

Administração de Sistemas Linux

Ari Frazão Jr.
Marcelo Braga

A RNP – Rede Nacional de Ensino e Pesquisa – é qualificada como uma Organização Social (OS), sendo ligada ao Ministério da Ciência, Tecnologia e Inovação (MCTI) e responsável pelo Programa Interministerial RNP, que conta com a participação dos ministérios da Educação (MEC), da Saúde (MS) e da Cultura (MinC). Pioneira no acesso à Internet no Brasil, a RNP planeja e mantém a rede Ipê, a rede óptica nacional acadêmica de alto desempenho. Com Pontos de Presença nas 27 unidades da federação, a rede tem mais de 800 instituições conectadas. São aproximadamente 3,5 milhões de usuários usufruindo de uma infraestrutura de redes avançadas para comunicação, computação e experimentação, que contribui para a integração entre o sistema de Ciência e Tecnologia, Educação Superior, Saúde e Cultura.



Ministério da
Cultura

Ministério da
Saúde

Ministério da
Educação

Ministério da
**Ciência, Tecnologia
e Inovação**



Administração de Sistemas Linux

Ari Frazão Jr.
Marcelo Braga



Administração de Sistemas Linux

Ari Frazão Jr.
Marcelo Braga

Rio de Janeiro
Escola Superior de Redes
2015

Copyright © 2015 – Rede Nacional de Ensino e Pesquisa – RNP
Rua Lauro Müller, 116 sala 1103
22290-906 Rio de Janeiro, RJ

Diretor Geral
Nelson Simões

Diretor de Serviços e Soluções
José Luiz Ribeiro Filho

Escola Superior de Redes

Coordenação
Luiz Coelho

Edição
Lincoln da Mata

Revisão Técnica
Bruno Alves Fagundes

Coordenação Acadêmica de Administração de Sistemas
Renato Duarte

Equipe ESR (em ordem alfabética)
Adriana Pierro, Celia Maciel, Cristiane Oliveira, Derlinéa Miranda, Edson Kowask, Elimária Barbosa, Evellyn Feitosa, Felipe Nascimento, Lourdes Soncin, Luciana Batista, Luiz Carlos Lobato e Yve Marcial

Capa, projeto visual e diagramação
Tecnodesign

Versão
2.0.0

Este material didático foi elaborado com fins educacionais. Solicitamos que qualquer erro encontrado ou dúvida com relação ao material ou seu uso seja enviado para a equipe de elaboração de conteúdo da Escola Superior de Redes, no e-mail info@esr.rnp.br. A Rede Nacional de Ensino e Pesquisa e os autores não assumem qualquer responsabilidade por eventuais danos ou perdas, a pessoas ou bens, originados do uso deste material.
As marcas registradas mencionadas neste material pertencem aos respectivos titulares.

Distribuição
Escola Superior de Redes
Rua Lauro Müller, 116 – sala 1103
22290-906 Rio de Janeiro, RJ
<http://esr.rnp.br>
info@esr.rnp.br

Dados Internacionais de Catalogação na Publicação (CIP)

F848a Frazão Júnior, Ari
Administração de sistemas Linux / Ari Frazão Júnior, Marcelo Braga. – 2. ed. – Rio de Janeiro: RNP/ESR, 2015.
180 p. : il. ; 28 cm.

ISBN 978-85-63630-52-0

1. Linux (Sistema operacional de computador) – administração. 2. Sistema operacional (computadores). I. Braga, Marcelo. II. Título.

CDD 005.432

Sumário

Escola Superior de Redes

A metodologia da ESR ix

A quem se destina x

Convenções utilizadas neste livro xi

Permissões de uso xi

Comentários e perguntas xii

Sobre os autores xii

1. Introdução ao Sistema Operacional Linux

Introdução 1

Atribuições de um administrador de sistemas 2

Histórico 3

Slackware 5

Red Hat Enterprise Linux 5

Debian 5

Suse 5

Fedora 5

CentOS 5

Ubuntu 6

Arquitetura 6

Kernel 6

Biblioteca de funções padrão 8

Shell 9

Aplicações 9



Sistema de arquivos	10
Inode	11
Tipos de arquivos	12
Arquivo regular	12
Diretório	12
Arquivo de dispositivo	13
Socket	13
Named pipe	14
Link	14
Vantagens e desvantagens	17
Comando ls	17
Permissões de arquivos	18
Bits de permissão	18
Bits especiais	19
Mudando permissões	21
Máscara de usuário	22

2. Usuários e grupos

Introdução	25
Grupos	26
Arquivo /etc/gshadow	27
Usuários	28
Problemas de segurança	29
Shadow passwords	29
Tipos de contas de usuários	30
Senhas	31
O risco das senhas prováveis	31
Criando contas de usuários	31
Criar entrada nos arquivos /etc/group e /etc/gshadow	32
Criar entrada nos arquivos /etc/passwd e /etc/shadow	32
Definir senha inicial	34
Criar diretório de trabalho	34
Copiar arquivos de inicialização	34
Testar nova conta	35
Administrando grupos	35
Criando grupos	36
Modificando grupos	36
Removendo grupos	36

Administrando contas de usuários 37

Criando contas de usuários 37

Modificando contas de usuários 38

Desabilitando contas de usuários 38

Removendo contas de usuários 39

Monitorando usuários 39

3. Processos

Processos 41

Tipos de processos 42

Componentes de um processo 43

Ciclo de vida de um processo 45

Multiprocessamento 47

Estados de um processo 49

Prioridades e escalonamento de processos 51

Uso de sinais 53

Monitoramento de processos 54

Execução periódica de tarefas 56

Cron 56

Anacron 58

Fcron 58

4. Sistema de arquivos

Estrutura dos discos 59

Particionamento 60

Tipos de partição 61

Sistema de arquivos 61

Comandos 63

Tipos de sistemas de arquivos suportados 66

Sistema de quotas 67

Preparando o sistema 67

Habilitando 67

Inicializando 68

Editando quotas 69

Verificação e gerenciamento 71

5. Backup

Introdução 73

Tipos de backup 74

Mídias de backup 75

Fitas magnéticas 75

Mídias ópticas 76

Storage 77

Comandos do sistema 79

Comando tar 79

Comando cpio 80

Comando dump 80

Comando rsync 81

Comandos gzip/gunzip e bzip2/bunzip2 81

Softwares de backup 82

Pacotes gratuitos 82

Pacotes comerciais 83

Políticas de backup 84

Cuidados no armazenamento das mídias 85

6. Serviço de impressão

Introdução 87

Histórico 87

Arquitetura do sistema de impressão 88

Softwares gerenciadores de impressão 89

LPD (Line Printer Daemon) 90

LPRng 91

CUPS 92

7. Registro de eventos

Introdução 99

Sysklogd 100

Syslogd 100

Klogd 103

Rsyslog 104

Syslog-ng 105

Rotacionamento de arquivos de log	108
Arquivo de configuração do logrotate	109
O comando logger	110
Servidor de logs	110
Configurando os servidores clientes	111
Aplicativos para análise de arquivos de log	112
Logwatch	112
Swatch	113
Logcheck	115
Recomendações básicas de segurança	116

8. Boot & Shutdown e Kernel

Inicialização do sistema	119
Basic Input/Output System (BIOS)	120
Carregadores de boot (boot loaders)	120
Iniciando o kernel	124
Processo init	124
Upstart	127
systemd	129
Shutdown	130
Arquitetura do kernel	131
Kernel Monolítico	131
Kernel Modular ou Híbrido	132
Configurando e compilando o kernel	133
Configurando os componentes do kernel	133
Compilando o kernel	135
Instalando o kernel	136
Habilitando o kernel	136
Testando o kernel	139

9. Segurança básica e procedimentos operacionais

Segurança básica	141
Analísadores de senhas	143
Contas compartilhadas	144
SUID e SGID	145
Atualização de software	145



Procedimentos operacionais 148

Política de utilização de recursos 149

Informações sobre os novos usuários 150

Atendimento aos usuários 151

Controle de recursos computacionais 152

Diagnóstico de falhas 153

Rotina de backup 154

10.Webmin

Características gerais 157

Instalação 158

Perl 159

OpenSSL 159

Net_SSLeay.pm 160

Webmin 161

Interface de administração 162

Módulos 163

Usermin 165

Escola Superior de Redes

A Escola Superior de Redes (ESR) é a unidade da Rede Nacional de Ensino e Pesquisa (RNP) responsável pela disseminação do conhecimento em Tecnologias da Informação e Comunicação (TIC). A ESR nasce com a proposta de ser a formadora e disseminadora de competências em TIC para o corpo técnico-administrativo das universidades federais, escolas técnicas e unidades federais de pesquisa. Sua missão fundamental é realizar a capacitação técnica do corpo funcional das organizações usuárias da RNP, para o exercício de competências aplicáveis ao uso eficaz e eficiente das TIC.

A ESR oferece dezenas de cursos distribuídos nas áreas temáticas: Administração e Projeto de Redes, Administração de Sistemas, Segurança, Mídias de Suporte à Colaboração Digital e Governança de TI.

A ESR também participa de diversos projetos de interesse público, como a elaboração e execução de planos de capacitação para formação de multiplicadores para projetos educacionais como: formação no uso da conferência web para a Universidade Aberta do Brasil (UAB), formação do suporte técnico de laboratórios do Proinfo e criação de um conjunto de cartilhas sobre redes sem fio para o programa Um Computador por Aluno (UCA).

A metodologia da ESR

A filosofia pedagógica e a metodologia que orientam os cursos da ESR são baseadas na aprendizagem como construção do conhecimento por meio da resolução de problemas típicos da realidade do profissional em formação. Os resultados obtidos nos cursos de natureza teórico-prática são otimizados, pois o instrutor, auxiliado pelo material didático, atua não apenas como expositor de conceitos e informações, mas principalmente como orientador do aluno na execução de atividades contextualizadas nas situações do cotidiano profissional.

A aprendizagem é entendida como a resposta do aluno ao desafio de situações-problema semelhantes às encontradas na prática profissional, que são superadas por meio de análise, síntese, julgamento, pensamento crítico e construção de hipóteses para a resolução do problema, em abordagem orientada ao desenvolvimento de competências.

Dessa forma, o instrutor tem participação ativa e dialógica como orientador do aluno para as atividades em laboratório. Até mesmo a apresentação da teoria no início da sessão de aprendizagem não é considerada uma simples exposição de conceitos e informações. O instrutor busca incentivar a participação dos alunos continuamente.

As sessões de aprendizagem onde se dão a apresentação dos conteúdos e a realização das atividades práticas têm formato presencial e essencialmente prático, utilizando técnicas de estudo dirigido individual, trabalho em equipe e práticas orientadas para o contexto de atuação do futuro especialista que se pretende formar.

As sessões de aprendizagem desenvolvem-se em três etapas, com predominância de tempo para as atividades práticas, conforme descrição a seguir:

Primeira etapa: apresentação da teoria e esclarecimento de dúvidas (de 60 a 90 minutos).

O instrutor apresenta, de maneira sintética, os conceitos teóricos correspondentes ao tema da sessão de aprendizagem, com auxílio de slides em formato PowerPoint. O instrutor levanta questões sobre o conteúdo dos slides em vez de apenas apresentá-los, convidando a turma à reflexão e participação. Isso evita que as apresentações sejam monótonas e que o aluno se coloque em posição de passividade, o que reduziria a aprendizagem.

Segunda etapa: atividades práticas de aprendizagem (de 120 a 150 minutos).

Esta etapa é a essência dos cursos da ESR. A maioria das atividades dos cursos é assíncrona e realizada em duplas de alunos, que acompanham o ritmo do roteiro de atividades proposto no livro de apoio. Instrutor e monitor circulam entre as duplas para solucionar dúvidas e oferecer explicações complementares.

Terceira etapa: discussão das atividades realizadas (30 minutos).

O instrutor comenta cada atividade, apresentando uma das soluções possíveis para resolvê-la, devendo ater-se àquelas que geram maior dificuldade e polêmica. Os alunos são convidados a comentar as soluções encontradas e o instrutor retoma tópicos que tenham gerado dúvidas, estimulando a participação dos alunos. O instrutor sempre estimula os alunos a encontrarem soluções alternativas às sugeridas por ele e pelos colegas e, caso existam, a comentá-las.

Sobre o curso

Seu objetivo é apresentar os conceitos básicos, história e arquitetura do sistema operacional Linux, com práticas de instalação, configuração e administração do sistema de parâmetros, senhas e contas de usuários, backup, serviço de impressão, registro de eventos – syslog, monitoramento de usuários, contabilidade de processos, configuração do núcleo do sistema operacional, aplicação de noções básicas de segurança, entre outras atividades relacionadas à maior comunidade de software livre do mercado. O curso é composto de 10 capítulos de embasamento teórico e atividades correlatas para aprendizado e fixação do conhecimento. O curso tem como objetivo apresentar as facilidades de administração e gerenciamento, que serão exploradas com maior profundidade nos demais cursos da área de Administração de Sistemas da Escola Superior de Redes da RNP.

A quem se destina

Profissionais da área de informática que serão responsáveis por administrar ambientes que utilizam a infraestrutura com sistemas operacionais Linux. Podem também participar outros profissionais de TI, programadores e analistas de suporte de sistemas.

Convenções utilizadas neste livro

As seguintes convenções tipográficas são usadas neste livro:

Itálico

Indica nomes de arquivos e referências bibliográficas relacionadas ao longo do texto.

Largura constante

Indica comandos e suas opções, variáveis e atributos, conteúdo de arquivos e resultado da saída de comandos. Comandos que serão digitados pelo usuário são grifados em negrito e possuem o prefixo do ambiente em uso (no Linux é normalmente # ou \$, enquanto no Windows é C:\).

Conteúdo de slide

Indica o conteúdo dos slides referentes ao curso apresentados em sala de aula.

Símbolo

Indica referência complementar disponível em site ou página na internet.

Símbolo

Indica um documento como referência complementar.

Símbolo

Indica um vídeo como referência complementar.

Símbolo

Indica um arquivo de áudio como referência complementar.

Símbolo

Indica um aviso ou precaução a ser considerada.

Símbolo

Indica questionamentos que estimulam a reflexão ou apresenta conteúdo de apoio ao entendimento do tema em questão.

Símbolo

Indica notas e informações complementares como dicas, sugestões de leitura adicional ou mesmo uma observação.

Permissões de uso

Todos os direitos reservados à RNP.

Agradecemos sempre citar esta fonte quando incluir parte deste livro em outra obra.

Exemplo de citação: TORRES, Pedro et al. *Administração de Sistemas Linux: Redes e Segurança*.

Rio de Janeiro: Escola Superior de Redes, RNP, 2013.

Comentários e perguntas

Para enviar comentários e perguntas sobre esta publicação:

Escola Superior de Redes RNP

Endereço: Av. Lauro Müller 116 sala 1103 – Botafogo

Rio de Janeiro – RJ – 22290-906

E-mail: info@esr.rnp.br

Sobre os autores

Ari Frazão Jr. é bacharel em Ciência da Computação pela Universidade Federal da Paraíba (UFPB) e mestre em Ciência da Computação, na área de redes de computadores, pela Universidade Federal de Pernambuco (UFPE). Atualmente é responsável pelas áreas de engenharia e operações da Rede Nacional de Ensino e Pesquisa (RNP), onde atua desde 1993.

Marcelo Castellan Braga possui graduação em Engenharia Eletrônica pelo CEFET-RJ, pós-graduação em Análise, Projeto e Gerência de Sistemas pela PUC-RJ e mestrado em informática pela UNIRIO. Atualmente é sócio diretor da MCB Tech, empresa que presta consultoria em redes de computadores, serviços de internet, segurança de dados e desenvolvimento de software. Atuou durante mais de 10 anos na área de TI em empresas como Rede Nacional de Ensino e Pesquisa (RNP) e Embratel.

Bruno Fagundes Especialista em Segurança de Redes com mais de 7 anos de experiência em administração de sistemas Linux. Atualmente é Tecnologista no Laboratório Nacional de Computação Científica atuando no suporte e administração da plataforma de alto desempenho e gerenciamento de serviços de rede. Professor no Instituto Superior de Tecnologia de Petrópolis em 2014. Conteudista e Instrutor da Escola Superior de Redes nos cursos de Administração de sistemas Linux.

1

Introdução ao Sistema Operacional Linux

objetivos

Conhecer o histórico do Unix e do Linux; Entender a arquitetura do Linux;
Aprender o conceito de sistema de arquivos; Conhecer os tipos de arquivos do Linux;
Entender o conceito de permissão de arquivos.

Arquitetura; Sistema de arquivos; Tipos de arquivos; Permissões de arquivos.

conceitos

Introdução

Requisitos do administrador de sistemas:

- Responsabilidade pelos recursos computacionais.
- Conhecimento técnico.
- Experiência.
- Aperfeiçoamento constante.
- Entendimento dos objetivos e metas da empresa.
- Conhecimento das necessidades dos usuários.



Uma das características mais marcantes do ser humano é a capacidade de se organizar para tirar o melhor proveito de seus trabalhos e garantir que tudo funcione de maneira correta. Isso pode ser observado em empresas, escritórios, lojas, cinemas e até mesmo nos lares. Da mesma forma que um administrador de empresas administra e gerencia uma empresa, um administrador de sistemas é responsável por administrar e gerenciar os recursos computacionais de uma organização.

A informatização crescente nas instituições públicas e privadas, a disseminação de diversas tecnologias de redes e o uso cada vez maior de sistemas integrados de gestão fazem da administração de sistemas uma atividade complexa e de importância estratégica para as organizações. Portanto, para atuar nessa área, o administrador de sistemas deve possuir o conhecimento e a experiência necessários para assegurar que os sistemas de uma organização estejam sempre disponíveis, seguros e com desempenho adequado para executar as operações de que a organização necessita para atingir seus objetivos. Um bom administrador de sistemas deve aliar conhecimento técnico ao do negócio da empresa para conseguir tirar o melhor proveito possível dos recursos computacionais sob sua responsabilidade.



Apesar de existir desde a época dos mainframes, a área de administração de sistemas sofreu um grande impacto com o advento da microinformática, das redes de computadores e das tecnologias associadas à internet. Esse módulo, de caráter essencialmente prático, foi criado pela Escola Superior de Redes para explorar os conceitos e práticas mais importantes da administração de sistemas Linux. Ele representa a primeira etapa na formação de um administrador de sistemas.

Atribuições de um administrador de sistemas

Atribuições de um administrador:

- Instalar, configurar e manter o hardware dos equipamentos.
- Instalar, configurar e manter atualizado e seguro os softwares utilizados pela empresa.
- Interligar de maneira eficaz os recursos computacionais que funcionam em rede.
- Administrar contas de usuários (cadastro, atualização e remoção).
- Fazer e restaurar backup.
- Monitorar atividades do sistema, ajustando-as para o máximo desempenho.
- Solucionar problemas de software e hardware.
- Manter documentação atualizada sobre os sistemas.
- Desenvolver scripts para automação de tarefas.
- Garantir a segurança de rede de um modo geral.
- Prestar atendimento aos usuários dos sistemas.

Um administrador de sistemas deve ser capaz de:

- Instalar, configurar e manter operacional toda a infraestrutura de TI (servidores, estações de trabalho, impressoras, dispositivos de armazenamento, equipamentos de conectividade etc.);
- Instalar, configurar, manter atualizado e seguro todo o software necessário para o funcionamento correto dos sistemas (Sistemas Operacionais e aplicativos);
- Interligar os recursos computacionais que funcionam em rede, configurando os equipamentos de modo adequado;
- Elaborar e executar uma política de backup adequada;
- Elaborar políticas de uso dos recursos de TI;
- Executar e controlar as operações de gerenciamento de contas de usuários nos sistemas corporativos, bem como suas permissões de acesso aos recursos desses sistemas;
- Controlar e supervisionar o uso dos recursos computacionais para assegurar que sejam utilizados de forma segura e adequada;
- Diagnosticar e solucionar as situações de falha no funcionamento e na operação dos recursos computacionais que compõem os sistemas;
- Auxiliar os usuários, esclarecendo dúvidas e orientando-os na busca de soluções mais adequadas às suas necessidades.

É claro que existem sistemas computacionais de portes e fins muito diferentes, variando desde uma pequena rede local para uso administrativo em uma microempresa até uma rede de computadores abrangendo diversos países, em uma empresa multinacional. Naturalmente, o porte e o conhecimento da equipe de administração de sistemas deverão estar adaptados a essas diferentes situações, embora suas atribuições sejam basicamente as já descritas.



Entre as diversas características desejáveis em um administrador de sistemas, podemos destacar:

- Conhecimento técnico aprofundado e abrangente da área, o que requer constante atualização;
- Entendimento dos objetivos e metas da instituição;
- Conhecimento das necessidades dos usuários;
- Capacidade de diagnosticar e resolver problemas, o que depende de sua experiência prática;
- Competência para trabalhar em equipe e de se relacionar com pessoas, para proporcionar um bom atendimento aos usuários;
- Capacidade de organização;
- Proatividade e senso de responsabilidade, para assegurar que todas as tarefas programadas sejam adequadamente priorizadas e executadas.



Sem isso, a carreira profissional de um administrador de sistemas estará seriamente comprometida.

Histórico

1964:

- AT&T, GE e MIT – Projeto para criação de um novo Sistema Operacional (Multics).

1969:

- Início do Unix na Bell Labs da AT&T (Thompson).

1973:

- Unix reescrito na linguagem C.
- Novo código-fonte.

1977:

- Início da distribuição BSD.



MIT

Massachusetts Institute of Technology. É um centro universitário de educação e pesquisa privado em Cambridge, Massachusetts, nos EUA. Um dos líderes mundiais em ciência, engenharia e tecnologia.

Em meados dos anos 60, mais precisamente em 1964, o Bell Labs, da AT&T, a GE e o MIT formaram um grupo de pesquisadores para desenvolver um Sistema Operacional para o mainframe GE-645, que foi batizado de Multics. Esse projeto, no entanto, não foi muito bem-sucedido e, devido a interesses divergentes, o Bell Labs o abandonou em 1969. Nesse mesmo ano, Ken Thompson, um de seus pesquisadores envolvido no projeto de criação do Multics, iniciou o projeto de um novo Sistema Operacional baseado no Multics e escrito em linguagem assembly. Esse sistema foi inicialmente batizado de Unics e mais tarde teve seu nome alterado para Unix.

Em 1973, com a ajuda de Dennis Ritchie, outro pesquisador do Bell Labs, Thompson reescreveu o Unix em uma linguagem de programação de alto nível, chamada C, que foi desenvolvida pelo próprio Ritchie. Como a AT&T não atuava comercialmente na área de computação, o Bell Labs fornecia a licença de uso do Unix para as universidades interessadas, juntamente com o código-fonte. Isso gerou versões diferentes do sistema, à medida que as modificações no código eram feitas nas universidades e no próprio Bell Labs. Essa falta de padronização foi tão grande que, no final dos anos 80, havia várias versões de Unix totalmente incompatíveis, baseadas em duas distribuições principais: o System V (da AT&T) e o BSD, da Universidade da Califórnia em Berkeley. A distribuição BSD contribuiu para a disseminação da internet por ter desenvolvido e implementado a pilha de protocolos TCP/IP em seu sistema.



Década de 80:

- System V (AT&T).
- Incorporação do protocolo TCP/IP à distribuição BSD (Berkeley).
- Diferentes versões geradas pela distribuição do código.
- Padronização básica Portable Operating System Interface (POSIX) nos padrões IEEE.
- Disputa comercial gerou: AIX, HP-UX, XENIX, IRIX e SunOS, entre outros.
- Início do projeto GNU de Richard Stallman.
- Linux (System V) é uma alternativa aos sistemas comerciais.
 - ▣ Características:
 - ▣ Segurança.
 - ▣ Confiabilidade.
 - ▣ Versatilidade.
 - ▣ Estabilidade.
 - ▣ Licença gratuita.

Embora a necessidade de padronização fosse muito grande, as tentativas feitas nesse sentido falharam. Em 1988, o Institute of Electrical and Electronics Engineers (IEEE) elaborou a proposta do Portable Operating System Interface (POSIX), que permitiu uma padronização básica das muitas versões existentes. No entanto, diferenças continuaram existindo devido aos interesses comerciais dos grandes fabricantes de computadores, originando diversas versões comerciais do Unix, como: AIX, da IBM; HP-UX, da HP; XENIX, da Microsoft, e IRIX, da Silicon Graphics, todas baseadas na distribuição System V e SunOS, da Sun, e Tru64, da Digital Equipment Corporation, baseadas na distribuição BSD. Além disso, surgiram versões livres de Unix como FreeBSD, NetBSD e OpenBSD, todas com base na distribuição BSD.

Nesse contexto, em 1984, Richard Stallman começou um projeto para criar um Sistema Operacional compatível com o Unix, batizado de GNU. Stallman, junto com um grupo de programadores, começou a desenvolver os principais componentes do sistema, como compiladores, editores de texto etc.

Década de 90:

- Linus Torvalds desenvolve o kernel do Linux.
- Criação do Sistema Operacional GNU/Linux.
- Versões livres baseadas na distribuição BSD: FreeBSD, OpenBSD e NetBSD.
- Primeiras distribuições do Linux (Slackware, Red Hat, Debian, SUSE etc.).

Em 1991, o GNU já possuía diversos componentes, mas ainda não tinha um kernel funcional. Em paralelo a isso, em 1990 um então estudante de ciência da computação, o finlandês Linus Torvalds começou a desenvolver um sistema próprio, baseado no MINIX, um Sistema Operacional desenvolvido por Andrew Tanenbaum, baseado no Unix e utilizado em cursos de computação nas universidades. Linus estava insatisfeito com o MINIX e começou seu projeto inicialmente para rodar somente em seu computador, um 80386. Em pouco tempo Linus já tinha uma versão de kernel funcional e em 1991 resolveu divulgar à comunidade seu projeto, que foi batizado de Linux. O Linux era basicamente um kernel, e o GNU, um conjunto de aplicações. Em pouco tempo os dois projetos se juntaram, formando o Sistema Operacional GNU/Linux (muitas vezes referenciado somente por Linux).

Hoje o Linux possui diversas distribuições comerciais e gratuitas, cada uma com suas características. Entre elas podemos destacar: Slackware, Red Hat Enterprise Linux, Debian, SUSE, Fedora, CentOS e Ubuntu. As distinções entre as principais distribuições são resumidas a seguir.

Slackware

É a distribuição mais antiga ainda em desenvolvimento, tendo sua primeira versão lançada em 1993. O slackware é uma distribuição gratuita que tem como principais características: estabilidade, segurança, uso da interface texto para configuração do sistema, ampla documentação etc. Possui uma versão de produção que só inclui pacotes estáveis e uma versão current em desenvolvimento que não é disponibilizada para download.

Red Hat Enterprise Linux

Desenvolvida pela Red Hat Software, essa é uma distribuição voltada para uso em servidores de pequeno a grande porte, com versões que suportam de dois a um ilimitado número de processadores. A Red Hat vende essa distribuição juntamente com uma assinatura de suporte técnico, que varia em função dos dias de atendimento, vias de atendimento (web ou telefone) e número de incidentes por ano.

Debian

A distribuição Debian é desenvolvida e mantida por uma equipe de desenvolvedores voluntários. O Debian possui um ciclo de desenvolvimento onde suas versões passam por três fases: stable (que é a versão de produção), testing (contém versões mais atuais dos pacotes que ainda não foram homologadas para entrar em produção) e unstable (usada geralmente por desenvolvedores). O Debian é conhecido por sua excelente ferramenta de gerenciamento de pacotes, o APT.

Suse

É uma das distribuições comerciais mais antigas (teve sua primeira versão lançada em 1994, na Alemanha). A distribuição SUSE possui uma excelente ferramenta gráfica de instalação e configuração do Sistema Operacional, chamada YaST. Em 2003, a Novell comprou a SUSE e decidiu em 2005 compartilhar o desenvolvimento de uma versão gratuita de seu Sistema Operacional com a comunidade de desenvolvedores, criando a distribuição openSUSE. Em 2011, a Novell e a SUSE foram compradas pela empresa Attachmate.

Fedora

É uma distribuição gratuita patrocinada pela Red Hat e mantida por uma comunidade aberta de usuários e desenvolvedores. Essa distribuição teve sua primeira versão lançada em 2003, quando a distribuição gratuita Red Hat Linux foi descontinuada. A partir daí, a Red Hat Software resolveu manter seu foco no mercado corporativo e lançou a distribuição paga Red Hat Enterprise Linux. Em paralelo a isso, foi lançado o projeto Fedora, uma distribuição gratuita baseada no Red Hat Linux.

CentOS

Essa distribuição deriva diretamente do Red Hat Enterprise Linux através dos seus arquivos fonte originais. A Red Hat permite o uso dos fontes na condição de que se retire todas as referências comerciais à sua marca. A principal vantagem é que o CentOS é uma sólida e poderosa distribuição Linux com o mesmo potencial do Red Hat e com a vantagem de ser gratuita.

Ubuntu

O Ubuntu é uma distribuição baseada na distribuição Debian, patrocinada pela empresa Canonical. Uma das principais propostas do Ubuntu é a popularização do Linux, oferecendo um sistema com interface mais amigável, com foco em acessibilidade e internacionalização.

Segurança, confiabilidade, versatilidade, estabilidade e gratuidade são as características do Linux que fazem com que ele desponte como a principal alternativa ao Microsoft Windows.

Arquitetura

O GNU/Linux ou Linux (como é mais conhecido) é um Sistema Operacional multitarefa, multiusuário e multiplataforma. Seu código-fonte é aberto e disponibilizado gratuitamente, podendo ser alterado por qualquer pessoa, o que o torna um sistema extremamente flexível. O Linux é distribuído pela licença General Public License (GPL), que permite executar, alterar e distribuir qualquer software regido por ela. Sua arquitetura, como mostra figura 1.1, pode ser representada por uma pirâmide dividida em camadas. O papel de cada uma dessas camadas é descrito a seguir.

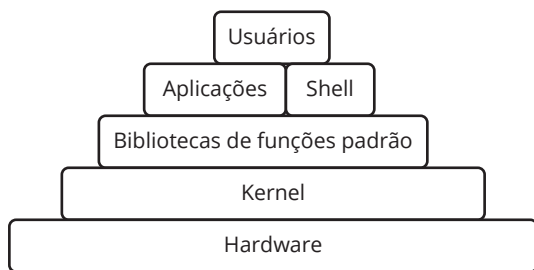


Figura 1.1
Arquitetura.

Kernel

- Núcleo do Sistema Operacional.
- Principais funções:
 - ▣ Detecção de hardware.
 - ▣ Gerenciamento de entrada e saída.
 - ▣ Manutenção do sistema de arquivos.
 - ▣ Gerenciamento de memória e swapping.
 - ▣ Controle da fila de processos.



O kernel é o núcleo do Sistema Operacional e se encarrega de executar todas as funções básicas e necessárias ao funcionamento correto do sistema. Uma de suas principais funções é prover uma interface entre o hardware e as aplicações. O kernel possui uma arquitetura monolítica, ou seja, é composto por um grande e único bloco de código com milhões de linhas. Há algum tempo o conceito de módulos foi introduzido no Linux. Esses módulos são geralmente drivers de dispositivos e podem ser carregados em memória dinamicamente. É importante ressaltar que os módulos não fazem parte do kernel.

Uma das grandes vantagens que o Linux oferece é a possibilidade de o usuário poder fazer alterações em seu kernel, habilitando somente as funcionalidades necessárias para cada sistema. Com isso, é possível ganhar em performance, pois o kernel se torna muito mais enxuto, gerando uma imagem com um tamanho consideravelmente menor. As principais funções do kernel são:

Saiba mais

Para evitar esse tipo de problema, é importante usar sempre a última versão estável do kernel. Vale ressaltar que drivers desenvolvidos como módulos podem ser instalados no sistema sem a necessidade de recompilar o kernel. Os módulos podem ser carregados em memória dinamicamente à medida que seu uso se torna necessário.

- Detecção de hardware;
- Gerenciamento de entrada e saída;
- Manutenção do sistema de arquivos;
- Gerenciamento de memória e swapping;
- Gerenciamento de processos.

Hoje, todas as distribuições de Linux disponíveis usam basicamente o mesmo kernel, com pequenas alterações. São as aplicações incorporadas a essas distribuições que as diferenciam.

Detecção de hardware

- Identificação de dispositivos (memória, discos, processadores, impressora etc.).
- Interação com esses dispositivos (drivers).



Todo Sistema Operacional, para funcionar, depende de um determinado hardware, composto por processador, memória, disco, controladores de vídeo etc. Devido à existência de diversos fabricantes de hardware no mercado, as características dos componentes variam muito entre si. Quando um novo dispositivo, como, por exemplo, uma placa de rede, é instalado no sistema, o kernel é responsável pela detecção e interação básica com esse dispositivo. Embora o kernel possa reconhecer e controlar uma grande quantidade de dispositivos disponíveis no mercado, existem alguns que não são reconhecidos, em geral os lançados após a data de lançamento da versão do kernel utilizado.

Gerenciamento de entrada e saída

- Controle dos dispositivos de entrada e saída.
- Envio de requisições solicitando execuções de operações.
- Controle de sinais de interrupção.



Todos os computadores possuem dispositivos de entrada e saída conectados, como teclado, monitor, impressora, placa de rede, disco etc. Esses dispositivos são controlados pelo kernel, que envia requisições para solicitar a execução de operações específicas ou recebe sinais para indicar que os dispositivos estão demandando determinadas operações. A comunicação entre o kernel e os dispositivos é realizada por meio de sinais de interrupção. Nesse contexto, o kernel funciona como um controlador de sinais de interrupção, atendendo a todas essas requisições.

Manutenção do sistema de arquivos

- É o responsável pela organização dos arquivos do sistema e dos usuários;
- Forma de implementação definida e gerenciada pelo kernel.



O sistema de arquivos tem por objetivo organizar os arquivos do sistema e dos usuários, assegurando que eles possam ser manipulados adequadamente por seus proprietários. No Linux, o sistema de arquivos é visualizado como uma árvore invertida: a raiz está no topo e os ramos, embaixo. Para ser lido ou escrito, o arquivo precisa ser aberto. Ao abri-lo, uma série de cuidados devem ser tomados, principalmente se esse arquivo já estiver sendo utilizado por outro usuário ou programa. Todos esses cuidados com o sistema de arquivos, bem como a forma como ele é implementado, são definidos e gerenciados pelo kernel. A figura 1.2 mostra um exemplo de organização do sistema de arquivos do Linux.

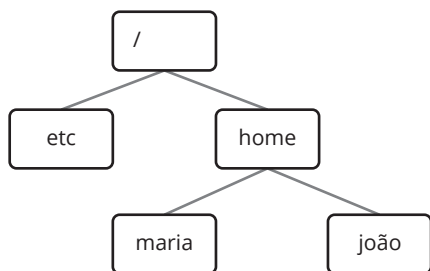


Figura 1.2
Sistema de arquivos
do Linux.

Gerenciamento de memória e swapping

- Responsável pela alocação de memória aos processos em execução.
- Suporta o conceito de memória virtual.
- Permite que os processos compartilhem áreas de memória.



Ao longo do tempo, foram criadas várias técnicas, como paginação e swapping, para otimizar o uso da memória pelos programas em execução. O kernel é responsável pela alocação de memória aos processos em execução. Ele suporta o conceito de memória virtual, permitindo que processos ocupem mais espaço em memória que aquela disponível na máquina. A memória virtual pode ser muito maior que a memória física. Cada processo tem seu próprio espaço de endereçamento virtual. Esses espaços de endereçamento são completamente separados, de modo que um processo não pode interferir no código e nos dados de outro processo. Além disso, o kernel permite que processos compartilhem áreas de memória, reduzindo assim o consumo desse recurso ou viabilizando um mecanismo de comunicação entre processos.

O swapping é o processo em que o Sistema Operacional transfere dados que estão na memória principal (programa em execução e os seus dados associados) para uma área em disco (memória virtual) e vice-versa, dando a impressão de que o sistema possui uma área de memória maior que a real. Essa área em disco utilizada como memória virtual faz parte da partição de swap do Linux. A paginação é uma técnica utilizada por Sistemas Operacionais que fazem uso do conceito de memória virtual, que divide a área de memória em “páginas”, de forma a permitir o swapping.

Gerenciamento de processos

- Suporta a execução simultânea de vários processos.
- Funciona como um supervisor, autorizando o uso dos recursos entre os processos.



O Linux é um Sistema Operacional que suporta a execução simultânea de vários processos que compartilham os recursos do sistema. O compartilhamento desses recursos deve ser organizado de forma a atender às necessidades de todos os processos. Nesse contexto, o kernel funciona como um supervisor e autoriza o uso desses recursos para um determinado processo, quando necessário. Mais à frente, veremos que há um parâmetro especial chamado de prioridade, observado pelo kernel na alocação de um recurso a um processo.

Biblioteca de funções padrão

- Chamadas feitas por processos ao Sistema Operacional para acessar recursos.
- Uso de funções padrão (open, close, read, write etc.).
- Comunicação entre as aplicações e o núcleo do Sistema Operacional, constituindo a biblioteca de funções padrão.



Quando o usuário executa um comando através do shell, normalmente é iniciado um ou mais processos. Muitas vezes, esses processos realizam chamadas ao Sistema Operacional para acessar recursos gerenciados pelo kernel. Tais chamadas são feitas por meio de funções padrão suportadas pelo kernel. Esses acessos, no entanto, não podem ser feitos de forma desorganizada, pois comprometeriam a própria segurança do sistema. Para isso, existe uma série de funções (open, close, read, write, fork etc.) que realizam, de forma segura, a comunicação entre as aplicações e o kernel, constituindo a biblioteca de funções padrão.

Shell

- Lê e interpreta comandos de entrada de um terminal.
- Cria novos processos à medida que são requisitados.
- Permite ao usuário trocar de interpretador durante sessão.
- Exemplos: sh, csh, bash, ksh, tcsh etc.



O shell ou interpretador de comandos nada mais é do que um processo responsável por ler os comandos de entrada de um terminal, interpretá-los e criar novos processos à medida que vão sendo requisitados. Baseado na configuração do ambiente do usuário, o interpretador de comandos é iniciado logo após o processo de login. Veremos, mais adiante, que cada usuário pode utilizar seu próprio interpretador de comandos. Da mesma forma que o MS-DOS, que apresenta a sequência de caracteres `c:>` (também conhecida como prompt), o interpretador de comandos também exibe uma mensagem indicando que está pronto para executar comandos. Vale ressaltar que essa mensagem pode ser configurada pelo próprio usuário.

O Linux também suporta um ambiente gráfico baseado em janelas, denominado X-Window. Esse ambiente possui diversas aplicações gráficas, entre elas o xterm, que emula um interpretador de comandos. Quando o usuário inicia o xterm, esse executa um interpretador de comandos que será responsável por interpretar os comandos digitados pelo usuário na janela do xterm. Para definir qual interpretador de comando será executado, o xterm verifica inicialmente a variável de ambiente SHELL. Caso esta não tenha sido definida, o xterm verifica no arquivo `passwd` qual interpretador de comandos está definido para o usuário. Se não for definido nenhum interpretador válido, o xterm utiliza o shell sh.

Saiba mais

O usuário pode trocar de interpretador durante sua sessão de trabalho. Para isso, basta digitar na linha de comando o nome do interpretador que deseja utilizar.

Quando o usuário digita um comando, o interpretador identifica o programa pela primeira palavra informada na linha de comando. O interpretador procura então por um arquivo com o nome do programa e, caso encontre, inicia sua execução. Os demais termos da linha de comando são passados como parâmetros para o programa. No universo Linux, há vários interpretadores disponíveis (como: sh, csh, bash, ksh, ash, zsh etc.). Cada usuário pode escolher e trabalhar com o interpretador mais adequado às suas necessidades. O interpretador de comandos padrão definido para cada usuário é mantido em um arquivo que é a base de usuários do sistema. Esse arquivo é consultado durante o processo de login.

Aplicações

Programas com os quais os usuários interagem:

- Compiladores.
- Editores de texto.
- Planilhas.
- Jogos etc.



São os programas com os quais o usuário interage diretamente, como compiladores, editores de texto, planilhas, navegadores, gerenciadores de arquivos etc. Um Sistema Operacional sem aplicações ficaria sem utilidade, já que estas fazem parte da única camada da arquitetura que é acessada pelos usuários. As aplicações são desenvolvidas geralmente para automatizar e facilitar a realização de tarefas que precisam ser realizadas pelos usuários diariamente. Elas também facilitam a comunicação entre pessoas, além de reduzir custos, como é o caso das aplicações de voz sobre IP.

Sistema de arquivos

Bloco de boot:

- Tem a função de carregar o kernel do Sistema Operacional;
- Toda partição contém um bloco de boot.

Superbloco:

- Contém informações essenciais sobre o sistema de arquivos.

Tabela de blocos:

- Identifica os blocos de dados livres no disco.

Tabela de inodes:

- Contém as informações de cada inode do sistema de arquivos.

Blocos de dados:

- São os blocos do disco destinados a armazenar o conteúdo dos arquivos.

Em qualquer Sistema Operacional, é necessário armazenar dados em arquivos e organizá-los em diretórios. Os arquivos são gerenciados por um componente do Sistema Operacional conhecido como sistema de arquivos, que define como os arquivos são estruturados, identificados, acessados, modificados e protegidos. A figura 1.3 mostra como estão dispostos os blocos em uma partição do Linux. O bloco de boot é o primeiro bloco do disco, e não é gerenciado pelo sistema de arquivos, sendo utilizado somente para dar início ao processo de boot do Sistema Operacional.

O segundo bloco é chamado superbloco e contém informações essenciais sobre o sistema de arquivos, como seu tamanho, número de inodes, número de blocos de dados etc.

