** normalmente ocorrem através do arquivo de inicialização do sistema em modo multiusuário */etc/rc* , o qual torna as áreas de troca disponíveis, tornando as atividades de paginação e troca disponíveis para diversos arquivos e dispositivos.

Normalmente a primeira forma é usada:

-h Apresenta uma mensagem de ajuda.

-V Lista a versão do programa.

-s Lista um resumo do uso da área de troca por dispositivo.

-a Todos os dispositivos marcados como "sw" - dispositivos de troca em */etc/fstab* ficam disponíveis.

-p *prioridade* Prioridade especial para **swapon**. Esta opção está disponível somente se **swapon** foi compilado antes e é usado com a versão 1.3.2 ou posterior. *Prioridade* é um valor entre 0 e 32767. Veja **swapon(2)** para uma descrição completa das prioridades de troca. Adicione **pri**=*valor* ao campo de opção de */etc/fstab* para uso com **swapon -a**.

Swapoff desabilita a troca em dispositivos e arquivos especificados, ou em todas as áreas de troca especificadas em */etc/fstab* quando a opção **-a** for informada.

VEJA TAMBÉM

swapon(2), **swapoff**(2), **fstab**(5), **init**(8), **mkswap**(8), **rc**(8), **mount**(8)

ARQUIVOS

/dev/hd?? Dispositivos padrão de paginação

/dev/sd?? Dispositivos SCSI padrão de paginação

/etc/fstab tabela de descrição do sistema de arquivos em formato ASCII

HISTÓRICO

O comando **swapon** apareceu na versão 4.0BSD.

AUTORES

Veja a página de manual **mount(8)** para uma lista completa de autores. Contribuições principais por Doug Quale, H. J. Lu, Rick Sladkey, e Stephen Tweedie.

E.39 tune2fs

NOME

tune2fs - ajusta parâmetros do sistema de arquivos second extended

SINOPSE

tune2fs [**-l**] [**-c** *montagem-máxima*] [**-e** *comportamento-em-erro*] [**-i** *intervalo-entre-verificações*] [**-m** *porcentagem-de-blocos-reservados*] [**-r** *número-de-blocos-reservados*] [**-s marca-super-esparso**] [**-u** *usuário*] [**-g** *grupo*] [**-C** *número-de-montagens*] [**-L** *nome-do-volume*] [**-M** *diretório-da-última-montagem*] [**-U** *UUID*] dispositivo

DESCRIÇÃO

tune2fs ajusta parâmetros em um sistema de arquivos Linux second extended

Nunca use tune2fs em uma partição montada para leitura/gravação para mudar parâmetros!

OPÇÕES

-*c* montagem-máxima Ajusta o número máximo de montagens entre duas ve-rificações do sistema de arquivos.

-*e* comportamento-em-erro Muda o comportamente do código do kernel quando erros são detectados. *comportamento-em-erro* pode ser um dos seguintes:

continue Continua a execução normal.

remount-ro Remonta o sistema de arquivos somente para leitura.

panic Causa pânico do kernel.

-g grupo Define o grupo de usuários que pode se beneficiar dos blocos reservados.

grupo pode ser um gid numérico ou o nome de um grupo.

-i intervalo-entre-verificações[d|m|w] Ajusta o tempo máximo entre duas verificações do sistema de arquivos. Nenhum sufixo ou 'd' resulta em dias, 'm' em meses e 'w' em semanas. Um valor de zero vai desabilitar a verificação dependente do tempo.

-l Lista o conteúdo do superbloco do sistema de arquivos.

-m número-de-blocos-reservados Ajusta a porcentagem de blocos reservados no dispositivo dado.

-r contagem-blocos-reservados Ajusta a quantidade de blocos reservados no dispositivo dado.

-s marca-super-esparso Ajusta e reinicia a marca superbloco-esparso. A característica superbloco-esparso economiza espaço em sistemas de arquivos realmente grandes. **Aviso:** O kernel do Linux 2.0 não suporta corretamente esta característica, nem todos os kernels do Linux 2.1; por favor não use isto a não ser que você saiba o que está fazendo!

