

Linux

Fundamentos

Linux

Fundamentos



EDITORIA AFILIADA

Seja Nosso Parceiro no Combate à Cópia Ilegal

A cópia ilegal é crime. Ao efetuá-la, o infrator estará cometendo um grave erro, que é inibir a produção de obras literárias, prejudicando profissionais que serão atingidos pelo crime praticado.

Junte-se a nós nesta corrente contra a pirataria. Diga não à cópia ilegal.

Seu Cadastro É Muito Importante para Nós

Ao preencher e remeter a ficha de cadastro constante em nosso site, você passará a receber informações sobre nossos lançamentos em sua área de preferência.

Conhecendo melhor nossos leitores e suas preferências, vamos produzir títulos que atendam suas necessidades.

Obrigado pela sua escolha.

Fale Conosco!

Eventuais problemas referentes ao conteúdo deste livro serão encaminhados ao(s) respectivo(s) autor(es) para esclarecimento, excetuando-se as dúvidas que dizem respeito a pacotes de softwares, as quais sugerimos que sejam encaminhadas aos distribuidores e revendedores desses produtos, que estão habilitados a prestar todos os esclarecimentos.

Os problemas só podem ser enviados por:

1. E-mail: producao@erica.com.br
2. Fax: (11) 2097.4060
3. Carta: Rua São Gil, 159 - Tatuapé - CEP 03401-030 - São Paulo - SP



Walace Soares

Gabriel Fernandes

Linux

Fundamentos

1^a Edição

São Paulo

Editora Érica Ltda.

Copyright © 2010 da Editora Érica Ltda.

Todos os direitos reservados. Proibida a reprodução total ou parcial, por qualquer meio ou processo, especialmente por sistemas gráficos, microfílmicos, fotográficos, reprodutivos, fonográficos, videográficos, internet, e-books. Vedada a memorização e/ou recuperação total ou parcial em qualquer sistema de processamento de dados e a inclusão de qualquer parte da obra em qualquer programa juscibernético. Essas proibições aplicam-se também às características gráficas da obra e à sua editoração. A violação dos direitos autorais é punível como crime (art. 184 e parágrafos, do Código Penal, conforme Lei nº 10.695, de 07/01/2003) com pena de reclusão, de dois a quatro anos, e multa, conjuntamente com busca e apreensão e indenizações diversas (artigos 102, 103 parágrafo único, 104, 105, 106 e 107 itens 1, 2 e 3 da Lei nº 9.610, de 19/06/1998, Lei dos Direitos Autorais).

Os Autores e a Editora acreditam que todas as informações aqui apresentadas estão corretas e podem ser utilizadas para qualquer fim legal. Entretanto, não existe qualquer garantia, explícita ou implícita, de que o uso de tais informações conduzirá sempre ao resultado desejado. Os nomes de sites e empresas, porventura mencionados, foram utilizados apenas para ilustrar os exemplos, não tendo vínculo nenhum com o livro, não garantindo a sua existência nem divulgação. Eventuais erratas estarão disponíveis para download no site da Editora Érica.

Conteúdo adaptado ao Novo Acordo Ortográfico da Língua Portuguesa, em execução desde 1º de janeiro de 2009.

"Algumas imagens utilizadas neste livro foram obtidas a partir do CorelDRAW 12, X3 e X4 e da Coleção do MasterClips/MasterPhotos® da IMSI, 100 Rowland Way, 3rd floor Novato, CA 94945, USA."

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

Soares, Wallace.

Linux: Fundamentos / Wallace Soares, Gabriel Fernandes -- 1. ed. -- São Paulo: Érica, 2010.

Bibliografia.

ISBN 978-85-365-1176-4

1. Linux (Sistema operacional de computador) 2. Redes de computadores I. Fernandes, Gabriel II. Título.

10-12002

CDD-005.43

Índices para catálogo sistemático

1. Linux: Sistema Operacional: Computadores: Processamento de dados 005.43

Coordenação Editorial:	Rosana Arruda da Silva
Capa:	Maurício S. de França
Editoração e Finalização:	Adriana Aguiar Santoro Roseane Gomes Sobral Marlene Teresa S. Alves Carla de Oliveira Moraes Rosana Ap. A. dos Santos

Editora Érica Ltda.

Rua São Gil, 159 - Tatuapé

CEP: 03401-030 - São Paulo - SP

Fone: (11) 2295-3066 - Fax: (11) 2097-4060

www.editoraerica.com.br

Fabricante

Produto: **Linux**

Fabricante: centOS Team

Site: www.centos.org

Requisitos de Hardware e de Software

- » Sistema operacional CentOS 5 ou superior;
- » CentOS 5.4 ou superior;
- » GNOME 1.16 ou superior;
- » Pentium IV ou superior;
- » 512 MB de memória RAM;
- » Disco rígido de 20 GB ou superior;
- » Monitor de vídeo com resolução mínima de 1024x768 pontos;
- » Unidade de DVD-ROM;
- » Modem ou placa de rede e acesso à Internet.

Sobre o Material Disponível na Internet

O material disponível no site da Editora Érica contém as respostas dos exercícios do livro. Para sua utilização, é necessário ter instalado o Acrobat Reader 5 ou superior.

Respostas.exe

Procedimento para download

Acesse o site da Editora Érica Ltda.: www.editoraerica.com.br. A transferência do arquivo disponível pode ser feita de duas formas:

- » **Por meio do módulo pesquisa.** Localize o livro desejado, digitando palavras-chave (nome do livro ou do autor). Aparecem os dados do livro e o arquivo para download. Com um clique o arquivo executável é transferido.
- » **Por meio do botão "Download".** Na página principal do site, clique no item "Download". É exibido um campo no qual devem ser digitadas palavras-chave (nome do livro ou do autor). Aparecem o nome do livro e o arquivo para download. Com um clique o arquivo executável é transferido.

Procedimento para descompactação

Primeiro passo: após ter transferido o arquivo, verifique o diretório em que se encontra e dê um duplo clique nele. Aparece uma tela do programa WINZIP SELF-EXTRACTOR que conduz ao processo de descompactação. Abaixo do Unzip To Folder há um campo que indica o destino do arquivo que será copiado para o disco rígido do seu computador.

C:\Linux

Segundo passo: prossiga a instalação, clicando no botão Unzip, o qual se encarrega de descompactar o arquivo. Logo abaixo dessa tela aparece a barra de status que monitora o processo para que você acompanhe. Após o término, outra tela de informação surge, indicando que o arquivo foi descompactado com sucesso e está no diretório criado. Para sair dessa tela, clique no botão OK. Para finalizar o programa WINZIP SELF-EXTRACTOR, clique no botão Close.

Dedicatória

À minha esposa Mara e às minhas filhas Carol e Isabelle.

Walace Soares

À minha esposa Raquel, por sua compreensão;

Aos meus filhos Nicolas e Maria Cecilia, que são a razão da minha vida;

Ao meu pai Mauri, irmão Leonardo e meus amigos pelo apoio em todos os momentos desta importante etapa da minha vida;

Em memória à minha avó Benta Gertrudes.

Gabriel Fernandes

*"Bem-aventurado o homem que acha sabedoria,
e o homem que adquire conhecimento.
Porque melhor é a sua mercadoria do que a mercadoria
de prata, e a sua renda do que o ouro mais fino."
Prov. 3, 13-14*

Agradecimentos

A todo o pessoal envolvido no processo de construção do livro, pois são todos importantes e fundamentais para um trabalho benfeito;

Aos amigos e familiares, que compreenderam meu estresse durante as últimas semanas.

Walace Soares

Nada realmente bom se faz sozinho. Esta obra não foge a esta regra, pois fala de um sistema iniciado de forma singular, que hoje recebe a colaboração de vasta quantidade de pessoas.

Professores, amigos, família, gerentes, diretores, colaboradores e clientes contribuíram muito na minha trajetória e, consequentemente, na elaboração deste livro, seja criando oportunidades ou ajudando a encontrar soluções nos mais diversos problemas do dia a dia. Portanto, é praticamente impossível completar a lista de agradecimentos. Se tivesse de citar nomes nesta etapa, eles seriam muitos.

Entretanto, quero destacar o meu parceiro neste trabalho, Wallace Soares, que me deu esta oportunidade.

Gabriel Fernandes

Sumário

Capítulo 1 - A História.....	15
1.1 - UNIX.....	15
1.1.1 - A Filosofia UNIX.....	16
1.1.2 - Variantes do UNIX.....	19
1.2 - Linux	20
1.3 - Vantagens do Linux.....	21
1.4 - Projeto GNU	23
1.5 - Free Software Foundation	24
Exercícios.....	26
Capítulo 2 - Familiarização com o Linux.....	27
2.1 - Windows e Linux.....	27
2.2 - POSIX.....	27
2.3 - Kernel	28
2.4 - Shell	30
2.5 - Multitarefa.....	32
2.6 - Multusuário	33
2.7 - Modo de Execução.....	34
2.7.1 - Modo Usuário.....	34
2.7.2 - Modo Kernel.....	34
2.8 - Memória Virtual.....	35
2.9 - Linux Foundation.....	36
2.10 - Sistemas de Arquivos	37
2.10.1 - Partição.....	38
2.10.2 - Journaling.....	39
2.11 - Tipos de Arquivos	40
2.12 - Filesystem Hierarchy Standard (FHS).....	41
2.13 - As Distribuições do Linux.....	44
2.14 - Onde Encontrar Ajuda	47
2.14.1 - As Man Pages	47
2.14.2 - As Info Pages	49
2.14.3 - O Comando Whatis	50
2.14.4 - O Comando Apropos	50

2.14.5 - A Opção --Help	51
Exercícios.....	52
Capítulo 3 - Tarefas Essenciais do Administrador	54
3.1 - Comandos Básicos.....	54
3.1.1 - Obtenção da Ajuda do Linux	54
3.1.2 - Gerenciamento de Diretórios e Arquivos.....	55
3.2 - Gerenciamento de Grupos	62
3.2.1 - Adição de Grupos	62
3.2.2 - Remoção de Grupos	63
3.3 - Gerenciamento de Usuários	64
3.3.1 - Adição de um Novo Usuário	64
3.3.2 - Gerenciamento de Senha de um Usuário	65
3.3.3 - Alteração de um Usuário Cadastrado	66
3.3.4 - Remoção de um Usuário Cadastrado	67
3.4 - Gerenciamento de Programas.....	67
3.4.1 - RPM.....	67
3.4.2 - Apt.....	69
3.4.3 - Instalador Próprio.....	69
3.4.4 - Cópia.....	70
3.4.5 - Instalação de Pacotes	70
3.4.6 - Atualização de Pacotes	71
3.4.7 - Remoção de Pacotes	72
3.5 - Monitoramento do Sistema.....	73
3.5.1 - CPU	73
3.5.2 - Memória	77
3.5.3 - Disco	79
3.5.4 - Rede	80
3.6 - Arquivos de Mensagens e Logs do Sistema	82
3.7 - Serviços (Daemons)	84
3.7.1 - Chkconfig	84
3.7.2 - Service	86
3.8 - Gerenciamento de Processos	87
3.8.1 - Kill e killall.....	88
3.9 - Superusuário	89
Exercícios.....	93

Capítulo 4 - Instalação	94
4.1 - Introdução.....	94
4.2 - Preparação para Instalar	94
4.3 - Instalação	95
Exercícios.....	111
Capítulo 5 - Inicialização e Desligamento do Linux.....	112
5.1 - Inicialização.....	112
5.2 - Gerenciadores de Boot (Boot Loaders).....	113
5.2.1 - LILO.....	113
5.2.2 - GRUB	114
5.2.3 - Inicialização em Modo Monousuário ou de Manutenção	116
5.2.4 - Desligamento.....	118
Exercícios.....	122
Capítulo 6 - Sistemas de Arquivos	123
6.1 - Montagem e Desmontagem.....	123
6.2 - Tipos de Arquivo	126
Exercícios.....	129
Capítulo 7 - Shell como Ferramenta do Administrador	130
7.1 - O Que é o Shell	130
7.2 - Shell Script	131
7.3 - Utilização de Shell e Shell Script	132
7.3.1 - Exibição	132
7.3.2 - Variáveis.....	135
7.3.3 - Entrada de Dados.....	142
7.3.4 - Passagem de Parâmetro para o Script Shell	144
7.3.5 - Redirecionamento.....	147
7.3.6 - Comandos Condicionais e Operadores	148
7.3.7 - Miscelâneas	152
7.3.8 - Comandos de Controle	154
7.3.9 - Trap	159
7.3.10 - Alias.....	160
7.3.11 - Funções	161
Exercícios	163

Capítulo 8 - Administração de Rede	164
8.1 - As Redes TCP/IP	164
8.1.1 - Pilha de Rede.....	164
8.1.2 - Endereço de Rede.....	166
8.1.3 - Portas.....	167
8.2 - Configuração do TCP/IP.....	169
8.2.1 - Configuração do Hardware de Rede	169
8.2.2 - O Uso do IP Dinâmico - Cliente DHCP.....	171
8.2.3 - Uso do IP Fixo.....	174
8.3 - Compartilhamento de Arquivos	178
8.3.1 - NFS - Compartilhamento com UNIX ou Linux.....	178
8.3.2 - Samba - Compartilhamento com Windows	180
8.4 - Login Remoto	183
8.4.1 - Configuração de um Servidor Telnet.....	183
8.4.2 - Uso de um Cliente Telnet	184
8.4.3 - Configuração de um Servidor OpenSSH.....	185
8.4.4 - Uso de Clientes SSH	186
8.5 - Roteador de Rede.....	187
8.6 - Servidor DHCP.....	190
8.7 - Servidor FTP	191
8.8 - Servidor WWW	193
8.9 - Servidor NTP	196
8.9.1 - Sincronização de um Cliente NTP em Linux	197
8.10 - Servidor DNS	198
Exercícios.....	202
Bibliografia	203
Marcas Registradas	204
Índice Remissivo	205

Prefácio

O Linux vem se tornando um sistema operacional cada vez mais amigável e prático. É multiusuário, robusto, estável, resistente a falhas, totalmente customizável e flexível, por isso deve conquistar cada vez mais espaço nas empresas e, no futuro, também nos lares.

As interfaces gráficas disponíveis atualmente, tais como KDE e GNOME, estão suficientemente desenvolvidas para proporcionar ao usuário leigo um controle eficiente do sistema operacional e da máquina, deixando transparentes as rotinas mais complexas do Linux.

Utilitários do Linux de todos os tipos, como editores de texto e de imagem, planilhas de cálculo, edição eletrônica, e vários outros não deixam nada a desejar com relação a outros softwares consagrados nessas áreas. O melhor é que, na ampla maioria, são de uso gratuito, reduzindo significativamente o custo total de propriedade.

Para a total utilização e customização do Linux, principalmente em sua versão servidor, é necessário um conhecimento mais profundo do sistema operacional e de suas ferramentas internas.

Este livro mostra os fundamentos do Linux. Vai da teoria à prática, apresentando os conceitos e o uso diário de cada ferramenta.

O capítulo 1 aborda a história do Linux, sua relação com o Unix e outros assuntos relacionados. O capítulo 2 trata da estrutura do programa, suas vantagens e aspectos iniciais.

O capítulo 3 é totalmente prático. Mostra o dia a dia do administrador, as tarefas exigidas, as ferramentas comumente utilizadas, comandos básicos, gerenciamento de usuários e grupos, monitoramento dos vários aspectos do servidor, administração de serviços (daemons) e outros itens interessantes e necessários ao administrador.

A instalação do Linux em detalhes, com todos os passos necessários, e a sua execução de forma organizada e eficiente são temas do quarto capítulo. Já o capítulo 5 traz os métodos para inicializar e desligar o servidor, introduz os gerenciadores de boot mais utilizados (GRUB e LILO), como configurá-los e dicas de segurança. Também são mostradas as ferramentas para desligamento e reinicialização do Linux.

O capítulo 6 trata dos sistemas de arquivos, como montá-los e gerenciá-los. O sétimo capítulo descreve o Shell e mostra ao administrador como utilizar essa ferramenta poderosa para automatizar processos e tarefas diárias. Os principais conceitos são exibidos e devidamente explicados, capacitando o leitor a construir scripts simples ou complexos, conforme a necessidade.

O capítulo 8 abrange especificamente a administração de redes, definindo protocolos, instalação, configuração, DHCP, SAMBA, NFS, compartilhamento de arquivos com servidores Windows, login remoto, SSH, roteamento, FTP, NTP, DNS, servidor Web e outros assuntos interessantes nessa área, sempre com uma visão prática.

Sobre os Autores

Walace Soares

Capixaba, formado em Matemática, trabalha desde 1986 com tecnologia. Teve seu primeiro contato com o mundo Unix no SCO Xenix, e desde então não deixou de acompanhar a evolução desse sistema. Em 1999, conheceu o Linux por meio do Slackware, enfatizando administração e programação para Linux, principalmente com PHP, Apache, PostgreSQL e MySQL. É proprietário da MWS Serviços e Sistemas, empresa franqueada Bit Company, voltada para a formação de profissionais de informática, inclusive em produtos Linux.

Gabriel Fernandes

Nascido em Florianópolis, é formado em Eletrônica pela Escola Técnica Federal e está cursando especialização em Sistemas Microprocessados.

Trabalha com informática desde 1999. É autodidata e entusiasta do Linux desde 2002, e nesse tempo desenvolveu desde projetos de softwares até distribuições Linux para fins específicos.

Atualmente é sócio-proprietário da Duel Sistemas e analista de sistemas em empresa privada, onde se dedica a soluções e treinamentos em Linux.

1

A História

1.1 - UNIX

Para compreender o que é UNIX, vamos voltar no tempo pelo menos uns 40 anos, o suficiente para tornar-se obsoleto como muitos sistemas, por exemplo, o MS-DOS criado na década de 1980, descontinuado há alguns anos, o qual não tem suporte para a maioria dos hardwares atuais, restringindo o seu uso.

Para começar a entender um sistema com mais de 40 anos, cujo desenvolvimento iniciou-se antes mesmo de existirem os computadores na forma como conhecemos atualmente, e ainda considerá-lo sinônimo de confiança em soluções corporativas, tente imaginar computadores gigantes, mas gigantes mesmo, muito grandes, do tamanho de uma casa e até mesmo de um estádio de futebol. O tamanho dessas máquinas dificultava a manutenção e também tornava o ambiente mais propício a problemas. Mas este não era o maior problema, pois além do fator hardware, cada um tinha o seu sistema operacional próprio, porque na época os softwares eram desenvolvidos para propósitos específicos e não funcionavam em outros sistemas.

Um computador atualmente é muito barato se comparado aos computadores produzidos no final da década de 1970. Naquela época eles eram muito caros e a única forma de desenvolver um sistema para eles era após sua compra. Para ter acesso a esses computadores, era necessário ser um estudante universitário ou funcionário de empresa de TI.

Com os avanços tecnológicos, em 1969, uma equipe de desenvolvedores dos laboratórios Bell Labs iniciou o que seria a solução para o problema de compatibilidade entre sistemas diferentes, criando um sistema operacional em linguagem de programação C em vez de Assembly (linguagem de máquina), simples, eficiente e com a capacidade de reaproveitar partes específicas do código, disponibilizando os mesmos recursos em diferentes sistemas.

O primeiro UNIX foi desenvolvido em um PDP-7 (Programmable Data Processor), um equipamento com processador de dados programável construído pela DEC (Digital Equipment Corporation), por Dennis Ritchie e Ken Thompson, desenvolvedores da equipe da Bell Labs. A história conta que a motivação para eles criarem o UNIX foi o fato de Ken ter escrito um jogo chamado Space Travel para o Multics, sistema operacional desenvolvido pela Bell Labs com a participação deles. Como queriam jogar no PDP-7, fizeram um código base para rodar o jogo, que futuramente se transformaria no UNIX.

Na Figura 1.1 podemos ver Dennis Richie e Ken Thompson diante do conhecido UNIX Genesis Machine, o PDP-7.



Figura 1.1 - PDP-7, a UNIX Genesis Machine, e os criadores do UNIX.

Fonte: <http://blog.iso50.com/wp-content/uploads/2009/01/leopard-preview-server-1.jpg>

Com a capacidade de reutilização do código, sua popularidade aumentou rapidamente, pois até então todos os softwares eram desenvolvidos para soluções específicas, e precisavam ser totalmente reescritos para outros equipamentos. No UNIX, a única parte do código que precisaria ser alterada para funcionar em outro sistema (computador) era o núcleo do sistema, hoje conhecido como kernel. Neste contexto, os softwares evoluíram muito, pois a portabilidade entre sistemas facilitou a continuação dos sistemas independentemente da arquitetura de hardware.

Agora já era possível separar o computador (equipamento) do software, pois um mesmo equipamento poderia servir para mais de uma solução, assim os softwares ganharam "vida própria", ou seja, puderam evoluir paralelamente aos hardwares. Com isso os fabricantes rapidamente adequaram seus softwares específicos ao UNIX, pois assim aumentariam muito as vendas.

Por estes e outros motivos, os sistemas UNIX ainda são desenvolvidos, estão em constante evolução e cada vez mais reconhecidos como sistemas estáveis e seguros.

1.1.1 - A Filosofia UNIX

"UNIX é basicamente um sistema operacional simples, mas você precisa ser um gênio para entender a simplicidade", disse Dennis Ritchie. A simplicidade do UNIX não é encontrada somente no seu núcleo original, mas também em todas as suas variantes diretas e "reimplementações" (clones). Esses derivados do UNIX são conhecidos popularmente como

sistemas operacionais Unix-Like, sendo de longe considerados os melhores existentes e sempre em constante evolução, pois são mantidos ininterruptamente há muitos anos. Além disso, são os sistemas mais antigos e difundidos do planeta.

Não há uma única definição para a filosofia usada na construção do UNIX, mas em todas elas encontramos o item de maior destaque, a modularidade, pois ele tem a capacidade de gerenciar um sistema composto por componentes que podem ser arranjados e utilizados de diversas maneiras distintas.

Vamos usar algumas citações de membros importantes da comunidade para ajudar a definir a filosofia UNIX:

Doug McIlroy, o criador dos pipes no UNIX (estudados mais adiante), no livro *A Quarter Century of UNIX*, escrito por Peter H. Salus's, em 1994, resume:

"Esta é a filosofia UNIX:

Escrever softwares com apenas um propósito e fazer benfeito.

Escrever softwares que funcionem em conjunto.

Escrever softwares que manipulem fluxos de textos, pois esta é uma interface universal.

Frequentemente isso é abreviado pela comunidade para simplesmente:

Faça apenas uma coisa e faça bem."

Mike Gancarz, em seu livro *The UNIX Philosophy*, publicado em 1995, diz que a filosofia UNIX é formada por um conjunto de princípios fundamentais e secundários que asseguram a qualidade dos sistemas UNIX.

Os princípios fundamentais, segundo ele, são:

- » Pequeno é belo.
- » Construa programas que fazem uma coisa benfeita.
- » Disponibilize uma versão protótipo o mais breve possível.
- » Prefira portabilidade a eficiência.
- » Armazene dados em arquivos simples, como arquivo do tipo texto.
- » Use as funcionalidades disponíveis do sistema para tirar vantagem.
- » Use shell script para incrementar as funcionalidades e portabilidades.
- » Evite fazer programas engessados.
- » Sempre faça de seu programa um filtro.

Os secundários:

- » Permita que o usuário personalize o ambiente.
- » Desenvolva núcleo de sistemas operacionais pequenos e leves.
- » Use letras minúsculas e faça textos curtos.
- » Salve as árvores.

- » O silêncio vale ouro.
- » Pense paralelamente.
- » A soma de todas as partes é maior que o todo.
- » Atenção para os 90% da solução.
- » O pior é o melhor.
- » Pense de forma hierárquica.

Por fim, a contribuição de Eric S. Raymond, um grande ativista e formador de opinião da comunidade, em seu livro *The Art of UNIX Programming*, em que diz que a filosofia UNIX se resume a uma lei de ferro, chamada Kiss Principle, princípio do beijo. O significado mais aceitável para este acrônimo entre os autores é Keep It Simple, Stupid!, que significa "mantenha-o simples, estúpido!".



Figura 1.2

Eric defende esse princípio como uma norma para a cultura UNIX. No livro, ele descreve as regras da filosofia UNIX sob o seu ponto de vista. Acompanhe:

- » **Regra da modularidade:** escreva peças simples conectadas por interfaces simples.
- » **Regra da clareza:** clareza é melhor que inteligência.
- » **Regra da composição:** construa programas para serem conectados com outros programas.
- » **Regra da separação:** separe a configuração dos mecanismos; separe as interfaces da implementação de algoritmos.
- » **Regra da simplicidade:** projete para simplicidade; adicione complexidade apenas onde é necessário.
- » **Regra da parcimônia:** escreva um programa de tamanho grande somente quando fica claro, por demonstração, que de outra maneira não seria possível resolver o problema.
- » **Regra da transparência:** projete de forma a dar visibilidade, para fazer do processo de depuração e inspeção uma tarefa fácil.
- » **Regra da robustez:** a robustez é a filha da transparência e simplicidade.
- » **Regra da representação:** armazene o conhecimento na forma de dados, para que a lógica dos programas possa ser robusta e estúpida.
- » **Regra da menor surpresa:** no projeto de interfaces, sempre faça aquilo que é o menos surpreendente.

- » **Regra do silêncio:** quando um programa não tem nada surpreendente para dizer, ele não deve dizer nada.
- » **Regra do reparo:** quando é inevitável falhar, falhe ruidosamente e o mais cedo possível.
- » **Regra da economia:** o tempo do programador é caro; conserve-o em detrimento do tempo da máquina.
- » **Regra da geração:** evite escrever código; escreva programas que gerem código quando é possível.
- » **Regra da otimização:** faça protótipo antes do refinamento. Coloque em funcionamento antes de ser otimizado.
- » **Regra da diversidade:** desconfie de todas as alegações sobre a existência de um "modo correto de fazer as coisas".
- » **Regra da extensibilidade:** projete para o futuro, pois ele chegará antes do que você imagina.

1.1.2 - Variantes do UNIX

O UNIX nunca foi verdadeiramente gratuito, mas até a sua versão 6 era distribuído com as fontes. Inclusive havia publicações sobre o código-fonte, sendo uma das mais famosas o livro *A Commentary on the Sixth Edition UNIX Operating System*, escrito por John Lyons, em 1977. Essa publicação comentava passo a passo as linhas do código-fonte do UNIX versão 6. Nesse período de distribuição das fontes e consequente especificação formal para sistemas UNIX, a Single UNIX Specification, mantida pela Open Group, um consórcio da indústria que se formou em 1996, propiciou o surgimento de diversas variantes do UNIX, como Solaris da Sun Microsystems, a família BSD (FreeBSD, NetBSD, OpenBSD e Darwin) que é o núcleo do Mac OS X e Linux.

Dentre essas variantes algumas são de acordo com a especificação e outras apenas se comportam como um sistema UNIX, mas por dentro possuem suas próprias características. Existem, portanto, dois grandes grupos de sistemas variantes de UNIX, sendo aqueles em acordo com a Single UNIX Specification e os clones. Um programa clone possui funções similares a outro programa, porém tem um código-fonte totalmente diferente.

Para descrever esses sistemas operacionais, o termo Unix-like é frequentemente utilizado. Na família daqueles descendentes diretos, ou seja, daqueles cujo código-fonte é diretamente descendente do UNIX original, que possuem características óbvias de UNIX e, por fim, são descritos oficialmente como sendo UNIX, temos, por exemplo, AIX (desenvolvido pela IBM), HP-UX (desenvolvido pela HP), IRIX (desenvolvido pela Silicon Graphics), Solaris (desenvolvido pela Sun Microsystems), entre outros.

Na família de clones do UNIX, ou dos Unix-like, existem muitas semelhanças com os UNIX que alguns administradores consideram um sistema UNIX, porém oficialmente a maioria dos seus criadores não os caracteriza como um sistema UNIX. Alguns sistemas Unix-like clone são os da família BSD (FreeBSD, NetBSD, OpenBSD e Darwin) e também o Linux, MINIX, QNX e Cygwin.

1.2 - Linux

Linux não é UNIX!

O núcleo Linux é uma reinvenção ou "reimplementação" do núcleo do UNIX idealizada por Linus Torvalds, um estudante da Universidade de Helsinque na Finlândia, que em 3 de julho de 1991 divulgava sua aventura na criação de um pequeno sistema operacional, no qual estava trabalhando há alguns meses.

Na Figura 1.3 podemos ver Linux Tovarlds.



Figura 1.3 - Criador do Linux.

Fonte: <http://www.coated.com/wp-content/uploads/2010/02/linus-torvalds.jpg>

Naquela época, ele procurava a colaboração de outras pessoas com sugestões porque seu sistema era muito parecido com o Minix, um sistema operacional Unix-like, e já tinha algo funcional, pois estava migrando dois programas superimportantes do projeto GNU, sendo o compilador C e o Shell, mas ainda havia muito a fazer e ele queria melhorar.

Bons programadores inventam programas; grandes programadores reinventam.

O Linux foi desenvolvido em conformidade com o POSIX, um padrão para construção de sistemas operacionais criado para normatizar o UNIX. Ele funciona em centenas de arquiteturas diferentes e é compatível com a maioria dos softwares disponíveis para UNIX. A versão 0.01 do Linux tinha um desenvolvedor e aproximadamente dez mil linhas de código na linguagem C e algumas centenas de quilobytes, e hoje conta com uma comunidade gigante de desenvolvedores, milhões de linhas e algumas centenas de megabytes de tamanho, tudo sob a licença GNU GPL (abordada no item 1.5 - Free Software Foundation). O Linux é livre, de código-fonte aberto, desenvolvido cooperativamente e mantido por milhões de usuários e empresas espalhadas por todo o mundo.

Ele não é a invenção do século, nem o Linus é o novo Einstein, pois também tinha dificuldades e assuntos fora de seu domínio. Um ponto a ser observado é que os computadores da época, apesar de difícil acesso, não eram tão complexos se comparados aos computadores atuais. Naquela época havia poucas aplicações e os recursos de hardware eram bastante limitados.

O Linux foi construído com base em um conceito já considerado ultrapassado na época por alguns engenheiros da computação, como, por exemplo, Andrew S. Tanenbaum, criador do Minix, pois o núcleo do Linux é monolítico, tornando o kernel uma entidade indivisível, apenas um único "programa". Enquanto Linus "reinventava a roda", o pessoal do projeto GNU já trabalhava no núcleo GNU Hurd, conhecido somente como Hurd, o qual tem um formato diferente chamado de microkernel. Nele existe apenas uma pequena entidade núcleo que conversa com uma coleção de servidores de funções distintas de forma assíncrona, porém todos utilizando o mesmo protocolo. Isso faz do núcleo um sistema muito avançado e poderoso, no entanto mais difícil de depurar e mais suscetível a falhas. O GNU Hurd ainda está em desenvolvimento e mais informações podem ser encontradas no site oficial: <http://www.gnu.org/software/hurd/>.

Difícil acreditar que um projeto no qual todos contribuem pode dar tão certo.

Assim, o Linux possui muitas melhorias em relação ao UNIX, portanto não podemos dizer que ele é simplesmente um clone do UNIX. É muito mais que isso. Trata-se de uma nova entidade que não é UNIX; é como ele, mas não igual.

Além da existência do Linux, existem outros sistemas operacionais baseados no UNIX, então porque esses sistemas não são tão populares quanto o Linux?

A melhor estratégia de Linus foi disponibilizar seu código sob a licença GPL, da Free Software Foundation (FSF), criada por Richard Stallman no início da década de 1980, instituição de cunho social que estimula e desenvolve software livre. A FSF mantém até hoje o Projeto GNU, que desenvolveu clones de diversas aplicações do UNIX. Quando praticamente todas as ferramentas do Projeto GNU estavam prontas, com exceção do kernel, surgiu o kernel do Linux.

É importante lembrar que as pessoas apenas usam programas. Um sistema operacional não faz nada sozinho, mas quase ninguém nota sua presença, pois ele existe para fazer os programas funcionarem, e era o que faltava para o Projeto GNU libertar-se dos sistemas UNIX.

O Projeto GNU completava o Linux, e este seria o caminho do Linux: viabilizar o uso das ferramentas do Projeto GNU nos computadores residenciais sem nenhum custo, criando uma nova era no desenvolvimento cooperativo. Assim as pessoas passam a ter liberdade e condições de desenvolver aplicações sem a necessidade de investimentos de grandes empresas ou das universidades.

O Linus não foi o criador da comunidade de desenvolvimento aberto, mas o seu sistema foi o grande impulsionador do aumento exponencial de colaboradores, pois até então não existia um sistema completo gratuito sob a licença GPL.

1.3 - Vantagens do Linux

Grande parte das vantagens do Linux é em razão de ele ser um sistema Unix-like, ou seja, herda de seu antecessor características importantes. Entretanto, a primeira vantagem do

Linux destacada a seguir, e talvez uma das mais importantes, não é encontrada no UNIX, a saber:

- » **Gratuito:** a licença GNU GPL (estudada na seção Free Software Foundation) garante a liberdade e gratuitude do Linux. Ser gratuito significa que o usuário não terá nenhum ônus para baixar e instalar o sistema completo, inclusive os códigos-fonte. Com o avanço da velocidade de conexão com a Internet, nem mesmo é preciso pagar por um CD ou DVD para adquirir o Linux, pois ele pode ser totalmente baixado da Internet na maioria das distribuições.
- » **Comunidade:** "havendo olhos suficientes, todos os erros são triviais", por Eric S. Raymond, no manifesto *The Cathedral and the Bazaar*. Ao contrário do que a maioria pode pensar, você terá mais opções de ajuda usando o Linux do que qualquer outro sistema proprietário. A comunidade do Linux é muito grande, e há vasta documentação e fóruns disponíveis na Internet que ajudam bastante na maior parte das resoluções dos problemas. Assim, os problemas com o sistema operacional e aplicações são depurados e resolvidos rapidamente pelos milhões de usuários e desenvolvedores espalhados pelo mundo.
- » **Estável:** foi construído para ser unbreakable, inquebrável, ou seja, feito para nunca parar. Essa característica do Linux permite que ele seja usado em servidores de missão crítica, além de poder assumir novas configurações sem reiniciar. Em ambientes de servidores com Linux é comum encontrar computadores que não são reiniciados há meses.
- » **Seguro:** mais uma herança do UNIX, o modelo de segurança do Linux é amplamente reconhecido por sua robustez e qualidade comprovada. Com o Linux você pode proteger desde sua empresa com instalações de firewalls corporativos até seu desktop, tão seguramente quanto o seu firewall.
- » **Portável:** esta também é uma das características importantes, oriunda do UNIX. A portabilidade é fundamental para a difusão de qualquer sistema. Com o Linux, um fabricante de equipamento que não sabe o tipo de sistema operacional que a sua nova máquina vai executar (máquina pode ser desde um computador até uma máquina de lavar roupas) pode alterar o kernel Linux para funcionar em seu hardware, pois toda documentação necessária está disponível gratuitamente.
- » **Escalável:** um sistema Linux é totalmente personalizável, pois é constituído basicamente de um kernel e pacotes de software. Essa característica faz com que ele funcione em um simples dispositivo móvel, como celulares e coletores de dados, e até em supercomputadores. E ainda, se você quiser fazer um sistema operacional para um novo processador embutido ou apenas ressuscitar seu computador antigo, o Linux faz isso também.
- » **Antivírus:** para quê? O sistema de permissões nativo do Linux é suficiente para torná-lo um sistema livre de vírus e ameaças. No Linux o sistema é protegido inclusive em tempo de execução e para fazer qualquer ação que danifique as partes principais do sistema, é necessário ter privilégio, ou seja, o suposto vírus ou worm, que é um script malicioso, terá de saber a senha do administrador do sistema antes de ter acesso a ele. Sem acesso aos arquivos do sistema, a menos que alguém lhe informe a senha, o worm nunca descobrirá sozinho, portanto apenas cuide para não ser vítima de engenharia social e nunca use o seu desktop com o usuário administrador do sistema. Prefira um usuário normal, sem privilégios.

» **Flexível:** com as fontes livres para acesso de todos, muitas opções de personalização e configurações tornam o Linux o sistema mais flexível que existe, podendo adaptar-se a qualquer solução, sem contar os milhares de softwares também de código-fonte aberto que o Linux carrega, impulsionando seu desenvolvimento a quem quiser colaborar e alterar.

1.4 - Projeto GNU

GNU é originalmente um mamífero africano, parecido com um búfalo, mas o nome não tem nada a ver com o animal. O nome do projeto é uma brincadeira irônica dos programadores da época. GNU é o acrônimo de "GNU is not UNIX", e todo programa livre que acompanhava os códigos-fonte era um GNU, em contraposição ao Unix que, a partir da versão 7, fechou o acesso ao código-fonte. A mascote símbolo do projeto é uma caricatura da cabeça do próprio boi gnu, como é possível ver na Figura 1.4.



Figura 1.4 - Símbolo do projeto GNU.

A ideia do projeto consiste na construção de um sistema operacional como o UNIX, contendo somente programas livres suficientes para substituírem qualquer sistema não livre existente, no caso o próprio UNIX. Esse sistema operacional GNU pretendia ser livre, sem restrições, de forma definitiva, podendo ser redistribuído por qualquer pessoa.

Criado por Richard M. Stallman, em 27 de setembro de 1983, o projeto GNU, ou sistema operacional GNU, era a estratégia para manter o método como os programas eram repassados na época apenas entre seus próprios usuários, pois poucos se importavam em proteger o código-fonte do sistema.

O projeto, então, começou a escrever substitutos ou clones de programas importantes do UNIX, como compilador, editor de textos, editor de planilhas, entre outros, ou seja, tudo que era necessário para compor um sistema operacional completo, porém livre.

Informalmente já havia uma comunidade de cooperação entre os desenvolvedores, ou seja, quando alguém queria usar um programa, solicitava à pessoa que o desenvolveu e normalmente ele era fornecido com os códigos-fonte, e quando alguém precisava de

ajuda, e não era muito difícil, recebia-a. Quanta confusão! Mas era assim mesmo. Uns ajudavam os outros, pois o propósito era prático, resolver problemas, e não comercial.

A história conta que tudo começou quando Richard era desenvolvedor e pesquisador no MIT Lab AI (Massachusetts Institute of Technology Laboratory Artificial Intelligence) e, na década de 1970, deparou-se com um problema no driver de uma impressora e logo decidiu melhorá-lo e corrigir o problema, mas em contato com a empresa descobriu que isso não seria possível, pois os códigos-fonte do driver eram fechados, e mesmo sabendo que havia uma melhoria a ser implementada no driver e que ele seria capaz de fazê-lo, a empresa não liberou as fontes.

Os tempos realmente estavam mudando. No final da década de 1970 e durante a de 1980, os usuários começaram a fechar seus códigos, muito por influência do modelo comercial de sucesso da Microsoft.

A Microsoft de Bill Gates firmou sua preocupação e sua batalha com esses usuários e métodos em uma carta enviada por ele ao Homebrew Computer Club, um clube criado por programadores no Vale do Silício. Bill defende o modelo proprietário e compara o compartilhamento de código ao comunismo. Essa carta tinha o título An Open Letter to Hobbyists (Carta Aberta aos Entusiastas).

Diante dessa tendência inevitável, Richard idealizou o projeto GNU ou o sistema operacional GNU. Ele acreditou que por ser um dos desenvolvedores de um sistema operacional criado no MIT, em 1971, chamado de ITS (Sistema de Tempo Compartilhado Incompatível, Incompatible Tomesharing System, um trocadilho com CTTS, sistema operacional criado anteriormente pela mesma instituição), e pela motivação de programadores dessa instituição, ele também poderia desenvolver um sistema operacional e convidar pessoas para ajudá-lo gratuitamente em troca de poderem usá-lo também gratuitamente.

O primeiro e talvez o mais importante programa livre criado pelo projeto foi o compilador C, chamado de GCC (GNU C Compiler), pois com um compilador gratuito as portas para as contribuições se multiplicavam, já que agora era possível criar programas sem ter de pagar pelos compiladores. Além do GCC, que é um dos grandes programas criados pelo GNU, destaca-se também o GNU Emacs, um editor de texto muito difundido e utilizado até hoje.

Uma lista completa dos projetos GNU pode ser vista no link: <http://directory.fsf.org/GNU/>.

1.5 - Free Software Foundation

"A Fundação para o Software Livre (FSF) é uma organização sem fins lucrativos com a missão global de promover a liberdade de usuários de computadores e de defender os direitos de todos os usuários de software livre." (Fonte: <http://www.fsf.org/>)

Também criada por Richard M. Stallman, essa fundação financia o projeto GNU. Todo projeto tem um custo e para não vincular o projeto GNU a nenhuma empresa, Richard criou a fundação para angariar fundos para seus projetos de software focados na liberdade

da informação para fomentar o conhecimento. Esta é a principal característica dos softwares livres, diferentemente de um software open source, cujo propósito é fomentar o seu próprio desenvolvimento, disponibilizando as fontes para que possam ser analisados e alterados.

Para garantir as liberdades dos softwares, a FSF precisava de um documento formal, portanto ela criou a licença GNU GPL (General Public License), que em linhas gerais fornece quatro liberdades:

1. Liberdade de executar o programa para qualquer propósito.
2. Liberdade de estudar como o programa funciona e adaptá-lo às suas necessidades.
3. Liberdade de redistribuir cópias de modo que você possa ajudar o seu próximo.
4. Liberdade de melhorar o programa e liberar os seus melhoramentos, de modo que toda a comunidade se beneficie deles.

Além destas, existem outras licenças de uso da FSF, algumas comentadas a seguir:

- » **GNU AGPL (Afferro General Public License):** é uma licença minimamente modificada da GNU GPL para fornecer liberdade em softwares como serviços (SaaS, Software as a Service), softwares que não oferecem acesso direto ao binário/código-objeto.
- » **GNU LGPL (Lesser General Public License):** a principal diferença da GPL é que ela permite a associação com programas que não estejam sob as licenças GPL ou LGPL, inclusive software proprietário.
- » **GNU FDL (Free Documentation License):** licença para documentos e textos livres publicados pela Free Software Foundation.

Exercícios

1. Como eram e quais as dificuldades encontradas nos computadores na época da criação do UNIX?
2. Qual a característica do UNIX que contribuiu para a sua difusão rápida? Explique.
3. O que significa o termo Unix-like?
4. Qual a diferença entre sistemas UNIX e Unix-like clone?
5. Como foi desenvolvido e como é mantido o Linux?
6. Qual a relação do projeto GNU com o Linux? Explique.
7. Como podemos adquirir uma cópia do Linux?
8. Apesar da gratuidade, por que é fácil encontrar ajuda sobre o Linux?
9. Como se proteger de scripts maliciosos no Linux?
10. Cite duas características do Linux que justificam o uso em servidores.
11. Qual a finalidade do Projeto GNU?
12. Explique a licença GNU GPL.

2

Familiarização com o Linux

2.1 - Windows e Linux

As chances de você querer usar aplicativos do Microsoft Windows e do Linux simultaneamente são gigantes. Existem muitas maneiras de rodar aplicativos Windows no Linux e também interagir e utilizar aplicações ou impressoras do Windows no Linux e do Linux no Windows.

Entre as possibilidades estão o fato de rodar os aplicativos do Windows em um desktop Linux por meio de emuladores. Também é possível ter o Windows funcionando no Linux sob uma máquina virtual e, se preferir, ao instalar o Linux você pode continuar com o Windows instalado da forma como estiver. Diferentemente do Windows, pois se instalá-lo em uma máquina que possua o Linux, pode ter certeza de que a instalação do Windows vai pelo menos tornar inacessível o Linux se você não tiver um conhecimento mais avançado.

O Windows é sistema operacional fácil de usar, desta forma dominou o mercado de desktops no mundo, com capacidade de encurtar a distância entre o usuário e o mundo da informática.

O Linux é uma plataforma com infraestrutura confiável e escalável, muito poderosa e também utilizada para desktops, com designer muito atraente e moderno.

2.2 - POSIX

POSIX é uma sigla que significa Portable Operating System Interface (Interface Portável entre Sistemas Operacionais). Ela é uma coleção de regras definidas pelo IEEE (Institute of Electrical and Electronics Engineers), a maior instituição profissional do mundo dedicada ao avanço e excelência das inovações tecnológicas, que normatiza uma interface entre programas e o sistema operacional. O IEEE é responsável por determinar diversas normas e padrões para equipamentos e sistemas aceitos por todo o mundo.

A norma IEEE 1003 tem o objetivo de garantir uma interface padrão entre programas e o sistema operacional, fazendo com que qualquer programa desenvolvido para um determinado sistema operacional padrão POSIX funcione em todos os sistemas operacionais desse padrão. Um sistema operacional no padrão POSIX deve garantir a portabilidade do código-fonte de um programa, para que ele funcione em qualquer outro sistema operacional que também esteja no padrão.

Sabemos que POSI vem do acrônimo POSIX, mas e o X? Mais uma vez temos a influência de Richard M Stallman, pois o nome foi sugerido por ele a pedido da IEEE, por volta de 1985, quando normatizaram o padrão de API para as variantes de sistemas UNIX, o POSIX. O X representa o vínculo com o UNIX; era praxe colocar um X no final de programas UNIX e suas variantes.

2.3 - Kernel

Kernel é o núcleo do sistema operacional, o programa ou uma coleção de programas mais importante dentro de um computador, porém não é visto nem tampouco utilizado diretamente pelos usuários em sua maioria.

O núcleo tem como objetivo principal criar uma camada de abstração entre o hardware e o software, viabilizando o acesso aos dispositivos de hardware, como memória, processador e periféricos, e, consequentemente, viabilizar o funcionamento dos programas. São os programas que interagem diretamente com o núcleo. Podemos dizer que um sistema operacional não serve para nada sem os programas. A Figura 2.1 permite visualizarmos uma concepção básica do kernel.

Basicamente o sistema operacional faz o reconhecimento dos dispositivos de hardware e inicia o primeiro processo ou programa, depois ele fica esperando uma chamada de um programa ou de uma interrupção por hardware, como quando digitamos uma tecla no teclado. As duas principais tipificações de núcleos são os núcleos monolíticos e os micronúcleos.

Um núcleo monolítico é apenas um programa no qual rodam todos os serviços do sistema operacional em uma mesma área de alocação da memória. Esse método é considerado o mais fácil de programar, porém sua principal desvantagem é que o tamanho dos núcleos pode ficar muito grande e difícil de manter por ser uma entidade indivisível. Pelo mesmo motivo, um defeito em um determinado driver de um dispositivo pode comprometer todo o sistema e travar, o famigerado Kernel Panic no caso do Kernel Linux. Na Figura 2.2 podemos ver um diagrama básico do núcleo monolítico.

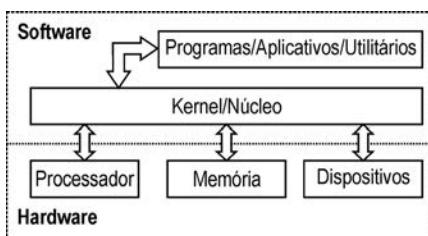


Figura 2.1 - Núcleo do sistema e suas conexões.

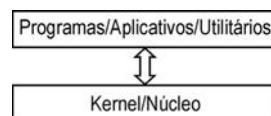


Figura 2.2 - Diagrama de um núcleo monolítico.

O micronúcleo é uma tecnologia muito mais sofisticada. Consiste em um programa com um conjunto mínimo de instruções que gerenciam os serviços básicos do sistema operacional; os outros serviços são programados no nível de usuário, às vezes em linguagem de alto nível. Neste contexto teremos um núcleo mínimo fácil de manter e uma coleção

de servidores que se comunicam com ele em um único protocolo. Assim o micronúcleo permite diferentes sistemas operacionais em um mesmo núcleo e também alternar entre um sistema e outro dinamicamente, mantendo os dois funcionando ao mesmo tempo. No entanto, esse modelo tende a ser mais suscetível a falhas e mais difícil de depurar.

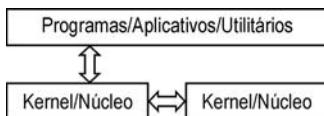


Figura 2.3 - Diagrama básico de um micronúcleo.

O kernel do Linux possui um desenvolvimento colaborativo, mas as principais diretivas e partes mais difíceis do núcleo ainda estão nas mãos de Linus. Ele conta com a ajuda de um grande colaborador e convededor do kernel, o Alan Cox, que ajuda no projeto desde suas versões alfas e betas, em 1991.

A Figura 2.4 mostra Alan Cox.



Figura 2.4 - Alan Cox, o segundo homem no comando do Kernel Linux.

Fonte: http://upload.wikimedia.org/wikipedia/commons/e/e2/Alan_Cox_at_FOSS_2007.jpg

Além de Alan Cox, existem outras figuras importantes no desenvolvimento do Kernel Linux. Um destaque é Dave Miller, que já enviou mais de 2.700 colaborações para o Kernel Linux, é responsável pela pilha TCP/IP do Linux e também já contribuiu com muitos drivers para placas de redes.

O Kernel Linux possuía uma sistemática de numeração de versão simples, até o Kernel 2.6, em que o primeiro número indicava a versão principal do kernel, e o segundo as modificações significativas e sua estabilidade, pois versões estáveis tinham o segundo número par e as instáveis ou em desenvolvimento, ímpar. Por último, o terceiro número significava pequenas correções e submissões.

A partir do Kernel 2.6 a nomenclatura das versões passou a ser controlada por um apêndice "rc", que significa release candidate, ou seja, é uma versão candidata a estável. Como exemplo podemos ter a versão 2.6.35 estável e a versão 2.6.36-rc3 em desenvolvimento, ou seja, candidata a uma versão estável para o Kernel 2.6.36.

Mais informações sobre o Kernel Linux podem ser encontradas em <http://www.kernel.org>.

2.4 - Shell

Shell, em inglês, significa "concha". Este nome faz alusão ao fato de ele ser a entidade que envolve o kernel do sistema, como uma espécie de concha mesmo. É uma classe intermediária porque, de maneira genérica, está entre o computador e os usuários. Também podemos dizer que ela é uma interface entre o usuário e o sistema operacional. O Shell é o primeiro aplicativo executado ao efetuar o login no sistema. Ele é responsável por muitas outras tarefas repetitivas, a fim de aliviar o uso do kernel e liberá-lo para cuidar de recursos mais importantes.