Em seguida, vem a tabela de blocos, que permite identificar os blocos de dados livres e ocupados no disco. Logo após, encontra-se a tabela de inodes (index-nodes), que são estruturas de dados que armazenam informações sobre os arquivos. Por fim, temos os blocos de dados, nos quais o conteúdo dos arquivos é efetivamente armazenado.

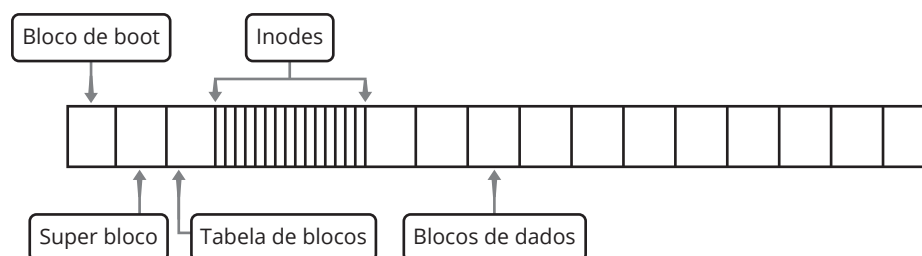


Figura 1.3
Sistemas de
Arquivos Linux.

Inode

Estrutura de dados mantida pelo kernel.

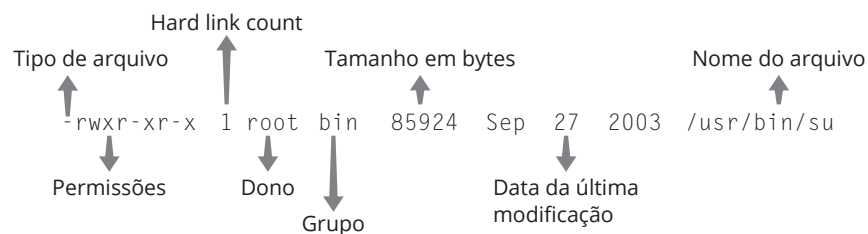
Possui informações sobre arquivos:

- Tipo, dono, grupo e permissões.
- Datas de criação ou da última modificação.
- Número de links para o arquivo.
- Tamanho.
- Endereço no disco rígido.



É uma estrutura de dados que armazena informações sobre um arquivo como: tipo, permissões associadas, proprietário e grupo, tamanho, última vez em que foi modificado, localização dos blocos onde o conteúdo do arquivo está armazenado etc. Todo sistema de arquivos possui uma tabela de inodes e cada arquivo possui um inode associado que é identificado por um número inteiro, conhecido como *i-number* ou *ino*. O número de inodes é função do tamanho do sistema de arquivos e, por padrão, é criado um inode para cada 2 KB do tamanho total do sistema de arquivos. Parte das informações presentes nos inodes pode ser visualizada por meio do comando `ls`, utilizando a opção `-l`, como mostra a figura 1.4.

Figura 1.4
Informações
contidas nos
inodes.



O comando `stat` também pode ser utilizado para visualizarmos informações contidas nos inodes, como mostra o exemplo a seguir.

```
# stat /etc/hosts
File: '/etc/hosts'
Size: 250    Blocks: 8    IO Block: 4096 regular file
Device: 801h/2049d    Inode: 523277    Links: 1
Access: (0644/-rw-r--r--)  Uid: (0/  root)  Gid: (0/  root)
Access: 2012-01-15 12:18:33.056920612 -0200
Modify: 2011-11-21 16:51:49.228547005 -0200
Change: 2011-11-21 16:51:49.228547005 -0200
```

O número de inodes também pode ser definido pelo usuário durante a criação de um sistema de arquivos. Existem aplicações que criam grandes quantidades de arquivos pequenos, o que pode resultar na utilização de todos os inodes disponíveis em um sistema de arquivos. Quando isso acontece, mesmo que ainda haja espaço suficiente em disco, não é mais possível a criação de novos arquivos nesse sistema de arquivos. Por isso, é necessário que o administrador de sistemas faça uma análise das aplicações que serão instaladas em um servidor, para que possa dimensionar corretamente o número de inodes que será criado em um sistema de arquivos.

```
# df -i

Filesystem Inodes IUsed IFree IUse% Mounted on
/dev/sda1 2559088 48963 2510125 2% /
```

Saiba mais

Para verificarmos o número de inodes livres em um sistema de arquivos, podemos utilizar o comando **df** com a opção **-i**.

Tipos de arquivos

- Arquivos regulares.
- Diretórios.
- Arquivos de dispositivo (bloco e caractere).
- Sockets.
- Named pipes.
- Symbolic link.

No Linux, qualquer objeto que é gerenciado pelo Sistema Operacional é tratado como arquivo, desde um arquivo texto convencional a um dispositivo de hardware. Para identificar o tipo do arquivo, o Sistema Operacional consulta as informações contidas em seu inode. A seguir veremos os diversos tipos de arquivos existentes no Linux.

Arquivo regular

- Conjunto de bytes.
- Programa executável, arquivo texto, imagem etc.
- Criados por editores de texto, comandos etc.

Trata-se do tipo de arquivo mais comum que podemos encontrar em um sistema Linux. Pode ser um relatório feito no OpenOffice, um banco de dados do MySQL, um programa executável (como o Firefox), uma imagem (arquivo JPG), o código-fonte de um programa em C etc. Arquivos desse tipo podem ser criados por meio de editores de texto, de aplicativos para tratamento de imagem, de comandos do Linux (exemplo: *touch*) etc. A sua remoção pode ser feita com o comando *rm*.

Diretório

- Podem conter qualquer tipo de arquivo.
- Criados com o comando *mkdir*.
- Removidos com o comando *rmdir* (se não tiverem nenhum conteúdo) ou *rm -r* (caso contrário).

Um diretório nada mais é do que um arquivo cujo conteúdo é o nome dos arquivos nele contidos e os números de seus respectivos inodes. Um diretório pode conter arquivos de quaisquer tipos, inclusive diretórios. As entradas especiais *"."* e *".."*, que encontramos nos diretórios de um sistema Linux, referem-se ao próprio diretório e ao seu diretório pai, respectivamente.

Saiba mais

Diretórios podem ser criados com o comando *mkdir* e removidos com o comando *rmdir*, se estiverem vazios, ou *rm -r*, independentemente de estarem vazios ou não.

Arquivo de dispositivo



- Mecanismo usado para operações de entrada e saída.
- Tipos de arquivos de dispositivo:
 - Caractere: as operações de entrada e saída são realizadas byte a byte de modo sequencial.
 - Bloco: as operações de entrada e saída são realizadas em blocos de modo aleatório.
- Encontram-se, geralmente, no diretório `/dev`.
- São criados com `mknod` e removidos com `rm`.

Arquivos de dispositivos permitem aos usuários estabelecer comunicação com o hardware do sistema e seus periféricos, sem se preocupar com os detalhes necessários a esse tipo de comunicação. Essa forma de o Linux interagir com os dispositivos também facilita muito o trabalho dos programadores. Assim, drivers de dispositivos representam uma interface de comunicação padrão que se parece com um arquivo normal. Quando o kernel recebe um pedido referente a um arquivo de dispositivo, ele simplesmente passa o pedido para o driver apropriado. Arquivos de dispositivo podem ser criados via comando `mknod` e removidos com o comando `rm`. Existem dois tipos de arquivos de dispositivos que serão vistos a seguir.

Dispositivos orientados a caractere

Os dispositivos orientados a caractere realizam suas transferências de dados byte a byte e de modo sequencial. As portas seriais são exemplos de dispositivos orientados a caractere. Geralmente, esses dispositivos não utilizam buffers (espaço em memória) em suas operações de entrada e saída, ou seja, os dados são lidos e escritos diretamente no dispositivo.

Dispositivos orientados a bloco

Os dispositivos orientados a bloco realizam suas transferências de dados em blocos de tamanho que pode variar entre 512 bytes e 32 Kbytes, sendo o acesso feito de modo aleatório. Os discos rígidos e as unidades de fita são exemplos de dispositivos orientados a bloco. As operações de entrada e saída desses dispositivos são feitas utilizando buffers intermediários.

Socket



- Utilizado para comunicação bidirecional entre processos.
- Unix Domain Sockets.
- Sockets de rede.
- São criados com a chamada de sistema `socket` e removidos através do comando `rm` ou da chamada de sistema `close`.

O socket é um tipo de arquivo usado para a comunicação bidirecional entre dois processos. Existem basicamente dois tipos de sockets: o primeiro deles, conhecido como Unix domain socket ou IPC socket (Inter Process Communication socket) é usado para a comunicação entre processos executados em um mesmo Sistema Operacional. O outro tipo é o socket de rede, que é usado para a comunicação entre processos executados em computadores diferentes, interligados por uma rede. Entre os sockets de rede, podemos destacar o datagram socket, que é um tipo de socket não orientado à conexão, usado pelo protocolo UDP; o stream socket, que é um tipo de socket orientado à conexão, usado pelo protocolo TCP, e o raw socket, utilizado por protocolos de gerenciamento e monitoramento

de redes como o protocolo ICMP. Um socket é criado pela chamada de sistema `socket` e pode ser removido com o comando `rm` ou por meio da chamada de sistema `close`, quando o socket não estiver mais sendo utilizado. A maioria das aplicações em um sistema Linux faz uso de sockets.

Named pipe

- Permite a comunicação entre dois processos rodando em uma mesma máquina.
- Faz parte do sistema de arquivos.
- É criado com o comando `mkfifo` ou `mknod`.
- É removido com o comando `rm`.
- Interliga processos que não possuem relação entre si.



O named pipe, também conhecido como FIFO, permite a comunicação entre dois processos executados no mesmo Sistema Operacional. Um named pipe é referenciado pelos processos que conecta através de seu nome e faz parte do sistema de arquivos. Ele é criado através dos comandos `mkfifo` ou `mknod`, e removido com o comando `rm` ou por meio da chamada de sistema `unlink`. Um named pipe permite a comunicação entre processos que não possuam relação entre si. Existe também o pipe convencional, representado pelo caractere `|`, que conecta a saída de um processo à entrada de outro. A diferença entre eles é que no pipe convencional os processos conectados devem possuir uma relação pai para filho ou serem “irmãos”. Ao contrário do named pipe, que precisa ser explicitamente encerrado após seu uso, o pipe convencional é encerrado automaticamente após a execução dos processos que ele conecta.

Link

Hard link:

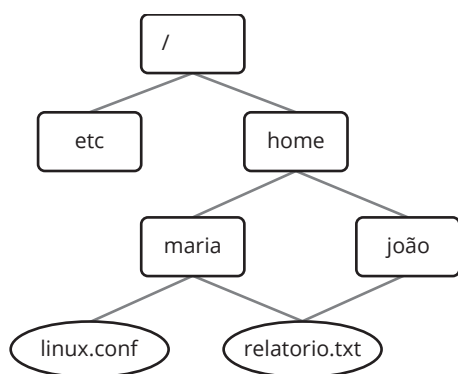
- Dois arquivos apontando para o mesmo inode.

Symbolic link:

- Ponteiro para um arquivo existente.
- É associado a outro inode number.



O Linux suporta dois tipos de links: “hard link” e “symbolic link”. Para explicar as diferenças entre eles, vamos tomar como exemplo uma situação em que dois usuários (Maria e João) compartilham um arquivo denominado *relatorio.txt*, cujo tamanho é 200 bytes (figura 1.5).



Exemplos:

Figura 1.5
Hard link para
arquivo.

```
# ls -la ~maria/relatorio.txt
-rw-r--r-- 1 maria maria 200 Feb 01 18:00 relatorio.txt

# ln ~maria/relatorio.txt ~joao/relatorio.txt

# ls -la ~maria/relatorio.txt ~joao/relatorio.txt
-rw-r--r-- 2 maria maria 200 Feb 01 18:00 /home/maria/relatorio.txt
-rw-r--r-- 2 maria maria 200 Feb 01 18:00 /home/joao/relatorio.txt
```

Hard link

- Associa dois ou mais nomes de arquivos ao mesmo inode.
- Compartilha a mesma área de dados.
- Não pode ser criado para diretórios.
- Criado com o comando *ln*.
- Removido com o comando *rm*.



Como mencionado anteriormente, as informações sobre um arquivo são armazenadas em seu inode. No hard link, existe uma única cópia do arquivo armazenada nos blocos de dados. Essa cópia é referenciada por um único inode, que, por sua vez, é referenciado pelos arquivos nos diretórios dos usuários. Dessa forma, sempre que um dos usuários modificar o conteúdo do seu arquivo, as mudanças serão automaticamente percebidas pelos outros, uma vez que seus arquivos apontam para a mesma área em disco. No exemplo descrito, o espaço de 200 bytes do arquivo *relatorio.txt* é alocado uma única vez nos blocos do disco, mas o inode que aponta para esses blocos é referenciado pelos arquivos nos diretórios dos usuários Maria e João, como mostra a figura 1.6. Um hard link é criado com o comando *ln* e só pode apontar para arquivos regulares ou symbolic links. A existência desse tipo de link não é tão simples de ser verificada.

Ele pode ser identificado pelo número que aparece após as permissões associadas a um arquivo, como pode ser visto no exemplo a seguir. O número 2 identifica que além do arquivo original, existe um hard link apontando para ele.

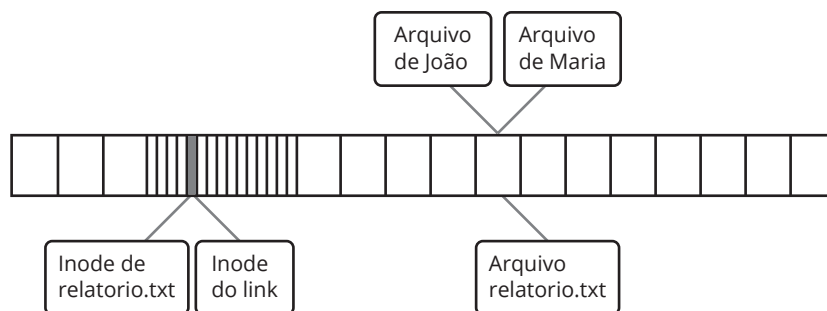


Figura 1.6
Hard link.

Symbolic link

- Arquivos que são ponteiros para outros arquivos.
- Fornecem referência por nome.
- Ocupam espaço mínimo.
- Podem apontar para qualquer área do disco, inclusive em outras partições.
- Criados com o comando `ln -s`.
- Removidos com o comando `rm`.



É representado por um arquivo cujo conteúdo é o nome do arquivo original para o qual o symbolic link está apontando. Nesse caso, existem dois arquivos com conteúdos diferentes, cada um deles com seu inode associado. Quando um symbolic link é acessado, o sistema verifica através das informações contidas em seu inode que ele é do tipo symbolic link. Após isso, recupera a identificação do arquivo original e realiza as operações solicitadas diretamente sobre ele.

Um symbolic link é criado com o comando `ln`, utilizando a opção `-s` e pode apontar para qualquer tipo de arquivo. Um symbolic link é identificado pela letra "l", que é o primeiro caractere em uma lista de arquivos gerada pelo comando `ls`, utilizando a opção `-l`. É possível também ver o nome do arquivo apontado no final da linha, como mostra o exemplo a seguir:

```
# ls -la ~maria/relatorio.txt
-rw-r--r-- 1 maria maria 200 Feb 01 18:00 relatorio.txt

# ln -s ~maria/relatorio.txt ~joao/relatorio.txt

# ls -la ~maria/relatorio.txt ~joao/relatorio.txt
-rw-r--r-- 1 maria maria 200 Feb 01 18:00 /home/maria/relatorio.txt
lrwxrwxrwx 1 joao joao 20 Feb 05 11:30 relatorio.txt -> ~maria/
relatorio.txt
```

No exemplo, o espaço de 200 bytes ocupado pelo arquivo `relatorio.txt` é alocado uma única vez nos blocos do disco. Apenas o arquivo no diretório da usuária Maria referencia essa área de dados. No diretório do usuário João, existe outro arquivo, do tipo symbolic link, cujo conteúdo é o nome do arquivo no diretório da usuária Maria, como mostra a figura 1.7.

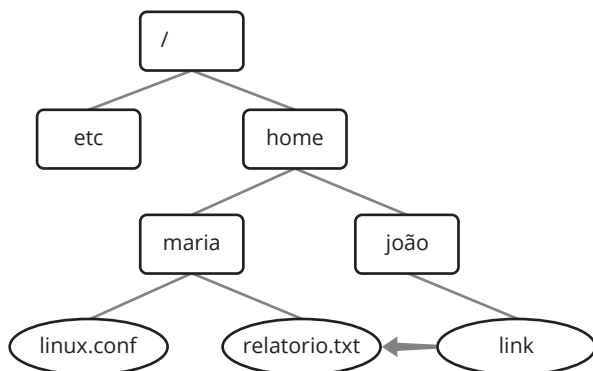


Figura 1.7
Symbolic link para
arquivo.

O arquivo de João está armazenado em outro local do disco sendo referenciado por outro inode, e seu tamanho é menor do que o tamanho do arquivo original, como pode ser visto na figura 1.8, já que seu conteúdo é somente o nome do arquivo original para o qual está apontando.

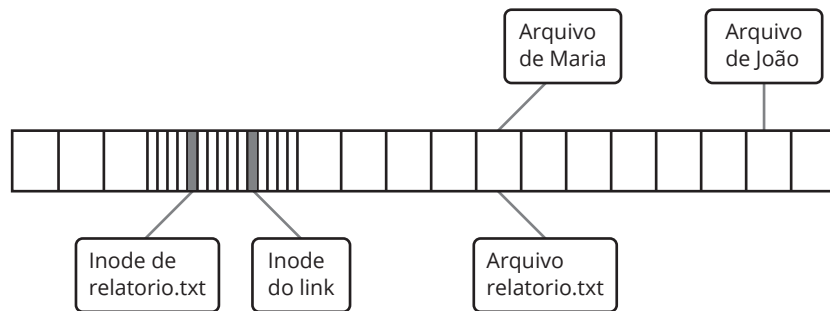


Figura 1.8
Symbolic link.

Vantagens e desvantagens

Cada um dos tipos de link vistos tem suas vantagens e desvantagens. Utilizando hard links, independentemente do número de links existentes, apenas um inode será usado. Porém, como esse inode guarda informações sobre o proprietário do arquivo, em determinadas situações isso pode gerar problemas. Vamos retornar ao exemplo anterior para visualizar essa situação.

Supondo que Maria seja a proprietária do arquivo *relatorio.txt* e um hard link seja criado para que João tenha acesso ao arquivo. O inode mantém um atributo (número de links) indicando que dois arquivos apontam para esse mesmo inode. Quando Maria remove o arquivo de seu diretório, o sistema apenas decrementa esse atributo do inode do arquivo. Assim, Maria não terá mais acesso ao arquivo, mas João ainda poderá utilizá-lo. Entretanto, uma vez que Maria é a proprietária do arquivo, esse é contabilizado no sistema como recurso alocado para Maria, até que João decida removê-lo, se tiver permissão. Com o symbolic link isso não acontece, pois apenas o proprietário do arquivo pode removê-lo. No exemplo, Maria seria a proprietária do arquivo e João teria apenas um symbolic link para ele. Se João apagasse o symbolic link, o arquivo original não seria apagado. Porém, se Maria apagasse o arquivo, João não poderia mais utilizá-lo, já que o symbolic link é apenas um ponteiro para o arquivo original. Nesse caso, o symbolic link ficaria sem referência. Note que o symbolic link é muito parecido com os atalhos do MS Windows. A utilização de symbolic links é obrigatória quando se quer criar um link para um diretório, ou entre arquivos localizados em diferentes partições. Em ambos os casos, não é possível definir um hard link.

Comando ls

Para trabalhar com arquivos, é fundamental que o usuário conheça suas características. Essas informações podem ser visualizadas com o comando *ls*. Com ele, o usuário pode visualizar várias informações sobre arquivos, desde o nome dos arquivos existentes em um diretório até o número de hard links que apontam para esses arquivos. As informações mostradas pelo comando *ls* dependem das opções passadas na linha de comando. Por exemplo, quando informada a opção *-l*, o comando *ls* lista as informações guardadas no inode do arquivo, como: permissões, número de hard links que apontam para o arquivo, proprietário do arquivo, grupo do arquivo, tamanho e data de criação e modificação. Para ver todas as funcionalidades do comando *ls*, consulte sua página de manual com o comando *man ls*.

Comando	Função
ls	Listar informações sobre arquivos
Tipo do arquivo	Símbolo
Arquivo regular	-
Diretório	d
Arquivo de dispositivo orientado a caractere	c
Arquivo de dispositivo orientado a bloco	b
Socket	s
Named pipe	p
Symbolic link	l

Tabela 1.1
Comando ls
(tipos de arquivos).

Permissões de arquivos

- Nove bits controlam quem pode ler, escrever ou executar um arquivo.
- Três outros afetam a operação de arquivos executáveis e o comportamento de diretórios.
- Quatro bits guardam a informação do tipo do arquivo (não podem ser modificados).



No Linux, todo arquivo possui um conjunto de permissões de acesso associadas. Essas permissões são definidas por nove bits que determinam quem pode ler, escrever e executar um arquivo. Além dos bits de permissão, existem três bits especiais que afetam o modo de execução dos arquivos e o comportamento dos diretórios. Por fim, o conjunto formado por esses doze bits é armazenado, juntamente com outros quatro bits, que indicam o tipo do arquivo em uma palavra de 16 bits. Os quatro bits que indicam o tipo do arquivo são definidos no momento da sua criação e não podem ser alterados. Como já vimos, o comando `ls` pode ser utilizado para verificar o valor desses bits. O campo que, no exemplo a seguir, apresenta o valor “-rwxr-xr-x”, representa as permissões associadas ao arquivo, bem como seu tipo. A seguir veremos em detalhes os bits de permissão.

```
#ls -l /bin/bash
-rwxr-xr-x 1 root root 85924 01 Dec 2002 /bin/bash
```

Bits de permissão

São nove os bits de permissão, usados para determinar as operações que podem ser executadas em um arquivo, e quem pode executá-las. O Linux define conjuntos de permissões para o dono do arquivo, para o grupo ao qual o arquivo pertence e para os outros usuários do sistema. Cada conjunto possui três bits: um bit de leitura (r), um bit de escrita (w) e um bit de execução (x), chamado “bit de pesquisa”, quando se trata de um diretório. Assim, os três primeiros bits controlam o acesso do dono, os três do meio controlam o acesso do grupo e os últimos três controlam o acesso para os demais usuários do sistema (“outros”). Em cada trio, o bit mais alto é o que controla a leitura, o bit do meio controla a escrita e o último bit controla a execução do arquivo.

Cada usuário de um sistema Linux enquadra-se em, pelo menos, uma das três categorias descritas. Caso ele não seja o dono do arquivo, nem pertença ao mesmo grupo ao qual o

arquivo pertence, as permissões dadas aos “outros” serão levadas em consideração. O bit de leitura permite a abertura e a leitura de um arquivo normal. O bit de escrita, por sua vez, permite que o conteúdo do arquivo seja modificado. A possibilidade de renomear ou remover um arquivo, contudo, é controlada pelas permissões do diretório ao qual ele pertence. O bit de execução, como o nome já sugere, permite que o arquivo seja executado, por isso, esse tipo de permissão só faz sentido em arquivos executáveis. Para um diretório, o bit de execução tem um significado um tanto diferente.

Ele permite a entrada em um diretório, mas não necessariamente que o seu conteúdo seja listado. A listagem do conteúdo de um diretório só pode ser realizada se os bits de leitura e execução forem definidos. Já a combinação dos bits de escrita e execução permite que arquivos sejam criados, removidos e renomeados dentro de um diretório. Quando um bit de permissão não é definido, o caractere “-” é inserido em seu lugar.

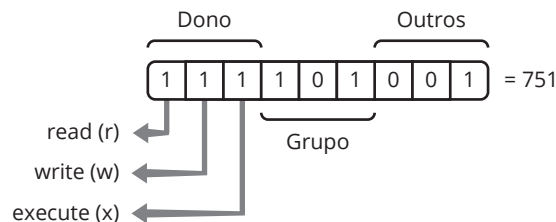


Figura 1.9
Bits de permissão.

Acesso	Significado para arquivos	Significado para diretórios
r	permitir abrir e ler	listar, se “x” está setado
w	modificar ou truncar; deletar ou renomear: ver perm. dir.	criar, renomear e deletar arquivos, se “x” está setado
x	executar arquivo	entrar, mas não listar

Bits especiais

Sticky bit:

- Evita mecanismo de swap-on/swap-off em arquivos executáveis (em desuso).
- Controla a escrita em diretórios de uso geral.
- Representado pela letra “t”.

Setgid:

- Usados em arquivos executáveis e diretórios.
- Representado pela letra “s”.

Setuid:

- Utilizados em arquivos executáveis.
- Representado pela letra “s”.

Dissemos anteriormente que, além dos nove bits de permissão, outros três afetam a execução de programas executáveis e diretórios, e que juntos eles formam os 12 bits de modo. Esses outros três bits, setuid (SUID), setgid (SGID) e sticky bit, são descritos a seguir.

Sticky Bit

O estabelecimento desse bit remonta aos tempos em que memória era um recurso caro e escasso em sistemas Linux, e acessos a unidades de disco eram lentas. Sistemas com pouca memória necessitavam de que alguns programas se mantivessem na memória continuamente.



O sticky bit era importante nesse contexto. Ele garantia que um determinado processo permaneceria na memória principal, não tendo sua performance prejudicada por conta do esquema de swapping. Hoje, com o relativo baixo custo de memória e as unidades de disco cada vez mais velozes, o estabelecimento desse bit em arquivos executáveis caiu em desuso.

Atualmente, o sticky bit é definido somente em diretórios e é utilizado para impedir que um usuário apague ou renomeie um arquivo, a menos que seja dono do diretório ou do arquivo. Desse modo, ter permissão de escrita e pesquisa em um diretório com sticky bit definido não é suficiente para remover ou renomear arquivos e diretórios de outros usuários. É dessa forma que o Linux evita que, em diretórios de uso público, como o `/tmp`, os usuários removam ou renomeiem arquivos de outros usuários. Essa regra só não é válida para o usuário root. O sticky bit quando definido substitui a letra “x” da permissão de execução dos “outros” pela letra “t”, caso a permissão de execução tenha sido definida. Caso contrário, a letra “T” é inserida no lugar do caractere “-”.

Setgid (SGID)

O bit SGID pode ser definido em arquivos executáveis e diretórios. Ele permite que um arquivo seja executado com as permissões do grupo dono do arquivo, independente do usuário que o tenha executado. Embora qualquer tipo de arquivo possa ter esse bit definido, muitas versões do Linux só o leva em conta quando são definidos em programas executáveis ou em diretórios.

Quando o bit SGID é definido em um diretório, todos os arquivos criados nesse diretório pertencerão ao grupo que é dono do diretório, independente do grupo primário do usuário que os tenha criado. O SGID, quando definido, substitui a letra “x” da permissão de execução do grupo pela letra “s”, caso a permissão de execução tenha sido definida. Caso contrário, a letra “S” é inserida no lugar do caractere “-”.

Setuid (SUID)

O bit SUID só possui efeito em arquivos executáveis. Ele permite que um arquivo seja executado como se estivesse sendo executado pelo dono do arquivo, independente do usuário que o tenha executado. O arquivo executável *passwd*, usado para troca de senha, é um exemplo de arquivo que tem o bit SUID definido. Isso é necessário para que usuários comuns possam alterar suas senhas, já que somente o usuário root tem permissão de escrita no arquivo `/etc/shadow`, que armazena as senhas dos usuários do sistema. Embora qualquer tipo de arquivo possa ter esse bit definido, muitas versões do Linux só o leva em conta quando são definidos em programas executáveis.

O SUID, quando definido, substitui a letra “x” da permissão de execução do dono do arquivo pela letra “s”, caso a permissão de execução tenha sido definida. Caso contrário, a letra “S” é inserida no lugar do caractere “-”.

- r w **S** - **s** r - **t**

t aqui indica que o programa é sticky
s aqui indica que o programa é SGID
s aqui indica que o programa é SUID

Figura 1.10
SUID, SGID
e sticky bit.

Mudando permissões

Os doze bits de permissão, referenciados anteriormente, podem ser modificados pelo dono do arquivo ou pelo administrador do sistema através do comando *chmod*. Os primeiros sistemas Unix exigiam que os usuários entendessem de notação binária ou octal para utilizar esse comando. As versões mais recentes do sistema, entretanto, aceitam tanto a notação octal quanto uma sintaxe mais mnemônica. Entretanto, preferimos trabalhar com a notação octal, por ser mais conveniente para administradores de sistema.

O comando *chmod* recebe dois blocos de argumentos. O primeiro deles é um número octal, que representa as permissões a serem atribuídas, e o segundo bloco é composto de um ou mais nomes de arquivos, separados por espaço em branco, cujas permissões devem ser alteradas. Na forma normal:

- **Primeiro dígito octal:** refere-se às permissões do dono do arquivo;
- **Segundo:** refere-se às do grupo dono do arquivo;
- **Terceiro:** refere-se às dos outros usuários.

Para estabelecer as permissões associadas aos bits *setuid*, *setgid* e *sticky bit*, é preciso utilizar quatro dígitos octais no lugar de três. Nesse caso, os três bits especiais são definidos pelo primeiro dígito. A tabela 1.2 ilustra as oito possíveis combinações para cada um dos três bits, em que “r”, “w” e “x” correspondem às permissões de leitura (Read), de escrita (Write) e de execução (eXecute) respectivamente.

Comando	Função	
chmod	Modificar a permissão de arquivos e diretórios	
Octal	Binária	Permissões
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwX

Tabela 1.2
Mudando
permissões
de arquivos.

A tabela 1.3 ilustra as oito possíveis combinações usadas para definir ou não os bits especiais SUID, SGID e *sticky bit*.

Comando	Função	
<code>chmod</code>	Modificar a permissão de arquivos e diretórios	
Octal	Binária	Descrição
0	000	SUID, SGID e sticky bit não definidos
1	001	Sticky bit definido
2	010	SGID definido
3	011	Sticky bit e SGID definidos
4	100	SUID definido
5	101	Sticky bit e SUID definidos
6	110	SGID e SUID definidos
7	111	Sticky bit, SGID e SUID definidos

Tabela 1.3
Mudando permissões especiais.

A seguir, podemos ver um exemplo de execução do comando `chmod`.

```
# chmod 0751 /home/aluno1/arquivo.txt
```

Esse comando atribui todas as permissões para o dono, ou seja, ele pode ler, escrever e executar o arquivo; atribui ao grupo permissão para apenas ler e executar o arquivo e atribui aos outros somente permissão para executá-lo. O primeiro número octal passado como parâmetro para o comando `chmod` é referente às permissões especiais; o segundo, terceiro e quarto são relativos às permissões aplicadas ao dono do arquivo, ao grupo e aos outros, respectivamente. É importante ressaltar que o primeiro número octal é opcional e, se não for definido, será considerado como tendo o valor 0.

Máscara de usuário

O comando `umask` (user mask) pode ser utilizado para estabelecer as permissões padrões para os arquivos criados no sistema. Para esse comando, é passado um parâmetro na forma de um valor octal de três dígitos, os quais representam as permissões que devem ser atribuídas aos arquivos no momento da sua criação. Os dígitos estabelecem as permissões mostradas na tabela 1.4. A relação entre os valores passados para o comando `umask` e as permissões resultantes diferem bastante da relação aplicada para o comando `chmod`. O valor padrão da `umask` para a maioria das distribuições Linux é 022. Para verificarmos o valor da máscara atual, devemos executar o comando `umask` sem parâmetros. A tabela 1.4 mostra que o valor da `umask` resulta em diferentes permissões para arquivos binários, arquivos texto e diretórios.

Comando	Função			
umask	Estabelecer permissões padrões para arquivos criados.			
Octal	Binária	Permissões		
		Arquivo		Diretório
		Binário	Texto	
0	000	r-x	rw-	rwX
1	001	r--	rw-	rw-
2	010	r-x	r--	r-x
3	011	r--	r--	r--
4	100	--x	-w-	-wX
5	101	---	-w-	-w-
6	110	--x	---	--X
7	111	---	---	---

Tabela 1.4
Comando umask.

2

Usuários e grupos

objetivos

Conhecer os parâmetros associados a grupos e à conta de um usuário; Saber quais são os comandos para modificar os parâmetros de uma conta; Entender o mecanismo de shadow passwords; Conscientizar-se da importância das senhas para a segurança do sistema; Aprender a manipular grupos: criar, alterar e excluir; Saber como manipular contas de usuários: criar, alterar, desabilitar e excluir; Como monitorar as atividades dos usuários logados no sistema.

conceitos

Grupos; Usuários; Administrando grupos; Administrando contas de usuários.

Introdução

Quando o Sistema Operacional Unix foi criado, era comum a criação de contas de usuários. Estes utilizavam o sistema conectados diretamente através de sua console ou através de acesso remoto, trabalhando em aplicativos que eram executados no próprio servidor. Nessa época, a maioria dos usuários eram pesquisadores ou estudantes de computação e áreas relacionadas. Com a adoção em massa dos computadores pessoais, a partir da década de 1980, esse modelo começou a cair em desuso.

Com a popularização da internet, os sistemas começaram a se tornar alvo de atacantes que exploravam vulnerabilidades em softwares e realizavam tentativas de acesso utilizando contas de terceiros. Com isso, os administradores passaram a adotar práticas mais seguras, restringindo ao máximo a criação de contas de usuários. Com o passar do tempo, a ocorrência de incidentes de segurança se tornou cada vez mais constante, e outra prática que passou a ser adotada foi a criação de contas sem shell, não permitindo que os usuários fizessem login no sistema. Assim, os usuários acessam somente os serviços que são executados nos servidores, como e-mail e compartilhamento de arquivos, por exemplo, já que não têm necessidade nenhuma de fazerem login no servidor.

A criação de usuários e grupos em sistemas Unix-like é importante para definir que recursos podem ser acessados por quais usuários e/ou grupos e permite ao Sistema Operacional gerenciar a execução dos processos de cada usuário de forma adequada. Atualmente, é cada vez mais comum o uso de bases de usuários centralizadas e gerenciadas por programas como o LDAP, que acabam com a necessidade de se criar contas de usuários em cada um dos diversos sistemas de uma instituição. No entanto, o modelo de criação de usuários e grupos tradicional dos sistemas Unix-like ainda é bastante utilizado e será o objeto de estudo desta sessão de aprendizagem.

Grupos

Arquivo `/etc/group`:

- grupo:senha:GID:lista_de_usuários
 - ▣ grupo: nome do grupo.
 - ▣ senha: senha do grupo (em desuso).
 - ▣ GID: número de identificação do grupo.
 - ▣ lista_de_usuários: relação dos usuários que pertencem ao grupo, separados por vírgula.
 - ▣ Exemplo:
 - ▣ `logistica::1001:funcionario1,funcionario2`

A criação de grupos de usuários geralmente é feita para controlar o acesso a arquivos ou serviços. Cada grupo no sistema possui um nome e um identificador numérico único, denominado Group ID (GID). As informações sobre os grupos do sistema estão contidas nos arquivos `/etc/group` e `/etc/gshadow`. Cada linha desses arquivos possui informações relativas a um determinado grupo. Suponhamos que o setor financeiro de uma empresa, que controla o salário dos funcionários, deseja disponibilizar as estatísticas consolidadas desses salários no sistema de informações da empresa, para que seus colaboradores possam utilizá-las para cálculos. Porém, essas informações não podem ser vistas por todos os funcionários da empresa. Assim, a criação do grupo financeiro e a disponibilização desses arquivos somente para os usuários desse grupo resolveria o problema. A linha referente ao grupo financeiro no arquivo `/etc/group` poderia ser semelhante à do exemplo a seguir:

```
financeiro::1002:funcionario3,funcionario4,funcionario5
```

- Um usuário pode fazer parte de vários grupos:
 - ▣ Grupo primário: GID do usuário (presente no arquivo `/etc/passwd`).
 - ▣ Grupos secundários: relacionados nos arquivos `/etc/group` e `/etc/gshadow`.

O comando `chgrp` modifica o grupo ao qual um arquivo ou diretório pertence.

Bit SGID: todos os arquivos criados em um diretório com o bit SGID definido pertencerão automaticamente ao grupo proprietário desse diretório.

Os campos presentes nas linhas do arquivo `/etc/group` são separados pelo caractere “:” e são os seguintes: nome do grupo; senha do grupo; GID e lista de usuários pertencentes ao grupo, separados por vírgulas. É importante ressaltar que não é uma prática comum atribuir senhas aos grupos, ficando esse campo em branco. Para incluir novos usuários em um grupo, é preciso editar os arquivos `/etc/group` e `/etc/gshadow`, e acrescentar ao final da linha referente ao grupo o nome desses usuários. Um usuário pertence obrigatoriamente a um grupo primário e pode fazer parte de vários grupos secundários, tendo assim diversos tipos de permissões. Nesta sessão serão vistos comandos que podem automatizar tarefas de manipulação de grupos, eliminando a necessidade de edição dos arquivos `/etc/group` e `/etc/gshadow`.

Alguns cuidados adicionais devem ser tomados para evitar problemas na utilização de grupos. Por exemplo, suponhamos que foi criado o diretório `/home/financeiro` para que os usuários do grupo financeiro possam compartilhar arquivos. Nesse caso, as permissões do grupo proprietário nesse diretório e em seus arquivos seriam `rxw` e `rw-`, respectivamente, o que possibilita acesso irrestrito aos usuários do grupo financeiro. Nesse contexto, se o usuário `funcionario3`, cujo grupo primário é `funcionario3`, criar um arquivo nesse diretório, esse arquivo terá como grupo proprietário o seu grupo primário.

Assim, apenas os usuários pertencentes ao grupo `funcionario3` poderiam ter acesso ao arquivo, e não os usuários do grupo `financeiro`, como era pretendido. Há duas alternativas para resolver esse problema. A primeira é pouco prática e consiste em usar o comando `chgrp` para mudar o grupo ao qual o arquivo pertence, lembrando de utilizar esse comando sempre que um novo arquivo for criado. A outra solução é utilizar o bit especial SGID, visto na sessão de aprendizagem 1, para que todos os arquivos criados no diretório `"/home/financeiro"` pertençam automaticamente ao grupo proprietário desse diretório. Exemplos:

```
# ls -la tabelas.doc
-rw-r--r-- 1 aluno1 aluno1 23 Abr 9 1523 tabelas.doc

Alterando permissão no arquivo tabelas.doc para o grupo users.

# chgrp -c users tabelas.doc
# ls -la tabelas.doc
-rw-r--r- 1 aluno users 23 Abr 9 1523 tabelas.doc

# chmod 664 tabelas.doc
# ls -la tabelas.doc
-rw-rw-r-- 1 aluno users 23 Abr 9 1523 tabelas.doc
```

Arquivo `/etc/gshadow`

grupo:senha:administradores_do_grupo:lista_de_usuários

- grupo: nome do grupo.
- senha: senha do grupo (em desuso).
- `administradores_do_grupo`: relação dos usuários que podem inserir ou remover usuários no grupo.
- `lista_de_usuários`: relação dos usuários que pertencem ao grupo, separados por vírgula.
 - Exemplo:
 - `logistica::510:marcio,marcelo,joel`

O arquivo `/etc/gshadow` é criado pelo comando `pwconv`, que habilita o esquema de shadow passwords no Linux, que será apresentado mais à frente. Cada linha desse arquivo possui informações relativas a um determinado grupo. O exemplo a seguir mostra uma linha típica do arquivo `/etc/gshadow`.

```
financeiro:!:funcionario3:funcionario3,funcionario4,funcionario5
```

Os campos presentes nas linhas do arquivo `/etc/gshadow` são separados pelo caractere `:` e são os seguintes: nome do grupo, senha do grupo, `administradores_do_grupo` (relação de usuários que podem inserir ou remover usuários neste grupo utilizando o comando `gpasswd`) e `lista_de_usuários` (relação de usuários que pertencem ao grupo).



Usuários

Arquivo */etc/passwd*:

- username: password:UID:GID:GECOS:homedir:shell
- username: nome do usuário.
- password: senha criptografada.
- UID e GID: user e group identification.
- GECOS: dados do usuário (nome, telefone comercial etc.).
- homedir: diretório de trabalho (home directory).
- shell: interpretador de comandos.



No Linux, apenas os usuários cadastrados podem acessar o sistema. Eles são identificados por um nome de usuário e uma senha, possuem um diretório de trabalho (home directory) e um interpretador de comandos (shell) associado. Internamente, o sistema reconhece um usuário através de um número inteiro que o identifica de forma única. Esse número é o User ID (UID).

Todo usuário pertence a pelo menos um grupo, denominado grupo primário, que é representado pelo seu GID. As informações sobre os usuários cadastrados estão armazenadas nos arquivos */etc/passwd* e */etc/shadow*. Cada linha desses arquivos possui informações relativas a um único usuário. O exemplo a seguir mostra uma linha típica do arquivo */etc/passwd*.

```
aluno1:0wei3945ai/qio:503:200:Aluno 1:/home/aluno1:/bin/bash
```

Os campos presentes nas linhas do arquivo */etc/passwd* são separados pelo caractere ":" e são os seguintes: nome de usuário, senha criptografada, UID (User ID), GID (Group ID), GECOS (campo utilizado para armazenar informações sobre o usuário, como nome completo e telefone de contato), diretório de trabalho e interpretador de comandos. O diretório de trabalho ou *homedir* é o espaço em disco reservado ao usuário na hora de sua inclusão. Por questões de segurança, alguns administradores definem contas de usuários, mas não atribuem a elas um shell válido. Dessa forma, esses usuários não conseguem se logar no sistema e apenas utilizam algum serviço, como o correio eletrônico ou o compartilhamento de arquivos e impressoras.