-u usuário Define o usuário que pode se beneficiar dos blocos reservados. *usuário* pode ser um uid numérico ou um nome de usuário.

-C número-de-montagens Define o número de vezes que o sistema de arquivos foi montado.

-L rótulo-do-volume Define o rótudo do volume no sistema de arquivos.

-M diretório-da-última-montagem Define o diretório da última montagem do sistema de arquivos.

-U UUID Define o UUID do sistema de arquivos. Um exemplo de UUID parece assim: "c1b9d5a2-f162-11cf-9ece-0020afc76f16". O uuid também pode ser "null", o que define o UUID do sistema de arquivos como um UUID nulo. O uuid também pode ser "random", o que gera um UUID aleatório para o sistema de arquivos.

FALHAS

Não achamos nenhuma falha ainda. Talvez elas existam, mas é improvável.

AVISO

Use esta utilidade por sua conta e risco. Você está modificando sistemas de arquivos.

AUTOR

tune2fs foi escrito por Remy Card <card@masi.ibp.fr>, o desenvolvedor e mantenedor do sistema de arquivos ext2.

tune2fs usa a biblioteca ext2fs escrita por Theodore Ts'o <tytso@mit.edu>.

Esta página de manual foi escrita por Christian Kuhtz <chk@data-hh.Hanse.DE>.

Verificação dependente do tempo foi adicionada por Uwe Ohse <uwe@tirka.gun.de>.

DISPONIBILIDADE

tune2fs está disponível através de ftp anônimo em ftp.ibp.br e tsx-11.mit.edu no diretório /pub/linux/packages/ext2fs.

VEJA TAMBÉM

dumpe2fs(8), **e2fsck**(8), **mke2fs**(8)

E.40 umount

NOME

umount - desmonta sistemas de arquivos

SINOPSE

umount [-hV]

umount -a [-nrv] [-t *vfstype***]**

umount [-nrv] *dispositivo* | *dir* **[...]**

DESCRIÇÃO

O comando **umount** retira o sistema de arquivos indicado da hierarquia de arquivos. Um sistema de arquivos é especificado tanto através da informação do diretório onde ele foi montado, quanto pelo nome do dispositivo onde ele reside.

Note que um sistema de arquivos não pode ser desmontado quando ele está ocupado - por exemplo quando há arquivos abertos ou quando alguns processos tenham seu diretório de trabalho nele, ou quando um arquivo de swap esteja em uso. O processo pode ser o próprio **umount** pois ele abre a libc, e esta por sua vez pode abrir arquivos locais.

Opções do comando **umount :**

-V Apresenta a versão do comando e finaliza.

-h Imprime a mensagem de ajuda e finaliza.

-v Modo de apresentação de mensagens.

-n Desmontar sem escrever em */etc/mtab*.

-r No caso da desmontagem falhar, tenta remontar somente para leitura.

-a Todos os sistemas de arquivos descrito em */etc/mtab* são desmontados (com **umount** versão 2.7 ou posterior: o sistema de arquivos *proc* não é desmontado).

-t *vfstype* Indica que as ações podem ser realizadas nos sistemas de arquivos do tipo especificado. Mais de um tipo pode ser especificado, separado por vírgulas. A lista de tipos de sistemas de arquivos pode ter um prefixo **no** para especificar os tipos de sistemas de arquivos nos quais as ações não podem ser exercidas.

DISPOSITIVO de LOOP

O comando **umount** irá liberar um dispositivo de loop (se houver) associado com o mount, no caso de encontrar a opção 'loop=...' em */etc/mtab*. Qualquer dispositivo de loop podem ser liberado através do comando 'losetup -d', veja **losetup(8)**.

ARQUIVOS

/etc/mtab tabela dos sistemas de arquivos montados

VEJA TAMBÉM

umount(2), **mount(8)**, **losetup(8)**.

HISTÓRICO

O comando **umount** apareceu na Versão 6 do AT&T Unix.

Índice Remissivo