O Shell funciona como um interpretador de comandos, ou seja, possui uma linguagem de programação interpretada, possibilitando o uso de scripts que façam tarefas úteis como se fossem programas. A vantagem de usar o Shell para automatizar tarefas e resolver problemas está em sua relativa simplicidade de programação e linguagem de alto nível.

Como ele faz a interface do usuário com o sistema operacional, é o único caminho que os programas possuem para alcançar o kernel do sistema, pois todos os programas precisam trocar mensagens com o kernel para funcionar normalmente, no entanto não é possível chegar até o kernel diretamente, apenas após a filtragem do Shell. Ele interpreta os comandos digitados no teclado, verifica as sintaxes (forma como o comando foi escrito) e prepara o comando para a execução. A Figura 2.5 mostra um diagrama de peças do sistema, em que podemos visualizar a posição do Shell nesse contexto.

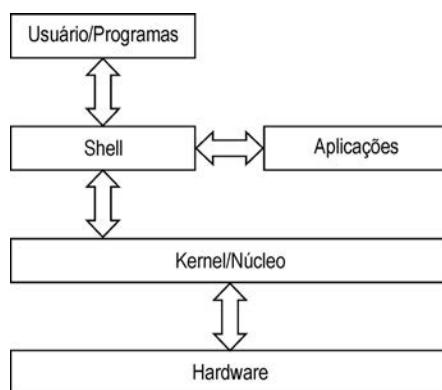


Figura 2.5 - Diagrama básico das peças do sistema.

O Shell possui uma ordem de execução dos comandos, cujas principais tarefas vamos entender. Primeiramente ele verifica a linha de comando, identificando os caracteres especiais (símbolos reservados) que possuem significado na interpretação da linha, em seguida ele procura na linha um comando ou uma atribuição; às vezes pode haver os dois. Comando é o conjunto de palavras ou símbolos digitados no prompt do Shell. Ele é separado por espaços e normalmente a primeira parte é o nome do programa que queremos executar. Ele pega esse nome e verifica se o programa existe. Caso ele não exista, o Shell retorna uma mensagem de erro; caso ele exista, verificam-se as permissões do programa. Havendo permissão de execução desse programa para o usuário, então é lido o restante do conteúdo do comando na seguinte ordem: primeiramente os parâmetros, depois os redirecionamentos e por último as variáveis.

Uma atribuição consiste nas definições de variáveis, ou seja, é o recurso que permite ao usuário a criação de variáveis diretamente no Shell, as quais podem ser usadas enquanto a sessão atual estiver aberta. O Shell procura a existência de dois campos separados pelo sinal de igual (=) sem espaços entre eles. Se houver, ele entende que é uma atribuição.

Após esses passos o Shell verifica se há redirecionamentos, que é a capacidade de o Shell enviar o retorno dos comandos para saídas diferentes. Um exemplo prático é salvar o conteúdo do comando em um arquivo texto, em vez de simplesmente exibi-lo na saída padrão.

Agora o Shell verifica se existem variáveis no comando e as substitui pelos seus valores armazenados em memória anteriormente por uma atribuição, e também substitui os metacaracteres por seus valores. Metacaracteres são caracteres que facilitam, em alguns comandos, por exemplo, copiar todos os arquivos de uma pasta para outra. Em vez de colocarmos todos os nomes de arquivos, podemos simplesmente usar o metacaractere asterisco (*), pois o Shell vai substituí-lo pelo nome de todos os arquivos na pasta.

Terminados esses processos, o Shell monta a linha de comando já formatada e filtrada para enviar ao kernel, que executa o comando em um novo Shell. Enquanto o programa está sendo executado, o Shell principal, aquele que formatou e enviou o comando ao kernel, fica inativo, aguardando o fim da execução do programa. Ao término da execução do processo, juntamente com o segundo Shell que carregou esse programa, o Shell principal assume novamente o controle e fica aguardando um novo comando.

Cada usuário pode utilizar o Shell que desejar e também é possível, a partir de um Shell, invocar outro. Como vimos anteriormente, o próprio Shell faz isso automaticamente para executar os programas, e trabalha com ele normalmente. Isso é possível porque o Shell, apesar de realizar diversas tarefas importantes, é apenas mais um programa do sistema operacional.

Existem vários tipos de Shell. Os mais conhecidos são sh, bash, csh e ksh. Todos podem conviver juntos em um mesmo sistema operacional do tipo UNIX, como o Linux. A seguir temos uma breve descrição deles:

- » **sh - Bourne Shell:** lançado em 1977, concomitantemente ao UNIX versão 7, foi desenvolvido pela equipe de desenvolvedores da AT&T Bell Labs. Foi durante muitos anos o Shell padrão do sistema operacional UNIX. É conhecido como Standard Shell porque durante muito tempo ele foi o único Shell existente.
- » **bash - Bourne Again Shell:** é um dos mais modernos e mais utilizado atualmente, pois além de ser o Shell padrão para sistemas Linux, também é muito poderoso. Muito difundido, esse Shell tem vasta documentação e exemplos disponíveis, além de ser uma ótima ferramenta para administradores do sistema. Mais adiante abordaremos o uso do Shell como apoio as tarefas do administrador.
- » **ksh - Korn Shell:** mais um projeto realizado pela equipe de desenvolvedores da AT&T Bell Labs, é uma evolução do Shell sh (Bourne Shell), pois tem todas as facilidades dele e apresenta muitas outras novidades. É totalmente compatível com o Bourne Shell, portanto é uma boa opção para usuários mais avançados acostumados com ele.
- » **csh - C Shell:** se tivéssemos de elencar o mais complexo para usuários novatos, sem dúvida é este. Desenvolvido pela Universidade de Berkeley, no final da década de 1970, para os sistemas BSDs, tem sua sintaxe de comandos muito próxima da linguagem C. De fato, esse Shell é para usuários verdadeiramente experientes, pois não é compatível com nenhum dos demais Shells apresentados aqui.

2.5 - Multitarefa

Multitasking ou multitarefa em um sistema operacional refere-se ao poder de executar vários processos, programas ou tarefas simultaneamente em um único computador, aparentemente sem interferência de um no outro. Portanto, cada processo tem a percepção de que ele é o único em execução no computador e que possui poder exclusivo sobre todos os serviços oferecidos pelo sistema operacional, o kernel.

A quantidade de programas ou processos que podem ser executados ao mesmo tempo depende de diversos fatores, como o tamanho da memória, o clock do processador e o tamanho dos programas.

Em um sistema operacional bem construído, o kernel, que é um controlador de processos, deve estar totalmente protegido, assim como os processos também devem estar totalmente independentes uns dos outros, pois quando houver um problema em determinado processo, ele não influencia os outros nem tampouco o Kernel. Assim, um programa não fará o outro falhar e, consequentemente, o sistema operacional também não.

É importante saber que um processador executa apenas uma função por vez, portanto a multitarefa é concebida por uma técnica chamada timeslice (fazendo fatias de tempo), que alterna o uso do processador rapidamente entre velocidades muito altas, entre os vários processos em execução. Para entender melhor, podemos analisar um sistema unitasking ou monotarefa. O famigerado MS-DOS é um sistema monotarefa. Nele o processador permanece inativo até que o recurso do sistema solicitado pelo processo esteja disponível, reduzindo significativamente o desempenho do sistema.

Em termos práticos, ao enviar um comando para uma impressora, o processador ficaria inativo, inoperante até que ela estivesse disponível, em linha. No multitarefa os processos ficam na memória principal, a memória RAM, fazendo com que ele atenda imediatamente algum outro processo, em vez de ficar esperando alguma resposta ou dispositivo.

A maioria dos sistemas operacionais é multitarefa. As versões antigas do Windows e Macintosh usavam a multitarefa cooperativa, que era gerenciada pelos próprios processos, que deveriam ceder voluntariamente o controle para outros programas em determinados momentos. Um grande problema é que nesse modelo um programa mal escrito poderia falhar todo o sistema, pois caso ele parasse de responder e não fornecesse o comando do kernel para o outro processo, provavelmente o kernel também ficaria inacessível.

Os sistemas Linux, UNIX, variações do UNIX, Windows 2000 ou superior e Mac OS X utilizam um modelo muito mais estável de multitarefa, chamado multitarefa preemptiva, por antecipação. Esse método reorganiza a maneira como as tarefas serão executadas pelo processador. Nesse formato o kernel mantém em memória um registro do contexto atual de cada processo, uma fotografia de todos os processos em execução, chamado árvore de processos.

A Figura 2.6 mostra uma estrutura em árvore. A partir da árvore e do atributo prioridade existente em cada processo, o núcleo do sistema calcula o tempo que deve fornecer a cada um deles, o timeslice, criando uma fila de processos. Quando terminar o tempo

de um processo, o núcleo cede o controle do processamento para o próximo e assim sucessivamente até chegar ao fim da fila e retornar para o topo.

Entretanto, o kernel do Linux não alterna entre um processo e outro somente quando estoura o timeslice. Ele também alterna quando o processo decide acessar um recurso ou dispositivo, pois isso faz com que ele durma (sleep) enquanto aguarda o recurso, então o kernel passa o comando para a próxima tarefa. Um terceiro motivo seria em função do uso do pipe, que é a possibilidade de utilizar as informações de saídas de um processo como dados de entrada de outro.

A multitarefa pode ser observada facilmente em ambientes Linux e Windows quando utilizamos uma interface gráfica, pois podemos abrir vários programas diferentes em janelas diferentes simultaneamente, sem que um interfira diretamente em outro.

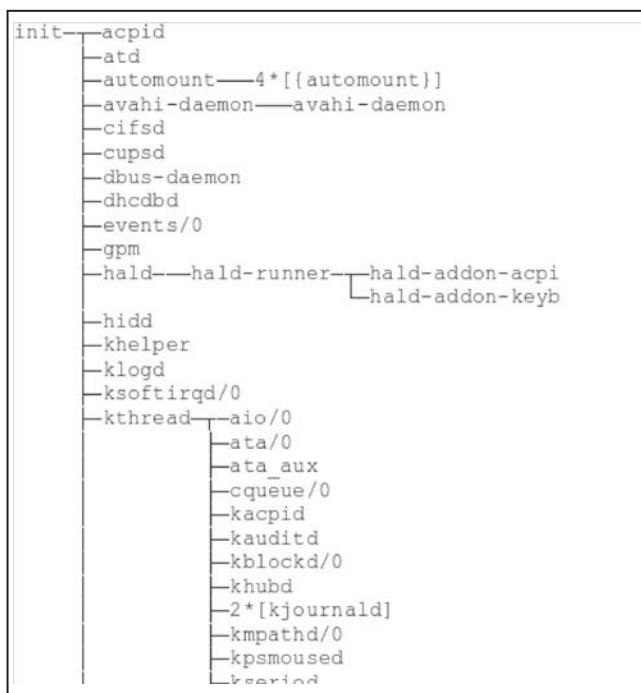


Figura 2.6 - Exemplo de árvore de processos.

2.6 - Multiusuário

Multiuser ou multiusuário é a capacidade de haver diversos usuários em um mesmo sistema operacional. A multitarefa possibilitou que vários usuários utilizassem o mesmo computador simultaneamente por meio de terminais virtuais ou remotamente.

No Linux existem dois tipos de usuários, sendo o normal e o superusuário. Este último é o administrador do sistema e chama-se root.

O root possui permissão para acessar todos os arquivos do sistema sem nenhuma restrição, inclusive incluir, editar e excluir arquivos dos outros usuários. Já os usuários normais possuem diversas restrições de acesso a arquivos do sistema, como os arquivos de configuração e dispositivos, além de restrições de espaço em disco, memória, entre outras. Vale ressaltar que um arquivo criado por um usuário normal tem restrição para ser acessado por outro usuário normal, ou seja, os arquivos podem ser acessados pelo seu dono e pelo root, mas seu acesso pode ser proibido a outros usuários normais ou não.

As propriedades relacionadas aos usuários estão gravadas em cada um dos arquivos existentes no sistema. Como no Linux podemos considerar que tudo é tratado como arquivo, essa estrutura de permissões contempla um total controle do sistema por parte do superusuário, pois em cada arquivo, diretório ou dispositivo podemos alterar três níveis de permissões, sendo aqueles relacionados ao dono do arquivo, ao grupo de usuários ao qual o dono pertence e a todos os outros usuários.

2.7 - Modo de Execução

No Linux, como na maioria dos sistemas modernos, existem dois modos distintos de execução que o processador do sistema pode assumir, sendo o Modo Usuário (User Mode) e o Modo Kernel (Kernel Mode). Eles determinam a restrição de acesso a todos os recursos do sistema. Vamos entender a diferença entre os dois modos.

2.7.1 - Modo Usuário

Modo que inicia processos sem privilégios, como quando executamos uma instância de um programa com usuário normal do sistema. Não ter privilégios significa estar proibido de realizar algumas tarefas, como acessar partes de memória alocadas pelo kernel ou drivers de dispositivos do sistema. Devido a essa proteção, as falhas no modo de usuário são normalmente recuperáveis. Quando um processo estiver sendo executado nesse modo e houver a necessidade de utilizar serviços providos pelo kernel, como acessar mais memória do que aquela alocada para os programas de usuários, o modo é chaveado automaticamente e temporariamente para o Modo Kernel. Quando o kernel terminar de processar a requisição, ele retorna imediatamente o processo para o modo normal, garantindo que somente o kernel tenha acesso a todos os recursos do sistema. Vale destacar que a maioria dos programas em execução no seu computador provavelmente é executada no Modo Usuário.

2.7.2 - Modo Kernel

Quando estamos nesse modo, assume-se que estamos executando um software verdadeiramente seguro, por isso todo o controle do hardware é cedido a esse software. Problemas nesse modo de execução certamente travam todo o sistema, por isso esse modo é reservado para o nível mais baixo de funções do sistema, como, por exemplo, um driver de dispositivo. O kernel do Linux é um software confiável e possui controle total de tudo que acontece no sistema. Neste contexto, todos os outros programas são considerados não confiáveis. Quando um programa executado em modo normal necessitar de algum

recurso do sistema restrito ao modo kernel, esse programa precisa fazer uma chamada para que o kernel execute a função e disponibilize o recurso, o que é chamado de system call (chamadas de sistema).

2.8 - Memória Virtual

Em termos práticos, a memória virtual consiste em usar o disco rígido para simular uma memória adicional à memória principal existente no sistema. Essa memória é utilizada para armazenar partes do sistema operacional, programas e dados que estão sendo utilizados atualmente ou que são necessários com frequência.

A memória principal é a de acesso aleatório (memória RAM), que é um dispositivo de hardware com chips de memória, conhecido com "pente de memória", os quais estão conectados na placa-mãe por meio de slots. Os tempos de acesso a uma memória RAM são extremamente rápidos e praticamente iguais para qualquer endereço de memória que se deseja acessar, no entanto o acesso à memória virtual localizada em discos rígidos ou qualquer outro dispositivo de armazenamento possui tempos distintos e muito grandes em relação à RAM.

Um sistema operacional com memória virtual tem a capacidade de liberar espaço na memória principal, copiando os dados que não são imediatamente necessários para o disco rígido. Quando esses dados voltam a ser úteis para a operação atual, ele copia novamente para a memória principal. Ou seja, quando toda a memória principal do sistema estiver ocupada, ele habilitará a troca (swap) de dados para o disco rígido e retornará a memória quando necessário. Isso aumenta a memória total disponível no sistema. Sendo assim, com ela é possível executar programas cujo tamanho seja superior ao da memória RAM e vários programas simultaneamente, pois cada vez mais os programas são maiores e ocupam mais espaço, acompanhando a evolução dos dispositivos para disponibilizar novos recursos.

Os programas não conseguem diferenciar a memória principal da virtual e normalmente os usuários também não percebem sua existência. Por vezes ela pode ser percebida em função da perda de desempenho do sistema, quando usada em excesso, pois o processador gasta mais tempo para copiar da memória para o disco rígido e do disco rígido para a memória, desperdiçando tempo que poderia ser utilizado para processar itens mais úteis. Esse efeito produzido pelo uso em excesso da memória virtual é conhecido como thrashing (surra) e afeta significativamente o desempenho do sistema, pois os discos rígidos não foram concebidos para acessar dados tão rapidamente como a memória RAM nem foram projetados para acessar porções de dados tão pequenas como um simples byte por vez.

No GNU/Linux o espaço no disco rígido reservado para ser a memória virtual está localizado em uma partição separada (partição é uma seção lógica independente no disco rígido, abordada no item Sistemas de Arquivos), que é definida no momento da instalação e identificada como partição Swap. Na maioria dos sistemas Linux é recomendado que a partição Swap tenha duas vezes o tamanho da memória RAM do sistema.

O espaço para troca é dividido em segmentos chamados de páginas; cada página está associada a um endereço de memória. Quando um endereço de memória é chamado, a

página é enviada para a memória e depois retornada ao disco rígido quando ele não for mais útil por um longo período de tempo, deixando espaço para referenciar outras páginas da memória virtual.

Vale lembrar que, atualmente, a maioria dos sistemas operacionais utiliza memória virtual, porque ela fornece um grande benefício ao usuário a um custo muito baixo, se compararmos o valor da mesma porção de memória RAM ao da memória virtual utilizada no disco rígido.

2.9 - Linux Foundation

A Linux Foundation é uma instituição sem fins lucrativos que promove, protege e padroniza o Linux, disponibilizando recursos unificados e os serviços necessários para o sucesso do código-fonte aberto, perante os sistemas de código fechado. Fundada em 22 de janeiro de 2007, é fruto da fusão de duas instituições muito importantes na história do Linux, a FSG (Free Standards Group), um consórcio de indústrias sem fins lucrativos, com o objetivo de especificar e conduzir a adoção dos padrões de código aberto, e a OSDL (Open Source Development Labs), com o objetivo de fomentar o desenvolvimento do Linux para uso profissional.

Os padrões desenvolvidos por eles são licenciados pela GFDL (GNU Free Documentation License) e os programas são livres.

A Linux Foundation mantém diversos grupos de trabalhos importantes. Os mais populares são LSB (Linux Standard Base), MeeGo, Accessibility e Open Printing. Mais informações podem ser encontradas em <http://www.linuxfoundation.org/>.

LSB (Linux Standard Base)

O LSB é um trabalho em constante desenvolvimento que padroniza a estrutura interna de sistema operacional Linux, incluindo a hierarquia de arquivos. Ele é baseado no POSIX, Single UNIX Specification, e em uma série de outros padrões livres. O objetivo da LSB é definir uma coleção de padrões livres que permitam a compatibilidade entre distribuições, viabilizando que qualquer programa desenvolvido em um Linux no padrão LSB funcione em qualquer outro sistema Linux, mesmo já estando no formato binário. Além disso, ajuda a coordenar os esforços a fim de convencer desenvolvedores e fabricantes a portarem seus programas para um sistema operacional Linux.

Open Printing

É um grupo de trabalho muito importante quando se fala em suporte à impressão em sistemas gratuitos como GNU/Linux e BSDs, além de sistemas comerciais baseados em UNIX, como Solaris e Mac OS X. Ele desenvolve o Foomatic, um sistema de banco de dados orientado para a integração de drivers de impressoras com softwares livres spoolers dos UNIX. O repositório de drivers disponíveis pode ser visto em <http://www.openprinting.org/printers/>.

O projeto está caminhando para um padrão não oficial utilizado por diversas distribuições GNU/Linux e atualmente possui por volta de 15 mil acessos diários ao

site. Apesar do sucesso do projeto, ele precisa de colaboração urgentemente. O projeto hoje é mantido praticamente por uma pessoa, Till Kamppeter, tendo a ajuda de um brasileiro, Vitor Baptista, único estudante selecionado no Google Summer of Code 2010 (programa global que oferece bolsas a estudantes programadores para contribuírem com projetos de software de código-fonte aberto). Para mais informações, acesse <http://www.openprinting.org/>.

2.10 - Sistemas de Arquivos

No Linux, sistema de arquivos (file system) refere-se comumente a mais de um item do sistema. É importante entender e diferenciar os dois significados de sistemas de arquivos, pois um está relacionado à hierarquia de arquivos e subdiretórios e o outro ao formato em que os dados são armazenados no disco.

Sistema de arquivos referindo-se à hierarquia de arquivos e subdiretórios é também conhecido como árvore de diretórios, utilizada para organizar a disposição dos arquivos no sistema. Nessa árvore os subdiretórios iniciam-se no diretório identificado como raiz, representado pelo caractere "/" (barra para frente), o qual possui uma série de subdiretórios encadeados e arquivos. Ele configura todo o sistema como arquivo, o que inclui não somente arquivos de textos, imagens ou programas, mas também diretórios, partições e dispositivos de hardware.

Essa estrutura possui um documento que especifica regras para o sistema de arquivos com o propósito de haver maior compatibilidade entre diferentes sistemas Linux, chamado Filesystem Hierarchy Standard (FHS), estudo mais adiante. A Figura 2.7 mostra um exemplo da estrutura em árvore do sistema de arquivos de um Linux.

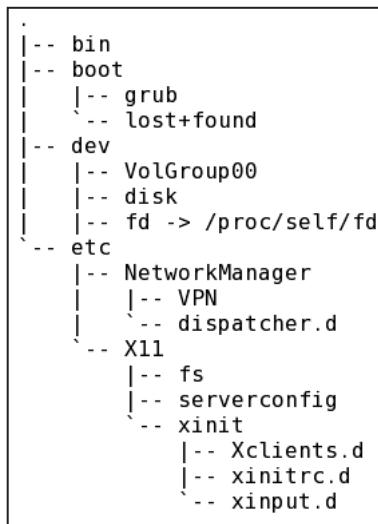


Figura 2.7 - Exemplo de sistema de arquivos em árvore.

A partir de agora vamos nos concentrar no outro sistema de arquivos, aquele que representa os tipos de armazenamento de dados ou de sistemas de arquivos.

A forma como os dados são organizados em um disco ou em uma partição dele constitui sistemas de arquivos. Cada um dos tipos de sistema de arquivos possui suas próprias regras de controle e alocação de espaço em disco para arquivos e para os dados referentes a eles, tais como o diretório em que eles estão localizados, o seu nome e as permissões.

A menor fatia de um sistema de arquivos são os inodes, nos quais ficam armazenados os dados e as informações dos arquivos. Basicamente os inodes são organizados em blocos de controle e de dados. Cada inode armazena as informações sobre os arquivos regulares, diretórios e outros arquivos do sistema, ou seja, nos inodes, além dos dados sobre as propriedades e localização dos arquivos, temos o conteúdo desses arquivos nos blocos de dados.

Os arquivos normais são armazenados nesses blocos de inodes. Diretório é um arquivo que contém uma lista com os endereços que apontam para os inodes de cada arquivo e/ou subdiretório existente no diretório. A Figura 2.8 indica um exemplo genérico dessa estrutura de inodes.

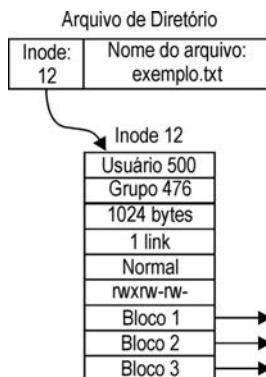


Figura 2.8 - Diagrama básico de diretório e inode.

Existem várias implementações de sistemas de arquivos no Linux, cada uma com suas características próprias e propósitos. Antes de estudarmos um pouco as principais, vamos entender o que são partições e conhecer um recurso muito importante utilizado nos sistemas de arquivos mais modernos, chamado journaling.

2.10.1 - Partição

Um disco pode ser dividido em partições, que são fatias lógicas independentes. É possível haver sistemas de arquivos diferentes, ou seja, num mesmo disco pode haver várias partições e cada uma delas será tratada como uma unidade de disco independente no sistema operacional. Quando compramos um disco novo, normalmente, ao instalar um sistema operacional, essas partições são criadas automaticamente. Em um sistema Linux temos pelo menos duas partições, sendo uma para os arquivos e outra para a memória

virtual, conhecida como partição de SWAP, a qual possui um sistema de arquivos com o mesmo nome.

Um disco novo, que nunca foi utilizado, precisa ser inicializado. Inicializar um disco significa basicamente criar a sua tabela de partições, a qual indica o tipo, o tamanho e a posição das partições no disco, entre outras informações, e mesmo que não haja nenhuma, a estrutura deve ser criada. Após criar uma partição, para poder disponibilizá-la para uso, é preciso escolher um tipo de sistema de arquivo e formatá-la.

Com a evolução dos métodos de armazenamento em disco surgiu uma nova camada intermediária denominada LVM (Logical Volume Management), que também é uma partição. A LVM é capaz de gerenciar diferentes tipos de sistemas de arquivos. Em termos práticos, com a LVM criamos uma grande partição no sistema, a qual pode ter mais de um Physical Volume (PV), ou seja, mais de um disco ou mais de uma partição LVM inicializada como PV, e dentro do sistema operacional pode ser visualizada como um único volume chamado Volume Group (VG).

Em um grupo de volumes é possível criar diversos Logical Volumes (LV), partições virtuais que devem ser formatadas com um dos sistemas de arquivos compatíveis. Após a formatação, essa partição fica disponível para uso como se fosse uma partição normal. A Figura 2.9 exibe um diagrama da LVM.

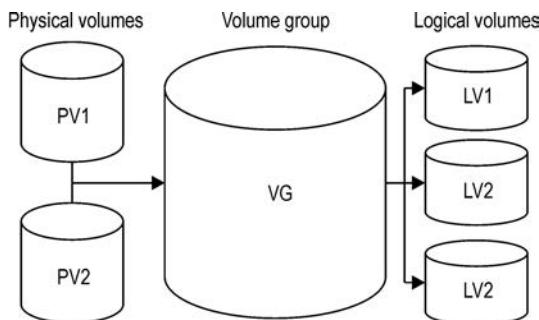


Figura 2.9 - Diagrama básico da LVM.

A principal vantagem do uso de volumes lógicos sobre um grupo de volumes do tipo LVM é a possibilidade de redimensionar e mover essas partições sem perda de dados.

2.10.2 - Journaling

Um sistema de arquivos com journaling é capaz de manter em um log, chamado journal, todas as alterações no sistema de arquivos antes de gravá-lo efetivamente no disco.

Em termos práticos, sistemas de arquivos com journaling são mais resistentes a falhas em arquivos e se recuperam mais rapidamente. Quando há um problema, como um travamento ou desligamento indevido do sistema em um computador com sistema de arquivos desse tipo, a probabilidade de os dados serem corrompidos é de longe muito

menor e o tempo de recuperação do sistema é significativamente menor se comparado a um sistema de arquivos sem journal.

A recuperação do sistema torna-se mais ágil nesses momentos adversos, porque com o journal o sistema de arquivos não necessita verificar todo o disco, mas somente os arquivos contidos nesse log. Na Figura 2.10 podemos observar um diagrama típico do funcionamento de um sistema com journaling.

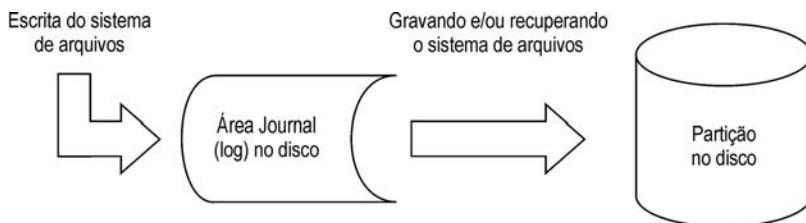


Figura 2.10 - Diagrama típico de funcionamento do journaling.

Os sistemas de arquivos com journaling compatíveis com Linux mais conhecidos são:

- ▶ **Ext3:** o Third Extended file system é uma reimplementação do famoso Ext2, primeiro sistema de arquivos popular do Linux. A principal diferença é que o Ext3 tem implementação para Journaling.
- ▶ **Ext4:** Fourth Extended file é uma reimplementação do Ext3 que enfatizou a melhoria das suas características e incluiu algumas funcionalidades importantes já existentes em outros sistemas de arquivos com journaling, como o ReiserFS. É atualmente o sistema de arquivos mais popular nas distribuições Linux para desktop.
- ▶ **JFS:** Journaling FileSystem é um sistema desenvolvido pela IBM inicialmente para o AIX, uma das variantes de UNIX criada por eles, atualmente de código aberto. Além do journal, uma característica interessante é a possibilidade de redimensionar as partições do sistema sem a necessidade de reiniciar o computador.
- ▶ **XFS:** é uma implementação de sistema de arquivos journaling iniciada pela Silicon Graphics para a sua variante de UNIX chamada IRIX. Atualmente é de código aberto. Dentre suas características, uma das mais importantes é o fato de ser um sistema de arquivos de 64 bits compatível com 32 bits.
- ▶ **Reiser4:** é uma evolução do ReiserFS, o primeiro sistema de arquivos para Linux com journal. Foi criado por Hans Reiser, mantido pela empresa Namesys e patrocinado pela Novell e Linspire.

2.11 - Tipos de Arquivos

Para facilitar o entendimento dos tipos de arquivos no Linux, é comum considerarmos que tudo em um sistema Linux é um arquivo, e se não for um arquivo, é um processo.

Apesar de esta afirmação ser verdadeira, existem arquivos e "arquivos", pois alguns não são simplesmente arquivos, sendo especiais, como pipes e conexões sockets.

Para os usuários, o sistema de arquivos é como uma hierarquia de diretórios que contêm outros subdiretórios e/ou arquivos. Neste contexto, os arquivos e subdiretórios são identificados pelo nome e os subdiretórios iniciam-se por "/", conhecido como root, directory root ou diretório raiz. Já para o kernel do Linux, no sistema de arquivos não há estrutura hierárquica, diferenciação entre diretórios, arquivos e programas, tampouco identificação dos arquivos por nomes, ou seja, na visão do kernel, não há diferença entre um arquivo e um diretório, uma vez que o diretório é somente um arquivo com informações dos arquivos que estão inseridos nele, conforme abordado no item anterior sobre sistemas de arquivos.

Neste contexto, temos que os programas, serviços, imagens e assim por diante são sempre arquivos e dispositivos de entrada e saída, e genericamente todos os outros dispositivos também são considerados arquivos para o sistema.

Os arquivos simples, aqueles que são somente arquivos, são conhecidos como arquivos regulares ou normais. Eles possuem dados comuns como arquivos de texto puro, executáveis ou programas.

Os arquivos que não são simplesmente arquivos podem ser:

- » **Diretórios:** arquivos que contêm informações em formato de lista sobre os arquivos contidos no diretório.
- » **Arquivos especiais:** um modelo de arquivo utilizado para entrada e saída de informações, por exemplo, para fazer referência a drivers de dispositivos. Normalmente estão localizados abaixo do subdiretório /dev.
- » **Links:** um mecanismo que torna um diretório ou arquivo visível em vários outros diretórios da árvore de diretórios do sistema de arquivos, sem a necessidade de copiá-lo para esses outros caminhos.
- » **Sockets:** um tipo de arquivo especial, similar aos sockets do TCP/IP, que provê um serviço de "rede" seguro para comunicação entre os processos.
- » **Pipes:** funcionam de forma muito parecida com os sockets, pois disponibilizam um caminho para comunicação de um processo com outro, porém não é semelhante a uma conexão socket.

2.12 - Filesystem Hierarchy Standard (FHS)

O FHS (padrão para sistema de arquivos hierárquico) é um projeto que era mantido pela Free Standards Group, que se transformou em Linux Foundation, como vimos no item 2.9. A versão atual é a 2.3 e foi lançada em 29 de janeiro de 2004; desde então outras não foram lançadas. O projeto determina os principais diretórios e descreve os seus requisitos de conteúdo para um sistema operacional GNU/Linux e as variantes do UNIX.

O objetivo do padrão criado no projeto é ser utilizado por desenvolvedores de distribuição e de programas e consultores de sistema. Entretanto, esse documento serve apenas como uma referência, não sendo um manual nem tampouco um tutorial que define como construir um sistema de arquivos ou hierarquia de diretórios em sistemas GNU/Linux

e UNIX. A grande maioria das distribuições Linux não adota o padrão proposto em sua totalidade, principalmente alguns diretórios criados pelo FHS, como, por exemplo, o /srv/, que nunca teve grande aceitação na comunidade GNU/Linux. Alguns sistemas GNU/Linux e UNIX desprezam totalmente o padrão FHS e outros fazem apenas algumas alterações. Em destaque podemos citar o Mac OS X, que mantém a maioria das funções dos diretórios descritos no FHS, no entanto seus nomes são mais compridos e legíveis, facilitando o entendimento de usuários leigos.

O diretório raiz, simbolicamente representado por "/", é o nível mais alto da hierarquia de arquivos do FHS. Seu conteúdo deve ser suficiente para iniciar, restaurar, recuperar e reparar o sistema.

Na raiz são requeridos os seguintes diretórios:

- ▶ **/bin:** comandos binários essenciais para todos os usuários. Alguns podem ser utilizados tanto pelo administrador do sistema quanto pelos usuários normais, mas que sejam necessários quando não houver outros sistemas de arquivos montados. Pode conter também comandos utilizados indiretamente por shell scripts e não deve haver subdiretórios.
- ▶ **/boot:** arquivos estáticos dos gerenciadores de boot. Ele deve conter tudo que for necessário ao processo de inicialização. Armazena os dados utilizados antes de o kernel iniciar a execução de programas em modo usuário.
- ▶ **/dev:** arquivos de dispositivos. Aqui ficam os arquivos especiais ou dispositivos do sistema.
- ▶ **/etc:** arquivos de configuração específicos dessa instalação de sistema (host). Um arquivo de configuração é um arquivo local utilizado para ajustar as preferências de operação de um programa. Nessa pasta não pode haver nenhum programa executável.
- ▶ **/lib:** este diretório contém as bibliotecas compartilhadas essenciais e módulos do kernel. Ela deve conter as imagens das bibliotecas para a inicialização do sistema e para executar os comandos binários dos diretórios /bin e /sbin.
- ▶ **/media:** reservado para ser utilizado como diretório inicial para os pontos de montagens das mídias removíveis. Ele deve conter os subdiretórios que são utilizados como ponto de montagem para as mídias removíveis como pendrive, DVD-ROM e CD-ROM.
- ▶ **/mnt:** ponto de montagem para sistemas de arquivos montados temporariamente. Esse diretório foi criado para os administradores de sistemas montarem sistemas de arquivos quando for necessário. O seu conteúdo deve ser de uso temporário e local, e não pode influenciar de maneira nenhuma o funcionamento de qualquer programa do sistema.
- ▶ **/opt:** diretório para a instalação de pacotes de programas de terceiros, que não fazem parte da distribuição instalada. Dentro de /opt deve existir um diretório para cada programa instalado que precisa conter todos os binários executáveis desse pacote.
- ▶ **/sbin:** executáveis do sistema. Programas essenciais para a administração do sistema ou que somente podem ser executados pelo administrador dele. São armazenados em /sbin, /usr/sbin e /usr/local/bin. Nesses diretórios estão os programas essenciais para a inicialização, restauração, recuperação e reparação do sistema, além daqueles

em /bin. Em /usr/sbin devem ficar os arquivos executáveis que não são vitais para o sistema e no /usr/local/sbin, os arquivos instalados localmente pelo administrador do sistema.

- » **/srv:** dados dos serviços providos por esse sistema. Esse diretório deveria conter os dados específicos dos serviços oferecidos pelo sistema, no entanto vale ressaltar que ele nunca foi adotado em larga escala, ou seja, não há distribuições populares que o utilizam.
- » **/tmp:** arquivos temporários. Ele disponibiliza um local para gravar os arquivos temporários dos programas, mas não se pode assumir que os arquivos e diretórios nele hospedados sejam preservados entre uma execução e outra do programa.
- » **/usr:** hierarquia secundária. É a segunda maior seção do diretório raiz. Deve estar compartilhado como somente leitura, ou seja, qualquer informação específica da máquina ou que varie com o tempo não pode ficar nesse diretório. Pacotes de software muito grandes não devem ser instalados abaixo de /usr. Alguns subdiretórios são requeridos em /usr, como o bin, no qual fica a maioria dos comandos do usuário; o include para os arquivos de cabeçalhos dos programas em C; o lib que guarda as bibliotecas do usuário; o local para as instalações locais, ficando vazio após a instalação; o sbin para armazenar os programas não vitais do sistema e o share para os dados de somente leitura independentes de arquiteturas.
- » **/var:** contém arquivos de dados variáveis. Inclui diretórios e arquivos de spool, de administração e logs, além de arquivos transientes e temporários. Não pode ser montado em uma partição separada. Em função disso, quando o espaço do diretório raiz é pequeno, utiliza-se o /usr/var em vez de /var, no entanto um link para /var sempre deve existir para não haver conflito entre as definições de /usr e /var. Alguns subdiretórios são requeridos, como o cache que armazena os dados de cache das aplicações; o lib que contém as informações sobre o estado da variável; o local para os dados variáveis de /usr/local; o lock para os arquivos de bloqueios; o log que contém os arquivos e diretórios de log; o opt para os dados variáveis de /opt; o run que hospeda os dados relevantes dos processos em execução; o spool que contém os dados de spool dos programas e o tmp para armazenar os arquivos temporários que devem ser preservados mesmo após o reinício do sistema.

Além dos diretórios requeridos, temos os opcionais, entre os quais dois são encontrados em praticamente todas as distribuições GNU/Linux. São eles:

- » **/root:** local padrão para hospedar os arquivos e preferências de alguns programas do usuário administrador do sistema. Ele pode ser trocado, no entanto é o local mais indicado para isso. Esse diretório normalmente é de acesso restrito ao administrador.
- » **/home:** local padrão para armazenar as pastas de documentos e preferências de alguns programas dos usuários. Normalmente cada usuário possui um subdiretório com o nome de usuário e abaixo dele estão esses arquivos. Na maioria das vezes o subdiretório do usuário só pode ser acessado por ele mesmo e pelo administrador do sistema.

A versão completa do FHS pode ser encontrada em <http://www.pathname.com/fhs/>.

2.13 - As Distribuições do Linux

O conceito de distribuição Linux surgiu ao final de 1992, quando as versões do Kernel Linux já estavam bastante evoluídas e com características distintas do UNIX. Nessa época os usuários começaram a montar "kits de instalação" que acompanhavam uma série de pacotes, contendo o Kernel Linux e, principalmente, os aplicativos do projeto GNU, transformando-se em um enorme pacote com dezenas de disquetes que os usuários trabalhosamente baixavam pela Internet.

Esses pacotes enormes que se transformariam em distribuições foram inicialmente construídos por estudantes e entusiastas de software livre, logo chamando a atenção de empresas e instituições que fomentaram cada vez mais a ideia. Um detalhe importante é que nessa época não havia nenhum padrão estabelecido, como o LSB e o FHS vistos anteriormente; logo, essas distribuições eram de acordo com o sentimento de cada um, o que dificultava muito a sua difusão.

Com o tempo a comunidade se organizou e hoje temos diversos padrões que estreitam a compatibilidade entre as distribuições, permitindo que os usuários se adaptem facilmente a qualquer uma.

Não confunda distribuição com sistema operacional. O kernel do Linux é o sistema operacional e uma distribuição pode ser o conjunto de vários aplicativos, programas ou utilitários que rodam no sistema operacional Linux. Não se pode generalizar que o kernel e os pacotes de uma distribuição são o sistema operacional, pois desta forma, além de menosprezar o principal elemento do Linux, o kernel, equaliza-se a visão de sistemas proprietários como o Windows, em que os usuários não conseguem distinguir o sistema operacional dos aplicativos, parecendo que tudo é um grande sistema operacional e limitando o controle dos usuários.

Com base nesta afirmação, podemos definir distribuição como um sistema operacional completo que possui um kernel (o núcleo do sistema operacional), utilitários (itens necessários ao funcionamento do sistema operacional) e uma variedade de programas e aplicativos.

Embora tenhamos uma miscelânea de distribuições Linux, encontraremos muitas semelhanças entre elas em função dos padrões de construção de um sistema Linux. Como sabemos, em uma distribuição Linux toda instalação pode ser baseada em necessidades e preferências pessoais, o que vai depender do nível de cada usuário. Quanto mais avançado for o usuário, mais ele vai personalizar o sistema, pois a instalação do sistema é apenas o início de um relacionamento de longo prazo. Quando imaginar que está com um sistema completo, sendo um usuário entusiasta ou baseando-se em comunidades, será estimulado a melhorá-lo. E conforme se percebe o poder que o sistema oferece, mais estimulado a romper os próprios limites o usuário fica.

Então, apesar dos padrões de criação de distribuições, o Linux pode ser personalizado. Assim, ele pode parecer diferente dependendo da distribuição utilizada, do hardware e das preferências pessoais, mas existem alguns itens fundamentais que estão presentes em todas as distribuições Linux. Uma distribuição Linux é baseada principalmente nas ferramentas

do projeto GNU, que provém um conjunto básico de controle e uso do sistema por meio de seus programas, utilitários e aplicativos, como, por exemplo, o compilador gcc e o editor de textos vi, que sem dúvida estarão disponíveis em 99,99% das distribuições existentes.

A maioria das distribuições fornece pacotes de programas pré-compilados das ferramentas mais comuns e todos os outros que o fazem uma distribuição, por meio de pacotes RPM (RedHat Package Manager) para as variantes de RedHat, dpkg (Debian Package) para as variantes do Debian e tgz para a distribuição Slackware e suas variantes.

A distribuição Slackware é uma das mais antigas, e foi baseada naquela que pode ser considerada a primeira distribuição que existiu oficialmente, chamada SLS (Safe Landing System Linux), criada com base no BSD (uma versão do UNIX chamada Berkeley Software Distribution). O Slack (apelido carinhoso dado pela comunidade) é a distribuição que mais se assemelha e mantém as características UNIX e seus usuários se orgulham disso, portanto pense em experimentar outras distribuições mais simples, como variantes do RedHat e Debian, antes de aventurar-se com ele. No entanto, quando achar que está apto, não deixe de conhecê-lo, pois é um verdadeiro parque de diversões para usuários avançados.

Você não precisa ser um programador para instalar um novo programa em um sistema Linux, pelo menos para alguns casos, entretanto, se for, pode desfrutar do melhor do Linux, fazendo tudo você mesmo, pois as distribuições normalmente fornecem uma coleção completa de ferramentas de desenvolvimento, possibilitando a instalação de pacotes personalizados a partir do código-fonte. Com as fontes você pode instalar qualquer programa no seu sistema, independente de fazer parte do pacote da sua distribuição ou não.

Sabendo que todas as distribuições possuem o kernel do Linux e um conjunto básico em comum que pode ser totalmente personalizado, antes de pensar em instalar devemos considerar o hardware, porque se escolhermos uma distribuição para um hardware incompatível com aquele que estamos tentando instalar, a instalação não se concluirá com sucesso, o que pode ser frustrante para iniciantes, principalmente se for a primeira experiência com Linux.

Neste contexto, você precisa saber se a distribuição vai funcionar em seu hardware, pois existem distribuições para rodar em diversas arquiteturas de hardware diferentes, como Macintosh, PowerPC, IBM PC x86, Sun Sparc, Sun UltraSparc e até mesmo para Playstation 2. Porém, para a quase totalidade de usuários desktops no Brasil, as distribuições ideais são aquelas baseadas na arquitetura IBM PC x86, a mais difundida no País.

Algumas distribuições são compiladas para um tipo específico de hardware, porém elas não são muito interessantes para o mundo corporativo, pois são testadas por poucos usuários. Em consequência disso, a maioria das distribuições possui um conjunto de pacotes para PCs genéricos e normalmente otimizados para os processadores Intel da família x86. Essas distribuições são muito difundidas, extremamente bem testadas, mantidas por uma base de profissionais regular, umas com foco em desktop e outras em implementações de servidores, além de oferecerem interface amigável para instalação e atualização dos pacotes.

Distribuição	Site	Comentários
CentOS	www.centos.org	O CentOS é uma das melhores opções gratuitas para substituir o não gratuito RedHat Enterprise, sendo totalmente baseada nele. É também a distribuição adotada como base para este livro.
Fedora	www.fedoraproject.org	Um projeto comunitário da RedHat, sendo gratuito e uma ótima opção para iniciantes mais autossuficientes, pois possui vasta documentação disponível na Internet. Ótima opção para quem utiliza hardwares mais atuais e gosta de estar na vanguarda do que há de novo no mundo do Linux e do código aberto.
Debian	www.debian.org	Uma das mais antigas e mais estáveis distribuições existentes. A comunidade Debian prima a estabilidade e não se aventura em novos recursos. É bastante conservadora e só libera novas versões após testes exaustivos. Ela possui o menor tempo de solução de problemas.
Gentoo	www.gentoo.org	Distribuição um pouco baseada em FreeBSD, sendo umas das melhores para quem gosta de fazer tuning ou configurações mais específicas. O Gentoo tem como característica marcante a alta capacidade de personalização avançada do sistema para qualquer necessidade.
Mandriva	www.mandriva.com	Boa opção para quem está começando. É uma das mais amigáveis neste sentido.
Ubuntu	www.ubuntu.com	Variante da Debian; é muito amigável, foi desenvolvida com o propósito de atingir usuários realmente iniciantes que não necessitam de muitas personalizações, ou seja, especial para desktops.
openSUSE	www.opensuse.org	Opção gratuita para a distribuição não gratuita SUSE Linux Enterprise. Boa opção para substituir servidores baseados no SUSE.
RedHat Enterprise	www.redhat.com	Distribuição comercial mantida pela RedHat, sem dúvida uma distribuição muito poderosa para servidores corporativos.
Slackware	www.slackware.com	Mais familiar ao UNIX. Conservadora, muito estável, extremamente básica e simples. Seu uso é recomendado para usuários mais experientes.
SUSE Linux Enterprise	www.novell.com/linux	Mais uma distribuição poderosa mantida por instituição privada, ótima opção para soluções corporativas.

O processo de instalação evoluiu muito ao longo do tempo, a ponto de permitir escolher entre vários tipos de configuração, como, por exemplo, uma simples estação de trabalho, que instala um conjunto de pacotes necessários para o uso diário em escritórios, ou uma estação de trabalho para desenvolvedores, com pacotes específicos para desenvolvimento, e por último uma instalação de servidor, que tem as ferramentas de administração padrão e os serviços selecionados pelo usuário. Ressaltamos que, dependendo do propósito da distribuição, outras opções de conjunto de pacotes podem estar disponíveis, assim como haverá distribuições que não oferecem esse tipo de recurso.

Quanto mais você experimentar as distribuições do Linux, mais experiência vai adquirir para configurar uma instalação nas combinações de pacotes que desejar, durante o processo de instalação. Aos iniciantes, contudo, é extremamente indicado utilizar distribuições bem difundidas, como as gigantes Fedora e Ubuntu para desktops e Debian e CentOS para servidores, pois elas possuem muito mais documentação disponível na Internet e o tempo para correção de problemas é extremamente breve.

2.14 - Onde Encontrar Ajuda

Caso precise de ajuda, a comunidade Linux celebra a autossuficiência. No Linux, por vezes temos várias maneiras de resolver um mesmo problema, então tenha calma quando estiver diante de um deles e tente ser o mais criterioso possível nos procedimentos que está executando. Anote tudo em ordem cronológica, todos os passos para reproduzir o erro e tudo que você já tentou fazer para resolvê-lo.

Após ter alcançado o limite de seus conhecimentos em relação ao problema, pode pedir ajuda a alguém ou utilizar um ou mais recursos dos abordados a seguir, a fim de tentar solucionar o problema.

2.14.1 - As Man Pages

A maioria dos usuários novatos teme as man pages, páginas do man ou páginas do manual, porque elas são uma grande fonte de informação em modo texto aparentemente para usuários avançados, no entanto é um grande equívoco, pois qualquer usuário consegue entendê-lo e a partir dele resolver muitos problemas. O único requisito para o uso de man pages é saber um pouco de inglês, podendo ser o inglês instrumental.

As man pages são muito bem estruturadas, como veremos no exemplo a seguir, usando o comando man man.

```
$ man man
                                         NOME
man - formatar e mostrar as páginas do manual on-line
manpath - determinar o manpath inicial para o utilizador

SINOPSE
       man [-acdfhktw]  [-m sistema]  [-p string]  [-C fich_config]  [-M path]
       [-P paginador]  [-S lista_sec] [secção] nome ...

DESCRIÇÃO
       man formata e mostra as páginas do manual 'on-line'. Esta versão reconhece as
       variáveis de ambiente (environment) MANPATH e (MAN) PAGER (ver a seguir). Se
       a secção for indicada, man apenas procura nessa secção de manuais. Pode também
       indicar por que ordem das secções deve procurar e que
       pré-processamento efetuar nos manuais, por meio de opções na linha de comando
       ou variáveis de ambiente. Se nome contiver uma / tentará primeiro o ficheiro
       com esse nome, permitindo fazer man ./foo.5 ou mesmo man /cd/foo/bar.1.gz para
       formatar e ver um ficheiro em particular.

OPÇÕES
       -C fich_config
           Indica o ficheiro de configuração a usar; por omissão será usado /etc/man.
           config. (Veja man.conf(5).)
```

Que incrível! O man tem a capacidade de mostrar seu próprio manual, ou seja, a documentação do man foi exibida na tela após pressionar Enter. Abaixo do exemplo do comando man man temos um trecho do que você deve ter visto; não estranhe por estar em português, pois existem muitos manuais em português, mas ainda são predominantes as documentações em inglês. Fique atento!

Dica

Você pode acessar as man pages por meio de um terminal, seja em modo gráfico ou em um console de texto puro, conforme sua preferência.

Com o manual aberto, para navegar entre as páginas, use a barra de espaço se quiser avançar e a letra "b" no teclado para retroceder. Para sair a qualquer instante, pressione a letra "q" no teclado e você voltará imediatamente ao prompt de comando.

Cada manual normalmente contém uma coleção de seções padrão. Vejamos a seguir quais são:

- ▶ A primeira linha contém o nome do programa em que estamos lendo o manual e o id da seção onde está localizada a man page. Um único comando pode conter diversas páginas, pois as man pages são organizadas em capítulos.
- ▶ **NOME ou NAME:** mostra o nome do comando e uma breve descrição utilizada no índice das man pages. Podemos procurar qualquer palavra nesse índice usando o comando apropos, que veremos a seguir.
- ▶ **SINOPSE ou SYNOPSIS:** aqui temos sintaxes possíveis do uso do comando com todas as opções e argumentos disponíveis. Entenda como opção as várias formas de se executar um determinado comando, e argumento são as variáveis dessas opções. O exemplo anterior invoca o programa man com o argumento man. As opções e os argumentos que não são obrigatórios aparecem entre colchetes - "[" e "]" - indicando que podem ser desprezados. Vale lembrar que alguns comandos não possuem opções ou argumentos.
- ▶ **DESCRIÇÃO ou DESCRIPTION:** descrição completa do comando.
- ▶ **OPÇÕES ou OPTIONS:** descreve detalhes de cada opção possível. Normalmente elas podem ser utilizadas de forma combinada, encadeando diversas opções em um mesmo comando.
- ▶ **AMBIENTE ou ENVIRONMENT:** indica as variáveis do ambiente que podem influenciar no funcionamento desse comando.
- ▶ **VEJA TAMBÉM ou SEE ALSO:** nessa seção são dadas referências a outras man pages. Entre os parênteses está o número da seção man page que faz referência ao comando.
- ▶ **BUGS:** indica os bugs conhecidos e como relatar novos erros.
- ▶ **AUTOR ou AUTHOR:** informações sobre o autor.
- ▶ **COPYRIGHT:** informações sobre os direitos de uso.