Os usuários possuem diferentes permissões de acesso aos recursos do sistema. O usuário root é conhecido como superusuário e tem permissão para acessar qualquer recurso do sistema e executar qualquer tipo de tarefa.

Esse usuário, que possui UID 0, é usado pelo administrador do sistema, que deve utilizar essa conta com bastante cautela, já que pode cometer algum erro grave e danificar todo o sistema.

A senha do usuário root geralmente é definida durante a instalação do Linux. Durante a instalação do sistema, além do usuário root, são criados diversos usuários de sistema, com propósitos específicos. Os usuários *bin*, *daemon* e *sys* são exemplos de usuários de sistema. Alguns deles não podem fazer login e são utilizados apenas para controlar os recursos acessados por processos. Cada processo em execução no sistema possui um usuário proprietário, que determina os recursos que podem ser acessados pelo processo. Por exemplo, o usuário *nobody* pode acessar pouquíssimos recursos do sistema e é bastante utilizado para executar processos que disponibilizam seus serviços através da rede. Em caso de ataques ao sistema explorando falhas de um processo em execução pelo usuário *nobody*, o invasor ganhará acesso ao sistema com os privilégios desse usuário e, a princípio, pouco poderá fazer.



Caso esse processo atacado estivesse sendo executado pelo usuário root, o invasor teria acesso imediato a qualquer recurso do sistema.

Problemas de segurança

Limitações de */etc/passwd*:

- Qualquer usuário tem acesso de leitura ao arquivo.
- Contém as senhas criptografadas (vulnerável ao uso de crackers).
- Usa o caractere “x” no lugar da senha quando é utilizado o esquema de shadow passwords.

/etc/shadow:

- Somente o administrador pode ler esse arquivo.

Para fazer login, o usuário digita sua senha, que é criptografada e comparada com a senha armazenada no sistema. Caso ambas sejam iguais, seu acesso é autorizado. Esse método de controle de acesso parece seguro, mas tem seus problemas. Por exemplo, para a memorização ficar mais fácil, os usuários escolhem senhas baseadas em seus nomes, datas de nascimento, nomes de filhos, animais de estimação etc. Essas senhas são fáceis de serem lembradas, mas também são muito fáceis de serem descobertas por pessoas mal-intencionadas, constituindo um dos pontos mais vulneráveis da segurança dos sistemas. Para tentar minimizar esse problema, a maioria das distribuições Linux requer que as senhas possuam letras maiúsculas e minúsculas, além de um dígito ou caractere especial.



Shadow passwords

username:password:last_changed:minimum:maximum:warn:inactive:expire

- **username**: nome do usuário.
- **password**: senha criptografada.
- **last_changed**: número de dias desde o dia 1/1/1970, desde que a senha foi trocada pela última vez.
- **minimum**: número de dias que o usuário deve aguardar para poder alterar sua senha.
- **maximum**: número de dias que a senha será válida.
- **warn**: número de dias antes de a senha expirar em que o usuário será avisado de que deve alterá-la.
- **inactive**: número de dias após a senha ter sido expirada em que a conta será desabilitada.
- **expire**: número de dias desde o dia 1/1/1970 em que a conta será desabilitada.



Há ainda outro problema que compromete a segurança dos sistemas Linux. Qualquer usuário tem acesso de leitura ao arquivo */etc/passwd*, que contém as senhas criptografadas. Assim, é possível utilizar programas chamados crackers, que geram senhas aleatoriamente ou utilizam palavras de dicionários e as comparam com as senhas armazenadas, possivelmente quebrando algumas delas. Para resolver esse problema, foi implementado o conceito de “shadow passwords”, no qual as senhas criptografadas são armazenadas no arquivo */etc/shadow*. A grande vantagem desse método é que o arquivo */etc/shadow* só pode ser lido pelo administrador, ficando inacessível para um usuário comum. Quando se utiliza *shadow passwords*, o arquivo */etc/passwd* possui apenas o caractere “x” no campo que armazena as senhas.



No entanto, havia um sério problema relacionado ao uso de *shadow passwords*: alguns programas utilizados no Linux assumiam que as senhas estavam armazenadas no arquivo */etc/passwd*, gerando algumas complicações. Limitações como essas, no entanto, são cada vez mais raras, e o uso de *shadow passwords* é utilizado por padrão no Linux.

O arquivo */etc/shadow*, além de armazenar os nomes de usuário e suas senhas criptografadas, também possui informações que definem propriedades sobre as senhas e as contas dos usuários. Cada linha desse arquivo possui informações relativas a um único usuário. O exemplo a seguir mostra uma linha típica do arquivo */etc/shadow*:

```
aluno1:0wei3945ai/qio:15360:0:99999:7:5::
```

Os campos presentes nas linhas do arquivo */etc/shadow* são separados pelo caractere ":" e são os seguintes:

- nome de usuário;
- senha criptografada;
- last_changed: representa o número de dias desde o dia 1/1/1970, quando a senha foi trocada pela última vez;
- minimum: representa o número de dias que o usuário deve aguardar para poder alterar sua senha;
- maximum: o número de dias que a senha será válida (após isso o usuário é obrigado a alterá-la);
- warn: representa o número de dias antes de a senha expirar em que o usuário será avisado de que deve alterá-la;
- inactive: o número de dias após a senha ter sido expirada em que a conta será desabilitada;
- expire: representa o número de dias, desde o dia 1/1/1970, em que a conta será desabilitada.

O arquivo */etc/shadow* é gerado a partir do arquivo */etc/passwd* e de informações contidas no arquivo */etc/login.defs*. Para habilitar o esquema de shadow passwords e gerar o arquivo */etc/shadow*, basta executar o comando *pwconv*. Esse comando cria os arquivos */etc/shadow* e */etc/gshadow*. Para desabilitar o esquema de shadow passwords, basta executar o comando *pwunconv*. Esse comando remove os arquivos */etc/shadow* e */etc/gshadow*, e armazena novamente as senhas criptografadas dos usuários no arquivo */etc/passwd*.

Tipos de contas de usuários

Em um sistema Linux, existem basicamente três tipos de contas de usuários: a conta root, que é usada pelo administrador e possui acesso irrestrito a todos os recursos do sistema; as contas de sistema, que são utilizadas por serviços para gerenciar seus processos, e as contas de usuário. A tabela 2.1 mostra os tipos de contas com seus respectivos níveis de permissão e exemplos de usuários.

Tipos de usuários	Permissões	Usuários
Administrador	Total	root
Padrão	Parcial	aluno
Sistema	Específica	sys, bin, ftp

Tabela 2.1
Tipos de contas de usuários.

Senhas



- Escolher senhas complexas.
- Somente gravá-las em sua memória.
- Digitá-las sem a presença de terceiros.
- As senhas devem ser pessoais e intransferíveis.

Senhas simples são mais fáceis de memorizar, porém também são muito mais fáceis de serem quebradas. Para ajudar a manter a segurança do sistema, o administrador deve recomendar aos usuários que tomem alguns cuidados, como:

- Escolher senhas complexas;
- Nunca escrever senhas em papel ou armazená-las em arquivos de computador;
- Evitar digitar a senha na presença de pessoas;
- Nunca fornecer a senha para outros, mesmo que sejam pessoas de confiança;
- Modificar a senha com frequência.

É claro que escolher senhas complexas também torna a memorização mais difícil. O mais indicado é escolher uma senha que lembre alguma coisa para o usuário, mas só para ele. Um exemplo de senha difícil de ser quebrada é aquela composta pelas letras iniciais das palavras de uma frase do tipo: "Meu nome é José, nasci em 47". A senha, nesse caso, será Mnejne47.

O risco das senhas prováveis

Em 1978, Robert Morris e Ken Thompson compilaram uma lista de senhas prováveis, como primeiros nomes, últimos nomes, datas de nascimento, endereços etc., incluindo as sequências invertidas dos caracteres dessas mesmas informações. Eles criptografaram esse conjunto de senhas possíveis e compararam com um arquivo de senhas */etc/passwd*. O resultado foi assustador: 86% de acertos. Atualmente, os algoritmos de criptografia são mais complexos e difíceis de serem quebrados. Porém, senhas fáceis continuam contribuindo para aumentar a falta de segurança dos sistemas.

Criando contas de usuários



- Criar entrada nos arquivos */etc/group* e */etc/gshadow*.
- Criar entrada nos arquivos */etc/passwd* e */etc/shadow*.
- Definir senha inicial.
- Criar diretório de trabalho.
- Copiar arquivos de inicialização.
- Criar quota de uso de disco (opcional).
- Testar nova conta.

O processo de criação de um novo usuário consiste na execução dos seguintes passos:

1. Editar os arquivos */etc/group* e */etc/gshadow* para criar o grupo primário do usuário e adicionar o usuário aos grupos secundários necessários. Este passo só é necessário caso o grupo primário ainda não exista e se for necessário incluir o usuário em outros grupos.
2. Editar os arquivos */etc/passwd* e */etc/shadow*, e inserir uma nova linha com os parâmetros relativos à conta do novo usuário.



3. Definir uma senha temporária para a conta, utilizando o comando *passwd*.
4. Criar o diretório de trabalho (home) do usuário.
5. Copiar os arquivos de inicialização contidos no diretório */etc/skel* para o diretório home do usuário.
6. Configurar a quota de disco para o usuário. Este passo só é necessário se o sistema utilizar quotas.
7. Testar se a conta foi criada corretamente.

Mais adiante, veremos que existem comandos no Linux, e mesmo ferramentas gráficas, como o Webmin, que automatizam a administração de usuários e grupos. Optamos, no entanto, por descrever cada uma das etapas, como forma de registrar os passos envolvidos no processo de criação de uma conta no Linux. Assim, você, como administrador, saberá de que maneira agir quando tiver de criar um novo usuário no sistema, independente da ferramenta disponível. Antes de criar uma conta, recomenda-se que o administrador obtenha do seu dono não apenas as informações necessárias à criação dessa conta, mas outras que facilitarão o contato com o usuário. Na sessão de aprendizagem 9, ao explicar procedimentos administrativos, vamos sugerir um formulário que deve ser adotado no atendimento a usuários.



Os passos descritos a seguir devem ser executados pelo administrador do sistema usando a conta root.

Criar entrada nos arquivos */etc/group* e */etc/gshadow*

Arquivo */etc/group*:

- Criar grupo primário para o novo usuário.
- Inserir o usuário em grupo(s) secundário(s).

Arquivo */etc/gshadow*:

- Criar grupo primário para o novo usuário.
- Inserir o usuário em grupo(s) secundário(s).

Só será necessário executar esse passo caso o grupo primário do usuário não existir ou mesmo se for preciso incluir o usuário em algum grupo além do seu grupo primário.



Criar entrada nos arquivos */etc/passwd* e */etc/shadow*

Arquivo */etc/passwd*

- nome:senha:UID:GID:GECOS:home_dir:shell
 - ▣ Uso do caractere "x" no campo da senha (shadow passwords).

Arquivo */etc/shadow*:

- nome:senha:last_changed:minimum:maximum:warn:inactive:expire
 - ▣ Definir senha temporária.

Conforme vimos no início desta sessão, o arquivo */etc/passwd* é composto por uma série de campos que descrevem desde o nome do usuário até o interpretador de comandos utilizado por ele. Em um ambiente composto por vários servidores Linux, as contas de usuários são frequentemente gerenciadas por uma base global, como o NIS ou o LDAP, de forma que ao se criar um usuário ele passa a ser reconhecido em qualquer um dos servidores.



Saiba mais

Em servidores de e-mail, deve-se evitar o uso de apelidos, já que os nomes de usuários são utilizados nos e-mails, que têm associado o nome de domínio da instituição. Um exemplo de política é criar o nome do usuário usando o seu nome e o último sobrenome (exemplo: nome.sobrenome).

Trabalharemos, no entanto, com a criação de usuários em um único servidor. Um sistema Linux pode ter mais de um administrador, e nesse caso é possível acontecer a situação em que um deles esteja criando a conta de um usuário enquanto outro esteja removendo a conta de um ex-usuário. Ao criar um novo usuário no sistema, o administrador deve preencher cada um dos campos do arquivo */etc/passwd*, tomando alguns cuidados especiais.

O nome do usuário, por exemplo, deve ser único e pode conter até 32 caracteres na maioria das distribuições Linux. Ele pode incluir letras minúsculas, números e alguns caracteres especiais, como ".", "_" e "\$", este último para manter compatibilidade com o samba, que utiliza o caractere "\$" no final do nome das contas de computadores da rede. É importante ressaltar que o primeiro caractere deve ser, por padrão, uma letra minúscula ou o caractere "_". Os caracteres permitidos para uso em nomes de usuários são definidos pela variável *NAME_REGEX*, mas as regras definidas por essa variável só valem quando se utiliza o comando *useradd* para criar as contas de usuário. O comando *useradd* pode ser utilizado com a opção *--force-badname* para ignorar as regras definidas na variável *NAME_REGEX*. Por fim, é importante definir uma política para a criação dos nomes dos usuários.

Ao editar o arquivo */etc/passwd*, o administrador deve colocar o caractere "*" no campo referente à senha criptografada. Assim, ele evita o uso não autorizado dessa conta até que tenha estabelecido uma senha para ela. Se o sistema utilizar o esquema de *shadow passwords*, o administrador também deve editar o arquivo */etc/shadow* para criar a entrada para a nova conta.

Na maioria das distribuições Linux, o campo correspondente à identificação do usuário, o UID, é representado por um número de 16 bits, variando entre 0 e 65.535. Em sistemas mais novos, o UID é representado por um número de 32 bits, resultando em mais de 4 bilhões de UIDs possíveis. Em um ambiente de rede utilizando Network File System (NFS), os UIDs devem ser únicos em toda a rede, isto é, um UID precisa se referir sempre ao mesmo nome de usuário. Usuários de sistema geralmente possuem UID entre 1 e 100. Dessa forma, é recomendável atribuir para os usuários criados valores maiores que 100. Também é aconselhável evitar reutilizar UIDs, mesmo de contas que já tenham sido removidas. O administrador pode usar qualquer editor de texto para fazer essas alterações, mas quando disponível, o comando *vipw* deve ser usado para editar os arquivos */etc/passwd* e */etc/shadow*, pois previne possível corrupção nos arquivos por conta de acessos simultâneos.

Assim como o UID, o GID também é representado por um número de 16 ou 32 bits, que deve ser atribuído de acordo com o grupo primário ao qual o usuário pertence. O GID de número 0 é reservado ao grupo chamado *root* ou *wheel*. Grupos de sistema geralmente possuem GIDs entre 1 e 999. O campo GECOS é usado para identificar o usuário. Dessa forma, costuma-se colocar informações, como seu nome completo, número de telefone do trabalho, ramal etc. O diretório de trabalho do usuário é o local do sistema de arquivos para onde este é direcionado quando faz login no sistema e corresponde à área do sistema onde ele tem permissão para armazenar seus arquivos. Os diretórios de trabalho dos usuários de um sistema, de maneira geral, têm o mesmo nome do usuário e se localizam a seguir de um mesmo diretório, chamado *"/home"*.

Existem alguns interpretadores de comandos disponíveis no sistema. Exemplos de shells utilizados são o */bin/sh*, o */bin/tcsh* e o */bin/bash*. A maioria das distribuições Linux permite que o usuário defina o interpretador de comandos com o qual quer trabalhar.

Definir senha inicial

Comando *passwd*

■ Exemplo:

■ # *passwd* aluno

■ Enter new UNIX password:

Retype new UNIX password:

passwd: password updated successfully

O comando usado para mudar a senha de um usuário é o *passwd*. Quando o usuário root deseja mudar a senha de algum usuário, deve passar o nome deste como parâmetro para o comando *passwd*. Nesse caso, o comando *passwd* pedirá que seja informada a nova senha e que esta seja repetida. O administrador deve criar uma senha de fácil memorização pelo usuário, de forma que ele não ceda à tentação de anotá-la. No entanto, é bom incentivar a troca por uma senha própria, tão logo tenha acesso ao sistema. Para não ficar dependendo de o usuário trocar a senha, o administrador pode utilizar o comando *chage* para forçá-lo a trocar sua senha antes desse realizar o primeiro login. Nesse caso, o administrador pode definir uma senha temporária mais complexa e não precisa informá-la ao usuário. A seguir, a sintaxe usada pelo comando *chage* para forçar a alteração de senha.

```
# chage -d 0 aluno1
```

Assim, quando o usuário *aluno1* se logar pela primeira vez no sistema, ele será forçado a trocar de senha. A troca de senha é feita antes do processo de login. O comando *chage* também pode ser utilizado para definir parâmetros relativos às senhas dos usuários.

Saiba mais

Para consultar mais informações a respeito desse comando, sua página de manual pode ser visualizada através do comando *man chage*.

Criar diretório de trabalho

- Qualquer diretório criado pertence ao usuário root.
- O dono e o grupo devem ser alterados utilizando os comandos *chown* e *chgrp*.

Qualquer diretório que você criar, pertence, inicialmente, ao usuário root. Assim, você deve mudar o dono e o grupo desse diretório por meio dos comandos *chown* e *chgrp*. A sequência de comandos a seguir deve criar, de forma apropriada, o diretório de trabalho do usuário *gerente1* no sistema. Exemplo:

```
# mkdir /home/gerente1
# chown gerente1 /home/gerente1
# chgrp gerencia /home/gerente1
# chmod 750 /home/gerente1
```

O grupo dono de um arquivo ou diretório também pode ser definido por meio do comando *chown*. Assim, na sequência de comandos anterior, poderíamos ter os dois passos intermediários resumidos em uma única linha:

```
# chown gerente1:gerencia /home/gerente1
```

Copiar arquivos de inicialização

- Configuração de comandos e utilitários.
- Geralmente, começam com "." (ponto) e finalizam com *rc*.

Os arquivos de inicialização são arquivos que são copiados para o diretório de trabalho dos usuários e utilizados para a configuração e definição de parâmetros de comandos e aplicativos. Exemplos desses comandos com os seus respectivos arquivos de inicialização são os próprios interpretadores de comando (bash: *.bash_profile*), os editores de texto (ex: *.vimrc*), o comando *mail* (*.mailrc*) etc. Os nomes desses arquivos, em geral, começam com um ponto e finalizam com as letras “rc” (*run command*). Os arquivos que têm seu nome começando por um ponto são classificados como arquivos ocultos e só são listados pelo comando *ls* se a opção *-a* for passada como parâmetro. Na maioria das distribuições Linux, os arquivos de inicialização modelo ficam localizados no diretório “*/etc/skel*”. O administrador também pode criar arquivos de inicialização e armazená-los no diretório “*/etc/skel*”, para que sejam copiados para os diretórios dos usuários no momento de sua criação.

Exemplo para o shell bash:

```
# cp /etc/skel/.bash* /home/aluno
# chmod 644 /home/aluno/.bash*
# chown aluno:aluno /home/aluno/.bash*
```

Testar nova conta

Criando quota de uso de disco:

- Uso do comando *edquota*.
- As quotas podem ser copiadas de outro usuário.

Para verificar se a nova conta foi criada corretamente, é importante fazer login no sistema e executar a seguinte sequência de comandos:

```
# pwd
# ls -la
```

O primeiro comando serve para verificar se o diretório de trabalho do usuário está correto. O segundo serve para verificar se os arquivos de configuração foram devidamente copiados e se os parâmetros referentes a dono, grupo e permissões estão corretos. Nesse ponto, o administrador deve informar ao usuário seu nome de usuário e sua senha inicial. Essa também é a hora de fornecer alguma documentação sobre as regras de utilização dos recursos de TI da instituição. Se for exigido que o usuário assine um termo de compromisso para ter acesso ao sistema, é importante obtê-lo antes de liberar o acesso à conta.

Administrando grupos

- Criando grupos.
- Modificando grupos.
- Removendo grupos.

A administração de grupos no Linux pode ser totalmente automatizada por alguns comandos que permitem criar, alterar e remover grupos. Isso simplifica bastante o trabalho do administrador, já que esses comandos eliminam a necessidade de se editar os arquivos */etc/group* e */etc/gshadow*, operação que requer muita atenção e apresenta o risco de provocar alguma modificação indesejada. Além disso, o administrador também pode utilizar o comando *groups* para verificar os grupos aos quais um usuário pertence.



Criando grupos

`groupadd:`

- Cria um novo grupo.

`addgroup:`

- Cria um novo grupo.

Estes comandos criam uma entrada no arquivo `/etc/gshadow` automaticamente (shadow passwords).

Podem ser usados para criar grupos de sistema.

Um grupo pode ser criado através dos comandos `addgroup` e `groupadd`, que criam o novo grupo, utilizando os parâmetros passados na linha de comando. Se o esquema de shadow passwords estiver habilitado, a entrada referente ao novo grupo é criada automaticamente no arquivo `/etc/gshadow`. Para consultar mais informações a respeito desses comandos, suas páginas de manual podem ser consultadas. Algumas distribuições podem conter apenas um dos comandos.

Exemplo: CentOS possui apenas o comando `groupadd`. Já o Ubuntu possui ambos os comandos.



Modificando grupos

`groupmod`

- Modifica informações de grupos no arquivo `/etc/group`.
- nome, GID e senha (desuso).

`gpasswd`

- Modifica informações de grupos no arquivo `/etc/gshadow`.
- Insere ou remove administradores e usuários.
- Também permite alterar a senha do grupo (em desuso).

O comando `groupmod` modifica informações referentes a um determinado grupo.

Para consultar mais informações a respeito desse comando, sua página de manual pode ser consultada. Assim como a criação de grupos, algumas distribuições podem conter apenas um dos comandos.

Exemplo: CentOS possui apenas o comando `groupadd`. Já o Ubuntu possui ambos os comandos.



Removendo grupos

`groupdel`

- Remove um grupo.

`delgroup`

- Remove um grupo.

Remove entrada no arquivo `/etc/gshadow` automaticamente (shadow passwords).

O comando `deluser` também pode ser utilizado com a opção `--group` para remover um grupo.

Um grupo pode ser removido através dos comandos `groupdel` e `delgroup`. Se o esquema de shadow passwords estiver habilitado, a entrada referente ao grupo é removida automaticamente do arquivo `/etc/gshadow`. Para consultar mais informações a respeito desses



comandos, suas páginas de manual podem ser consultadas. Além desses comandos, ainda podemos utilizar o comando *deluser* com a opção *--group* para remover um grupo. A edição de grupos em algumas distribuições pode conter apenas um dos comandos.

Exemplo: CentOS possui apenas o comando *groupadd*. Já o Ubuntu possui ambos os comandos.

Administrando contas de usuários

- Criando contas de usuários.
- Modificando contas de usuários.
- Desabilitando contas de usuários.
- Removendo contas de usuários.
- Monitorando usuários.



A manipulação de contas de usuários no Linux pode ser totalmente automatizada por alguns comandos, o que simplifica bastante o trabalho do administrador. Há outras formas de administrar contas de usuários utilizando ferramentas gráficas do Linux ou até mesmo o Webmin, que é uma interface web para administração de diversas funcionalidades do sistema e será vista em detalhes na sessão de aprendizagem 10. Nesta sessão, abordaremos apenas as ferramentas utilizadas através da linha de comando.

Criando contas de usuários

Modo manual:

- Apresentado no tópico “Criando contas de usuários”.

Modo automatizado.

- Comandos *useradd* e *adduser*: criam usuários ou modificam valores e atributos para novos usuários.
- Comando *newusers*: cria ou atualiza contas de usuário de forma automatizada.



Saiba mais

Para consultar mais informações a respeito desse comando, sua página de manual pode ser visualizada através do comando *man adduser*.



Os comandos *adduser* ou *useradd* permitem criar contas sem a necessidade de editar os arquivos */etc/passwd* e */etc/shadow*, operação que requer muita atenção e apresenta o risco de provocar alguma modificação indesejada. Todos os parâmetros da conta podem ser passados ao comando *adduser*, caso queiramos modificar algum valor padrão, como mostra o exemplo a seguir.

```
# useradd -c "Aluno 1" -d /home/aluno1 -g users -s /bin/tcsh aluno1
```

Esses valores padrão podem ser listados utilizando a opção *-D* do comando.

O comando *newusers* também pode ser usado para automatizar o processo de criação ou mesmo alteração de parâmetros em contas de usuários. Os dados relativos aos usuários devem ser inseridos em um arquivo que será passado como parâmetro para o comando. Esse arquivo deve possuir o formato a seguir:

```
name:passwd:uid:gid:GECOS:dir:shell
```

Os campos devem ser separados pelo caractere “:” e são os seguintes: nome do usuário; senha (não criptografada); uid; gid; GECOS (as informações como nome e telefone devem ser separadas por vírgulas); diretório de trabalho e interpretador de comandos. Como esse arquivo possui as senhas dos usuários sem criptografia, ele deve ter permissão somente de leitura e só para o usuário root. Por questões de segurança, assim que os usuários forem criados, o arquivo deve ser removido.



Modificando contas de usuários

Modo manual:

- Edição dos arquivos `/etc/passwd`, `/etc/shadow`, `/etc/group` e `/etc/gshadow`.

Modo automatizado:

- Comandos `usermod`, `passwd`, `chfn` e `chsh`.

O comando `usermod` permite alterar vários parâmetros da conta de um usuário, como seu diretório `home` e `shell`, além de permitir a inserção do usuário em grupos secundários. Ele também permite que se defina uma data para expiração da conta, além de bloqueio e desbloqueio.

O comando `passwd` pode ser usado para alterações de senha por qualquer usuário, desde que seja para alterar somente sua senha. Ele também pode ser utilizado pelo usuário `root` para realizar alterações de senha de qualquer usuário, bloqueio e desbloqueio de contas etc.

Além dos comandos mencionados anteriormente, temos o comando `chfn`, que permite aos usuários alterarem suas informações do campo GECOS (com exceção do nome completo) e o comando `chsh`, que permite aos usuários alterarem seus interpretadores de comando. O usuário `root` pode utilizar esses comandos para alterar informações de qualquer conta. Exemplos:

```
# grep aluno /etc/passwd
aluno:x:500:500:Aluno ESR:/home/aluno:/bin/bash

# chfn -f "Aluno RNP/ESR" aluno

# usermod -c "Aluno 1 do curso ADS2" aluno1
```

Desabilitando contas de usuários

- Impedir acesso do usuário por algum tempo.
- Colocar qualquer caractere no início da senha criptografada.
- Trocar o `shell` por mensagem explicativa.
- Comandos: `usermod`, `passwd` e `chage`.

Para desabilitar uma conta de usuário, impedindo-o de entrar no sistema, basta editar o arquivo `/etc/passwd` ou `/etc/shadow`, caso seja utilizado o esquema de *shadow passwords* e colocar o caractere `"*"` no início da senha criptografada. A rigor, bastaria inserir qualquer caractere ou mesmo remover algum existente em qualquer posição da senha do usuário.

Deve-se usar sempre o mesmo caractere na mesma posição, como uma forma que possibilita voltar atrás a qualquer instante sem ter de redefinir a senha do usuário. Outra forma mais segura é substituir o `shell` do usuário por um programa que imprime uma mensagem na tela, explicando a causa da suspensão de sua conta e as providências necessárias para remediar a situação. Também é possível desabilitar uma conta utilizando o comando `usermod` com a opção `-L`. Para habilitar novamente a conta, basta utilizar a opção `-U` do comando `usermod`. O comando `passwd` também pode ser utilizado com a opção `-l` para travar uma senha. Para destravar a senha, basta utilizar a opção `-u`. Além disso, o comando `chage` também pode ser usado para bloquear uma conta, como mostra o exemplo a seguir:

```
# chage -E 2015-12-31 aluno1
```

Esse comando fará com que a conta do usuário `aluno1` seja bloqueada no dia 31/12/2015.

Removendo contas de usuários

Removendo contas de usuários:

- Modo manual:
 - ▣ Remover entradas dos arquivos */etc/passwd*, */etc/shadow*, */etc/group* e */etc/gshadow*.
 - ▣ Fazer backup dos arquivos do usuário.
 - ▣ Realocar arquivos de uso comum.
 - ▣ Remover diretório de trabalho.
 - ▣ Remover as referências à quota do usuário.
 - ▣ Evitar a reutilização do UID.
- Modo automatizado:
 - ▣ Comandos *userdel* e *deluser*.

Quando um funcionário sai da empresa em que trabalha, é necessário remover sua conta e seus arquivos do sistema. É preciso, então, retirar as entradas criadas nos arquivos */etc/passwd*, */etc/shadow*, */etc/group* e */etc/gshadow*, relativas ao usuário em questão. Por fim, deve-se realocar os arquivos que têm utilidade para outros usuários e fazer um backup do diretório de trabalho do usuário, para só depois removê-lo do sistema. Os comandos *deluser* e *userdel* removem a conta de um usuário, sem a necessidade de seguir todos os passos necessários para remover uma conta manualmente. O comando a seguir remove o usuário *aluno1* e seu diretório de trabalho.

```
# userdel -r aluno1
```

Se o sistema utilizar quotas de disco, também devem ser removidas as referências a quotas criadas, utilizando o comando *edquota*.

Monitorando usuários

- *who*: mostra os usuários conectados ao sistema.
- *w*: exibe os usuários conectados e os comandos executados.
- *last*: mostra as últimas vezes em que o usuário entrou no sistema.
- Logs do sistema e das aplicações.

Uma condição importante para manter a segurança de um sistema é saber quem o está utilizando e para quais finalidades. No Linux, há várias ferramentas que permitem visualizar essas informações, vitais para o diagnóstico de eventuais ocorrências de violação. No arquivo */var/log/wtmp*, ficam guardadas informações sobre os logins que ocorreram no sistema. Em caso de violação, basta verificar o conteúdo desse arquivo, utilizando o comando *last*. Há também dois comandos que servem para informar os usuários que estão utilizando o sistema em um dado momento. Esses comandos são o *who*, que exibe informações baseadas no arquivo */var/log/wtmp*, e o *w*, que exibe informações baseadas no arquivo */var/run/utmp*. Os arquivos utilizados pelos comandos *last*, *who* e *w* armazenam informações em um formato não legível. Sendo assim, suas informações só são úteis quando exibidas através dos respectivos comandos.

A seguir, são apresentados exemplos de execução dos comandos *who*, *w* e *last*.

```
#who
root                pts/0  Feb 16 12:29  (172.16.3.191)
root                pts/1  Feb 16 13:18  (:0.0)
aluno1              pts/2  Feb 16 18:48  (172.16.3.191)

# w
18:49:37 up 13 days, 10:39, 3 users, load average: 0.03, 0.01, 0.00
USER                TTY                FROM                LOGIN@  IDLE
JCPU                PCPU              WHAT
root                pts/0  172.16.3.191  12:29pm        2:21   0.84s
0.84s              -bash
root                pts/1  :0.0                1:18pm  5:26m  0.44s  0.44s
bash
aluno1             pts/2  172.16.3.191  6:48pm        0.00s  0.07s  0.01s
w

# last
root                pts/1                :0.0                Mon Feb
16 13:18  still logged in
root                pts/0                172.16.3.189        Mon Feb 16
12:29    still logged in
aluno1             pts/2                172.16.3.191        Fri Feb 16 18:48
still logged in
wtmp begins Tue Feb 10 18:11:33 2004
```

Existem outras formas mais eficazes de se monitorar as atividades dos usuários no sistema. Isso pode ser feito visualizando os arquivos de log, geralmente localizados no diretório “/var/log”. O programa *syslog* é responsável por gerenciar o registro de eventos do Linux e seu arquivo de configuração, */etc/syslog.conf*, contém as informações sobre o local onde ficam armazenados os diversos arquivos de logs do sistema e das aplicações.



Saiba mais

O *syslog* será visto em detalhes na sessão de aprendizagem 7.

3

Processos

objetivos

Entender o que é um processo; Conhecer os tipos de processos existentes, os componentes de um processo e o ciclo de vida de um processo; Aprender os princípios de um sistema multiprocessado; Saber o que é um programa rodando em background e foreground; Conhecer os estados de um processo; Entender como funciona o esquema de escalonamento de processos no Linux; Saber os sinais que podem ser enviados a um processo e como utilizá-los; Aprender como executar tarefas periodicamente.

Processos; Multiprocessamento; Estados de um processo; Prioridades e escalonamento de processos; Monitoramento de processos; Execução periódica de tarefas; Cron, Anacron e Fcron.

conceitos

Processos

Programa em execução.

■ Composição:

- ▣ Código do programa armazenado em memória.
- ▣ Dados usados pelo programa.
- ▣ Valores armazenados nos registradores do processador.
- ▣ Recursos alocados ao processo.
- ▣ Atributos de segurança.

Programa:

- Código executável armazenado em disco.



Processo é certamente é um dos conceitos mais importantes no estudo de Sistemas Operacionais. Os sistemas Linux utilizam uma forma muito simples e poderosa de gerenciar processos. Um processo pode ser definido como uma abstração que representa um programa em execução e é composto pelo código do programa, seus dados e valores armazenados nos registradores do processador. Um processo também aloca recursos do sistema, como arquivos e dispositivos, e possui atributos que, baseados no usuário que o executou, definem que recursos esse processo pode usar. Os Sistemas Operacionais modernos permitem a execução simultânea de vários processos, alocando o processador a cada um deles durante pequenas unidades de tempo. Isso possibilita que um programa seja usado



enquanto, por exemplo, dados são lidos do disco e um documento é impresso. É difícil notar a diferença entre processo e programa, mas seu entendimento é fundamental. Enquanto o programa é apenas o código executável armazenado em disco, um processo é o código em execução, juntamente com seus dados e valores dos registradores do processador.

Tipos de processos

Processos interativos:

- Interagem constantemente com o usuário.
- Devem ter prioridade no uso do processador, visando oferecer ao usuário tempo de resposta adequado.

Processos em batch:

- Não interagem com o usuário.
- Possuem baixa prioridade.

Processos em tempo real:

- Possuem prioridade sobre os interativos e em batch.
- Utilizados em aplicações críticas onde o tempo de resposta é crucial.

Os processos podem ser classificados basicamente em três tipos:

- **Processos interativos:** são caracterizados por interagir constantemente com o usuário e geralmente realizam diversas operações de entrada e saída. Aplicativos utilizados pelo usuário, como editores de texto, navegadores etc. são exemplos de programas que geram processos interativos. O tempo de resposta dos processos interativos deve ser o mais rápido possível, visando garantir nível de interatividade satisfatório aos usuários do sistema;
- **Processos em batch:** têm como característica interagirem com o usuário. Esse tipo de processo possui baixa prioridade e geralmente é executado em background, tipo de execução que será visto mais à frente. Aplicativos como compiladores e programas de backup são tipos de programas que são executados por processos em batch. Esse tipo de processo é o mais penalizado pelo Sistema Operacional;
- **Processos em tempo real:** são caracterizados por necessitarem de tempos de respostas previsíveis. Aplicações críticas que controlam processos industriais e aplicações de controle de tráfego aéreo, por exemplo, são associadas a processos em tempo real. Esse tipo de processo tem prioridade máxima no sistema e não pode ter sua execução interrompida.
- Tipos de processos:
 - Processos CPU bound: usam o processador na maior parte do tempo.
 - Processos I/O bound: realizam operações de entrada e saída na maior parte do tempo.

Além disso, os processos podem ser classificados de acordo com a porcentagem de tempo em que utilizam o processador ou realizam operações de entrada e saída. Processos que usam o processador na maior parte do tempo são classificados como CPU bound. Processos que realizam operações de entrada e saída na maior parte do tempo são classificados como I/O bound. O Sistema Operacional sempre privilegia processos do tipo I/O bound, de modo que possa oferecer melhores tempos de resposta às aplicações interativas.

Saiba mais

Processos são as únicas entidades ativas no Linux, ou seja, tudo que é executado no sistema possui um ou mais processos associados.

Componentes de um processo

Os componentes de um processo são:

- Espaço de endereçamento:
- Conjunto de páginas de memória.
- Contém segmentos para código do programa em execução, variáveis, pilhas e outras informações.

Estruturas de dados do kernel:

- Armazenam informações a respeito dos processos em execução:
- Mapa do espaço de endereçamento.
- Status atual do processo.
- Prioridade de execução.
- Recursos utilizados.
- Máscara de sinalização do processo.
- Usuário proprietário.

Um processo típico tem duas entidades: um espaço de endereçamento e um conjunto de estruturas internas de dados do kernel. O primeiro diz respeito a um conjunto de páginas de memória marcadas pelo kernel para o uso do processo, contendo:

- Os segmentos para o código do programa que o processo está executando;
- As variáveis usadas pelo processo;
- A pilha do processo;
- Outras informações de que o kernel necessita.

As estruturas internas de dados do kernel registram vários tipos de informação sobre cada processo, onde as mais importantes são:

- Mapa do espaço de endereçamento;
- Status atual;
- Prioridade de execução;
- Recursos utilizados;
- Máscara de sinalização;
- Usuário proprietário.

Muitos parâmetros associados a um processo afetam diretamente a sua execução. A seguir, veremos o significado e a importância dos parâmetros mais interessantes do ponto de vista de um administrador de sistemas.

PID e PPID

Process ID (PID):

- Associado pelo kernel.
- Único no sistema.

Parent Process ID (PPID).

No Linux, os processos estão organizados hierarquicamente. Cada processo criado pelo kernel é identificado no sistema, de forma única, por um número inteiro atribuído a ele no momento de sua criação. Esse número é denominado identificador de processo, mas



conhecido pela sigla PID (ProcessID) e pode variar de 1 a 65536. Quando o kernel atinge o valor máximo que um PID pode assumir, ele reinicia do zero, saltando os números que ainda estão em uso. Como veremos a seguir, o Linux não possui uma chamada de sistema única que cria um novo processo que executará um novo programa. No entanto, há um método onde um processo existente cria um clone de si mesmo para dar origem a um novo processo. O processo original, nesse caso, é chamado de processo pai, e o clone é chamado de processo filho. O atributo PPID (ParentPID) de um processo é o PID do seu processo pai.

RUID, EUID, RGID e EGID

Real and Effective User ID (RUID e EUID):

- RUID:
 - ▣ UID do usuário que criou o processo.
- EUID:
 - ▣ Usado para determinar o direito de acesso do processo aos recursos do sistema (SUID).
 - ▣ Geralmente, é o mesmo que o RUID.

O Real User Identification (RUID) é o UID do usuário que criou o processo. Já o Effective User Identification (EUID) é o UID efetivo do processo, que é usado para determinar os recursos e arquivos que o processo tem permissão de acessar. Na maioria dos processos, o RUID e o EUID são iguais. A exceção ocorre quando o programa em execução tem o bit SUID ativo. Nesses casos o RUID está relacionado ao usuário executor do programa, e o EUID, ao usuário dono do programa. Um exemplo de programa como bit SUID ativado é o comando *passwd*. Quando esse comando é executado por um usuário comum, o RUID do processo é igual ao UID do usuário executor e o EUID do processo é igual ao UID do usuário dono do programa, nesse caso, o root.

O Real Group Identification (RGID) é o GID do usuário que criou o processo. Já o Effective Group Identification (EGID) é o GID efetivo de um processo, que é utilizado para determinar os recursos e arquivos que o processo tem permissão de acessar. Na maioria dos processos, o RGID e o EGID são iguais. A exceção ocorre quando o programa em execução tem o bit SGID ativo. Nesses casos o RGID está relacionado ao grupo primário do usuário que está executando o programa e o EGID ao grupo proprietário do programa em si.

Quando um processo é iniciado, seu RGID é estabelecido a partir do RGID do processo pai. Se esse processo tenta acessar um arquivo para o qual não tem permissão, o kernel vai automaticamente verificar se a permissão pode ser garantida por intermédio do EGID.

Real and Effective Group ID (RGID e EGID):

- Mapeados no arquivo */etc/group*.
- Quando um processo é inicializado, o RGID é o mesmo do processo pai.
- EGID:
 - ▣ Análogo ao EUID em relação ao UID.
 - ▣ Se um processo não tem permissão para acessar um arquivo, o kernel verifica se é possível, com base no EGID.

Prioridade

- Determina quanto tempo o processo utilizará a CPU.
- O Kernel seleciona qual processo será executado de acordo com a prioridade do processo definida no **nice value**.



Nice value

É número inteiro que influencia a prioridade de um processo no sistema. Varia de -20 (alta prioridade) a 19 (baixa prioridade).



A prioridade de um processo determina quanto tempo de CPU ele receberá. Assim, no momento em que o kernel vai selecionar um processo para ser executado, ele seleciona aquele com a maior prioridade interna. A quantidade de tempo de CPU que o processo já consumiu e o tempo que ele está esperando para rodar são fatores que influenciam a prioridade interna. Como veremos de forma detalhada mais à frente, não é possível determinar a prioridade interna diretamente. Essa é uma prerrogativa do kernel. Contudo, é possível influenciar esse valor por meio de um comando que altera o nice value de um processo.

Terminal de controle

- Terminal de controle (Teletype - TTY).
- Determina os canais que devem ser utilizados pelo processo.



O terminal de controle determina os canais que devem ser utilizados pelo processo, como entrada (recepção de dados), saída (saída do processamento) e saída de erros (informações sobre erros no processamento). Muitos processos, mas não obrigatoriamente todos, têm um terminal de controle associado a eles. Assim, quando digitamos um comando a partir do shell, o seu terminal se torna terminal de controle do processo. O conceito de terminal de controle está intimamente associado ao envio de sinais, tópico que será visto mais à frente.

Ciclo de vida de um processo

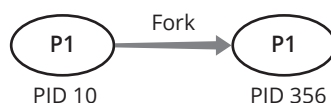
Chamada de sistema fork:

- Cria uma cópia idêntica ao processo pai, mas com outros PID e PPID.
- Códigos de retorno:
 - Se Ok: retorna o PID do filho, no contexto do processo pai e 0 no contexto do filho.
 - Se não: retorna -1 no contexto do processo pai, não cria o filho e envia o código de erro.