Por vezes, na seção VEJA TAMBÉM ou SEE ALSO pode haver referências para outros man pages, conforme dito anteriormente. Veja a seguir um exemplo desse conteúdo para o comando man ssh:

SEE ALSO

```
scp(1), sftp(1), ssh-add(1), ssh-agent(1), ssh-keygen(1), ssh-keyscan(1),  
tun(4), hosts.equiv(5), ssh_config(5), ssh-keysign(8), sshd(8)
```

Chamamos a atenção para alguns comandos que podem conter várias man pages. Neste caso o programa sempre exibe, por padrão, somente a primeira seção de man page. No exemplo anterior, observe que uma das referências dadas é `ssh_config(5)`. Este número entre parênteses indica em qual man page do comando `ssh_config` está a referência para o comando em que estamos lendo a man page. Para visualizar uma man page específica de um comando, informe o número dela antes do nome do programa sobre o qual deseja mais informações. No exemplo seguinte é solicitada a seção 5 da man page do comando `ssh_config`:

```
$ man ssh_config 5
```

Se você não sabe quantas man pages tem um comando, ou deseja visualizar todas de uma única vez, use:

```
$ man -a ssh_config
```

Esse comando abre as man pages em ordem crescente, da menor para maior. Para avançar as páginas, pressione a letra "q" no teclado. Quando atingir a última man page, o man será finalizado e você voltará ao prompt imediatamente.

Uma sugestão é a possibilidade de buscar uma palavra ou expressão em qualquer parte do texto da man page aberta. Para isso, use a barra normal "/" e o texto a ser procurado, como "/BUGS". Isso faz o man pular para o item BUGS, quando houver, é claro.

2.14.2 - As Info Pages

Além das man pages, podemos procurar ajuda nas info pages, páginas de informação, por meio do comando `info`. As informações contidas nele geralmente são mais recentes e por vezes, em alguns comandos, o man page faz referência a info page.

Em vias gerais, é mais fácil utilizar as info pages, portanto mais acessíveis aos novatos.

Para começar, vejamos um exemplo como o anterior. Vamos pedir a info pages que mostre a sua própria documentação, com o comando `info info` em um terminal ou janela console:

```
$ info info  
  
File: info.info, Node: Top, Next: Getting Started, Up: (dir)  
Info: An Introduction  
*****  
The GNU Project distributes most of its on-line manuals in the "Info format",  
which you read using an "Info reader". You are probably using an Info reader  
to read this now.  
There are two primary Info readers: `info', a stand-alone program designed  
just to read Info files (*note Stand-alone Info: (info-stnd)Top.), and the `info'  
package in GNU Emacs, a general-purpose editor. At present, only the Emacs  
reader supports using a mouse.  
If you are new to the Info reader and want to learn how to use it, type the
```

command `h' now. It brings you to a programmed instruction sequence.

To read about advanced Info commands, type `n' twice. This brings you to 'Advanced Info Commands', skipping over the 'Getting Started' chapter.

- * Menu:
- * Getting Started:: Getting started using an Info reader.
- * Advanced:: Advanced Info commands.
- * Expert Info:: Info commands for experts.
- * Index:: An index of topics, commands, and variables.

Para navegar, utilize as setas para saltar de uma linha para a outra, page down e page up para avançar ou retroceder as páginas respectivamente. As linhas que começam com asteriscos são links para o assunto, podendo ser usadas como um sumário no info. Se colocarmos o cursor em uma dessas linhas e pressionarmos Enter, o info nos leva à página que trata do assunto em questão automaticamente. Também é possível navegar entre um assunto e outro usando as letras "p" para voltar e "n" para avançar ao próximo assunto.

Para sair, pressione a letra "q" no teclado.

Dica

O programa info normalmente tem mais informações que o man. Por vezes pode não haver nenhuma informação de determinado comando no info, no entanto existente no man, por isso devemos utilizar os dois. Experimente o comando info man e veja que o info mostra, além do conteúdo do comando man man, muitas outras informações.

2.14.3 - O Comando Whatis

Ele procura nas man pages um determinado comando e exibe uma breve descrição dele, que seria o mesmo conteúdo do campo NAME ou NOME nas man pages.

```
$ whatis cp
cp                  (1) - copy files and directories
cp                  (1p) - copy files
```

2.14.4 - O Comando Apropos

Também exibe uma descrição breve do man pages de um comando, no entanto com ele é possível procurar palavras nessa descrição, o que facilita muito quando não sabemos qual item queremos ou precisamos usar, por exemplo:

```
$ apropos "copy files"
File::Copy          (3pm) - Copy files or filehandles
cp                 (1) - copy files and directories
cp                 (1p) - copy files
cpio               (1) - copy files to and from archives
install            (1) - copy files and set attributes
```

Desta forma, supondo que não soubéssemos qual programa utilizar para efetuar uma cópia de arquivo, descobriríamos rapidamente pedindo ao apropos para procurar a expressão "copy files", conforme descrito no exemplo anterior.

2.14.5 - A Opção --Help

A maioria dos programas possui uma ajuda breve que explica como utilizar o comando e uma lista das opções existentes. No exemplo a seguir vemos a opção --help do comando man:

```
$ man --help
man, versão 1.6f
uso: man [-adfhktwW] [seção] [-M path] [-P paginador] [-S lista] [-m sistema]
[-p string] nome ...
  a : encontrar todas as entradas
  d : imprimir informação de debug
  f : o mesmo que whatis(1)
  h : imprimir esta ajuda
  k : o mesmo que apropos(1)
  t : usar troff para formatar o manual para imprimir
  w : imprimir a localização do manual a ver(se não indicar o nome: imprime as
directórias dos manuais)
  M path      : indica 'path' como as directórias dos manuais
  P paginador : use o programa 'paginador' para ver os manuais
  S lista     : indicar lista de secções (separadas por ,)
  m sistema   : procura manual para o sistema indicado
  p string    : string indica o pré-processamento a efetuar
```

Essa opção ajuda bastante e é disponibilizada rapidamente ao usuário.

Por fim haverá comandos que não possuem documentação distinta porque eles fazem parte de um outro comando. Um exemplo prático são os comandos exit, pwd, cd e logout, pois eles são componentes do Shell em execução, estando embutidos nele. Para ter ajuda em relação a eles, invoque o man pages para o Shell que está utilizando, por exemplo:

```
$ man sh
usuário@host~> man sh
```

Exercícios

1. O que é POSIX?
2. Qual o objetivo do kernel (núcleo) do Linux?
3. Qual a diferença entre um kernel monolítico e microkernel?
4. Como identificar se uma versão do Linux é estável ou está em desenvolvimento?
5. O que é o Shell?
6. Como funciona o Shell?
7. Quais são as principais tarefas da ordem de execução de comandos do Shell?
8. Cite exemplos de Shells disponíveis no Linux.
9. O que é multitarefa?
10. Como deve ser a multitarefa em um bom sistema operacional?
11. Apesar de os processadores executarem uma tarefa por vez, como é possível a multitarefa em um sistema operacional?
12. Como funciona um sistema operacional monotarefa?
13. Qual a diferença entre multitarefa cooperativa e preemptiva?
14. O que significa o termo multiusuário?
15. Quais os dois tipos de usuários do Linux e suas principais características?
16. Em que consiste a memória virtual?
17. Como funciona a memória virtual em um sistema operacional?
18. Por que devemos nos preocupar com o uso em excesso da memória virtual?
19. Quais as duas interpretações para o termo sistema de arquivos?
20. O que são partições?
21. Para que serve uma partição do tipo LVM? Quais são as vantagens?
22. O que são sistemas de arquivos com Journaling?
23. Quais as vantagens de um sistema de arquivos com Journaling?
24. Quais são os tipos de arquivo do Linux?
25. O que é e qual o objetivo do FHS?
26. Descreva resumidamente cada um dos diretórios requeridos no diretório raiz no FHS.
27. Quando e como surgiram as distribuições Linux?

Exercícios

- 28.** Por que não devemos considerar que uma distribuição Linux é um sistema operacional?
- 29.** Quais os pacotes (ferramentas) encontrados em praticamente todas as distribuições Linux?
- 30.** Como são disponibilizados os pacotes de programas pré-compilados?
- 31.** Quais as principais diferenças entre info e man pages?
- 32.** Para que servem os programas whatis e apropos?

3

Tarefas Essenciais do Administrador

3.1 - Comandos Básicos

Este tópico apresenta os comandos básicos para o dia a dia do profissional responsável pela administração de uma máquina Linux, sendo útil também a usuários em geral, quando for necessária uma intervenção diretamente com o sistema operacional (apesar de, na maioria das ocasiões, a interface gráfica ser suficiente para esses usuários). Mostra os comandos na interface texto do Linux, pois esta é a melhor forma de interagir completa e agilmente com o sistema operacional (é claro que nada impede que o administrador utilize a interface gráfica do Linux, mas em muitas situações ela será um limitador). Caso você já tenha domínio desses comandos, pode pular este tópico.

Vamos aprender agora como cumprir as tarefas mais simples no sistema operacionais, tais como listar o conteúdo de um diretório, criar um arquivo ou diretório, alterar seu nome e permissões e excluir um arquivo ou diretório. Além disso, veremos outras tarefas simples, mas que permitem a plena interação do administrador com o sistema operacional.

3.1.1 - Obtenção da Ajuda do Linux

Antes de entrarmos no assunto, é preciso saber que mesmo no modo texto do Linux existe uma ajuda disponível (às vezes completa, outras bem simples, sempre disponível para garantir um mínimo de auxílio).

Existem várias formas de obter uma preciosa ajuda do Linux:

- » **Ajuda do próprio comando:** basta digitar o nome do comando seguido de --help para obtermos alguma ajuda, a qual, na maioria das vezes, é apenas a sintaxe do comando e a lista de opções disponíveis. Veja um exemplo:

```
# net --help
```

Para os comandos nativos do sistema operacional temos três formas extras de obter ajuda:

- » **Pelo comando help:** basta digitar o comando help seguido do comando sobre o qual desejamos a ajuda, exemplo:

```
# help net
```

- » Pelo comando **man**: sua sintaxe é igual à do comando help:

```
# man net
```

- » Pelo comando **info**: sintaxe idêntica aos demais, exemplo:

```
# info net
```

Cada um destes comandos apresenta um formato de ajuda, e são muitas vezes complementares.

3.1.2 - Gerenciamento de Diretórios e Arquivos

As tarefas mais comuns no quesito gerenciamento de diretórios e arquivos no Linux são:

- » Criação de um diretório
- » Alteração do nome do diretório ou arquivo
- » Alteração das permissões
- » Remoção de um diretório ou arquivo
- » Exibição da lista de arquivos e diretórios

3.1.2.1 - Criação de um Diretório

Para criar um diretório, utilizamos o comando **mkdir** que possui a seguinte sintaxe:

```
Mkdir <opções> <nome do diretório>
```

O comando mkdir tenta criar o diretório especificado conforme as opções informadas. Caso já exista um diretório ou arquivo com o mesmo nome, uma mensagem de erro é retornada e o diretório não é criado.

A utilização básica do comando é:

```
Mkdir <nome do diretório>
```

Por exemplo:

```
# Mkdir teste
```

Esse comando tenta criar na pasta atual o diretório teste com as permissões padrão do usuário.

As opções disponíveis para o comando mkdir são:

- m ou –mode= Define a máscara de criação do diretório. Caso não seja informada, a máscara padrão do usuário é utilizada.
- p ou –parents Informa ao comando que toda árvore informada deve ser testada e caso o diretório não exista, deve ser criado. Caso não seja informada essa opção, o comando retorna um erro, se alguns dos diretórios da árvore não existirem, exceto o último.

-v ou --verbose Exibe o que está sendo executado, muito útil junto com a opção -p.

Exemplos:

```
# mkdir --m 777 teste  
# mkdir --p teste1/teste2/teste3  
# mkdir --v -p teste11/teste12/teste13
```

O primeiro exemplo cria o diretório teste com a máscara 777 (veja mais sobre permissões no item 3.1.2.3), ou seja, com permissão de leitura, gravação e execução para todos os usuários e grupos.

O segundo cria a árvore teste1/teste2/teste3.

O terceiro exemplo cria a árvore e exibe o que está sendo executado, neste caso teremos:

```
Mkdir: created directory 'teste11'  
Mkdir: created directory 'teste11\teste12'  
Mkdir: created directory 'teste11\teste12\teste13'
```

3.1.2.2 - Alteração do Nome de um Diretório ou Arquivo

No Linux, diretórios são gerenciados como arquivos, por isso vários comandos utilizados para arquivos servem também para diretórios. Desta forma, para alterar o nome de um diretório, podemos utilizar os comandos mv ou rename, também usados para arquivos.

O comando mv é simples e direto, permite a alteração do nome de um diretório e possui opções para esse controle. Já o comando rename é mais complexo e permite a alteração controlada do nome de vários arquivos.

O comando mv possui a seguinte sintaxe:

```
Mv <opções> nome_atual novo_nome
```

O padrão do comando é gerar um erro caso o novo_nome informado já exista. A forma mais simples de utilização do comando é:

```
# mv teste testex
```

As opções disponíveis mais úteis são:

-f ou --force Força a alteração do nome_atual para novo_nome, mesmo que novo_nome já exista; neste caso o arquivo ou diretório existente será perdido.

-b ou --backup Uma cópia do arquivo ou diretório existente é criada antes que a troca de nome seja executada.

-i ou --interactive Solicita confirmação antes de sobrescrever o arquivo ou diretório existente.

Exemplos:

```
# mv -f teste teste1  
# mv -f -b teste teste1  
# mv -f -i teste teste1
```

O comando rename realiza trocas complexas, como em vários arquivos que precisam ter seus nomes trocados. Podemos utilizar critérios de busca para que a troca seja realizada. Sua sintaxe é:

Rename <de> <para> <lista_arquivos>

O comando substitui a primeira ocorrência de <de> por <para> encontrada em cada um dos arquivos da lista informada. A lista informada pode conter caracteres especiais de pesquisa, tais como ? ou *. O caractere ? indica qualquer letra, número ou símbolo e o caractere * indica qualquer sequência de caracteres. Veja alguns exemplos:

```
# rename teste teste0 teste?  
# rename teste teste0 teste*  
# rename teste teste0 teste??
```

O primeiro exemplo troca teste por teste0 em qualquer arquivo que comece por teste e contenha mais um caractere qualquer no seu nome, tais como teste1, teste2, testex, teste_, mas não trocará em teste, teste11, testexx ou qualquer outro arquivo que tenha mais de um caractere após teste.

O segundo exemplo troca teste por teste0 em qualquer arquivo que contenha teste, seguido de zero ou mais caracteres extras.

O terceiro troca teste por teste0 em qualquer arquivo que contenha teste seguido de dois outros caracteres, assim teste11, teste22, testex1, teste_a serão afetados, já teste, teste1, testex, testa, teste111, teste_2 não serão.

3.1.2.3 - Alteração das Permissões de um Diretório ou Arquivo

Aqui temos novamente um comando comum tanto para arquivos quanto para diretórios. Utilizamos o comando chmod para alterar as permissões de um diretório ou arquivo.

Tem a seguinte sintaxe:

Chmod <permissões> <lista_de_arquivos>

As permissões podem ser atribuídas a três categorias:

- » Usuário dono do arquivo ou diretório
- » Grupo ao qual o arquivo pertence
- » Demais usuários do sistema
- » Para cada uma destas categorias podemos definir as permissões para leitura, escrita e execução do arquivo/diretório.

Para definir as permissões, podemos utilizar basicamente dois formatos:

Octal

Informamos as permissões por valores octais, e cada categoria de permissão é definida por um dígito octal. Os valores possíveis são:

- » **0:** nenhuma permissão
- » **1:** permissão para execução
- » **2:** permissão para escrita
- » **4:** permissão para leitura

Os valores podem ser combinados, isto é, somados para obter a permissão desejada. Por exemplo, se desejamos liberar leitura e gravação, utilizamos o valor 6; para permitir tudo, utilizamos o valor 7.

No formato octal cada dígito define as permissões para uma categoria. Como temos três categorias, devemos utilizar três dígitos octais para definir as permissões. Veja um exemplo:

```
# chmod 522 teste
```

Agora o arquivo teste terá permissão de leitura e execução para o dono do arquivo e de escrita para os demais usuários.

Modo simbólico

Nesse modo informamos por meio de símbolos as permissões que desejamos atribuir. É mais flexível que o modo octal, porém mais complicado de ser utilizado. O formato de utilização é:

```
[ugoa] [+-=] [rwx][,...]
```

Sendo:

- u Usuário dono do arquivo/diretório
- g Grupo ao qual o arquivo pertence
- o Outros usuários do sistema
- a Todos
- + Adiciona as permissões
- Remove as permissões
- = Define as permissões (substitui)
- r Leitura
- w Escrita
- x Execução

Desta forma, se desejamos adicionar a permissão de escrita ao grupo ao qual o arquivo pertence, devemos fazer g+w. Podemos atribuir várias permissões, bastando separá-las por vírgula. Veja um exemplo:

```
# chmod a=,u=rwx,g=r,o=x,g+w,o-x teste
```

Ao final da execução, o usuário dono do arquivo terá permissão para leitura, escrita e execução, já os usuários pertencentes ao grupo dono do arquivo não terão permissão e os demais usuários somente terão permissão de executar o arquivo.

3.1.2.4 - Remoção de um Diretório ou Arquivo

Temos aqui dois comandos. Um exclui arquivos e pode também remover diretórios, o outro remove apenas diretórios.

O comando que exclui arquivos e pode ser utilizado para excluir diretórios (o padrão do comando é não remover diretório. Para alterá-lo, devemos utilizar uma opção do comando) é o rm. Em seu formato mais simples, precisamos apenas do comando seguido da lista de arquivos a serem excluídos, sendo possível a utilização de caracteres especiais para definição da lista, tais como * e ?. Sua sintaxe é:

```
Rm <opções> <lista de arquivos>
```

A lista de arquivos pode ser uma sequência de nomes separados por espaço ou apenas o nome de um único arquivo, sendo possível a utilização dos caracteres especiais de busca em qualquer lugar da lista ou no nome do arquivo. Veja alguns exemplos:

```
# rm teste
```

```
# rm tes*
```

```
# rm teste??
```

```
# rm teste teste0001 teste02 teste003*
```

Caso algum dos arquivos da lista seja um diretório, a mensagem “rm: cannot remove directory '<nome>': É um diretório” ou semelhante (dependendo da distribuição utilizada) será exibida e o diretório não é excluído.

As opções disponíveis (principais) são:

-i ou --interactive Pergunta antes de excluir cada um dos arquivos da lista.

-f ou --force Não faz nenhum questionamento antes de excluir e ignora os arquivos não encontrados.

-r ou --recursive Exclui diretórios e todos os seus arquivos recursivamente, ou seja, caso existam diretórios ou arquivos dentro do diretório a ser excluído, estes serão excluídos antes da exclusão do diretório.

Logo, para eliminar a mensagem de erro na remoção de diretórios, devemos utilizar a opção `-r` do comando `rm`. Veja um exemplo:

```
# rm -r teste*
```

Para a remoção exclusiva de diretórios, podemos utilizar o comando `rmdir`. Mas lembre-se de que esse comando somente exclui diretórios vazios. Desta forma, na maioria das vezes é melhor utilizar `rm -r` para excluir diretórios que contenham arquivos ou outros diretórios (tenha certeza disso, evitando a exclusão indesejada de um ou mais diretórios). Sua sintaxe é:

```
Rmdir <opções> diretórios
```

A principal opção desse comando é `-p` ou `--parents`, a qual permite que uma árvore de diretórios (todos vazios) seja excluída. Veja um exemplo:

```
# rm teste/teste1/teste2 teste/teste1 teste
```

Utilizando a opção `-p` teremos:

```
# rm -p teste/teste1/teste2
```

3.1.2.5 - Exibição da Lista de Arquivos e Diretórios

Para exibir a lista de arquivos e diretórios, temos o comando `ls`, que em sua forma mais simples lista de forma sucinta o conteúdo do diretório atual. Sua sintaxe é:

```
Ls <opções> <lista de arquivos>
```

A lista de arquivos é opcional e em geral contém, quando utilizada, uma máscara para filtragem, utilizando os caracteres `*` e `?`, dos arquivos que serão exibidos. Veja um exemplo simples:

```
# ls teste*
```

Teremos como resultado uma lista simples dos arquivos e diretórios que satisfazem o critério de filtragem.

Esse comando possui uma lista enorme de opções. Mostramos as principais e mais utilizadas:

- d Lista os nomes dos diretórios e não seus conteúdos.
- f Não faz nenhuma classificação. Além disso, exibe arquivos normalmente escondidos pelo comando, tais como arquivos começados com ponto (.)
- h Exibe as informações sobre tamanhos dos arquivos em formato humano, ou seja, 1K, 2M, 5G.
- l Exibe uma lista longa, isto é, com mais informações sobre os arquivos
- R Exibe a lista em modo recursivo, isto é, lista os subdiretórios de cada diretório.
- t Exibe a lista classificada por ordem da data de modificação dos arquivos

As opções podem ser combinadas livremente, veja um exemplo:

```
# ls -lth /usr
```

```
[root@centOS ~]# ls -lth /usr
total 260K
drwxr-xr-x  2 root root  68K Set 7 20:12 bin
drwxr-xr-x 227 root root 12K Set 7 20:12 share
drwxr-xr-x 130 root root 68K Set 7 20:12 lib
drwxr-xr-x  2 root root 20K Set 7 20:12 sbin
drwxr-xr-x 14 root root 4,0K Set 7 20:12 libexec
drwxr-xr-x 11 root root 4,0K Set 7 20:12 include
drwxr-xr-x  6 root root 4,0K Set 7 19:59 kerberos
drwxr-xr-x  3 root root 4,0K Set 7 19:58 X11R6
lrwxrwxrwx  1 root root 10 Set 7 19:56 tmp -> ../../var/tmp
drwxr-xr-x 11 root root 4,0K Set 7 19:56 local
drwxr-xr-x  2 root root 4,0K Mar 9 2009 etc
drwxr-xr-x  2 root root 4,0K Mar 9 2009 games
drwxr-xr-x  2 root root 4,0K Mar 9 2009 src
[root@centOS ~]#
```

Figura 3.1

Outro comando interessante, que exibe a árvore de diretórios, é o tree. Veja um exemplo de sua utilização:

```
# tree -s
```

```
[root@centOS ~]# tree -s
.
|-- [ 4096] Desktop
|-- [ 1564] anaconda-ks.cfg
|-- [ 4096] help
|-- [ 36358] install.log
|-- [ 0] install.log.syslog
|-- [ 195] scsrn.log
|-- [ 4096] teste000
|-- [ 4096] teste001
|   '-- [ 4096] teste12
|       '-- [ 4096] teste13
`-- [ 4096] teste002
    '-- [ 4096] teste2
        '-- [ 4096] teste3

9 directories, 4 files
[root@centOS ~]#
```

Figura 3.2

O comando tree possui várias opções. As mais importantes são:

- d Lista apenas diretórios.
- s Exibe o tamanho de cada arquivo antes de seu nome.
- H Permite que a lista seja gerada em formato HTML, o que é útil para sites ftp, por exemplo.
- o Define o nome do arquivo de saída, ou seja, em vez de o resultado ser exibido na tela, é enviado ao arquivo informado no parâmetro.

3.2 - Gerenciamento de Grupos

Antes de criar outros usuários, podemos definir os grupos de usuários que desejamos utilizar. Esses grupos são opcionais, e devemos utilizá-los sempre que se faz necessária a sua organização, em geral quando temos um número grande de usuários e precisamos agrupá-los em categorias. Por padrão, as distribuições Linux trazem alguns grupos previamente cadastrados, os quais podem ser utilizados para criação de usuários administrativos do Linux.

Com os grupos podemos definir permissões de acesso específicas para os usuários que fazem parte do grupo, desta forma fica mais simples o gerenciamento, tanto na criação de usuários como na manutenção dos existentes. Além disso, ao instalarmos novos pacotes ou criarmos diretórios, fica fácil gerenciar as permissões de acesso, uma vez que precisamos apenas nos preocupar com os grupos e, se necessário, com as exceções (usuários de um grupo que necessitam de permissões diferenciadas).

3.2.1 - Adição de Grupos

Em geral temos uma lista de grupos existentes no momento da instalação do Linux, a qual pode ser vista no arquivo /etc/group. Observe uma lista típica na Figura 3.3.

```
[root@centOS ~]# cat /etc/group
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon
tty:x:5:
disk:x:6:root
lp:x:7:daemon,lp
mem:x:8:
kmem:x:9:
wheel:x:10:root
mail:x:12:mail
news:x:13:news
uucp:x:14:uucp
man:x:15:
games:x:20:
gopher:x:30:
dip:x:40:
ftp:x:50:
lock:x:54:
nobody:x:99:
users:x:100:
utmp:x:22:
utempter:x:35:
audio:x:63:gdm
distcache:x:94:
nscd:x:28:
floppy:x:19:
apache:x:48:
mailnull:x:47:
smmsp:x:51:
rpc:x:32:
ais:x:39:
piranha:x:60:
webalizer:x:67:
dovecot:x:97:
squid:x:23:
pcap:x:77:
slocate:x:21:
ntp:x:38:
luci:x:101:
ecryptfs:x:102:
dbus:x:81:
avahi:x:70:
rpcuser:x:29:
nfsnobody:x:65534:
named:x:25:
mysql:x:96:
sshd:x:74:
pegasus:x:65:
ricci:x:103:
haldaemon:x:68:
avahi-autoipd:x:104:
xfs:x:43:
gdm:x:42:
sabayon:x:86:
[root@centOS ~]#
```

Figura 3.3

Para adicionar um novo grupo ao sistema, utilizamos o comando groupadd, cuja sintaxe é:

Groupadd <opções> nome_do_grupo

As opções disponíveis são:

- r Cria um grupo do sistema, ou seja, com código entre 0 e 999.
- g Define o código do grupo. Se essa opção não for informada, o sistema atribui o código automaticamente (próximo código disponível, maior que qualquer código existente e maior que 500, a menos que -r seja informado).
- o Permite a criação de grupo com o código duplicado.

Alguns exemplos:

```
# groupadd contábil  
# groupadd -r diretoria  
# groupadd -g 5050 ti  
# groupadd infra
```

O resultado será (olhando o final do arquivo /etc/group):

```
contabil:x:500:  
diretoria:x:105:  
ti:x:5050:  
infra:x:5051:
```

Figura 3.4

3.2.2 - Remoção de Grupos

Para remover um grupo, utilizamos o comando groupdel seguido do nome do grupo que desejamos remover.

Antes de remover um grupo, é preciso verificar se não existem usuários ligados a ele; caso contrário, ficarão órfãos.

Não há opções nesse comando. Sua sintaxe é:

Groupdel nome_do_grupo

Exemplo:

```
# groupdel infra
```

3.3 - Gerenciamento de Usuários

O gerenciamento de usuários é uma das tarefas mais executadas pelo administrador Linux, uma vez que esta é uma lista dinâmica e exige atualização constante, seja para inclusão de novos usuários, retirada de outros e alteração de permissões e grupos dos usuários existentes.

Este tópico aborda:

- » Inclusão de um novo usuário
- » Gerenciamento da senha do usuário
- » Alteração de um usuário existente
- » Remoção de um usuário

A lista de usuários pode ser vista nos arquivos /etc/passwd e /etc/shadow.

3.3.1 - Adição de um Novo Usuário

A inclusão de um novo usuário é realizada pelo comando useradd do Linux, o qual exige algumas informações e dispõe de uma série de opções extras que complementam a criação de um usuário no sistema. A sua sintaxe é:

Useradd <opções> <nome_do_usuário>

O nome do usuário deve ser único no sistema, e caso haja tentativa de inserir um usuário existente no sistema, uma mensagem de erro é retornada e o processo, cancelado.

A forma mais simples de utilização do comando é useradd <nome_do_usuário>. Este formato cria o usuário e utiliza os valores padrão do sistema para definição dos parâmetros necessários a essa atividade. Para alterar um ou mais parâmetros, devemos utilizar as opções do comando. A lista de opções disponíveis é:

- c Qualquer texto explicativo, sendo utilizado comumente para informar o nome completo do usuário.
- b Diretório base para criação do diretório do usuário. Caso esse parâmetro não seja informado, o diretório padrão será utilizado (em geral /home). O nome do usuário será anexado ao final do diretório base informado. Este deve existir a menos que a opção -m seja utilizada.
- d Define o diretório do usuário que está sendo criado. Por padrão utiliza-se o diretório base (veja a opção -b) juntamente com o nome do usuário. Essa opção permite alterar esse diretório, podendo ser definido qualquer um desejado. O diretório informado não precisa existir no momento da criação do usuário, mas caso não exista, ele não será criado pelo comando.
- e Informa a data de expiração da conta do usuário que está sendo criado. A data deve estar no formato aaaa-mm-dd.

- f Número de dias após a expiração da senha do usuário em que a conta será desativada. Zero indica desativação imediata e -1 desabilita essa opção, sendo este o valor padrão do comando.
- g O nome ou número do grupo ao qual o usuário pertence.
- G Lista de grupos complementares do usuário. Os grupos devem ser separados por vírgula.
- M O diretório do usuário não será criado.
- m O diretório do usuário será criado, caso não exista.
- n Cria um grupo com o mesmo nome do usuário que está sendo criado pelo comando.
- o Permite a criação de usuário com um código duplicado (nome duplicado não é permitido em nenhuma hipótese).

Exemplos:

```
# useradd walace  
# useradd -d /usuarios/geral -g ti walace2  
# useradd -n walace3
```

3.3.2 - Gerenciamento de Senha de um Usuário

Para gerenciar a senha de um usuário, utilizamos o comando `passwd`, com o qual podemos definir a senha do usuário, alterá-la e ainda gerenciar os seus parâmetros, tais como data de expiração, número mínimo e máximo de dias para troca da senha pelo usuário e outras configurações necessárias ao gerenciamento de senhas. O usuário já deve ter sido criado anteriormente para que o comando `passwd` seja utilizado (veja o tópico anterior sobre como criar um usuário no sistema).

A sintaxe do comando `passwd` é:

```
Passwd <opções> <nome_do_usuário>
```

Temos duas utilizações básicas para o comando `passwd`. A primeira é a definição ou troca da senha de um usuário; a segunda é o gerenciamento da senha, definindo validade, intervalos de trocas e outras configurações úteis.

Para definição ou troca da senha, utilizamos o formato mais simples do comando, ou seja, sem nenhuma opção extra, por exemplo:

```
[root@centOS ~]# passwd walace  
Changing password for user walace.  
New UNIX password:  
Retype new UNIX password:  
passwd: all authentication tokens updated successfully.  
[root@centOS ~]#
```

Figura 3.5

Caso a senha seja fraca, uma mensagem de alerta é exibida, porém o processo não é interrompido. Se for desejado, a senha, mesmo fraca, pode ser atribuída ao usuário.

Para um gerenciamento completo, temos as seguintes opções (principais):

- d Desabilita a senha de um usuário, ou seja, a senha não será solicitada no processo de login.
- n Define o número mínimo de dias que uma senha é válida.
- x Define o número máximo de dias de uma senha. Após esse período a senha deve ser obrigatoriamente trocada pelo usuário.
- w Indica o número de dias, antes que a senha expire, para avisar o usuário da necessidade de troca da senha.
- S Exibe de forma resumida as informações de senha do usuário.

Veja um exemplo no qual definimos que a senha terá validade mínima de sete dias e máxima de 180 dias, e o usuário será avisado dez antes de a senha expirar:

```
# passwd -n 7 -x 180 -w 10 walace
```

3.3.3 - Alteração de um Usuário Cadastrado

A alteração dos dados de um usuário cadastrado no sistema é efetuada pelo comando usermod, cuja sintaxe é:

```
Usermod <opções> <nome_do_usuário>
```

As opções disponíveis são basicamente as mesmas existentes para o comando useradd, adicionando as seguintes opções:

- a Acrescenta os grupos informados aos existentes, devendo ser utilizada juntamente com a opção -G (definição de grupos extras).
- l Define um novo nome de usuário (login do usuário). Nada mais é alterado, apenas o nome do usuário. Se necessário, um novo diretório deve ser definido para esse usuário, para isso veja as opções -d (a opção -b não está disponível nesse comando).
- L Bloqueia o acesso da conta, acrescentando o símbolo ! ao início da senha do usuário.
- U Libera o bloqueio da conta, removendo-o ! colocado no início da senha do usuário.

Veja alguns exemplos:

```
# usermod -L walace
```

```
# usermod -l walace_1 -d /home/walace_1 walace
```

```
# usermod -U walace_1
```

3.3.4 - Remoção de um Usuário Cadastrado

Para remover um usuário do sistema, utilizamos o comando userdel. Sua sintaxe é:

```
Userdel <opções> <nome_do_usuario>
```

Em geral não é necessário definir nenhuma opção, mas precisamos saber que:

- » Um usuário ativo (logado) no sistema não será removido, a não ser que a opção `-f` seja utilizada.
- » O diretório do usuário não será removido a não ser que a opção `-r` seja utilizada.

As opções disponíveis para esse comando são:

- `-f` Força a remoção do usuário, mesmo que ele esteja ativo no sistema (logado), mas isso é perigoso e pode deixar o sistema instável.
- `-r` Remove o diretório do usuário e todos os seus arquivos, mas não remove outros arquivos ligados ao usuário, o que deve ser feito manualmente.

3.4 - Gerenciamento de Programas

Uma das tarefas rotineiras de um administrador de sistemas é o gerenciamento de programas, seja do sistema operacional, seja de terceiros. As principais rotinas no gerenciamento são:

- » Instalação de programas
- » Atualização de versão
- » Remoção de programas

O gerenciamento pode ser feito por meio dos seguintes métodos:

- » Utilitário rpm
- » Utilitário apt
- » Instalador do próprio programa
- » Cópia simples dos arquivos

3.4.1 - RPM

RPM é um poderoso utilitário para gerenciamento de pacotes, principalmente os pacotes ligados direta e indiretamente ao sistema operacional (nem todas as distribuições Linux tem suporte a RPM), facilitando a instalação, atualização e remoção de pacotes de programas.

As quatro funções básicas do utilitário são:

- » Instalação
- » Atualização
- » Remoção
- » Consulta

Cada uma das opções exige um formato próprio na execução do utilitário (cada um explicado no tópico correspondente, com exceção de consulta que será mostrado a seguir).

A sintaxe geral do programa é:

```
Rpm <modo> <opções> <pacote>
```

Em que modo pode ser:

- i ou --install Instalação de um novo pacote.
- U ou --upgrade Atualização de um pacote já instalado.
- e ou --erase Remoção de um pacote.
- q ou --query Consulta de pacotes instalados.

Cada um dos modos tem sua lista de opções. Para o modo de consulta temos as seguintes opções disponíveis (principais):

- l Para listar todos os arquivos do pacote.
- a Consulta a todos os pacotes.
- f Consulta aos pacotes que contêm o arquivo especificado.
- whatrequires Consulta aos pacotes que contêm a dependência informada.
- whatprovides Consulta aos pacotes que fornecem uma dependência.

A execução de rpm -q -a lista todos os pacotes instalados no sistema.

Veja um outro exemplo:

```
[root@centos ~]# rpm -q -l openssh-4.3p2-36.el5
/etc/ssh
/etc/ssh/moduli
/usr/bin/ssh-keygen
/usr/libexec/openssh
/usr/libexec/openssh/ssh-keysign
/usr/share/doc/openssh-4.3p2
/usr/share/doc/openssh-4.3p2/CREDITS
/usr/share/doc/openssh-4.3p2/ChangeLog
/usr/share/doc/openssh-4.3p2/INSTALL
/usr/share/doc/openssh-4.3p2/LICENCE
/usr/share/doc/openssh-4.3p2/OVERVIEW
/usr/share/doc/openssh-4.3p2/README
/usr/share/doc/openssh-4.3p2/README.dns
/usr/share/doc/openssh-4.3p2/README.nss
/usr/share/doc/openssh-4.3p2/README.platform
/usr/share/doc/openssh-4.3p2/README.privsep
/usr/share/doc/openssh-4.3p2/README.smartcard
/usr/share/doc/openssh-4.3p2/README.tun
/usr/share/doc/openssh-4.3p2/rfc.nroff
/usr/share/doc/openssh-4.3p2/TODD
/usr/share/doc/openssh-4.3p2/WARNING.RNG
/usr/share/man/man1/ssh-keygen.1.gz
/usr/share/man/man8/ssh-keysign.8.gz
```

Figura 3.6

3.4.2 - Apt

O apt é um pacote com ferramentas para gerenciamento de pacotes em uma distribuição Linux, mas nem todas as distribuições possuem esse pacote. Em geral, quem usa RPM não trabalha com apt e vice-versa. Os principais programas do pacote são:

- » Apt-get
- » Apt-cdrom
- » Apt-cache
- » Apt-config

Este tópico aborda o apt-get, responsável pela instalação, atualização e remoção de programas.

A sintaxe do comando apt-get é:

```
Apt-get [modo] [opções] [programa]
```

Os modos disponíveis (principais) são:

- » **Install:** instala um novo pacote.
- » **Upgrade:** atualiza um pacote instalado.
- » **Remove:** remove um pacote.
- » **Source:** baixa os códigos-fonte de um pacote.
- » **Clean:** apaga os arquivos de instalação.

As opções disponíveis são:

- d Apenas baixa e não instala.
- y Assume a resposta “sim” para qualquer pergunta do instalador.

O programa apt-get utiliza um repositório apt para localização das informações sobre os pacotes que serão instalados. O arquivo que contém a lista é o /etc/apt/sources.list e pode ser editado para inclusão, alteração e remoção dos endereços dos repositórios nos quais o apt vai buscar os pacotes.

3.4.3 - Instalador Próprio

Alguns programas, feitos por terceiros, possuem seus próprios instaladores e seu comportamento e utilização dependem de como foram construídos. Em sua maioria disponibilizam um script para gerenciamento da aplicação que cuida dos detalhes da instalação e remoção do pacote.

A instalação pode ser feita de forma direta, ou seja, os programas já vêm no formato binário e precisam apenas ser instalados e configurados no sistema operacional, ou podem vir com os códigos-fonte e exigirem que o pacote seja primeiramente compilado para depois ser instalado.

Um exemplo é o servidor web Apache, o qual é disponibilizado com os códigos-fonte e exige a execução da configuração, compilação e instalação. Neste caso utilizamos o configurador que vem com o aplicativo (configure) e compilamos e instalamos o pacote por meio do comando make.

Esse formato é mais flexível, todavia exige algum conhecimento, principalmente quando ocorrem erros na geração do código binário.

3.4.4 - Cópia

Este é o formato mais simples de instalação, sendo necessário apenas copiar os arquivos dentro de um diretório e pronto, já é possível utilizar o programa. A cópia pode ser direta ou então é preciso primeiramente executar o programa tar ou gunzip (ou ambos).

Neste caso os programas já estão em formato binário e prontos para serem utilizados, exigindo pouco ou nenhum trabalho extra.

3.4.5 - Instalação de Pacotes

A instalação por cópia ou com uso de instaladores próprios não será tratada no livro, uma vez que o primeiro é muito simples e não necessita de maiores explicações, e o segundo é específico de cada pacote, dependendo de consulta ao guia de instalação fornecido com o pacote em questão.

Trataremos da instalação por rpm ou apt.

3.4.5.1 - Instalação por RPM

Para instalação de um pacote por meio do utilitário RPM, devemos utilizar o modo -i ou --install. Neste caso teremos as seguintes opções disponíveis:

- allfiles Instalar todos os arquivos do pacote, mesmo arquivos de configuração que poderiam ser ignorados.
- excludedocs Não instala documentação.
- fileconflictcs Detecta conflitos entre pacotes.
- force Instala mesmo que haja problema de conflitos e dependências.
- F Atualiza o pacote se já estiver instalado.
- h Exibe o símbolo # à medida que o pacote é instalado, geralmente utilizado com -v.
- ignorearch Ignora a arquitetura do pacote.
- ignoreos Ignora o sistema operacional do pacote.
- ignoresize Não verifica o espaço em disco antes de instalar.

--nodeps	Não verifica as dependências do pacote.
--nomd5	Não verifica o md5 dos arquivos.
--nosuggest	Não sugere como resolver problemas de dependência.
--noscripts	Não executa scripts de instalação contidos no pacote.
--percent	Exibe o percentual instalado durante o processo.
--test	Não instala, apenas informa se seria executada com sucesso.
-v	Exibe detalhes da instalação.

De modo geral utilizamos a forma mais simples do utilitário, por exemplo:

```
# rpm -i -v -h postgresql-8.4.5-1.fc15.i686.rpm
```

3.4.5.2 - Instalação por Apt

A instalação de um novo pacote por meio do apt-get é feita pelo modo `Install` e em geral não necessita de nenhuma opção extra.

Um exemplo é:

```
# apt-get install php  
# apt-get install -y mysql
```

3.4.6 - Atualização de Pacotes

A atualização é uma rotina comum, pois sempre temos algum pacote sendo atualizado, quer seja para melhorias quer seja para correção de erros (bugs). Trataremos da atualização por RPM ou apt-get.

Antes de atualizar um pacote, precisamos tomar alguns cuidados, principalmente se houver dependências, uma vez que a atualização pode afetar o comportamento dos programas dependentes deste ou mesmo fazê-los parar de funcionar.

3.4.6.1 - Atualização por RPM

Para atualizar um pacote pelo RPM, devemos utilizar o modo `-U` ou `--UPGRADE`, juntamente com as opções necessárias, que são as seguintes:

--excludedocs	Não instalar documentação.
--oldpackage	Permite a instalação de uma versão anterior a que já está instalada (<code> downgrade</code>).
--force	Instala mesmo que haja problema de conflitos e dependências.

-h	Exibe o símbolo # à medida que o pacote é instalado, geralmente utilizado com -v.
--ignorearch	Ignora a arquitetura do pacote.
--ignoreos	Ignora o sistema operacional do pacote.
--nodeps	Não verifica as dependências do pacote.
--noscripts	Não executar scripts de instalação contidos no pacote.
--percent	Exibir o percentual instalado durante o processo.
--test	Não instala, apenas informa se seria executada com sucesso.
-v	Exibe detalhes da instalação.

Um exemplo:

```
# RPM -U -v -h php-5.3.3-2.fc15.i686.rpm
```

3.4.6.2 - Atualização por Apt

Para atualizar um pacote utilizando o apt-get, devemos usar o modo UPGRADE disponível na ferramenta e em geral não é necessária nenhuma opção extra. Exemplos:

```
# apt-get upgrade php
# apt-get upgrade -y postgresql
```

É possível obter uma nova lista de pacotes e novas versões pela opção UPDATE do apt-get.

```
# apt-get update -u mysql
```

3.4.7 - Remoção de Pacotes

A remoção de pacotes é sempre possível, mas devemos redobrar os cuidados nessa ação, uma vez que podemos comprometer seriamente o sistema, podendo inclusive fazê-lo parar de funcionar ou ficar extremamente instável.

3.4.7.1 - Remoção por RPM

Para remover um pacote usando o RPM, utilizamos o modo -e ou --erase. Temos as seguintes opções para esse modo:

--nodeps	Não verifica as dependências do pacote.
--noscripts	Não executa scripts de remoção contidos no pacote.
--test	Não remove, apenas informa se seria executada com sucesso.
-vv	Exibe detalhes da instalação.

Todos os traços do pacote no sistema são removidos, a não ser que exista alguma dependência, o que pode impedir a remoção (veja a opção `--nodeps`). Os arquivos do pacote são removidos caso pertençam somente ao pacote em questão; caso contrário, são mantidos no sistema. Exemplo:

```
# RPM -erase -vv php-5.3.3-2.fc15.i686.rpm
```

3.4.7.2 - Remoção por Apt

A remoção pelo apt-get é feita utilizando o modo REMOVE e em geral não é necessária nenhuma opção extra. Exemplo:

```
# apt-get remove php
```

```
# apt-get remove mysql
```

3.5 - Monitoramento do Sistema

Monitorar a saúde do servidor é uma tarefa fundamental para o administrador Linux e sempre é preciso estar atento aos seguintes itens do sistema:

- » CPU
- » Disco
- » Memória
- » Rede

Cada um destes itens deve ser monitorado frequentemente pelo administrador, pois influencia diretamente na performance e na estabilidade do sistema operacional. A seguir, mostramos as ferramentas disponíveis para o gerenciamento de cada um dos itens. Algumas vezes a mesma ferramenta pode ser utilizada em mais de um item.

3.5.1 - CPU

Pelo monitoramento da CPU podemos verificar possíveis gargalos no sistema, bem como saber quais processos gastam mais tempo e recursos do sistema, o que ajuda em um plano para equalizar a performance, seja investindo em um novo servidor ou simplesmente alterando o comportamento do sistema, por exemplo, alterando o horário de processamento de um ou mais aplicativos para liberar tempo e recurso de CPU para processos mais críticos.

O primeiro passo é usar o utilitário top que mostra em tempo real os processos que estão rodando e os recursos que estão sendo consumidos, inclusive tempo de processamento e alocação de CPU.

O padrão é exibir primeiramente as tarefas que mais consumem CPU. Um exemplo de sua execução é:

```
# top
```

top - 19:20:46 up 3 min, 2 users, load average: 1.06, 0.70, 0.28											
Tasks: 135 total, 2 running, 133 sleeping, 0 stopped, 0 zombie											
Cpu(s): 0.0%us, 0.3%sy, 0.0%ni, 99.7%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st											
Mem: 1471992k total, 523652k used, 948340k free, 45328k buffers											
Swap: 1146872k total, 0k used, 1146872k free, 324152k cached											
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2922	root	15	0	2332	1036	796	R	0.3	0.1	0:00.02	top
1	root	15	0	2068	612	528	S	0.0	0.0	0:00.54	init
2	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	migration/0
3	root	34	19	0	0	0	S	0.0	0.0	0:00.01	ksoftirqd/0
4	root	RT	-5	0	0	0	S	0.0	0.0	0:00.00	watchdog/0
5	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	events/0
6	root	10	-5	0	0	0	S	0.0	0.0	0:00.02	khelper
7	root	10	-5	0	0	0	S	0.0	0.0	0:00.01	kthread
10	root	10	-5	0	0	0	S	0.0	0.0	0:00.27	kblockd/0
11	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	kacpid
47	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	cqueue/0
50	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	khubd
52	root	10	-5	0	0	0	S	0.0	0.0	0:00.00	kseriod
111	root	17	0	0	0	0	S	0.0	0.0	0:00.00	pdfflush
112	root	15	0	0	0	0	S	0.0	0.0	0:00.00	pdfflush
113	root	12	-5	0	0	0	S	0.0	0.0	0:00.00	kswapd0
114	root	12	-5	0	0	0	S	0.0	0.0	0:00.00	aio/0
267	root	11	-5	0	0	0	S	0.0	0.0	0:00.00	kpsmoused
296	root	16	-5	0	0	0	S	0.0	0.0	0:00.00	ata/0
297	root	16	-5	0	0	0	S	0.0	0.0	0:00.00	ata_aux
302	root	18	-5	0	0	0	S	0.0	0.0	0:00.00	kstriped
311	root	20	-5	0	0	0	S	0.0	0.0	0:00.00	ksnapd

Figura 3.7

Por meio deste utilitário obtemos informações importantes, tais como a taxa de ocupação da CPU, quanto cada processo está requisitando de CPU e outras informações importantes.

Outro utilitário importante no gerenciamento de CPU é o vmstat, que exibe informações diversas, entre elas o consumo de CPU. Temos a opção de exibição do resultado de várias formas. Acompanhe alguns exemplos:

```
[root@centOS ~]# vmstat 3
procs      memory          swap      io      system      cpu
 r b  swpd   free  buff  cache  si  so  bi  bo  in  cs us sy id wa st
 2 0    0 948228 45332 324256  0  0  441  42 1027 266  2  5 87  6  0
 0 0    0 948104 45332 324256  0  0  0  4 1017 201  0  0 100  0  0
 0 0    0 948104 45332 324256  0  0  0  9 1007 181  0  1 99  0  0
 0 0    0 948104 45332 324256  0  0  0  0 1015 188  0  0 100  0  0
 0 0    0 948104 45332 324256  0  0  0  0 1008 178  0  0 100  0  0
```

Figura 3.8

Este exemplo exibe os dados em uma única linha, e cada linha mostra a situação do sistema três segundos após a linha anterior (o parâmetro 3 indica ao utilitário que deve exibir uma nova estatística a cada três segundos).

```
[root@centOS ~]# vmstat -s
      1471992 total memory
      523764 used memory
      200728 active memory
      287428 inactive memory
      948228 free memory
      45332 buffer memory
      324256 swap cache
     1146872 total swap
          0 used swap
     1146872 free swap
     1606 non-nice user cpu ticks
       13 nice user cpu ticks
     3788 system cpu ticks
     87009 idle cpu ticks
      5267 IO-wait cpu ticks
       330 IRQ cpu ticks
        45 softirq cpu ticks
          0 stolen cpu ticks
    369497 pages paged in
    35029 pages paged out
          0 pages swapped in
          0 pages swapped out
  1005069 interrupts
  249832 CPU context switches
1287436656 boot time
     2966 forks
```

Figura 3.9

Neste segundo exemplo os dados são exibidos de forma mais completa e uma informação por linha (o parâmetro `-s` é utilizado para exibir um sumário).

Outro utilitário que podemos utilizar para busca de informações da CPU e de outros itens do sistema operacional é o `ps`, que mostra basicamente os processos executados no momento, mas pode, por meio de suas várias opções, oferecer uma gama enorme de informações. Por exemplo, para ver todos os processos sendo executados, utilizamos a opção `-A`:

```
# ps -A
# ps -A
```

Sample Outputs:

```
PID TTY      TIME CMD
 1 ?        00:00:02 init
 2 ?        00:00:02 migration/0
 3 ?        00:00:01 ksoftirqd/0
 4 ?        00:00:00 watchdog/0
 5 ?        00:00:00 migration/1
 6 ?        00:00:15 ksoftirqd/1
...
...
4881 ?        00:53:28 java
4885 tty1      00:00:00 mingetty
4886 tty2      00:00:00 mingetty
4887 tty3      00:00:00 mingetty
4888 tty4      00:00:00 mingetty
4891 tty5      00:00:00 mingetty
4892 tty6      00:00:00 mingetty
4893 ttys1     00:00:00 agetty
12853 ?       00:00:00 cifsoplockd
12854 ?       00:00:00 cifsnotifyd
14231 ?       00:10:34 lighttpd
```

```
14232 ? 00:00:00 php-cgi
54981 pts/0 00:00:00 vim
55465 ? 00:00:00 php-cgi
55546 ? 00:00:00 bind9-snmp-stat
55704 pts/1 00:00:00 ps
```

Para obter mais informações, podemos utilizar as opções –Alf.

Para obter a árvore de processos, são usadas as opções –ejH ou axjf ou ainda o utilitário pstree. Veja um exemplo de sua execução:

```
[root@centOS ~]# pstree
init─┬─/usr/bin/sealer
      └─acpid
      ├─anacron
      ├─atd
      ├─auditd─┬─audispd─{audispd}
                 └─{auditd}
      ├─automount─4─{automount}
      ├─avahi-daemon─avahi-daemon
      ├─bonobo-activat─{bonobo-activati}
      ├─brcm_iscsiuio─3─{brcm_iscsiuio}
      ├─bt-applet
      ├─clock-applet
      ├─crond
      ├─cupsd
      ├─2─{dbus-daemon─{dbus-daemon}}
      ├─dbus-launch
      ├─dhclient
      ├─dnsmasq
      ├─eggcups
      ├─escd─{escd}
      ├─events/0
      ├─gdm_server
      ├─gconfd-2
      ├─gdm-binary─gdm-binary─Xorg
                     └─gnome-session─ssh-agent
      ├─gdm-rh-security─{gdm-rh-security}
      ├─gnome-keyring-d
      └─gnome-panel
```

Figura 3.10

Detalhes sobre a CPU do servidor podem ser encontrados no arquivo /proc/cpuinfo. Acompanhe um exemplo:

```
[root@centOS ~]# cat /proc/cpuinfo
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 15
model name : Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz
stepping : 11
cpu MHz : 2399.437
cache size : 4096 KB
fdt_bug : no
hlt_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 2
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic mtrr pge mca cmov pat pse36 clflush mmx
        fxsr sse sse2 constant_tsc up pnpi monitor
bogomips : 4798.87
```

Figura 3.11

Caso você deseje, pode usar (se a interface gráfica estiver disponível) o utilitário gráfico do GNOME para monitorar o sistema.

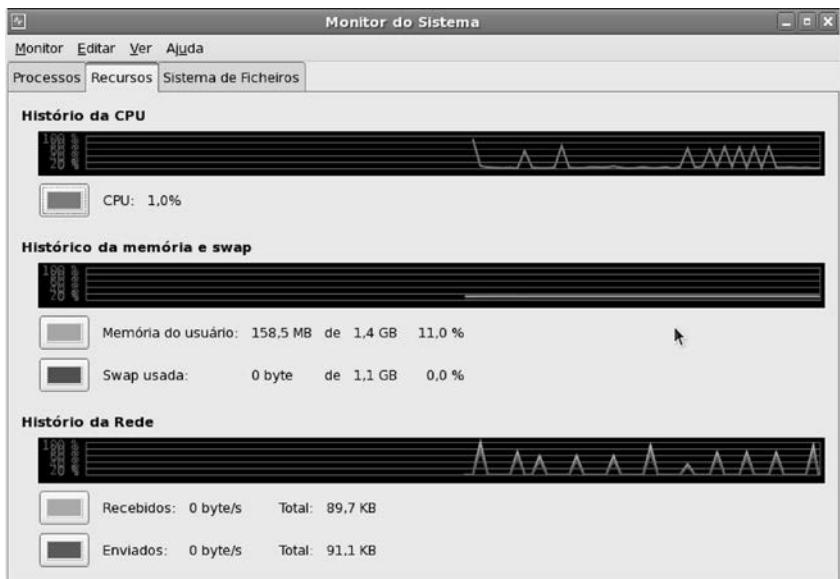


Figura 3.12

Podemos ver neste utilitário tanto a estatística de utilização quanto os processos que estão sendo executados.

3.5.2 - Memória

Além dos utilitários top e vmstat, que servem também para monitorar a memória do servidor, existem ainda outros utilitários específicos.

O primeiro dos utilitários é o free, que tem como função exibir a memória total, utilizada e disponível do sistema, além de informações extras sobre cache e swap.

```
[root@centOS ~]# free
              total        used        free      shared  buffers  cached
Mem:       1471992      553588      918404          0     48104   341700
-/+ buffers/cache:    163784     1308208
Swap:        1146872          0     1146872
```

Figura 3.13

Utilizando a opção `-s <tempo>`, podemos exibir a estatística de forma contínua, a cada `<tempo>` segundos.

Para obter informações específicas de um dado processo, considera-se o utilitário pmap. Para saber qual o processo desejado, usa-se o utilitário os. Veja um exemplo de pmap:

```
[root@centOS ~]# pmap 2909
2909: bash
002c6000 36K r-x-- /lib/libnss_files-2.5.so
002cf000 4K r-x-- /lib/libnss_files-2.5.so
002d0000 4K rwx-- /lib/libnss_files-2.5.so
0079b000 104K r-x-- /lib/ld-2.5.so
007b5000 4K r-x-- /lib/ld-2.5.so
007b6000 4K rwx-- /lib/ld-2.5.so
007be000 1276K r-x-- /lib/libc-2.5.so
008fd000 8K r-x-- /lib/libc-2.5.so
008ff000 4K rwx-- /lib/libc-2.5.so
00900000 12K rwx-- [ anon ]
0092e000 8K r-x-- /lib/libdl-2.5.so
00930000 4K r-x-- /lib/libdl-2.5.so
00931000 4K rwx-- /lib/libdl-2.5.so
00da2000 4K r-x-- [ anon ]
03f01000 12K r-x-- /lib/libtermcap.so.2.0.8
03f04000 4K rwx-- /lib/libtermcap.so.2.0.8
08047000 696K r-x-- /bin/bash
080f5000 20K rw--- /bin/bash
080fa000 20K rw--- [ anon ]
08ad7000 132K rw--- [ anon ]
b7dba000 2048K r---- /usr/lib/locale/locale-archive
b7fba000 8K rw--- [ anon ]
b7fc5000 8K rw--- [ anon ]
b7fc7000 28K r-s- /usr/lib/gconv/gconv-modules.cache
bfddb000 84K rw--- [ stack ]
total 4536K
```

Figura 3.14

Para obter informações mais detalhadas, podemos usar as opções `-d` e `-x`.

```
[root@centOS ~]# pmap -d 2909
2909: bash
Address Kbytes Mode Offset Device Mapping
002c6000 36 r-x-- 0000000000000000 0fd:00000 libnss_files-2.5.so
002cf000 4 r-x-- 00000000000008000 0fd:00000 libnss_files-2.5.so
002d0000 4 rwx-- 00000000000009000 0fd:00000 libnss_files-2.5.so
0079b000 104 r-x-- 00000000000000000 0fd:00000 ld-2.5.so
007b5000 4 r-x-- 00000000000190000 0fd:00000 ld-2.5.so
007b6000 4 rwx-- 000000000001a0000 0fd:00000 ld-2.5.so
007be000 1276 r-x-- 00000000000000000 0fd:00000 libc-2.5.so
008fd000 8 r-x-- 0000000000013f000 0fd:00000 libc-2.5.so
008ff000 4 rwx-- 00000000000141000 0fd:00000 libc-2.5.so
00900000 12 rw--- 00000000000000000 000:00000 [ anon ]
0092e000 8 r-x-- 00000000000000000 0fd:00000 libdl-2.5.so
00930000 4 r-x-- 00000000000010000 0fd:00000 libdl-2.5.so
00931000 4 rwx-- 00000000000002000 0fd:00000 libdl-2.5.so
00da2000 4 r-x-- 00000000000d20000 000:00000 [ anon ]
03f01000 12 r-x-- 00000000000000000 0fd:00000 libtermcap.so.2.0.8
03f04000 4 rwx-- 00000000000002000 0fd:00000 libtermcap.so.2.0.8
08047000 696 r-x-- 00000000000000000 0fd:00000 bash
080f5000 20 rw--- 00000000000ae0000 0fd:00000 bash
080fa000 20 rw--- 0000000000807a000 000:00000 [ anon ]
08ad7000 132 rw--- 0000000008ad70000 000:00000 [ anon ]
b7dba000 2048 r---- 00000000000000000 0fd:00000 locale-archive
b7fba000 8 rw--- 00000000b7fba0000 000:00000 [ anon ]
b7fc5000 8 rw--- 00000000b7fc50000 000:00000 [ anon ]
b7fc7000 28 r-s- 00000000000000000 0fd:00000 gconv-modules.cache
bfddb000 84 rw--- 00000000bffa0000 000:00000 [ stack ]
mapped: 4536K writeable/private: 304K shared: 28K
```

Figura 3.15

Finalmente, temos o arquivo `/proc/meminfo` que contém informações extras sobre a memória do sistema.

```
[root@centOS ~]# cat /proc/meminfo
MemTotal:      1471992 kB
MemFree:       844764 kB
Buffers:        71900 kB
Cached:         383328 kB
SwapCached:      0 kB
Active:        220836 kB
Inactive:      356160 kB
HighTotal:     571328 kB
HighFree:       56620 kB
LowTotal:      900664 kB
LowFree:        788144 kB
SwapTotal:    1146872 kB
SwapFree:      1146872 kB
Dirty:          12 kB
Writeback:       0 kB
AnonPages:     121780 kB
Mapped:         42128 kB
Slab:          37952 kB
PageTables:     4228 kB
NFS_Unstable:      0 kB
Bounce:          0 kB
CommitLimit:   1882868 kB
Committed_AS:  519684 kB
VmallocTotal:   114680 kB
VmallocUsed:    5044 kB
VmallocChunk:  109504 kB
HugePages_Total: 0
HugePages_Free: 0
HugePages_Rsvd: 0
Hugepagesize:  4096 kB
```

Figura 3.16

3.5.3 - Disco

Para o gerenciamento do disco existem os utilitários fdisk, df e du. O primeiro exibe informações sobre os discos e suas partições, já o segundo mostra a ocupação atual do sistema e o último apresenta a ocupação de cada um dos arquivos do diretório atual e todos os seus subdiretórios de forma recursiva ou do sistema inteiro (caso estejamos no diretório raiz).

O utilitário fdisk tem como principal função o gerenciamento de partições, executando a criação, formatação e remoção de partições (não é o escopo deste tópico tratar deste assunto). Vamos ver agora a lista de partições disponíveis, usando a opção `-l`. O resultado será algo parecido com:

```
[root@centOS ~]# fdisk -l
Disk /dev/hda: 68.7 GB, 68719476736 bytes
255 heads, 63 sectors/track, 8354 cylinders
Units = cilindros of 16065 * 512 = 8225280 bytes

Dispositivo Boot      Start        End      Blocks   Id  System
/dev/hda1      *           1          13      104391   83  Linux
/dev/hda2            14        8354     66999082+   8e  Linux LVM
```

Figura 3.17

Veja que há informações sobre o tamanho do disco, os dispositivos e o tamanho e tipo de cada partição.

O utilitário df exibe a ocupação atual do disco, informando o tamanho de cada partição, sua ocupação, a disponibilidade de espaço, o percentual do disco utilizado e o ponto de montagem da partição.

```
[root@centos /]# df -h
Sist. Arq.           Tam Usad Disp Uso% Montado em
/dev/mapper/VolGroup01-LogVol00
                                616  3,5G  55G  7% /
/dev/hdal              99M   20M  75M 21% /boot
tmpfs                  719M     0  719M  0% /dev/shm
```

Figura 3.18

A opção –h permite uma leitura mais fácil e direta dos dados apresentados.

Por último temos o utilitário du, que exibe a ocupação de cada arquivo do diretório atual e de todos os seus diretórios (é recursivo). Para exibir a ocupação de todo o disco, devemos estar na raiz do sistema ao executar o comando. Há basicamente duas formas de ver o resultado. A primeira é o resultado detalhado com a exibição da ocupação de cada arquivo ou diretório, a segunda é a exibição apenas do resumo total. Na primeira forma, basta executar o comando sem nenhuma opção (ou apenas de exibição com a opção –h, que tem o mesmo efeito mostrado em df). A segunda forma, isto é, a exibição apenas do resumo final é conseguida com a opção –s. Veja um exemplo das duas opções:

```
[root@centos teste0002]# du -h
8,0K    ./teste2/teste3
16K    ./teste2
24K    .
[root@centos teste0002]# cd /
[root@centos /]# du -s -h
3,9G    .
```

Figura 3.19

3.5.4 - Rede

O gerenciamento da rede, seu tráfego, conexões, pontes, utilização de sockets e outras informações são fundamentais para que o administrador gerencie o sistema e consiga deixá-lo estável, funcional e otimizado.

Para essa tarefa existem alguns utilitários, tais como netstat, traceroute e tcpdump. Com exceção do último utilitário da lista (tcpdump), todos são de fácil utilização e entendimento.

O utilitário netstat exibe as conexões de rede, tabela de roteamento, estatísticas de cada interface e outras informações úteis. Veja um exemplo de sua utilização:

```
[root@centOS /]# netstat -i
Tabela de Interfaces do Kernel
Iface      MTU Met      RX-OK RX-ERR RX-DRP RX-OVR      TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0       1500 0        1568    0     0     0      1626    0     0     0     0 BMRU
lo        16436 0        2502    0     0     0      2502    0     0     0     0 LRU
virbr0     1500 0        0      0     0     0       65     0     0     0     0 BMRU
[root@centOS /]# netstat -rn
Tabela de Roteamento IP do Kernel
Destino      Roteador      MascaraGen.      Opções      MSS Janela irtt Iface
10.0.0.1     0.0.0.0      255.255.255.0 U          0 0      0 eth0
192.168.122.0 0.0.0.0   255.255.255.0 U          0 0      0 virbr0
169.254.0.0  0.0.0.0   255.255.0.0 U          0 0      0 eth0
0.0.0.0     10.0.2.2    0.0.0.0   UG         0 0      0 eth0
[root@centOS /]# netstat -a
Conexões Internet Ativas (servidores e estabelecidas)
Proto Recv-Q Send-Q Local Address           Foreign Address           State
tcp     0      0 centOS.walace.soares.c:2208  *:*
tcp     0      0 *:vce                   *:*
tcp     0      0 *:sunrpc                *:*
tcp     0      0 *:16851                 *:*
root@centOS /]# netstat -s
p:
 3934 total packets received
 0 forwarded
 0 incoming packets discarded
 3933 incoming packets delivered
 3935 requests sent out
cmp:
 1379 ICMP messages received
 0 input ICMP message failed.
 Histograma de entrada ICMP:
   destination unreachable: 1377
   timeout in transit: 2
 1 ICMP messages sent
 0 ICMP messages failed
 Histograma de saída ICMP:
   destination unreachable: 1
cmpMsg:
  InType3: 1377
  InType1: 2
  OutType3: 1
cp:
 122 active connections openings
 2 passive connection openings
 120 failed connection attempts
 0 connection resets received
 0 connections established
```

Figura 3.20

O utilitário ss exibe as estatísticas de sockets. Em geral utilizamos com a opção **-a**, que indica que todos os sockets devem ser exibidos, porém o utilitário dispõe de várias opções para filtragem e maior detalhamento das informações. Acompanhe um exemplo:

```
[root@centOS /]# ss -a
State      Recv-Q Send-Q          Local Address:Port          Peer Address:Port
LISTEN     0      0              127.0.0.1:2208          *
LISTEN     0      0              *:vce                  *
LISTEN     0      0              *:sunrpc               *
LISTEN     0      0              *:16851                *
LISTEN     0      0          192.168.122.1:domain  *
LISTEN     0      0              127.0.0.1:ipp            *
LISTEN     0      0              127.0.0.1:smtp           *
LISTEN     0      0              127.0.0.1:2207           *
LISTEN     0      0              *:863                  *
LISTEN     0      0              :::ssh                 *:  
:::
```

Figura 3.21

Já o utilitário tcpdump serve para monitorar o tráfego de rede, mas é necessário um conhecimento razoável do protocolo TCP/IP para fazer bom uso dessa ferramenta. Veja um exemplo de utilização:

```
[root@centOS /]# tcpdump -ni eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 96 bytes
21:22:16.231026 arp who-has 10.0.2.2 tell 10.0.2.15
21:22:16.231490 IP 10.0.2.15.56628 > 200.149.55.140.domain: 20529+[|domain]
21:22:16.232019 arp reply 10.0.2.2 is-at 52:54:00:12:35:02
21:22:16.232757 IP 10.0.2.2 > 10.0.2.15: ICMP net 200.149.55.140 unreachable, length 92
21:22:21.231070 IP 10.0.2.15.55359 > 200.165.132.147.domain: 20529+[|domain]
21:22:21.231457 IP 10.0.2.2 > 10.0.2.15: ICMP net 200.165.132.147 unreachable, length 92
21:22:26.230344 IP 10.0.2.15.40854 > 200.149.55.140.domain: 43271+ A? yum.singlehop.com. (35)
21:22:26.232387 IP 10.0.2.2 > 10.0.2.15: ICMP net 200.149.55.140 unreachable, length 71

8 packets captured
8 packets received by filter
0 packets dropped by kernel
```

Figura 3.22

3.6 - Arquivos de Mensagens e Logs do Sistema

O Linux possui vários arquivos. Além disso, grande parte dos pacotes disponibiliza seus próprios arquivos de mensagens e logs.

A maioria dos arquivos de mensagens e logs do sistema está no diretório /var/logs e pode ser vista por meio dos comandos cat, more ou tail.

Os arquivos de mensagens e logs mais comuns são:

/var/log/message	Mensagens gerais do sistema.
/var/log/auth.log	Log de autenticação de usuários.
/var/log/kern.log	Log do Kernel do Linux.
/var/log/cron.log	Log do crond (agendador de tarefas).
/var/log/mail.log	Log do servidor de correio (mail).
/var/log/boot.log	Log com informações de boot (inicialização) do sistema.
/var/log/wtmp	Registro de entrada de usuários ou wtmp.

Muitos aplicativos têm seu próprio diretório de mensagens e log. Por exemplo, o servidor web apache tem por padrão criar seu próprio diretório de logs e armazenar neles os arquivos com informações de acesso, erros e outros.

No modo gráfico podemos visualizar e gerenciar os arquivos de log pelo aplicativo intitulado gnome-system-log (pressupondo que você está usando o gnome como interface gráfica). Em uma única tela é possível monitorar os logs existentes, alterá-los, removê-los ou incluir novos arquivos de controle.

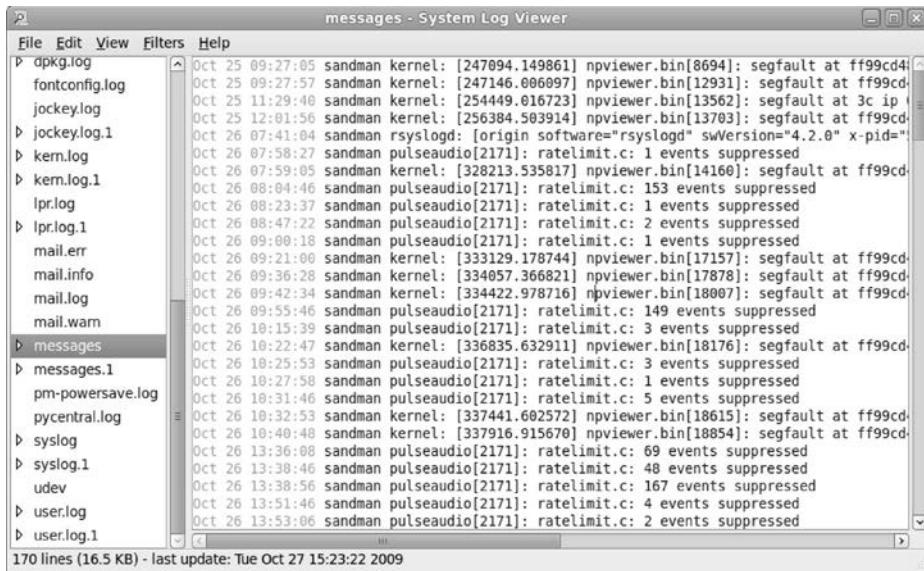


Figura 3.23

Para gerenciamento dos arquivos de log, principalmente limitação de tamanho (alguns arquivos crescem rapidamente), utilizamos o aplicativo logrotate. Podemos rodar o aplicativo manualmente ou inseri-lo no crontab para que rode diariamente. A configuração pode ser geral, por meio do arquivo /etc/logrotate.conf, ou específica para cada arquivo. Neste caso inserimos a configuração no diretório /etc/logrotate.conf, e o nome do arquivo de configuração é o nome do processo que desejamos gerenciar. Veja dois exemplos:

/etc/logrotate.conf:

```
# sample logrotate configuration file
compress
/var/log/messages {
    rotate 5
    weekly
    postrotate
        /usr/bin/killall -HUP syslogd
    endscript
}

"/var/log/httpd/access.log" "/var/log/httpd/error.log" {
    rotate 5
    mail www@my.org
    size 100k
    sharedscripts
    postrotate
        /usr/bin/killall -HUP httpd
    endscript
}

/var/log/news/news.crit {
    monthly
    rotate 2
    olddir /var/log/news/old
    missingok
```

```

postrotate
    kill -HUP `cat /var/run/inn.pid`
endscript
nocompress
}

```

Log específico de um processo qualquer:

```

/var/log/process-name.log {
    rotate 12
    monthly
    errors root@localhost
    missingok
    postrotate
        /usr/bin/killall -HUP process-name 2> /dev/null || true
    endscript
}

```

3.7 - Serviços (Daemons)

O Linux disponibiliza uma série de aplicativos executados em modo escondido (background), o que chamamos de serviços. Seja o servidor de impressão, o gerenciador de correio, o servidor web e muitas outras aplicações, todos são executados de forma transparente, atuando nos bastidores do sistema operacional.

Serviços são programas chamados “daemons”. Uma vez instalados e iniciados, são executados continuamente até que sejam interrompidos, quer seja manualmente ou por algum parâmetro de sua configuração. Um exemplo de daemon é o serviço do servidor web apache conhecido como httpd. Uma vez inicializado, fica observando (ouvindo) a porta 80 (ou outra porta, conforme configuração) e, quando existe atividade, realiza as operações solicitadas pelos programas clientes (em geral um browser HTML).

Para gerenciar os serviços no Linux existem os aplicativos chkconfig e service, ambos localizados na pasta /sbin.

O primeiro, chkconfig, configura quais serviços serão executados durante o processo de inicialização (boot) do Linux. Qualquer alteração processada por esse comando terá efeito somente no próximo processo de inicialização (boot) do servidor. Caso seja necessário ativar ou desativar um serviço manualmente e imediatamente, deve-se usar o utilitário service.

3.7.1 - Chkconfig

O utilitário chkconfig define os serviços que serão inicializados junto com o sistema operacional e em que nível ocorre a inicialização. Os níveis variam de 0 a 6:

0. Modo de desligamento (shutdown). Aqui temos o processo de parada do serviço.
1. Modo monusuário, conhecido como modo de manutenção.
2. Modo multiusuário básico.

3. Modo multiusuário completo, exceto interface gráfica.
4. Não utilizado.
5. Modo multiusuário completo com interface gráfica (padrão).
6. Modo de reinicialização do sistema (reboot).

De maneira geral definimos o nível 5 como o ideal para execução de um serviço, mas pode ser necessário definir em outros níveis em alguns casos específicos.

Para obter uma lista de serviços e seus níveis, utilizamos a opção `--list`. Veja um exemplo:

```
# /sbin/chkconfig --list
NetworkManager      0:off  1:off  2:off  3:off  4:off  5:on   6:off
NetworkManagerDispatcher 0:off  1:off  2:off  3:off  4:off  5:on   6:off
acpid              0:off  1:off  2:off  3:on   4:on   5:on   6:off
anacron             0:off  1:off  2:on   3:on   4:on   5:on   6:off
apmd               0:off  1:off  2:on   3:on   4:on   5:on   6:off
atd                0:off  1:off  2:off  3:on   4:on   5:on   6:off
ati-fglrx           0:off  1:off  2:on   3:on   4:on   5:on   6:off
auditd              0:off  1:off  2:on   3:on   4:on   5:on   6:off
autofs              0:off  1:off  2:off  3:on   4:on   5:on   6:off
bluetooth            0:off  1:off  2:on   3:on   4:on   5:off  6:off
cpuspeed             0:off  1:on   2:on   3:on   4:on   5:on   6:off
...
...
```

Para verificar quais serviços estão sendo executados em um nível específico:

```
# chkconfig --list | grep 5:on
```

Para procurar um serviço específico, podemos executar o seguinte comando:

```
# chkconfig --list| grep httpd
```

A inclusão de um novo serviço é feita pela opção `-add`. A sintaxe neste caso é:

```
Chkconfig --add <nome_do_serviço>
```

Antes de executar esse comando, devemos ter certeza de que o script do serviço em questão está dentro da pasta `/etc/init.d` ou `/etc/rc.d/init.d`, dependendo da distribuição Linux.

Ao executar a opção `-add`, o serviço é instalado em todos os níveis de inicialização, ou seja, do nível 0 até o nível 6. Uma cópia do serviço é feita nos diretórios `/etc/rc0.d`, `/etc/rc1.d`, `/etc/rc2.d`, ..., `/etc/rc6.d`. Exemplo:

```
# chkconfig --add httpd
```

Neste exemplo estamos adicionando o serviço `httpd` do servidor web apache ao processo de inicialização do Linux.

Nem sempre é preciso que um serviço seja executado em todos os níveis de inicialização. Para alterar em quais níveis o serviço deve ser executado, ou para habilitar ou desabilitar um determinado nível, utilizamos a opção `-level`:

```
Chkconfig --level <nível> <serviço> <on|off>
```

O exemplo a seguir mostra como habilitar apenas o nível 5 para o serviço httpd:

```
# chkconfig --level 12346 httpd off
```

Lembre-se de que o serviço deve estar criado antes de executar essa opção. Quando o serviço foi criado, ele foi habilitado em todos os níveis. Para confirmar que o serviço vai rodar no nível apropriado, utilize a opção `-list`:

```
# chkconfig --list | grep httpd
```

A remoção de um serviço do processo de inicialização deve ser feita pela opção `--del` do utilitário, cuja sintaxe é:

```
Chkconfig --del <serviço>
```

Veja um exemplo:

```
# chkconfig --del httpd
```

Com isso o processo será retirado de todos os níveis de inicialização, porém o original não será excluído de init.d.

3.7.2 - Service

Para o controle manual de um processo, utilizamos o aplicativo service, com o qual podemos iniciar ou parar manualmente um serviço e verificar a situação de um ou vários serviços. O arquivo relacionado ao serviço deve estar instalado no diretório init.d antes de executarmos o aplicativo service (não é necessário que chkconfig tenha sido executado previamente).

Para iniciar um serviço, utilizamos a opção `start`, cuja sintaxe é:

```
Service <serviço> start
```

Um exemplo:

```
# service httpd start
```

O resultado será algo como:

```
Starting httpd: [ OK ]
```

Caso o serviço não possa ser iniciado, uma mensagem de erro é retornada.

A parada de um serviço é conseguida com a opção `stop`, cuja sintaxe é:

```
Service <serviço> stop
```

Veja um exemplo:

```
# service httpd stop
```

O resultado será algo como:

```
Stopping httpd: [ OK ]
```

Novamente, caso haja algum problema, uma mensagem de erro será exibida.

Para ver a situação atual de um processo, utilizamos a opção status, cuja sintaxe é:

```
Service <serviço> status
```

Veja o exemplo:

```
# service httpd status
```

Obteremos como resultado algo como:

```
Httpd is Stopped
```

Se for necessário visualizar todos os processos e a situação de cada um deles, utilizamos a opção --status-all. Acompanhe um exemplo:

```
# service --status-all
```

O resultado será a lista completa de todos os serviços. Caso seja necessário filtrar somente os serviços ativos, podemos executar o comando da seguinte forma:

```
# service --status-all | grep running
```

O resultado será algo como:

```
acpid (pid 2146) is running...
anacron (pid 2236) is running...
atd (pid 2243) is running...
Init-Script is running.
auditd (pid 1944) is running...
cpuspeed (pid 1880) is running...
crond (pid 2206) is running...
cupsd (pid 2156) is running...
```

3.8 - Gerenciamento de Processos

Há várias ferramentas para gerenciamento de processos, algumas já abordadas em itens anteriores, tais como top e ps.

Muitas vezes precisamos eliminar um ou mais processos que estão sendo executados, seja por algum problema na sua execução, consumo excessivo de memória ou CPU, escassez de recursos no servidor, necessidade de troca de versão ou manutenção do sistema etc.

Todo e qualquer processo recebe um identificador único no sistema, o qual é conhecido como PID (Process Identifier ou identificador de processo). Por meio dele o sistema gerencia os processos, e sempre que for preciso realizar alguma ação no processo, devemos utilizá-lo (exceto com o comando killall).

Todos os processos em execução são registrados no diretório/proc/<PID>, dentro do qual obtemos uma gama enorme de informações do processo, tais como alocação de memória, módulo carregados, situação do processo, dispositivos alocados e outras informações úteis para o completo gerenciamento do processo.

3.8.1 - Kill e killall

Kill e killall são utilizados, como o próprio nome sugere, para remoção de processos do sistema. Kill geralmente faz a remoção de um processo, enquanto killall é útil quando queremos remover um grupo de processos.

A sintaxe do comando kill é:

```
Kill -s sinal PIDs
```

Sinal pode ser o número do sinal ou seu nome. Os principais sinais são:

» SIGINT	2
» SIGQUIT	3
» SIGKILL	9
» SIGSEGV	1
» SIGTERM	5

De forma geral utilizamos -9 (SIGKILL) ou -15 (SIGTERM).

PIDs define um ou mais pids que serão removidos. Se for uma lista, os pids devem ser separados por espaço. Exemplo:

```
# ps -A | grep php  
# kill -9 55605
```

O comando killall permite que todos os processos de um determinado programa sejam removidos. Sua sintaxe é:

```
Killall <opções> <processo>
```

As opções mais utilizadas são:

- i Remove de forma interativa, solicitando confirmação antes de remover cada um dos processos.
- q Modo silencioso, não exibe nenhuma informação do processo.
- w Diz ao comando que aguarde que os processos sejam removidos.

Veja um exemplo:

```
# killall mysqld  
# killall nautilus
```

3.9 - Superusuário

Superusuário ou usuário raiz ou usuário “root” é o usuário do Linux com direitos totais e irrestritos. Ele é criado durante a instalação do Linux e deve ser utilizado para dar manutenção ao sistema (para executar os itens mostrados - a maioria deles - neste capítulo, você teve de entrar como superusuário no sistema).

Ele deve ser utilizado somente pelo administrador do sistema e em hipótese nenhuma deve ser liberado para usuários comuns, sob risco de comprometimento da integridade do sistema operacional.

Você pode notar a diferença entre um usuário comum e o superusuário pelo símbolo exibido no prompt de comando. No caso do superusuário temos o símbolo # e para usuários comuns, o símbolo \$.

Se for necessário, é possível dar direitos de superusuário a um usuário comum, usando o comando su:

```
$ su -
```

A senha do superusuário será solicitada e, caso esteja correta, o usuário passa a ter as permissões do superusuário (inclusive o prompt é alterado para #).

Para sair do modo superusuário, basta digitar o comando exit:

```
# exit
```

Dica

Se for necessário apenas executar um comando como superusuário, é possível utilizar a opção -c no comando su.

Há ainda a opção de permitir que certos usuários executem comandos como superusuário sem a necessidade de informar a senha (o que é uma boa prática na maioria dos casos, pois senhas conhecidas por outros, além do dono, costumam tornar-se públicas).

Para isso utilizamos, em vez de su, o comando sudo, que autoriza os usuários a executarem comandos como se fossem o superusuário, sem necessidade da senha do superusuário.

É necessário configurar o arquivo /etc/sudoers, que é bem flexível e permite a definição de permissões, limitações e outras configurações úteis. O usuário deve sempre digitar sua senha para poder rodar o sudo (é possível inibir a senha pela opção NOPASSWD).

Para editar o arquivo de configuração, utilizamos o programa visudo.

Veja um exemplo do arquivo sudoers:

```
#  
# Sample /etc/sudoers file.  
#  
# This file MUST be edited with the 'visudo' command as root.  
#
```

```

# See the sudoers man page for the details on how to write a sudoers file.
#
## 
# User alias specification
##
User_Alias      FULLTIMERS = millert, mikef, dowdy
User_Alias      PARTTIMERS = bostley, jwfox, crawl
User_Alias      WEBMASTERS = will, wendy, wim

##
# Runas alias specification
##
Runas_Alias    OP = root, operator
Runas_Alias    DB = oracle, sybase

##
# Host alias specification
##
Host_Alias      SPARC = bigtime, eclipse, moet, anchor:\n
                SGI = grolsch, dandelion, black:\n
                ALPHA = widget, thalamus, foobar:\n
                HPPA = boa, nag, python
Host_Alias      CUNETS = 128.138.0.0/255.255.0.0
Host_Alias      CSNETS = 128.138.243.0, 128.138.204.0/24, 128.138.242.0
Host_Alias      SERVERS = master, mail, www, ns
Host_Alias      CDROM = orion, perseus, hercules

##
# Cmnd alias specification
##
Cmnd_Alias     DUMPS = /usr/sbin/dump, /usr/sbin/rdump, /usr/sbin/restore, \
                /usr/sbin/rrestore, /usr/bin/mt
Cmnd_Alias     KILL = /usr/bin/kill
Cmnd_Alias     PRINTING = /usr/sbin/lpc, /usr/bin/lprm
Cmnd_Alias     SHUTDOWN = /usr/sbin/shutdown
Cmnd_Alias     HALT = /usr/sbin/halt, /usr/sbin/fasthalt
Cmnd_Alias     REBOOT = /usr/sbin/reboot, /usr/sbin/fastboot
Cmnd_Alias     SHELLS = /usr/bin/sh, /usr/bin/csh, /usr/bin/ksh, \
                /usr/local/bin/tcsh, /usr/bin/rsh, \
                /usr/local/bin/zsh
Cmnd_Alias     SU = /usr/bin/su
Cmnd_Alias     VIPW = /usr/sbin/vipw, /usr/bin/passwd, /usr/bin/chsh, \
                /usr/bin/chfn

##
# Override built-in defaults
##
Defaults        syslog=auth
Defaults:FULLTIMERS !lecture
Defaults:millert !authenticate
Defaults@SERVERS log_year, logfile=/var/log/sudo.log

##
# User specification
##
# root and users in group wheel can run anything on any machine as any user
root    ALL = (ALL) ALL
%wheel  ALL = (ALL) ALL

# full time sysadmins can run anything on any machine without a password
FULLTIMERS     ALL = NOPASSWD: ALL

```

```
# part time sysadmins may run anything but need a password
PARTTIMERS      ALL = ALL

# jack may run anything on machines in CSNETS
jack    CSNETS = ALL

# lisa may run any command on any host in CUNETS (a class B network)
lisa    CUNETS = ALL

# operator may run maintenance commands and anything in /usr/oper/bin/
operator      ALL = DUMPS, KILL, PRINTING, SHUTDOWN, HALT, REBOOT, \
               /usr/oper/bin/

# joe may su only to operator
joe    ALL = /usr/bin/su operator

# pete may change passwords for anyone but root on the hp snakes
pete    HPPA = /usr/bin/passwd [A-z]*, !/usr/bin/passwd root

# bob may run anything on the sparc and sgi machines as any user
# listed in the Runas_Alias "OP" (ie: root and operator)
bob    SPARC = (OP) ALL : SGI = (OP) ALL

# jim may run anything on machines in the biglab netgroup
jim    +biglab = ALL

# users in the secretaries netgroup need to help manage the printers
# as well as add and remove users
+secretaries   ALL = PRINTING, /usr/bin/adduser, /usr/bin/rmuser

# fred can run commands as oracle or sybase without a password
fred    ALL = (DB) NOPASSWD: ALL

# on the alphas, john may su to anyone but root and flags are not allowed
john    ALPHA = /usr/bin/su [!-]*, !/usr/bin/su *root*

# jen can run anything on all machines except the ones
# in the "SERVERS" Host_Alias
jen    ALL, !SERVERS = ALL

# jill can run any commands in the directory /usr/bin/, except for
# those in the SU and SHELLS aliases.
jill    SERVERS = /usr/bin/, !SU, !SHELLS

# steve can run any command in the directory /usr/local/op_commands/
# as user operator.
steve   CSNETS = (operator) /usr/local/op_commands/

# matt needs to be able to kill things on his workstation when
# they get hung.
matt    valkyrie = KILL

# users in the WEBMASTERS User_Alias (will, wendy, and wim)
# may run any command as user www (which owns the web pages)
# or simply su to www.
WEBMASTERS      www = (www) ALL, (root) /usr/bin/su www

# anyone can mount/unmount a cd-rom on the machines in the CDROM alias
ALL      CDROM = NOPASSWD: /sbin/umount /CDROM, \
          /sbin/mount -o nosuid\,nodev /dev/cd0a /CDROM
```

O objetivo do arquivo de configuração tanto é liberar usuários para execução de comandos do superusuário quanto limitar sua atuação, impedindo o acesso irrestrito ao sistema, o que seria potencialmente perigoso. A flexibilidade é tal que podemos criar vários apelidos (alias), e desta forma agrupar usuários, comandos, programas etc. Veja, por exemplo, a linha:

```
jill    SERVERS = /usr/bin/, !SU, !SHELLS
```

Aqui temos que o usuário Jill pode executar qualquer comando no diretório /usr/Bin, exceto os que estão listados nos apelidos SU e SHELLS, que são:

```
Cmnd_Alias    SHELLS = /usr/bin/sh, /usr/bin/csh, /usr/bin/ksh, \
                     /usr/local/bin/tcsh, /usr/bin/rsh, \
                     /usr/local/bin/zsh
Cmnd_Alias    SU = /usr/bin/su
```

Já o usuário fred pode executar qualquer comando como oracle e Sybase sem que seja solicitada a sua senha.

```
fred    ALL = (DB) NOPASSWD: ALL
```

Já jen pode executar qualquer comando em todos os servidores, com exceção dos listados em SERVERS.

```
Host_Alias    SERVERS = master, mail, www, ns
```

Exercícios

- 1.** Explique o comando mkdir.
- 2.** Dê um exemplo do comando mv utilizando o modo forçado.
- 3.** Explique o comando chmod 755 teste*.
- 4.** O que acontece no comando chmod a=,u=rwx,g=r,o=x,g+w,o-x teste?
- 5.** Qual o comando usado para adicionar um novo grupo?
- 6.** O que faz o comando useradd? Dê um exemplo.
- 7.** Que comando gerencia senhas?
- 8.** O que faz o comando RPM?
- 9.** Para que serve apt-get?
- 10.** O que faz o programa free?
- 11.** Qual o resultado de fdisk -l?
- 12.** O que faz o comando SS?
- 13.** Como configurar um serviço para inicialização automática?
- 14.** Explique o comando kill.
- 15.** Para que serve o comando su?
- 16.** Qual a função do arquivo sudoers?

4

Instalação

4.1 - Introdução

Se este é o seu primeiro contato com uma instalação Linux, tenha certeza de que você vai instalar e reinstalar muitas distribuições até encontrar uma que lhe agrade. Graças ao mundo colaborativo e às iniciativas de torná-lo um sistema mais difundido, os instaladores de hoje não são cheios de mistérios e macetes, nem todos na linha inglesa, pois a maioria deles é multilíngua, além de possuir interface gráfica familiar àqueles acostumados com o famigerado Microsoft Windows.

Os instaladores modernos são verdadeiramente muito bem elaborados e ajudam o usuário, pois em quase sua totalidade, os instaladores gráficos são muito intuitivos. Basicamente eles ajustam as configurações de hardware e instalam os pacotes selecionados.

O hardware também era um grande problema há alguns anos, mas hoje a dinâmica de atualizações do kernel e módulos é muito rápida, pois os fabricantes cada vez mais se preocupam em lançar seus dispositivos com drivers também para Linux. Isso significa que as distribuições Linux estão cada vez mais compatíveis com computadores de marca, e a menos que você tenha uma máquina muito velha ou um computador montado com peças muito diferentes, não deve ter problemas com a instalação.

Neste capítulo veremos detalhes de cada uma das opções existentes em praticamente todos os instaladores. Usamos o CentOS como exemplo de instalação para conduzir as explicações. Quando você for instalar uma outra distribuição, provavelmente encontrará telas com layouts diferentes, mas o conceito sempre será o mesmo, por isso, ao entender o princípio de uma instalação, estará apto a se aventurar com a distribuição que desejar.

4.2 - Preparação para Instalar

Existem diversas maneiras de instalar uma distribuição Linux. Antes de iniciar é preciso avaliar a que melhor se encaixa com a sua realidade. Você pode instalar pelo método mais tradicional que é a partir de um CD-ROM ou DVD, ou pela rede através dos protocolos NFS, HTTP e FTP, além de poder instalar diretamente de um disco rígido ou pen drive.

Uma instalação via rede consiste basicamente em disponibilizar uma imagem da instalação do sistema através de um dos protocolos de rede compatíveis, ou seja, se fosse instalar a partir de um compartilhamento da Internet, teríamos de copiar a imagem do CD-ROM ou DVD de instalação para essa pasta, e ao iniciar a instalação, informar os dados para conexão ao servidor NFS.

A instalação através de um pen drive também não tem nenhum segredo. Basicamente é o mesmo que queimar uma imagem de uma instalação do Linux em um pen drive com boot para usar na instalação.

Veremos uma instalação a partir de um DVD para arquitetura de 32 bits. É importante verificar se o seu computador é de 64 ou 32 bits na hora de baixar a imagem de instalação. Este capítulo mostra como fazer para instalar em modo texto, pois se você consegue instalar uma distribuição em modo texto e entendeu como fez, para instalar qualquer outra não terá problemas, porque os instaladores são bem semelhantes e seguem uma mesma lógica.

A instalação pelo LiveCD, quando em modo gráfico, é praticamente igual àquela com DVD, no entanto oferece a capacidade de usar o sistema antes de ter instalado.

Para os estudos precisamos baixar essa imagem de DVD. Ela pode ser encontrada no site oficial do CentOS ou www.centos.org, ou no link seguinte você acessa o diretório com as imagens para download:

<http://centos-mirror.hostdime.com.br/centos/5.5/isos/i386/>

Após efetuar o download você precisa criar o DVD. Para queimar um DVD a partir da imagem baixada no formato ISO, pode usar qualquer programa de gravação de CDs e DVDs, como o Nero, por exemplo.

4.3 - Instalação

Para instalar a partir de um DVD, precisamos de um leitor de DVD instalado e também é necessário dizer para a BIOS do computador que deve perguntar ou efetuar o boot diretamente pelo leitor de DVD. Faça isso antes de prosseguir. Consulte o manual do fabricante da placa-mãe caso seja necessário.

A instalação será feita em modo texto. Não usaremos o ambiente gráfico, mas se você quiser experimentar o modo gráfico, não há problema, pois a sequência do instalador é muito parecida.

Início da Instalação

Coloque o DVD no leitor e reinicie ou ligue o computador. Em seguida aparece uma tela com as opções de boot e instalação, como na Figura 4.1.

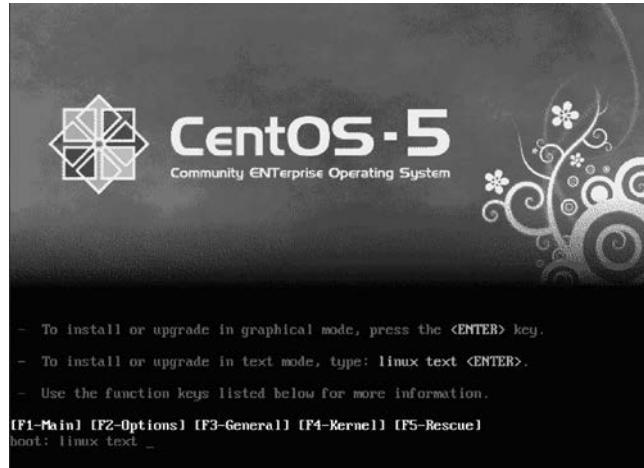


Figura 4.1

Nessa tela, se simplesmente pressionarmos ENTER, a instalação em modo gráfico é iniciada, mas não é o que queremos fazer nesse instante. Vamos observar que nessa tela existem logo abaixo algumas opções para as teclas de função: F2 mostra algumas opções de instalação, F3 exibe uma ajuda geral do sistema, F4 exibe informações sobre parâmetros do kernel e F5 informa sobre opções de emergência para recuperação de um sistema.

Para iniciar a instalação, vamos usar a opção de instalação “text”. Digite linux text seguido de ENTER. Isso inicia a instalação em modo texto.

Teste do DVD

Em seguida será solicitado que o DVD seja testado. Testar o DVD gravado é uma boa ideia nesse momento para evitar que a instalação seja interrompida por causa de problemas com o DVD gravado. Use as setas ou o TAB para alternar entre as opções, selecione OK para testar e Skip para ignorar o teste.



Figura 4.2

As Boas-Vindas

Terminado o passo de teste, o instalador tenta reconhecer alguns dispositivos de hardware necessários nesse momento, em seguida surge a tela de boas-vindas. Apenas pressione ENTER para prosseguir.



Figura 4.3

Seleção do Idioma

Uma tela com a seleção do idioma é mostrada, como na Figura 4.4. O idioma padrão é o English. Use as setas para selecionar o idioma desejado, TAB para chegar até o ENTER e confirme.



Figura 4.4

Seleção do Layout do Teclado

A seleção do teclado é sugerida conforme o idioma selecionado na tela anterior, mas você pode alterar caso não esteja correto.



Figura 4.5

Seleção de um Método para Configuração do Disco Rígido

Se estiver instalando em um disco rígido novo, que nunca foi formatado, pode ser surpreendido com a mensagem da tela mostrada na Figura 4.6. Isso não é motivo para pânico, pois ele está afirmando que não conseguiu ler a tabela de partição do disco e precisa iniciá-lo para prosseguir, mas avisa que todos os dados serão perdidos. Aqui vale o bom-senso. Você deve saber se pode ou não apagar tudo que existe no disco rígido, se é que há alguma informação.

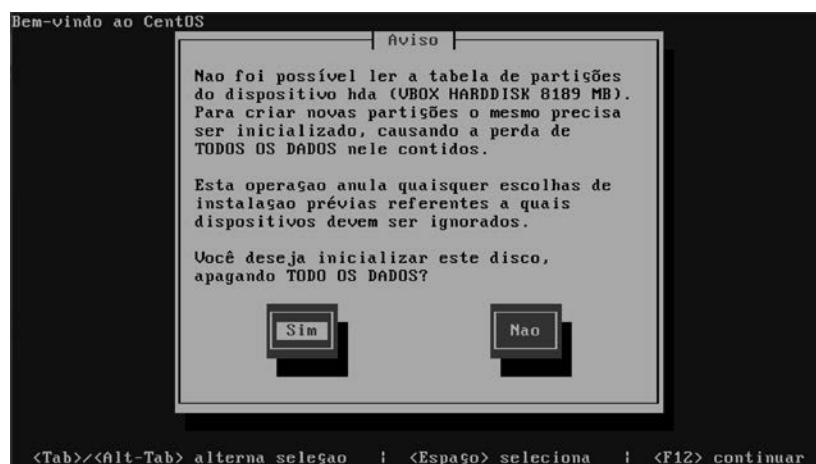


Figura 4.6

Se o seu disco rígido já havia sido formatado anteriormente, será direcionado para a tela de seleção do tipo de particionamento, conforme a Figura 4.7.

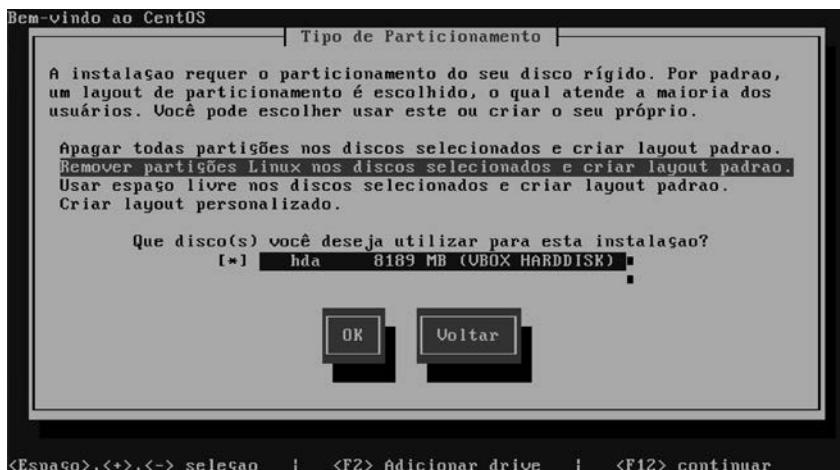


Figura 4.7

Esta é uma das etapas mais importantes da instalação, principalmente se você estiver instalando em um disco rígido que já possui outros sistemas operacionais, como o Microsoft Windows. Mas não se preocupe, pois a comunidade Linux se preocupa em coexistir com outros sistemas. Veremos o que significa cada um dos tipos de particionamento:

- » **Apagar todas partições nos discos selecionados e criar layout padrão:** é a melhor opção se você estiver usando um disco rígido que pode ser totalmente formatado e se quer apenas utilizar o CentOS como desktop, pois nela o instalador cria um layout padrão automaticamente sem intervenção do usuário.
- » **Remover partições Linux nos discos selecionados e criar layout padrão:** similar à primeira opção, entretanto desta vez apaga apenas as partições Linux para criar outro layout padrão. Ideal quando já temos um sistema Linux instalado e queremos nos desfazer dele e instalar o novo por cima. Nesse tipo todos os dados das partições Linux existentes são apagados.
- » **Usar espaço livre nos discos selecionados e criar layout padrão:** esta é a sem dúvida a melhor opção para quem está saindo, tem o Windows instalado e deixou um espaço livre para iniciar suas aventuras em Linux. Ela não altera nenhuma das partições existentes, ou seja, nenhum dado é perdido. Nessa opção o instalador cria um layout padrão automaticamente no espaço livre do disco.
- » **Criar layout personalizado:** essa opção é para usuários avançados. Se selecionada, o instalador abre um sistema completo de gerenciamento de partições para que você possa criar, alterar ou excluir as partições da forma que desejar, criando um layout personalizado. Não vamos detalhar esse processo porque está fora do escopo deste capítulo.