No Linux, processos não são gerados de forma espontânea, ou mesmo criados aleatoriamente pelo kernel. Na verdade, novos processos são criados unicamente a partir de outros processos, por intermédio da chamada de sistema fork, que cria uma cópia exatamente igual à do processo que a executou. As principais diferenças ficam por conta do PID e do PPID. Após a execução da chamada fork, os processos pai (o criador) e filho (o criado) são tratados de maneira independente, como mostra a figura 3.1, podendo criar novos processos e dar origem a uma árvore hierárquica de processos.

Figura 3.1
Processos
pai e filho.



Um processo filho herda uma cópia de todos os descritores de arquivos mantidos pelo processo pai, compartilhando os respectivos arquivos. O Linux permite que os processos identifiquem-se como pai ou como filho e sigam linhas de execução diferentes, após o retorno da chamada fork. Para tal, no retorno da chamada fork, o Sistema Operacional retorna para o processo pai o PID do processo filho. Este, por sua vez, recebe do Sistema Operacional apenas o valor 0.

Chamada de sistema exec:

- Altera o programa fonte que está em execução.
- Inicializa os segmentos de dados e pilha para um estado inicial pré-definido.



- Inicializa os registradores do processador.
- Algumas informações do processo são mantidas, como a lista de arquivos abertos.
- Os vários tipos de chamadas exec permitem que os programas sejam iniciados com diferentes opções.



Após um fork, o novo processo deverá fazer uso da chamada de sistema exec, para começar a execução de um novo programa. Existe, na realidade, uma família de chamadas de sistema do tipo exec. Todas as chamadas da família exec, no entanto, executam basicamente a mesma função: mudam o código que o processo está executando e reiniciam os segmentos de dados e a pilha para um estado inicial pré-definido. Quando uma das rotinas exec é chamada, o conteúdo da memória do processo que a chamou, geralmente uma cópia herdada da memória do processo pai, é substituído pelo conteúdo (código e dados) do novo programa e, em seguida, sua execução é iniciada no processo. A figura 3.2 ilustra uma chamada exec.

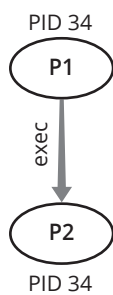


Figura 3.2
Chamada de sistema exec.

Na inicialização do Linux, o primeiro processo (PID=1) a ser executado é o init. Esse processo cria diversos processos filhos, que controlam a inicialização do sistema. Portanto, todos os processos em execução no sistema, com exceção de uns poucos criados diretamente pelo kernel, são descendentes do processo init. Ao final da inicialização do sistema, o init identifica os terminais conectados a esse e, por meio das chamadas fork e exec, cria um processo associado a cada terminal para controlar o acesso ao sistema. O processo que controla os terminais é o getty, que, quando recebe um pedido de conexão, ativa o programa login para fazer a verificação do nome do usuário e sua respectiva senha, armazenados no arquivo */etc/shadow* ou */etc/passwd*, caso o esquema de shadow passwords não esteja habilitado. Se as informações fornecidas pelo usuário estiverem corretas, um novo processo será criado para executar o interpretador de comandos do usuário, que fica aguardando por um comando. O esquema descrito anteriormente é mostrado na figura 3.3.

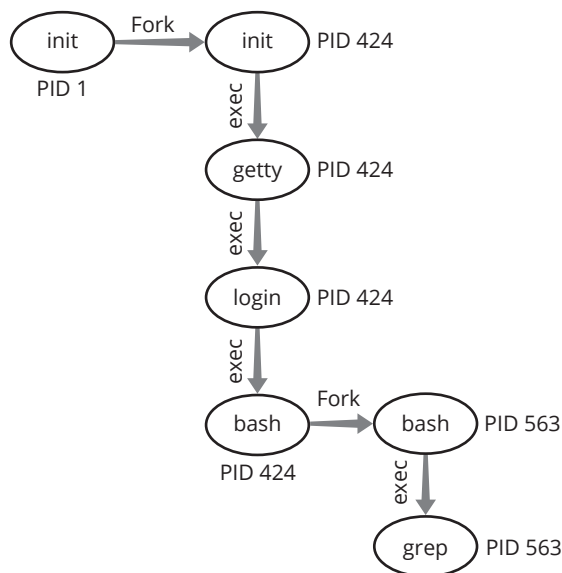


Figura 3.3
Criação de processos.

Multiprocessamento

Um usuário pode ter vários processos ativos:

- **Foreground:**
 - ▣ #ps aux
- **Background:**
 - ▣ #firefox & (associado a um terminal)
- **Daemon:**
 - ▣ Squid (não associado a terminal)

Por ser um ambiente multiusuário e multitarefa, o Linux permite que vários usuários executem, simultaneamente, diversos processos independentes. Cada usuário pode ter vários processos ativos. Em um sistema de grande porte, podem existir milhares de processos em execução. Os processos com que o usuário interage diretamente estão associados a um terminal, sendo possível para esse processo ler ou escrever dados. Diz-se que esses processos estão sendo executados em primeiro plano ou em foreground. No entanto, existem processos que, apesar de poderem estar associados a terminais, não permitem a interação direta do usuário. Diz-se que esses processos estão sendo executados em segundo plano – ou em background. Muitos processos são executados em segundo plano, controlando alguma atividade ou fornecendo algum serviço. Geralmente, processos em segundo plano realizam operações de entrada e de saída em arquivo.

No entanto, pode existir apenas um processo executado em primeiro plano associado a um determinado terminal. Existe um tipo especial de processo que é executado em segundo plano, denominado daemon, que é geralmente ativado na inicialização do sistema e não possui um terminal associado. Daemons realizam operações de entrada e de saída em arquivo ou por meio da rede. O interpretador de comandos permite ao usuário executar processos em primeiro ou em segundo plano e também iniciar ou parar um determinado daemon a qualquer momento.

Quando o usuário executa um processo em primeiro plano, o interpretador de comandos somente libera o prompt após o término da execução do comando solicitado, uma vez que o processo poderá realizar operações de entrada e de saída no terminal.

Porém, quando o usuário executa um processo em segundo plano, o interpretador de comandos imediatamente libera a linha de comando, permitindo que o usuário forneça novos comandos para serem executados.

Para executar um processo em segundo plano, o usuário precisa apenas acrescentar ao final do comando o caractere “&”. Por exemplo, para executar o navegador Firefox em segundo plano, deve ser digitado o seguinte comando:

```
#firefox &
```

Nesse caso, o Firefox será executado em segundo plano, e o interpretador de comandos ainda permitirá ao usuário fornecer novos comandos e executá-los de forma independente. A noção de processos executados em background e em foreground está diretamente relacionada aos terminais. Em ambientes gráficos de janelas, como o X-Window, os conceitos de processos em background e foreground ainda são válidos, porém seu entendimento não é tão simples. Para um processo trabalhar com janelas, ele se comunica diretamente com um processo gerenciador do ambiente gráfico, enviando mensagens de um protocolo de aplicação suportado pelo gerenciador.

No X-Window, o usuário pode executar uma janela emuladora de terminal usando o comando *xterm*, que, por sua vez, executa um interpretador de comandos. O gerenciador do ambiente gráfico define um terminal virtual para cada janela *xterm*. Assim, cada janela *xterm* comporta-se como um terminal, no qual podem ser executados processos em background ou em foreground, da mesma forma como foi explicado anteriormente.

Daemon

Daemons (Disk and execution monitor) são programas que gerenciam serviços que são executados em segundo plano. Os daemons não estão associados a nenhum terminal, e geralmente são iniciados durante o processo de inicialização do sistema. Por padrão, os nomes dos daemons terminam com a letra “d”. Por exemplo, o daemon que gerencia o serviço http é chamado de *httpd*. A maioria dos serviços de rede é gerenciada por daemons que ficam permanentemente em execução, aguardando por conexões. Os serviços podem ser gerenciados por um daemon, modo conhecido como *standalone*, ou através do *inetd* ou *xinetd*, que são daemons que gerenciam diversos serviços simultaneamente. Quando o *inetd* e o *xinetd* recebem uma requisição, eles a repassam para o daemon correspondente ao serviço solicitado.

Dessa forma, os daemons responsáveis pelo serviço gerenciado pelo *inetd* ou pelo *xinetd* só são executados quando recebem alguma requisição. Isso otimiza o uso de recursos do sistema e simplifica a implementação desses daemons, já que não precisam mais gerenciar os serviços correspondentes, deixando essa tarefa a cargo do *inetd* ou do *xinetd*.

Alternando execuções

Comando *bg*:

- Sua função é colocar um processo para rodar bem background.

Comando *fg*:

- Sua função é colocar um processo para rodar em foreground.

Comando *jobs*:

- Sua função é exibir os processos do sistema que estão suspensos ou rodando em segundo plano.



Um determinado processo executando em foreground pode ser chaveado para execução em background e vice-versa. Vejamos como isso acontece no exemplo a seguir. Suponhamos a situação em que o usuário acaba de iniciar um processo em foreground, via linha de comando, e recebe um telefonema urgente requisitando uma informação que está armazenada no arquivo `~aluno1/texto.txt`. Para executar essa nova tarefa, ele pode cancelar o processo que havia sido iniciado (como uso das teclas “Ctrl + C”). No entanto, como ele é um usuário experiente e não quer perder o trabalho que estava sendo realizado, decide colocar o processo anterior em background e executar o editor de textos vi em foreground. Para isso, inicialmente, o usuário precisa suspender a execução do processo e, em seguida, retomar sua execução em background.

Para proceder à suspensão, o usuário deve pressionar as teclas “Ctrl + Z”, liberando, assim, a linha de comando do interpretador. Em seguida, digita o comando `bg` para retomar a execução em background. Nesse ponto, a linha de comando do interpretador fica liberada, e o usuário pode executar o vi para visualizar o arquivo `~aluno1/texto.txt`. Após a utilização do vi, o usuário pode colocar o processo para ser executado novamente em foreground, usando o comando `fg`. O chaveamento de primeiro para segundo plano somente funciona se o processo chaveado não fizer uso do terminal para interagir com o usuário. O comando `Jobs` é usado para mostrar os processos que estão suspensos ou rodando em segundo plano. Exemplo:

```
#man ls - (durante a leitura, digitar Ctrl+Z)

[1]+  Stopped man ls

#bg

[1]+  man ls &

#vi ~aluno1/texto.txt - (terminada a edição)

#fg (continua a leitura do manual do comando ls a partir do ponto
em que parou)
```

Estados de um processo

- Executando.
- Bloqueado.
- Pronto.
- Parado.



Um processo pode assumir, basicamente, três estados possíveis: executando, bloqueado e pronto. Durante seu ciclo de vida, um processo pode passar diversas vezes por esses estados, dependendo das operações que precisa executar. Quando um processo é criado, ele vai automaticamente para o estado pronto e fica aguardando que o Sistema Operacional aloque o processador a ele. Um processo também pode se encontrar diversas vezes no estado pronto durante sua execução, aguardando a alocação do processador para continuar sua execução. Quando o processador é alocado ao processo, este passa para o estado executando. Um processo que esteja executando alguma operação de entrada e saída encontra-se no estado bloqueado. A figura 3.4 mostra o diagrama de estados de um processo.

A transição o número 1 representa o momento em que um processo sai da fila e passa a ser processado. A transição número 2 representa o momento em que um processo deixa de ser processado e volta para a fila, enquanto o processador é alocado a outro processo. A transição número 3 representa o momento em que um processo deixa de ser processado para realizar alguma operação de entrada e saída. A transição número 4 representa o momento em que um processo termina uma operação de entrada e saída e volta para fila para aguardar sua vez de ser processado.

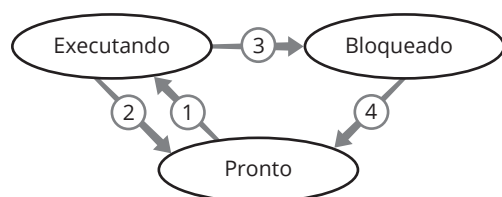


Figura 3.4
Diagrama de estados de um processo.

A pesar de os processos serem independentes, muitas vezes é necessário que se comuniquem. Um exemplo de comunicação entre processos é através de pipes, representados pelo caractere “|”. O pipe redireciona a saída de um processo para a entrada de outro processo. Veja o exemplo a seguir, que concatena os arquivos *file1* e *file2* por intermédio do comando *cat* e procura por linhas que contenham a palavra *word* utilizando o comando *grep*:

```
# cat file1 file2 | grep word
```

Nesse caso, o processo relativo à execução do comando *grep* tem de esperar pela saída gerada pelo processo *cat*. Enquanto um processo aguarda algum recurso ou o término de uma operação de entrada e saída, diz-se que ele se encontra no estado bloqueado ou dormindo (sleeping). O bloqueio de um processo é causado pela falta de condições que o tornem apto à execução, como é o caso do exemplo visto recentemente. Um processo bloqueado retorna ao estado pronto quando o recurso é liberado ou quando a operação de entrada e saída é concluída. No exemplo, o processo relativo ao comando *grep* volta para o estado de pronto quando o processo relativo ao comando *cat* termina de escrever os dados na sua entrada.

Há outros dois estados de processos listados na literatura: suspenso e zumbi. Em alguns casos, o usuário ou o próprio Sistema Operacional decide interromper a execução de um processo. Um processo suspenso ou parado (stopped) está proibido de executar. O exemplo do usuário que necessita consultar o arquivo *~aluno1/texto.txt*, apresentado anteriormente, mostra o caso em que o próprio usuário decide interromper a execução do processo, pressionando as teclas “Ctrl + Z”.

Um processo é finalizado através da chamada de sistema *exit*. Nesse momento, o processo pai é notificado do término do processo filho através do sinal SIGCHLD. A partir daí, o processo filho assume o estado “zumbi”. Nesse estado o processo terminou sua execução, mas ainda consome recursos do sistema. O processo pai então invoca a chamada de sistema *wait* que libera o descritor de processos do processo filho, que é então finalizado. Exemplo:

```
#cat /etc/passwd/etc/group | grep patricia
```

- Bloqueado pela falta de condições que o tornem apto à execução;
- O comando *grep* espera pelo processamento do comando *cat*;
- Pronto: quando o recurso é liberado ou a operação de entrada ou saída é concluída;
- O comando *grep* recebe a saída do comando *cat*;
- Parado: quando o processo encontra-se proibido de executar (“Ctrl+ Z”).

Prioridades e escalonamento de processos



- Paralelismo aparente (concorrência).
- Algoritmo de escalonamento:
 - ▣ Política de escalonamento:
 - ▣ Round-robin.
 - ▣ FIFO (First In, First Out).
 - ▣ Escalonamento baseado em prioridades.
 - ▣ Garantir que todos os processos possam ser processados.
 - ▣ Utilizar a capacidade máxima do processador.
- Fila de escalonamento.

Saiba mais

O módulo do Sistema Operacional que se encarrega de realizar essa alocação é chamado escalonador de processos, e toma suas decisões baseado em um algoritmo de escalonamento.

Em Sistemas Operacionais multitarefa como o Linux, vários processos são executados simultaneamente. Na verdade, os processos dividem o uso do processador gerando o chamado paralelismo aparente, ou seja, o chaveamento rápido dos recursos do sistema entre os processos produz a falsa impressão de que eles estão sendo executados simultaneamente. Quando diversos processos estão prontos para serem executados, o Sistema Operacional deve decidir em qual deles alocar o processador. Os processos que estão no estado pronto são colocados na fila de escalonamento e ficam aguardando até que o processador se torne livre para eles.

Em muitos casos, fatores externos ao ambiente do Sistema Operacional fazem com que alguns processos tenham prioridade sobre outros. Por exemplo, em um computador de uso geral de uma empresa, os processos gerados pela equipe de produção podem ter prioridade sobre os processos gerados pela equipe de desenvolvimento. Para atender casos nos quais a prioridade deve ser levada em consideração, o Linux suporta um esquema de prioridade de processos. A prioridade de um processo determina a quantidade de tempo de processador que ele utilizará quando estiver no estado executando. Vale ressaltar que, no Linux, quanto maior o valor do atributo de prioridade (PRI) de um processo, menor é a prioridade de escalonamento. Dessa forma, o escalonador aloca o processador ao processo de menor PRI que está na fila de escalonamento. Assim, processos com prioridade baixa (maior PRI) somente têm a chance de serem executados quando não existirem processos de alta prioridade sendo executados ou, se existirem, mas estiverem bloqueados ou suspensos.

Para evitar que um processo de alta prioridade monopolize o processador, a prioridade de um processo é calculada levando-se em consideração o tempo de CPU recentemente consumido e o tempo de espera na fila de escalonamento. A prioridade de um processo não pode ser definida ou alterada diretamente. No entanto, cada processo possui um atributo adicional, denominado *nice value*, que influencia substancialmente o cálculo da sua prioridade e que pode ser modificado diretamente pelo usuário proprietário ou pelo administrador do sistema. No Linux, o *nice value* varia entre -20 e 19 e, quanto maior for seu valor, maior é o valor da prioridade (PRI) do processo. Entretanto, como foi descrito anteriormente, quanto maior é o valor da prioridade (PRI), menor é a prioridade de escalonamento efetivado processo, ou seja, menos chance o processo tem de ser alocado ao processador.

O Linux possui dois comandos que servem para manipular o *nice value* de processos. Esses comandos são o *nice* e o *renice*. O *nice* é utilizado para ativar um processo como *nice value* corrente incrementado pelo valor fornecido. Por exemplo, para executar o comando *cat* com *nice value* corrente incrementado de 10, o usuário deve fornecer o seguinte comando:

```
# nice -n 10 cat /etc/hosts
```



O comando *nice* também permite identificar o nice value corrente. Para isso, basta digitar o comando sem parâmetros. O *renice*, por sua vez, é utilizado para modificar o nice value de um processo que já se encontra em execução. Por exemplo, supondo que o processo cujo PID é 1502 possui nice value igual a 0, o seguinte comando modifica o nice value para 10:

```
# renice 10 1502
```

O usuário root pode atribuir qualquer valor entre -20 e 19 ao nice value de qualquer processo. Por outro lado, usuários normais podem somente incrementar o valor do nice value de seus próprios processos.

Algoritmo de escalonamento

Baseado em prioridade:

- Prioridade dinâmica:
 - ▣ A prioridade dos processos é alterada constantemente.
 - ▣ Tenta equalizar o uso do processador.
 - ▣ Definida somente pelo escalonador.
- Prioridade estática:
 - ▣ Utilizada somente em processos de tempo real.
 - ▣ Definida pelo usuário.

Define uma política que permite ao escalonador decidir a qual processo alocar o processador. Vários fatores tornam um algoritmo de escalonamento mais ou menos eficiente. A lista a seguir apresenta algumas características desejáveis para um algoritmo de escalonamento.

- Garantir que cada processo terá a oportunidade de ser alocado ao processador;
 - Explorar ao máximo a capacidade do processador;
 - Balancear o uso dos recursos do sistema.
-
- Como o sistema evita o monopólio do processador por um processo de alta prioridade?
 - A prioridade é calculada levando em consideração o tempo de CPU recentemente consumido e o tempo de espera na fila de escalonamento.
 - Comando *nice*:
 - ▣ Exibe o nice value atual ou executa um processo alterando seu nice value.
 - Comando *renice*:
 - ▣ Modifica o nice value de um processo em execução.

Existem vários algoritmos de escalonamento, como escalonamento round-robin, First In, First Out (FIFO), filas múltiplas, entre outros. Como o objetivo deste curso não é o estudo dos algoritmos de escalonamento, será apresentado apenas o algoritmo de escalonamento preemptivo, utilizado no Linux. Nesse tipo de algoritmo, o escalonador aloca a cada processo pequenas fatias de tempo de uso do processador, denominadas *squantum*. O tempo de duração de cada quantum é definido pelo escalonador, baseado na prioridade de cada processo. Quando um processo entra no estado executando, o algoritmo de escalonamento verifica se existe algum processo com prioridade maior do que a prioridade do processo em execução. Caso encontre algum, o processo em execução é interrompido e o de prioridade maior passa a ser executado.

O escalonador trabalha com dois tipos de prioridade. A prioridade dinâmica, que é alterada constantemente pelo escalonador, de modo que processos que estão há mais tempo sem utilizar o processador têm sua prioridade aumentada automaticamente. Por outro lado, processos que utilizaram o processador recentemente têm sua prioridade automaticamente diminuída.

Além da prioridade dinâmica, existe a prioridade estática, que é utilizada exclusivamente por processos em tempo real. O valor da prioridade estática pode variar de 1 a 99, e é definido pelo usuário, não podendo ser alterada pelo escalonador. Processos em tempo real possuem sempre maior prioridade com relação aos processos interativos ou em batch e só podem ser criados por usuários com privilégios especiais. O escalonador só executa processos interativos ou em batch, se não houver nenhum processo em tempo real sendo executado.

Vale ressaltar que as políticas utilizadas pelo escalonador para tratar processos em tempo real são diferentes da política baseada em prioridades, utilizada para os processos interativos e em batch. Essas políticas utilizadas no gerenciamento de processos em tempo real não serão vistas neste curso por não fazerem parte de seu escopo.

Uso de sinais

- Forma de comunicação entre processos.
- Modo de ocorrência:
 - ▣ Ao receber um sinal, o processo interrompe sua execução.
 - ▣ Trata o sinal – ou o kernel atua com uma ação padrão (diferente para cada sinal).
 - ▣ Após o tratamento do sinal, o processo retoma sua execução ou é finalizado.
- Uso de sinais:
 - ▣ Comando *kill*:
 - ▣ Envia sinais para processos utilizando seu PID como parâmetro.
 - ▣ Comando *pkill*:
 - ▣ Envia sinais para processos utilizando seu nome como parâmetro.



O Linux permite que eventos externos sejam comunicados a processos por meio do envio de sinais. Uma vez que um processo pode enviar sinais a outros processos, eles podem ser vistos como uma forma de comunicação entre processos. Quando um processo recebe um sinal, ele interrompe a sua execução. Em seguida, o processo executa um procedimento de tratamento do sinal, ou o próprio kernel executa uma ação padrão, diferente para cada tipo de sinal. Após o tratamento do sinal, o processo retoma sua execução ou é encerrado. Como pode ser visto, o tratamento de sinais é bastante semelhante ao tratamento de interrupções no sistema de entrada e saída.

Um processo pode ser configurado para tratar o sinal (o procedimento de tratamento é executado) ou para ignorá-lo (a execução do processo não é afetada). O usuário pode enviar sinais para processos utilizando o comando *kill*. Nesse comando, o tipo de sinal é informado como parâmetro, juntamente como identificador do processo (PID). O comando *pkill* também pode ser usado para enviar sinais a um processo sem a necessidade de conhecermos seu PID. O *pkill* passa o nome do processo com o parâmetro, no lugar de seu PID. Outro comando que referencia processos pelo nome é o *killall*, que pode ser usado para matar todos os processos de um determinado programa.



A seguir, são apresentados os principais sinais utilizados para manipulação de processos:

SIGSTOP

Interrompe a execução do processo. O processo passa para o estado de bloqueado (stopped). O interpretador de comandos envia esse sinal ao processo executado em foreground quando o usuário pressiona as teclas "Ctrl + Z".

SIGCONT

Retoma a execução de um processo bloqueado. O processo retorna para o estado de pronto (runnable).

SIGHUP

É geralmente usado para fazer o processo reler um arquivo de configuração. Esse sinal permite a reconfiguração de serviços sem a necessidade de reiniciá-los.

SIGTERM

Pode ser tratado ou ignorado. O processo programado para tratar esse sinal termina voluntariamente, salvando dados relevantes mantidos em memória.

SIGKILL

Não pode ser tratado nem ignorado pelo processo. O kernel termina o processo de forma abrupta, e por isso os dados do processo mantidos na memória são perdidos.

O Linux possui 63 tipos de sinais que podem ser enviados para os processos. Os apresentados na tabela 3.1 são os mais utilizados. Para obter mais detalhes sobre todos os sinais interpretados pelo Sistema Operacional, consulte a página do manual dos sinais com o comando: *man 7 signal*.

Sinal	Numeração
SIGHUP	1
SIGKILL	9
SIGTERM	15
SIGCONT	18
SIGSTOP	19

Tabela 3.1
Principais sinais utilizados na comunicação entre processos.

Monitoramento de processos

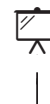
Comando *ps*:

- Lista os processos que estão sendo executados no sistema:
- PID, UID, prioridade, terminal de controle, tempo de CPU, memória utilizada etc.
- Apresenta uma fotografia dos processos que estão rodando no sistema em um determinado momento.

Programa *top*:

- Apresenta um relatório atualizado dinamicamente dos processos em execução no sistema.
- Exibe uma lista dos processos ativos e dos recursos utilizados.
- Taxa de amostragem: por padrão, a cada três segundos.





Programa pstree:

- Exibe os processos em execução em forma de árvore hierárquica.

O monitoramento dos processos no Linux pode ser realizado através do comando *ps*, utilizado para identificar os processos que estão sendo executados no sistema em um determinado momento, e é uma das principais ferramentas que um administrador de sistemas possui para monitorar processos no Linux. Ele pode ser usado para mostrar diversas informações a respeito dos processos em execução, como PID, UID associado, prioridade, terminal de controle, percentual de memória e de CPU utilizados, status atual etc. Entender a saída do comando *ps* é uma habilidade importante que o administrador precisa desenvolver. Olhando para a saída desse comando, você pode determinar, entre outras coisas, os processos que estão rodando no seu sistema, o percentual de CPU e de memória que eles estão utilizando e a qual usuário cada processo pertence.

O comando *ps* possui diversas opções de uso, cada uma delas exibindo suas informações correspondentes a respeito dos processos em execução. Esse comando pode ser uma importante ferramenta para diagnosticar problemas no sistema. Suponha, por exemplo, que você perceba que o sistema está lento e, ao rodar o comando *ps*, ele mostra que dezenas de processos *sshd* estão rodando, mas você verifica que não há ninguém logado nesse servidor via SSH. Nesse caso, é possível deduzir que o daemon do SSH não está funcionando corretamente, portanto, é recomendável que tais processos sejam encerrados.

A figura 3.5 exibe a saída do comando *ps*, utilizado com as opções *aux*, que possuem as seguintes funcionalidades: a opção “a” exibe todos os processos, com exceção dos processos não associados a um terminal; a opção “u” exibe o nome do usuário que iniciou os processos e a opção “x” exibe todos os processos que não são controlados por um terminal.

Figura 3.5
Saída do
comando *ps*.

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.2	2044	632	?	Ss	04:28	0:01	init [5]
root	2	0.0	0.0	0	0	?	S	04:28	0:00	[migration/0]
root	3	0.0	0.0	0	0	?	SN	04:28	0:00	[ksoftirqd/0]
root	4	0.0	0.0	0	0	?	S	04:28	0:00	[watchdog/0]
root	5	0.0	0.0	0	0	?	S<	04:28	0:00	[events/0]
root	6	0.0	0.0	0	0	?	S<	04:28	0:00	[khelper]
root	7	0.0	0.0	0	0	?	S<	04:28	0:00	[kthread]
root	10	0.0	0.0	0	0	?	S<	04:28	0:00	[kblockd/0]
root	11	0.0	0.0	0	0	?	S<	04:28	0:00	[kacpid]
root	68	0.0	0.0	0	0	?	S<	04:28	0:00	[cqueue/0]
root	71	0.0	0.0	0	0	?	S<	04:28	0:00	[khubd]
root	73	0.0	0.0	0	0	?	S<	04:28	0:00	[kseriod]
root	135	0.0	0.0	0	0	?	S	04:28	0:00	[pdflush]
root	136	0.0	0.0	0	0	?	S	04:28	0:00	[pdflush]
root	137	0.0	0.0	0	0	?	S<	04:28	0:00	[kswapd0]
root	138	0.0	0.0	0	0	?	S<	04:28	0:00	[aio/0]
root	292	0.0	0.0	0	0	?	S<	04:28	0:00	[kpsmoused]
root	323	0.0	0.0	0	0	?	S<	04:28	0:00	[scsi_eh_0]
root	326	0.0	0.0	0	0	?	S<	04:28	0:00	[ata/0]
root	327	0.0	0.0	0	0	?	S<	04:28	0:00	[ata_aux]
root	330	0.0	0.0	0	0	?	S<	04:28	0:00	[kjournald]



O comando `ps` apresenta uma fotografia dos processos que estão rodando no sistema em um determinado momento, e isso pode dificultar o diagnóstico de uma situação de mau funcionamento do sistema, causado por um processo que esteja, por exemplo, monopolizando o uso da CPU. Um programa que pode ser utilizado nesses casos é o `top`, que apresenta um resumo atualizado regularmente, dos processos ativos e dos recursos utilizados por esses. Inicialmente, o `top` era disponibilizado como um pacote à parte. No entanto, mostrou-se uma ferramenta tão eficaz que passou a ser incorporado às distribuições Linux.

Outro comando usado para visualizar e monitorar processos é o `ps tree`. Esse comando exibe os processos que estão sendo executados em um determinado momento, utilizando um formato de árvore hierárquica.

Execução periódica de tarefas

Existem algumas atividades que o administrador do sistema ou um usuário convencional realizam periodicamente. Por exemplo, considere uma escola em que os alunos fazem atividades escolares diárias nos computadores. O administrador de sistemas precisa, frequentemente, remover arquivos desnecessários criados pelos alunos. No entanto, cumprir essa tarefa diariamente em diversos computadores é bastante cansativo. Utilizando a execução periódica de tarefas, o administrador pode configurar os sistemas para, automaticamente, executarem uma determinada tarefa todos os dias, à noite, que remova os arquivos desnecessários.

Assim, o sistema de execução periódica de tarefas é uma poderosa ferramenta usada para automatizar diversas atividades administrativas, como remoção de arquivos em disco, verificação de e-mails, realização de backup, testes de conectividade etc.

Cron

Comando `crontab`:

- Executa tarefas de modo automático em horários pré-determinados.
- Pode ser usado por todos os usuários.

Automação de tarefas:

- Remoção de arquivos temporários em discos.
- Realização de backups.
- Ativação de serviços.



O Linux implementa um serviço de execução periódica de processos suportado pelo daemon `cron`. Esse daemon, geralmente, é iniciado no boot do sistema e permanece rodando enquanto o sistema estiver no ar. O `cron` lê um ou mais arquivos, denominados `crontabs`, que contêm uma ou mais linhas de comando e os tempos nos quais eles devem ser executados, no caso dos `crontabs` de usuários. Nos `crontabs` do sistema os nomes dos usuários sob os quais esses comandos devem ser executados também devem ser especificados. As linhas de comando são executadas pelo interpretador de comando `ssh`. Desse modo, qualquer coisa que é feita na linha de comandos também pode ser efetuada através do `cron`.

Usando o comando `crontab`, o usuário pode criar, modificar ou remover a lista de comandos que serão periodicamente ativados pelo daemon `cron`. Cada usuário pode ter seu próprio arquivo de `crontab`. A cada minuto, o daemon `cron` avalia esses arquivos no diretório `/var/spool/cron/crontabs` do sistema Ubuntu ou no diretório `/var/spool/cron`, no caso do sistema CentOS, para identificar tarefas dos usuários a serem executadas. Além disso, também verifica o arquivo `/etc/crontab` e os arquivos do diretório `/etc/cron.d` que possuem

tarefas do sistema e das aplicações. Os arquivos de crontab dos usuários somente devem ser manipulados por intermédio do comando *crontab*. Em alguns sistemas, o administrador pode controlar os usuários que têm permissão para criar atividades periódicas utilizando o crontab. Para isso, basta inserir, nos arquivos */etc/cron.allow* e */etc/cron.deny*, os usuários permitidos e os proibidos de utilizarem o crontab, respectivamente. Caso esses arquivos não existam, qualquer usuário poderá utilizar o cron para agendar tarefas.

Exemplos de uso do crontab

O *crontab* tem um formato característico que deve ser utilizado em todas as suas linhas. Esse formato é:

```
Minuto hora dia mês dia_da_semana usuário comando
```

A seguinte especificação de tempo: 15, 45 14 * * 1-5, significa “às 2:15 e às 2:45 da tarde, de segunda a sexta-feira”. Veja, a seguir, o exemplo de um arquivo de crontab que informa ao usuário, todas as sextas-feiras, às 17h, que chegou o fim de semana:

```
00 17 * * 5 echo “Enviado em `date`” | mail -s “E-mail enviado pelo  
cron” alunol@localhost
```

Note que, nesse arquivo, existe apenas uma tarefa a ser executada. Os campos preenchidos com o caractere “*” representam todos os valores válidos.

Muitos programas criam arquivos temporários nos diretórios “/tmp” ou “/var/tmp”, que não são apagados por alguma razão e alguns programas, especialmente os editores de texto, costumam fazer cópias de backup de cada arquivo trabalhado nesses diretórios. A solução é criar uma entrada no crontab para remover, todos os dias, à uma hora da manhã, por exemplo, todos os arquivos desses diretórios que não tenham sido modificados há mais de 72 horas. A entrada ficaria da seguinte forma:

```
00 01 * * * find /tmp/var/tmp ! -name . -mtime +3 -exec /bin/rm -rf  
{ } ‘;’
```

Formato do crontab

Crontab:

- Formato:
- Minuto hora dia mês dia_da_semana usuário comando.



A tabela 3.2 mostra a sintaxe utilizada nos campos do crontab. No campo dia da semana os valores 0 e 7 representam domingo e o valor 1 representa segunda-feira. Assim por diante, até sábado, que é representado pelo valor 6.

Campo	Descrição	Faixa
Minuto	Minuto da hora	0 a 59
Hora	Hora do dia	0 a 23
Dia	Dia do mês	1 a 31
Mês	Mês do ano	1 a 12
Dia da semana	Dia da semana	0 a 7

Tabela 3.2
Sintaxe do crontab.



Anacron

Anacron:

- Permite executar tarefas que não foram executadas pelo Cron durante o tempo em que um servidor ficou desligado.
- Pode ser iniciado no boot ou através do cron.
- Não é executado como um daemon, e sim como um processo normal.

Em situações onde um servidor fique desligado por um período de tempo, seja por falta de energia ou por qualquer outro motivo, as tarefas agendadas pelo Cron para serem executadas durante esse período não serão executadas quando o servidor for ligado. Para solucionar esse problema, foi criado o programa Anacron, que verifica quais tarefas não foram executadas pelo Cron durante o período de tempo em que o servidor ficou desligado e em seguida as executa.

Ao contrário do Cron, que é executado como um daemon, o Anacron é executado como um processo comum e, após sua execução, é encerrado. Portanto, é interessante automatizar sua execução através de um script a ser executado durante a inicialização do sistema ou através de uma tarefa do Cron. O Anacron também pode ser utilizado para executar tarefas, mas em servidores essa função deve ser feita pelo Cron, que é mais completo e oferece mais opções de configuração. A execução de tarefas pelo Anacron não será abordada por não fazer parte do escopo deste curso.

Fcron

Fcron:

- Possui as funcionalidades do Cron e do Anacron, resolvendo problemas de indisponibilidade dos servidores.
- Possui opções de configurações mais flexíveis, resolvendo algumas limitações do cron.

O fcron também é um daemon que foi criado para executar tarefas de modo automático e periódico, com a vantagem de unir as características do cron e do anacron e um só programa. Além disso, suas opções de configuração são mais flexíveis e resolvem alguns problemas e limitações do cron. A seguir são listadas algumas opções que podem ser utilizadas somente com o fcron, não estando disponíveis no cron.

- Permite definir um valor máximo de carga do sistema com o critério para executar uma tarefa;
- Permite definir o nice value de uma tarefa que será executada;
- Executa tarefas que não foram executadas enquanto o sistema esteve desligado;
- Possibilita o envio de e-mail aos usuários, informando o motivo de uma tarefa não ter sido executada.

Existem diversas opções além dessas apresentadas que tornam o fcron uma alternativa bastante interessante ao cron. O fcron não vem instalado por padrão nas distribuições Linux, e seu uso, configurações e funcionamento não serão abordados nesse curso. O administrador pode consultar suas páginas de manual se optar por utilizá-lo no lugar do cron.

4

Sistema de arquivos

objetivos

Diferenciar os conceitos de diretório, discos e partições; Conhecer a estrutura de diretórios típica do Linux; Saber particionar um disco e conhecer as vantagens do particionamento; Entender o conceito de sistema de arquivos no Linux; Conhecer a estrutura do arquivo */etc/fstab* e sua importância para o sistema; Saber utilizar comandos de criação de sistemas de arquivos; Aprender a usar os comandos de montagem e desmontagem de partições; Conhecer os principais tipos de sistemas de arquivos reconhecidos pelo Linux; Entender a importância e saber como utilizar os comandos *mkfs*, *fsck*, *mount*, *umount*, *du* e *df*; Saber como habilitar e configurar o esquema de quotas; Editar os limites de quotas para os usuários.

Sistema de arquivos; Estrutura dos discos; Sistema de quotas.

conceitos

Estrutura dos discos

- No Linux, cada disco é dividido em uma ou mais partições.
- Uma partição deve ser formatada com algum sistema de arquivos.
- Partições são logicamente integradas para compor uma só árvore de diretórios.
- Cada partição compõe uma subárvore, cuja raiz é o seu ponto de montagem.
- A estrutura de diretórios e arquivos é uma árvore hierárquica.
- O diretório de mais alto nível é a raiz ("/").



Todos os diretórios no Linux aparecem a seguir do diretório-raiz, representado pelo caractere "/", que é o nível mais alto da árvore de diretórios. O diretório-raiz possui arquivos e subdiretórios, que, por sua vez, possuem seus arquivos e subdiretórios, e assim sucessivamente. Por exemplo, o caminho */usr/bin/man* indica que o arquivo *man* está localizado dentro do diretório *bin*, que está localizado dentro do diretório *usr*, que está localizado no diretório */*.

Assim como no Windows, a estrutura de arquivos e diretórios no Linux pode ser representada por uma árvore hierárquica. Porém, enquanto as partições de disco no Windows são visualizadas como unidades de disco independentes, com árvores de diretórios distintas, as partições de disco no Linux são logicamente integradas para compor uma árvore de diretórios única. Nessa árvore, cada partição compõe uma subárvore, cuja raiz é o seu ponto de montagem (mount point), que é definido pelo administrador. Assim, o conjunto de subárvores contidas nas

partições compõe a árvore de diretórios do Linux. É necessário ter, no mínimo, uma partição montada e essa deve obrigatoriamente apontar para o diretório-raiz. Também é possível criar outras partições e montá-las como for mais conveniente. Assim, se o administrador definir que o ponto de montagem de uma determinada partição é o diretório `"/home"`, então toda a subárvore armazenada nessa partição será acessada a partir do diretório `"/home"`.

A figura 4.1 mostra um esquema hipotético de organização de diretórios. Os retângulos representam diretórios e as áreas sombreadas representam as diferentes partições dos discos. Note que a estrutura faz uso de dois discos e três partições. Os pontos de montagem são `/usr/src` para a partição 2 do disco 1 e `/home` para a partição 1 do disco 2. O segundo disco pode estar instalado no mesmo servidor ou em algum servidor da rede, que provê o acesso à partição remota utilizando o protocolo Network File System (NFS).

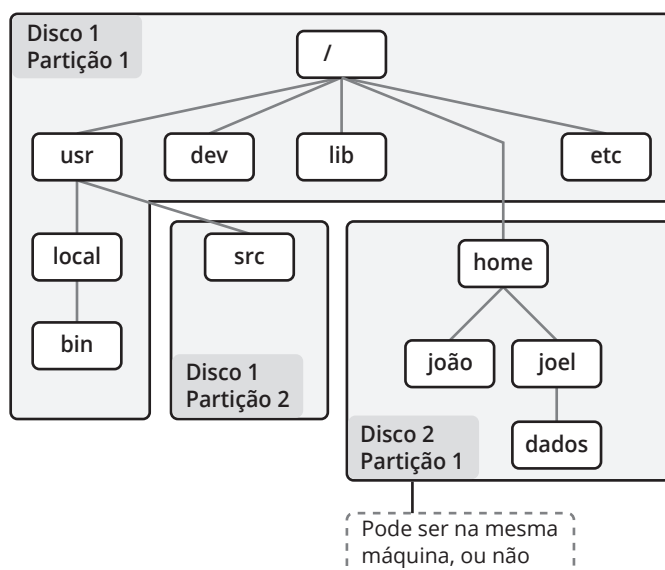


Figura 4.1
Estrutura de
diretórios do Linux.

Particionamento

- O particionamento e a estrutura de diretórios são definidos durante a instalação do sistema.
- Vantagens do particionamento de disco:
 - ▣ Isolamento de falhas.
 - ▣ Maior facilidade para execução de backups.
 - ▣ Ganho de performance no acesso aos dados.
- Comando `fdisk`:
 - ▣ Utilizado para particionar um disco.

Os recursos do sistema de arquivos do Linux permitem que o administrador escolha a implementação física e lógica mais adequada para a estrutura de arquivos, de forma totalmente transparente para o usuário. O administrador também pode montar qualquer partição diretamente sob o diretório-raiz, atribuindo-lhe o nome que melhor represente seu conteúdo, como, por exemplo, `"/dados"`. O particionamento de discos é extremamente vantajoso. Em caso de falha em uma determinada partição, as demais partições não são afetadas. Assim, caso aconteça um problema físico na partição que armazena os arquivos dos usuários (`"/home"`), basta o administrador recuperar essa partição específica e em seguida,

se necessário, recuperar os arquivos do backup para restaurar o conteúdo dessa partição. Por outro lado, caso o sistema possua uma única partição, provavelmente o administrador terá de reinstalar o sistema e recuperar todos os arquivos do backup. No Linux, um dos comandos utilizados para o particionamento de um disco é o *fdisk*. A tabela 4.1 mostra os principais diretórios padrão do Linux.