Seja qual for a opção selecionada neste item, uma confirmação será solicitada antes de prosseguir, pois o instalador quer ter certeza de que você não selecionou por engano.

Depois de definir o tipo de particionamento, é mostrada uma tela que pergunta se você deseja rever e modificar o layout das partições. Sempre responda sim, mesmo que não deseje alterar nada, pois é importante saber o que o processo automático está fazendo, para que no futuro você escolha o melhor layout. Na Figura 4.8 é possível ver essa tela.

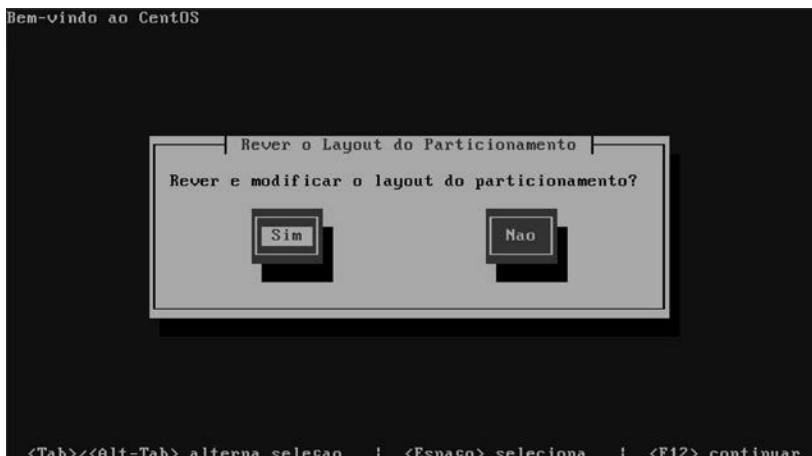


Figura 4.8

Se optou por rever o layout, uma tela como a da Figura 4.9 é aberta.

Particionamento					
Dispositivo	Início	Fim	Tamanho	Tipo	Ponto de Mon
UG VolGroup00			8864M	VolGroup	
LV LogVol01			1792M	swap	
LV LogVol00			6272M	ext3	/
/dev/hda					
hda1		1	13	101M	ext3
hda2	14	1844	8087M	physical	/boot

At the bottom of the window, there is a row of buttons: "Nova" (New), "Editar" (Edit), "Apagar" (Delete), "RAID" (RAID), "OK" (OK), and "Voltar" (Back). Below the window, a keyboard legend provides key mappings for various actions: F1-Ajuda, F2-Novo, F3-Editar, F4-Apagar, F5-Restaurar, and F12-OK.

Figura 4.9

O exemplo tem um disco de 8 GB apenas, e por padrão o instalador cria volumes lógicos do tipo LVM. Ele controla as partições do disco rígido como uma unidade lógica, permitindo que você facilmente adicione novas partições à unidade LVM. Com esse método você pode facilmente expandir o espaço em disco no seu sistema sem a necessidade de reconfigurar ou perder dados. Então automaticamente ele criou um grupo de volume com dois volumes lógicos, um para a partição SWAP com o dobro da quantidade de memória RAM disponível e outro com a partição / (raiz) com o restante do tamanho do grupo. A partição /boot não pode ficar em um volume LVM, por isso ela foi criada em separado com o sistema de arquivos ext3 para não haver problemas com os gerenciadores de boot.

Configuração do Gerenciador de Boot

Normalmente você vai querer instalar um gerenciador de boot. A menos que seja um usuário avançado e saiba instalá-lo em separado, sempre escolha a opção que confirma a instalação do gerenciador de boot. Uma tela como a da Figura 4.10 deve ser mostrada.

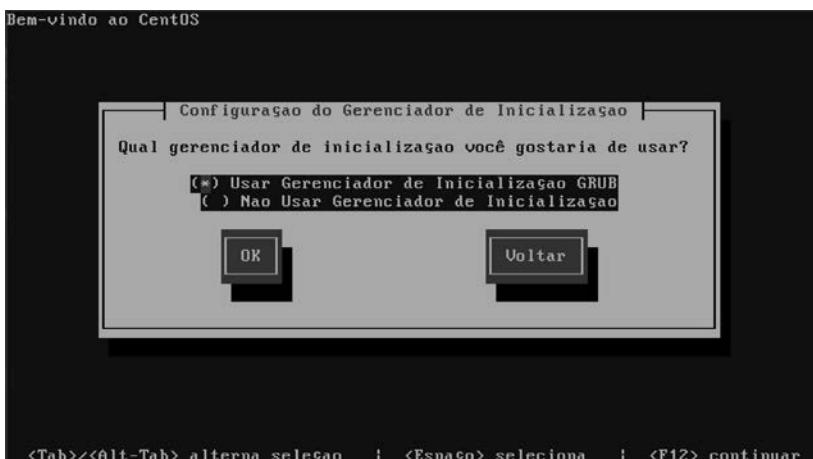


Figura 4.10

Em seguida surge uma tela solicitando parâmetros adicionais para o kernel que o gerenciador de boot irá iniciar. Isso acontece porque alguns sistemas às vezes precisam de parâmetros específicos para iniciar corretamente. Novamente este é um item para usuários avançados e na maioria das vezes você vai apenas confirmar no OK para prosseguir a instalação.

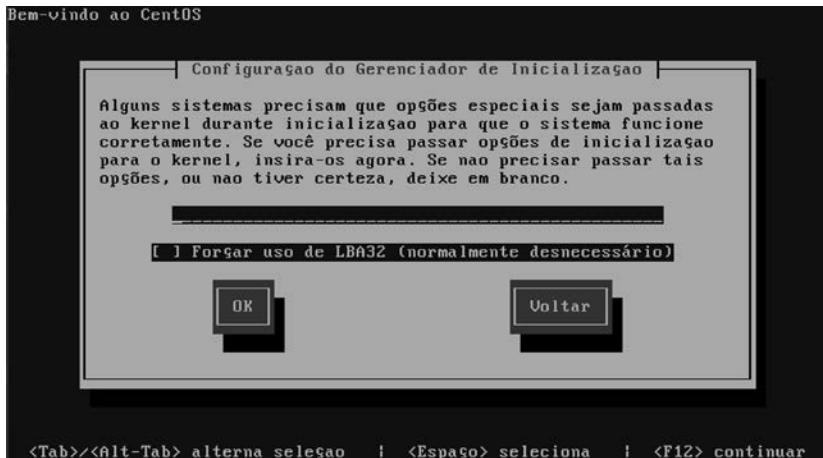


Figura 4.11

Agora o instalador quer saber se desejamos colocar uma senha na inicialização do sistema, antes mesmo de efetuar o boot. Na Figura 4.12 podemos ver esta tela.



Figura 4.12

O instalador quer saber se além do sistema que estamos instalando e daqueles que ele reconheceu, existe mais algum outro que desejamos colocar como opção de boot no gerenciador de inicialização ou se queremos editar um existente. O sistema que estiver como padrão iniciará automaticamente caso nenhum seja selecionado durante a inicialização. Observe na Figura 4.13 um único sistema, portanto ele também é o sistema padrão.

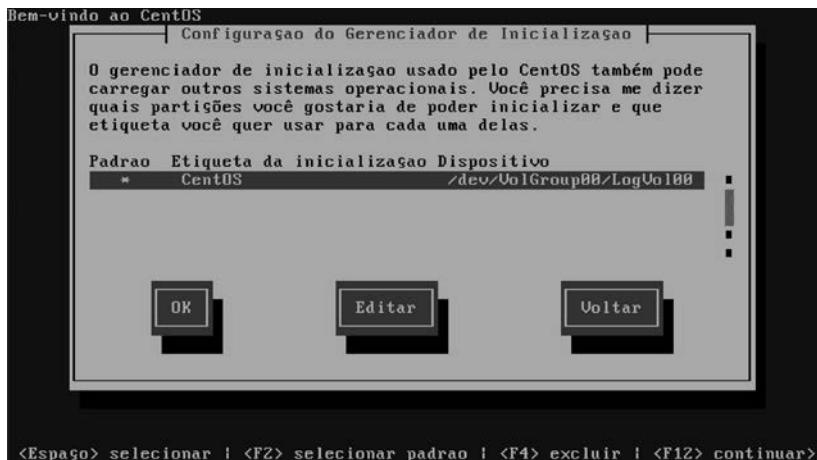


Figura 4.13

Para finalizar a configuração do gerenciador de inicialização, o instalador solicita em qual partição queremos instalá-lo. Se não for um usuário avançado, sempre escolha instalar no MBR (Registro Master de Inicialização), Figura 4.14.

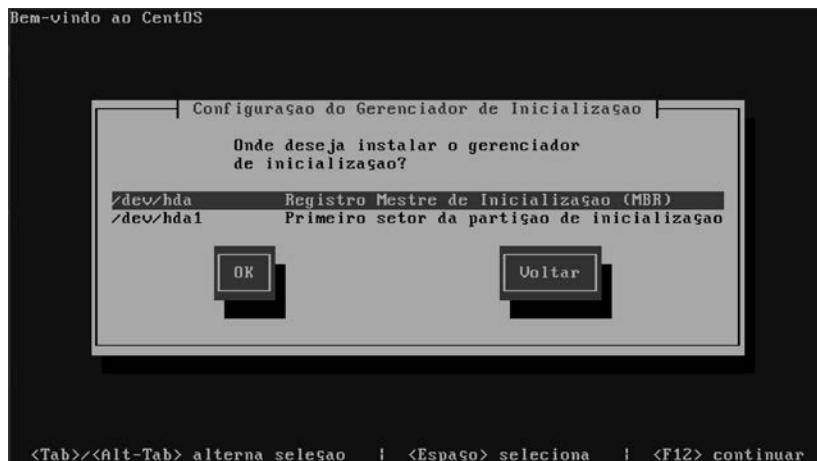


Figura 4.14

Configuração da Rede

Se o instalador detectar pelo menos uma placa de rede, será perguntado ao usuário se deseja configurá-la nesse instante. É prudente configurar a interface de rede nesse momento, pois economiza um serviço após a instalação. Siga as instruções e defina as configurações da sua rede.

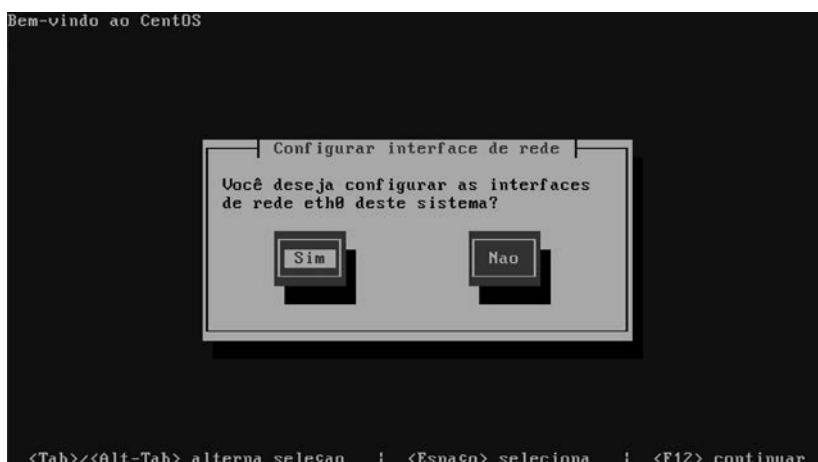


Figura 4.15

Após confirmar que deseja configurar uma interface de rede, o instalador pergunta se desejamos iniciar a placa automaticamente e qual protocolo ativar. Normalmente queremos que ative na inicialização e utilizamos o protocolo Ipv4. Use a barra de espaços para marcar as opções.



Figura 4.16

Em seguida informe as configurações de IP e máscara ou somente marque a opção DHCP. Observe a Figura 4.17.



Figura 4.17

Informe o nome do host ou deixe a opção automática se estiver usando DHCP, conforme a Figura 4.18.



Figura 4.18

Configuração do Fuso Horário

Nessa etapa apenas selecione o fuso horário correto e prossiga.



Figura 4.19

Definição de uma Senha para o Usuário Administrador

Estamos quase no fim desse emocionante “quiz”. Informe agora uma senha para o usuário root. É muito importante ter critério ao criar uma senha para o root, pois ele é o único usuário que tem poder total sobre o sistema instalado.

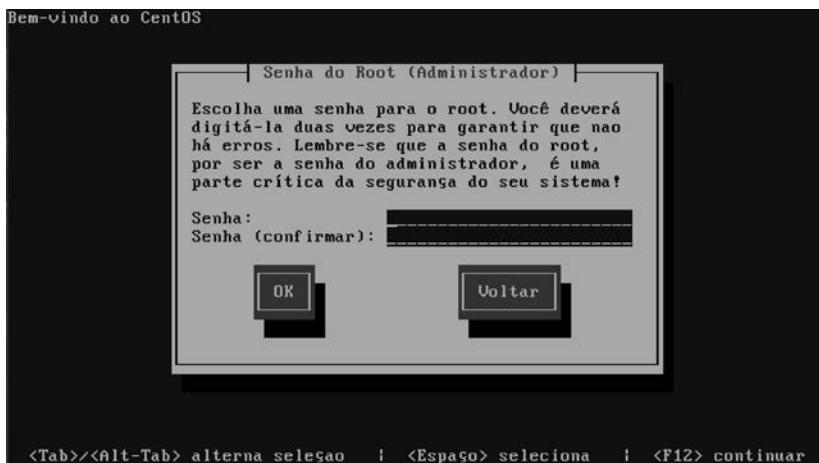


Figura 4.20

Seleção de Pacotes

Finalmente chegamos à seleção de pacotes. O CentOS e a maioria das distribuições fornecem conjuntos de pacotes macros para fins específicos, facilitando a tarefa de seleção de pacotes. Portanto, na maioria dos casos haverá um conjunto de pacotes predefinidos para um desktop com GNOME e outro com KDE, um para servidor, entre outros perfis de pacotes, Figura 4.21.

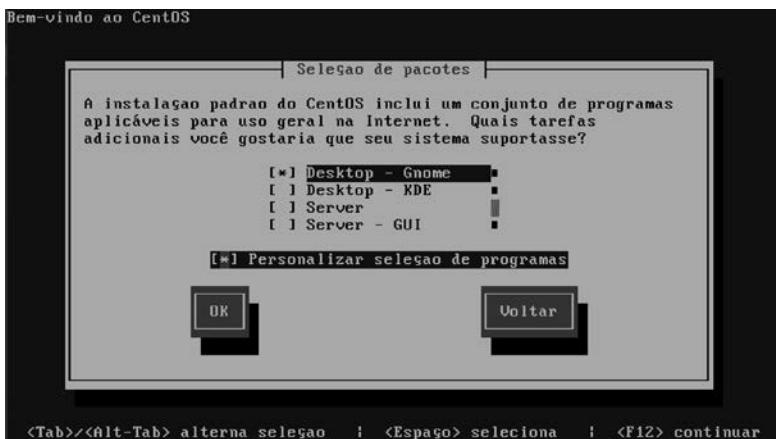


Figura 4.21

Mesmo assim, se você deseja selecionar algum pacote adicional, pode marcar a opção “Personalizar seleção de programas”. Então, ao confirmar alguns dos perfis macros, o instalador vai enviá-lo a uma tela onde pode escolher os grupos de pacotes que deseja instalar. Na instalação gráfica é possível, inclusive, selecionar um pacote específico individualmente. Veja a Figura 4.22.

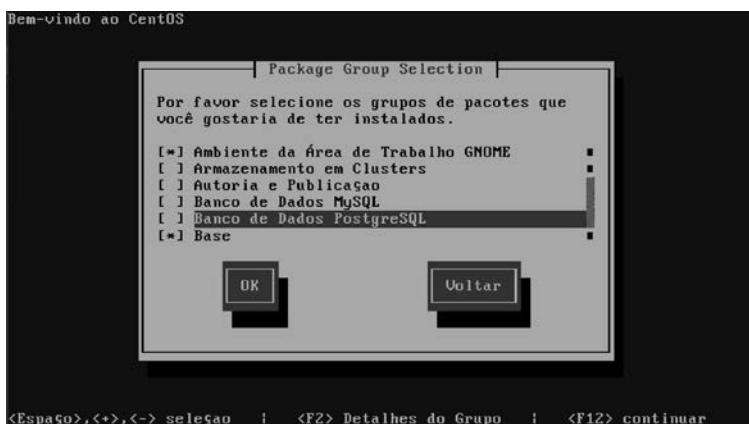


Figura 4.22

Após a seleção de pacotes o instalador verifica e inclui as dependências necessárias.

Finalização da Instalação

O instalador agora vai formatar e copiar a imagem de instalação para instalar os pacotes selecionados. Este é o último momento para desistir de tudo sem afetar nenhuma configuração em seu computador, portanto tenha certeza de que deseja prosseguir antes de pressionar OK. Você pode nesse instante voltar e alterar alguma configuração da instalação, mas se tudo estiver correto, apenas confirme pressionando OK nessa tela e aguarde a formatação, o cálculo do tempo necessário para instalar e a instalação dos pacotes.

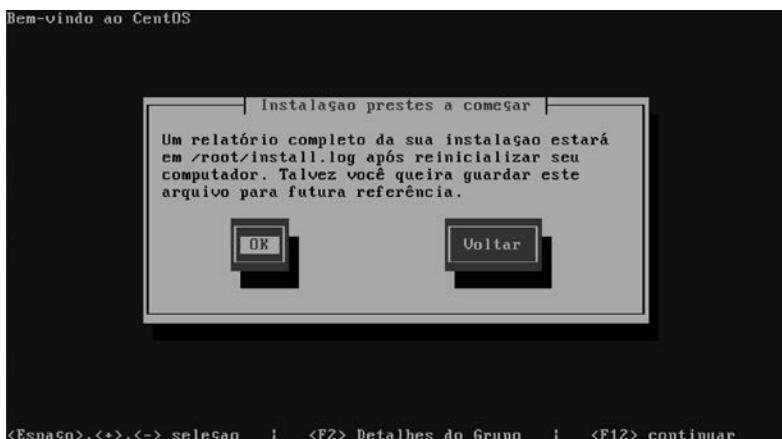


Figura 4.23

Um arquivo contendo detalhes da instalação será gravado na pasta do usuário root com o nome `install.log`. Ele pode ser útil no futuro.

Aguarde o término da instalação. A tela da Figura 4.24 deve ser mostrada durante o processo de instalação.



Figura 4.24

Ao terminar a instalação dos pacotes, o sistema solicita uma confirmação para reiniciar e iniciar o sistema pela primeira vez. Nesse momento é interessante retirar o DVD da leitora para evitar que ele entre novamente na instalação.

Como fizemos uma instalação em modo texto, mesmo que tenhamos instalado um perfil de pacotes para um desktop GNOME, ele inicia em modo texto. Para não perder tempo procurando como fazer, para iniciar o GNOME pode digitar startx a qualquer momento depois do login e também alterar o arquivo /etc/inittab e o parâmetro “id:3:initdefault:” para “id:5:initdefault:”.

Instalação pela Rede ou Disco

Para instalar a partir da rede usando NFS, HTTP ou FTP ou a partir de um disco rígido ou pen drive, você precisa iniciar a instalação com o parâmetro “askmethod”, ou seja, em vez de pressionar ENTER para instalar em modo gráfico, “linux text” para instalar em modo texto, vamos digitar linux askmethod e pressionar ENTER. Selecione o idioma e o teclado igual ao método de instalação por CD-ROM ou DVD que em seguida será solicitado o local da imagem de instalação, como podemos ver na Figura 4.25.



Figura 4.25

Para cada um dos métodos de rede são solicitadas as informações de configuração de rede igual àquela da instalação por DVD. Configure a rede e entre com os parâmetros para cada protocolo de rede. Se optar por instalar a partir de um disco, o sistema exibe uma tela com os discos que foram reconhecidos para você selecionar e um campo para informar o diretório onde a imagem da instalação baixada está. Lembrando que a imagem a ser utilizada no caso é a mesma que você usa para criar o DVD ou o CD.

Instalação Remota

Para instalar remotamente usando o protocolo VNC, você precisa iniciar a instalação com o parâmetro “vnc”, ou seja, em vez de pressionar ENTER para instalar em modo gráfico, linux text para instalar em modo texto, linux askmethod para escolher a origem da imagem, vamos digitar linux vnc e pressionar ENTER. Selecione o idioma, o teclado e as configurações de rede igual ao método de instalação por CD-ROM ou DVD, que em seguida será exibida uma tela com as informações para conexão via VNC a fim de prosseguir a instalação. A Figura 4.26 ilustra essa tela.

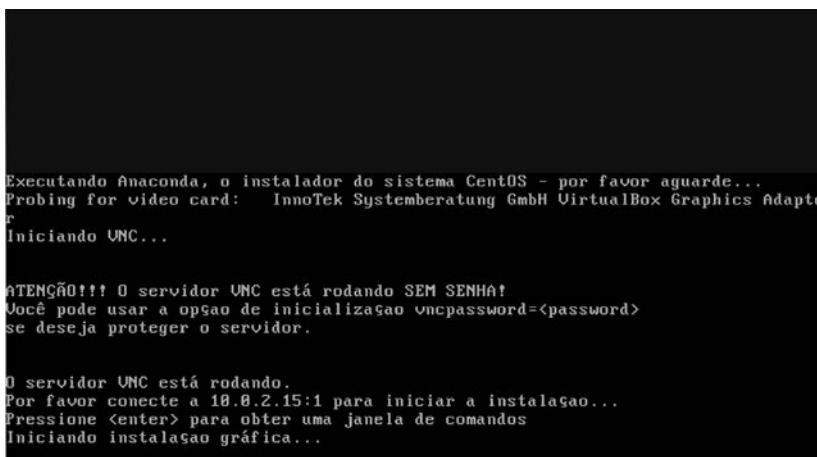


Figura 4.26

Conecte por VNC de qualquer computador na rede para continuar com a instalação.

Exercícios

1. Cite alguns meios de instalação do Linux.
2. Cite três fases da instalação.
3. O que acontece na finalização da instalação?
4. É possível instalar remotamente? Explique.

5

Inicialização e Desligamento do Linux

5.1 - Inicialização

O processo de inicialização do Linux, conhecido como bootstrapping ou apenas boot, é responsável pela execução inicial do sistema operacional, ou seja, pela carga do sistema operacional, que está no disco rígido ou em outra área de armazenamento secundária (CD-ROM, DVD, flash), para a memória RAM do computador.

O termo original bootstrapping é uma expressão inglesa e como metáfora, significa melhorar a si mesmo por seu próprio esforço e sem ajuda.

Em computação é sinônimo do processo de inicialização do sistema operacional. O processo é o primeiro a ser executado quando o computador é ligado e é responsável pelo carregamento do sistema operacional para a memória RAM.

O processo de boot ocorre em duas fases:

Fase 1

Nessa fase um programa simples que está na ROM do computador em um endereço predefinido é carregado e executado, o qual é conhecido por BIOS. A CPU é programada para executar esse programa sempre que o computador é ligado.

Durante o processo de boot o BIOS busca um dispositivo de inicialização, e caso seja encontrado, o BIOS executa o MBR (Master Boot Record) ou Registro mestre de inicialização. O MBR verifica a tabela de partições em busca de uma partição ativa que, uma vez encontrada, o MBR carrega e executa a inicialização que está na partição.

Fase 2

Nessa fase o BIOS acessa um segundo programa, que ainda não é o sistema operacional, e sim um gerenciador de inicialização ou de boot. É nessa fase que programas como o LILO ou GRUB são carregados e executados. Esses programas são lidos da memória secundária, uma vez que durante a carga e a execução do BIOS, as informações necessárias para leitura do disco ou de outra memória secundária já foram carregadas para a RAM.

A partir desse momento o sistema operacional já pode ser carregado na memória do computador e o controle do computador é transferido para ele. Nesse processo os drivers de dispositivos e outros processos são necessários para que o computador esteja operacional.

O tempo de carga pode variar de alguns segundos a alguns minutos, dependendo da complexidade do sistema operacional e das rotinas necessárias para sua operação. Assim um servidor tende a ter um boot mais demorado, pois é necessário inicializar vários processos e drivers.

5.2 - Gerenciadores de Boot (Boot Loaders)

No mundo Linux temos basicamente dois gerenciadores, sendo o mais utilizado atualmente o GRUB, e temos também o LILO.

5.2.1 - LILO

O LILO ou Linux Loader (Carregador Linux) é um gestor de arranque que permite inicializar um sistema operacional entre os instalados na máquina (somente um sistema operacional pode ser carregado por vez).

O LILO instala-se nos primeiros 512 bytes de qualquer dispositivo de armazenamento, imediatamente antes da tabela de partições, sendo desta forma independente de sistema operacional.

No LILO podemos ter até 16 núcleos (kernels), cada um com suas opções específicas.

A configuração do LILO pode ser feita pela edição do arquivo /etc/lilo.conf, mas requer algum conhecimento para sua manipulação.

Veja um exemplo desse arquivo:

```
#Lilo.conf

#Seção global do LILO
lba32
boot=/dev/hda
map=/boot/map
compact
delay 20
timeout 50
prompt
vga=0x317
default=Linux2.6.12.3

#Seção de imagens do LILO
#Imagem do Linux
image=/boot/Linux2.6.12.3
label=Linux2.6.12.3
root=/dev/hdah
read-only

#Imagem do Windows
```

```
other=/dev/hda2
table=/dev/hda
label=Windows
#Fim do arquivo
```

Entre as opções disponíveis as principais são:

- Boot Define o nome do dispositivo em que será gravado o setor de partida do LILO (normalmente é usada a partição ativa ou o Master Boot Record - MBR). Caso não seja especificado, o dispositivo montado como a partição raiz será usado.
- Install Instala o arquivo setor-boot como novo setor de boot do disco. Se omitido, /boot/boot.b é usado por padrão.
- lba32 Permite que o LILO quebre o limite de 1024 cilindros do disco rígido, inicializando o GNU/Linux em um cilindro acima deste. É altamente recomendado o uso dessa opção.
- map Especifica a localização do arquivo de mapa (.map). Se não for especificado, /boot/map é usado.
- password Permite proteger todas as imagens de disco com uma única senha. Caso a senha esteja incorreta, o LILO é novamente carregado.



Figura 5.1

5.2.2 - GRUB

O GNU GRUB ou apenas GRUB foi criado pelo projeto GNU para ser um multicarregador de sistemas operacionais. É utilizado, assim como o LILO, quando desejamos que a máquina tenha dois ou mais sistemas operacionais e desejamos realizar o boot em um ou em outro, conforme a necessidade.

GRUB é um programa que pode carregar qualquer arquivo executável que contenha um cabeçalho multiboot nos seus 8Kb iniciais. O cabeçalho é composto por uma sequência de bits, divididos da seguinte forma:

- » 32 bits para um número mágico
- » 32 bits com flags
- » 32 bits para um segundo número mágico
- » imagem do arquivo executável

O GRUB foi construído a partir do pacote Grand Unified Bootloader, de onde vem este acrônimo. Várias distribuições utilizam o GRUB como gerenciador de boot, entre eles o centOS.

Diferentemente dos gerenciadores convencionais que precisam manter uma tabela de blocos, o GRUB pode rastrear o sistema de arquivos existente. Alguns dos sistemas de arquivos suportados são:

- » Ext2
- » Ext3
- » Ext4
- » Iso9660
- » Minix
- » NTFS
- » VFAT
- » FAT16
- » FAT32

O processo de carga do sistema operacional pelo GRUB é:

1. O BIOS busca o dispositivo para gerenciamento de boot e carregamento do sistema operacional e, uma vez encontrado, move o controle para o MBR.
2. O MBR contém o primeiro estágio do GRUB, que é pequeno e faz apenas a carga do próximo estágio do GRUB que pode ser o 1.5 ou o 2.
3. O estágio 1.5, que está nos primeiros 30 KB da memória secundária imediatamente após o MBR, faz a carga do estágio 2.
4. É no estágio 2 que o usuário vê o menu com as opções de inicialização e pode escolher o sistema desejado ou realizar as alterações necessárias antes de sua carga.
5. Uma vez selecionado o sistema operacional, o GRUB executa o carregamento do seu núcleo (kernel) e este recebe o controle.

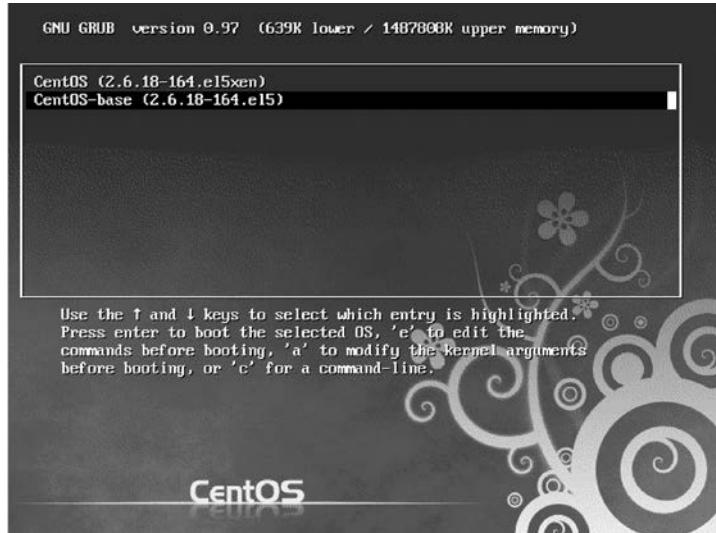


Figura 5.2

```
Booting 'CentOS-base (2.6.18-164.el5)'

root (hd0,0)
Filesystem type is ext2fs, partition type 0x83
kernel /vmlinuz-2.6.18-164.el5 ro root=/dev/VolGroup01/LogVol00 rhgb quiet
    [Linux-bzImage, setup=0x1e00, size=0xc31b4]
initrd /initrd-2.6.18-164.el5.img
    [Linux-initrd @ 0x37cda000, 0x315964 bytes]

Memory for crash kernel (0x80 to 0x80) not within permissible range
Red Hat nash version 5.1.19.6 starting
Reading all physical volumes. This may take a while...
Found volume group "VolGroup01" using metadata type lvm2
2 logical volume(s) in volume group "VolGroup01" now active
    Welcome to CentOS release 5.4 (Final)
    Press 'I' to enter interactive startup.
Configurando relógio (utc): Ter Out 19 21:28:15 BRST 2018 [ OK ]
Iniciando udev: _
```

Figura 5.3

5.2.3 - Inicialização em Modo Monousuário ou de Manutenção

Muitas vezes e por várias razões faz-se necessária a inicialização do Linux em modo usuário simples (single user mode). Entre estas razões está a perda da senha do root ou algum problema de software ou mesmo hardware que impede a inicialização completa do servidor nos níveis acima (o modo usuário simples é o correspondente ao nível 1).

No LILO isso é feito pelo parâmetro single ou 1 no processo de boot:

boot: linux single

ou

boot: linux 1

Se você acha isso um pouco inseguro, pode impedi-lo, editando o arquivo lilo.conf, adicionando as linhas logo após a seção image correspondente:

```
Password="sai_senha"
```

```
Restricted
```

Altere o modo do arquivo lilo.conf para 600 e evite que alguém não autorizado tenha acesso a ele e veja a senha, uma vez que ela não é criptografada. Deste modo, sempre que ocorrer o boot fora do padrão, isto é, que seja acrescentado algum parâmetro extra ao comando de boot, a senha será solicitada.

Já no GRUB precisamos dos seguintes passos:

1. Na tela inicial do GRUB selecione o sistema operacional desejado.
2. Utilize a tecla ‘a’ para adicionar conteúdo à linha de comando do sistema operacional.
3. Vá ao final da linha e digite a palavra single (use um espaço para separá-la da última palavra existente).
4. Pressione Enter para sair do modo de edição e executar a carga do sistema operacional.

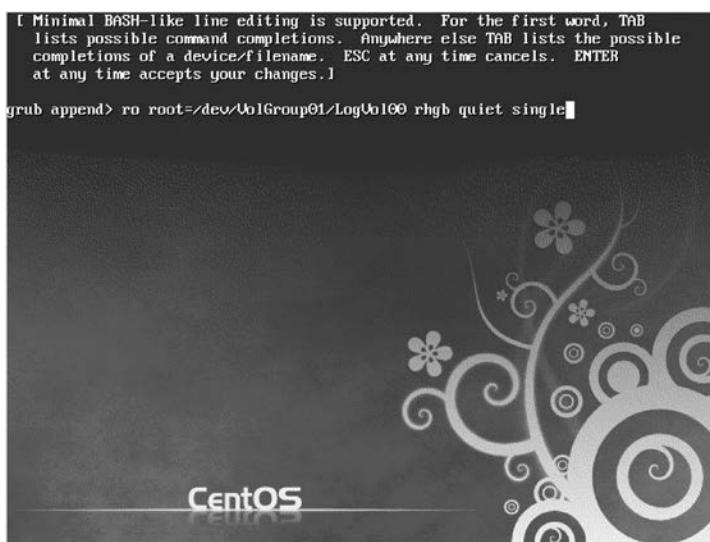


Figura 5.4

Assim como no LILO, é possível evitar o acesso a esse modo pelo GRUB. Para isso precisamos seguir alguns passos:

1. Digite o comando grub-md5-crypt para gerar uma senha criptografada (uma melhoria em relação ao LILO).

```
# grub-md5-crypt
```

2. Informe a senha que você deseja e a confirme.

```
Password:<SUASENHA>
Retype password:<SUASENHA>
$2$DDoR7r5Sgv6pxQ6LG4GXpfihIJyL0
```

3. Copie o resultado exibido na tela.
4. Edite o arquivo /boot/grub/menu.lst e logo em seguida insira o parâmetro password --md5 <sua_senha_md5>.

```
# vi /boot/grub/menu.lst
default      0
timeout      15
password    --md5 $2$DDoR7r5Sgv6pxQ6LG4GXpfihIJyL0
title        Debian GNU/Linux, kernel 2.6.18.6-cust-en-smp
root         (hd0,0)
kernel       /boot/vmlinuz root=/dev/hda2 ro
savedefault
boot
```

Pronto! O GRUB está seguro e sempre que for acionado o modo single boot, a senha será solicitada.

Lembre-se de que nesse nível poucos serviços estarão disponíveis. Por exemplo, a sua rede não estará ativa, quer seja local ou remota, além de outros processos que não estarão iniciados nesse nível.

Uma vez no prompt de comando do Linux, o qual entra diretamente em modo superusuário sem que seja necessário informar a senha, podemos utilizar o programa passwd para realizar a troca da senha do superusuário, ou tentar recuperar o Linux de algum problema existente.

Terminadas as ações necessárias, devemos reiniciar o sistema operacional (veja como no tópico 5.2.4).

5.2.4 - Desligamento

O desligamento não forçado do sistema operacional, isto é, de forma segura e ordenada, é fundamental para manter a integridade dele.

Para realizar o desligamento do sistema operacional, temos alguns métodos:

- ▶ Shutdown
- ▶ Init
- ▶ Halt, reboot e poweroff

Shutdown

Esse programa executa o desligamento do sistema operacional de forma segura e controlada. Todos os usuários ativos são notificados e novas entradas são bloqueadas. Os processos são notificados e têm tempo hábil para encerrarem de forma segura.

É possível executar o encerramento imediatamente ou após um determinado tempo ou em uma determinada hora e minuto.

É possível ainda comandar o sistema para que seja desligado ou reiniciado.

A sintaxe do comando shutdown é:

```
Shutdown <opções> <tempo> <mensagem de alerta>
```

As opções disponíveis são:

- t Informa o número de segundos entre o envio do aviso aos processos ativos e o envio do sinal de desativação do processo.
- r Reinicia o servidor após o término do processo.
- h Para ou desliga o servidor ao término do processo.
- k Não realiza o desligamento, apenas envia a mensagem de alerta.
- f Não executa o fsck na reinicialização.
- F força a execução de fsck na reinicialização.
- c cancela o processo já em andamento.

O parâmetro tempo pode ser informado nos seguintes formatos:

Hh:mm Hora e minuto no qual o processo será iniciado.

+mm Tempo de espera em minutos antes de iniciar o processo; +0 indica execução imediata.

now O mesmo que +0.

Caso o processo não seja executado imediatamente, o programa cria o arquivo de aviso /etc/nologin o qual informa ao sistema que nenhuma nova entrada de usuários deve ser autorizada.

Ao utilizar o parâmetro -f, o programa cria o arquivo /fastboot indicando ao sistema operacional que o comando fsck não deve ser executado.

```
[root@centOS ~]# shutdown -t 1 -r +2 0 sistema será reiniciado em 2 minutos
Broadcast message from root (pts/1) (Tue Oct 19 21:12:08 2010):
O sistema será reiniciado em 2 minutos
The system is going DOWN for reboot in 2 minutes!

Broadcast message from root (pts/1) (Tue Oct 19 21:13:08 2010):
O sistema será reiniciado em 2 minutos
The system is going DOWN for reboot in 1 minute!
■
```

Figura 5.5

```
CentOS release 5.4 (Final)
Kernel 2.6.18-164.el5 on an i686

INIT: Sending processes the KILL signal
Shutting down Cluster Module - cluster monitor: [ OK ]
Shutting down ricci: [ OK ]
Parando o setroubleshootd: [ OK ]
Desligando o smartd: [ OK ]
Desligando o servidor Avahi: [ OK ]
Desligando o oddjobd: [ OK ]
Stopping libvirtd daemon: [ OK ]
Parando atualização do yum: [ OK ]
Parando o anacron: [ OK ]
Parando o atd: [ OK ]
Parando o saslauthd: [ OK ]
Parando o cups: [ OK ]
Parando hpiod: [ OK ]
Parando hpssd: [ OK ]
Desligando o xfs: [ OK ]
Desligando serviços do mouse no console: [ OK ]
Parando o sshd: [ OK ]
Desligando sm-client: [ OK ]
Desligando o sendmail: [ OK ]
Parando o xiuctd: [ OK ]
```

Figura 5.6

init

Init é o pai de todos os processos, sendo sua função principal a criação de processos conforme definido em /etc/inittab. Podemos utilizar o programa init manualmente para troca do nível de execução.

A sintaxe de init é:

```
Init <nível>
```

O parâmetro nível pode variar de 0 a 6 (opcionalmente pode utilizar a letra s ou S para indicar o nível 1). Veja alguns exemplos:

```
# init 0
```

```
# init 6
```

No primeiro caso o sistema é desligado totalmente, sendo equivalente a shutdown -h now, e no segundo caso acontecem o desligamento e a reinicialização do sistema, o equivalente a shutdown -r now.

Não é recomendado utilizar init para desligar ou reiniciar o servidor, a não ser que tenhamos certeza de que não existem usuários ativos ou processos críticos em execução, pois a sua execução é imediata e irreversível.

Halt, reboot e poweroff

Esses utilitários permitem a parada, a reinicialização e o desligamento do servidor. Eles somente têm valor efetivo quando estamos nos níveis 0 ou 6. Caso o sistema esteja no demais níveis, o programa shutdown é executado no lugar destes, sendo passado o parâmetro apropriado a cada situação, ou seja, -h para Halt e poweroff e -r para reboot.

Os seguintes parâmetros podem ser utilizados junto com esses utilitários:

- f Força a execução do programa em vez de shutdown.
- i Derruba todas as interfaces de rede antes de continuar o processo.
- h Coloca todos os discos em estado de esperada antes de continuar o processo.

Tenha cuidado ao utilizar esses processos porque os resultados podem ser inesperados. Prefira sempre utilizar o programa shutdown que dispõe de mais recursos de controle, inclusive a possibilidade de abortar sua execução caso seja necessário (somente no modo de espera).

Exercícios

1. O que é MBR?
2. Quais são os gerenciadores de boot mais utilizados no Linux?
3. Como restringir a entrada no modo single user no LILO?
4. E no GRUB?
5. O que faz o comando shutdown?

6

Sistemas de Arquivos

Em capítulos anteriores já abordamos sistemas de arquivos, suas definições e estruturas no Linux. Agora vamos aprender um pouco como montar, desmontar, além de partições e outros dispositivos. Ainda vamos apresentar os principais tipos de arquivos do Linux.

A tabela seguinte mostra os tipos mais comuns de sistemas de arquivos.

Tabela 6.1

Sistema de arquivos	Descrição
ext3	O mais conhecido sistema de arquivos com journaling existente em Linux, ele é basicamente uma extensão do sistema de arquivos ext2 com journaling.
ext4	É o mais novo sistema de arquivos para Linux com journaling. É uma implementação do ext3 com muitas melhorias.
ext2	Sistema de arquivo nativo Linux não journaling.
ReiserFS	Foi o primeiro sistema de arquivo com suporte a journaling no Linux.
JFS	Sistema de arquivo com Journal criado pela IBM e bastante utilizado pelo sistema AIX UNIX.
XFS	Sistema de arquivos journaling de alta performance criado pela Silicon Graphics para o sistema operacional IRIX.
"SWAP"	Área da memória virtual.

6.1 - Montagem e Desmontagem

Para que uma partição, driver de DVD-ROM e outros dispositivos sejam utilizados, é necessário montá-los no sistema, e somente após esse processo a partição ou dispositivo estará disponível para utilização. O comando para montagem é o mount. Sua sintaxe é:

Mount <opções> <dispositivo> <diretório>

Existem várias opções para o comando. As principais são:

- t Define o tipo do arquivo de sistema (fstype).
- a Monta todas as partições existentes em fstab.
- r Monta o dispositivo apenas com permissão de leitura.
- w Monta o dispositivo para leitura e gravação. Este é o padrão.

O comando mount sem nenhum argumento extra mostra os dispositivos montados atualmente.

O parâmetro que mais utilizaremos é o -t, que informa o tipo de sistema de arquivo a ser empregado.

Caso o parâmetro -t não seja informado ou seja utilizado o valor auto (-t auto), o comando tenta descobrir o tipo de dispositivo que se está tentando montar. Caso não seja possível definir o tipo, uma mensagem de erro é retornada.

Em geral podemos usar a forma curta do comando:

```
Mount <dispositivo> <diretório>
```

Veja alguns exemplos:

```
# mount /dev/hda2 /mnt/disco1  
# mount /dev/cdrom /cdrom  
# mount /dev/fd0 /mnt/dsk  
# mount -t vfat /dev/hda3 /mnt/windows
```

Para desmontar uma partição, utilizamos o comando umount seguido do nome do dispositivo ou do diretório no qual ele foi montado. A sintaxe do comando é:

```
Unmount <dispositivo> | <diretório>
```

Exemplos:

```
# umount /dev/hda2  
# umount /mnt/dsk
```

É possível automatizar o processo de montagem de partições, tornando-as disponíveis durante o processo de inicialização do Linux. Para isso devemos editar o arquivo fstab, que se encontra geralmente no diretório /etc, e acrescentar uma linha de montagem referente a cada partição que desejamos automatizar na montagem.

O arquivo fstab exige que cada linha tenha o seguinte formato:

```
<dispositivo> <diretório> <tipo> <opções> <dump> <fsck>
```

Os parâmetros dispositivo, diretório e tipo são os mesmos informados no comando mount.

As opções disponíveis (principais) são:

- Auto Monta o dispositivo automaticamente. Este é o padrão.
- Noauto Montagem manual, ou seja, o dispositivo não é montado durante a inicialização do Linux.
- User Qualquer usuário pode montar a partição.
- Nouser Apenas o superusuário pode montar a partição. Este é o padrão.
- Exec Permite que arquivos binários sejam executados. Este é o padrão.
- Noexec Não permite que binários sejam executados.
- Rw Monta com permissão de leitura e escrita. Este é o padrão.
- RO Monta apenas com permissão de leitura.
- Sync Sincroniza os dados, ou seja, os dados são enviados diretamente para o dispositivo.
- async Trabalha na forma assíncrona, ou seja, os dados são gravados no dispositivo algum tempo depois que são gerados. Este é o padrão.
- Default Equivalente a rw, exec, auto, nouser, async.

Podemos utilizar várias opções juntas. Basta separá-las por vírgula.

O parâmetro <dump> é um sinalizador para o utilitário dump, que é responsável por realizar cópias de segurança em sistemas ext2/3. Se o parâmetro <dump> for zero, o dispositivo é ignorado; caso contrário, o valor é utilizado para decidir se a cópia deve ou não ser realizada.

O parâmetro <fsck> define se o comando fsck deve ser utilizado no dispositivo. Caso o seu valor seja zero, o dispositivo será ignorado. Se o valor informado for 1, o dispositivo será considerado o sistema de arquivo raiz (root) e 2 deve ser utilizado para outros dispositivos a serem verificados com fsck.

Exemplos:

/dev/cdrom	/media/cdrom	auto	ro,noauto,user,exec	0 0
/dev/fd0	/media/floppy	auto	rw,noauto,user,sync	0 0
/dev/hda2	/	ext2	defaults	1 1
/dev/hdb1	/home	ext2	defaults	1 2
proc	/proc	proc	defaults	0 0
/dev/hda1	swap	swap	pri=42	0 0

6.2 - Tipos de Arquivo

O Linux trata tudo como arquivo, seja um texto, um diretório, um dispositivo etc.

Para saber o tipo de um arquivo no Linux, usa-se o comando file, cuja sintaxe é:

```
File <opções> <nome_do_arquivo>
```

O comando tenta descobrir o tipo do arquivo por meio de suas características internas e compara-as com a biblioteca disponível em /usr/share/magic.

Em geral não é necessária a utilização de opções no comando. As principais são:

- b Formato curto de saída. Não mostra o nome do arquivo, somente o seu tipo.
- i Exibe o tipo no formato mime (algo como text/x-c ou application/x-executable).
- m Informa uma lista alternativa de arquivos contendo números mágicos para a avaliação dos arquivos.
- s Informa que os arquivos devem ser tratados como dispositivos especiais.
- z Tenta verificar arquivos comprimidos.

Exemplos:

```
# file file.c file /dev/wd0a /dev/hda
file.c:    C program text
file:      ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
          dynamically linked (uses shared libs), stripped
/dev/wd0a: block special (0/0)
/dev/hda:  block special (3/0)
# file -s /dev/hda{ ,1,2,3,4,5,6,7,8,9,10}
/dev/hda:   x86 boot sector
/dev/hda1:  Linux/i386 ext2 filesystem
/dev/hda2:  x86 boot sector
/dev/hda3:  x86 boot sector, extended partition table
/dev/hda4:  Linux/i386 ext2 filesystem
/dev/hda5:  Linux/i386 swap file
/dev/hda6:  Linux/i386 swap file
/dev/hda7:  Linux/i386 swap file
/dev/hda8:  Linux/i386 swap file
/dev/hda9:  empty
/dev/hda10: empty
# file -i file.c file /dev/{wd0a,hda}
file.c:     text/x-c
file:       application/x-executable, dynamically linked (uses shared
libs),not stripped
/dev/hda:   application/x-not-regular-file
/dev/wd0a:  application/x-not-regular-file
```

No Linux, como dito anteriormente, tudo é tratado como arquivo, desta forma fica mais simples (às vezes mais complicado) manipulá-lo. Se você quer lidar com um CD-ROM, por exemplo, verá que este é um arquivo especial em /dev, um diretório é um arquivo; enfim, tudo é tratado como tal.

Além dos arquivos comuns e dispositivos, temos ainda a possibilidade de criar arquivos que representam ligações (links) simbólicas ou físicas, ou seja, apontam para um outro arquivo. Ligações são úteis quando desejamos apontar para arquivos em outros diretórios ou mesmo partições.

As ligações (links) são identificadas com a letra `l` no início da listagem do arquivo. Por exemplo, no comando `ls -la`. No comando `file` aparecem como “symbolik link”. Acompanhe:

```
# ls -la /tmp/cpuspeed
```

```
lrwxrwxrwx. 1 root root 18 Out 22 11:14 /tmp/cpuspeed -> /usr/sbin/cpuspeed
```

```
# file /tmp/cpuspeed
```

```
/tmp/cpuspeed: symbolic link to `/usr/sbin/cpuspeed'
```

Link Simbólico

A ligação simbólica, ou link simbólico, é um arquivo especial do tipo link que aponta para localização do arquivo alvo. Suas características são:

- » Um link simbólico pode apontar para diretórios.
- » O alvo do link não precisa estar na mesma partição do link.
- » Caso o link simbólico seja removido, o alvo não será afetado.
- » Qualquer usuário, conforme suas permissões, pode criar um link simbólico.

Hard Link

O hard link, ou ligação física, aponta para o arquivo alvo e compartilha com ele o mesmo inode e o mesmo device, ou seja, são o mesmo arquivo. Isso afeta o comportamento desse tipo de ligação, tornando-o bem diferente de um link simbólico. Suas características são:

- » Não é possível fazer um hard link de um diretório.
- » O link e o alvo devem estar na mesma partição.
- » Se o hard link for removido, o alvo também será.
- » Somente o superusuário pode criar hard links.

A criação tanto de ligações simbólicas quanto físicas é realizada pelo comando `ln`, que possui as seguintes sintaxes possíveis:

```
ln <opções>[-T] <alvo> <link>
```

```
ln <opções> <alvo>
```

```
ln <opções> <alvos> <diretório>
```

```
ln <opções> -t <diretório> <alvos>
```

A primeira forma cria um link do alvo especificado com o nome definido no comando.

A segunda forma cria um link para o alvo especificado no diretório atual. O link terá o mesmo nome do alvo.

A terceira e quarta formas criam links no diretório especificado para a lista de alvos informada.

Por padrão, o comando cria hard links. Para gerar um link simbólico, deve-se utilizar a opção `-s`.

As opções disponíveis são:

- `-b` Faz uma cópia do arquivo caso o link informado exista.
- `-d` ou `-F` Permite que o superusuário tente criar um hard link para diretórios (geralmente isso falha).
- `-f` Remove qualquer arquivo destino da ligação que exista.
- `-i` Pergunta antes de remover arquivos.
- `-s` Define que o link será simbólico.
- `-t` Define o diretório no qual serão criadas as ligações.
- `-T` Trata o link como um arquivo comum.

Exemplos:

```
# ln -s /lib/libcrypt-2.12.1.so libcrypt.so  
# ln -s /lib/libk5crypto.so.3 libl5crypto.so  
# ln /root/executaprocesso
```

Exercícios

1. Explique o comando mount.
2. Para que serve a opção -r de mount?
3. Como desmontar um dispositivo montado com mount?
4. Como automatizar a montagem de dispositivos?
5. O que faz o comando file?
6. O que é um link simbólico?
7. O que é um hard link ou link físico?

7

Shell como Ferramenta do Administrador

7.1 - O Que é o Shell

O Shell é o interpretador de comandos do Linux. É um programa executado logo após a autenticação do usuário (login), sendo responsável por ler os comandos do usuário no prompt, interpretá-los e executar uma determinada ação. O processo realizado é o seguinte:

1. O Shell verifica se o que foi digitado corresponde a um comando interno do próprio Shell. Caso seja, o comando é executado.
2. Se não for um comando interno, o Shell procura na lista de diretórios definidos em PATH (caminho) um arquivo com o nome do comando digitado. Se encontrá-lo, ele é executado.
3. Caso o que foi digitado não seja um comando interno nem um programa dentro do PATH, uma mensagem de erro é exibida.

Funcionando como uma ponte de ligação, o Shell é o intermediário do núcleo do Linux e do usuário.

Veja alguns exemplos:

```
# ls -l  
# cat /etc/passwd  
# ps -ef|grep init
```

Existem muitos tipos de Shell, tais como csh, ksh, bsh, sh, bash etc. O bash (GNU Born Again Shell) é o mais utilizado atualmente, desta forma será utilizado como referência no livro.

7.2 - Shell Script

Quando desejamos executar uma lista de comandos do Shell, precisamos criar um arquivo que contenha esse conjunto de comandos, o qual chamamos de Shell script.

A primeira informação que todo Shell script deve ter é a indicação de qual Shell será utilizado e sua localização no sistema. A linha deve começar com os símbolos `#!`.

```
#!/bin/bash
```

Aqui indicamos que o Shell a ser utilizado é o bash e que ele está no diretório `/bin`.

Para executar um Shell script, devemos dar permissão de execução ao arquivo que o contém, para isso utilizamos o comando `chmod`. Veja um exemplo:

```
# chmod +x shell_script.sh
```

Acompanhe um exemplo de Shell script (não se preocupe se não entender muita coisa, pois tudo será explicado mais tarde):

```
#!/bin/sh
# listar.sh
# Script para leitura do diretório informado
# e exibição de informações sobre arquivos e diretórios
# Outubro-2010
DATUAL=${PWD}
read -p "Informe o diretório para busca: " DIR;
if [ -d ${DIR} ]; then
    cd ${DIR};
    for i in `ls`; do
        if [ -d ${i} ]; then
            echo "${i} é um diretório";
        else
            echo -n "${i} é um arquivo do tipo" `file -b ${i}`;
            if [ -s ${i} ]; then
                echo " e não está vazio";
            else
                echo " e está vazio";
            fi;
        fi;
    done;
else
    echo "${DIR} não existe ou não é um diretório";
fi;
echo "*** FIM ***";
cd ${DATUAL};

exit 0;
```

```
[root@centOS ~]# ./listar.sh
Informe o diretório de busca: /x
/x não existe ou não é um diretório

** FIM **
[root@centOS ~]# ./listar.sh
Informe o diretório de busca: ../
anaconda-ks.cfg é um arquivo do tipo ASCII English text e não está vazio
Desktop é um diretório
help é um diretório
install.log é um arquivo do tipo ASCII English text e não está vazio
install.log.syslog é um arquivo do tipo empty e está vazio
listar.sh é um arquivo do tipo Bourne-Again shell script text executable e não está vazio
scsrn.log é um arquivo do tipo ASCII text e não está vazio
teste000 é um diretório
teste0001 é um diretório
teste0002 é um diretório

** FIM **
[root@centOS ~]#
```

Figura 7.1

Salve esse script Shell no arquivo listar.sh e altere sua permissão para execução. Um possível resultado de sua execução é:

Ao mandar executar um script, o Shell atual, ou seja, o Shell no qual foi digitado o comando, carrega um novo Shell para processar o script (o qual é eliminado assim que o script termina de ser executado).

7.3 - Utilização de Shell e Shell Script

Como qualquer linguagem de programação, o Shell possui comandos e regras próprias para seu correto funcionamento. Vamos aprender agora um pouco desses comandos e regras.

A primeira regra você já sabe: informe sempre no início do arquivo script Shell o Shell que será utilizado e onde ele está localizado. Nossa padrão será:

```
#!/bin/bash
```

7.3.1 - Exibição

Para enviar dados para a saída padrão, em geral a tela do terminal, podemos utilizar os comandos echo e printf.

Echo

Esse comando é muito útil para exibição de textos e variáveis em formato mais simples com pouca ou nenhuma formatação. Veja alguns exemplos:

```
# nome="walace"
# echo "Meu nome é $nome"
# echo 10
```

Por padrão, echo insere um salto de linha ao final do texto exibido. Para inibir esse comportamento, utilizamos o parâmetro -n juntamente com o comando echo:

```
# valor=100  
# echo -n "O valor do produto é "  
# echo ${valor}
```

Neste exemplo a saída será:

O valor do produto é 100

Outros parâmetros para echo são:

- e Habilita o processamento de caracteres de controle.
- E Desabilita o processamento de caracteres de controle. Este é o comportamento padrão do comando echo.

Os caracteres de controle disponíveis são:

\a	Alerta (bell).
\b	Backspace.
\c	Suprime a terminação de nova linha (\n).
\e	Escape.
\f	Alimentação de formulário.
\n	Nova linha.
\r	Retorno de carro (utilizado antigamente para comandar impressoras em conjunto com \n).
\t	Tabulação horizontal.
\\\	Barra invertida.
\0NNN	NNN é traduzido como um código octal e o caractere ASCII correspondente, caso exista, é exibido.
\xnnn nnn	É traduzido como um código hexadecimal (base 16) o caractere ASCII correspondente, caso exista, é exibido.

Veja um exemplo:

```
# linhas=5  
# echo -e "\e[34mSalto de ${linhas} linhas\e[${linhas}L"
```

Aqui utilizamos o caractere especial \e para trocar a cor do primeiro plano para azul (\e[34m) e gerar um comando de salto de várias linhas (\e[\${linhas}L).

Printf

Com o comando printf podemos formatar e imprimir textos, valores etc. A vantagem em relação ao echo é sua flexibilidade, principalmente para formatação de valores. A sintaxe do comando é:

```
Printf <formato> <argumentos>
```

No parâmetro <formato> definimos como os dados serão apresentados. Os seguintes controles estão disponíveis:

- \a Alerta (bell).
- \b Backspace.
- \c Suprime a terminação de nova linha (\n).
- \e Escape.
- \f Alimentação de formulário.
- \n Nova linha.
- \r Retorno de carro (utilizado antigamente para comandar impressoras em conjunto com \n).
- \t Tabulação horizontal.
- \\\ Barra invertida.
- \0NNN NNN É traduzido como um código octal (base 8) de 3 dígitos e o caractere ASCII correspondente, caso exista, é exibido.
- \xNNN NNN É traduzido como um código hexadecimal (base 16) de 3 dígitos e o caractere ASCII correspondente, caso exista, é exibido.
- \uNNNN NNNN é traduzido como um código hexadecimal (base 16) de 4 dígitos.
- \UNNNNNNNN NNNNNNNN É traduzido como um código hexadecimal (base 16) de 8 dígitos.
- %% Exibe o caractere %.
- %b O argumento passado é tratado como texto e os caracteres escapes (iniciados com \) são processados.
- %Wd O argumento é tratado como um número inteiro com W dígitos de tamanho.
- %W.De O argumento é tratado no formato científico com um mínimo de W dígitos e D dígitos após a vírgula.

%W.Df O argumento é tratado no formato ponto flutuante com um mínimo de W dígitos e D dígitos após a vírgula.

%s O argumento é tratado como um texto.

Alguns exemplos:

```
# printf "Primeira Linha\nSegunda Linha\n"  
# valor=100  
# printf "o Valor é de %7.2f" $valor  
# for((i=2;i<=20;i+=2)); do printf "%5d" $i; done
```

7.3.2 - Variáveis

Variáveis são locais onde o Shell armazena valores a serem utilizados dentro do script Shell, ou seja, são áreas de memória onde se armazenam dados diversos. Os dados armazenados podem ser números, textos, vetores, listas variadas (a lista de arquivos de um diretório, por exemplo). Uma variável deve receber uma identificação, um nome, a qual será utilizada para fazer referência à variável em qualquer lugar do script.

O nome de uma variável pode conter números, letras e o caractere subscrito, e deve começar obrigatoriamente com uma letra ou subscrito, ou seja, não pode ser iniciada por um número. Letras maiúsculas e minúsculas são diferenciadas, portanto nome, Nome e NOME são consideradas variáveis distintas.

Para criar uma variável, precisamos apenas informar seu nome e seu valor:

```
FATOR=10
```

```
NOME="Walace"
```

```
Nome="Mara"
```

Caso o conteúdo seja alfanumérico, é necessário colocá-lo entre aspas.

Para acessar o conteúdo de uma variável, basta inserir o símbolo \$ antes do nome:

```
Echo $FATOR
```

```
Echo $NOME
```

```
Echo $Nome
```

Outra forma de acessar uma variável é colocar o seu nome entre chaves:

```
Echo ${FATOR}
```

```
Echo ${NOME}
```

```
[root@centOS ~]# env
SSH_AGENT_PID=2773
HOSTNAME=centOS.walace.soares.com.br
DESKTOP_STARTUP_ID=
TERM=xterm
SHELL=/bin/bash
HISTSIZE=1000
KDE_NO_IPV6=1
GTK_RC_FILES=/etc/gtk/gtkrc:/root/.gtkrc-1.2-gnome2
WINDOWID=25185985
USER=root
LS_COLORS=no=00:fi=00:di=00:34:ln=00:36:pi=40:33:so=00:35:bd=40:33:01:cd=40:33:0
1:or=01:05:37:41:mi=01:05:37:41:ex=00:32:*.cmd=00:32:*.exe=00:32:*.com=00:32:*.b
tm=00:32:*.bat=00:32:*.sh=00:32:*.csh=00:32:*.tar=00:31:*.tgz=00:31:*.arj=00:31:
*:tar=00:31:*.lzh=00:31:*.zip=00:31:*.z=00:31:*.Z=00:31:*.gz=00:31:*.bz2=00:31:*
*.bz=00:31:*.tz=00:31:*.rpm=00:31:*.cpio=00:31:*.jpg=00:35:*.gif=00:35:*.bmp=00:3
5:*.xbm=00:35:*.xpm=00:35:*.png=00:35:*.tif=00:35:
GNOME_KEYRING_SOCKET=/tmp/keyring-FJNFYt/socket
SSH_AUTH_SOCK=/tmp/ssh-sLFUwQ2737/agent.2737
KDEDIR=/usr
SESSION_MANAGER=local/centOS.walace.soares.com.br:/tmp/.ICE-unix/2737
MAIL=/var/spool/mail/root
PATH=/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin:/sbin/:
bin:/usr/sbin:/usr/bin:/usr/X11R6/bin:/root/bin
DESKTOP_SESSION=default
GDM_XSERVER_LOCATION=local
INPUTRC=/etc/inputrc
PWD=/root
XMODIFIERS=@im=none
```

Figura 7.2

Uma variável pode ser local, isto é, está disponível somente na função atual ou no Shell atual. E pode ser global, ou seja, disponível para o Shell atual e todos os subprocessos. Uma variável é tornada global pelo comando export do Shell. Para verificar as variáveis globais disponíveis, utilize o comando env.

Existem diversas variáveis predefinidas, muito úteis na construção de scripts Shell. No comando env mostrado anteriormente, temos a lista das variáveis predefinidas disponíveis. As principais são:

PS1

Essa variável apresenta o texto para o prompt primário, isto é, o texto que é exibido pelo Shell em cada linha de comando. É possível alterar o formato dessa variável utilizando alguns controles especiais. Observe os principais:

- \d Exibe a data atual.
- \h Nome do servidor.
- \s Nome do Shell.
- \t Hora atual - padrão 24 horas.
- \T Hora atual - padrão 12 horas.
- \u Usuário.
- \w Diretório atual, caminho completo.
- \W Diretório atual, somente o diretório.

Veja um exemplo:

```
# export PS1="\d \t | \u | \W#\ ";
```

O resultado será algo como:

```
[ Qui Out 21 19:45:16 | root | ~]#
```

PWD

Essa variável contém o diretório atual.

PATH

A variável PATH armazena o caminho de busca para o Shell, ou seja, a lista de diretórios para a busca de um programa ou arquivo. Quando digitamos o nome de um programa no Shell, todos os diretórios listados em PATH (e somente estes) são verificados em busca do arquivo, caso o caminho completo do arquivo não tenha sido especificado.

USER

Contém o nome do usuário ativo.

UID

Nessa variável está a identificação do usuário ativo.

SHLVL

Armazena o nível do Shell atual, isto é, quantos Shells foram executados a partir do Shell base (o Shell que foi executado após o login).

HOSTNAME

Contém o nome completo do servidor.

SHELL

Armazena o Shell que está sendo executado.

No Shell é possível tornar uma variável para leitura apenas, ou seja, o seu conteúdo, uma vez definido, não pode ser alterado. Para isso devemos utilizar a palavra-chave `readonly` antes de definir a variável. Sua sintaxe é:

```
 Readonly <nome_variavel> = <valor>
```

Um exemplo:

```
# readonly PI=3.14
```

```
# PI=10
```

```
// Erro readonly variable
```

Utilizamos este conceito em geral para constantes em nosso script Shell, assim impedimos que seu valor seja accidentalmente alterado.

Para ver todas as variáveis readonly, ou seja, somente leitura, usamos o comando readonly sem nenhum argumento ou com o parâmetro -p:

```
# readonly  
# readonly -p
```

Outra forma de declarar uma variável somente leitura é pelo comando declare, cuja sintaxe é:

```
Declare -r <nome_variavel>=<valor>
```

Veja um exemplo:

```
# declare -r K=5
```

O efeito de declare é o mesmo de readonly, ou seja, a variável não pode ter seu valor redefinido no restante do script.

Dica

Um outro uso para declare é definir uma variável como do tipo inteiro, o que pode ser útil em algum momento do script. Para definir uma variável como do tipo inteiro, utilizamos o parâmetro -i junto com declare. Veja um exemplo:

```
# declare -i numero=10
```

Caso se tente alterar o valor da variável com um dado não numérico, o Shell assume o valor zero para a variável.

Para remover uma variável utilizamos o comando unset, cuja sintaxe é:

```
Unset <nome_variavel>
```

Veja o exemplo:

```
# k=10  
  
# readonly x="teste"  
  
# unset k  
  
# unset x
```

O último comando não será executado e retornará um erro, uma vez que variáveis somente leitura (readonly) não podem ser removidas depois de criadas.

7.3.2.1 - Escopo

Uma variável é, por padrão, limitada ao Shell atual, ou seja, não é exportada para outros processos filhos. Veja um exemplo:

```
# nome="Linux"  
  
# echo ${nome}  
  
Linux  
  
# bash  
  
# echo ${nome}
```

O resultado deste último comando será uma string vazia, pois a variável não foi exportada para a nova instância do Shell, ou seja, é uma variável local.

Para exportar uma variável para os subprocessos, devemos utilizar o comando `export` antes da definição da variável. Acompanhe o exemplo anterior alterado para exportar a variável:

```
# export nome="Linux"  
  
# echo ${nome}  
  
Linux  
  
# bash  
  
# echo ${nome}  
  
Linux
```

Para ver todas as variáveis exportadas, utilizamos `export` sem argumentos ou então com o parâmetro `-p`:

```
# export  
  
# export -p
```

7.3.2.2 - Aspas Simples e Aspas Duplas

Nos comandos Shell, como, por exemplo, o `echo`, utilizar aspas simples ou duplas para delimitar textos define o comportamento desejado do Shell. Se utilizamos aspas simples, informamos ao Shell que o texto delimitado por elas não deve ser processado, isto é, o Shell não tentará substituir variáveis informadas. Já se delimitarmos o texto com aspas duplas, o processamento e substituição serão executados. Veja um exemplo da utilização desses dois delimitadores:

```
# echo "O diretório atual é $PWD"
```

O diretório atual é /root

```
# echo 'O caminho definido no sistema é $PATH'
```

O caminho definido no sistema é \$PATH

Nos textos delimitados com aspas podemos utilizar a barra invertida para informar que caracteres especiais devem ser tratados como comuns e não ser processados. É útil também quando desejamos incluir uma ou mais aspas duplas dentro do texto. Veja:

```
# echo "Conteúdo de \$PWD é $PWD"
```

Conteúdo de \$PWD é /root

```
# echo "\"aspas\""
```

"aspas"

7.3.2.3 - Vetores (Arrays)

Um vetor ou array é utilizado para armazenar vários valores em um mesmo nome de variável. Cada valor é associado a um índice. O índice deve ser utilizado para atribuição de valores e para recuperação dos valores armazenados.

Podemos definir um vetor de duas maneiras:

1. NomedoVetor[índice]=Valor
2. NomedoVetor=(valor1 valor2 valor3 ... valorn)

Exemplos:

```
# semana[0]="Domingo"  
# semana[1]="Segunda-feira"  
# semana[2]="Terça-feira"  
# semana[3]="Quarta-feira"  
# semana[4]="Quinta-feira"  
# semana[5]="Sexta-feira"  
# semana[6]="Sábado"  
# meses=(Jan Fev Mar Abr Mai Jun Jul Ago Set Out Nov Dez)
```

A recuperação do valor de uma posição do array é idêntica à recuperação de valores de variáveis, exceto pelo fato de que precisamos informar o índice e utilizar chaves para delimitar tanto o nome da variável quanto o índice. A seguir, alguns exemplos:

```
# echo ${semana[6]}
```

```
# printf "O primeiro mês do ano é %s" ${meses[0]}
```

7.3.2.4 - Substituição de Variáveis

No Shell é possível controlar as variáveis condicionalmente, ou seja, caso a variável não esteja definida ou esteja vazia, pode-se instruir o Shell a executar uma determinada ação. Temos:

```
 ${variável:-texto}
```

Se a variável não estiver definida ou seu conteúdo for vazio, é retornado o conteúdo de texto, porém a variável não é alterada.

```
# echo ${URL:-"http://www.erica.com.br"}
```

```
http://www.erica.com.br
```

```
# echo $URL
```

```
 ${variável:=texto}
```

Se a variável não estiver definida ou seu conteúdo for vazio, é retornado o conteúdo de texto e a variável recebe o mesmo valor.

```
# echo ${URL:="http://www.erica.com.br"}
```

```
http://www.erica.com.br
```

```
# echo $URL
```

```
http://www.erica.com.br
```

```
 ${variável:?texto}
```

Se a variável não estiver definida ou seu conteúdo for vazio, o valor de texto é escrito na saída de erro padrão (stderr) e a variável não é alterada.

```
# echo ${URL:??"Nenhuma URL definida"}
```

```
Bash: URL: Nenhuma URL definida
```

```
 ${variável:+texto}
```

Se a variável estiver definida, será substituída por texto, mas seu valor não será alterado.

```
# URL= "http://www.erica.com.br"
```

```
# echo ${URL:+ "URL atual definida como ${URL}"}
```

```
 URL atual definida como http://www.erica.com.br
```

```
# echo $URL
```

```
http://www.erica.com.br
```

7.3.2.5 - Substituição de Comandos

Muitas vezes precisamos executar um comando e obter o resultado gerado. Isso é conseguido no Shell pelo método de substituição de comando, que pode ser efetuado de duas formas:

`$(comando)`

``comando``

A substituição do comando permite que você capture o resultado, armazenando-o em uma variável ou manipulando-o diretamente.

Veja um exemplo:

```
#!/bin/bash
if [ $# -ne 1 ]; then
    echo -e "Favor informar o diretório origem\n"
else
    dir=$(pwd)
    echo "Realizando cópia de segurança de $1"
    cp -R -p -v $(ls $1) /home/backup/${basename $1} /
    echo -e "\n Fim da cópia\n"
fi
exit 0
```

7.3.2.6 - Substituição do Til (~)

Outra ação interessante que em algumas situações pode ser necessária é a substituição do til (~), a qual pode retornar as seguintes informações:

- ~ Retorna o valor da variável HOME, por exemplo /home/walace.
- ~+ Retorna o valor da variável PWD.
- ~- Retorna o valor de OLDPWD.

7.3.3 - Entrada de Dados

A captura de dados pode ser feita de vários locais:

1. Entrada de dados por parâmetros.
2. Captura de dados diretamente do usuário.
3. Recebimento do resultado de um outro processo.

A entrada de dados por parâmetro será tratada no tópico seguinte.

Para capturar os dados diretamente do usuário, ou seja, pelo teclado, devemos utilizar o comando read. Sua sintaxe é:

`Read <opções> <variáveis>`

- n Determina o número de caracteres que serão lidos pelo comando. Ao chegar ao número de caracteres definidos, o comando interrompe sua execução.
- p Informa um texto que será exibido pelo comando antes de iniciar a captura dos dados.
- s O que for digitado não é exibido na tela. Opção útil para entrada de senha.

Podemos informar uma ou mais variáveis para receber os dados digitados, mas o que separa o conteúdo de uma variável de outra é o espaço, desta forma a entrada de textos compostos por mais de uma palavra não será possível. Neste caso, opte por vários comandos read, um para cada variável.

Alguns exemplos:

```
#!/bin/bash  
echo -n "Informe a sua altura em cm: "  
read -n 3 altura  
echo  
read -p "Informe seu peso: " -n 3 peso  
fator=$(echo "scale=3; ${peso}/(${altura}/100 ^ 2)" | bc -l)  
printf "\nSeu fator de massa corporal é %s\n" $fator
```

Apesar de ser utilizado principalmente para entrada de dados pelo teclado, o comando read serve para o processamento de uma entrada automatizada, vindas de um arquivo ou de outro processo, para isso utilizamos o redirecionador de saída | (pipe). Acompanhe um exemplo:

```
#!/bin/bash  
echo -e "\e[34m\n**Tabela de preços 2010 **\n"  
egrep -v "^\#|^*$" $1 |  
while read CODIGO PRODUTO PRECO; do  
    echo -e "\e[31m${CODIGO}-${PRODUTO}=${PRECO}\n"  
done  
echo -e "\e[0m"
```

Salve o Shell script como imprimetabela.sh e altere sua permissão para execução:

```
# chmod +x imprimetabela.sh
```

O script espera que o nome de um arquivo contendo a tabela de preços seja informado como parâmetro e que esse arquivo tenha em cada linha três valores separados por espaço.

O primeiro indica o código do produto, o segundo sua descrição e o último seu preço. É algo como indicado na Figura 7.3.

O script lê o arquivo, ignora linhas em branco e comentários (linhas iniciadas com #) e processa cada linha como entrada para o comando read, em seguida o produto e seu preço são exibidos.

Observe um possível resultado na Figura 7.4.

```
#Tabela de Preços base 2010
1001 Refrigerante 1,50
1200 Arroz 6,80
3200 Feijão 3,10
3250 Farinha 0,90
4000 Leite 1,84
4100 Iogurte 3,45
5000 Alcatra 12,56
5100 filé-mignon 23,56
5150 maminha 9,78
```

Figura 7.3

```
[root@centOS ~]# ./imprimetabela.sh ./tabela.txt
** Tabela de Preços **

1001-Refrigerante = R$ 1,50
1200-Arroz = R$ 6,80
3200-Feijão = R$ 3,10
3250-Farinha = R$ 0,90
4000-Leite = R$ 1,84
4100-Iogurte = R$ 3,45
5000-Alcatra = R$ 12,56
5100-filé-mignon = R$ 23,56
5150-maminha = R$ 9,78
```

Figura 7.4

A função de | (pipe) é conectar a saída padrão de um comando com a entrada padrão de outro, permitindo assim um fluxo de dados entre diferentes processos.

7.3.4 - Passagem de Parâmetro para o Script Shell

Muitas vezes precisamos ou queremos executar o script Shell enviando pela linha de comando os parâmetros necessários, mas a questão é: como tratar os parâmetros recebidos dentro do script Shell? A resposta é: utilize as variáveis especiais, conhecidas como variáveis de parâmetro.

Elas são diferenciadas das variáveis comuns do script Shell (abordaremos variáveis mais adiante), possuindo características próprias. A lista a seguir mostra as variáveis de parâmetros disponíveis:

- \$0 Contém o nome do script Shell executado.
- \$n Enésimo parâmetro passado ao script Shell; n é maior ou igual a 1 e não tem limite. Caso n seja maior que 9, a referência deve ser \${n}. Desta forma, para acessar o primeiro parâmetro utilizamos \$1 e o de número 15 será \${15}.
- \$* Todos os parâmetros em uma única linha.

- \$@ Todos os parâmetros. Cada parâmetro é uma linha.
- \$# Número de parâmetros passados ao script Shell.
- \$? Valor do retorno do último comando executado.
- \$\$ PID (identificador do processo) do script Shell.
- \$! PID do último comando iniciado com &.

Veja um exemplo:

```
#!/bin/bash
# multiplica.sh
# multiplicação de vários números
# Outubro-2010
if [ $# -le 1 ]; then
    echo "Você deve informar pelo menos dois números";
else
    total=1;
    mult="";
    for num in $@;do
        total=$((total * num));
        if [ -z "$mult" ]; then
            mult=$num;
        else
            mult=$((mult * num));
        fi;
    done;
    echo "${mult} = ${total}";
fi;
exit 0;
```

Verifique um exemplo da execução desse script Shell:

```
[root@centos ~]# ./multiplica.sh 5 2
5 * 2 = 10
[root@centos ~]# ./multiplica.sh 5 2 3
5 * 2 * 3 = 30
[root@centos ~]# ./multiplica.sh 5 2 3 7
5 * 2 * 3 * 7 = 210
[root@centos ~]# ./multiplica.sh 5 2 3 7 11
5 * 2 * 3 * 7 * 11 = 2310
[root@centos ~]# ./multiplica.sh 5 2 3 7 11 123
5 * 2 * 3 * 7 * 11 * 123 = 284130
[root@centos ~]#
```

Figura 7.5

A limitação desse método para processamento de parâmetros é percebida quando precisamos lidar com parâmetros que possuem argumentos, algo como -n <valor> -t <tempo>. Utilizar as variáveis de parâmetros para lidar com esse tipo de situação é um pouco complicado e exige bastante programação. O melhor é utilizar o comando getopt que lida com essa situação perfeitamente. Sua sintaxe é:

`getopts <opções> <variável>`

Em opções informamos os parâmetros que desejamos receber. Caso o parâmetro precise de um argumento extra (um valor), devemos inserir o símbolo : logo após o nome do parâmetro. Acompanhe um exemplo:

'sd:eh:'

Neste exemplo temos que os parâmetros "s" e "e" não solicitam um argumento extra, já os parâmetros "d" e "h" requerem o argumento extra. Existe a limitação de apenas uma letra por parâmetro.

Assim que um parâmetro é encontrado, a variável informada na chamada de getopt, isto é, <variável> recebe o nome do parâmetro e caso exista um argumento extra, a variável OPTARG recebe esse valor.

Getopts exige que os parâmetros passados sejam iniciados com o sinal "-", por exemplo, -d <valor> -e -h <valor>.

Getopts é processado até que todos os parâmetros passados na linha de comando sejam processados.

Veja um exemplo de sua utilização:

```
#!/bin/bash
if [ $# -le 1 ]; then
    echo "Sintaxe: `basename $0` [-n nome] [-e email] [-t titulo]";
else
    while getopts 'n:e:t:' PARAM; do
        case $PARAM in
            n) $NOME=$OPTARG;;
            e) $EMAIL=$OPTARG;;
            t) $TITULO=$OPTARG;;
        esac;
    done;
    echo "Enviar mensagem para ${NOME} no e-mail ${EMAIL} com o título
${TITULO}";
fi;

exit 0;
```

7.3.4.1 - Shift

O comando shift presente no Shell é utilizado para fazer a rolagem dos parâmetros, ou seja, toda vez que esse comando é executado, um ou mais parâmetros são descartados, sempre da esquerda para a direita, conforme a opção informada no comando. A sintaxe do comando é:

Shift <deslocamento>

Se <deslocamento> não for informado, o comando assume o valor 1 por padrão. Cada parâmetro deslocado é perdido, ou seja, seu valor não pode ser mais recuperado. O deslocamento de 1, por exemplo, faz com que o parâmetro \$1 seja colocado em \$0, \$2 em \$1 e assim sucessivamente.

Caso o deslocamento informado seja maior que os parâmetros existentes, nada é feito. Veja um exemplo:

```
#!/bin/bash  
echo "$# Parâmetros: $*"  
echo "removendo o primeiro parâmetro"  
shift  
echo "$# Parâmetros: $*"  
echo "removendo todos os parâmetros exceto o último"  
num=$[ $# - 1 ]  
shift $num  
echo "$# Parâmetros: $*"
```

7.3.5 - Redirecionamento

Sempre que precisamos capturar alguma informação ou exibir o resultado de um processo qualquer, o Shell utiliza os dispositivos de entrada e saída padrão (stdin e stdout respectivamente).

Para alterar esse padrão, podemos utilizar os caracteres <, > e >>. É possível ainda redirecionar a saída de erros. Veja como utilizar cada controle:

Programa < arquivo	Utiliza o arquivo informado como entrada.
Programa > arquivo	Utiliza o arquivo informado como saída. Se o arquivo existir, será sobreescrito, ou seja, perderá seu conteúdo anterior.
Programa >> arquivo	Utiliza o arquivo informado como saída, porém o seu conteúdo atual não é permitido, pois a saída é acrescentada ao final dele.
Programa 2> arquivo	Envia mensagens de erro ao arquivo informado.
Programa &> arquivo	Utiliza o arquivo informado para saídas e erros.
Programa 1>&2	Envia a saída padrão para a saída de erro.
Programa 2>&1	Envia a saída de erro para a saída padrão.

Exemplos:

```
# ./imprimetabela.sh ./tabela2010.txt > resultado.txt
```

```
# ./imprimetabela.sh ./tabela2010.txt >> resultado.txt  
# ./imprimetabela.sh ./tabela2010.txt 2>resultado.txt 1>&2  
# ./imprimetabela.sh ./tabelaxxx.txt > resultado.txt 2>/dev/null
```

O primeiro exemplo envia a saída do programa para o arquivo resultado.txt, que será criado caso não exista e sobreescrito se existir.

O segundo envia a saída do programa ao arquivo resultado.txt, que será criado caso não exista ou terá o seu conteúdo atual concatenado com a saída produzida pelo programa.

O terceiro exemplo envia tanto os erros como a saída do programa para o arquivo resultado.txt, que será criado caso não exista e sobreescrito se existir.

O último exemplo envia a saída do programa para o arquivo resultado.txt, que será criado caso não exista e sobreescrito se existir, e os erros encontrados para /dev/null (esse dispositivo não faz nada, e é útil para ignorarmos mensagens de erro que seriam normalmente exibidas na tela do usuário).

7.3.6 - Comandos Condicionais e Operadores

Em qualquer linguagem é necessário que existam estruturas de controle para gerenciamento de fluxo de decisão. No Shell temos os comandos condicionais if e case.

7.3.6.1 - Comando If

Tem por finalidade permitir que um grupo de comandos seja executado conforme uma determinada condição. Sua sintaxe é:

```
If <condição>; then  
    <comandos se condição for verdadeira>  
else  
    <comandos se condição for false>  
fi
```

O bloco else é opcional, ou seja, podemos utilizar a forma curta do comando apenas com um bloco de comandos se a condição informada for verdadeira.

A condição é o resultado de algum comando executado. Os comandos retornam 0 quando executados corretamente; caso contrário, algum número diferente de zero. Temos que se a condição for zero, o bloco pertencente a then será executado; senão, o bloco de else (se existir) será executado.

Em geral, o que desejamos no comando if é avaliar uma expressão, uma variável e não exatamente um comando. Então, como fazer?

Para avaliar uma expressão temos o comando do Shell test, cujo formato é:

Test <expressão>

Ou

[<expressão>]

Exemplos:

```
# a="10";b="15";test "$a" = "$b";echo $?
```

```
# a="10";b="15";teste "$a" != "$b";echo $?
```

No primeiro exemplo teremos como saída 1 (falso) e no segundo, 0 (verdadeiro).

Os operadores = e !=, igual e diferente, respectivamente, só podem ser utilizados para textos, não funcionando corretamente com números (10 é considerado diferente de 010). Para operações aritméticas (somente com inteiros) temos um grupo específico de operadores. Os operadores disponíveis são:

Operadores de texto

-n texto	O texto não é vazio, isto é, seu tamanho é maior que zero.
Texto	Equivalente a -n texto.
-z texto	O texto é vazio, isto é, seu tamanho é zero.
Texto1 = texto2	Texto1 é igual a texto2.
Texto1 != texto2	Texto1 e texto2 são diferentes.

Operadores aritméticos (somente inteiros)

Num1 -eq num2	Num1 e num2 são iguais.
Num1 -ne num2	num1 e num2 são diferentes.
Num1 -lt num2	num1 é menor que num2.
Num1 -le num2	num1 é menor ou igual a num2.
Num1 -gt num2	num1 é maior que num2.
Num1 -ge num2	num1 é maior ou igual a num2.

Operadores de arquivos

-b arquivo	O arquivo é um arquivo de bloco (/dev/hda, por exemplo).
-c arquivo	O arquivo é um arquivo especial de caractere (/dev/tty2).
-d arquivo	O arquivo é um diretório.

-e arquivo	O arquivo existe.
-f arquivo	O arquivo existe e é do tipo comum.
-s arquivo	O arquivo existe e não está vazio.
-h arquivo	O arquivo é um link simbólico.
-p arquivo	O arquivo é um named pipe (fifo, lilo).
-S arquivo	O arquivo é um socket.
-r arquivo	O arquivo tem permissão de leitura para o usuário atual.
-w arquivo	O arquivo tem permissão de escrita para o usuário atual.
-x arquivo	O arquivo tem permissão de execução para o usuário atual.
-O arquivo	O arquivo pertence ao usuário atual.
-G arquivo	O arquivo pertence ao grupo do usuário atual.
Arquivo1 -ef Arquivo2	Arquivo1 e arquivo2 têm os mesmos números de device e inode.
Arquivo1 -nt Arquivo2	Arquivo1 é mais novo que Arquivo2 (data da última modificação).
Arquivo1 -ot Arquivo2	Arquivo1 é mais velho que arquivo2.

Operadores lógicos

- a Operador lógico E (and).
- o Operador lógico OU (or).
- ! Operador lógico de negação (not).

Veja um exemplo:

```
#!/bin/bash
read -p "informe o nome de um arquivo: " arq
if [ -e $arq ]; then
    echo -n "${arq} existe .... ";
    if [ -d $arq ]; then
        echo "é um diretório"
    else
        echo
    fi
else
    echo "${arq} não existe!"
fi

echo -e "\n"
read -p "Informe um número inteiro " n1
read -p "Informe um segundo número " n2
read -p "Informe mais um número " n3
if [ $n1 -le $n2 ]; then
```

```
if [ $n1 -le $n3 ]; then
    menor=$n1
    if [ $n2 -le $n3 ]; then
        inter=$n2
        maior=$n3
    else
        inter=$n3
        maior=$n2
    fi
else
    if [ $n3 -le $n2 ]; then
        menor=$n3
        inter=$n1
        maior=$n2
    else
        menor=$n1
        inter=$n3
        maior=$n2
    fi
fi
else
    if [ $n1 - le $n3 ]; then
        menor=$n2
        inter=$n1
        maior=$n3
    else
        if [ $n3 -le $n2 ]; then
            maior=$n1
            inter=$n2
            menor=$n3
        else
            maior=$n1
            inter=$n3
            menor=$n2
        fi
    fi
fi
echo -e "\e[34m"
printf "%d <= %d <= %d" $menor $inter $maior
echo -e "\e[0m"
```

7.3.6.2 - Comando Case

É muito útil quando queremos testar um número finito de possibilidades. Sua sintaxe é:

Case <valor> in

Opção 1) comandos

;;

Opção 2) comandos

;;

```
...
*)      comandos
;;
esac
```

Case compara o conteúdo do valor informado (um parâmetro ou uma variável) com as opções fornecidas. Caso encontre uma opção que seja igual, os comandos relacionados a essa opção são executados até que seja encontrado `;;`. Um exemplo da utilização do comando case são os scripts de inicialização de processos. Veja um exemplo:

```
#!/bin/bash

case "$1" in
start) echo "iniciando..."
        start
        ;;
stop) echo "Parando..."
       stop
       ;;
restart) echo "Parando..."
          stop
          echo "Iniciando..."
          start
          ;;
*)     echo "Opção não suportada..."
        ;;
esac
```

7.3.7 - Miscelâneas

7.3.7.1 - Let

Permite realizar algumas operações aritméticas, tais como somas, subtrações, multiplicações, divisões e comparações aritméticas.

Sua sintaxe é:

Let <expressão>
Ou
((<expressão>))'

Alguns exemplos:

```
# num=1  
# let num++  
# echo $num  
# let num—  
# echo $num  
# (( num++ ))  
# echo $num
```

Note que tanto em let <expressão> quanto em ((<expressão>)) não utilizamos \$ para referenciar a variável.

7.3.7.2 - Decisão com && e ||

Quando precisamos executar comandos condicionais curtos, isto é, onde só há um bloco para executar, sem um bloco alternativo (o else do comando if), podemos utilizar os operadores && e ||.

Ambos têm a seguinte sintaxe:

```
Comando1 && comando2  
Comando1 || comando2
```

O operador && executa comando2 se e somente se comando1 retornar verdadeiro (zero).

O operador || executa comando2 se e somente se comando1 retornar falso (qualquer valor diferente de zero).

Veja dois exemplos:

```
# [ -e ./backup ] || mkdir backup  
# [ -d ./backup ] && ls -l backup
```

No primeiro exemplo o diretório backup é criado (mkdir) caso não exista um arquivo ou diretório com este nome.

O segundo exemplo lista o conteúdo do diretório backup caso ele exista e seja realmente um diretório.

7.3.7.3 - Listas

Uma lista é um grupo de comandos agrupados por chaves ou parênteses.

```
# echo "Nível do Shell atual: ${SHLVL}"  
# [ -d ./backup ] && {  
    echo "executando com chaves...Conteúdo de ./backup: "  
    ls -lt ./backup  
    echo "Nível Atual: ${SHLVL}"  
}  
# [ -d ./backup ] && (  
    echo "executando com parênteses...Conteúdo de ./backup: "  
    ls -lt ./backup  
    echo "Nível Atual: ${SHLVL}"  
}
```

7.3.8 - Comandos de Controle

Executam grupos de comandos de forma repetitiva e controlada. Os comandos de controle disponíveis são:

- » While
- » For
- » Until
- » Select

7.3.8.1 - While

Esse comando permite executar um grupo de comandos enquanto a condição informada for verdadeira, saindo do laço quando for falsa. O grupo de comandos pertencentes ao while é executado zero ou mais vezes. Sua sintaxe é:

While <condição>; do

 Comandos

Done

O argumento <condição> tem o mesmo formato apresentado para o comando if. Veja um exemplo:

```
#!/bin/bash
num=1
while [ $num -ne 0 ]; do
    read -p "Informe um número(0=Sair) " num
    [ $num -ne 0 ] && echo "Quadrado de ${num} = " ${(( $num * $num ))}
done
echo "[FIM]"
```

```
[root@centos ~]# ./quadrado.sh
Informe um número(0=sair) 10
Quadrado de 10 = 100
Informe um número(0=sair) 13
Quadrado de 13 = 169
Informe um número(0=sair) 17
Quadrado de 17 = 289
Informe um número(0=sair) 193
Quadrado de 193 = 37249
Informe um número(0=sair) 0
[FIM]
```

Figura 7.6

7.3.8.2 - Until

O comando until é muito semelhante ao while. O detalhe está no fato de que o while executa o bloco de comandos enquanto a condição for verdadeira, e until executa enquanto for falsa, ou seja, quando a condição se tornar verdadeira, a repetição é encerrada. Sua sintaxe é:

Until <condição>; do

Comandos

done

Observe o exemplo mostrado no item anterior (while) refeito para usar until.

```
#!/bin/bash
num=1
until [ $num -eq 0 ]; do
    read -p "Informe um número(0=Sair) " num
    [ $num -ne 0 ] && echo "Quadrado de ${num} = " ${(( $num * $num ))}
done
echo "[FIM]"
```

7.3.8.3 - For

É utilizado quando desejamos executar um grupo de comandos determinado número de vezes, sendo equivalente ao while com um contador. Para executar o comando for, precisamos entregar uma lista que será processada elemento por elemento. A sintaxe é:

For <variável> in <lista>; do

Comandos

Done

Cada item da lista é associado à variável e o grupo de comandos é executado; quando a lista termina, o laço (loop) do comando for é encerrado. Desta forma, teremos tantas iterações quantos forem os elementos da lista.

Veja um exemplo:

```
# for num in 0 1 2 3 4 5 6 7 8 9 10; do echo $num; done
```

O resultado será a sequência de 0 a 10:

```
# for arq in $(ls);do echo $arq; done
```

Neste exemplo \$arq receberá os nomes dos arquivos no diretório em que o script está sendo executado.

Para gerar uma sequência de números, utilize o programa seq, cuja sintaxe é:

Seq <inicial> <incremento> <final>

Ou

Seq <inicial> <final>

Ou

Seq <final>

Dica

No primeiro formato definimos o incremento, que pode ser tanto positivo quanto negativo. Utilizamos um incremento negativo para contagem regressiva.

No segundo formato, o incremento é definido como 1 e só aceita-se contagem progressiva.

O terceiro formato assume que o valor inicial é 1 e o incremento também é 1.

Assim, para gerar uma lista de 1 a 10:

```
for num in $(seq 1 10);do echo $num; done
```

Para gerar uma sequência de 10 a 1:

```
for num in $(seq 10 -1 1);do echo $num; done
```

Outro formato para for é:

```
for((expressão1; expressão2; expressão3)); do
```

Comandos

Done

Este formato é muito útil para um laço com controle de contadores. O conteúdo de expressão1 é avaliado na primeira execução do laço (primeira iteração). O conteúdo de expressão2 é avaliado repetidamente, isto é, em cada iteração. Quando expressão2 retornar 0, o laço é interrompido, ou seja, expressão2 é o controle de repetição. Enquanto expressão2 retornar um valor diferente de zero, isto é, verdadeiro, a lista de comandos é executada e em seguida o conteúdo de expressão3 é avaliado. Expressão3 geralmente contém o incrementador do contador do laço.

Veja um exemplo:

```
for((num=1;$num<=15;num=$[ num+1 ]); do
```

```
echo $num
```

```
done
```

O resultado será a lista de números ímpares entre 1 e 15.

7.3.8.4 - Break e Continue

Os comandos break e continue são utilizados dentro das estruturas de controle while, until e for.

Break tem por objetivo encerrar a execução do laço, enquanto continue faz com que a próxima iteração seja executada.

Veja um exemplo:

```
For((num=0;$num<=20;num++)); do  
    [ $num -eq 17 ] && break  
    [ $(( $num % 2 )) -eq 0 ] && continue  
    echo $num  
done
```

Neste exemplo somente os números ímpares serão exibidos e quando chegar a 17, o laço será interrompido.

7.3.8.5 - Select

O comando select é um controlador especial. Ele monta um menu de opções conforme a lista definida e solicita ao usuário que informe a opção desejada. Uma vez escolhida a opção, a lista de comandos é executada e o menu é novamente exibido. Sua sintaxe é:

```
Select <variável> in <lista de opções>; do
```

```
    Comandos
```

```
Done
```

Veja um exemplo de sua utilização:

```
Select arq in $(ls sair; do  
    [ $arq = sair ] && break  
    Echo $(file $arq)  
Done
```

```

1) bin          4) include    7) libexec   10) share     13) X11R6
2) etc          5) kerberos   8) local      11) src       14) sair
3) games        6) lib         9) sbin      12) tmp

#? 4
include: directory
#? 14
[root@centOS usr]# ■

```

Figura 7.7

Outro exemplo:

Select opc in Instalar Atualizar Remover Sair; do

Case \$opc in

Instalar) echo "...Instalando o aplicativo..."

;;

Atualizar) echo "...Buscando atualizações..."

;;

Remover) echo "...Executando processo de remoção..."

;;

Sair) break

;;

esac

done

```

1) Instalar
2) Atualizar
3) Remover
4) Sair
#? 1
..Instalando Aplicativo..
#? 2
...Buscando Atualizações.;...
#? 3
..Executando processo de remoção..
#? 4
[Fim]

```

Figura 7.8

7.3.9 - Trap

Esta é a forma como se podem capturar sinais transmitidos a um processo e então executar uma ou mais ações conforme o sinal detectado. Mas você deve estar perguntando o que é, afinal de contas, um sinal.

7.3.9.1 - Sinal

Sinal é a forma como se pode enviar um aviso para um determinado processo em execução. É possível sinalizar várias situações, e cada sinal representa uma situação. Os sinais mais comuns são:

HUP	Hung-Up.
INT	^C, Interrupção.
KILL	Parada (morte) - não pode ser capturado ou ignorado.
TERM	Terminação por programa.

Para uma lista completa de sinais, digite kill -l no Shell do Linux.

1) SIGHUP	2) SIGINT	3) SIGQUIT	4) SIGILL
5) SIGTRAP	6) SIGABRT	7) SIGBUS	8) SIGFPE
9) SIGKILL	10) SIGUSR1	11) SIGSEGV	12) SIGUSR2
13) SIGPIPE	14) SIGALRM	15) SIGTERM	16) SIGSTKFLT
17) SIGCHLD	18) SIGCONT	19) SIGSTOP	20) SIGTSTP
21) SIGTTIN	22) SIGTTOUT	23) SIGURG	24) SIGXCPU
25) SIGXFSZ	26) SIGVTALRM	27) SIGPROF	28) SIGWINCH
29) SIGIO	30) SIGPWR	31) SIGSYS	34) SIGRTMIN
35) SIGRTMIN+1	36) SIGRTMIN+2	37) SIGRTMIN+3	38) SIGRTMIN+4
39) SIGRTMIN+5	40) SIGRTMIN+6	41) SIGRTMIN+7	42) SIGRTMIN+8
43) SIGRTMIN+9	44) SIGRTMIN+10	45) SIGRTMIN+11	46) SIGRTMIN+12
47) SIGRTMIN+13	48) SIGRTMIN+14	49) SIGRTMIN+15	50) SIGRTMAX-14
51) SIGRTMAX-13	52) SIGRTMAX-12	53) SIGRTMAX-11	54) SIGRTMAX-10
55) SIGRTMAX-9	56) SIGRTMAX-8	57) SIGRTMAX-7	58) SIGRTMAX-6
59) SIGRTMAX-5	60) SIGRTMAX-4	61) SIGRTMAX-3	62) SIGRTMAX-2
63) SIGRTMAX-1	64) SIGRTMAX		

Figura 7.9

7.3.9.2 - Comando Trap

Esse comando deve ser utilizado para gerenciar os sinais recebidos pelo script. Podemos definir uma ação específica para determinados sinais, ignorar um sinal ou então voltar à situação padrão para manipulação de sinais.

A sintaxe do comando trap é:

Trap '<lista de comandos>' <lista de sinais>

Para ignorar um sinal, devemos deixar a lista de comandos vazia:

```
Trap '' 2
```

Com isso ignoramos o sinal INT ou ^C.

Para voltar ao comportamento padrão, informamos apenas a lista de sinais:

```
Trap 2
```

Acompanhe um exemplo de ação durante a captura de sinais:

```
#!/bin/bash
```

```
Trap 'echo -e "\n\e[31m**fim**\n\e[0m";exit 1' 2 3 15
```

```
While true; do
```

```
    Echo -n "."
done
```

Neste exemplo o script, ao receber INT QUIT ou TERM, exibe a mensagem ** fim ** e encerra o processamento.

A captura e o processamento de sinais são úteis em processos críticos nos quais uma parada abrupta pode trazer problemas de instabilidade, ou então quando desejamos, antes de encerrar o script, realizar tarefas de limpeza, isto é, remoção de arquivos temporários, liberação de portas, limpeza de memória etc.

7.3.10 - Alias

No Shell é possível definir apelidos (alias) para comandos. O objetivo é economizar digitação dentro do script. Para criar os apelidos, utilizamos o comando alias, cuja sintaxe é:

```
Alias [-p] [nome=comando]
```

O parâmetro -p exibe os apelidos definidos.

Exemplos:

```
# alias l='ls -l'
```

```
# alias rm='rm -i'
```

```
# alias d='echo ${PWD}'
```

Para remover um alias, devemos utilizar o comando unalias:

```
Unalias [-a] [nome1 nome2 ...]
```

O parâmetro -a indica que todos os alias devem ser removidos.

Se você tem um grupo de apelidos que deseja sempre ter disponível, basta editar o arquivo `~/.bash_aliases` e colocar os comandos alias dentro dele. Toda vez que você entrar no sistema esse arquivo será executado, disponibilizando seus apelidos favoritos.

7.3.11 - Funções

Função é um agrupamento de comandos que podem ser executados de forma repetitiva em qualquer lugar do script. Uma função deve ter um nome e um corpo com os comandos a ela pertencentes. A função pode ou não trabalhar com parâmetros e retornar um resultado de sua execução.

A sintaxe para declaração de uma função é:

```
Function <nome> {  
    Comandos da função  
    [Return <retorno>]  
}  
  
Ou  
  
<nome_função>() {  
    Comandos  
    [Return <retorno>]  
}
```

O retorno é opcional e pode estar em qualquer lugar da função. Quando o comando `return` é encontrado, nenhum outro comando da função é executado e o processamento da função é encerrado. O valor retornado pela função pode ser recuperado por `$?`.

Veja um exemplo:

```
# Function quadrado {  
    return $(( $1 * $1 ));  
}  
  
# quadrado 10  
# echo $?
```

A manipulação de parâmetros recebidos pela função é idêntica à mostrada no tópico sobre parâmetros de Shell scripts.