/boot	Arquivos de inicialização do sistema e imagem do kernel
/bin	Utilitários do sistema
/sbin	Ferramentas de administração
/usr	Utilitários e ferramentas de administração adicionais
/etc	Arquivos de configuração de serviços
/dev	Arquivos de dispositivos do sistema
/lib	Bibliotecas de funções compartilhadas
/home	Diretório de trabalho dos usuários
/var	Logs do sistema, e-mails dos usuários etc.
/tmp	Arquivos temporários
/root	Diretório de trabalho do usuário root
/proc	Diretório virtual que representa as estruturas de dados do kernel

Tabela 4.1
Diretórios padrão
do sistema.

Tipos de partição

Existem três tipos de partição que serão vistos a seguir.

- **Partição primária:** é o principal tipo de partição e contém algum tipo de sistema de arquivos associado. Em um disco deve haver no mínimo uma e no máximo quatro partições primárias. Caso existam quatro partições primárias em um disco, esse não poderá conter mais nenhum outro tipo de partição. Uma das partições primárias deve ser marcada como inicializável para que o Sistema Operacional possa ser carregado. No Linux as partições primárias são identificadas em um disco IDE, por exemplo, como hda1, hda2, hda3 e hda4;
- **Partição estendida:** é um tipo especial de partição que não pode conter sistemas de arquivos, mas contém até 255 partições lógicas. Um disco pode conter somente uma partição estendida. É importante ressaltar que em um disco que contenha uma partição estendida só poderão existir três partições primárias. Ou seja, esse tipo de partição ocupa o lugar de uma partição primária no disco. No Linux a partição estendida pode ser identificada em um disco IDE, por exemplo, como hda1, hda2, hda3 ou hda4;
- **Partição lógica:** também conhecidas como unidades lógicas, esse tipo de partição fica localizado dentro da partição estendida e deve conter algum sistema de arquivos. No Linux, as partições lógicas são identificadas em um disco IDE, por exemplo, de hda5 até hda16.

Sistema de arquivos

- Abstração utilizada pelo kernel para representar e organizar os recursos de armazenamento do sistema (discos, unidades de CD-ROM, pen drives etc.).
- Composto por uma árvore hierárquica de diretórios, que tem como seu nível mais alto o diretório "/" (raiz).



O sistema de arquivos ou filesystem representa a organização lógica de uma partição, que define como os arquivos são armazenados e acessados nos blocos de disco. Assim, para manipular os dados de um determinado tipo de sistema de arquivos, o Sistema Operacional deve suportar a organização lógica desse sistema de arquivos.

Uma das principais características do Linux é o suporte a diferentes tipos de sistemas de arquivos, o que permite ao administrador configurá-lo para acessar dados em servidores executando outros tipos de Sistemas Operacionais. Complementado pelo serviço Network File System (NFS), o Linux suporta a montagem de diretórios disponíveis remotamente em outros servidores Linux através da rede.

Para suportar os vários tipos de sistemas de arquivos, o Linux agrega a cada tipo um módulo de software responsável por traduzir seus formatos para um formato único denominado Virtual File System (VFS). Assim, o administrador pode utilizar o comando *mount* para montar diversos tipos de sistemas de arquivos simultaneamente, criando a árvore de diretórios única do Linux. As informações que descrevem as partições, seus tipos de sistemas de arquivos e seus pontos de montagem são armazenadas no arquivo */etc/fstab*. Nesse arquivo, cada partição é descrita em uma linha que possui o seguinte formato:

Filesystem	Mountpoint	Type	Options	Dump	Fsck
------------	------------	------	---------	------	------

Esses campos são descritos a seguir:

- **Filesystem:** partição a ser montada. Em alguns sistemas, também é chamado de *device*, porque diz respeito a algum dispositivo referenciado por meio de um arquivo presente no diretório *"/dev"*;
- **Mountpoint:** ponto de montagem;
- **Type:** tipo do sistema de arquivos a ser montado;
- **Options:** lista de atributos funcionais (por exemplo: o atributo *rw*, significa que o sistema de arquivos deve ser montado para leitura e escrita);
- **Dump:** define se o comando *dump* deve considerar (valor 1) ou não (valor 0) a partição para a realização de backups;
- **Fsck:** indica a ordem de prioridade na qual a partição será verificada pelo comando *fsck* a partir da inicialização do sistema. O número 1 representa a maior prioridade e, à medida que esse número aumenta de valor, a prioridade diminui. O número 0 indica que a partição não deve ser verificada.

Na inicialização do sistema, cada linha desse arquivo é processada para montar as partições, compondo assim a estrutura de diretórios do sistema. Cada dispositivo no Linux é reconhecido por um nome. Discos IDE, por exemplo, são nomeados da seguinte maneira:

- **/dev/hda:** unidade de disco *master* da primeira controladora IDE;
- **/dev/hdb:** unidade de disco *slave* da primeira controladora IDE;
- **/dev/hdc:** unidade de disco *master* da segunda controladora IDE;
- **/dev/hdd:** unidade de disco *slave* da segunda controladora IDE.

Dispositivos Small Computer System Interface (SCSI), Serial Attachment Technology Advanced (SATA) e pen drives são denominados *sda*, *sdb*, *sdc* etc. As unidades de disco flexível e CD-ROM são denominadas */dev/fd0* e */dev/cdrom*, respectivamente. As partições dos discos são também identificadas por um nome de partição, formado pelo nome do disco seguido de um número inteiro. Por exemplo, as partições do disco IDE master conectado na primeira controladora são denominadas */dev/hda1*, */dev/hda2* etc. Veja a seguir um exemplo de arquivo *fstab*.

#Filesystem	Mountpoint	Type	Options	Dump	Fsck
/dev/hda1	/	ext2	rw	1	1
/dev/hda2	/usr	ext2	rw	1	2
/dev/hdc	none	swap	rw	0	0
/dev/hdb	/home	ext2	rw	1	1
/dev/sda1	/dosc	msdos	defaults	0	0

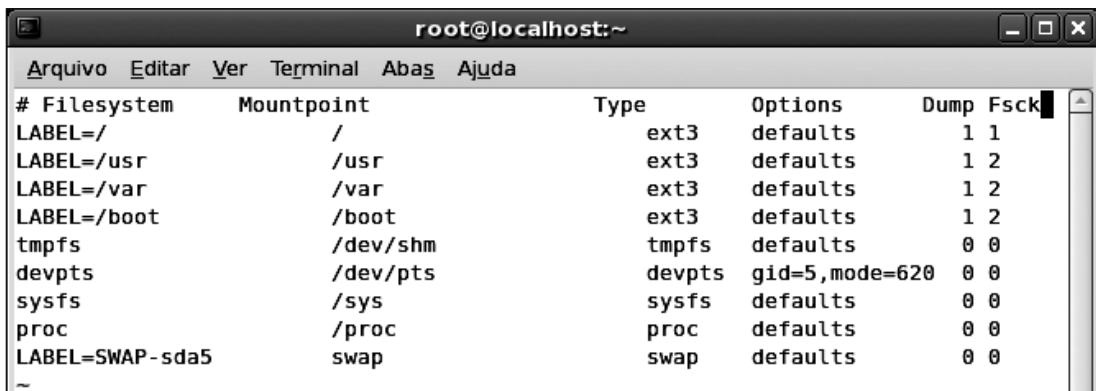
Algumas distribuições mais recentes do Linux, como o CentOS e o Ubuntu, apresentam uma ligeira mudança no arquivo */etc/fstab*. No campo “Filesystem”, o nome de dispositivo que referencia a partição é substituído por um label ou pelo Universally Unique Identifier (UUID) do hardware. Esse novo padrão foi adotado por conta de mídias de armazenamento removíveis, que podem ser associados a diferentes nomes de dispositivos quando são conectadas ao sistema. Sendo assim, não temos como referenciar no arquivo */etc/fstab* um pen drive, por exemplo, pelo mesmo nome de dispositivo sempre, o que impossibilita sua montagem automática. Para evitar esse tipo de problema, algumas distribuições utilizam a notação de labels ou UUIDs. Para verificarmos o label (quando disponível) e o UUID de um dispositivo, devemos conectá-lo ao computador e em seguida verificar a qual dispositivo ele foi associado. Basta então executar o comando *blkid*, como mostra o exemplo a seguir.

Saiba mais

Para obter mais informações sobre labels, consulte a página de manual do comando *e2label*.

```
# blkid /dev/sdb1
/dev/sdb1: UUID="FABC69BDBC697553" TYPE="ntfs"
```

Manteremos a nomenclatura que referencia as partições por seus nomes de dispositivo, por ser didaticamente mais interessante. A figura 4.2 mostra um exemplo de arquivo */etc/fstab* que utiliza labels para identificar as partições.



# Filesystem	Mountpoint	Type	Options	Dump	Fsck
LABEL=/	/	ext3	defaults	1	1
LABEL=/usr	/usr	ext3	defaults	1	2
LABEL=/var	/var	ext3	defaults	1	2
LABEL=/boot	/boot	ext3	defaults	1	2
tmpfs	/dev/shm	tmpfs	defaults	0	0
devpts	/dev/pts	devpts	gid=5,mode=620	0	0
sysfs	/sys	sysfs	defaults	0	0
proc	/proc	proc	defaults	0	0
LABEL=SWAP-sda5	swap	swap	defaults	0	0

Figura 4.2
Arquivo */etc/fstab*.

Comandos

Comando *mkfs*:

- Cria, em uma partição de um disco, a estrutura lógica correspondente ao sistema de arquivos a ser utilizado.

Comando *fsck*:

- Verifica a consistência e repara possíveis erros em um sistema de arquivos.
- Pode ser rodado automaticamente na inicialização do sistema.

Comando *mount*:

- Monta uma partição.

Comando *umount*:

- Desmonta uma partição.

Há uma série de comandos que são utilizados para trabalhar com sistemas de arquivos. Entre eles, podemos destacar os seguintes:

mkfs

O comando *mkfs* cria em uma partição de um disco a estrutura lógica correspondente ao sistema de arquivos especificado na linha de comando. O nome da partição também deve ser especificado como parâmetro para o comando *mkfs*.

fsck

Uma vez que, para aumento de desempenho, o kernel costuma armazenar blocos de dados que deveriam ser salvos em disco, na memória do sistema, a imagem mais recente do sistema de arquivos encontra-se dividida entre o disco e a memória. Assim, caso haja falha de energia, pequenas inconsistências podem ser introduzidas no sistema de arquivos, uma vez que informações armazenadas na memória não puderam ser salvas no disco.

O comando *fsck* (file system consistency check) analisa um filesystem à procura de erros e, caso encontre algum, tenta corrigi-lo. O nome da partição deve ser informado como parâmetro para o comando *fsck*. Na inicialização do sistema, as partições marcadas para serem verificadas no arquivo */etc/fstab* podem ser verificadas automaticamente pelo comando *fsck*, que examina e corrige eventuais erros. Os sistemas de arquivos são verificados obedecendo à ordem numérica crescente indicada no campo "Fck" do arquivo */etc/fstab*. Se dois sistemas de arquivos estão localizados em diferentes discos, pode ser fornecido a eles um mesmo número de sequência, que fará com que o *fsck* verifique os dois simultaneamente. A partição raiz deve sempre ser verificada antes de qualquer outra. Os erros mais comuns encontrados em sistema de arquivos corrompidos são:

- Inodes não referenciados;
- *Link counts* inexplicavelmente grandes;
- Blocos de dados não utilizados não registrados nos mapas de blocos;
- Blocos de dados registrados como livres, mas que estão sendo utilizados por algum arquivo;
- Informação de resumo incorreta no superbloco.

Erros que não se enquadram em uma das cinco categorias citadas são potencialmente perigosos. Na sua detecção, o *fsck* entra no modo interativo e solicita ao administrador que confirme cada um dos reparos que ele executa. Nesse caso, são grandes as chances de perda de dados importantes do sistema. Alguns erros graves são listados a seguir.

- Blocos reclamados por mais de um arquivo;
- Blocos reclamados por um arquivo, mas que estão fora do sistema de arquivos;
- *Link counts* muito pequenos;
- Blocos não contabilizados;
- Diretórios referentes a inodes não alocados;
- Vários erros de formato.



Infelizmente, é impossível reparar um disco sem ter o conhecimento profundo do modo como o sistema de arquivos é implementado. Não há, portanto, muito a fazer a não ser aceitar os reparos sugeridos pelo *fsck*. É possível tentar minimizar os problemas, registrando as mensagens exibidas pelo comando, uma vez que elas costumam indicar os arquivos problemáticos detectados. Se o *fsck* pede permissão para remover um arquivo, é preciso tentar copiá-lo para outra partição, antes de permitir a sua remoção. Deve-se estar atento para o fato de que, ao tentar acessar um arquivo danificado, corre-se o risco de travar o sistema.

mount

O comando *mount* é utilizado para montar uma partição, que pode conter qualquer tipo de sistema de arquivos suportado pelo Linux. A princípio, a partição pode ser montada em qualquer ponto da árvore de diretórios, desde que não utilize nenhum nome de diretório existente e respeite o esquema de permissões de arquivos e diretórios do Linux.

umount

O comando *umount* é utilizado para desmontar uma partição que tenha sido anteriormente montada. Quando uma partição é desmontada, sua subárvore de diretórios passa a não mais fazer parte da árvore de diretórios principal. É importante ressaltar que uma partição só pode ser desmontada se não possuir nenhum recurso sendo utilizado pelo sistema.

du e df

Comando *du*:

- Mostra o tamanho de arquivos e diretórios.
- Útil para verificação dos maiores utilizadores de espaço em disco.

Comando *df*:

- Lista o uso de espaço em disco de cada partição.
- Exemplo:
 - ▣ `# df -k`

Para verificar a taxa de ocupação dos diretórios e partições, o administrador pode utilizar os comandos *du* e *df*. O primeiro percorre a árvore do diretório especificado e identifica o espaço ocupado pelos seus arquivos e subdiretórios. O segundo fornece informações sobre o espaço livre e o espaço utilizado nas partições. O comando *df* também pode mostrar a taxa de utilização de inodes por partição. Para obter mais informações sobre esses comandos, suas páginas de manual podem ser consultadas. Os exemplos a seguir mostram as saídas dos comandos *df* e *du*, respectivamente.

```
# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       32G   9.4G   21G   32% /

# df -i
Filesystem      Inodes   IUsed   IFree IUse% Mounted on
/dev/sda1       2039808  181908 1857900    9% /

# du -hcs /var/*
4.0K    /var/backups
48M     /var/cache
```

Saiba mais

Para obter mais informações sobre esses comandos, suas páginas de manual podem ser consultadas.

4.0K	/var/crash
81M	/var/lib
4.0K	/var/local
0	/var/lock
3.9M	/var/log
4.0K	/var/mail
4.0K	/var/opt
36K	/var/run
28K	/var/spool
4.0K	/var/tmp
132M	total

Tipos de sistemas de arquivos suportados

A tabela 4.2 mostra os principais tipos de sistemas de arquivos suportados pelo Linux.

Sistema de Arquivos	Descrição
ext2, ext3 e ext4	Sistemas de arquivos padrão utilizados no Linux, sendo a versão atual o ext4. A partir da versão ext3, já possui suporte a journaling. O journaling mantém um registro de todas as alterações realizadas no sistema de arquivos, o que facilita a recuperação em situações onde este não foi desmontado corretamente, causando inconsistências de dados.
Minix	Primeiro sistema de arquivos do Linux. Foi desenvolvido por Andrew Tanenbaum.
msdos, vfat e ntfs	Sistemas de arquivos que permitem integração com MS-DOS e Windows. Atualmente disponível apenas para leitura.
ISO 9660	Sistema de arquivos padrão utilizado em CDs e DVDs.
proc	Sistema virtual de arquivos do Linux. Não ocupa espaço no disco, pois os arquivos são criados instantaneamente quando acessados. Serve para prover informações sobre a configuração do kernel, ou até mesmo sobre o estado dos dispositivos.
nfs	Sistema de arquivos utilizado para acessar arquivos armazenados remotamente em redes baseadas em sistemas Unix.
ufs	Sistema de arquivos utilizado para acessar arquivos armazenados em sistemas BSD.
smbfs	Sistema de arquivos utilizado para montar compartilhamentos do Windows.
ReiserFS	Primeiro sistema de arquivos com suporte a journaling utilizado no Linux.
xfs	Sistema de arquivos com suporte a journaling desenvolvido pela Silicon Graphics.
swap	Sistema de arquivos utilizado para troca e dados entre a memória e o disco.

Saiba mais

A partir do kernel 2.6.0, existem drivers disponíveis que permitem que o Linux escreva em volumes NTFS, entretanto, não são consideradas soluções completamente seguras e por isso não são recomendadas para ambientes de produção.

Tabela 4.2
Tipos de sistemas de arquivos suportados.

Sistema de quotas



- Administra o uso de espaço em disco pelos usuários.
- Como preparar o sistema.
- De que forma habilitar o uso de quotas.
- Como editar as quotas dos usuários.
- De que forma verificar as quotas dos usuários.

O sistema de quotas é empregado para limitar o consumo de espaço em disco pelos usuários. Esse recurso é especialmente útil em sistemas multiusuários, nos quais é necessário administrar a ocupação do espaço em disco por cada um dos usuários, para evitar que eles utilizem todo o espaço existente, prejudicando o funcionamento geral do sistema. Existem dois tipos de limitações que podem ser impostas: número máximo de arquivos (inodes) e espaço máximo utilizado em disco (blocos de disco). A seguir, são listadas as quatro versões de quotas que podem ser habilitadas em um sistema Linux.

- **Versão 1 ou vfsold:** essa foi a primeira versão do sistema de quotas do Linux. Essa versão é mais utilizada em kernels da série 2.4 ou anteriores;
- **Versão 2 ou vfstp0:** é a versão atual do sistema de quotas do Linux. Essa versão possui melhor desempenho, especialmente em partições de grande tamanho. O conteúdo dessa sessão será baseado na versão 2 ou vfstp0;
- **RPC:** esse tipo de versão é utilizado somente em partições que possuem compartilhamento de arquivos através da rede, utilizando o protocolo Network File System (NFS);
- **XFS:** esse tipo de versão é utilizado somente em sistemas de arquivos XFS.

A configuração do sistema de quotas pode ser realizada utilizando comandos específicos do Linux ou alguma ferramenta gráfica como o Webmin, que será visto na sessão de aprendizagem 10. Nesta sessão, veremos como configurar o sistema de quotas utilizando a linha de comandos.

Preparando o sistema



Preparando o sistema:

- O kernel deve estar configurado.
 - Padrão nas versões 2.4 e 2.6.
- O pacote “quota” deve estar instalado.

As versões 2.4 e 2.6 do kernel já suportam nativamente o sistema de quota, mas nas versões mais antigas o kernel deve estar configurado corretamente para suportar o sistema de quotas. Na sessão de aprendizagem 8 serão apresentados os métodos para configurar e recompilar o kernel caso seja necessário. Levando em conta que será utilizada uma versão de sistema com kernel 2.6, não será necessária a sua recompilação. Além do suporte no kernel, o pacote quota deve estar instalado no Sistema Operacional.

Habilitando



- /etc/rc.local.
- /etc/fstab.
- Informar se o controle será por usuários (usrquota), por grupos (grpquota) ou ambos.
- Controle realizado por partição.

- A configuração deve ser feita no campo “Options”.
- Filesystem do tipo *ext2* ou *ext3*.



No Linux, a habilitação do sistema de quotas deve ser realizada durante a inicialização do sistema e pode ser incluída no script */etc/rc.local*. Esse script ativa os comandos que habilitam o sistema de quotas. Dessa forma, sempre que o sistema for inicializado, as quotas serão habilitadas. É importante ressaltar que o sistema de quotas deve ser habilitado após a montagem das partições definidas no arquivo */etc/fstab*.



É recomendável habilitar o sistema de quotas ao final do arquivo */etc/rc.local* ou imediatamente após a montagem de todas as partições.

Uma vez configurada a habilitação do sistema de quotas, é preciso informar em quais partições ele deve ser ativado. Essas partições são indicadas no arquivo */etc/fstab*, que também guarda informações sobre a montagem das partições. Além de indicar que o controle de quotas está ativo na partição, é igualmente necessário informar se o controle será realizado por usuário, por grupo ou por ambos. Deve ser ressaltado que o controle de quotas é feito por partição, e não por diretório. Como visto anteriormente, o arquivo */etc/fstab* é semelhante a uma tabela, em que cada linha descreve uma partição e cada coluna representa um campo com características da partição. A configuração do sistema de quotas é realizada no campo “Options”.

Para cada partição em que o sistema de quotas for ativado, o administrador deve criar, na raiz dessas partições, arquivos vazios com os seguintes nomes: *aquota.user* e *aquota.group*. Esses arquivos são utilizados pelo sistema de quotas para manter os limites de quotas dos usuários e dos grupos na partição e para garantir a utilização corrente da partição por esses usuários e grupos. Esses arquivos devem ter permissão de leitura e escrita apenas para o usuário root. Para estabelecer um sistema de quotas na partição “/” do sistema, por exemplo, o administrador deve fazer as alterações necessárias no arquivo */etc/fstab* e executar a sequência de comandos a seguir:

```
# touch /aquota.user
# touch /aquota.group
# chmod 600 /aquota.user
# chmod 600 /aquota.group
```

Esses comandos criam arquivos de quotas vazios.

Inicializando

- Criação dos arquivos *aquota.user* e *aquota.group* na raiz da partição “/home”:
 - # touch /home/aquota.user
 - # touch /home/aquota.group
 - # chmod 600 /home/aquota.user
 - # chmod 600 /home/aquota.group
- Desmontar e montar todas as partições com quota habilitada.
- Reiniciar o sistema se for habilitada quota na partição raiz.





- Inserir no arquivo */etc/rc.local*:
 - ▣ Comando *quotacheck*:
 - ▣ Inicializa os arquivos *aquota.user* e *aquota.group*.
 - ▣ *# quotacheck -augv*
 - ▣ Comando *quotaon*:
 - ▣ Ativa o suporte a quota de disco.
 - ▣ *# quotaon -augv*

Após a criação dos arquivos *aquota.user* e *aquota.group*, todas as partições com quota habilitada devem ser desmontadas e montadas novamente, para que as alterações entrem em vigor. Se for habilitado o sistema de quotas na partição raiz, o sistema deve ser reiniciado, já que essa partição não pode ser desmontada com o sistema em uso. Em seguida, o administrador deve usar o comando *quotacheck* com a opção *-a* para inicializar os arquivos de controle de quotas criados, com a situação corrente de utilização das partições. Se for utilizado com as opções *-augv*, cria ambos os arquivos e inicializa-os com os dados atuais de uso de disco por usuário e grupo. Para finalizar o processo, o comando *quotaon* deve ser executado com as opções *-augv* para ativar o suporte a quota de disco em todas as partições.

Editando quotas



Comando *edquota*:

- Edita as quotas de usuários e de grupos.

Comando *warnquota*:

- Verifica as quotas de usuários em todas as partições com quota habilitada.
- Envia por e-mail uma mensagem de alerta aos usuários que tenham ultrapassado seus soft limits.

Uma vez que o sistema de arquivos foi configurado para suportar quotas, é necessário definir os limites de uso para usuários e grupos do sistema. Os limites de quotas são especificados por dois parâmetros: *soft* e *hard*.

O limite *soft* pode ser ultrapassado por um determinado intervalo de tempo (*grace period*). Já o limite *hard* nunca pode ser ultrapassado. Esses limites são definidos para o número de inodes e para o número de blocos que o usuário ou grupo pode alocar. Sempre que a utilização de espaço em disco do usuário ultrapassa o limite *soft*, o sistema envia uma notificação ao usuário, indicando que esse limite foi ultrapassado.

A partir desse momento, o sistema de quotas permite que o usuário permaneça acima do limite *soft* apenas durante o *grace period*. Se esse período for esgotado e o usuário ainda estiver acima do limite *soft*, o sistema bloqueia qualquer operação de escrita em disco do usuário, até que esse remova parte dos arquivos e passe a ocupar um espaço a seguir do limite *soft*. Durante o *grace period*, se o usuário atingir o limite *hard*, imediatamente o sistema bloqueará qualquer operação de escrita em disco do usuário, até que esse remova arquivos e passe a ocupar menos espaço que o limite *soft*.

O comando *warnquota* pode ser utilizado para alertar os usuários que tenham ultrapassado seus soft limits. Esse comando analisa as quotas de todos os usuários, em todas as partições com quota habilitada e envia por e-mail uma mensagem alertando os usuários que tiverem ultrapassado seus soft limits. Para que esse comando cumpra sua função satisfatoriamente, ele deve ser agendado no crontab do usuário root para ser executado periodicamente.



Para especificar os limites de quotas de disco para usuários e para grupos, deve-se utilizar o comando *edquota*. Esse comando ativa o editor vi para modificar os arquivos *aquota.user* ou *aquota.group*. É possível utilizar outro editor, especificando-o na variável de ambiente editor. Por exemplo, supondo que se deseja definir quotas de disco para o usuário *aluno1*, deve-se ativar o comando *edquota* da seguinte forma:

```
# edquota -u aluno1
```

Esse comando ativa o editor vi e exibe a seguinte mensagem:

```
Quotas for user aluno1:
/dev/hda1: blocks in use: 12667, limits (soft = 0, hard = 0) inodes
in use: 1749, limits (soft = 0, hard = 0)
```

Comando *edquota*:

- Utilizado para editar a quota de usuários e grupos.
- Opções:
 - ▣ -u, -g, -t e -p
- Exemplos:
 - ▣ # *edquota -u aluno1*
 - ▣ Mostra a quota do usuário *aluno1*.
 - ▣ # *edquota -p aluno1 aluno2 aluno3*
 - ▣ Copia as configurações de quota do usuário *aluno1* para os usuários *aluno2* e *aluno3*.

Nesse exemplo, os parâmetros *soft* e *hard* estão com o valor 0, pois esses limites ainda não foram definidos. Nesse ponto, o administrador modifica os valores dos limites *soft* e *hard*, conforme planejado. Imediatamente após salvar o arquivo, as quotas estabelecidas tornam-se ativas. De forma análoga, a configuração para grupos pode ser realizada com o comando *edquota* utilizando a opção *-g* e passando o nome do grupo como parâmetro. Para editar o *grace period*, utiliza-se o comando *edquota* com a opção *-t*:

```
# edquota -t
```

É possível facilitar a definição de quotas em um sistema, fazendo uso da opção *-p* do comando *edquota*. Com essa opção, podemos definir as quotas de um usuário com base nas de outro. Exemplificando, se quisermos definir as quotas dos usuários *aluno2* e *aluno3* de forma igual às quotas do usuário *aluno1*, podemos utilizar o seguinte comando:

```
# edquota -p aluno1 aluno2 aluno3
```

Isso evita que o administrador tenha de editar os parâmetros de quota para cada usuário e facilita a definição de quotas para múltiplos usuários. Essa opção também pode ser utilizada para definir quotas para grupos. Basta utilizá-la em conjunto com a opção *-g*.

Verificação e gerenciamento

Comando *quota*:

- Utilizado para verificar as quotas de um usuário.

Comando *repquota*:

- Utilizado para verificar as quotas de uma partição.

Para verificar as quotas estabelecidas para um determinado usuário, deve ser utilizado o comando *quota*. Esse comando informa o status das quotas para um determinado usuário nas partições em que ele ultrapassou o limite estabelecido. A informação completa sobre as quotas em todas as partições pode ser obtida adicionando a opção *-v*. Usuários podem, individualmente, verificar o seu status de quota, usando esse comando quando estiverem trabalhando no sistema. Entretanto, apenas o administrador, por meio da conta root, pode ver as quotas de outros usuários. O exemplo a seguir mostra as estatísticas de uso de espaço em disco e número de arquivos do usuário *aluno1*.

```
# quota -u 1001

Disk quotas for user aluno1 (uid 1001):
Filesystem blocks quota limit grace files quota limit grace
/dev/sda2      876 100000 105000          83 10000 10500
```

O comando *repquota* pode ser usado para exibir as quotas de usuários e grupos, separadas por partição. São exibidas informações como grace time e limites soft e hard para blocos e inodes. O exemplo a seguir mostra o relatório do uso de quotas por usuário, na partição */dev/sda2*.

```
# repquota -aug

*** Report for user quotas on device /dev/sda2

Block grace time: 7days; Inode grace time: 7days

                Block limits                File limits
User           used  soft  hard  grace  used  soft  hard
-----
root  --      15431     0     0   none    5318     0     0
none
aluno1 +-      5731 200500 300000   none   11245     0     0
```


5

Backup

objetivos

Conhecer os tipos de backup e os principais tipos de mídias usadas em backups; Aprender os principais comandos do Linux usados para fazer backup; Conhecer os softwares Amanda e Bacula, e saber como configurá-los e utilizá-los para fazer backup e restore de dados; Ver os principais softwares comerciais de backup; Entender o que é uma política de backup e saber como defini-la.

Tipos de backup; Mídias de backup; Softwares de backup; Comandos do sistema; Política de backup.

conceitos

Introdução

- Política de backup é eficiente para combater:
 - ▣ Ação de vírus e outras ameaças.
 - ▣ Ataques de hackers.
 - ▣ Falhas de usuários.
 - ▣ Problemas de hardware e software.
- Política de backup baseada na frequência de modificação dos arquivos e em seu grau de importância.
- Backups podem ser realizados por meio de comandos simples ou por meio de pacotes sofisticados.



Frequentemente, recebemos informações sobre vírus devastadores e ataques de hackers, que vivem ameaçando a segurança dos sistemas. Na maioria das instituições, a informação armazenada nos computadores vale mais do que os próprios equipamentos. Planejar e executar uma eficiente política de backup dos dados é, sem dúvida, a melhor maneira para a recuperação de acidentes e desastres; falhas dos usuários; problemas de hardware e de software, constituindo uma das tarefas mais importantes de um administrador de sistemas. A política de backup deve ser baseada, principalmente, na frequência de modificação e na importância dos arquivos.

Por exemplo, para um computador de uso pessoal, a frequência de backup é mínima e considera apenas alguns arquivos do usuário. Por outro lado, um sistema bancário precisa manter diversas cópias dos arquivos de forma instantânea (utilizando técnicas de espelhamento de discos); fazer backup com frequência (provavelmente várias vezes por dia) e, em complemento, manter cópias desses backups em localizações físicas distintas. O Linux dispõe de diversos métodos que podem ser utilizados para a realização de backups.



Esses métodos vão desde comandos simples, como *tar* e *cpio* (incluídos na instalação da maioria dos sistemas), até sofisticados pacotes comerciais.



Um backup pode ser tão simples quanto copiar toda a árvore de diretórios em uma mídia de gravação, utilizando o comando *tar*, ou tão sofisticado quanto os complexos mecanismos utilizados pelos softwares profissionais de backup.

Tipos de backup

Existem três tipos de backup:

- **Completo:**
 - ▣ Todo o conteúdo é armazenado.
 - ▣ Consome grande quantidade de tempo para ser realizado.
- **Diferencial:**
 - ▣ Só é armazenado o conteúdo que foi alterado após a realização do último backup completo.
 - ▣ Em um processo de recuperação de dados, a mídia com o último backup completo também é necessária.
- **Incremental:**
 - ▣ Só é armazenado o conteúdo que foi alterado após a realização do último backup, independente de seu tipo.
 - ▣ Em um processo de recuperação de dados, várias mídias podem ser requeridas.



Existem três tipos de backups, descritos a seguir:

- **Completo:** no backup completo, todo o conteúdo é copiado para a mídia de backup utilizada. A vantagem desse tipo de backup é que a recuperação dos dados é mais rápida, já que para recuperar um conteúdo perdido logo após um backup completo só é preciso termos em mãos a mídia que foi usada para armazenar esse backup. A principal desvantagem do backup completo é a grande quantidade de tempo gasto para realizá-lo. Uma política de backup eficiente geralmente inicia um ciclo de backup com um backup completo e prossegue com backups incrementais ou diferenciais de acordo com a necessidade do administrador. Cada novo ciclo deve ser iniciado com um backup completo;
- **Diferencial:** no backup diferencial, somente os conteúdos modificados após a realização do último backup completo são copiados para a mídia de backup. Esse tipo de backup representa a diferença entre o estado atual e o último backup completo. A recuperação de dados nesse caso é um pouco mais demorada, já que para recuperar um conteúdo perdido após um backup diferencial é preciso recuperar o último backup completo e o último backup diferencial. Mesmo que o arquivo tenha sido modificado após o backup completo, esse é necessário no processo de recuperação dos dados;
- **Incremental:** No backup incremental, somente os conteúdos modificados após o último backup são copiados, independentemente se esse foi completo, diferencial ou incremental. Esse tipo de backup representa a diferença entre o estado atual e o último backup realizado. A recuperação de dados nesse caso pode ser bastante complexa, já que pode haver diversas combinações de tipos de backup realizados entre o backup incremental que se deseja recuperar e o último backup completo. Para restaurar um backup incremental, é necessário que estejam disponíveis pelo menos as mídias

utilizadas no último backup completo, no último backup diferencial entre o backup completo e o backup incremental que se deseja recuperar e todos os backups incrementais desde o último backup diferencial. Esse é um bom motivo para se evitar o uso desse tipo de backup.

Mídias de backup

- A escolha depende de alguns fatores:
 - ▣ Quantidade de dados a armazenar.
 - ▣ Confiabilidade e duração desejáveis.
 - ▣ Quanto se pretende gastar.
- É indicado o uso de mídia removível.
- Vários tipos de mídias podem ser utilizados:
 - ▣ Fitas magnéticas (DAT, DLT e LTO).
 - ▣ Mídias ópticas (CD, DVD e Blu-Ray).
 - ▣ Storage (DAS, NAS, SAN e iSCSI).



O tipo de mídia em que se deve fazer o backup do sistema também é importante. A escolha da mídia deve levar em conta fatores que vão do volume de dados que se necessita armazenar ao montante disponível para esse tipo de investimento. Considerando que os sistemas de computação estão sujeitos a uma infinidade de tipos de falhas, que podem danificar diferentes partes do hardware de uma só vez, devemos ter como pressuposto que backups precisam ser feitos em algum tipo de mídia removível.

Fazer o backup de um disco em outro disco do mesmo computador, por exemplo, embora seja melhor do que não ter nenhum backup, é uma estratégia ineficaz contra defeitos na controladora. Fazer o backup em um disco de outro computador poderia contornar esse tipo de problema. No entanto, isso não evitaria problemas, caso um pico de energia danificasse os discos de ambos os servidores. Existe uma grande variedade de mídias removíveis nas quais os arquivos de backup podem ser armazenados: fitas magnéticas (DLT, LTO etc.), discos ópticos (CD, DVD e Blu-Ray) etc. Além disso, em alguns casos o uso de um storage pode ser recomendado. Faremos, a seguir, algumas breves considerações a respeito de cada um desses tipos de mídias.

Fitas magnéticas

Linear Tape Open (LTO):

- Tecnologia de formato aberto.
- Capacidade de armazenamento nativa de até 1,5 TB e taxa de transmissão de até 140 MB/s.
- Durabilidade de 15 a 30 anos.

Digital Linear Tape (DLT):

- Desenvolvida pela DEC em 1984.
- Capacidade de armazenamento nativa de até 800 GB e taxa de transmissão de 60 MB/s.
- Durabilidade de até 30 anos.



As fitas magnéticas são as mídias mais utilizadas para a realização de backups. Suas principais vantagens em relação aos outros tipos de mídia são: grande capacidade de armazenamento, ótimo custo-benefício e alto grau de confiabilidade. A seguir, os principais tipos de fitas magnéticas:



Linear Tape Open (LTO)

Tecnologia de formato aberto, desenvolvida em conjunto pela HP, IBM e Certance. Surgiu como resposta às necessidades de armazenamento de um volume crescente de informações que contavam com soluções complicadas, como o uso de múltiplas unidades de backup ou de robôs com várias mídias, o que dificultava bastante a administração do esquema de backup de um sistema. As mídias LTO conseguem armazenar até 1,5 TB de informação de forma não comprimida em uma única fita, e possuem taxa de transferência de dados de até 140 MB/s. Essa é atualmente a melhor tecnologia de armazenamento em fita magnética e é amplamente utilizada.

Digital Linear Tape (DLT)

A tecnologia DLT foi desenvolvida pela DEC em 1984. Atualmente, dispositivos de backup baseados nessa tecnologia são fabricados por diferentes empresas. As gerações mais recentes desses dispositivos conseguem armazenar até 800 GB de informação de forma não comprimida, a uma taxa de transmissão de 60 MB/s, podendo chegar a 1,6 TB e a 120 MB/s de transmissão, se for utilizada a compressão de dados.

Mídias ópticas

CD/DVD:

- Mídia popular.
- Baixa capacidade de armazenamento.

Blu-Ray:

- Mídia que vem se popularizando.
- Maior capacidade que a do DVD, mas bem menor que a oferecida pelas fitas magnéticas.
- Custo por gigabyte relativamente alto.

As mídias ópticas são mais utilizadas para a realização de backups em casos específicos. Elas não costumam ser utilizadas em rotinas de backup, devido à sua baixa capacidade de armazenamento. Suas principais vantagens em relação aos outros tipos de mídia são: baixo custo das mídias e dos dispositivos de gravação. A seguir serão vistos os principais tipos de mídias ópticas:

CD

Esse tipo de mídia é bastante popular e pode ser lida, praticamente, em qualquer computador doméstico. A sua principal desvantagem, no entanto, é a capacidade de armazenamento de somente 700 MB, o que torna sua utilização inviável para armazenamento de grandes volumes de dados. Outra característica que inviabiliza seu uso em sistemas de backup é sua baixa taxa de transferência de dados. Embora esse tipo de mídia não seja particularmente bom para a execução de backups regulares, o CD pode ser utilizado para guardar informações de ex-usuários do sistema. Existem tipos de CDs que pode ser regravados (RW), o que torna seu uso mais interessante.

DVD

As unidades de DVD possuem capacidade de armazenamento consideravelmente maior do que a das mídias de CD, chegando a 17,08 GB nos DVDs dual layer/double side. Ainda assim, não são muito utilizados em backups devido à baixa capacidade de armazenamento e à baixa taxa de transferência de dados. Geralmente são utilizados em ocasiões específicas, como para armazenar dados de ex-usuários. Também podem ser reutilizados.



Blu-Ray

Disco óptico de tamanho igual ao CD e ao DVD, criado para reprodução de vídeo de alta definição e armazenamento de dados de alta densidade, podendo armazenar até 50 GB de dados. Sua principal desvantagem é o alto custo por gigabyte, além de possuir capacidade de armazenamento pequena quando comparada à das mídias magnéticas.

Storage

- Hardware para armazenamento de dados.
- Alto desempenho no acesso aos dados.
- Escalabilidade.
- Tecnologias de transferência de dados: SCSI, Fibre Channel, SAS e SATA.
- Tipos de conexão: DAS, NAS, SAN e iSCSI.
- Provê redundância de dados: Redundant Array of Inexpensive Disks (RAID).



O Storage é um dispositivo de alto desempenho, composto por um conjunto de discos, e é projetado especialmente para armazenar dados. A maioria dos storages é compatível com as tecnologias SCSI, Fibre Channel, Serial Attached SCSI (SAS) e SATA. O storage é um dispositivo altamente escalável, podendo ter sua capacidade expandida facilmente, através da compra de novos discos ou até mesmo de novas unidades de armazenamento (storages adicionais). Quanto ao tipo de conexão, os storages podem ser divididos em duas categorias: conexão local, que utiliza a tecnologia DAS e conexão remota, que pode utilizar as tecnologias NAS, SAN e iSCSI.

- **Direct Attached Storage (DAS):** são dispositivos de armazenamento externos ligados diretamente ao servidor de backup, através de uma controladora de discos. O DAS trabalha com transferência de dados no nível de bloco e não permite compartilhar seu espaço de armazenamento com outros servidores. A taxa de transferência de dados é limitada pela velocidade da controladora;
- **Network Attached Storage (NAS):** dispositivos de armazenamento externos ligados ao servidor de backup através da rede. O NAS pode ser considerado um appliance, que possui um Sistema Operacional próprio, otimizado para a função de armazenamento de dados. O NAS trabalha com transferência de dados no nível de arquivos e utiliza os protocolos NFS e SMB/CIFS para prover dados a sistemas Linux e Windows respectivamente. A área de armazenamento de um NAS pode ser compartilhada entre vários servidores. A taxa de transferência de dados é limitada pela velocidade do link usado para interligar o servidor e o storage (podendo alcançar até 10 Gbps em uma rede local);
- **Storage Area Network (SAN):** é uma rede composta por dispositivos de armazenamento externos ligados ao servidor de backup através de switches especiais que utilizam a tecnologia fibre channel para realizar a transferência de dados entre o servidor e o storage. Assim como no DAS, em uma SAN as transferências de dados são feitas no nível de bloco. Uma das vantagens da SAN é a possibilidade de compartilhar a capacidade de armazenamento do storage entre vários servidores. Sua principal desvantagem é o seu alto custo. A taxa máxima de transferência de dados do fibre channel é de 6,8 Gbps;



- **internet Small Computer System Interface (iSCSI):** um protocolo que utiliza a rede IP para transportar comandos SCSI entre o storage e o servidor. Sendo assim, é possível que o servidor se comunique com os discos do storage utilizando o padrão SCSI, como se esses discos estivessem fisicamente localizados nesse servidor. O protocolo iSCSI é usado por dispositivos de armazenamento externos ligados ao servidor de backup através da rede IP. Assim como no DAS e em uma SAN, as transferências de dados do iSCSI são feitas no nível de bloco. A taxa de transferência de dados é limitada pela velocidade do link utilizado para interligar o servidor e o storage (podendo alcançar até 10 Gbps em uma rede local).

Os storages também podem fazer uso da arquitetura Redundant Array of Inexpensive Disks (RAID) para prover redundância de dados. O RAID nada mais é do que a combinação de dois ou mais discos para formar uma unidade lógica de armazenamento. O RAID pode ser implementado por hardware ou por software, e além de redundância também traz ganho de desempenho no acesso aos dados, dependendo de como for configurado. Os principais e mais utilizados tipos de RAID serão descritos a seguir.

- **RAID 0:** também conhecido como striping, nada mais é do que a junção de dois ou mais discos, formando uma unidade lógica de armazenamento. A capacidade da unidade lógica é a soma das capacidades individuais de cada disco. Esse tipo de RAID não oferece redundância e, se algum dos discos falhar, todo o conteúdo é perdido. Por outro lado, seu desempenho no acesso aos dados é maior, já que o conteúdo está dividido em vários discos que podem ser lidos simultaneamente. Além disso, não há desperdício de armazenamento, já que a capacidade total de todos os discos é utilizada;
- **RAID 1:** é também conhecido como mirror e necessita de dois discos para ser implementado. Ele utiliza o mecanismo de espelhamento de disco, ou seja, todos os dados são gravados simultaneamente nos dois discos. O RAID 1 oferece redundância de dados, pois mesmo que um dos discos falhe, todos os dados podem ser recuperados no outro disco. Por outro lado, a capacidade de armazenamento é reduzida pela metade, já que o segundo disco possui o mesmo conteúdo do primeiro. Além disso, o desempenho é menor, já que os dados são gravados duas vezes. O RAID 1 também pode ser utilizado em conjunto com o RAID 0, formando dois conjuntos com o mesmo número de discos. Os discos de cada conjunto formam, através do RAID 0, duas unidades lógicas de armazenamento. É extremamente recomendável que a soma das capacidades dos discos dos dois conjuntos seja igual, caso contrário, quando o RAID 1 for habilitado entre as duas unidades lógicas, ele vai considerar sempre o tamanho da menor unidade. Sendo assim, a unidade de maior capacidade ficará subutilizada;
- **RAID 5:** utiliza um mecanismo de paridade, onde informações extras sobre os dados são armazenadas, de modo distribuído entre todos os discos, como forma de redundância. O desempenho para leitura de dados aumenta, já que os dados estão espalhados entre os discos. No entanto, as operações de escrita são mais lentas, devido ao tempo gasto com as informações de paridade. O RAID 5 é um meio termo entre o RAID 0 e o RAID 1. Ele oferece redundância, mas sem desperdiçar metade do espaço total, como ocorre no RAID 1. O RAID 5 deve ser formado por pelo menos três discos e, caso um deles falhe, os dados ainda podem ser acessados, porém, com queda no desempenho nas operações de leitura e escrita devido à atividade de recuperação das informações de paridade realizada pela controladora de discos em tempo real. Para calcular o espaço perdido no RAID 5, deve-se aplicar a seguinte fórmula: $1: 1/n$, onde n representa o número de discos utilizados. O RAID 5 pode utilizar ainda um mecanismo chamado de hot spare, que consiste em disco extra que só é utilizado em caso de falha de algum disco. Quando a falha ocorre, a controladora substitui o disco com problema pelo disco extra, gravando neste último todos os dados do disco que falhou.

Comandos do sistema

Existem alguns comandos do Linux que podem ser utilizados para fazer backup de dados. No entanto, em ambientes mais complexos, é extremamente recomendável o uso de algum software profissional de backup. A seguir, serão vistos os principais comandos do Linux que podem ser usados para realizar backups.

Comando tar

Comando tar:

- Realiza backup de arquivos e diretórios.

Opções básicas:

- **-c**: cria um arquivo *tar*.
- **-t**: lista o conteúdo de um arquivo *tar*.
- **-x**: extrai os arquivos de um arquivo *tar*.
- **-z** e **-j**: comprimem dados usando os comandos *gzip* e *bzip2*, respectivamente.
- **-f**: especifica o nome do arquivo *tar* que será criado.

Realização de backups por meio do *cron*, exemplo:

```
# crontab -e  
  
00 01 * * 1-5 tar -czf /dev/rst0 /home
```

Uso do tar para espelhamento de árvore. Exemplo:

```
# tar -cv . | (cd /espelho; tar -xvp )
```

Extraindo o arquivo *documento.txt* do aluno1, armazenado na fita para o diretório */tmp*, exemplo:

```
# cd /tmp ; tar -xf /dev/rst0 /home/aluno1/documento.txt
```

O comando *tar* (tape archive), apesar de simples, ainda é utilizado pelos administradores de sistemas para a realização de rotinas de backup. Ele pode ser usado para realizar o backup diário de um servidor, transferindo-o para uma unidade de fita de forma automática, usando o daemon *cron*. A seguir, serão listadas as principais opções do comando *tar*.

- **-c**: cria um arquivo *tar*;
- **-t**: lista o conteúdo de um arquivo *tar*;
- **-x**: extrair os arquivos de um arquivo *tar*;
- **-z e -j**: comprimir dados usando respectivamente os comandos *gzip* e *bzip2*;
- **-f**: utilizada para especificar o nome do arquivo *tar* que será criado.

O backup em dispositivos de acesso sequencial utilizando o *tar* é feito respeitando as características próprias desse tipo de mídia. Um aspecto que deve ser observado com atenção é a propriedade de rebobinagem, que divide esses dispositivos em duas categorias: os rebobináveis e os não rebobináveis. Os dispositivos rebobináveis voltam ao início da fita após cada operação de gravação, enquanto os não rebobináveis permanecem na posição em que a última operação terminou. Portanto, se dois comandos *tar* de gravação forem executados consecutivamente, no caso de uma fita não rebobinável, o resultado será a gravação de dois backups. Já no caso de uma fita rebobinável, o segundo backup será gravado sobre o primeiro, inutilizando-o.



Em geral, na convenção utilizada pelo Linux, os dispositivos rebobináveis têm nomes de dispositivos que começam pela letra “r”, como: /dev/rft0, e os nomes dos não rebobináveis começam por pela letra “n”, como: /dev/nrft0. Existem casos como o formato “SCSI Type”, que tem o padrão /dev/st.

Dessa forma, teríamos:

- Floppy Tape:
 - ▣ /dev/rft0 – rebobinável
 - ▣ /dev/nrft0 – não rebobinável
- SCSI Tape:
 - ▣ /dev/st0 – rebobinável
 - ▣ /dev/nst0 – não rebobinável

Backup em fita:

- Dispositivo de acesso sequencial:
 - ▣ Rebobinável: armazena apenas um backup.
 - ▣ # tar -czf /dev/rft0 /home
 - ▣ Não rebobinável: armazena uma ou mais sequências de backups.
 - ▣ # tar -czf /dev/nrst0 /home
 - ▣ # tar -czf /dev/rst0 /var/spool/mail

Por exemplo, o comando a seguir grava, em uma fita rebobinável em modo compactado, todo o conteúdo do diretório home.

```
# tar -czf /dev/rft0 /home
```

Já o comando a seguir grava em uma fita não rebobinável, e em modo compactado, todos os arquivos do diretório home.

```
# tar -czf /dev/nrft0 /home
```

Comando cpio

- Realiza backup de arquivos e diretórios.
- Pode ser utilizado para ler arquivos no formato *tar*.
- Pode acessar dispositivos de armazenamento remotos.

O comando *cpio* é semelhante ao *tar* e pode ser usado para copiar arquivos individuais ou uma árvore de diretórios completa. O *cpio* suporta diversos formatos, inclusive aquele utilizado pelo *tar*, e também pode acessar dispositivos remotos, por exemplo, uma unidade de fita magnética conectada a outro servidor.

Comando dump

- Além de manipular arquivos e diretórios, pode manipular diretamente os blocos do disco.
- Permite que o backup ocupe diversos volumes.
- Não possui quaisquer restrições quanto a tamanho e pathnames dos arquivos.



Saiba mais

Para saber quais dispositivos são rebobináveis e quais não são, é preciso ler a documentação do sistema.



Saiba mais

Para mais informações a respeito do comando *cpio*, sua página de manual pode ser consultada.



Saiba mais

Para mais informações a respeito do comando *dump*, sua página de manual pode ser consultada.

O comando *dump* não possui qualquer restrição quanto ao tamanho dos arquivos ou de seus nomes completos (pathnames). Enquanto os comandos *tar* e *cpio* manipulam arquivos e diretórios, o *dump* manipula diretamente os blocos de disco, podendo, assim, processar qualquer tipo de arquivo. Em adição, o *dump* permite que o arquivo de backup ocupe múltiplos volumes. O comando *dump* suporta o mecanismo de backup multinível incremental, que permite armazenar apenas os arquivos modificados desde o último backup de nível inferior. A técnica multinível incremental reduz o consumo de recursos de armazenamento e o tempo de realização do backup. O funcionamento do backup incremental provido pelo comando *dump* baseia-se no conceito de níveis de backup, que variam de 0 a 9. Um backup de nível 0 é sempre um backup completo. Qualquer outro nível indica um backup parcial, em que são copiados somente os arquivos que foram modificados depois do último backup realizado com nível inferior ao que está sendo executado. Vale notar que a referência para comparação é o último backup de nível inferior, e não simplesmente o backup mais recente.

A restauração de backups feitos com o comando *dump* é realizada através do comando *restore*, que também restaura de forma incremental os conjuntos de mídias resultantes dos backups. Para recuperar arquivos individuais dos usuários, é recomendável que o administrador o faça no diretório `/tmp` e só depois o copie para a área do usuário o conteúdo necessário. Isso evita uma série de problemas que podem surgir, como a substituição de arquivos do usuário pelos arquivos restaurados no backup.

Comando rsync

- Utilizado para fazer backup entre dois computadores.
- Usa o protocolo SSH para realizar a transferência dos dados.
- Pode ser utilizado por qualquer usuário.

O comando *rsync* é usado para copiar arquivos entre dois computadores através de uma rede de forma segura. Para isso, é utilizado o protocolo SSH para realizar a transferência dos dados. O *rsync* funciona de maneira a acelerar a transferência de dados, copiando somente as diferenças entre os conteúdos remoto e local. A cópia completa dos dados só precisa ser feita na primeira vez que os conteúdos são sincronizados. No exemplo a seguir, o comando *rsync* copia o conteúdo do diretório `/home` do computador local para o diretório `/tmp/backup` do computador remoto.

```
# rsync -auzv /home/usuario usuario@192.168.254.20:/tmp/backup
```

De forma semelhante, o *rsync* também é utilizado para copiar arquivos de um computador remoto para o computador local.

```
# rsync -auzv usuario@192.168.254.20:/tmp/backup /backup
```

Comandos gzip/gunzip e bzip2/bunzip2

Compactação de dados:

- São usados algoritmos para comprimir os dados de forma a ocupar menos espaço em disco.
- É uma das técnicas utilizadas em backups.
- Comandos *gzip* e *bzip2*:
 - Usados para compactar arquivos.
- Comandos *gunzip* e *bunzip2*:
 - Utilizados para descompactar arquivos.

Saiba mais

Para mais informações a respeito do comando *rsync*, sua página de manual pode ser consultada.

Os backups, em geral, utilizam a técnica de compactação de dados, visando o melhor aproveitamento da mídia de armazenamento. Os comandos mais utilizados para essa função são o *gzip*, e o *bzip2*, que após a compactação, colocam os sufixos *.gz* e *.bz2*, respectivamente, no nome do arquivo compactado. O comando *bzip2* possui um algoritmo de compressão mais eficiente que o algoritmo usado no comando *gzip*, mas por outro lado é mais lento quando comparado ao *gzip*. O exemplo a seguir mostra como compactar um arquivo, utilizando o *gzip*.

```
# gzip /home/maria/backup.tar
```

Para descompactar arquivos, podemos usar os comandos *gunzip* e *bunzip2*. O comando do próximo exemplo descompacta o arquivo */home/users/maria/relat_prospec.pdf.gz*.

```
# gunzip /home/users/maria/relat_prospec.pdf.gz
```

Softwares de backup

Os softwares de backup têm como principais funções automatizar, gerenciar e centralizar o backup de uma organização. Nesta sessão vamos apresentar em detalhes os softwares Amanda e Bacula, que são distribuídos gratuitamente. Também serão apresentadas algumas características dos principais softwares comerciais disponíveis no mercado.

Pacotes gratuitos

- Amanda.
- Bacula.



Existem vários softwares gratuitos de backup, oferecendo diversas funcionalidades em comum, porém, cada um com suas características específicas. Esses softwares geralmente são compatíveis com as mais diversas plataformas de hardware e Sistemas Operacionais. Dos softwares gratuitos de backup podemos destacar o Amanda e o Bacula, que serão descritos a seguir:

Amanda

O Amanda (Advanced Maryland Automatic Network Disk Archiver) é um programa que utiliza arquitetura cliente/servidor para realizar de modo centralizado o backup dos servidores de uma organização. Cada computador que fizer parte da rotina de backup necessita da versão cliente do Amanda instalada. O Amanda possui alguns pré-requisitos para funcionar adequadamente. Os seguintes softwares devem estar instalados no servidor: samba (somente se algum computador cliente utilizar o Sistema Operacional Windows); Perl; GNU tar; e readline (utilizado pelo programa amrecover). Após instalar as dependências necessárias, o pacote do Amanda deve ser instalado. As principais distribuições já possuem um pacote do Amanda, que pode ser instalado diretamente sem necessidade de compilar seu código-fonte.

Bacula

O Bacula é um programa que usa a arquitetura cliente/servidor e permite ao administrador gerenciar de modo centralizado o backup dos computadores de uma organização. Seu design modular o torna escalável, sendo capaz de operar em grandes redes com centenas de computadores clientes. Entre suas principais características, podemos destacar:

- Suporte a diversos tipos de mídia, como unidades de fita e discos;
- Suporte a diversos tipos de Sistemas Operacionais como: Linux, *BSD, Windows, MAC etc.;
- Armazena todas as informações em uma base de dados (MySQL, Postgresql etc.).

É composto por cinco módulos:

- **Director:** é o módulo principal e supervisiona todas as atividades de backup e restore, além de coordenar os outros módulos. O Bacula Director é executado como um daemon e, por padrão, utiliza a porta TCP 9101 para receber conexões;
- **Console:** permite ao administrador interagir com o Director através de uma interface que pode ser textual ou gráfica. É utilizado para realizar backups e restaurar dados, além de outras funcionalidades;
- **File:** é o cliente do Bacula e deve ser instalado em todos os computadores clientes que fizerem parte da rotina de backup. O File é executado como um daemon, e por padrão utiliza a porta TCP 9102 para receber conexões;
- **Storage:** é responsável por armazenar e recuperar dados das mídias de backup. O Storage é executado como um daemon e por padrão usa a porta TCP 9103 para receber conexões;
- **Monitor:** esse módulo é responsável por gerenciar os módulos Director, File e Storage, e é disponibilizado somente através de uma interface gráfica.

Pacotes comerciais

Comerciais:

- Arkeia Network Backup.
- ARCserve.
- Backup Exec.
- NetBackup.



Existem vários softwares comerciais de backup, oferecendo diversas funcionalidades em comum, porém, cada um com suas características específicas. Esses softwares geralmente são compatíveis com as mais diversas plataformas de hardware e Sistemas Operacionais, e seus preços podem chegar a alguns milhares de reais. Dos softwares comerciais de backup, podemos destacar os seguintes:

- **Arkeia Network Backup:** é um software de backup que foi desenvolvido em 1998 pela empresa Arkeia Software. Além do software, a empresa também oferece como solução um appliance e um virtual appliance que é executado no VMware. Entre suas principais características, podemos destacar: console de administração web, que gerencia todas as atividades de backup e restore; suporte aos mais diversos tipos de mídias e plataformas de hardware e software; suporte a backups físicos, virtuais e em nuvem; criptografia de dados etc.;
- **ARCserve:** software de backup que foi desenvolvido em 1990 pela empresa Cheyenne Software. Em 1996, a empresa CA Technologies adquiriu a Cheyenne Software e passou a manter e aprimorar o ARCserve. Entre suas principais características, podemos destacar: console de administração gráfico, que gerencia todas as atividades de backup e restore; suporte aos mais diversos tipos de mídias e plataformas de hardware e software; suporte a backups físicos, virtuais e em nuvem; integração com aplicativos como MS Exchange e SQL Server; criptografia de dados; recuperação total de servidores, mesmo migrando para hardwares diferentes etc.;

- **Backup Exec:** um software de backup que foi desenvolvido em 1982 pela empresa Maynard Electronics. Foi comprado por diversas empresas e passou por várias melhorias até ser comprado pela Symantec em 2005. O nome Backup Exec só foi adotado em 1998, quando ainda pertencia à empresa Seagate. Entre as principais características do Backup Exec, podemos destacar: console de administração gráfico, que gerencia todas as atividades de backup e restore; suporte aos mais diversos tipos de mídias e plataformas de hardware e software; backup de máquinas virtuais; integração com aplicativos como MS Exchange e SGBDs (Oracle, SQL Server etc.); criptografia de dados etc.;
- **NetBackup:** É um software de backup que foi desenvolvido em 1987 pela empresa Control Data Corporation. Passou por diversos aprimoramentos e foi comprado por outras empresas, entre elas a Veritas, quando ficou conhecido com Veritas NetBackup. Atualmente pertence à Symantec, que o adquiriu em 2005. Entre as principais características do NetBackup, podemos destacar: console de administração gráfico, que gerencia todas as atividades de backup e restore; criptografia de dados; controle de acesso; suporte aos mais diversos tipos de mídias e plataformas de hardware e software; backup de máquinas virtuais; relatórios diversos etc.

Políticas de backup

Fatores para definição da política de backup:

- Criticidade, volatilidade e volume dos dados armazenados no sistema.
- Tipo de uso e horário de operação.
- Capacidade e velocidade do dispositivo utilizado para realizar o backup.

Empresa com ambiente de Missão Crítica:

- Deve estabelecer diversas políticas de backup.

Para cada sistema, é necessário decidir quando e como fazer o backup dos dados, o que depende dos seguintes fatores:

- Criticidade, volatilidade e volume dos dados armazenados no sistema;
- Tipo de uso e horário de operação do sistema;
- Capacidade e velocidade do dispositivo utilizado para armazenar os dados do backup etc.

Com base nesses fatores, cada organização deve definir uma política de backup apropriada às suas necessidades operacionais. Uma empresa que opere um sistema de missão crítica vai ter políticas de backup muito diferentes das políticas de, por exemplo, uma instituição educacional.

Eventualmente, diferentes departamentos de uma mesma empresa podem ter necessidades diversas, e, por conseguinte, diferentes políticas de backup. Geralmente, quanto mais crítica for a operação do sistema para o negócio da empresa e quanto maior a volatilidade dos dados, maior a frequência e abrangência das operações de backup.

Essa norma vale tanto para a periodicidade com que os backups são executados, quanto para a escolha dos dados que devem ser incluídos no backup. Como regra geral, temos que, em sistemas nos quais há grande atividade por parte dos usuários, os diretórios de trabalho devem ser salvos diariamente. Já sistemas de arquivos que são modificados com menor frequência (o diretório `/usr`, por exemplo) podem ser salvos apenas uma vez por mês. Quando houver necessidade de operação ininterrupta do sistema, a operação de backup deve ser planejada de modo a se adequar ao regime e horário de uso do sistema e interferir o mínimo possível na rotina dos usuários.



É razoável supor que o tipo de dispositivo a ser utilizado nos backups deva ser selecionado em função das condições anteriores, embora isso nem sempre seja possível, devido a limitações orçamentárias. O administrador de sistemas terá de dispor de todo o seu conhecimento para fazer o melhor uso possível dos equipamentos disponíveis, sem colocar em risco a segurança das operações da empresa. Aqui vale lembrar que, independentemente do procedimento normal de backup, administradores de sistemas experientes adquirem o bom hábito de realizar backup regular de dados e programas que serão manipulados em operações de reconfiguração do Sistema Operacional, de reorganização de diretórios e partições ou de atualização de software.

Isso porque sempre existe a possibilidade de se cometer erros, que podem levar à perda de informações importantes ou muito trabalhosas para serem recuperadas. A administração da rotina de backup é muito facilitada se for centralizada em um único servidor e executada por um software profissional de backup, seja ele gratuito ou não.

O processo de identificação das mídias demonstra ser de extrema importância em uma política de backup que prime pela eficiência. As fitas devem ser identificadas, isto é, possuir labels que indiquem unicamente o seu conteúdo. Informações como a lista dos sistemas de arquivos copiados, o nível do backup executado e a data da sua execução devem ser escritas na própria caixa que abriga a mídia. Tal procedimento facilitará bastante o processo de recuperação de dados a partir dos backups realizados, o que ajuda bastante em situações críticas. O administrador deve tentar utilizar uma única mídia para realizar os backups diários. Com isso, ele pode programar o backup para ser executado em um horário de pouca utilização do sistema, minimizando os problemas de inconsistência de dados e queda de desempenho nas aplicações.

Se o backup não couber em uma única mídia, recomenda-se comprar uma unidade de maior capacidade, usar múltiplos dispositivos ou adquirir um robô. Para minimizar a possibilidade de haver um sistema de arquivos que não possa ser copiado para uma única mídia, os administradores devem levar em conta a capacidade máxima da sua mídia de backup quando forem definir os tamanhos das partições de seus sistemas.

Cuidados no armazenamento das mídias

Mídia de backup:

- Centralizar todos os backups em um só servidor.
- Rotular as fitas.
- Definir o conteúdo e a periodicidade de armazenamento para cada computador.
- Fazer os sistemas de arquivos menores que o seu dispositivo de backup (dumps diários).
- Manter algumas fitas fora do local de trabalho.
- Executar o backup em horários de pouca utilização dos sistemas.
- Acondicionar as mídias adequadamente.
- Verificar, periodicamente, a boa condição de gravação das mídias de backup.

Tão importante quanto definir e cumprir à risca os procedimentos de backup é acondicionar adequadamente a mídia utilizada nas operações, independentemente do seu tipo. Cada um desses meios de armazenamento possui características próprias de confiabilidade, capacidade e requisitos de manuseio e armazenamento, que devem ser cuidadosamente observadas para que se obtenha o desempenho esperado. É uma boa prática manter o registro



das condições ambientais (temperatura e umidade) do local de armazenamento. A necessidade de armazenamento de mídias fora do local de trabalho vai depender da criticidade dos dados para a empresa, bem como dos riscos associados às suas instalações.

Uma refinaria de petróleo apresenta riscos consideravelmente altos em relação ao campus de uma universidade. Em qualquer dos casos, é recomendável o armazenamento de um backup de nível 0 em algum local fora da empresa, como uma garantia contra desastres naturais. Uma empresa que possua filiais, por exemplo, pode adotar a prática de, mensalmente, trocar fitas de backup entre elas. Atenção também deve ser dada à segurança do local de armazenamento da mídia de backup, visando minimizar riscos de perda dos arquivos devido a acidentes naturais ou ações criminosas. Empresas e instituições que trabalham com informações sensíveis ou confidenciais devem possuir locais seguros para armazenamento das mídias utilizadas para backup de seus sistemas.

Finalmente, a boa condição de gravação da mídia de backup precisa ser verificada periodicamente. Para isso, uma solução eficaz consiste na execução de operações de leitura e recuperação a partir de uma amostra aleatória das fitas armazenadas. Outro exercício recomendado consiste em explorar alguns cenários que exijam a restauração dos dados.

O cenário de pior caso, com a perda total dos equipamentos, precisa ser estudado. O administrador, nesse caso, deve determinar quanta informação seria perdida e quanto tempo levaria para tornar o sistema novamente operacional, incluindo o tempo de aquisição de um novo hardware. É necessário, então, fazer uma avaliação das respostas obtidas para determinar se elas atendem às necessidades da empresa. Caso contrário, a política de backup terá de ser revista. Para sistemas de pequeno porte, as ferramentas de backup disponíveis no Linux são satisfatórias. No entanto, para sistemas de médio e grande porte, deve-se investir em ferramentas profissionais mais robustas e seguras.

6

Serviço de impressão

objetivos

Conhecer as principais características associadas à manutenção de um serviço de impressão em sistemas Linux; Entender como se dá o processo de impressão em sistemas Linux; Compreender a diferença entre os softwares gerenciadores de impressão; Saber qual sistema de impressão melhor atende às necessidades de uma organização; Conhecer os comandos necessários para a configuração e manutenção de um serviço de impressão baseado no CUPS.

Serviço de impressão; Arquitetura do sistema de impressão; Softwares gerenciadores de impressão; LPD; LPRng; CUPS.

conceitos

Introdução

- O serviço de impressão de arquivos é, certamente, um dos que mais exige atenção e demanda tempo do administrador de sistemas.
- O serviço de impressão do Linux começou a se mostrar mais flexível e fácil de ser administrado a partir do lançamento do software de gerenciamento de impressão CUPS.

O serviço de impressão de arquivos é, certamente, um dos que mais exige atenção e demanda tempo do administrador de sistemas. Alguns profissionais de TI costumam argumentar que instalar, configurar e manter um serviço de impressão no Linux é muito mais complexo do que em sistemas Windows. Até pouco tempo atrás, eles não deixavam de ter razão. Instalar uma impressora costumava ser mais complicado no Linux do que no Windows. Outro agravante era o fato de que fabricantes de impressoras costumavam desenvolver os drivers dos seus equipamentos somente para sistemas Windows, por ser esse um sistema de maior penetração no mercado. Entretanto, o serviço de impressão do Linux começou a se mostrar mais flexível e fácil de ser administrado a partir do lançamento do software de gerenciamento de impressão CUPS.

Histórico

- O sistema de impressão, no início, baseava-se em dois sistemas:
 - Line Printer Daemon (LPD), do BSD.
 - Line Printer System, do System V.
- Instalação e configuração eram tarefas complexas, até algum tempo atrás.

- Os sistemas de impressão não estavam atendendo às novas demandas.
- Inicia-se a busca por uma interface padrão para impressão:
 - ▣ POSIX do IEEE.
 - ▣ IPP do IETF.



Nos primórdios dos sistemas Unix, o serviço de impressão era baseado em dois sistemas projetados na década de 1970, para impressão de arquivos texto em impressoras matriciais:

- Line Printer Daemon (LPD), do BSD;
- Line Printer System, do System V.

Com o surgimento de novas demandas, entretanto, esse cenário começou a mudar. Desenvolvedores de sistemas Linux passaram a promover mudanças nos dois sistemas mencionados, para que eles pudessem atender às demandas por impressões de arquivos nos mais diferentes formatos e em dispositivos cada vez mais elaborados, como as impressoras a laser. Com o passar do tempo, foram surgindo variantes do LPD, com destaque para o PLP (Portable Line Printer spooler system), que mais tarde teve seu nome alterado para LPRng. Esses sistemas, contudo, não conseguiram explorar todo o potencial dos novos dispositivos de impressão que eram lançados no mercado. Foram feitas, então, muitas tentativas de se desenvolver uma interface padrão para impressão. Dentre elas, pode-se destacar o POSIX e o IPP Printing Protocol.

POSIX

Foi desenvolvido pelo Institute of Electrical and Electronics Engineers (IEEE) e definia um conjunto comum de ferramentas de linha de comando, assim como uma interface em C para administração de impressoras e gerenciamento de impressão. Todavia, foi descontinuado pelo próprio IEEE.

IPP

Foi desenvolvido pelo Internet Engineering Task Force (IETF), em 1999, e define extensões ao protocolo HTTP para prover suporte ao serviço de impressão. Diferentemente do POSIX, o IPP desfruta de uma ampla aceitação por parte da indústria e está cotado para tornar-se a solução padrão de impressão em rede para todos os Sistemas Operacionais. Nesse cenário, surgiu o CUPS, um moderno sistema de impressão para sistemas Unix, que pode ser estendido para suportar novas impressoras, dispositivos e protocolos, ao mesmo tempo em que mantém a compatibilidade com as aplicações existentes.

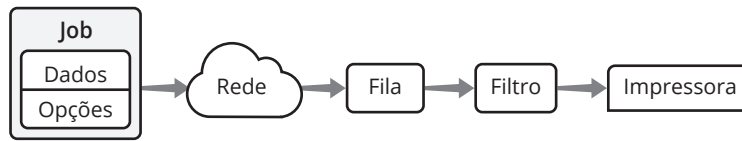
Arquitetura do sistema de impressão

- O usuário solicita uma impressão.
- O sistema cria um job.
- O job é enviado por meio da rede.
- O sistema de impressão trata o job.
- Enfileira se a impressora está ocupada.
- Filtros são aplicados ao job.
- Aplicação das opções selecionadas.
- Tradução dos dados para a linguagem nativa da impressora.
- A impressão é realizada.



Para manter o serviço de impressão operando de forma correta, é importante entender como ele funciona. A princípio, todos os sistemas de impressão funcionam da mesma forma e sua arquitetura é ilustrada na figura 6.1.

Figura 6.1
Arquitetura dos
sistemas de
impressão.



O esquema de impressão de um arquivo, geralmente, funciona da seguinte maneira:

1. O usuário solicita a impressão de um arquivo e passa para o sistema algumas opções (qualidade da impressão, número de cópias etc.). Geralmente, os dados encontram-se no formato PostScript, embora isso nem sempre seja verdade. Nesse momento, o sistema de impressão cria o que se chama de job.
2. O sistema envia, então, o arquivo e as opções por intermédio da rede, para o servidor de impressão. Se a impressora estiver localmente conectada ao sistema, essa ação não é executada.
3. Ao chegar ao servidor, o sistema de impressão local enfileira o job até que a impressora esteja disponível.
4. O sistema aplica as opções selecionadas pelo usuário e traduz os dados do arquivo para a linguagem nativa da impressora que, geralmente, não é PostScript. A esse passo é dado o nome de filtragem. Os filtros permitem que os usuários e as aplicações imprimam os mais diferentes tipos de arquivo.
5. Por fim, a impressão é feita. Nesse ponto, o sistema faz a remoção de alguns arquivos temporários. Caso haja algum erro nesse caminho, o usuário será notificado pelo sistema.

Softwares gerenciadores de impressão

- Softwares gerenciadores de impressão:
 - ▣ LPD.
 - ▣ LPRng.
 - ▣ CUPS.
- Diferentes softwares surgem na tentativa de contornar as limitações dos antigos sistemas de impressão.
- A seleção do melhor sistema depende de características do ambiente.

Existem, atualmente, alguns sistemas de impressão desenvolvidos para substituir os antigos sistemas baseados no LPD, como por exemplo, o LPRng. Entretanto, o foco desta sessão de aprendizagem será o CUPS, que além de ter se tornado um padrão, é a melhor opção de sistema de impressão disponível no mercado para sistemas Unix.



LPD (Line Printer Daemon)

- Foi o software padrão de impressão do BSD durante anos.
- Atendia bem a demandas de impressão de arquivos texto.
- Mostrou-se inapropriado para impressões mais sofisticadas (gráficos, imagens etc.).
- O uso do GNUlpr é recomendado caso o administrador opte por utilizar um sistema de impressão baseado no LPD.
- Não deve ser confundido com protocolo de impressão de rede do IETF (RFC 1179).



O LPD foi o sistema padrão de impressão do Unix BSD durante vários anos. Quando ele surgiu, as demandas por impressão eram de arquivos puramente textuais. Com o passar dos anos, mostrou-se inapropriado para atender às novas demandas por impressão de arquivos mais elaborados, como imagens, gráficos e até mesmo arquivos provenientes de editores de texto mais sofisticados. Apesar de suas limitações, o LPD conseguiu disseminar-se por uma vasta gama de sistemas Unix e, ainda hoje, tem grande utilidade como um sistema básico de impressão.

Entretanto, para usar plenamente o potencial das impressoras modernas, é necessário o uso de filtros e outros programas complementares. Esses programas existem e, apesar de sua complexidade, são uma alternativa para quem opta por utilizar o LPD. São programas que fazem o front-end da aplicação. Os mais conhecidos são o KDEPrint, o GPR e o XPP. Há um número considerável de versões do LPD, sendo o GNUlpr uma das versões que apresenta menos modificações com relação à versão original. Caso o administrador opte por utilizar um sistema baseado no LPD, é recomendável que ele use o GNUlpr.

Componentes do LPD

Componentes do LPD:

- Pode se referir ao daemon ou ao conjunto de programas que o compõem.
- Componentes:
 - ▣ lpd: daemon propriamente dito.
 - ▣ lpr: comando de impressão.
 - ▣ lpq: comando para listar os jobs que estão na fila de impressão.
 - ▣ lprm: comando para remover um job.
 - ▣ lpc: comando de controle do daemon lpd.



O LPD se refere tanto ao daemon quanto a uma vasta coleção de programas que compõem o sistema de impressão. Seus principais componentes são:

- **lpd**: daemon de impressão propriamente dito. Controla tudo o que diz respeito à impressão de um job. Para cada arquivo em fase de impressão, um novo processo é iniciado;
- **lpr**: trata-se do comando de impressão do usuário. Ao receber um pedido de impressão, o lpr contacta o lpd, criando um novo job de impressão no sistema;
- **lpq**: comando utilizado para listar os jobs que estão na fila de impressão;
- **lprm**: comando utilizado para remover um job da fila de impressão;
- **lpc**: trata-se do comando de controle do daemon lpd. Por meio desse comando, o administrador do sistema pode parar, iniciar, reordenar as filas de impressão e tomar outras medidas administrativas necessárias.

A forma como esses programas se relacionam, conforme o esquema apresentado na figura 6.1, é a seguinte:

1. No momento da inicialização do sistema, o daemon `lpd` é iniciado. Ele fica, então, aguardando por conexões e gerencia as filas das impressoras.
2. Quando o usuário solicita a impressão de algum arquivo utilizando o comando `lpr`, este contacta o `lpd` através da rede e submete os dados do usuário, que contém o arquivo a ser impresso e as opções selecionadas pelo usuário.
3. Quando a impressora se torna disponível, o `lpd` cria um processo `lpd` filho para realizar a impressão do job.
4. O processo `lpd` filho executa o(s) filtro(s) apropriados para o job e envia os dados resultantes à impressora.

Como já é sabido, o LPD foi projetado para ser utilizado em um tempo em que as impressoras eram do tipo matricial, e as demandas de impressão eram por arquivos texto. Pelo esquema mostrado anteriormente, vê-se que as necessidades mais complexas de impressão podem ser atendidas mediante a aplicação de filtros. Para ajudar nesse processo de filtragem, existem diversos programas, como o `gs`, `ppdfilt`, `ps2ps`, `mpage` e `a2ps`. A maioria desses programas faz conversões entre diferentes formatos de arquivos. Entretanto, como se verá adiante, existem alternativas mais eficientes e menos trabalhosas.

Protocolo LPD

A RFC 1179 define o protocolo Line Printer Daemon Protocol (LPD), criado pelo IETF, que é usado pelo servidor LPD e por todos os servidores de impressão de rede usados em sistemas Unix. Esse protocolo é usado na comunicação entre o servidor e os diversos componentes, como comandos auxiliares e dispositivos. Apesar do protocolo e do servidor de impressão possuírem o mesmo nome, eles não devem ser confundidos.

LPRng

- Implementação melhorada do LPD.
- Seu uso é recomendado quando o sistema possui muitas impressoras, impressoras seriais ou impressoras que não possuem suporte ao protocolo LPD.
- Apresenta facilidades de contabilidade e maior preocupação com a segurança.
- Utiliza o mesmo modelo de filtros do LPD.



O LPRng é uma implementação melhorada do LPD, que apresenta preocupação maior com aspectos de segurança, além de outras funcionalidades. O LPRng apresenta facilidades para a contabilidade de uso do ambiente de impressão. Essa funcionalidade é especialmente útil quando se deseja controlar o uso dos recursos de impressão. O LPRng usa o mesmo modelo de filtros do LPD, o que pode ser visto como uma facilidade adicional. Ele torna mais simples a administração de ambientes mais complexos. O uso do LPRng é recomendado se o sistema possui mais de uma impressora, impressoras seriais ou qualquer impressora de rede que não possua suporte ao protocolo LPD. Sua maior segurança advém do fato de que nenhum dos seus programas tem o bit SUID estabelecido, além de suporte à autenticação, utilizando o programa PGP ou o protocolo Kerberos, que trabalham com criptografia de dados.

CUPS

- Surge como alternativa às diversas implementações do protocolo LPD.
- Permite a configuração automática de clientes a partir da configuração de um único servidor.
- Usa o conceito de classes.
- Faz balanceamento de carga e é tolerante a falhas.
- Foi projetado em torno de um processo central de escalonamento.
- Vantagens da implementação do IPP:
 - Escalonador baseado em servidor HTTP 1.1.
 - Interface web para administração e configuração.
 - Maior número de opções de impressão.
 - Controle de acesso bastante apurado.
 - Suporte à criptografia e ao serviço de proxy.
 - Facilidade na configuração para clientes Windows.

Já há algum tempo o CUPS se tornou o sistema de impressão padrão da maioria das distribuições do Linux. Parte do seu sucesso se deve ao fato de ser uma implementação do Internet Printing Protocol (IPP), um novo padrão para impressão baseado no protocolo HTTP, que apresenta uma série de vantagens em relação ao velho protocolo LPD. Dentre elas, pode-se destacar.

- Possui escalonador baseado em um servidor web HTTP 1.1;
- Possui interface web para configuração e administração do sistema;
- Disponibiliza maior número de opções de impressão para o usuário;
- Pode ser feito um controle de acesso bastante apurado a uma impressora, baseado em diretivas Allow e Deny, comuns ao servidor Web Apache;
- Oferece suporte à criptografia e ao serviço de proxy.

Antes do lançamento do CUPS, muitos administradores de sistemas ressentiam-se da dificuldade de se estabelecer um sistema de impressão em rede no Linux que funcionasse de forma simples e confiável. Havia algumas razões para esse problema.

- As extensões feitas por muitos fabricantes ao protocolo LPD, o que tornava muito difícil a impressão entre diferentes plataformas, quando não impossível;
- O fato de o LPD não possuir mecanismo de configuração automática de impressoras.

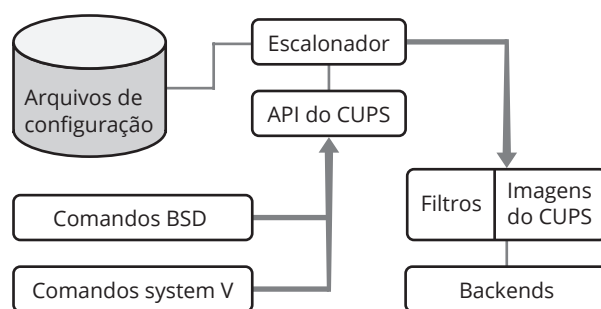
Arquitetura

- Escalonador.
- API do CUPS.
- Comandos BSD/System V.
- Filtros e imagens do CUPS.
- Backends.



O CUPS suporta balanceamento de carga e tolerância a falhas, com a utilização de múltiplos servidores para gerenciar o ambiente de impressão. Será visto mais à frente que é possível reunir múltiplas impressoras de rede em uma única classe, de forma a permitir que o usuário possa direcionar o seu job para uma classe e tê-lo impresso na primeira impressora disponível da classe. Como muitos sistemas de impressão, o CUPS é projetado em volta de um processo central de escalonamento, que gerencia os jobs de impressão, processa comandos administrativos, provê informações de status das impressoras para programas locais e remotos, e os repassa aos usuários quando necessário. A figura 6.2 apresenta a arquitetura de funcionamento do CUPS.

Figura 6.2
Arquitetura
do CUPS.



Escalonador

- O escalonador é um servidor web que gerencia jobs e impressoras.
- Usa filtros e backends para imprimir os jobs.
- Também pode ser utilizado como fonte de documentação.



O escalonador não gerencia apenas a impressão de Jobs. Por meio dele, é possível gerenciar impressoras, jobs e classes do sistema. Trata-se de um servidor HTTP que, além de cuidar das requisições de impressão feitas por intermédio do protocolo IPP, atua no monitoramento e na administração do serviço de impressão. O escalonador também utiliza filtros e backends apropriados para imprimir os jobs dos usuários. Além disso, pode ser usado como ferramenta de consulta, já que possui uma farta documentação sobre as funcionalidades do sistema.

API do CUPS

A API do CUPS contém uma série de funções específicas para enfileiramento de jobs de impressão. Para tal, ela obtém informações das impressoras do sistema, acessa recursos utilizando os protocolos HTTP e IPP e manipula arquivos PostScript Printer Description (PPD). O CUPS possui quatro tipos de arquivos de configuração, que serão apresentados na tabela 6.1.

Tabela 6.1
Arquivos de
configuração
do CUPS.

Arquivos de configuração	
Arquivos de configuração do servidor HTTP	Definem as propriedades de controle de acesso ao servidor. Possuem sintaxe semelhante à dos arquivos de configuração do servidor web Apache.
Arquivos de definição de impressoras e de classes	Contém informações sobre as impressoras e as classes definidas no sistema. Classes de impressoras nada mais são do que um conjunto de impressoras.
Arquivos de tipos MIME e de regras de conversão	Listam os filtros que, na impressão, permitem que uma aplicação envie o formato de arquivo adequado ao sistema de impressão, que então converte o documento para um formato que pode ser impresso pelo dispositivo indicado.
Arquivos PPD	Descrevem as características de todas as impressoras do sistema. Há um arquivo PPD para cada impressora do sistema.

Comandos BSD/System V

Funcionamento do CUPS:

- API contém funções específicas para enfileiramento dos jobs.
- Jobs são filtrados antes de serem impressos.