Somente valores numéricos podem ser retornados por uma função e o maior valor possível de ser retornado é 256.

```
function infos() {
    echo -e "\nInformações do servidor: " ; uname -a
    echo -e "\nUsuários ativos: " ; w -h
    echo -e "\ndata atual: " ; date
    echo -e "\nTempo de vida do servidor: " ; uptime
    echo -e "\nStatus da memória: " ; free
    echo -e "\nArquivo do sistema:"; df -h
}
function ll () {
    clear;
    tput cup 0 0;
    ls --color=auto -F --color=always -lhFrt;
    tput cup 40 0;
}
```

Exercícios

1. Defina Shell.
2. Qual deve ser a primeira linha de Shell script?
3. O que faz o comando echo?
4. Explique o comando printf.
5. Cite três variáveis predefinidas no Shell.
6. Para que serve o comando readonly?
7. Como remover uma variável do Shell?
8. O que faz o comando export?
9. Qual o resultado de echo ~?
10. Qual a função de \$0, \$1, \$# e \$? ?
11. Explique o comando “programa 2>&1 /dev/null”.
12. Para que serve o comando IF?
13. Dê o resultado do seguinte código:

```
#!/bin/bash
valor=10
limite=15
IF [ $valor -lt $limite ]; then
    Echo "O valor é menor que o limite"
Else
    Echo "O valor é maior ou igual ao limite"

Fi
Printf "Limite %3d , Valor %3d" $limite $valor
```

14. Para que serve o comando let?
15. Qual a diferença entre while e until?
16. Dê um exemplo de for para gerar um contador.
17. O que é sinal?
18. Explique o comando trap.
19. Como definir um apelido?
20. O que é função?
21. Qual o limite para o retorno de uma função?

8

Administração de Rede

8.1 - As Redes TCP/IP

No dia a dia convivemos com programas que realizam funções e serviços de rede. Claro que, na maioria das vezes, do ponto de vista de usuário, mas você provavelmente já está familiarizado com alguma ferramenta de rede, seja um dispositivo de rede wireless ou um programa cliente de correio eletrônico.

Assim como quase tudo em um sistema, há uma complexidade maior por trás de um processo aparentemente simples para um usuário final quando utilizamos ferramentas comuns de rede, como um cliente de ftp ou um navegador da Internet.

Compreender ao menos o básico sobre redes TCP/IP (Transmission Control Protocol/Internet Protocol) pode ser crucial na hora de configurar um sistema para utilizar esses recursos e, principalmente, quando as coisas não estão funcionando como deveriam, pois a compreensão pode auxiliar muito na resolução de problemas.

Este item apresenta uma visão geral das principais características do protocolo TCP/IP e de redes de computadores.

8.1.1 - Pilha de Rede

Em termos básicos o principal objetivo das redes é a transferência de informações de um programa que está sendo executado em um determinado computador a outro programa executado em outro computador ou host, como quando abrimos um navegador da Internet, por exemplo, o Mozilla Firefox. O seu objetivo é enviar uma solicitação a um servidor Web e ao digitarmos o endereço do servidor que desejamos acessar, ele se conecta a ele e solicita os documentos relativos àquela página, ou seja, troca informações com aquele computador remoto. Isso não é regra, pois existem programas que podem usar a rede para conversar entre si ou com outros programas, mesmo que estejam rodando em um único computador.

Quando instalamos um programa e o seu banco de dados na mesma máquina, provavelmente o programa usa a rede para acessar o banco de dados e esse computador, além de servidor (provendo acesso ao banco de dados), também é um cliente (quando utilizamos o programa que se conecta a esse banco).

Esse processo de troca de informações pode acontecer dezenas, centenas, milhares, ou seja, quantas vezes forem necessárias, então imagine se cada programa tivesse a responsabilidade de controlar o hardware diretamente. Isso seria impraticável e logo esses sistemas entrariam em conflito, pois em determinado momento um programa iria interferir em outro, ocasionando falhas no funcionamento. Para garantir a segurança no processo e padronizar a forma como os programas conversam em redes, foram necessários alguns padrões de controle do acesso à rede.

A solução para este problema está na pilha de rede. É uma coleção de camadas que são módulos de softwares de pequeno porte, que funcionam em série, ou seja, cada camada de software conversa com outras duas camadas. Uma dessas duas camadas pode ser o hardware da placa de rede ou o próprio ser humano para a camada de mais baixo nível e mais alto respectivamente.

As aplicações de rede dos clientes e servidores estão no nível mais alto da pilha de rede. Elas se comunicam com a camada abaixo dela, que conversa com aquela mais abaixo e assim sucessivamente até atingir a camada mais inferior. Quando atinge a camada de mais baixo nível, é o ponto em que os dados saem do computador e são efetivamente enviados através da rede até chegar ao host de destino. No destino o processo acontece de forma inversa para levar a informação até a camada de Aplicação. Os dados chegam pela camada de mais baixo nível e vão sendo enviados de uma camada para outra até atingirem o nível mais alto, entregando as informações ao aplicativo, que pode ou não enviar uma resposta. Assim, os dados são transmitidos pela rede, subindo e descendo a pilha de rede. Na Figura 8.1 podemos ver um diagrama deste fluxo.

A definição de pilha de rede tornou-se uma arquitetura formal para conectar computadores e é chamada de modelo OSI (Open Systems Interconnection). Esse modelo prevê que as redes entre computadores devem conter sete camadas, de tal forma que se obtenham camadas de abstração. Assim, cada protocolo de rede necessita apenas programar uma funcionalidade específica a uma determinada camada.

No TCP/IP, na prática temos um modelo baseado no OSI, mas com menos camadas. No modelo TCP/IP o hardware não é considerado, pois ele não se estende até a camada Física. Ele mescla parcialmente a camada de Sessão com toda a camada de Transporte em apenas uma, e o restante da camada de Sessão com as camadas de Apresentação e Aplicação em mais uma única camada.

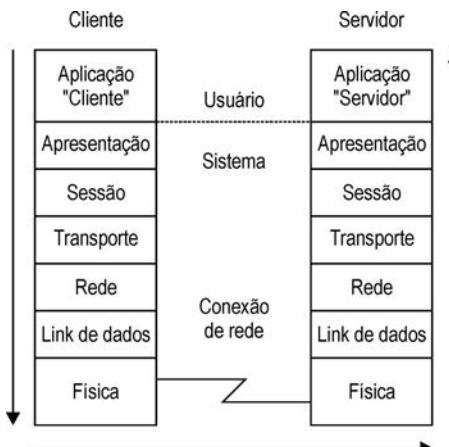


Figura 8.1 - Diagrama modelo OSI de fluxo básico de informações de um cliente para um servidor.

	OSI	TCP/IP
7	Aplicação	Aplicação
6	Apresentação	
5	Sessão	Transporte
4	Transporte	
3	Rede	Internet
2	Link de dados	Interface de rede
1	Física	

Figura 8.2 - Pilha de rede modelo OSI x TCP/IP.

8.1.2 - Endereço de Rede

Um componente vital para qualquer protocolo de rede é o endereço de rede, pois não há como interligar máquinas ou computadores sem que cada um possua uma identificação única, ao menos dentro do espaço alcançável por ela, ou seja, a rede à qual o equipamento pertence.

O TCP utiliza endereçamento do tipo IP (Internet Protocol). Os endereços IP na versão 4 são números de 32 bits, ou seja, quatro conjuntos de 8 bits, ou simplesmente 4 bytes, que normalmente são escritos como quatro números decimais separados por ponto ".", um decimal para cada byte, por exemplo, 192.168.1.103. Esses endereços IP estão divididos em dois componentes, sendo um deles o endereço da rede e o outro o endereço da máquina. Essa divisão se faz necessária para facilitar a tarefa de roteadores, que são equipamentos que enviam informações entre redes distintas.

A máscara de sub-rede, que se chama netmask, determina qual parte do endereço IP é o endereço da rede. Na máscara de rede 255.255.255.0, temos todos os bits dos três primeiros bytes em valor binário igual a 1. Isso indica que esses três primeiros bytes do endereço IP são o endereço da rede e o quarto ou último byte é o endereço da máquina. Para o endereço IP 192.168.1.254 o "192.168.1" é o endereço da rede e o "254" é o endereço da máquina. Na tabela 8.1 podemos visualizar melhor estas informações da configuração do IP para o exemplo dado.

Tabela 8.1

Campo	Endereços (Decimal)			
Endereço IP	192	168	1	254
Máscara de Rede	255	255	255	0
Endereço do Host (Máquina)	0	0	0	254
Endereço de Rede	192	168	1	0
Campo	Endereços (Binário)			
Endereço IP	11000000	10101000	00000001	11111110
Máscara de Rede	11111111	11111111	11111111	0
Endereço do Host (Máquina)	00000000	00000000	00000000	11111110
Endereço de Rede	11000000	10101000	00000001	00000000

Alternativamente o endereço IP ou de rede pode ser combinado com uma máscara de sub-rede, utilizando uma forma abreviada em que a quantidade de bits em valor binário igual a 1 na máscara de sub-rede é expressa em numeral decimal, logo após o endereço IP. Por exemplo, temos 192.168.1.0/24, o que significa o mesmo que dizer que o endereço de rede é 192.168.1.0 e a máscara é composta por 24 bits, e pode ser representada assim 255.255.255.0, ou seja, os três primeiros bytes com todos os bits em valor binário 1 representado em numeral decimal.

Para evitar conflitos dos endereços entre redes e também para organizar os endereçamentos IPs, os endereços de IP são divididos em redes com três classes principais, que são A, B e C. A partir da máscara de sub-rede, define-se uma classe de rede, em que uma rede classe A pode conter mais de 16 milhões de endereços distintos, as redes classe B podem endereçar até 65.534 endereços e a classe C apenas 254. Na Tabela 8.2 podemos ver que cada uma das três classes principais de rede suporta um intervalo finito de endereços privados que não podem ser utilizados em roteadores de Internet.

Tabela 8.2

Classe	Faixa	Máscara	Endereços Privados
A	1.0.0.0–127.255.255.255	255.0.0.0	10.0.0.0–10.255.255.255
B	128.0.0.0–191.255.255.255	255.255.0.0	172.16.0.0–172.31.255.255
C	192.0.0.0–223.255.255.255	255.255.255.0	192.168.0.0–192.168.255.255

Esses endereços privados podem ser utilizados em redes internas ou privadas sem nenhuma autorização oficial. Vale ressaltar que existem mais duas classes de redes, são as classes D e E. A primeira é uma classe especial utilizada para os endereços chamados de Multicast e a segunda é reservada para o futuro.

8.1.3 - Portas

Quando um pacote é enviado para o computador destino, esse host precisa identificar o que deve ser feito com ele. No TCP/IP as informações são transmitidas através de portas e por vezes mais de uma porta. A porta é um número de 16 bits, ou seja, os pacotes podem ser enviados por uma das 65.536 portas existentes. As portas normalmente estão associadas a um serviço/ligação específica e por convenção alguns serviços específicos possuem portas bem conhecidas que são numeradas de 1 a 1023, como o servidor Web que escuta na porta 80 e o FTP que usa a porta 21. Além destas existem diversas outras portas registradas, privadas ou dinâmicas. O arquivo /etc/services é uma lista das atribuições de portas para os servidores mais comuns.

Como usuários normalmente não precisamos nos preocupar com o uso das portas, porque os programas clientes sabem em qual porta o servidor está esperando uma conexão. Já como administradores do sistema podemos eventualmente ter de alterar parâmetros no /etc/services, como para adicionar um servidor que instalamos que não usa nenhuma porta já incluída nesse arquivo.

Segue um trecho do arquivo /etc/services do CentOS 5 como exemplo:

```

# /etc/services:
# $Id: services,v 1.42 2006/02/23 13:09:23 pknirsch Exp $
#
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, most entries here have two entries
# even if the protocol doesn't support UDP operations.
# Updated from RFC 1700, ``Assigned Numbers'' (October 1994). Not all ports
# are included, only the more common ones.
#
# The latest IANA port assignments can be gotten from
#     http://www.iana.org/assignments/port-numbers
# The Well Known Ports are those from 0 through 1023.
# The Registered Ports are those from 1024 through 49151
# The Dynamic and/or Private Ports are those from 49152 through 65535
#
# Each line describes one service, and is of the form:
#
# service-name    port/protocol    [aliases ...]    [# comment]

tcpmux          1/tcp           # TCP port service multiplexer
tcpmux          1/udp           # TCP port service multiplexer
rje             5/tcp           # Remote Job Entry
rje             5/udp           # Remote Job Entry
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
systat          11/udp          users
daytime         13/tcp
daytime         13/udp
qotd            17/tcp           quote
qotd            17/udp           quote
msp              18/tcp           # message send protocol
msp              18/udp           # message send protocol
chargen         19/tcp           ttystt source
chargen         19/udp           ttystt source
ftp-data        20/tcp
ftp-data        20/udp
# 21 is registered to ftp, but also used by fsp
ftp              21/tcp
ftp              21/udp           fsp fspd
ssh              22/tcp           # SSH Remote Login Protocol
ssh              22/udp           # SSH Remote Login Protocol
telnet           23/tcp
telnet           23/udp
# 24 - private mail system
lmtp             24/tcp           # LMTP Mail Delivery
lmtp             24/udp           # LMTP Mail Delivery
smtp             25/tcp           mail
smtp             25/udp           mail
time             37/tcp           timserver
time             37/udp           timserver
rlp              39/tcp           resource      # resource location
rlp              39/udp           resource      # resource location
nameserver       42/tcp           name         # IEN 116
nameserver       42/udp           name         # IEN 116
nicname          43/tcp           whois
nicname          43/udp           whois
tacacs           49/tcp
tacacs           49/udp           # Login Host Protocol (TACACS)
re-mail-ck       50/tcp           # Remote Mail Checking Protocol

```

```

re-mail-ck      50/udp          # Remote Mail Checking Protocol
domain         53/tcp           # name-domain server
domain         53/udp
whois++        63/tcp
whois++        63/udp
bootps         67/tcp           # BOOTP server
bootps         67/udp
bootpc         68/tcp           # BOOTP client
bootpc         68/udp
tftp            69/tcp
tftp            69/udp
gopher          70/tcp           # Internet Gopher
gopher          70/udp
netrjs-1       71/tcp           # Remote Job Service
netrjs-1       71/udp
netrjs-2       72/tcp           # Remote Job Service
netrjs-2       72/udp
netrjs-3       73/tcp           # Remote Job Service
netrjs-3       73/udp
netrjs-4       74/tcp           # Remote Job Service
netrjs-4       74/udp
finger          79/tcp
finger          79/udp
http            80/tcp           www www-http   # WorldWideWeb HTTP
http            80/udp           www www-http   # HyperText Transfer Protocol
kerberos        88/tcp           kerberos5 krb5 # Kerberos v5
kerberos        88/udp           kerberos5 krb5 # Kerberos v5
supdup          95/tcp

```

Importante ressaltar que os programas clientes também utilizam portas para originar uma solicitação a um servidor, as quais são controladas automaticamente pelo Linux, e não precisamos nos preocupar com isso. Basicamente quando um servidor responde a uma solicitação de um cliente, ele retorna pela mesma porta que o cliente se conectou, desta forma o Linux, do lado cliente, ao receber o pacote de volta, pelo número da porta sabe para qual programa entregar a resposta.

8.2 - Configuração do TCP/IP

Praticamente todas as distribuições Linux permitem que as configurações do TCP/IP sejam informadas durante a instalação do sistema operacional, no entanto nem sempre é o que desejamos fazer, ou não conseguimos fazer naquele momento por falta de driver, ou até mesmo pela inexistência de uma placa de rede. Também é muito comum as distribuições possuírem interfaces gráficas para as configurações TCP/IP. Entretanto, apesar desses agentes facilitadores criados, é muito importante entender as ferramentas em modo texto, por isso vamos estudar este tópico.

8.2.1 - Configuração do Hardware de Rede

Um grupo de trabalho em rede inclui uma série de hardwares de rede, como switches, servidores, roteadores, entre outros. Para facilitar consideramos que você tem uma rede em funcionamento, pois as possibilidades seriam tantas que se torna impossível comentar todos os tipos de arquiteturas de rede. Vamos nos concentrar na configuração de um sistema

Linux para funcionar em uma única rede. Para isso vamos localizar e instalar uma placa de rede ou network interface card (NIC) do tipo Ethernet e instalar o driver apropriado para essa placa.

Podemos verificar os dispositivos de rede que são suportados para um kernel Linux instalado, verificando a configuração dele. Para visualizar as configurações do kernel, são necessários os pacotes kernel-devel, kernel-headers e ncurses-devel. Para instalar esses pacotes podemos usar o gerenciador de pacotes yum como segue:

```
# yum install kernel-devel kernel-headers ncurses-devel
```

Após a instalação, acesse a pasta das fontes do kernel, normalmente localizada no diretório /usr/src/kernels/<Versão>, e execute o comando:

```
# make menuconfig
```

Na opção Device Drivers → Network Device Support aparece um menu com a lista de categorias de dispositivos. Entrando em cada categoria, é possível listar os dispositivos suportados, por exemplo, na opção Ethernet (10 or 100Mbit) são exibidos os dispositivos de 10Mbps e 100Mbps. Alternativamente também podemos consultar o fabricante do dispositivo, pois muitos constroem seu próprio driver para o kernel Linux. Normalmente disponibilizam os arquivos-fonte para que você mesmo compile o driver.

Ao compilar um driver embutido ou build-in no kernel Linux, ele será carregado automaticamente ao iniciar, mas se o driver for compilado como um módulo do kernel, que é o método mais comum e mais seguro para iniciantes, talvez seja necessário incluir alguns parâmetros adicionais, basicamente para carregá-lo ao iniciar. No CentOS o arquivo que provavelmente precisa ser alterado é o /etc/modprobe.conf, no qual vamos incluir a configuração de um alias, apelido, para cada dispositivo de rede apontando para o nome do módulo/driver da placa de rede.

Como exemplo de arquivo /etc/modprobe.conf, observe a primeira e a última linhas, em que temos as interfaces de rede eth0 e eth1:

```
alias eth0 e1000
alias snd-card-0 snd-intel8x0
options snd-card-0 index=0
options snd-intel8x0 index=0
alias eth1 pcnet32
```

Esse arquivo diz ao kernel que o driver a ser utilizado para o dispositivo eth0 (primeiro adaptador de rede do sistema) deve ser o e1000 e para o eth1 (segundo adaptador de rede) o driver pcnet32. Você também pode verificar a lista de módulos disponível no kernel instalado acessando a pasta /lib/modules/<Versão>/kernel/drivers/net. Procure arquivos com a extensão *.ko; o que vem antes da extensão é o nome do driver. Segue um exemplo:

```
# ls /lib/modules/2.6.18-194.17.1.el5/kernel/drivers/net/
3c59x.ko      amd811le.ko    cnic.ko      fealnx.ko      natsemi.ko      pcnet32.ko
ko          pppoe.ko      s2io.ko      slhc.ko      sungem_phy.ko   typhoon.ko
8139cp.ko     b44.ko       dl2k.ko      forcedeth.ko   ne2k-pci.ko   ppp_async.ko
```

```
ko pppox.ko      sis190.ko    slip.ko     sunhme.ko    via-rhine.ko
8139too.ko      bnx2.ko      dummy.ko    ifb.ko       netconsole.ko ppp_deflate.
ko ppp_synctty.ko sis900.ko    starfire.ko tg3.ko      via-velocity.ko 8390.ko
bnx2x.ko        e100.ko      mdio.ko     niu.ko       ppp_generic.ko qla3xxx.ko
skge.ko         sundance.ko   tlan.ko     virtio_net.ko acenic.ko    cassini.ko
epic100.ko      mii.ko       ns83820.ko  ppp_mppe.ko r8169.ko     sky2.ko
sungem.ko       tun.ko
```

Estando todas as dependências instaladas e as configurações corretas, o dispositivo deve funcionar normalmente. Para verificar se o kernel detectou o driver correto da placa de rede, podemos usar o comando dmesg para visualizar as mensagens do kernel a qualquer momento e conferir os hardwares detectados. Para melhor visualizar a saída do comando dmesg, vamos usar um outro programa para filtrar a saída, o grep. Veja um exemplo:

```
# dmesg | grep -i eth
eth0: registered as PCnet/FAST III 79C973
e1000: eth1: e1000_probe: Intel(R) PRO/1000 Network Connection
e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
eth1: link up, 100Mbps, full-duplex
```

Observe na saída do comando dmesg que o kernel reconheceu os dispositivos corretamente. A partir de agora podemos definir os parâmetros de rede para esses dispositivos.

8.2.2 - O Uso do IP Dinâmico - Cliente DHCP

DHCP é um protocolo de rede que funciona com o TCP/IP e permite que um sistema, no caso o servidor DHCP, mantenha as informações sobre propriedades importantes da rede para informar aos equipamentos conectados a ela, os clientes DHCP. Estudaremos mais sobre DHCP adiante. Neste item vamos ver como configurar um cliente DHCP. Se a sua rede utilizar DHCP, fica muito mais fácil configurar as estações de trabalho. Basicamente um programa cliente de DHCP ou simplesmente DHCP Client envia um sinal especial à rede, chamado de broadcast, solicitando um servidor DHCP, ou DHCP Server que, ao receber, retorna ao cliente as informações necessárias para que ele se conecte à rede.

Para dizer ao sistema que estamos conectados a uma rede que possui um servidor DHCP, podemos alterar o parâmetro BOOTPROTO no arquivo script de configuração da interface de rede que está conectada à rede DHCP. Esses arquivos estão localizados no diretório /etc/sysconfig/network-scripts/. O parâmetro BOOTPROTO define o tipo de protocolo utilizado. No exemplo que segue usamos o comando cat para exibir as configurações da interface eth0 - /etc/sysconfig/network-scripts/ifcfg-<interface> - conectada e configurada para uma rede DHCP:

```
# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
BOOTPROTO=dhcp
HWADDR=08:00:27:83:05:26
ONBOOT=yes
```

Observe que o BOOTPROTO está em dhcp. O restante dos parâmetros foi incluído automaticamente pelo sistema. O parâmetro ONBOOT pode ser útil, quando temos mais de uma placa de rede instalada, pois ele indica se essa interface deve ser inicializada

automaticamente no boot do sistema. Mais informações sobre os parâmetros desse arquivo podem ser encontradas em http://www.centos.org/docs/5/html/Deployment_Guide-en-US/s1-networkscripts-interfaces.html.

Alternativamente podemos utilizar a ferramenta setup ou system-config-network-tui na console do Linux. Com ele fica ainda mais simples configurar. Digite no terminal setup e uma tela é mostrada. Use as setas para navegar e ENTER para entrar em uma opção. Vamos selecionar Configuração de Rede.

```
# setup
```



Figura 8.3

Após selecionar Configuração de rede, será exibido outro menu de opções, selecione Edit Devices.



Figura 8.4

Na próxima tela selecione o dispositivo que deseja configurar. Note que são mostrados todos os dispositivos que foram reconhecidos, portanto se houver apenas uma placa, será mostrado apenas um dispositivo. No exemplo que segue, temos duas placas de rede e vamos configurar a interface eth0.



Figura 8.5

Finalmente use o ENTER para passar para o próximo campo. Siga até o campo Use DHCP e use a barra de espaço para marcar. Observe:

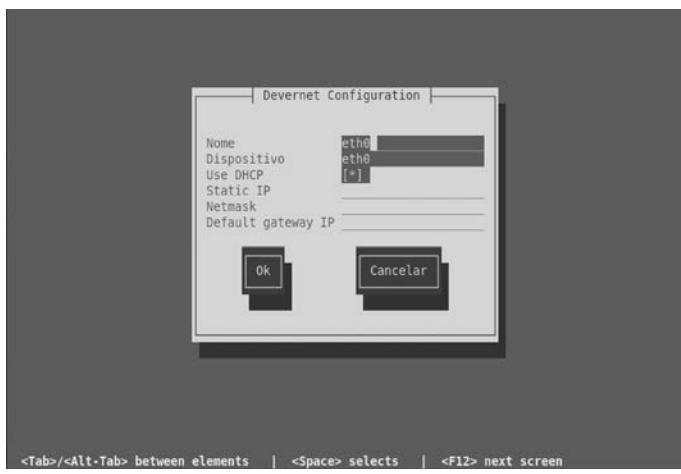


Figura 8.6

Saia, confirmando e salvando as configurações. Não se esqueça de reiniciar o serviço de rede tanto para configuração manual quanto pela ferramenta.

Para reiniciar o serviço de rede, use:

```
# service network restart
```

Para
refletir

Faça uma continha e verifique a quantidade de cliques necessários para configurar usando a ferramenta contra a parametrização diretamente no arquivo de configuração.

8.2.3 - Uso do IP Fixo

Se sua rede não usa DHCP, podemos configurar as definições de TCP/IP para usar um IP fixo. Neste caso a configuração é mais facilmente manipulada no momento da instalação ou utilizando uma ferramenta de configuração. Vamos conhecer duas formas de configurar uma rede manualmente para usar IP fixo. A primeira seria a forma universal, mais complexa, entretanto deve funcionar na maioria das distribuições e depois vamos manipular os arquivos de configuração. Nada impede de utilizar o setup novamente, mas após a leitura deste item você terá mais facilidade para encontrar as opções desejadas.

8.2.3.1 - Configuração Manual em Tempo de Execução

A primeira coisa a ser feita é definir o hostname, nome do computador. Podemos fazer isso usando o comando hostname. Se digitarmos apenas hostname seguido de ENTER em um console, será exibido o nome atual. Para definir um novo nome, use:

```
# hostname guiapratico.local
```

Agora já temos um nome na rede, então podemos ativar a interface de rede. Usaremos o comando ifconfig, que informa ao sistema a interface e o endereço IP a serem utilizados. A sintaxe básica para o comando ifconfig é:

```
# ifconfig interface [opções] [endereço]
```

Breve descrição dos parâmetros:

- » **interface:** é a interface que queremos configurar, por exemplo eth0.
- » **opções:** existem diversas opções para o ifconfig. Vamos utilizar apenas duas, sendo a opção up e a máscara de sub-rede. É recomendável ler o man ifconfig para mais informações.
- » **endereço:** é o endereço IP que será associado a essa interface.

Para ativarmos a interface de rede eth0 com o endereço IP 192.168.0.125 e máscara de sub-rede padrão 255.255.255.0, usamos:

```
# ifconfig eth0 192.168.0.125 up
```

Exemplo para uma máscara de sub-rede diferente da padrão:

```
# ifconfig eth0 netmask 255.255.255.128 192.168.0.125 up
```

Por fim é sempre prudente verificar se está tudo configurado como desejamos. Use o comando `ifconfig nome_interface` e são mostradas as configurações dessa interface. Caso seja omitido o nome da interface, será exibida a configuração de todas as interfaces de rede existentes no sistema. Segue um exemplo:

```
# ifconfig eth0
eth0      Link encap:Ethernet  Endereço de HW 08:00:27:83:05:26
          inet end.: 192.168.0.125  Bcast:192.168.0.127  Masc:255.255.255.128
          endereço inet6: fe80::a00:27ff:fe83:526/64 Escopo:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Métrica:1
          RX packets:9 errors:0 dropped:0 overruns:0 frame:0
          TX packets:108 errors:0 dropped:0 overruns:0 carrier:0
          colisões:0 txqueuelen:1000
          RX bytes:1169 (1.1 KiB)  TX bytes:21029 (20.5 KiB)
```

Observe a segunda linha da saída do comando, em que temos o endereço IP e a máscara de sub-rede que configuramos.

Agora o sistema possui acesso à rede, porém é importante que o sistema saiba qual interface de rede ele deve utilizar para enviar os dados. Isso em um computador com apenas uma interface de rede parece ser uma tarefa simples, mas não é tão simples assim no Linux, considerando que nele temos a interface local conhecida como loopback, ou localhost, que direciona o tráfego da rede para ele mesmo, e por convenção essa interface tem o endereço IP fixo 127.0.0.1. Mas não precisamos nos preocupar com essa interface loopback, pois o Linux cria e inicia automaticamente, mesmo sem estar conectado a uma rede.

Para definir as rotas que o sistema deve utilizar para acessar a rede, vamos utilizar o programa `route`.

A sintaxe básica do comando `route` é:

```
# route [add | del] [destino] [gw gateway] [[dev] interface]
```

Breve descrição dos parâmetros:

- » **add | del:** indica se estamos adicionando uma regra ou excluindo.
- » **destino:** é o local que queremos alcançar. Devemos informar o endereço IP da rede ou host específico que queremos acessar. Se o destino for 0.0.0.0, o sistema entende como rota padrão para qualquer host ou rede que não tiver uma rota específica.
- » **gw:** um gateway é um sistema conectado a uma rede que sabe como enviar pacotes para outra rede. Se você não sabe o IP do seu gateway, consulte o administrador da rede.
- » **dev:** indica que a rota deve ser aplicada à interface de rede definida na linha de comando.

Em um sistema simples temos pelo menos duas rotas. Normalmente não há necessidade de informá-las, pois o comando `route` tem a capacidade de definir algumas rotas dinâmicas

baseadas nas configurações de rede, mas como o exemplo ainda não tem um gateway definido, vamos configurar essas rotas manualmente. Acompanhe:

```
# route add 192.168.0.125 dev eth0
```

O comando anterior diz ao sistema que qualquer solicitação para o endereço IP 192.168.0.125, que é o endereço IP da interface de rede eth0, deve ser enviado para a interface eth0, ou seja, apenas estamos afirmando o que para nós é óbvio, mas para o sistema não é.

```
# route add 0.0.0.0 gw 192.168.0.1 dev eth0
```

Agora definimos que a rota padrão do sistema deve solicitar informações ao IP 192.168.0.1, que deve ser um gateway ou um roteador da rede capaz de conversar com outras redes, ou seja, para qualquer endereço IP de outra rede que desejarmos acessar, vamos enviar os dados pelo dispositivo eth0 para o gateway 192.168.0.1.

É sempre bom verificar se está tudo conforme definimos. Com o comando route podemos visualizar todas as rotas do sistema. Vamos usar a opção "-n" para mostrar endereços numéricos em vez de nomes. Observe:

```
# route -n
Tabela de Roteamento IP do Kernel
Destino      Roteador      MáscaraGen.      Opções Métrica Ref    Uso Iface
0.0.0.0      192.168.0.1   255.255.255.255 UGH     0      0          0 eth0
192.168.0.125 0.0.0.0     255.255.255.255 UH      0      0          0 eth0
```

Na tabela de rotas são mostrados no campo Opções os flags identificadores. Em seguida veja o que significa cada um deles:

- ▶ **U:** rota ativa;
- ▶ **H:** destino é um host;
- ▶ **G:** use gateway;
- ▶ **R:** rota dinâmica;
- ▶ **D:** rota dinâmica criada pelo serviço;
- ▶ **M:** rota modificada pelo serviço;
- ▶ **!:** rota negada.

Estamos quase no fim da configuração, restando apenas especificar um DNS, sistema que traduz os nomes dos computadores para endereços IP e vice-versa. Informar um DNS não é obrigatório para navegar em uma rede local, mas se desejarmos, por exemplo, acessar a Internet, ele passa a ser um item essencial. No Linux os IPs dos servidores de DNS e as informações sobre eles estão no arquivo /etc/resolv.conf.

Exemplo de arquivo /etc/resolv.conf:

```
nameserver 189.4.64.15
nameserver 189.4.64.16
```

No arquivo mostrado temos dois servidores de DNS, sendo um primário e outro secundário. A ordem de busca aos servidores é de acordo com a ordem colocada no arquivo resolv.conf.

8.2.3.2 - Configuração Manual dos Arquivos de Configuração

Aprendemos como subir uma interface de rede em tempo de execução. O ruim desse formato é que, quando reiniciamos o sistema, tudo é perdido, com exceção do DNS que definimos em um arquivo de configuração. Uma maneira de configurar é manipular os arquivos de configuração da rede.

Para definir o hostname, acesse o arquivo /etc/sysconfig/network, defina ou altere o parâmetro HOSTNAME e ajuste o conteúdo do arquivo /etc/hosts.

Exemplo de arquivo /etc/sysconfig/network:

```
NETWORKING=yes
NETWORKING_IPV6=no
HOSTNAME=guiapratico.local
```

No exemplo apresentado o hostname será guiapratico.local.

Exemplo de arquivo /etc/hosts de acordo com o /etc/sysconfig/network:

```
127.0.0.1           guiapratico.local   guiapratico
192.168.0.125       guiapratico.local   guiapratico
```

Agora podemos configurar o endereço IP fixo, a máscara de sub-rede e o gateway. Para isso devemos definir os parâmetros BOOTPROTO (tipo de protocolo), NETMASK (máscara de sub-rede), IPADDR (endereço IP) e GATEWAY no script de configuração da interface de rede, o arquivo /etc/sysconfig/network-scripts/ifcfg-eth0 para o dispositivo eth0.

Exemplo de arquivo /etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0
BOOTPROTO=none
HWADDR=08:00:27:83:05:26
ONBOOT=yes
NETMASK=255.255.255.0
IPADDR=192.168.0.125
GATEWAY=192.168.0.1
TYPE=Ethernet
```

Se precisarmos incluir rotas diferentes das que são padrões, é necessário usar o comando route, como descrito anteriormente, e adicionar as linhas de comando ao arquivo /etc/rc.local para que sejam executadas todas as vezes que o sistema reiniciar.

8.3 - Compartilhamento de Arquivos

Em pequena rede, normalmente o recurso mais utilizado é o compartilhamento de arquivos, pois com ele é possível que usuários de um computador acessem arquivos em outros computadores, como se estivessem acessando arquivos locais. O computador do usuário que está acessando os arquivos é tipificado como cliente e o computador que provê os arquivos é chamado de servidor.

O compartilhamento de arquivos necessita de um software cliente e outro servidor. Existem muitos softwares desse tipo, entretanto vamos nos concentrar em apenas dois, o NFS (Network File System), nativo em ambientes UNIX e Linux, e o Samba para utilização do protocolo SMB (Server Message Block), agora com versão superior mais conhecida como CIFS (Common Internet File System), utilizada pelo Microsoft Windows.

8.3.1 - NFS - Compartilhamento com UNIX ou Linux

Esse protocolo encaixa-se muito bem em ambientes Linux e UNIX porque é compatível com as características do sistema de arquivos, como as propriedades e permissões dos arquivos. Por essas características ele é o método de compartilhamento mais difundido em computadores UNIX e Linux.

8.3.1.1 - Configuração de um Servidor NFS

Para configurar um sistema Linux como um servidor NFS, vamos primeiramente instalar ou certificar que já está instalado o pacote do servidor NFS. Podemos usar o gerenciador de pacotes yum para fazer isso, como segue:

```
# yum install nfs-utils
```

As configurações dos diretórios que serão compartilhados, bem como as permissões de acesso, devem ser inseridas no arquivo /etc(exports no qual determinamos inclusive quais computadores podem montar essa pasta exportada.

Cada linha no /etc(exports define um diretório compartilhado. O primeiro item da linha é o caminho completo para o diretório que será compartilhado, seguido das autorizações cedidas para esse compartilhamento, incluindo opções específicas entre parênteses.

Exemplo de um arquivo /etc(exports:

```
/home maria(rw) 192.168.0.122(rw) guiapratico.local(ro)
/tmp * .local(rw)
/opt 192.168.0.0/24(ro)
```

As possibilidades no arquivo /etc(exports são muito amplas, portanto é recomendada a leitura do man exports para mais informações. Entretanto, vamos ver algumas possibilidades, conforme o exemplo dado. Segue a explicação sobre cada linha:

1. Estamos compartilhando a pasta /home para o computador com nome (hostname) maria e para o endereço IP 192.168.0.122 com permissão de leitura e escrita (rw), já para o hostname guipratico.local o acesso é de somente leitura (ro). Note que, quando o computador está no mesmo domínio, informá-lo é opcional. Veja que só informamos o domínio para o último hostname.
2. Podemos liberar o acesso a todos os computadores de um domínio, usando o curinga "*" antes do nome do domínio. Neste caso liberamos o acesso da pasta /tmp a todos os computadores do domínio "local".
3. Na última linha exportamos a pasta /opt para uma faixa de computadores em uma determinada rede usando o endereço dela seguido da máscara abreviada.

Após os ajustes necessários no arquivo /etc(exports, é preciso iniciar ou reiniciar o servidor NFS, para que as configurações sejam efetivadas. Use o comando como segue para realizar essa tarefa:

```
# service nfs restart
```

Alternativamente depois que o servidor estiver executando, você pode alterar configurações no arquivo /etc(exports e usar o comando exportfs para atualizar a lista dos compartilhamentos, sem necessidade de reinício do servidor NFS. Use o comando exportfs como segue:

```
# exportfs -av
```

8.3.1.2 - Montagem de um Compartilhamento NFS

O kernel do Linux possui suporte nativo ao NFS, portanto estará presente em praticamente todas as distribuições. Normalmente não há problemas com esse suporte, por isso vamos diretamente para a montagem do sistema de arquivo remoto em um diretório local. A montagem de sistemas de arquivos NFS remotos se dá no mesmo formato de montagem de qualquer outro sistema de arquivos, mudando somente o tipo e os parâmetros.

A sintaxe é a seguinte:

```
# mount [-t nfs] servidor:/caminho ponto_de_montagem
```

A opção "-t nfs" indica o tipo de sistemas de arquivos, "servidor" é o IP ou hostname do computador que está compartilhando os arquivos, "caminho" é a pasta compartilhada no servidor, assim como foi escrita no arquivo /etc(exports do servidor e o "ponto_de_montagem" é o diretório local em que será montado o compartilhamento. Usando o exemplo do /etc(exports usado na configuração do servidor, e considerando que o IP desse servidor é 192.168.0.254, podemos montar o compartilhamento /opt no diretório local /mnt/opt_no_meu_servidor da seguinte forma:

```
# mount -t nfs 192.168.0.254:/opt /mnt/opt_no_meu_servidor
```

Após a montagem você pode usar os arquivos como se fossem locais.

8.3.2 - Samba - Compartilhamento com Windows

Sistemas operacionais, como o Microsoft Windows, por padrão não possuem suporte para sistemas NFS. Em vez disso esses sistemas utilizam os protocolos de rede SMB ou CIFS, portanto estes são os melhores protocolos a serem usados com esses sistemas. Felizmente praticamente todas as distribuições Linux disponíveis possuem suporte a esses sistemas.

O pacote Samba não é tão simples e possui uma complexidade muito maior que o NFS porque ele precisa simular um sistema Windows em um computador Linux, complicando a simples tarefa de compartilhar arquivos. Uma das dificuldades está no nome dos arquivos. No Linux os arquivos são sensíveis à caixa, conhecidos como "case sensitive", ou seja, em um nome de arquivo letras maiúsculas são diferentes de letras minúsculas. No Windows, não são. Além destas existem muitas outras complicações, portanto vamos nos restringir a montar um servidor Samba básico. Se você deseja montar um servidor mais personalizado e seguro, consulte o site <http://www.samba.org/>.

8.3.2.1 - Configuração de um Servidor Samba Básico

Para configurar um sistema Linux como um servidor Samba, vamos primeiramente instalar ou certificar se já está instalado o pacote do servidor Samba. Podemos usar o gerenciador de pacotes yum para fazer isso, como segue:

```
# yum install samba
```

As configurações dos diretórios que serão compartilhados, bem como as permissões de acesso, devem ser inseridas no arquivo /etc/samba/smb.conf. Como falamos anteriormente, esse arquivo possui diversas configurações. Vamos editar o arquivo para configurar o servidor da forma mais simples possível sem controle de acesso por usuários, o chamado modo compartilhado que pode ser muito útil em pequenas redes.

No exemplo que segue vamos compartilhar a pasta /tmp sem senha. É preciso editar o arquivo /etc/samba/smb.conf e alterar os parâmetros seguintes:

```
[global]
workgroup = GERAL
server string = Servidor Samba Versão %v
security = share
```

O parâmetro "workgroup" indica o grupo de trabalho da rede, "server string" é o equivalente à descrição do computador no Windows e o "security" é a definição do tipo de acesso, "share" significa que não terá controle por usuários. Lembrando que este é o método mais simples e menos seguro de configurar um servidor Samba.

Incluir no final do arquivo as linhas apresentadas em seguida para especificar qual pasta deve ser compartilhada:

```
[tmp]
comment = Compartilhamento Temporário
path = /tmp
guest ok = yes
writable = yes
```

Pronto, para finalizar reinicie ou inicie o servidor Samba que o compartilhamento estará disponível para ambientes Windows. Para reiniciar, e se ainda não estiver executando, iniciar o serviço Samba, use:

```
# service smbd restart
```

8.3.2.2 - Uso e Montagem de Compartilhamentos com o Cliente Samba

O kernel do Linux possui suporte ao sistema de arquivos SMBFS e CIFS, portanto está presente em praticamente todas as distribuições. Normalmente não aparecem problemas com esse suporte no kernel, portanto precisamos apenas verificar ou instalar o pacote cliente do Samba. Podemos usar o gerenciador de pacotes yum, como segue:

```
# yum install samba-client
```

A montagem de sistemas de arquivos SMB remotos se dá no mesmo formato de montagem de qualquer outro sistema de arquivos, mudando somente o tipo e os parâmetros.

A sintaxe é a seguinte:

```
# mount [-t smbfs|cifs] //servidor/caminho ponto_de_montagem
```

A opção "-t smbfs" está praticamente obsoleta, pois só funciona com sistemas operacionais Windows até a versão 9x, ou seja, quase sempre utilizamos a opção "-t cifs" que indica o tipo de sistemas de arquivos, "servidor" é o IP ou hostname do computador que está compartilhando os arquivos, "caminho" é nome do compartilhamento no servidor SMB e o "ponto_de_montagem" é o diretório local onde será montado o compartilhamento. Para montarmos um compartilhamento sem senha com o nome "temp" no IP 192.168.0.254 no diretório local /mnt/temp_no_meu_servidor, use:

```
# mount -t cifs //192.168.0.254/temp /mnt/temp_no_meu_servidor
```

Para montar o mesmo compartilhamento, porém efetuando login como usuário "maria" e senha "difícil", use:

```
# mount -t cifs -o username=maria,password=difícil //192.168.0.254/temp /mnt/temp_no_meu_servidor
```

Após a montagem você pode usar os arquivos como se fossem locais.

Também é possível acessar compartilhamentos SMB usando um cliente samba muito parecido com o cliente ftp em modo texto. Basta invocar o smbclient em uma console do Linux, como segue:

```
# smbclient //192.168.0.254/temp
```

Após pressionar ENTER na linha de comando anterior, o sistema solicita uma senha. No caso do exemplo não é necessário informar nada, bastando pressionar ENTER novamente. Se você necessitar informar um usuário específico de nome diferente daquele que você

está usando no Linux, é preciso utilizar a opção "-U" para informar o nome do usuário, como segue no exemplo:

```
# smbclient //192.168.0.254/temp -U nicolas
```

Findo esse processo de autenticação do usuário, o smbclient abre um prompt para comandos, em que é permitido executar diversas ações, portanto digite "?" ou "help" para listar todas as opções possíveis. Seguem alguns exemplos práticos para uso.

Copiando de um computador Linux para um compartilhamento samba na rede com senha no compartilhamento de rede:

```
# smbclient //IP_do_Windows/compartilhamento -U usuario --pass senha -c "put /caminho_local/arquivo caminho_destino\arquivo;"
```

Na prática, ficaria assim para copiar o arquivo /etc/hosts da máquina Linux para a máquina 192.168.0.254 em Windows ou Linux com servidor Samba:

```
# smbclient //192.168.0.253/temp -U maria --pass dificil -c "put /etc/hosts teste_copia\hosts;"
```

Copiando da máquina Windows para a pasta atual da máquina Linux, como convidado sem senha:

```
# smbclient //IP_do_Windows/compartilhamento -U guest --pass "" -c "get arquivo;"
```

Exemplo prático para copiar o arquivo teste.txt para a pasta atual do Linux:

```
# smbclient //192.168.0.253/temp -U guest --pass "" -c "get teste.txt;"
```

Copiando da máquina Windows para uma pasta qualquer da máquina Linux, como convidado sem senha:

```
# smbclient //IP_Windows/compartilhamento -U guest --pass "" -c "get arquivo /caminho_destino\arquivo;"
```

Exemplo prático:

```
# smbclient //192.168.0.254/temp -U guest --pass "" -c "get lista-pessoas.xls /root/Desktop/lista-pessoas.xls;"
```

Para finalizar, como alternativa para enviar a senha, pode-se colocá-la ao final da linha de comando, conforme exemplo seguinte. Para o usuário "maria" temos a senha "dificil". Veja que desta forma não há necessidade do parâmetro "--pass":

```
# smbclient //192.168.0.254/temp -U maria -c "get windows.ini;" dificil
```

8.4 - Login Remoto

Para efetuar login em um sistema Linux de qualquer localidade e também para executar programas remotamente, existe uma grande variedade de servidores de login remoto. Os mais utilizados são os de Telnet e os SSH. O Telnet é bastante popular, no entanto não é seguro, pois por padrão ele envia todas as informações em texto puro, portanto a melhor alternativa atualmente é o SSH (Secure Shell), pois ele encripta todos os dados antes de enviar. O servidor mais conhecido e utilizado para login remoto via SSH é o OpenSSH. Neste item vamos estudar como configurar um servidor para aceitar conexões Telnet e SSH e também como utilizar os serviços.

8.4.1 - Configuração de um Servidor Telnet

Para configurar o sistema Linux para aceitar conexões remotas por Telnet, precisamos instalar o servidor Telnet. Primeiramente precisamos instalar ou certificar que já está instalado o pacote do servidor Telnet. Podemos usar o gerenciador de pacotes yum para fazer isso, como segue:

```
# yum install telnet-server xinetd
```

Note que estamos instalando também o xinetd, que é um Super Server (superservidor). Ele é muito útil para economizar processamento dos servidores, pois monitora portas específicas e quando há uma solicitação em determinada porta, ele executa o servidor que escuta naquela porta, assim os servidores não ficam todo o tempo em memória, somente quando estão em uso. O xinetd é muito poderoso e muito utilizado. Não vamos estudá-lo, pois é assunto para pelo menos um capítulo inteiro, no entanto mais informações podem ser encontradas em www.xinetd.org. Depois de instalarmos o servidor Telnet e o Super Server Xinetd, devemos alterar o arquivo de configuração do Xinetd que controla o servidor Telnet e fazer uma modificação muito simples, que consiste em alterar o parâmetro "disable" para "no", ou seja, vamos deixar habilitado o controle do servidor Telnet no Super Server.

Exemplo de arquivo de configuração do Super Server referente ao servidor Telnet, já com o parâmetro "disable=no" localizado em /etc/xinetd.d/telnet:

```
# default: on
# description: The telnet server serves telnet sessions; it uses \
#               unencrypted username/password pairs for authentication.
service telnet
{
    flags         = REUSE
    socket_type  = stream
    wait          = no
    user          = root
    server        = /usr/sbin/in.telnetd
    log_on_failure += USERID
    disable       = no
}
```

Agora devemos marcar na tabela de serviços para ativar o serviço telnet ao iniciar, como segue:

```
# chkconfig telnet on
```

Podemos verificar que somente o telnet padrão foi habilitado com o comando anterior usando o mesmo comando com a opção "--list". No exemplo seguinte associamos o comando grep para filtrar a saída e mostrar apenas as linhas com a palavra telnet.