Arquivos de configuração:

- Arquivos de configuração do servidor HTTP.
- Arquivos de definição de impressoras e de classes.
- Arquivos de tipos MIME e de regras de conversão.
- Arquivos PostScript Printer Description (PPD).

O CUPS pode utilizar tanto os comandos do LPD quanto os comandos do Line Printer System do System V, para imprimir um arquivo ou verificar o status de uma impressora, mantendo compatibilidade com esses sistemas.

Filtros e imagens do CUPS

Jobs de impressão são filtrados antes de serem enviados a uma impressora. Alguns filtros fazem a conversão dos dados do job para um formato compatível com o da impressora. Outros filtros fazem apenas a seleção de páginas e tarefas relacionadas à ordenação do arquivo. Todos os filtros, entretanto, devem suportar um conjunto comum de opções, incluindo:

- Nome da impressora;
- Identificação do job (job ID);
- Nome do usuário;
- Título do job;
- Número de cópias e opções do job.

Existem filtros para os mais diferentes formatos de arquivos existentes. A biblioteca de imagens do CUPS é especialmente útil na impressão de grandes arquivos de imagens. Nesse caso, ela é utilizada por filtros de arquivos de imagens e pelos drivers mais gerais de impressoras, para fazer conversões e gerenciamento de cores e de escala de grandes arquivos de imagens que devem ser impressos.

Backends

Um backend nada mais é do que um filtro especial que envia dados de impressão para um dispositivo ou conexão de rede. Existem backends para conexões paralelas, seriais, USB, IPP, LPD, SMB, AppSocket (JetDirect) e Netatalk. Os backends também são usados para determinar os dispositivos disponíveis. Na inicialização, cada backend é questionado sobre uma lista de dispositivos que ele suporta e sobre qualquer outra informação que esteja disponível. Isso permite, por exemplo, que um backend informe ao CUPS que uma impressora HP Deskjet 860C está conectada à porta paralela LPT1.

Interface web do CUPS

A interface web do CUPS possibilita o gerenciamento de impressoras, classes e jobs de modo rápido e intuitivo. Esta sessão de aprendizagem será baseada na administração do CUPS através da linha de comandos, ficando a cargo do aluno navegar pela interface web para explorar todas as suas funcionalidades. A figura 6.3 mostra a página inicial da interface web do CUPS na versão 1.4.3. A partir dela é possível navegar entre os diversos links e funcionalidades.



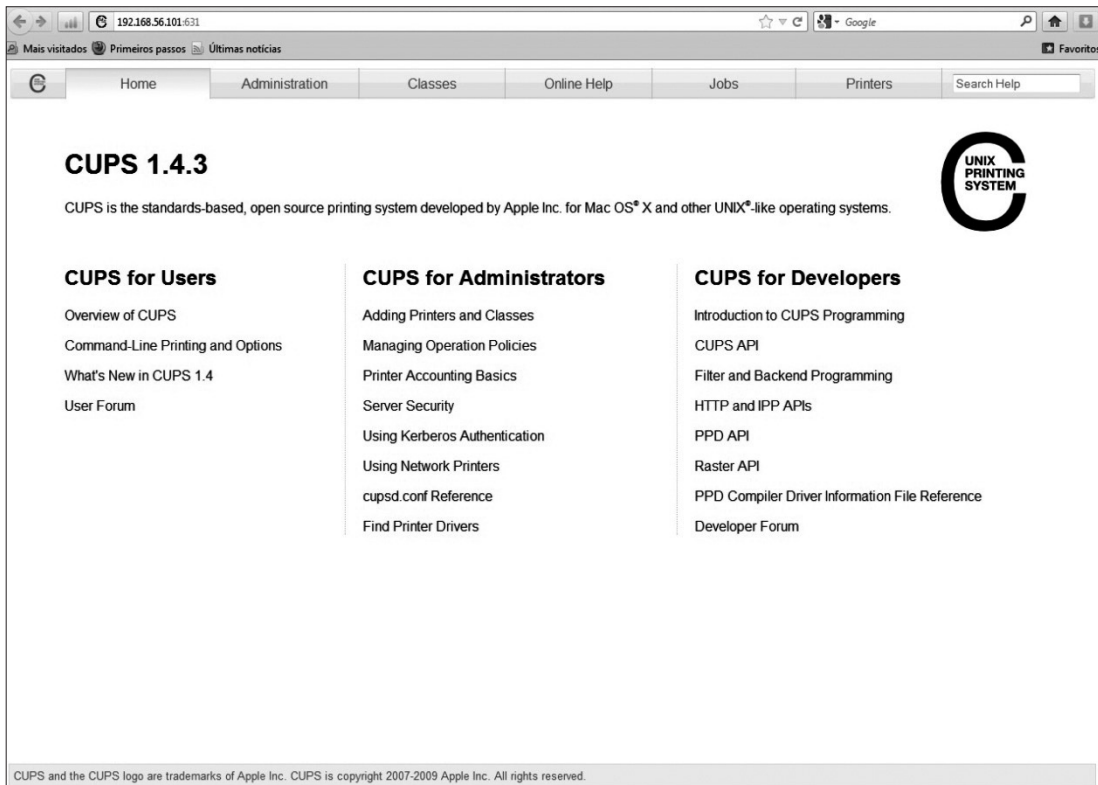


Figura 6.3
Interface web do
CUPS versão 1.4.3.

Gerenciamento de impressoras

Gerenciamento de impressoras:

- Através da linha de comandos.
- Por intermédio da interface web:
 - ▢ <http://localhost:631/admin>
- Classes de impressora:
 - ▢ Impressoras são reunidas em grupos.
 - ▢ Jobs enviados para uma classe são encaminhados para a primeira impressora livre da classe.
 - ▢ Balanceamento via round-robin é adotado.

Quando um usuário solicita que um arquivo seja impresso em uma determinada classe de impressora, o sistema direciona esse arquivo para a primeira impressora que se tornar disponível e que pertença à classe solicitada. É feito um esquema de rodízio, utilizando o escalonamento round-robin para evitar o uso excessivo de apenas um dispositivo. Nessa mesma linha, o CUPS também suporta o conceito de classes implícitas. Com isso, ele permite que se estabeleçam múltiplos servidores com configurações de impressoras idênticas. Os clientes, então, passam a enviar seus jobs de impressão para o primeiro servidor que estiver disponível.

Dessa forma, se um servidor para, os jobs passam a ser redirecionados para aqueles que estão operacionais, criando um sistema tolerante a falhas. O comando utilizado para criar classes de impressoras, inserir e remover impressoras nas classes é o *lpadmin*, que será visto a seguir.

Classes implícitas:

- Múltiplos servidores com configurações idênticas.
- Sistema tolerante a falhas (redundância).
- Criação e remoção de classes e impressoras feitas através do comando *lpadmin*.

Cada fila tem um dispositivo e um nome:

- Uniform Resource Identifier (URI) – Forma de identificação de dispositivo:
 - serial: /dev/ttyS1?baud=115200
 - parallel: /dev/lp1
 - socket: //192.168.1.10



Gerenciamento de clientes

Configuração manual:

- Executar o comando *lpadmin* em cada um dos clientes.
- A operação deve ser repetida para cada dispositivo.
- Em uma grande rede, o trabalho pode se tornar estafante.



A configuração de clientes Unix-like no CUPS pode ser feita de forma manual ou automática, especificando um ou múltiplos servidores, ou ainda por intermédio do repasse de informações de um cliente para outros clientes.

Configuração manual

Das várias formas de se configurar um cliente, a mais trabalhosa, certamente, é a configuração manual. Ela exige que o administrador execute, em cada cliente, o comando *lpadmin*, informando o servidor de impressão e o nome do dispositivo, por meio de seu URI. Esse procedimento deve ser repetido para cada impressora que se queira definir no cliente. Em um ambiente com múltiplas impressoras e um grande número de clientes, esse trabalho pode ser estafante.

Configuração automática

- O CUPS possibilita configuração automática das impressoras nos clientes.
- Facilita a administração em grandes ambientes.
- Todas as impressoras criadas ou removidas do servidor são automaticamente espelhadas nos clientes.



O CUPS também possibilita a configuração automática de filas de impressão em clientes que estão na mesma rede do servidor. Para configurar um cliente, nesse caso, o administrador não precisa fazer absolutamente nada. Todas as impressoras e classes criadas ou removidas do servidor são automaticamente atualizadas no cliente em um período máximo de 30 segundos. Para tal, o administrador terá de habilitar a diretiva *BrowseAddress* no arquivo de configuração do CUPS no servidor (*/etc/cups/cupsd.conf*). São propagados 80 bytes de informação por impressora a cada 30 segundos.

Outra opção é segmentar a rede e, nesse caso, a diretiva *BrowsePoll*, deverá ser incluída no arquivo de configuração do CUPS no cliente (*/etc/cups/cupsd.conf*), para definir o servidor que será consultado pelo cliente. Se for utilizado mais de um servidor e eles estiverem em sub-redes diferentes, é necessário configurar os clientes para que busquem informações nesses servidores, utilizando múltiplas diretivas *BrowsePoll*, uma por linha, para especificar cada servidor. Esse esquema pode se tornar ineficiente se existirem clientes e servidores espalhados em várias sub-redes.



Saiba mais

Apesar dessa quantidade de dados ser pequena, em um ambiente com muitos servidores e dispositivos de impressão, os valores padrão das diretivas *BrowseInterval* e *BrowseTimeout* podem ser alterados para reduzir esse fluxo de dados.

Para contornar essa deficiência, bem como limitar a quantidade de clientes que podem buscar informações em um servidor, o administrador pode configurá-lo para que apenas um cliente faça esse serviço e repasse as informações para os demais. É possível implementar essa limitação inserindo a diretiva BrowseRelay no arquivo de configuração do CUPS no servidor.

Vamos supor um cenário onde exista um servidor em uma sub-rede 1 com o IP 10.1.1.2 e outro servidor em uma sub-rede 2 com o IP 10.1.2.4. Além disso, vamos supor que exista um computador cliente conectado a uma terceira sub-rede, com o endereço IP 10.1.3.101 e também conectado às sub-redes 1 e 2. Para esse cliente fazer o repasse de informações (relay) para os demais clientes da sub-rede 3, seu arquivo de configuração deve estar configurado da seguinte maneira:

```
# Fazendo poll nos dois servidores
BrowsePoll <nome_servidor_sub-rede-1>
BrowsePoll <nome_servidor_sub-rede-2>

# Fazendo relay das sub-redes 1 e 2 para a sub-rede 3
BrowseRelay 10.1.1 10.1.3.255
BrowseRelay 10.1.2 10.1.3.255
```

À diretiva BrowseRelay devem ser passados dois parâmetros. O primeiro especifica um endereço fonte, e o segundo, um endereço de broadcast. Assim, quaisquer pacotes vindos dos servidores de impressão nas sub-redes 1 e 2 serão repassados para o endereço de broadcast da sub-rede 3.

7

Registro de eventos

objetivos

Aprender em detalhes o conceito de registro de eventos em sistemas Linux; Conhecer as principais soluções para gerenciamento de registros de eventos em sistemas Linux; Saber os passos necessários para implementar um servidor dedicado ao armazenamento de registros de eventos de todos os servidores e equipamentos da rede; Ter contato com as principais soluções de análise de registros de eventos; Conhecer as principais recomendações de segurança envolvendo uma solução completa de gerenciamento de registros de eventos no Linux.

Registro de eventos (syslogd; rsyslog; syslog-ng); Rotacionamento de arquivos de log; Servidor de logs; Aplicativos para análise de arquivos de log; Recomendações básicas de segurança.

conceitos

Introdução

Introdução:

- Registro dos eventos que ocorrem em um sistema.
- Diretório padrão para armazenamento dos arquivos de log no Linux: “/var/log”.
- Conteúdo de um registro:
 - ▣ Data e hora em que ocorreu o evento.
 - ▣ Nome do servidor que gerou o evento.
 - ▣ Nome do programa que gerou o evento.
 - ▣ Mensagem gerada pelo programa.



Todo Sistema Operacional gera eventos que, na maioria das vezes, ocorrem de modo assíncrono e em intervalos irregulares. Por exemplo, um servidor de arquivos gera eventos como a finalização de uma escrita em disco, a abertura de um arquivo etc. Muitos desses eventos possuem grau de importância relativamente baixo e, sendo assim, não precisam ser armazenados. Todas as ações executadas pelas aplicações que estão ativas em um sistema Linux podem ser registradas em arquivos, geralmente localizados no diretório “/var/log”. Esses arquivos que armazenam os registros relativos aos eventos gerados pelo Sistema Operacional e pelas aplicações são chamados de arquivos de log. Os arquivos que armazenam os registros e os tipos de registros que são armazenados podem ser definidos pelo administrador.

Alguns programas, como o Apache e o Qmail, trabalham com diversos arquivos de log e, para melhor organização, armazenam esses arquivos em subdiretórios dentro do diretório “/var/log”. Um arquivo de log contém, normalmente, os seguintes registros: data e hora



em que ocorreu o evento, nome do servidor que gerou o evento, nome do programa que gerou o evento e a mensagem emitida pelo programa. Tais informações são de extrema importância para o administrador de sistemas, auxiliando-o não só no monitoramento das atividades executadas pelos programas, como também na solução e prevenção de incidentes de segurança e falhas de software e hardware. Nesta sessão de aprendizagem, será visto como configurar um sistema Linux, para que possa armazenar os diversos registros de eventos gerados da maneira mais adequada possível e de acordo com a estratégia adotada pelo administrador da rede.

Sysklogd

- Provê suporte a registros de eventos do Sistema Operacional e das aplicações.
- Seu arquivo de configuração, `/etc/syslog.conf`, controla os registros que serão armazenados pelo syslogd por meio de regras específicas.
- Ativa o serviço de registro de eventos do sistema.



O pacote `sysklogd` contém os dois daemons responsáveis pelo principal serviço de gerenciamento de registros de eventos em sistemas Linux. O `syslogd`, que provê suporte aos registros de eventos gerados pelo Sistema Operacional e pelas aplicações, e o `klogd`, que provê suporte aos registros de eventos gerado pelo kernel.

Syslogd

Arquivo `/etc/syslog.conf`:

- Possui formato padrão que define três propriedades básicas para um registro de evento:
 - Facilidade: especifica o tipo de programa que está gerando o registro.
 - Prioridade: especifica o grau de importância do registro.
 - Ação: especifica para onde será enviado o registro.
- Pode ser especificada mais de uma facilidade com a mesma prioridade:
 - `auth,authpriv.info /var/log/auth.log`
- Múltiplos seletores podem executar uma ação em comum:
 - `*.*;auth,authpriv.none -/var/log/syslog`



É o daemon responsável pelo controle dos registros de eventos do Sistema Operacional e das aplicações. O `daemonsyslogd` provê suporte tanto a sockets do Linux quanto a sockets de internet, o que o torna capaz de trabalhar com registros de eventos gerados localmente e em servidores remotos. A principal desvantagem do `syslogd` é que ele não criptografa os dados que transporta, podendo gerar problemas de segurança se os registros forem armazenados em um servidor remoto, já que esses podem ser facilmente interceptados e alterados.

Arquivo de configuração do syslog

O arquivo `/etc/syslog.conf` controla os registros que serão armazenados pelo `syslogd`, por intermédio de regras específicas. Cada linha desse arquivo deve conter, obrigatoriamente, os campos no formato:

```
facilidade.prioridade    ação
```

Os campos “facilidade” e “prioridade”, juntos, são chamados de seletor e devem ser separados pelo caractere “.”. Já o campo “ação”, deve ser separado do campo seletor por um ou mais

Saiba mais

Veremos que existem outras versões de `syslog` que implementam criptografia de dados e, por isso, seu uso é recomendado.

caracteres de espaço ou de tabulação. Um mesmo tipo de facilidade pode armazenar seus registros em arquivos de log diferentes, dependendo do nível de prioridade do registro, assim como um mesmo tipo de prioridade pode armazenar seus registros em arquivos de log diferentes, dependendo da facilidade que gerou o registro, como mostram os exemplos a seguir:

```
auth,authpriv.info /var/log/auth.log
*.*;auth,authpriv.none -/var/log/syslog
daemon.* -/var/log/daemon.log
kern.* -/var/log/kern.log
mail.* /var/log/mail.log
*.crit /var/log/syslog
```

O caractere “,” é utilizado para especificar mais de uma facilidade com a mesma prioridade, e o caractere “;” para especificar múltiplos seletores (facilidade/prioridade) com uma ação em comum. A seguir veremos, em detalhes, cada um desses campos.

Facilidade

É utilizada para especificar o tipo de programa que está enviando o registro. Mais de uma facilidade pode ser especificada na mesma linha, separadas por vírgulas. Na tabela 7.1, podem ser vistas algumas facilidades utilizadas com maior frequência. Algumas facilidades, como a “security”, emitem um som de alerta e enviam uma mensagem para o console, como forma de alertar o administrador do sistema.

Nome	Facilidade
auth	Programas de autorização/autenticação de usuários.
authpriv	Registros privativos de programas de autorização/autenticação de usuários.
user	Registros genéricos de processos de usuários.
cron	Registros do cron.
ftp	Registros de daemons de FTP.
mail	Registros de daemons de e-mail.
kern	Registros do kernel.
security	Sinônimo para a facilidade auth (obsoleta).
syslog	Registros internos gerados pelo syslogd.
daemon	Registros de outros daemons que não possuem facilidades específicas.

Tabela 7.1
Facilidades.

Prioridade

É usada para especificar o grau de importância de um registro. Os tipos de prioridade estão listados na tabela 7.2, em ordem decrescente de importância. As prioridades também podem ser representadas por números que variam de 0 a 7, onde o número 0 corresponde a emerg, o número 1 corresponde a alert e assim por diante. Quando uma prioridade é definida em uma regra no arquivo */etc/syslog.conf*, as prioridades de maior nível são automaticamente consideradas. Conjuntos de facilidades e prioridades (seletores), podem ser agrupados, separados pelo caractere “;”.

Prioridade	Grau de importância
emerg/panic	Registros gerados antes de o sistema travar.
alert	Mensagem que indica um problema que deve ser resolvido imediatamente.
crit	Registros críticas (exemplo: falhas de hardware).
error/err	Registros de erro.
warning /warn	Registros de aviso.
notice	Mensagem que não indica erro, mas deve receber uma atenção especial.
info	Registros informativos.
debug	Registros de depuração.
none	Indica que nenhuma mensagem da facilidade indicada será registrada.

Ação

Tabela 7.2
Prioridades.

É utilizada para especificar o destino que será dado aos registros. O destino de uma determinada regra definida no arquivo `/etc/syslog.conf` nem sempre é um arquivo armazenado em disco. A tabela 7.3 mostra os tipos destinos que podem ser dados a um registro. Nos casos em que o destino for um arquivo, este deve obrigatoriamente existir. Caso contrário, ocorrerá um erro, pois o `syslogd` não criará o arquivo automaticamente. É importante ressaltar que o caminho completo do arquivo precisa ser especificado.

Destino	Ação
usuário	Os registros serão enviados aos usuários especificados.
arquivo	Os registros serão armazenados no arquivo especificado.
@servidor	Os registros serão enviados ao servidor especificado.
	Os registros serão enviados ao pipe especificado (utilizado para filtros com programas externos).
/dev/console	Os registros serão enviados aos terminais definidos no arquivo <code>/etc/securetty</code> .

Caracteres especiais

Tabela 7.3
Ações.

Existem alguns caracteres especiais que podem ser usados no arquivo `/etc/syslog.conf`.

A tabela 7.4 mostra esses caracteres e seus significados.

Caracteres especiais	Função
!	Todos os níveis de prioridade especificados e acima não serão registrados (operador de negação – NOT).
-	Desativa o sincronismo imediato do arquivo após a gravação de um registro.
*	Quando utilizado no campo seletor, pode abranger todas as facilidades e/ou prioridades. Já no campo “ação”, envia os registros a todos os usuários logados através do comando <i>wall</i> .
=	Utilizado para restringir o envio de registros apenas a um determinado nível de prioridade.

Tabela 7.4
Caracteres especiais.

Arquivos de log especiais

Arquivo `/var/log/lastlog`:

- Registra horário do último acesso ou tentativa de acesso de cada um dos usuários do sistema.

Arquivo `/var/log/utmp`:

- Registra usuários locais logados no sistema.

Arquivo `/var/log/wtmp`:

- Registra detalhes das últimas sessões abertas e encerradas pelos usuários.

Alguns arquivos de log do sistema apresentam-se no formato binário, visando aumentar a segurança dos dados neles armazenados. Dessa forma, torna-se mais complicada a alteração de um determinado registro presente nesses arquivos. Esses arquivos serão vistos a seguir.

Arquivo `/var/log/lastlog`

Arquivo binário que registra o horário do último acesso, ou tentativa de acesso, feito por cada um dos usuários do sistema. Seu conteúdo muda a cada login efetuado e é visualizado toda vez que um determinado usuário efetua login, ou quando os comandos *finger* ou *lastlog* são executados. No Linux, esses registros devem ser explicitamente habilitados, configurando a variável `LASTLOG_ENAB` do arquivo `/etc/login.defs`.

Arquivo `/var/log/utmp`

Arquivo binário que registra informações relacionadas a usuários locais que se encontram logados no sistema em um determinado momento. Para ter acesso aos dados contidos nesse arquivo, podem ser utilizados os comandos *w*, *who* e *finger*.

Arquivo `/var/log/wtmp`

Arquivo binário que registra informações sobre as últimas sessões abertas e encerradas pelos usuários. Para ter acesso aos dados contidos nesse arquivo, é necessária a execução do comando *last*. Recomenda-se configurar as permissões de acesso dos três arquivos acima para o valor 644, definindo o usuário root como o dono dos arquivos.

Klogd

- Provê suporte aos registros de eventos gerados pelo kernel.
- Envia os registros gerados ao `syslogd` por padrão.
- Pode enviar os registros gerados alternativamente para um arquivo à parte.
- Seus registros são classificados em níveis de prioridade.

Programa que controla o registro de eventos gerados pelo kernel, monitorando as mensagens geradas e enviando-as, posteriormente, para o programa `syslogd`. Entretanto, se o `klogd` for invocado com a opção `-f <arquivo>`, esse enviará as mensagens geradas pelo kernel para o arquivo especificado, ao invés de enviá-las ao programa `syslogd`.

No Linux, existem basicamente duas fontes geradoras de registros do kernel: o filesystem `/proc` e a interface `syscall`. Atualmente, as duas foram unificadas, e o `klogd` é responsável por determinar a origem pela qual as mensagens de log serão obtidas. O comando *klogd* verifica, primeiramente, se o file system `/proc` está montado; se estiver, o arquivo `/proc/kmsg` será usado como fonte geradora dos registros de eventos do kernel. Caso contrário, serão utilizadas chamadas de sistema para obter essas informações.



Se um sistema Linux estiver configurado para enviar as mensagens geradas pelo kernel diretamente ao syslogd, o klogd tem a habilidade de priorizar adequadamente essas mensagens. Na tabela 7.5, podem ser vistos os níveis de prioridade definidos pelo klogd, com suas respectivas descrições.

Nível	Prioridade	Grau de importância
0	KERN_EMERG	O sistema está inutilizável.
1	KERN_ALERT	Uma ação deve ser tomada imediatamente para resolver o problema.
2	KERN_CRIT	Mensagens críticas.
3	KERN_ERR	Mensagens de erro.
4	KERN_WARNING	Mensagens de aviso.
5	KERN_NOTICE	Mensagens informativas com um significado maior.
6	KERN_INFO	Mensagens informativas.
7	KERN_DEBUG	Mensagens de depuração

O klogd possui também a habilidade de definir, por meio dos níveis de prioridade, as mensagens que serão enviadas aos terminais definidos no sistema. Normalmente, é atribuído aos terminais o nível 7 de prioridade. Sendo assim, qualquer mensagem com nível de prioridade menor que 7 é enviada aos terminais definidos no sistema. O armazenamento de registros com nível de prioridade 7 é desnecessário, já que para situações de operação normal do sistema esses registros não são relevantes. O klogd deve ser invocado com a opção -c, para que as mensagens geradas pelo kernel não sejam enviadas aos terminais definidos no sistema. É possível definir também, por intermédio da opção -c, os registros de quais níveis de prioridade serão enviados a esses terminais. No exemplo a seguir, todas as mensagens geradas pelo kernel, com nível de prioridade abaixo de 4 serão enviadas aos terminais.

```
# klogd -c 4
```

Tabela 7.5
Prioridades dos registros gerados pelo kernel.

Rsyslog

- Versão aprimorada do syslog.
- Integração com bases de dados.
- Suporte à criptografia de dados.
- Definição de registros personalizados pelo administrador.
- Inclusão da facilidade e da prioridade nos registros.

O rsyslog é uma versão avançada do syslog, que oferece uma série de funcionalidades extras, onde podemos destacar as seguintes.

- Pode armazenar os registros de log em bases de dados MySQL, PostgreSQL etc.;
- Suporte à criptografia de dados utilizando o protocolo TLS sobre TCP (RFC 3195);
- Definição de registros personalizados pelo administrador;
- Inclusão da facilidade e da prioridade nos registros;
- Suporte a expressões regulares.



Devido a essas características, o rsyslog se tornou o pacote padrão para gerenciamento de logs na maioria das distribuições Linux.

Arquivo de configuração do rsyslog

O arquivo `/etc/rsyslog.conf` controla os registros que serão armazenados pelo rsyslog, por intermédio de regras específicas, assim como é feito no syslog. Além disso, é possível definir uma série de opções específicas do rsyslog. A seguir, é apresentado um exemplo comentado do arquivo de configuração do rsyslog:

```
# Módulos

$ModLoad imuxsock      # Suporte ao registro de eventos locais

$ModLoad imklog# Suporte ao registro de eventos do kernel

#$ModLoad immark# Suporte à marcação de mensagens

#$ModLoad imudp        # Suporte ao protocolo UDP

#$UDPServerRun 514

#$ModLoad imtcp        # Suporte ao protocolo TCP

#$InputTCPServerRun 514

# Diretivas Globais

$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat

$FileOwner root

$FileGroup adm

$FileCreateMode 0640

$DirCreateMode 0755

# Regras — Mesmo formato do syslog

auth,authpriv.* /var/log/auth.log

*.*;auth,authpriv.none -/var/log/syslog

cron.* /var/log/cron.log

daemon.* -/var/log/daemon.log

kern.* -/var/log/kern.log

lpr.* -/var/log/lpr.log

mail.* -/var/log/mail.log

user.* -/var/log/user.log
```

Syslog-ng

- Versão aprimorada do syslog.
- Pode filtrar registros com base em seu conteúdo.
- Suporte à criptografia de dados.
- Pode executar comandos caso detecte algum padrão pré-definido em um registro.



O syslog-ng (syslog new generation) é uma versão aprimorada do syslog, que tem como principais características:

- Possibilidade de filtrar os registros de log com base em seu conteúdo, por meio do uso de expressões regulares;
- Possibilidade de permitir o fluxo de registros entre servidores, utilizando qualquer porta TCP. Além disso, seu esquema de configuração é bastante intuitivo e poderoso;
- Suporte à criptografia de dados, utilizando o programa stunnel.

O syslog-ng pode ser configurado para executar um comando caso detecte um padrão em um registro que case com alguma expressão regular, utilizando a diretiva `program()`. A execução automática de programas deve ser bem planejada, pois pode sobrecarregar o servidor ou até mesmo causar indisponibilidades, já que um mesmo tipo de registro pode ser gerado diversas vezes em um pequeno intervalo de tempo e a execução do programa associado a esse pode ser demorada.

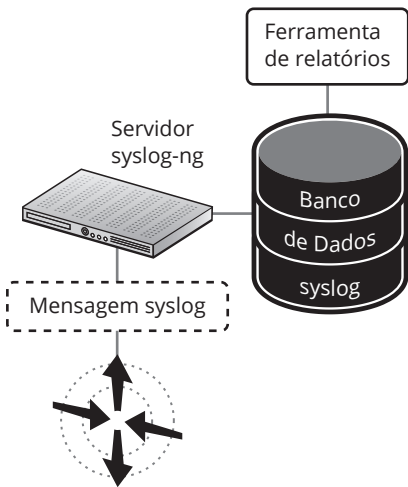


Figura 7.1
Características do
syslog-ng.

A sintaxe utilizada na configuração do syslog-ng é completamente diferente da sintaxe usada pelo syslog, sendo baseada em quatro elementos que serão descritos na tabela 7.6:

Elemento	Descrição
Source	Define a origem dos registros de eventos. Pode ser local ou de um servidor remoto.
Destination	Define o destino dos registros de eventos. Pode ser um arquivo, script ou um servidor de logs.
Filter	Define os filtros criados pelo administrador.
Log	Diretiva que associa os elementos source, destination e filter, criando uma ação.

Tabela 7.6
Sintaxe do
Syslog-ng.

Os elementos “source”, “destination” e “filter” são criados pelo administrador separadamente e posteriormente são associados em uma ação representada pela diretiva “log”. É possível especificar mais de uma origem para um determinado destino ou mais de um destino para mensagens de uma determinada origem em uma mesma regra.

Para que o syslog-ng seja definido no sistema como o daemon padrão de gerenciamento de registros de eventos, é necessário desinstalar o `sysklogd` ou o `rsyslog`, dependendo de qual deles estiver instalado. Além disso, é preciso remover quaisquer configurações referentes a eles que estejam associadas ao `logrotate`. Em seguida, deve ser criado o arquivo `/etc/logrotate.d/syslog-ng` com a sintaxe adequada para que os arquivos de log criados pelo syslog-ng sejam rotacionados de modo correto.

Arquivo de configuração do syslog-ng

O arquivo `/etc/syslog-ng/syslog-ng.conf` controla os registros que serão armazenados pelo syslog-ng. Sua sintaxe é bem diferente da sintaxe utilizada no syslog. A seguir, é apresentado um exemplo comentado do arquivo de configuração do syslog-ng:

Configurações globais

```
options {  
    owner(root);  
    group(root);  
    perm(0640);  
    dir_perm(0740);  
    create_dirs(yes);  
    use_fqdn(no);  
    keep_hostname(yes);  
    use_dns(no);  
    sync(0);  
};
```

Seção Source

Define a origem que vai gerar o evento de log.

```
source src { unix-stream("/dev/log"); internal(); };  
source kernsrc { file("/proc/kmsg"); };
```

Seção Destination

Define o destino onde um evento de log será armazenado, podendo ser um arquivo, terminal, host etc.

```
destination authlog { file("/var/log/auth.log"); };  
destination syslog { file("/var/log/syslog"); };  
destination cron { file("/var/log/cron.log"); };  
destination daemon { file("/var/log/daemon.log"); };  
destination kern { file("/var/log/kern.log"); };  
destination mail { file("/var/log/mail.log"); };  
destination all { file("/var/log/all.log"); };  
destination debug { file("/var/log/debug"); };  
destination messages { file("/var/log/messages"); };  
destination console { usertty("root"); };
```



Seção Filters

Criação dos filtros para interceptar apenas os eventos desejados.

```
filter f_authpriv { facility(auth, authpriv); };

filter f_syslog {not facility(authpriv, mail) and not
match(ppp.*LCP);};

filter f_cron { facility(cron); };

filter f_daemon { facility(daemon); };

filter f_kern { facility(kern); };

filter f_mail { facility(mail) and not match (imapd); };

filter f_debug { not facility(auth, authpriv, news, mail) and not
match(ppp.*LCP); };

filter f_messages { level(info..warn) and not facility(auth,
authpriv, mail, news); };

filter f_emergency { level(emerg); };
```

Seção Log

Nesta seção, a source, filter e destination, para efetivamente realizar o log dos eventos.

```
log { source(src); filter(f_authpriv); destination(authlog); };

log { source(src); filter(f_syslog); destination(syslog); };

log { source(src); filter(f_cron); destination(cron); };

log { source(src); filter(f_daemon); destination(daemon); };

log { source(kernsrc); filter(f_kern); destination(kern); };

log { source(src); filter(f_mail); destination(mail); };

log { source(src); destination(all); };

log { source(src); filter(f_debug); destination(debug); };

log { source(src); filter(f_messages); destination(messages); };

log { source(src); filter(f_emergency); destination(console); };
```

Rotacionamento de arquivos de log

Rotacionamento de arquivos de log:

■ Comando *logrotate*:

- Rotaciona, comprime e cria novos arquivos de log do sistema.
- O rotacionamento pode ser feito de modo automático através do *crontab* do sistema.

O rotacionamento dos arquivos de log do sistema é feito através do comando *logrotate*, que basicamente faz cópias dos arquivos de log em uso, criando novos arquivos que passarão a ser utilizados pelo sistema e pelas aplicações. Opcionalmente, os arquivos de log antigos podem ser compactados para diminuir a utilização de espaço em disco. Para automatizar



o processo de rotação dos arquivos de log, o *logrotate* pode ser ativado via *crontab* do sistema. A rotação de arquivos de log é feita de acordo com o tamanho do arquivo de log especificado, mas a opção *-f* pode ser utilizada para forçar a rotação de um arquivo.

Arquivo de configuração do logrotate

Arquivo */etc/logrotate.conf*:

- Arquivo de configuração do logrotate.
- Principais parâmetros:
 - ▣ Periodicidade em que é feito o rotacionamento.
 - ▣ Opção de compactar ou não os arquivos antigos.
 - ▣ Número de cópias de arquivos anteriores que serão mantidas.
 - ▣ Opção de criar ou não novos arquivos de log após rotacionar os antigos.

Diretório *"/etc/logrotate.d"*:

- Contém, opcionalmente, os arquivos de configuração de rotacionamento dos arquivos de log individuais para cada daemon do sistema.
- Deve ser enviado um sinal HUP ao daemon ao qual o arquivo de log tenha sido rotacionado, por meio do parâmetro *postrotate*.

O arquivo */etc/logrotate.conf* possui as regras utilizadas pelo comando *logrotate* para realizar o rotacionamento adequado dos arquivos de log. A seguir, é apresentado um exemplo comentado do arquivo de configuração do logrotate:

```
# Rotaciona os arquivos de log semanalmente
weekly

# Mantém as quatro últimas cópias dos arquivos de log anteriores
rotate 4

# Erros serão enviados ao usuário root
mailroot

# Cria novos arquivos de log, vazios, após rotacionar os antigos
create

# Compacta os arquivos de log que são rotacionados
compress

# Mantém o primeiro log rotacionado descompactado
delaycompress

# Diretório que contém as configurações para as aplicações do sistema
include /etc/logrotate.d
```



O diretório “/etc/logrotate.d” contém os arquivos de configuração de rotacionamento de arquivos de log individuais para cada daemon. No exemplo a seguir temos a configuração para o named:

```
#/etc/logrotate.d/named

/var/log/named.log {
    missingok

    create 0644 named named

    postrotate

        /bin/kill -HUP `cat /var/run/named/named.pid 2> /dev/null` 2> /
        dev/null || true

    endscript
}
```



É importante enviar um sinal HUP ao daemon que utiliza o arquivo de log que foi rotacionado, para que não ocorram problemas após sua rotação. Isso é feito utilizando o parâmetro postrotate.

O comando logger

- Permite o envio de registros aos arquivos de log do sistema por meio da linha de comando.
- Utilizado geralmente em shell scripts.
- Utilizado também para testar mudanças no arquivo de configuração do syslogd.



O comando *logger* permite que uma mensagem seja enviada aos arquivos de log do sistema por meio da linha de comando. Geralmente é utilizado em shell scripts desenvolvidos pelo administrador do sistema. É possível especificar a facilidade, a prioridade e outros parâmetros da mensagem. O comando *logger* também pode ser usado para testar mudanças feitas no arquivo de configuração do syslogd. Considere a hipótese de que o administrador tenha acrescentado a seguinte linha ao arquivo */etc/syslog.conf*:

```
auth.* /var/log/messages
```

Por intermédio do comando a seguir, esse administrador pode verificar se a alteração feita por ele surtiu o efeito esperado:

```
# logger -p auth.* "mensagem de teste"
```

Se a sintaxe da linha inserida no arquivo */etc/syslog.conf* estiver correta, a mensagem transmitida por meio do comando *logger* será gravada no arquivo */var/log/messages*.

Servidor de logs

- Centraliza os registros de eventos gerados por todos os servidores da rede.
- Evita que registros de eventos importantes sejam perdidos em caso de invasões.
- Facilita a vida do administrador na apuração de incidentes de segurança.



Um sistema Linux mantém um conjunto de arquivos de log que registram as diversas formas em que o sistema foi acessado, e podem registrar indícios da passagem de um invasor pelo sistema. Um invasor experiente, entretanto, pode remover esses indícios antes de sair do sistema. Uma boa estratégia a ser adotada pelo administrador é a instalação de um servidor dedicado ao armazenamento dos registros de eventos gerados por todos os servidores da rede. Essa técnica facilita o gerenciamento, a análise e a solução de problemas que ocorram nos servidores de uma organização, bem como dificulta a remoção de registros que evidenciem uma suposta invasão.

É importante que o administrador de sistemas tenha consciência de que é preciso que os arquivos de log sejam verificados com frequência – ou de nada valerá a implementação de um esquema de segurança. Para habilitar o syslogd a receber os registros de eventos gerados por outros servidores, deve-se iniciá-lo com a opção -r. Isso pode ser feito editando o arquivo */etc/init.d/syslog* e acrescentando a opção -r na variável *SYSLOGD_OPTIONS*, ou diretamente na linha de comando, caso o syslogd não seja iniciado por intermédio do script */etc/init.d/syslog*. Feita a alteração, o syslogd deve ser reiniciado. A instalação de um servidor de logs evita que dados importantes se percam, caso algum servidor da rede seja invadido e tenha seus dados comprometidos.

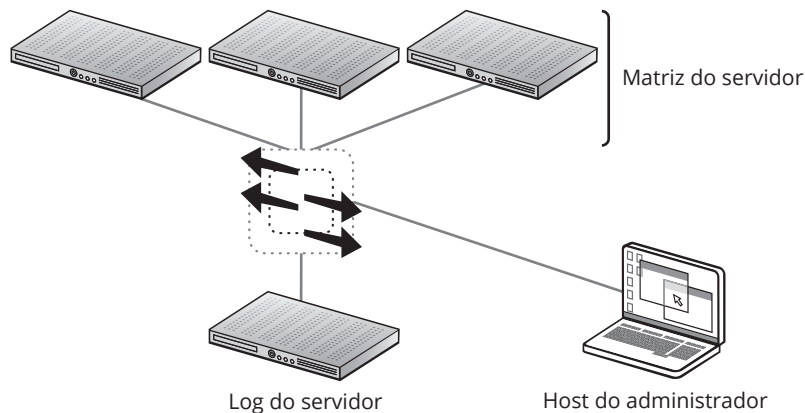


Figura 7.2
Servidor de logs.

Configurando os servidores clientes

Configurando os clientes:

- Arquivo */etc/syslog.conf* dos servidores clientes deve ser alterado, para que os registros de eventos gerados sejam enviados ao servidor de logs.
- Exemplos:
 - ▣ `auth,authpriv.* @servidor_de_logs`
 - ▣ `cron.* @servidor_de_logs`
 - ▣ `mail.* @servidor_de_logs`

O arquivo */etc/syslog.conf* dos servidores clientes também deve ser modificado, para que os registros de eventos gerados por estes sejam enviados ao servidor de logs. A sintaxe utilizada é um sinal de @ seguido do nome do servidor no campo “ação” de cada linha, para redirecionar os registros gerados ao servidor de logs. O daemon *syslogd* dos servidores clientes deve ser reiniciado para que as alterações entrem em vigor. A seguir, pode-se ver um exemplo comentado do arquivo de configuração do *syslogd* de um servidor cliente, que direciona seus registros de eventos a um servidor de logs:



Arquivo /etc/syslog.conf

```
auth,authpriv.* @servidor_de_logs
cron.* @servidor_de_logs
daemon.* @servidor_de_logs
kern.* @servidor_de_logs
mail.* @servidor_de_logs
user.* @servidor_de_logs
```

Aplicativos para análise de arquivos de log

- Auxiliam o administrador na tarefa de analisar arquivos de log.
- Monitoramento de registros em tempo real.
- Execução de ações em resposta a eventos.
- Existem diversas ferramentas que podem ser integradas aos daemonssyslog, rsyslog e syslog-ng.



Uma vez definida a solução de gerenciamento de registros de eventos, bem como a arquitetura adotada para armazenar os arquivos de log, é preciso definir também uma ferramenta de análise e tratamento dos eventos em tempo real. Existem ferramentas que executam essa função, monitorando os arquivos de log de forma otimizada e executando ações em resposta a eventos que ocorrem. Diversos tipos de ações podem ser definidas, como por exemplo: o envio de um e-mail ao administrador; a execução de um comando ou até mesmo a inclusão de uma nova regra no firewall.

Essas ferramentas são de extrema importância para o administrador, pois de nada adianta armazenar os registros de eventos e não for dado nenhum tipo de tratamento a eles. Em ambientes com diversos servidores e equipamentos de conectividade, torna-se praticamente impossível analisar os diversos arquivos de log existentes, em busca de problemas. A seguir, serão apresentadas três ferramentas de análise de arquivos de log.

Logwatch

- Gera relatórios personalizados e os envia por e-mail.
- Suporte à criptografia de dados.
- Compatível com diversas plataformas.
- Interface web para gerenciamento.



O LogWatch é uma ferramenta de monitoramento e análise de arquivos de log escrita em linguagem Perl que gera relatórios personalizados de acordo com a necessidade do administrador e os envia por e-mail.