```
# chkconfig --list | grep telnet
ekrb5-telnet:    não
krb5-telnet:    não
telnet:          sim
```

Finalmente podemos iniciar ou reiniciar o Super Server xinetd para que as configurações entrem em vigor, como segue:

```
# service xinetd restart
```

Finalmente o servidor Telnet está pronto! Vale ressaltar que por padrão o login com o usuário root é desabilitado por segurança, já que esse protocolo não oferece segurança. Se precisar efetuar login como root, a melhor opção é o SSH que veremos em seguida.

8.4.2 - Uso de um Cliente Telnet

Para acessar um sistema Linux em que está rodando um servidor Telnet, precisamos ter um usuário e senha no computador remoto e instalar um cliente Telnet. A maioria das distribuições Linux instala por padrão um cliente telnet, mas precisamos ter certeza de que o ambiente está apto. Vamos instalar ou certificar se já está instalado o pacote do cliente Telnet. Podemos usar o gerenciador de pacotes yum para fazer isso, como segue:

```
# yum install telnet
```

Alternativamente, se você ainda não conseguiu se livrar dele, pode se conectar a um servidor telnet através do Microsoft Windows, que por padrão vem com um cliente telnet instalado. Em Linux você será muito mais feliz, mas se mesmo assim quiser executar no Microsoft Windows, prefira utilizar o puttytel. Você pode baixá-lo gratuitamente no endereço <http://www.chiark.greenend.org.uk/~sgtatham/putty/>.

No Linux, após termos o cliente instalado, basta abrir uma sessão de terminal e executar o programa telnet na linha de comando, seguido do número de IP ou hostname do servidor de Telnet, que em seguida serão solicitados o usuário e a senha, como podemos ver em seguida:

```
# telnet 192.168.1.254
Trying 192.168.1.254...
Connected to 192.168.1.254.
```

```
Escape character is '^]'.
CentOS release 5.4 (Final)
Kernel 2.6.18-194.17.1.el5 on an i686
login:
```

Após o login você pode executar comandos do shell Linux como se estivesse no computador remoto.

8.4.3 - Configuração de um Servidor OpenSSH

Para configurar o sistema Linux para aceitar conexões remotas por SSH, vamos utilizar o servidor OpenSSH. Primeiramente precisamos instalar ou certificar que já está instalado o pacote do servidor OpenSSH, mesmo sabendo que a grande maioria das distribuições Linux por padrão instala esse servidor em função de ser uma ferramenta muito poderosa e segura, ideal para manutenção remota. Podemos usar o gerenciador de pacotes yum para fazer isso, como segue:

```
# yum install openssh-server
```

Normalmente nenhuma configuração precisa ser realizada após a instalação para termos um servidor ssh funcional. Vamos apenas citar um item de segurança importante que é o fato de a configuração padrão do OpenSSH permitir login de computador remoto com o usuário root. Essa configuração não é muito indicada para servidores, pois mesmo havendo segurança na conexão, se alguém descobrir a senha de outra forma, pode invadir o sistema e causar grandes estragos. Portanto, é prudente desligar a permissão de acesso com o root pela conexão SSH.

As configurações relativas ao servidor OpenSSH estão no arquivo /etc/ssh/sshd_config. Devemos editá-lo e colocar o parâmetro "PermitRootLogin" em "no".

Trecho de exemplo de um arquivo /etc/ssh/sshd_config, com o PermitRootLogin desligado:

```
...
# Authentication:
#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
...
```

Para marcar na tabela de serviços a ativação do serviço SSH ao iniciar, use:

```
# chkconfig sshd on
```

Agora podemos iniciar ou reiniciar o servidor OpenSSH, para que as configurações entrem em vigor, como segue:

```
# service sshd restart
```

8.4.4 - Uso de Clientes SSH

Para acessar um sistema Linux remoto em que está rodando um servidor ssh, precisamos ter um usuário e senha no computador remoto e ter instalado ou instalar pelo menos um cliente ssh. Na maioria das distribuições Linux um cliente ssh é instalado por padrão, mas precisamos ter certeza de que o ambiente está pronto. Vamos instalar ou certificar se já está instalado o pacote do cliente OpenSSH. Podemos usar o gerenciador de pacotes yum para fazer isso, como segue:

```
# yum install openssh-clients
```

Alternativamente, você pode conectar-se a um servidor SSH pelo Microsoft Windows, que por padrão não tem um cliente SSH instalado, mas se mesmo assim você quiser executar no Microsoft Windows, utilize o putty. Pode baixá-lo gratuitamente no endereço <http://www.chiark.greenend.org.uk/~sgtatham/putty/>.

No Linux, após termos o cliente instalado, podemos abrir uma sessão de terminal e executar o programa ssh na linha de comando, seguido do número de IP ou hostname do servidor de SSH. A sintaxe básica para o comando ssh é:

```
# ssh [opções] [usuario@]hostname [comando]
```

Breve descrição dos parâmetros:

- » **opções:** existem diversas opções para o programa ssh. Uma opção muito utilizada é a "-l" (Letra L minúscula) que indica o usuário para login no servidor. Para mais informações, consulte o man ssh.
- » **usuario@hostname:** hostname é o IP ou nome da máquina que queremos acessar. O usuário seguido do símbolo "@" é opcional e substitui a opção "-l" para informar o nome do usuário para login.
- » **comando:** alternativamente podemos conectar e entrar executando um programa ou comando específico.

Para conectar-se a um computador servidor SSH com um usuário desse servidor, utilize uma das duas alternativas a seguir. No exemplo o servidor SSH tem o endereço IP 192.168.1.254 e o login "maria" é de um usuário cadastrado nesse servidor:

```
# ssh maria@192.168.1.254  
# ssh -l maria 192.168.1.254
```

Se soubermos exatamente o que queremos fazer no servidor, podemos executar o comando diretamente e sair automaticamente ao terminar a execução do comando no computador remoto. No exemplo seguinte, ao entrar, será executado o comando init 6 que reinicia o computador remoto, considerando que o usuário tenha permissão para tal tarefa:

```
# ssh maria@192.168.1.254 init 6
```

Um outro cliente ssh muito útil é o scp, que serve para copiar arquivos entre o computador local e o servidor de ssh e vice-versa.

Veja no exemplo a seguir como é fácil utilizá-lo. Nele copiamos um arquivo do servidor ssh remoto para a pasta root da máquina local. Acompanhe:

```
# scp maria@192.168.1.254:/etc/hosts /root
```

Neste outro exemplo usamos a máquina local apenas como intermediadora na cópia do arquivo de um servidor ssh. Note que no servidor de endereço de IP 192.168.1.254 vamos utilizar o usuário "maria" para efetuar o login e no outro o usuário "nicolas". Observe:

```
# scp maria@192.168.1.254:/etc/hosts nicolas@192.168.1.253:/home/nicolas
```

Para finalizar os exemplos práticos do comando scp, veremos mais um exemplo com a opção "-r", que copia diretórios e subdiretórios recursivamente. Veja:

```
# scp -r /opt maria@192.168.1.254:/opt
```

Copiamos todo o conteúdo da pasta /opt da máquina local para o servidor em 192.168.1.254.

Os clientes de SSH do OpenSSH são bastante úteis e poderosos. Não hesite em consultar a man page de cada um, pois eles possuem muitas outras opções interessantes para o uso diário.

8.5 - Roteador de Rede

Roteadores são equipamentos que fazem a interligação de redes distintas. Um sistema Linux pode ser configurado como um roteador de rede. Neste item mostramos como configurar um roteador básico que interliga duas redes.

Para o exemplo vamos supor que temos duas redes, uma pequena empresa com uma matriz e uma filial, e precisamos interligar as duas redes. Assim vamos definir arbitrariamente os endereços de rede:

Tabela 8.3

Local	Rede	Máscara
1	192.168.1.0	255.255.255.0
2	192.168.2.0	255.255.255.0

O roteador que ficará instalado na central terá pelo menos duas interfaces de rede, cada uma ligada a uma das redes. Devemos configurar a primeira interface de rede com um IP da rede 1 e a segunda com um IP da rede 2. Para configurar as interfaces de redes, consulte o item X.6 Configuração do TCP/IP. Para o roteador vamos usar as seguintes configurações do TCP/IP:

Tabela 8.4

Interface	IP	Máscara
eth1	192.168.1.254	255.255.255.0
eth2	192.168.2.254	255.255.255.0

Ou seja, a interface de rede eth1 estará conectada à rede 1 e a eth2 à rede 2.

Para fazer um sistema Linux se transformar em um roteador, precisamos habilitar o encaminhamento de pacotes. Em termos gerais, o encaminhamento de pacotes permite que os pacotes sejam transferidos do sistema Linux de uma rede para outra. Devemos ativar a configuração do kernel do Linux colocando o valor "1" no parâmetro "net.ipv4.ip_forward" do arquivo /etc/sysctl.conf.

Trecho exemplo do arquivo /etc/sysctl.conf com o parâmetro ativado:

```
...
# Controls IP packet forwarding
net.ipv4.ip_forward = 1
...
```

Para verificar se está tudo corretamente configurado, podemos visualizar a configuração atual com o seguinte comando:

```
# sysctl -p
net.ipv4.ip_forward = 1
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.accept_source_route = 0
kernel.sysrq = 0
kernel.core_uses_pid = 1
net.ipv4.tcp_synccookies = 1
kernel.msgmnb = 65536
kernel.msgmax = 65536
kernel.shmmax = 4294967295
kernel.shmall = 268435456
```

Note que o parâmetro "net.ipv4.ip_forward" está em "1", ou seja, habilitado, mas para garantir que as configurações entrem em vigor sem a necessidade de reiniciar o computador, vamos executar o comando como segue:

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

Agora devemos habilitar o proxy ARP para que o sistema possa enviar pacotes de uma rede para outra. Ele pode ser habilitado para todas as interface de rede ou para uma específica. Para habilitar para todas as interfaces, inclua a linha seguinte no arquivo /etc/sysctl.conf como segue:

```
# Habilita Proxy ARP para todas as interfaces
net/ipv4/conf/all/proxy_arp = 1
```

Para habilitar interface específica, use:

```
# Habilita Proxy ARP para a interface eth0 e eth1
net/ipv4/conf/eth0/proxy_arp = 1
net/ipv4/conf/eth1/proxy_arp = 1
```

Para verificar a configuração atual, use o seguinte comando:

```
# sysctl -p
net.ipv4.conf.all.proxy_arp = 1
net.ipv4.ip_forward = 1
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.accept_source_route = 0
kernel.sysrq = 0
kernel.core_uses_pid = 1
net.ipv4.tcp_syncookies = 1
kernel.msgmnb = 65536
kernel.msgmax = 65536
kernel.shmmmax = 4294967295
kernel.shmall = 268435456
```

Note que o parâmetro "net.ipv4.conf.all.proxy_arp" está em "1", ou seja, habilitado, mas para garantir que as configurações entrem em vigor sem reiniciar o computador, vamos executar o comando como segue:

```
# echo 1 > /proc/sys/net/ipv4/conf/all/proxy_arp
```

Após configurarmos os parâmetros necessários e as interfaces de rede, podemos definir as tabelas estáticas de roteamento.

Precisamos interligar as redes. Devemos criar uma tabela de roteamento que diga ao sistema que, quando houver uma solicitação de um IP da rede 192.168.1.0, o tráfego deve ser direcionado para a eth1 e quando for para a rede 192.168.2.0, o tráfego será direcionado para a eth2. Podemos definir essas rotas com o comando route como segue:

```
# route add -net 192.168.1.0/24 dev eth1
# route add -net 192.168.2.0/24 dev eth2
```

Relembrando que o "/24" no final do endereço de rede indica que é uma máscara de rede de 24 bits, ou seja, é equivalente a escrever 255.255.255.0. Podemos executar o comando da outra forma apresentada em seguida e teremos o mesmo resultado:

```
# route add -net 192.168.1.0 netmask 255.255.255.0 dev eth1
# route add -net 192.168.2.0 netmask 255.255.255.0 dev eth2
```

Configure os computadores clientes na rede usando o IP da eth1 como gateway da rede 1 e da eth2 para a rede 2.

8.6 - Servidor DHCP

Um servidor DHCP (Dinamic Host Configuration Protocol) possui como função básica enviar as configurações de rede para as estações que estão configuradas para usar DHCP. Ele envia as informações necessárias para que a estação esteja disponível na rede como o endereço IP, máscara de sub-rede, gateway e DNS.

Existem algumas implementações de servidor DHCP para Linux, sendo o mais utilizado o dhcpcd, abordado neste item. Primeiramente vamos instalar ou certificar se já está instalado o dhcpcd. Podemos usar o yum para fazer isso, como segue:

```
# yum install dhcpcd
```

A configuração do dhcpcd deve ser inserida no arquivo /etc/dhcpcd.conf. Na instalação esse arquivo é criado sem nenhuma configuração. Um arquivo de exemplo pode ser encontrado em /usr/share/doc/dhcp-<versão>/dhcpcd.conf.sample. Podemos usá-lo como referência ou criar outro arquivo.

Para o servidor DHCP vamos deixar o arquivo /etc/dhcpcd.conf como segue:

```
ddns-update-style ad-hoc;
default-lease-time 600;
max-lease-time 7200;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.1.255;
option routers 192.168.1.254;
option domain-name-servers 192.168.1.1, 192.168.1.2;
option domain-name "guiapratico.com";

subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.10 192.168.1.100;
}
```

No arquivo /etc/dhcpcd.conf temos:

- ▶ **ddns-update-style ad-hoc:** define o modo da atualização do DNS usado como ad hoc para manter compatibilidade com versões anteriores.
- ▶ **default-lease-time:** tempo padrão em segundos para concessão de um IP para determinado host.
- ▶ **max-lease-time:** tempo máximo em segundos de concessão de um IP, caso solicitado pelo cliente.
- ▶ **option subnet-mask:** indica a máscara de sub-rede que deve ser enviada aos clientes.
- ▶ **option broadcast-address:** indica o endereço de broadcast da rede.
- ▶ **option routers:** indica o IP do gateway da rede a ser enviado aos clientes.
- ▶ **option domain-name-servers:** fornece uma lista de endereços IP dos servidores de DNS para os clientes.
- ▶ **option domain-name:** indica o nome do domínio para os clientes.

- »» **subnet 192.168.1.0 netmask 255.255.255.0:** abre a seção de configuração da sub-rede. Note a existência do caractere "{" e "}" para abrir e fechar a seção respectivamente.
- »» **range 192.168.1.10 192.168.1.100:** aloca a faixa de IP para enviar aos clientes da sub-rede, ou seja, neste exemplo os clientes dhcp conectados a essa sub-rede terão IPs dentro da faixa estipulada nesse parâmetro.

Para marcar na tabela de serviços a ativação do serviço dhcpcd ao iniciar, use:

```
# chkconfig dhcpcd on
```

Terminadas as configurações, podemos iniciar ou reiniciar o serviço caso já esteja iniciado para que as configurações entrem em vigor como segue:

```
# service dhcpcd restart
```

Uma base de dados contendo os IPs concedidos para os clientes, tempo de concessão, endereço MAC da placa de rede do cliente, entre outras informações, é criada em /var/lib/dhcpcd/dhcpcd.leases. Você não deve alterar esse arquivo nem criá-lo se ele não existir, pois periodicamente ele é recriado. Se um eventual problema surgir ao recriá-lo, portanto, o correto é deixar que o próprio servidor recrie o arquivo, forçando o reinício do servidor DHCP.

8.7 - Servidor FTP

Um servidor FTP (File Transfer Protocol) permite que um usuário transfira arquivos entre hosts, ou seja, transferir arquivo do computador local para o computador remoto. Existem diversas implementações de servidores FTP, entretanto estudaremos o vsftpd (Very Secure FTP Daemon) que é um dos mais utilizados em ambientes Linux e é o padrão nos derivados de Red Hat.

Nele os usuários são autenticados usando o controle de senhas do sistema. Também é possível entrar sem autenticação, conhecido como FTP anônimo, porém não é recomendado, pois é um método não seguro. O FTP utiliza por padrão a porta 21.

Então, antes de iniciar precisamos instalar ou verificar se já está instalado o servidor FTP vsftpd no sistema. Use o yum para resolver isso como segue:

```
# yum install vsftpd
```

As configurações do servidor FTP estão no arquivo /etc/vsftpd/vsftpd.conf. Para o funcionamento básico não é necessário fazer alterações nesse arquivo.

Exemplo de arquivo /etc/vsftpd/vsftpd.conf após instalação, sem os comentários, ou seja, somente com os parâmetros em uso:

```
anonymous_enable=YES
local_enable=YES
write_enable=YES
local_umask=022
dirmessage_enable=YES
xferlog_enable=YES
```

```
connect_from_port_20=YES
xferlog_std_format=YES
listen=YES
pam_service_name=vsftpd
userlist_enable=YES
tcp_wrappers=YES
```

Alguns comentários importantes sobre o arquivo /etc/dhcpd.conf:

- » **anonymous_enable:** define se é permitido o acesso com usuário anônimo. Por padrão ele está sempre ativo; inclusive se o parâmetro for omitido, ele fica ativo. Se desejar bloquear o acesso ao usuário anonymous, é necessário deixar esse parâmetro igual a "no".
- » **local_enable:** habilita ou desabilita o acesso aos usuários do sistema local.
- » **write_enable:** indica se permite ou não a gravação de arquivos.
- » **userlist_enable:** se estiver em "yes", indica que os usuários listados no arquivo /etc/vsftpd.user_list terão o acesso negado ao servidor FTP.

Com o parâmetro "userlist_enable" habilitado você pode determinar uma lista de usuários sem acesso ao FTP. Isso é importante porque bloqueia o acesso antes mesmo de solicitar a senha e previne que usuários importantes enviem senhas em modo texto. Para bloquear os usuários antes mesmo de solicitar a senha, use o arquivo /etc/vsftpd/user_list e para bloquear somente a senha de acesso do usuário, use /etc/vsftpd/ftpusers.

Exemplo de arquivo /etc/vsftpd/user_list:

```
# vsftpd userlist
# If userlist_deny=NO, only allow users in this file
# If userlist_deny=YES (default), never allow users in this file, and
# do not even prompt for a password.
# Note that the default vsftpd pam config also checks /etc/vsftpd/ftpusers
# for users that are denied.
root
bin
daemon
adm
lp
sync
shutdown
halt
mail
news
uucp
operator
games
nobody
```

Exemplo de arquivo /etc/vsftpd/ftpusers:

```
# Users that are not allowed to login via ftp
root
bin
daemon
adm
lp
sync
shutdown
```

```
halt  
mail  
news  
uucp  
operator  
games  
nobody
```

Existe uma grande variedade de configurações no vsftpd, mas informações podem ser lidas no endereço http://www.centos.org/docs/5/html/Deployment_Guide-en-US/s1-ftp-vsftpd-conf.html.

Para marcar na tabela de serviços a ativação do serviço vsftpd ao iniciar, use:

```
# chkconfig vsftpd on
```

Agora podemos iniciar ou reiniciar o servidor FTP com o comando seguinte:

```
# service vsftpd restart
```

O acesso anônimo por padrão aponta para o conteúdo da pasta /var/ftp/pub/, já o acesso feito por usuários do sistema aponta para a pasta de cada usuário no sistema.

8.8 - Servidor WWW

O servidor WWW (World Wide Web) é um serviço que está na camada de aplicação do modelo OSI e utiliza o protocolo HTTP (HyperText Transfer Protocol). Esses servidores também precisam de um cliente e um servidor. Neste caso o navegador da Internet é o cliente e WWW é o servidor. Existem algumas implementações de servidor WWW, inclusive para Windows como o IIS (Internet Information Service) da Microsoft. O sistema estudado neste item sem dúvida é o mais usado em todo o mundo. Foi com certeza um dos grandes impulsionadores da difusão de sistemas Linux, além de também funcionar no Windows. Ele é o Apache HTTP Server, desenvolvido pela Apache Software Foundation (<http://www.apache.org>), sendo extremamente confiável, flexível, robusto e poderoso.

Devido à ampla possibilidade de configurações do servidor Web Apache e ao fato de que a configuração padrão dele atende a maioria dos casos, veremos como montar um servidor WWW rapidamente, utilizando-se dos recursos disponíveis por padrão. Vamos adicionar um add-on que é o suporte ao PHP.

Para instalar o servidor Web Apache ou atualizar, podemos utilizar o gerenciador de pacotes yum como segue:

```
# yum install httpd
```

O diretório com os arquivos de configuração do Apache é o /etc/httpd/, no qual se encontra o subdiretório conf que abriga o principal arquivo de configuração do Apache, o httpd.conf que mantém parâmetros como nome do servidor, porta em uso, diretório dos arquivos

html, entre outras, e no subdiretório conf.d estão as configurações dos módulos adicionais, como o PHP, por exemplo. Ainda em /etc/httpd/ são encontrados o subdiretório modules, que hospeda os módulos instalados, e a logs em que estão os logs do sistema.

O arquivo de configuração /etc/httpd/conf/httpd.conf no padrão da instalação já funciona normalmente, por se tratar de um arquivo com muitas possibilidades. Neste item veremos apenas os parâmetros mais comumente alterados.

Alguns comentários sobre configurações básicas do /etc/httpd/conf/httpd.conf que são comumente utilizadas:

- » **ServerRoot:** é o diretório topo da árvore de diretório do servidor Apache. Nele estão as configurações, erros e arquivos de logs. O padrão é /etc/httpd.
- » **Listen:** é o parâmetro que indica ao Apache em que porta o servidor aceitará conexões. O padrão é a porta 80.
- » **User:** define o usuário que será proprietário dos arquivos e processos httpd. Esse parâmetro é um item importante para segurança do servidor, pois podemos definir um usuário específico para controle.
- » **Group:** define o grupo que será proprietário dos arquivos e processos httpd.
- » **ServerAdmin:** que indica o endereço de e-mail do administrador do sistema exibido em momentos adversos, como erros ou falhas do servidor.
- » **ServerName:** especifica um nome e uma porta que o servidor usará para se identificar. Isso pode ser determinado automaticamente pelo Apache ao iniciar, mas é prudente especificar para evitar problemas durante a inicialização. Normalmente aqui podemos colocar o IP e a porta do servidor ou o hostname e a porta.
- » **DocumentRoot:** diretório raiz em que serão hospedados os arquivos ofertados pelo servidor Apache e os arquivos referentes aos sites hospedados. O padrão é /var/www/html.
- » **DirectoryIndex:** define o arquivo a ser mostrado quando um diretório for solicitado. Normalmente é algo como index.html ou index.php, ou seja, é o primeiro ou principal arquivo do site.

Para marcar na tabela de serviços a ativação do serviço httpd ao iniciar, use:

```
# chkconfig httpd on
```

Para iniciar o servidor Web Apache, podemos usar o programa apachectl ou o gerenciador de serviços do sistema, como segue:

```
# apachectl start
# service httpd start
```

Os dois comandos têm o mesmo efeito.

Com o servidor iniciado podemos acessar usando um cliente WWW, por exemplo, o Mozilla Firefox. Tente acessar usando o endereço <http://localhost> ou <http://127.0.0.1/>. Deve ser exibida a tela inicial padrão do Apache Web Server, conforme a Figura 8.7.

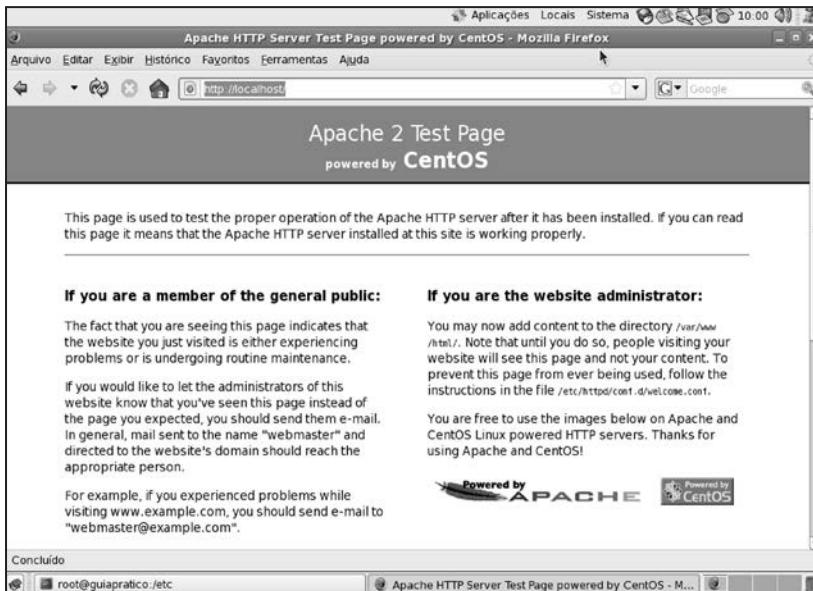


Figura 8.7

Neste ponto temos um servidor Web Apache funcionando e já podemos hospedar sites no diretório que aponta o parâmetro DocumentRoot. O padrão é /var/www/html, mas é comum querermos utilizar o PHP associado ao Apache. O PHP é uma linguagem de script embutida no HTML, muito poderosa, e utilizada em servidores Apache. Para mais informações visite <http://www.php.net>.

Para habilitar a integração com PHP no servidor Web Apache, precisamos instalar o php e o módulo para integração entre eles, bem como configurar o Apache para usar o PHP. Para instalar, a melhor alternativa é usar o gerenciador de pacotes yum, como segue:

```
# yum install php
```

Por padrão somente os pacotes essenciais são instalados. Se você quiser instalar todos os módulos do php, use:

```
# yum install php*
```

Para testar o php, vá ao diretório do DocumentRoot (/var/www/html) e crie um arquivo de teste do php, como segue.

A seguir, um exemplo de arquivo para teste do php /var/www/html/teste.php:

```
<?php  
phpinfo();  
?>
```

Para testar, abra o browser e coloque o endereço <http://localhost/teste.php>. A saída deve ser como a da Figura 8.8.



Figura 8.8

Para finalizar, devemos adicionar ao parâmetro `DirectoryIndex` a busca de arquivos do tipo `php`, ou seja, inclua nesse parâmetro as opções `index.php` e `index.php5` para que o Apache procure esses arquivos também quando um diretório for solicitado.

8.9 - Servidor NTP

O NTP (Network Time Protocol) é um protocolo utilizado como meio de trocar mensagens sobre tempo (data e hora) entre computadores. Um servidor NTP escuta na porta 123 udp e pode ser chamado de "servidor de hora" ou "servidor de horário", ou seja, sua principal função é disponibilizar a informação de data e hora para que os computadores solicitantes sincronizem o horário.

Para construir um servidor NTP precisamos ter instalados os pacotes referentes ao servidor NTP. Para instalar podemos usar o gerenciador de pacote yum, como segue:

```
# yum install ntp
```

Após a instalação vamos configurar para sincronizar data e hora toda vez que iniciar o serviço. Vamos incluir a opção "`-x`" na linha "`OPTIONS="-u ntp:ntp -p /var/run/ntpd.pid"`" do arquivo `/etc/sysconfig/ntpd`. O arquivo deve ficar como segue:

```
# Drop root to id 'ntp:ntp' by default.
OPTIONS="-x -u ntp:ntp -p /var/run/ntpd.pid"

# Set to 'yes' to sync hw clock after successful ntpdate
SYNC_HWCLOCK=no

# Additional options for ntpdate
NTPDATE_OPTIONS=""
```

O arquivo com as regras do servidor NTP é o /etc/ntp.conf, no qual podemos determinar regras para o acesso e servidores para atualização da hora. Normalmente não há necessidade de efetuar nenhuma alteração nele, pois vem com as configurações padrão e sincroniza a hora com os servidores oficiais do projeto Server Pool do NTP. Caso você deseje inserir outros servidores de hora para sincronização do seu servidor NTP, inclua em qualquer parte do arquivo, preferencialmente logo abaixo dos servidores já existentes, o parâmetro "server Servidor_NTP". Veja alguns exemplos de servidores:

```
server ntpsl.pads.ufrj.br # (UFRJ), Rio de Janeiro, Brazil
server ntp1.rnp.br          # Rede Nacional de Pesquisa (RNP)
server clock.via.net       # ViaNet Communications, Palo Alto, CA, USA
server ntp.pads.ufrj.br    # UFRJ, Rio de Janeiro, Brazil (PADS)
server ntp2.pads.ufrj.br   # UFRJ, Rio de Janeiro, Brazil (PADS)
server ntp.pop-mg.rnp.br  # POP-ES (RNP)
server ntp.pop-es.rnp.br # POP-MG (RNP)
```

Para marcar na tabela de serviços a ativação do serviço ntpd ao iniciar, use:

```
# chkconfig ntpd on
```

Agora podemos iniciar o servidor NTP e disponibilizar o serviço na rede para qualquer computador que requisitar. Use o comando descrito a seguir, pois se o servidor já estiver iniciado, ele será reiniciado e caso não esteja, ele será iniciado.

```
# service ntpd restart
```

8.9.1 - Sincronização de um Cliente NTP em Linux

Instale o pacote NTP, conforme instalamos para o servidor NTP, pois é o mesmo pacote. Podemos sincronizar a hora com um servidor NTP de duas maneiras. A primeira seria usando o ntpdate e a segunda usando o servidor ntpd.

Usando o ntpdate, execute o comando a qualquer momento para sincronizar o relógio. A sintaxe básica do comando é a seguinte:

```
# ntpdate Servidor_NTP
```

O "Servidor_NTP" é o endereço IP ou hostname do servidor NTP.

Usando o ntpd, execute o comando seguinte a qualquer momento que o relógio será sincronizado. Neste caso, o servidor ao qual será solicitado o horário deve estar no arquivo /etc/ntp.conf, ou seja, tenha certeza de que pelo menos uma das incidências do parâmetro "server" existente nesse arquivo aponta para o IP ou hostname do seu servidor NTP.

```
# ntpd -q -u ntp:ntp
```

Para manter sempre sincronizado, uma boa prática é incluir esse comando no agendador do sistema, o cron.

8.10 - Servidor DNS

DNS (Domain Name Server) é o serviço de resolução de nomes, que tem a função de traduzir nomes em endereços IP e vice-versa. Por uma questão prática, os endereços de IP precisam ser transformados em nomes, pois imagine se toda vez que fosse preciso acessar um site, tivéssemos de lembrar o endereço IP dele. Portanto, essa tarefa é realizada pelos servidores DNS espalhados pelo mundo, distribuídos e interligados de forma hierárquica, e cada servidor é responsável pelos seus domínios e subdomínios.

No topo dessa estrutura hierárquica estão os servidores raiz que identificam todos os TLDs (top-level domain) existentes. Um TLD pode ser o primeiro nome num FQDN (Fully qualified domain name). Para o www.shellinux.com.br o TLD é o "br"; descendo a hierarquia estão os demais servidores e cada um sabe os TLDs que possui, e assim por diante até chegar ao endereço da máquina. Todas as sessões de um FQDN, com exceção da última, a mais à esquerda dele, são zonas definidas por meio dos arquivos de configurações de zonas que descrevem o espaço daquela zona, como também os servidores de e-mail a serem utilizados em um ou mais domínios, entre outros parâmetros. Também é possível solicitar um nome de domínio enviando um endereço de IP para um servidor DNS, conhecido como DNS reverso.

O DNS é fundamento em dois pontos estratégicos que consistem na organização de domínios para a Internet e na distribuição organizada dos servidores de nomes de rede. Distribuição porque seria uma prática inviável manter todos os nomes de domínios em apenas uma base de dados.

Quando configuramos um sistema Linux para funcionar em rede e queremos acessar a Internet, é muito provável precisarmos de pelo menos um e até três servidores de DNS para informar ao sistema. Assim, o sistema ganha a capacidade de acessar computadores ou hosts a partir dos seus FQDNs.

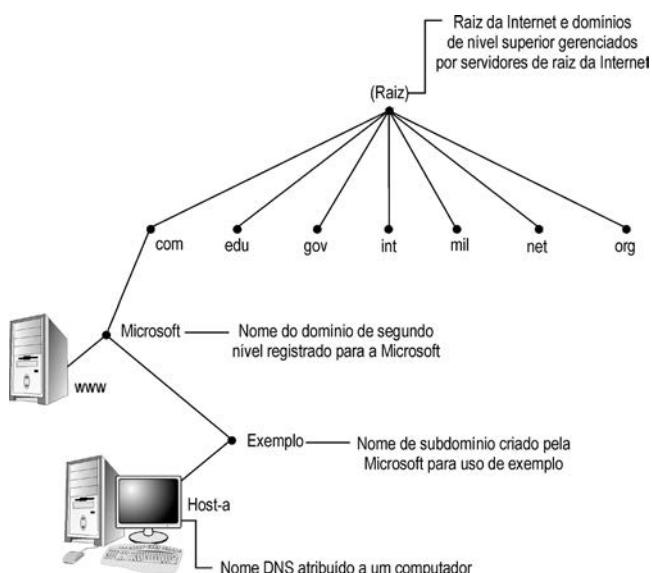


Figura 8.9 - Fonte: http://www.terraempresas.com.br/web/html/theme/default/images/faq/nocoes_basicas_clip_image001.gif

Um sistema Linux pode funcionar como um servidor DNS, mas configurar o Linux dessa maneira está além do escopo deste livro. Veremos os tipos de servidores e onde estão localizados os principais arquivos de configuração.

Um servidor DNS possui quatro tipos de configurações básicas, a saber: o Caching-Only (Somente Cache), servidores que não são fontes oficiais de domínios, pois consultam sempre um servidor remoto para obter a informação; o modo Master (Servidor Mestre), que se torna um servidor oficial de informações de domínios, e recebe este nome por ser capaz de responder a qualquer informação sobre seu domínio com total autoridade; o tipo Slave (Servidor Escravo), também conhecido como servidor secundário, que possui uma cópia do banco de dados completa do Master Server; o modo Forwarding (Encaminhamento), que encaminha solicitações a uma lista específica para a resolução de nomes, quando nenhum dos servidores de nome especificado pode executar a resolução.

No Linux e no UNIX, uma implementação de DNS chamada BIND é a mais utilizada. Ela possui algumas peças de servidor e cliente, como o resolver que é o resolvedor, um cliente que solicita informações sobre um computador e o name server que é o servidor de nomes, no BIND chamado de named, o qual responde as solicitações aos clientes.

Basicamente, ao acessar um site na Internet, o resolver consulta o arquivo /etc/resolv.conf para localizar o endereço IP do servidor DNS, solicita ao DNS o IP do domínio da página que queremos acessar. O named no servidor DNS verifica se o domínio está no seu cache; caso não esteja, ele procura pelo DNS responsável por aquele domínio até encontrá-lo, para retornar o endereço IP ao cliente ou retornar uma mensagem de erro.

Para instalar o BIND, podemos usar o gerenciador de pacotes yum como segue:

```
# yum install bind
```

Para marcar na tabela de serviços a ativação do serviço named ao iniciar, use:

```
# chkconfig named on
```

A implementação named do servidor DNS está localizada em /usr/sbin/named. O pacote BIND inclui também um utilitário de administração do DNS chamado rndc e localizado em /usr/sbin.

O principal arquivo de configuração é o named.conf localizado no diretório /etc.

Exemplo de /etc/named.conf:

```
//  
// Sample named.conf BIND DNS server 'named' configuration file  
// for the Red Hat BIND distribution.  
//  
// See the BIND Administrator's Reference Manual (ARM) for details, in:  
//   file:///usr/share/doc/bind-*/arm/Bv9ARM.html  
// Also see the BIND Configuration GUI : /usr/bin/system-config-bind and  
// its manual.  
//
```

```

options
{
    // Those options should be used carefully because they disable port
    // randomization
    // query-source      port 53;
    // query-source-v6   port 53;

    // Put files that named is allowed to write in the data/ directory:
    directory "/var/named"; // the default
    dump-file          "data/cache_dump.db";
    statistics-file    "data/named_stats.txt";
    memstatistics-file "data/named_mem_stats.txt";

};

logging
{
/*      If you want to enable debugging, eg. using the 'rndc trace' command,
 *      named will try to write the 'named.run' file in the $directory (/var/
named).
 *      By default, SELinux policy does not allow named to modify the /var/
named directory,
 *      so put the default debug log file in data/ :
 */
    channel default_debug {
        file "data/named.run";
        severity dynamic;
    };
};

// All BIND 9 zones are in a "view", which allow different zones to be served
// to different types of client addresses, and for options to be set for
groups
// of zones.
//
// By default, if named.conf contains no "view" clauses, all zones are in the
// "default" view, which matches all clients.
//
// If named.conf contains any "view" clause, then all zones MUST be in a view;
// so it is recommended to start off using views to avoid having to
restructure
// your configuration files in the future.
//
view "localhost_resolver"
{
/* This view sets up named to be a localhost resolver ( caching only
nameserver ).

 * If all you want is a caching-only nameserver, then you need only define
this view:
 */
    match-clients           { localhost; };
    match-destinations       { localhost; };
    recursion yes;
    # all views must contain the root hints zone:
    include "/etc/named.root.hints";

    /* these are zones that contain definitions for all the localhost
     * names and addresses, as recommended in RFC1912 - these names should
     * ONLY be served to localhost clients:
     */
    include "/etc/named.rfc1912.zones";
};

view "internal"
{
/* This view will contain zones you want to serve only to "internal" clients
that connect via your directly attached LAN interfaces - "localnets" .
*/
}

```

```
match-clients          { localnets; };
match-destinations     { localnets; };
recursion yes;
// all views must contain the root hints zone:
include "/etc/named.root.hints";

        // include "named.rfc1912.zones";
// you should not serve your rfc1912 names to non-localhost clients.

// These are your "authoritative" internal zones, and would probably
// also be included in the "localhost_resolver" view above :

zone "my.internal.zone" {
    type master;
    file "my.internal.zone.db";
};

zone "my.slave.internal.zone" {
    type slave;
    file "slaves/my.slave.internal.zone.db";
    masters { /* put master nameserver IPs here */ 127.0.0.1; } ;
    // put slave zones in the slaves/ directory so named can update them
};

zone "my.ddns.internal.zone" {
    type master;
    allow-update { key ddns_key; };
    file "slaves/my.ddns.internal.zone.db";
    // put dynamically updateable zones in the slaves/ directory so named
can update them
};

key ddns_key
{
    algorithm hmac-md5;
    secret "use /usr/sbin/dns-keygen to generate TSIG keys";
};

view    "external"
{
/* This view will contain zones you want to serve only to "external" clients
 * that have addresses that are not on your directly attached LAN interface
subnets:
*/
    match-clients          { any; };
    match-destinations     { any; };

    recursion no;
    // you'd probably want to deny recursion to external clients, so you
don't
        // end up providing free DNS service to all takers

    allow-query-cache { none; };
    // Disable lookups for any cached data and root hints

    // all views must contain the root hints zone:
    include "/etc/named.root.hints";

    // These are your "authoritative" external zones, and would probably
    // contain entries for just your web and mail servers:

zone "my.external.zone" {
    type master;
    file "my.external.zone.db";
};

};
```

No diretório /var/named/ estão armazenados as zonas, as estatísticas e os arquivos de cache.

Exercícios

1. Explique de forma resumida o TCP/IP.
2. Quais são as classes de endereços possíveis?
3. Qual a máscara da classe B?
4. Qual o intervalo de endereços da classe A?
5. Como configurar manualmente a interface de rede?
6. Para que serve o comando route –n?
7. O que é NFS?
8. Para que serve o telnet?
9. Defina servidor DHCP.
10. Qual a função de um servidor FTP?
11. O que faz um servidor DNS?

Bibliografia

BARRETT, D. J.; BYRNES, R. G.; SILVERMAN, R. **Linux Security Cookbook**. O'Reilly, June, 2003.

BLUM, R.; LEBLANC, D. A. **Linux FOR DUMMIES**. 9th Edition. Published by Wiley Publishing, Inc.

NEMETH, E.; SNYDER, G. **Linux Administration Handbook**. Second Edition. Trent R. Hein Copyright® 2007. Pearson Education, Inc.

RAYMOND, E. S. **The Art of Unix Programming**. Thryrus Enterprises Copyright®, 2003.

STANFIELD, V.; SMITH, R. W. **Linux System Administration**. Second Edition. SYBEX.

Sites

http://www.faqs.org/docs/linux_intro/

<http://www.faqs.org/faqs/unix-faq/>

http://www.linfo.org/main_index.html

http://www.centos.org/docs/5/html/Deployment_Guide-en-US/index.html

http://www.unix.org/what_is_unix/history_timeline.html

<http://www.linuxhomenetworking.com/wiki/index.php>

http://www.digibarn.com/collections/newsletters/homebrew/V2_01/gatesletter.html

[http://en.wikipedia.org/wiki/Kernel_\(computing\)](http://en.wikipedia.org/wiki/Kernel_(computing))

<http://www.gnu.org/>

<http://www.kernel.org/>

<http://www.fsf.org/>

<http://www.minix3.org/>

<http://www.pathname.com/fhs/>

<http://www.linuxfoundation.org/>

<http://wiki.sintectus.com/bin/view/GrupoLinux/ArtigoConceitosEHistorioaDoLinux>

<http://www.faqs.org/faqs/unix-faq/faq/part6/section-2.html>

http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO_:_Ch03_:_Linux_Networking

<http://www.tuxfiles.org>

<http://linux.die.net>

<http://www.vivaolinux.com.br>

<http://bash.cyberciti.biz/guide>

Revistas

Revista do Linux - Ano II - n. 20 - Agosto/2001.

Filesystem Hierarchy Standard por Filesystem Hierarchy Standard Group editado por Rusty Russell, Daniel Quinlan e Christopher Yeoh. Published January 28 2004.

Marcas Registradas

Linux é marca registrada da centOS Team.

Todos os demais nomes registrados, marcas registradas ou direitos de uso citados neste livro pertencem aos seus respectivos proprietários.

Índice Remissivo

A

- Adição de grupos 62
- Administrador 106
- Ajuda 54
- Aplicativos do Microsoft Windows 27
- Apt 69, 72
- Arquivo(s) 55, 126, 178

B

- Boot 101, 113

C

- Chkconfig 84
- Cliente(s)
 - SSH 186
 - Telnet 184
- Comandos básicos 54
- Configuração 177
- CPU 73, 76

D

- Daemons 84
- Desligamento 118
- Desmontagem 123
- Diretórios 55
- Disco 79
 - rígido 98

E

- Endereço de rede 166

F

- Filesystem Hierarchy Standard (FHS) 41
- Free Software Foundation 24
- Fuso horário 106

G

- Gerenciamento de grupos 62
- GNU
 - AGPL 25
 - FDL 25
 - LGPL 25
- GRUB 114

H

- Halt 121
- Hard link 127

I

- Inicialização 112
- Init 120
- Instalação 95, 108
 - remota 110
- IP fixo 174

K

- Kernel 28, 29
- Kill 88
- Killall 88

L

- Layout do teclado 98
- LILO 113
- Link simbólico 127
- Linux Foundation 36
- Login remoto 183
- Logs 82
- LSB (Linux Standard Base) 36

M

- Manutenção 116
- Math 15, 33, 40, 44, 47, 62

Memória 77
virtual 35
Mensagens 82
Micronúcleo 28
monousuário 116
simbólico 58
Montagem 123
Multitarefa 32
Multitasking 32

N

Núcleo
do sistema 28
monolítico 28

O

Octal 58
Open Printing 36

P

Pacotes 70-72, 107
Pilha de rede 164
Portas 167
POSIX 27
Poweroff 121
Processos 87
Projeto GNU 23

R

Reboot 121
Rede 80, 104, 109, 164

Remoção de grupos 63
Roteador de rede 187
RPM 71

S

Seleção do idioma 97
Senha 65
Service 86
Serviços 84
Servidor
DHCP 189
DNS 198
FTP 191
NTP 196
OpenSSH 185
WWW 193
Shutdown 119
Superusuário 89

T

TCP/IP 164, 169
Tempo de execução 174
Teste do DVD 96

U

UNIX 32
Usuário(s) 64, 66
cadastrado 67

W

Windows 27



Windows Server 2012 - Fundamentos

Autor: Marco Aurélio Thompson

Código: 4308 • 240 páginas • Formato: 17,5 x 24,5 cm • ISBN: 978-85-365-0430-8 • EAN: 9788536504308

Com exemplos práticos e telas que ilustram passo a passo as operações, o livro ensina os fundamentos do Windows Server 2012, versão em português, da Microsoft, orientando estudantes e profissionais da área na criação de uma infraestrutura de rede baseada nesse sistema.

Comenta as novidades desta versão e demonstra como criar um laboratório de testes para praticar o conteúdo. Explica procedimentos pós-instalação e modelagem de redes, revisa o funcionamento do DNS e trata do Active Directory. Instrui a criação de vários servidores, como o DHCP, o DNS, o de arquivos, o de impressão e o controlador de domínios. Fornece dicas para manter o sistema seguro e em funcionamento, além de um glossário com os termos mais comumente usados em redes e um apêndice com o roteiro para obtenção de certificação da Microsoft para essa plataforma.



Microsoft Windows Server 2012 - Instalação, Configuração e Administração de Redes

Autor: Marco Aurélio Thompson

Código: 4346 • 368 páginas • Formato: 17,5 x 24,5 cm • ISBN: 978-85-365-0434-6 • EAN: 9788536504346

Com o objetivo de auxiliar estudantes e profissionais no gerenciamento de redes com o Windows Server 2012, este livro foi desenvolvido, repleto de telas explicativas passo a passo.

Faz uma revisão de redes e TCP/IP, descreve as novidades dessa geração de servidores e explica como criar um laboratório virtual de práticas. Introduz a nova interface de usuário Modern UI (ex-Metro), procedimentos pós-instalação, tipos de RAID, funções e recursos do programa. Destaca os servidores de arquivos e armazenamento, de impressão e documentos, DHCP, DNS, Web (IIS) e o Active Directory, bem como Server Core, PowerShell e Hyper-V. Trata de administração remota, planejamento da carreira, segurança e gerenciamento de redes, trazendo um glossário com termos comuns e apêndices com comandos úteis e um roteiro para obter uma certificação da Microsoft para essa plataforma.



Windows Server 2008 R2 - Instalação, Configuração e Administração de Redes

Autor: Marco Aurélio Thompson

Código: 3066 • 336 páginas • Formato: 17,5 x 24,5 cm • ISBN: 978-85-365-0306-6 • EAN: 9788536503066

Com linguagem simples e ilustrações que esclarecem passo a passo cada operação, este livro ensina desde os conceitos básicos até os mais avançados, para que estudantes e profissionais da área possam construir uma rede de computadores segura, de alta confiabilidade e fácil gerenciamento.

Traz uma visão geral do ambiente de redes e apresenta protocolos do TCP/IP, principais funções, infraestrutura com DHCP, WINS, DNS, diretivas de rede, Active Directory, servidor de arquivos e de fax, servidor de impressão e de web com o IIS, a virtualização com o Hyper-V, as diferentes formas de acesso remoto ao servidor, segurança da rede, como usar scripts de configuração com o PowerShell, além da descrição de diversas tarefas de administração da rede, com seus problemas mais comuns e as respectivas soluções.



Windows Server 2003 - Administração de Redes

Autor: Marco Aurélio Thompson

Código: 9808 • 376 páginas • Formato: 17 x 24 cm • ISBN: 978-85-7194-980-5 • EAN: 9788571949805

Ensina a gerenciar o Windows 2003 Server em rede e mostra como realmente é o dia a dia do administrador.

Está organizado de forma didática, abordando conceitos básicos de redes, arquiteturas, protocolos e instalação da versão Server, os tipos de servidor que o Windows 2003 pode se tornar (controlador de domínio, servidor de arquivos, de impressão, DNS, WINS, DHCP, servidor Web (WWW e FTP) etc.), criação de uma Intranet, adotando uma política de segurança, além de dicas e macetes do Windows 2003 e orientações para certificação Microsoft.

É indicado aos profissionais e alunos da área de informática que desejam ingressar no lucrativo mercado de administração de redes.



Windows Server 2003 em português - Implementação e Administração

Autor: Eng. Francisco Baddini

Código: 9832 • 376 páginas • Formato: 17 x 24 cm • ISBN: 978-85-7194-983-6 • EAN: 9788571949836

Traz os procedimentos recomendados na implementação do Windows Server 2003. Trata desde os principais recursos de hardware existentes nos servidores até instalação, configuração, aplicações web, otimização e gerenciamento dos recursos da plataforma. Um capítulo especial aborda o Microsoft ISA Server 2000, com o qual é possível transformar o servidor numa poderosa solução de segurança. É destinado aos profissionais e alunos da área de informática que desejam implementar e entender como funcionam os principais recursos e produtos desenvolvidos para essa plataforma.



Estudo Dirigido de Microsoft Windows 8 Enterprise

Autor: André Luiz N. G. Manzano

Código: 437A • 168 páginas • Formato: 17 x 24 cm • ISBN: 978-85-365-0437-7 • EAN: 9788536504377

Com estrutura agradável, este livro traz as principais características operacionais do Windows 8 e sua nova interface, baseada no design Metro, seus blocos dinâmicos e atalhos de navegação. Explica como alternar dois ambientes de trabalho, criar contas de usuário, gerenciar arquivos e pastas, além do controle de acesso à Internet e noções básicas. Trata de terminologias, CPU, memórias, periféricos, novidades da versão, área de trabalho, principais aplicativos, acessórios e o Explorador de Arquivos. Contempla assuntos importantes e atuais, como segurança, tecnologia Bluetooth, Wireless, Windows Defender, Firewall e Windows Update. É indicado a professores, alunos e autodidata. Traz exemplos e exercícios didáticos para facilitar o aprendizado.



Microsoft Windows 7 Professional - Guia Essencial de Aplicação

Autor: José Augusto N. G. Manzano

Código: 3035 • 296 páginas • Formato: 17,5 x 24,5 cm • ISBN: 978-85-365-0303-5 • EAN: 9788536503035

O Microsoft Windows 7 e seus recursos essenciais estruturam este livro prático e dirigido, que conceitua sistema operacional, explana a estrutura organizacional do Windows, instalação e configurações iniciais. Aborda acesso ao computador por portadores de necessidades especiais e descreve as ferramentas de produtividade e de sistema, como bloco de notas, calculadora, prompt de comando, impressoras etc., o programa Windows Explorer e recursos de multimídia. Inclui gerenciamento do sistema, bibliotecas e pastas do usuário.

Conectividade e acessibilidade à Internet, Windows Update, manutenção do sistema e outros recursos auxiliares, como o modo de compatibilidade com o Microsoft Windows XP, são assuntos tratados.



Linux - Fundamentos

Autores: Wallace Soares e Gabriel Fernandes

Código: 3219 • 208 páginas • Formato: 17,5 x 24,5 cm • ISBN: 978-85-365-0321-9 • EAN: 9788536503219

Os conceitos e aspectos essenciais do Linux, sua instalação e gerenciamento básico são explanados de forma objetiva e prática. O livro aborda a história do programa, sua relação com o UNIX, estrutura, vantagens, aspectos práticos do dia a dia do administrador, as tarefas exigidas, ferramentas e comandos, gerenciamento de usuários e grupos, monitoramento do servidor, administração de serviços (daemons), execução da instalação de forma eficiente, os gerenciadores de boot comumente utilizados (GRUB e LILO) e dicas de segurança.

Detalha os sistemas de arquivos, a ferramenta Shell, protocolos, configuração, DHCP, SAMBA, NFS, compartilhamento de arquivos com servidores Windows, login remoto, SSH, roteamento, FTP, NTP, DNS e muito mais.



Montagem e Configuração de Computadores - Guia Prático

Autor: Renato Rodrigues Paixão

Código: 3196 • 304 páginas • Formato: 20,5 x 27,5 cm • ISBN: 978-85-365-0319-6 • EAN: 9788536503196

Indicado a estudantes, profissionais e entusiastas, o livro desmistifica o processo de montagem de um computador pessoal detalhando cada etapa, como configuração, testes de performance e interconexão numa rede local (LAN).

Aborda conceitos básicos de eletricidade e eletrônica digital, ferramentas e componentes, partes do PC, periféricos, ambiente de trabalho, aterramento e conexões, modelos de gabinete, motherboard, jumpeamento, overclocking e o processador.

Descreve pentes de memória, conexões a cabos, fonte de alimentação, HDD e FDD, controladoras, cabos internos e externos do gabinete, filtros de linha, estabilizadores de tensão e nobreaks. Ensina como ligar o PC pela primeira vez, configuração do setup e BIOS, janelas principais, boot do sistema, instalação do sistema operacional, placa de vídeo, impressoras, softwares básicos e de monitoramento de sinais, defrag e utilitários de desempenho.



Manutenção de Computadores - Guia Prático

Autor: Renato Rodrigues Paixão

Código: 3226 • 208 páginas • Formato: 20,5 x 27,5 cm • ISBN: 978-85-365-0322-6 • EAN: 9788536503226

Estudantes, entusiastas e profissionais da área encontram neste livro um conteúdo didático e objetivo sobre os cuidados necessários para a manutenção preventiva e corretiva de um PC.

De forma gradativa, aborda conceitos básicos de eletricidade, instrumentos de medida, principais componentes que integram um PC, microprocessadores, barramento, arquitetura interna, encapsulamentos, conexão do processador com a motherboard, memórias, chipsets, motherboards, drives (HDD, FDD, CD/DVD), conexão da BIOS, setup e controladoras de vídeo, como também aspectos práticos da manutenção, ferramentas e sobressalentes, ambiente de trabalho, cargas eletrostáticas, aterramento, conexões, filtros de linha, estabilizadores de tensão e nobreaks, cuidados com o PC, acessórios, limpeza, tipos de falha e solução de problemas.