Essa ferramenta possui um agente de coleta de dados que pode ser instalado em diversos tipos de Sistemas Operacionais, bases de dados, equipamentos de conectividade, appliances etc. Caso não seja possível instalar o agente, devido a algum problema de compatibilidade, os dados podem ser coletados através do syslog. Além disso, deve ser instalado no servidor de logs um agente para consolidar os dados provenientes dos diversos dispositivos da rede. A transferência de dados entre os diversos agentes e o agente central é toda criptografada e para diminuir a quantidade de registros, os agentes removem os menos significativos, reduzindo consideravelmente o volume de dados transmitido.

O LogWatch processa os dados coletados armazenando-os em uma base de dados que pode ser consultada através de uma interface web, que disponibiliza diversos tipos de relatórios. Para acessar a interface web, é preciso possuir uma conta de usuário nesta.

Arquivo de configuração do logwatch

O arquivo `/etc/logwatch/conf/logwatch.conf` possui as regras usadas pelo logwatch para realizar o monitoramento e a análise dos arquivos de log. A seguir, é apresentado um exemplo comentado do arquivo de configuração do logwatch.

```
# Opções globais

LogDir = /var/log

TmpDir = /tmp

mailer = /usr/bin/mail

MailTo = root

UseMkTemp = Yes

MkTemp = /bin/mktemp
```

Define se os relatórios serão enviados para STDOUT ou por e-mail.

```
Print = No
```

Armazena os relatórios em um arquivo (não manda por e-mail/STDOUT).

```
Save = /tmp/logwatch
```

Define se os arquivos de log rotacionados também serão analisados.

```
Archives = Yes

Range = All
```

Define o nível de detalhamento dos relatórios (Low/Med/High).

```
Detail = Med
```

Define os services que serão monitorados.

```
Service = All
```

Opção utilizada para que o logwatch analise somente o arquivo especificado.

```
LogFile = messages
```

Define se somente esse servidor será monitorado pelo logwatch.

```
HostLimit = Yes
```

Swatch

- Analisa em tempo real arquivos de log.
- Executa ações baseadas na detecção de padrões.
- Arquivo de configuração baseado em duas diretivas (watchfor e ignore).
- Tipos de ações: impressão de mensagens no console; emissão de beeps; execução de comandos e envio de e-mails.



O Swatch (Simple log watcher) é uma ferramenta que analisa em tempo real um arquivo de log e, caso encontre algum padrão definido pelo administrador, executa uma ação. Cada arquivo de log a ser analisado deve ser passado como parâmetro na linha de comando do swatch. Para evitar problemas de sobrecarga ou indisponibilidade no servidor, o swatch pode ser configurado para aguardar um intervalo de tempo entre a execução de duas ou mais ações disparadas por um ou mais eventos.

Arquivo de configuração do Swatch

O arquivo de configuração do swatch é o `.swatchrc`, e fica localizado por padrão no diretório home do usuário swatch. O arquivo `.swatchrc` possui basicamente duas diretivas.

- **watchfor:** contém padrões de texto e ações que serão executadas caso o padrão seja detectado;
- **ignore:** especifica padrões que serão ignorados caso sejam detectados.

A seguir, é apresentado um exemplo comentado do arquivo de configuração do swatch.

Tentativas de login que falharam:

```
watchfor /INVALID|REPEATED|INCOMPLETE/

    echo

    bell 3

    exec "/usr/local/sbin/badloginfinger $0"
```

Travamentos no sistema:

```
watchfor /(panic)/

    echo

    bell

    mail "suporte@dominio"

    ignore = ./*/
```

A tabela 7.7 mostra as ações que podem ser executadas pelo swatch e suas respectivas descrições:

Tabela 7.7
Ações do Swatch.

Ação	Descrição
echo modo	Imprime no console o padrão detectado. O modo é opcional e pode especificar a cor e o estilo da fonte.
bell número	Emite uma quantidade de beeps indicada pelo número após a ação bell.
exec comando	Executa o comando especificado.
pipe comando	Envia o comando especificado para outro comando através de um pipe.
mail endereço=email,subject=título	Envia um e-mail para um ou mais endereços.
write usuário1:usuário2	Envia uma mensagem para um ou mais usuários, utilizando o comando <i>write</i> .
throttle hora:minuto:segundo	Define o período de tempo que será aguardado para uma ação relacionada a um padrão ser executada novamente.

Logcheck



- Analisa arquivos de log.
- Análise baseada na detecção de anomalias.
- Emissão de relatórios personalizados.
- Mensagens organizadas em categorias ("Active System Attacks", "Security Violations" e "Unusual Activity").

O Logcheck é uma ferramenta de análise de arquivos de log e geração de relatórios. Quando alguma anomalia (variações em um padrão) é detectada, o logcheck a analisa na tentativa de identificar alguma intrusão ou violação do sistema. As mensagens de e-mail enviadas pelo logcheck são organizadas em categorias ("Active System Attacks", "Security Violations" e "Unusual Activity"), o que facilita a análise dos relatórios. Os tipos de mensagens que serão incluídos nos relatórios podem ser personalizados pelo administrador através dos arquivos e diretórios presentes no diretório "/etc/logcheck". Arquivos e diretórios que contêm a palavra "ignore" em seus nomes armazenam expressões regulares que não serão gravadas nos relatórios gerados pelo logcheck.

O logcheck, por padrão, envia relatórios automaticamente através do cron, a cada dois minutos após as horas cheias (1:02, 2:02 etc.), mas esse padrão pode ser alterado a critério do administrador no arquivo */etc/cron.d/logcheck*.

Os arquivos de log que são monitorados pelo logcheck são definidos no arquivo */etc/logcheck/logcheck.logfiles*. Cada linha desse arquivo contém o caminho completo de cada arquivo de log que será monitorado.

Arquivo de configuração do logcheck

O arquivo */etc/logcheck/logcheck.conf* possui as regras utilizadas pelo logcheck para realizar o monitoramento e a análise dos arquivos de log. A seguir, é apresentado um exemplo comentado do arquivo de configuração do logcheck.

Define o formato de data nos títulos dos e-mails enviados

```
DATE="$(date +%Y-%m-%d %H:%M%)"
```

Define se a mensagem terá cabeçalho/rodapé (header.txt/footer.txt)

```
INTRO=1
```

Define o nível de filtragem (workstation, server ou paranoid)

```
REPORTLEVEL="server"
```

Define os e-mails que receberão os relatórios das análises

```
SENDMAILTO=suporte@dominio
```

Define o formato do nome do servidor no subject

```
FQDN=1
```

Define se as entradas duplicadas serão removidas ou não

```
SORTUNIQ=0
```

Define se o arquivo *cracking.ignore.d* será verificado

```
SUPPORT_CRACKING_IGNORE=0
```



Define qual o diretório que contém os arquivos de regras

```
RULEDIR="/etc/logcheck"
```

Define o comportamento do syslog-summary

```
SYSLOGSUMMARY=0
```

Define o tipo de subject

```
ATTACKSUBJECT="Attack Alerts"
```

```
SECURITYSUBJECT="Security Events"
```

```
EVENTSSUBJECT="System Events"
```

Define se o prefixo [logcheck] será incluído ou não no subject

```
ADDTAG="no"
```

Recomendações básicas de segurança

- Manter os servidores da rede com horário sincronizado:
 - ▣ Um servidor de NTP pode ser usado para resolver esse problema.
 - ▣ Esse procedimento evita inconsistências nos horários dos registros armazenados nos arquivos de log.
- Desabilitar qualquer serviço que não seja o syslog no servidor de log.
- O acesso a esse servidor deve ser restringido ao máximo.
- Deve ser criada uma regra no firewall local do servidor de logs, permitindo o acesso somente à porta 514 UDP (syslogd) desse servidor, e apenas para servidores da rede local.
- Evitar que o servidor de log possua contas de usuários.
- Realizar um backup diário do servidor de logs.
- Habilitar algum módulo de segurança no servidor de logs:
 - ▣ SELinux.
 - ▣ LIDS.

São apresentadas a seguir algumas recomendações básicas de segurança, que não são obrigatórias, mas devem ser seguidas pelo administrador de sistemas, nos casos em que ele julgar necessário, a fim de tornar seus sistemas mais seguros.

1. Manter os relógios de todos os servidores da rede sincronizados por meio do protocolo Network Time Protocol (NTP), para evitar inconsistências nos horários dos registros dos eventos gerados pelos diversos servidores, facilitando, assim, possíveis auditorias em casos de incidentes de segurança.
2. Recompilar o syslogd, modificando o nome e a localização do arquivo */etc/syslog.conf*, para dificultar possíveis tentativas de apagar registros de invasões. Para isso, é necessário ter acesso ao código-fonte do syslogd. A mudança pode ser feita, alterando, no arquivo *syslogd.c*, a linha a seguir:

```
#define _ PATH _ LOGCONF "/etc/syslog.conf"
```

Instalar um servidor dedicado ao armazenamento de registros de eventos gerados por todos os servidores da organização, desativando qualquer outro serviço desse servidor. Deve ser criada uma regra, no firewall local do servidor de logs, que permita o tráfego de entrada somente para a porta desse servidor, partindo apenas de servidores da rede local.

3. Não criar contas de usuários no servidor de logs.
4. Fazer um backup diário do servidor de logs para garantir a integridade, a autenticidade e a disponibilidade dos arquivos de log, pois, se esses dados forem perdidos por algum motivo, o administrador ficará sem acesso aos registros de eventos de todos os servidores da rede.
5. É recomendável que o servidor de logs utilize módulos de segurança como o SELinux ou o LIDS.
 - **SELinux:** o Security Enhanced Linux (SELinux) é um projeto desenvolvido pela National Security Agency (NSA), do governo dos EUA, que implementa um sistema de controle de acesso chamado Mandatory Access Control (MAC). O MAC modifica a forma de acesso aos recursos do sistema, limitando os acessos dos usuários e das aplicações, deixando a cargo do Sistema Operacional o controle de acesso aos recursos de forma mandatória. As políticas impostas pelo MAC fornecem um nível de segurança extra contra acessos não autorizados;
 - **LIDS:** o Linux Intrusion Detection System (LIDS) é um patch para o kernel do Linux que adiciona funcionalidades de segurança, como: Mandatory Access Control (MAC); detecção de port scanners; proteção contra acessos a arquivos e diretórios (inclusive para o usuário root) e proteção extra para processos, módulos e interfaces. Com o LIDS é possível restringir qualquer tipo de acesso ao sistema, e qualquer tentativa de acesso não autorizada em sistemas protegidos por ele é reportada através de e-mails e registros nos arquivos de log.

8

Boot & Shutdown e Kernel

objetivos

Compreender os processos de inicialização e de desligamento de um sistema;
Entender as arquiteturas monolítica e modular do kernel e o conceito de módulos;
Aprender sobre o processo de configuração e compilação do kernel.

conceitos

Inicialização do sistema; Shutdown; Arquitetura do Kernel; Configurando e compilando o kernel.

Inicialização do sistema

- BIOS verifica o sistema e aciona carregador de primeiro estágio no MBR (512 bytes).
- Carregador de boot de primeiro estágio dispara o carregador de boot de segundo estágio.
- Carregador de boot de segundo estágio carrega o kernel na memória.
- O kernel aciona drivers de dispositivos e dispara o processo init.
- O init monta as partições e ativa os serviços.



Na aviação, dizem que os momentos mais críticos são o da decolagem e o da aterrissagem. Da mesma forma, pode-se dizer que a inicialização e o desligamento de um sistema Linux também são os pontos mais críticos da sua operação. Nesta sessão de aprendizagem, veremos que ambos os processos têm de ser executados de forma correta para garantir o bom funcionamento do sistema. Durante a inicialização do sistema, várias mensagens são mostradas na tela. Apesar de algumas dessas mensagens possuírem, à primeira vista, um certo grau de complexidade, o processo de inicialização do sistema pode ser visto de uma forma simples. A inicialização do sistema pode ser dividida nos seguintes estágios:

- **BIOS (Basic Input/Output System):** programa responsável por controlar os primeiros passos do processo de boot e também por prover a interface de mais baixo nível para os dispositivos periféricos. Verifica o sistema e dá início ao carregador de boot de primeiro estágio, localizado no Master Boot Record (MBR) do disco rígido primário, que representa o primeiro setor desse dispositivo;



- **Carregador de boot de primeiro estágio:** possui apenas 512 bytes e contém instruções, em código de máquina, para auxiliar o computador no seu processo de inicialização. Esse código é chamado de carregador de boot de primeiro estágio, e sua única função é localizar e carregar, na memória do sistema, o carregador de boot de segundo estágio. Carrega a si mesmo na memória e dispara o carregador de boot de segundo estágio a partir de uma partição inicializável;
- **Carregador de boot de segundo estágio:** sua principal tarefa é carregar o kernel do Linux na memória. Os dois principais carregadores de boot do Linux são o LILO (Linux Loader) e o Grand Unified Boot Loader (GRUB), que serão vistos a seguir;
- **Iniciando o kernel:** o arquivo de imagem do kernel é carregado para a memória e os drivers de dispositivos são carregados pelo kernel, já em execução. Nessa fase, mensagens são exibidas na tela, indicando o reconhecimento dos dispositivos e a carga de seus respectivos drivers. O kernel monta a partição raiz, inicialmente, no modo “somente leitura”, e transfere o controle do processo de boot para o programa `/sbin/init`;
- **Init:** o processo `init` passa a ser responsável pela ativação de todos os processos que implementam os serviços configurados no sistema. Ele também é responsável por montar todas as partições listadas no arquivo `/etc/fstab`. Após carregar todos os programas, o prompt de login é apresentado. O `init` está sempre em execução e sua morte provoca a reinicialização do sistema, ou seja, é realizado um shutdown e em seguida, o sistema é inicializado.

A seguir, cada um desses estágios será apresentado em mais detalhes.

Basic Input/Output System (BIOS)

- Controla os primeiros passos do processo de boot.
- Testa o hardware do sistema.
- Busca por um dispositivo através do qual o boot deve ser executado (disquete, CD, DVD ou HD).
- Realizar carga do programa presente no MBR.



Quando um computador é inicializado, a CPU procura, na memória não volátil do sistema, um programa especial chamado BIOS e o executa. Estamos tomando como base um computador com arquitetura do tipo x86. Outros tipos de computadores acabam usando esquemas similares, como o sistema Alpha, que utiliza o console SRM. Uma vez carregado, o BIOS executa o POST (Power-On Self Test), que se encarrega de testar todo o hardware do sistema. Em seguida, o BIOS identifica um dispositivo a partir do qual o sistema deve ser carregado. Geralmente, ele começa verificando os drives de disquete e de CD/DVD em busca de mídias que tenham algum código de boot do sistema e, em seguida, verifica os discos rígidos. A ordem de verificação dos dispositivos pode ser alterada pelo administrador através do setup do computador. O BIOS, então, carrega qualquer programa que esteja armazenado no MBR na memória RAM. Dessa forma, o controle do processo de boot do sistema é repassado pelo BIOS para esse programa.

Carregadores de boot (boot loaders)

LILO e GRUB:

- Apresentam ao administrador diferentes opções de Sistemas Operacionais ou versões de kernel que podem ser utilizadas.





- Localizam no diretório “/boot” o arquivo binário correspondente ao kernel selecionado.
- Carregam o initrd na memória.
- Passam o controle do processo de boot para o kernel.

Como visto anteriormente, existem dois tipos de carregadores de boot: o primeiro estágio e o de segundo estágio. O carregador de boot de primeiro estágio tem como principal função carregar o carregador de boot de segundo estágio, que, por sua vez, tem como principal função carregar o kernel em memória. Estando o carregador de boot de segundo estágio carregado na memória, ele apresenta diferentes opções de Sistemas Operacionais ou kernels que foram configurados para dar boot no sistema, oferecendo ao administrador a oportunidade de selecionar o que ele deseja utilizar no momento.

Saiba mais

Se o computador possui apenas o Linux instalado e somente uma versão de kernel, apenas uma opção de boot deverá ser exibida.



Após a seleção do kernel que deve ser utilizado para inicializar o sistema, o carregador de boot de segundo estágio localiza, no diretório “/boot”, o arquivo binário correspondente ao kernel selecionado, que geralmente segue o padrão /boot/vmlinuz-<versão_do_kernel>. Em seguida, o carregador de boot carrega, na memória do sistema, uma imagem do disco chamada de inicial RAM disk (initrd), que é utilizada pelo kernel para carregar drivers não compilados, necessários para dar boot no sistema. Essa técnica é particularmente útil em sistemas que possuem discos SCSI cujo driver foi compilado como um módulo. Após a carga do kernel e do initrd na memória, o carregador de boot de segundo estágio passa o controle do processo de inicialização do sistema para o kernel. A seguir, são apresentados mais detalhes a respeito dos dois principais carregadores de boot do Linux, o LILO e o GRUB.

LILO



- Pioneiro entre os carregadores de boot.
- Permite a passagem de parâmetros para o kernel.
- Arquivo de configuração:
 - ▣ `/etc/lilo.conf`
- Busca informações de opções de boot no MBR.
- Demanda atualização do MBR sempre que o arquivo de configuração for modificado ou kernel atualizado.

Conforme já foi dito, o programa LILO é um carregador de boot e, como tal, é responsável pela carga do kernel em memória. O LILO permite que o Linux seja instalado em computadores com múltiplos Sistemas Operacionais. Assim, durante a inicialização do sistema, o administrador pode selecionar o Sistema Operacional a ser carregado. Essa seleção é realizada através do rótulo do Sistema Operacional, previamente definido no arquivo de configuração do LILO (`/etc/lilo.conf`). No entanto, dependendo do disco rígido instalado no sistema, o kernel pode não reconhecê-lo automaticamente. Nesses casos, deve-se informar o rótulo do Linux seguido pelos parâmetros que permitem ao kernel reconhecer o dispositivo. Caso não seja selecionado nenhum Sistema Operacional no momento do boot, aquele que estiver definido como padrão será carregado. Embora possam existir mais de dois Sistemas Operacionais instalados, o termo dual-boot costuma ser utilizado nesses casos.

O arquivo de configuração do LILO (`/etc/lilo.conf`) contém as informações que ele utiliza durante a inicialização do sistema. É importante frisar que, após a edição desse arquivo, é necessário executar o comando `lilo` para efetivar as mudanças para a próxima inicialização do sistema. É possível informar parâmetros do sistema ao kernel por meio do LILO, tais como a partição de boot e características de dispositivos instalados, mas que não estão sendo automaticamente reconhecidos. Para mais informações, a página de manual do LILO pode ser consultada.



É importante frisar que toda vez que for feita uma atualização manual do kernel, o comando *lilo* deve ser executado para que a informação pertinente seja copiada para o MBR.

GRUB

- Possui mais recursos e está se tornando o gerenciador de boot padrão do Linux.
- Permite a passagem de parâmetros para o kernel.
- Arquivo de configuração:
 - ▣ `/boot/grub/grub.cfg`
- Oferece três diferentes interfaces com diferentes funcionalidades.
- Consegue ler diversos tipos de partições, como: ext2, ext3, ext4, Reiserfs, NTFS etc.
- Não demanda atualização do MBR.



O GRUB é um carregador de boot que permite selecionar o Sistema Operacional ou a versão do kernel que será carregada no momento da inicialização. Ele também permite a passagem de parâmetros para o kernel durante o processo de inicialização. O GRUB consegue ler diversos tipos de partições, como ext2 e ext3, e, portanto, pode acessar o seu arquivo de configuração durante a inicialização do sistema. Esse comportamento torna desnecessária a transferência de informações para o MBR toda vez que mudanças de configurações forem realizadas. O GRUB oferece três diferentes interfaces que são descritas a seguir:

- **Menu:** conforme citado anteriormente, quando o GRUB é carregado, ele procura pelo seu arquivo de configuração e, ao encontrá-lo, utiliza-o para construir a lista de menu e, em seguida, exibi-la com as opções de boot disponíveis;
- **Edição de menu:** o GRUB possibilita a edição das entradas de seu menu, no momento em que esse é exibido durante o boot do sistema. Para isso, basta pressionar-se a tecla “e”;
- **Linha de comando:** se o arquivo de configuração não puder ser encontrado, ou estiver ilegível, o GRUB carregará a interface de linha de comando a fim de permitir ao administrador digitar os comandos necessários para carregar um determinado Sistema Operacional. Mesmo que o menu do GRUB seja exibido, ainda é possível acessar sua linha de comandos, pressionando a tecla “C”. Se tiver sido definida uma senha de acesso no arquivo de configuração do GRUB, através do parâmetro “password”, o administrador deve utilizar a tecla “P” para informá-la, antes de ter acesso à linha de comandos. Para verificar os comandos disponíveis, o administrador deve pressionar duas vezes a tecla “TAB”, que também pode ser usada para completar comandos e parâmetros, assim como no shell do Linux. A seguir, são listados alguns dos comandos mais importantes:
- **boot:** pode ser utilizado para executar o boot a partir de um Sistema Operacional ou versão de kernel;
- **chainloader<nome-arquivo>:** alguns Sistemas Operacionais como o DOS e o Windows armazenam seus gerenciadores de boot na partição em que estão instalados. É através desse comando que o GRUB passa o controle do processo de boot para esses gerenciadores, que, por sua vez, carregam o Sistema Operacional selecionado;
- **initrd<nome-arquivo>:** permite ao administrador especificar uma imagem do disco a ser utilizada no momento do boot, cuja funcionalidade já foi descrita anteriormente;
- **kernel<nome-arquivo><opção-1><opção-n>:** especifica o nome do arquivo do kernel que deve ser utilizado para dar boot no sistema. As opções são repassadas para o kernel quando ele é carregado;

- **root<dispositivo-partição>**: configura a partição raiz do GRUB para um dispositivo e partição particular, tal como (hd0,0), e monta a partição de forma que ela possa ser lida;
- **rootnoverify<dispositivo-partição>**: faz o mesmo que o comando anterior, mas não monta a partição.

O GRUB usa uma notação diferente da utilizada pelo Linux para referenciar os discos e partições do sistema. A tabela 8.1 apresenta uma comparação entre as notações utilizadas pelo Linux e pelo GRUB.

Tabela 8.1
Comparativo
entre as notações
utilizadas pelo
Linux e pelo GRUB.

Notação utilizada pelo Linux	Notação utilizada pelo GRUB
/dev/hda e /dev/sda	(hd0)
/dev/hda1 e /dev/sda1	(hd0,0)
/dev/hda2 e /dev/sda2	(hd0,1)
/dev/hdb e /dev/sdb	(hd1)
/dev/hdb1 e /dev/sdb1	(hd1,0)
/dev/hdb2 e /dev/sdb2	(hd1,1)

O arquivo de configuração do GRUB (*/boot/grub/grub.cfg*) é composto por uma sequência de comandos. As configurações globais da interface de menu geralmente estão localizadas no início do arquivo e são seguidas pelas diferentes entradas relativas a cada Sistema Operacional ou kernel que são exibidas no menu. A seguir, é apresentado um exemplo comentado do arquivo de configuração do GRUB:

```
default=0

timeout=5

splashimage=(hd0,0)/GRUB/splash.xpm.gz

hiddenmenu

title Fedora (3.2.8-1.fc16.x86_64)
    root (hd0,0)
    kernel /boot/vmlinuz-3.2.8-1.fc16.x86_64 ro root=LABEL=/ rhgb quiet
    initrd /boot/initrd-3.2.8-1.fc16.x86_64.img

title windows
    rootnoverify (hd0,0)
    chainloader +1
```

Ao compilar um novo kernel manualmente, basta ao administrador criar um novo bloco title nesse arquivo, colocando os parâmetros referentes ao novo kernel. O processo de instalação de um novo kernel pode ser feito de forma a automatizar essa tarefa.

Iniciando o kernel

Funções do kernel:

- Inicializa e configura a memória do computador.
- Configura os dispositivos, carregando seus drivers.
- Monta a partição raiz temporariamente no modo “somente leitura”.
- Libera memória não utilizada.
- Chama o processo init.
- O comando *dmesg* pode ser utilizado para listar as mensagens apresentadas durante a inicialização do sistema.



No processo de boot, o kernel se encarrega de inicializar e configurar, primeiramente, a memória do computador e, em seguida, os vários dispositivos presentes no sistema, carregando todos os drivers necessários. Também é responsável por montar a partição raiz, temporariamente, no modo de “somente leitura”, e liberar partes da memória que não estão sendo utilizadas. Em seguida, chama o processo init, que cuidará de finalizar o boot do sistema.

Durante a inicialização do kernel, enquanto os dispositivos vão sendo detectados e seus respectivos drivers carregados em memória, várias mensagens são apresentadas na tela. Após a inicialização do sistema, o administrador pode analisar essas mensagens usando o comando *dmesg*. Essas mensagens são armazenadas no arquivo de log do sistema */var/log/dmesg* ou */var/log/messages*.

Processo init

Funções do init:

- Inicializa todos os demais processos que implementam os serviços do sistema.
- Define níveis de execução (*run levels*) para o sistema.
- Identifica os processos que devem estar ativos em cada nível.
- Arquivo de configuração: */etc/inittab*.



Após o kernel ser inicializado, o processo init é ativado. Esse processo é direta ou indiretamente responsável por inicializar todos os demais processos que implementam os serviços disponíveis no sistema. Entre esses processos, pode ser citado o *mingetty*, que realiza o controle de login no sistema. Sempre que uma sessão de trabalho é encerrada, o processo init reativa o processo *mingetty* para aquele console virtual. No Linux, seis processos *mingetty* são inicializados para os consoles virtuais e podem ser acessados através da combinação das teclas “Alt+F1” até “Alt+F6”.

O Linux define níveis de execução (*runlevels*) que identificam o estado atual do sistema. Cada nível possui um propósito específico, que determina o conjunto de processos que devem estar ativos. Para identificar os processos que devem ser ativados, o init lê o arquivo de configuração */etc/inittab*. Nesse arquivo, estão todas as informações sobre os níveis de execução e os processos a serem inicializados. Cada linha desse arquivo indica ao init um comando ou script que deve ser executado. Difícilmente o administrador precisa alterar o conteúdo do arquivo */etc/inittab*. Entretanto, é indispensável entender como esse arquivo é processado.

Arquivo `/etc/inittab`



- Executa o script `/etc/rc.d/rc.sysinit`.
- Executa os scripts do diretório «`/etc/rc.d/rcX.d`», onde X representa o nível de execução atual.
- Ativa as partições de swap, verifica a consistência dos sistemas de arquivos e inicializa o sistema de quotas.
- Por fim, executa o script `S99local` (link para `/etc/rc.d/rc.local`).
- O comando `telinit` pode ser usado para mudar o nível de execução atual.

Cada linha do arquivo `/etc/inittab` é composta por quatro campos separados pelo caractere “:”, que são descritos a seguir:

- **Id**: identificador único para a linha, que pode ter qualquer valor;
- **Rstate**: indica o nível de execução do processo disparado pela linha;
- **Action**: indica como o `init` deve tratar o processo disparado por essa linha. As principais opções para esse campo são:
 - **initdefault**: indica o nível de execução padrão na inicialização;
 - **wait**: espera o término do processo antes de ler a próxima linha do arquivo;
 - **once**: executa o processo uma única vez;
 - **respawn**: reinicia o processo caso ele termine.
- **Process**: informa o comando ou script a ser executado.

A primeira linha do arquivo `/etc/inittab` define o nível de execução padrão assumido durante a inicialização do sistema. O primeiro script que o `init` executa, presente na segunda linha do arquivo `/etc/inittab`, é o `/etc/rc.d/rc.sysinit`. Esse script ativa as partições de swap, verifica a consistência dos sistemas de arquivos e inicializa o sistema de quotas. Após concluir esse script, o `init` ativa o script `/etc/rc.d/rc`, passando como parâmetro o valor do nível de execução atual. Esse script ativa todos os scripts que estão armazenados no diretório “`/etc/rc.d/rcX.d`”, onde X é o valor do nível de execução atual. Assim, no diretório “`/etc/rc.d`” existem seis diretórios `rcX.d` diferentes, um para cada nível de execução, que contêm os scripts que devem ser executados em cada nível. Os scripts começam com as letras “K” (kill scripts) ou “S” (start script), seguidas de um número e um nome que identifica o serviço controlado pelo script.

Também são ativados em ordem crescente, baseada nos números que formam seus nomes. No caso dos níveis de execução 2, 3 e 5, o último script executado é o `S99local`, que é um link simbólico para o script `/etc/rc.d/rc.local`. Esse script pode ser editado pelo administrador para configurar serviços adicionais locais que devem ser executados durante a inicialização do sistema. É importante o administrador utilizar esse script para evitar a ativação manual de serviços, todas as vezes que o sistema for iniciado. Por fim, o processo `init` ativa os processos `mingetty` para cada console virtual.

- Define o nível de execução padrão do sistema.
- Indica quais processos devem ser iniciados, finalizados ou reiniciados.
- Indica que ações tomar quando o sistema entra em um novo nível de execução.

Saiba mais

Os kill scripts são responsáveis por “matar” os processos que implementam um determinado serviço. Vale ressaltar que eles são ativados, em ordem crescente, baseada nos números que formam seus nomes. Já os *start scripts* são responsáveis por ativar os processos que implementam um determinado serviço.



■ Formato:

■ id:rstate:action:process

O nível de execução pode ser modificado usando o comando *telinit*, sem precisar editar o arquivo */etc/inittab*. O comando *telinit* requer um único parâmetro, que é o nível de execução desejado. A tabela 8.2 descreve os níveis de execução do Linux.

Nível	Descrição
0	Desliga o sistema.
1	Modo monousuário.
2	Modo multiusuário sem NFS e SMB.
3	Modo multiusuário completo.
4	Não utilizado.
5	Modo multiusuário com X-Window System.
6	Reinicializa o sistema.

Tabela 8.2
RunLevels.

A seguir, pode ser visto um exemplo comentado do arquivo */etc/inittab*.

```
# Define o nível de execução padrão
id:3:initdefault:

# Executa o script /etc/rc.d/rc.sysinit
si::sysinit:/etc/rc.d/rc.sysinit

# Executa o script /etc/rc.d/rc, passando como parâmetro para ele o
nível de execução atual.
10:0:wait:/etc/rc.d/rc 0
11:1:wait:/etc/rc.d/rc 1
12:2:wait:/etc/rc.d/rc 2
13:3:wait:/etc/rc.d/rc 3
14:4:wait:/etc/rc.d/rc 4
15:5:wait:/etc/rc.d/rc 5
16:6:wait:/etc/rc.d/rc 6


# Habilita a reinicialização do sistema através das teclas
CTRL+ALT+DEL.
ca::ctrlaltdel:/sbin/shutdown -t3 -r now


# Executa um shutdown caso ocorra uma falha de energia. Para isso,
o daemonpowerd deve estar instalado no sistema e o servidor deve
ser alimentado por um no-break.
```

```

pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure. System
Shutting Down"

# Cancela o shutdown anterior caso a energia seja restabelecida.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored. Shutdown
Cancelled"

# Executa o processo mingetty nos níveis de execução 2, 3, 4 e 5.
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

# Executa o xdm no nível de execução 5
x:5:respawn:/etc/X11/prefdm -nodaemon

```

Formato: id:rstate:action:process

Figura 8.1
Formato das linhas
do arquivo
/etc/inittab.



Upstart

- Utilizado no Ubuntu desde 2006.
- Melhora o desempenho do boot através da execução de jobs em paralelo.
- Orientado a eventos.
- Compatível com o sysVinit.
- Supervisiona os serviços em execução.



Criado por Scott James Remnant, da empresa Canonical Ltd., o upstart tem a função de substituir o processo Init nos sistemas Linux. Ele é usado no Ubuntu desde 2006, sendo o responsável por iniciar e finalizar todas as tarefas e serviços durante o processo de boot e shutdown do sistema, além de supervisionar os processos em execução. O principal objetivo do projeto era melhorar o desempenho do boot através de paralelismo, substituindo o antigo init por um binário que fosse capaz de tomar decisões com base em eventos e ao mesmo tempo manter a total compatibilidade com o sistema antigo.

Conforme mencionado anteriormente, o Upstart é baseado em eventos: um evento pode iniciar um serviço que, por sua vez, pode disparar um outro evento – esse evento pode disparar um outro serviço, e assim por diante. Outra característica interessante é que o Upstart supervisiona os serviços em execução; dessa forma, o daemon pode reiniciar automaticamente um serviço que morreu de forma inesperada.

Configuração

Os arquivos de configuração dos serviços ficam no diretório “/etc/init” e possuem a extensão .conf. Cada arquivo define um job do Upstart que pode ser uma tarefa ou um serviço. Por questões de compatibilidade, a sintaxe é bem semelhante aos arquivos do sysvinit.

Exemplo 1: /etc/init/control-alt-delete.conf

```
1. # control-alt-delete – emergency keypress handling
2. #
3. # This task is run whenever the Control-Alt-Delete key combination is
4. # pressed, and performs a safe reboot of the machine.
5.
6. description      “emergency keypress handling”
7. author           “Scott James Remnant <scott@netsplit.com>”
8.
9. start on control-alt-delete
10.
11. task
12. exec shutdown -r now “Control-Alt-Delete pressed”
```

- **Linha 6:** descrição do job;
- **Linha 7:** informações sobre o autor do script;
- **Linha 9:** define quando e onde o job será executado (será executado quando o evento control-alt-delete for acionado);
- **Linha 11:** inicia a configuração da tarefa;
- **Linha 12:** comando que será executado.

Exemplo 2: /etc/init/udev.conf

```
1. # udev – device node and kernel event manager
2. #
3. # Theudev daemon receives events from the kernel about changes in the
4. # /sys filesystem and manages the /devfilesystem.
5.
6. description      “device node and kernel event manager”
7.
```



```

8. start on virtual-fileystems
9. stop on runlevel [06]
10.
11. expect fork
12. respawn
13.
14. exec /sbin/udev --daemon

```

- **Linha 6:** descrição do job;
- **Linha 8:** especifica quando o serviço deve ser iniciado, ou seja, após o evento virtual-fileystems ter acontecido;
- **Linha 9:** determina que esse job deve ser finalizado quando o evento runlevel for acionado em qualquer um dos níveis de 0 a 6;
- **Linha 11:** o Upstart vai aguardar o processo executar uma chamada de sistema fork uma vez;
- **Linha 12:** caso o job morra de forma inesperada (exitcode diferente de 0), ele será reiniciado;
- **Linha 14:** comando que será executado.

Gerenciamento dos jobs

O Upstart possui o binário initctl, que é utilizado para gerenciar os jobs existentes.

- Lista todos os jobs configurados:
 - ▣ # initctl list
- Iniciando um serviço:
 - ▣ # initctl start rsyslog
 - ▣ # start rsyslog
- Finalizando um serviço:
 - ▣ # initctl stop ssh
 - ▣ # stop ssh
- Consultando o status:
 - ▣ # initctl status cron
 - ▣ # status cron

systemd

Principais características:

- ▣ Compatível com scripts de inicialização SysV e LSB.
- ▣ Fornece recursos de paralelização.
- ▣ Utiliza sockets e D-Bus activation para inicializar os serviços.
- ▣ Inicialização sob demanda.



- Controla os processos utilizando control groups (cgroups).
- Suporte para snapshot e restore.
- Mount e automount.
- Implementa uma elaborada lógica de dependência de serviços.
- Também pode ser utilizado para substituir os syslogd.

Mantido por Lennart Poettering e um extenso grupo de parceiros, o systemd é uma nova proposta para o sistema de inicialização do Linux. Como principais metas, o systemd propõe criar menos processos e iniciar mais processos em paralelo. Seu funcionamento consiste basicamente em adiar o início dos processos até que eles sejam realmente necessários e, quando isso acontecer, utilizar todos os recursos disponíveis para iniciá-lo.

Shutdown

- Sua função é desligar ou reinicializar o sistema de forma correta.
- Envio dos sinais TERM e KILL para os processos.
- Opções de uso:
 - **-r**: reinicializa o sistema.
 - **-c**: cancela uma solicitação prévia de shutdown.
 - **-h**: desliga o sistema.
 - Especificação de tempo e mensagem.
- Os comandos *reboot*, *halt*, *poweroff* e *telinit* também podem ser utilizados para reiniciar ou desligar o sistema.

Se um sistema Linux não é desligado corretamente, os buffers de disco, mantidos pelo kernel em memória, não serão atualizados no disco e, assim, os sistemas de arquivos podem ser danificados. Em adição, processos podem utilizar mecanismos semelhantes, mantendo em memória dados relevantes que, periodicamente, são armazenados em disco.

Nesses casos, o desligamento do sistema de forma abrupta provoca a perda desses dados. O processo de parada do sistema (shutdown) deve ser feito de forma organizada, para evitar que os sistemas de arquivos sejam corrompidos e os processos em execução percam dados relevantes mantidos em memória. O procedimento correto é executar o comando *shutdown*, que realiza a parada do sistema de modo planejado. Inicialmente, o processo shutdown envia o sinal TERM para todos os processos. No entanto, muitos processos não tratam o sinal TERM ou, quando o fazem, se cancelam no final desse procedimento. O shutdown assume que esses processos não mantêm dados relevantes em memória e finaliza-os, através do envio do sinal KILL. O comando *shutdown* possui algumas opções que podem ser informadas na linha de comando:

- **-r**: reinicializa o sistema (reboot);
- **-h**: desliga o sistema (halt);
- **-c**: cancela uma solicitação prévia de shutdown;
- **time**: especifica o momento em que o shutdown deve ser executado.
Pode ser especificado como tempo relativo (+min) ou absoluto (hh:mm);
- **message**: envia uma mensagem aos usuários logados no sistema.



Saiba mais

O systemd começou a ser testado no Fedora 15 e atualmente é a versão default para sistemas Red Hat Enterprise Linux 7 e derivados, substituindo o Upstart. Existem outros sistemas para inicialização, como o SysVinit paralelo e o OpenRC, entretanto, não serão abordados neste curso.



Além do comando *shutdown*, o administrador pode reinicializar o sistema utilizando o comando *reboot*, ou pará-lo usando os comandos *halt* ou *poweroff*. O administrador também pode utilizar o comando *telinit* para desligar ou reinicializar o sistema. Esse comando altera o nível de execução do sistema e recebe como parâmetro um número que pode variar entre 0 e 6, e representa o nível de execução para o qual o sistema será chaveado. Sendo a parada e a reinicialização do sistema representadas pelos níveis de execução 0 e 6, respectivamente, o administrador pode usar o comando *telinit*, passando como parâmetro os números 0 ou 6 para desligar ou reinicializar o sistema respectivamente.

Saiba mais

O tamanho do kernel depende do número de componentes selecionados, por isso, é recomendado que ele só ofereça suporte aos dispositivos instalados no sistema. Por exemplo, se o computador não possui placas de som, não há necessidade de habilitar drivers de placas de som no kernel. A inclusão de drivers desnecessários somente torna o kernel maior e mais lento.

Arquitetura do kernel

- Responsável por reconhecer dispositivos.
- Pode ser configurado para otimizar o uso da memória e melhorar o desempenho do sistema.
- Deve ser compilado após qualquer alteração de configuração.
- O conceito de módulos:
 - Carga de drivers sob demanda.
 - Kernel monolítico com componentes modulares.

O kernel é responsável por reconhecer os dispositivos instalados no sistema e carregar os drivers desses dispositivos, para que funcionem adequadamente. Ele pode ser configurado de tal forma que suporte apenas os dispositivos efetivamente instalados no sistema, otimizando o uso de memória e melhorando o desempenho do sistema. Para isso, o administrador deve definir os dispositivos e funcionalidades que devem ser habilitados ou não e compilar o kernel para que as alterações entrem em vigor. A partir da versão 2.0 do kernel, o conceito de módulos foi introduzido, permitindo a carga de drivers sob demanda. Assim, durante a inicialização do sistema, o kernel pode carregar apenas os drivers dos dispositivos detectados.

Kernel Monolítico

- É composto por um grande e único bloco de código.
- Carrega estaticamente os drivers de dispositivo definidos.
- Ocupa mais espaço da memória, uma vez que os drivers de todos os dispositivos configurados são carregados, independentemente de estarem instalados no sistema.
- Utilizado em Sistemas Operacionais como Linux, FreeBSD, versões antigas do Windows (95, 98 E ME) etc.

Como visto na sessão 1, o kernel do Linux possui uma arquitetura monolítica, sendo composto por um grande e único bloco de código. O kernel monolítico carrega estaticamente seus componentes, independentemente de estarem instalados no sistema e, por isso, ocupa mais espaço em memória. É comum carregar diversos drivers para uma determinada classe de dispositivo, deixando para o kernel a responsabilidade de selecionar o driver adequado durante a inicialização. Nesse caso, os demais drivers ainda são mantidos em memória. O kernel monolítico é utilizado em Sistemas Operacionais como Linux e FreeBSD (versões antigas), e em algumas versões do Windows (95, 98, ME) etc.

Kernel Modular ou Híbrido

- Carrega dinamicamente os drivers de dispositivo (módulos) detectados na inicialização do sistema.
- Ocupa menos espaço em memória.
- Os módulos são carregados sob demanda e podem ser removidos da memória após um intervalo de inatividade.



Utilizado em Sistemas Operacionais como Linux e FreeBSD, o conceito de módulos foi introduzido no Linux com o objetivo de otimizar o uso da memória, já que os módulos só são carregados em memória quando necessário. Além disso, o código do kernel se torna menor quando os componentes do sistema são compilados como módulos. Esses módulos são geralmente drivers de dispositivos e podem ser carregados em memória dinamicamente quando forem solicitados por algum dispositivo. Assim, podemos manter em um mesmo kernel componentes permanentemente habilitados de forma estática e componentes habilitados dinamicamente configurados como módulos.

Os módulos, apesar de não fazerem parte do mesmo código do kernel, são executados em seu espaço de memória (kernel-space). Dessa forma, o kernel monolítico, mesmo oferecendo suporte a módulos, continua sendo único e centralizado. Os módulos podem ser removidos da memória após um intervalo de tempo de inatividade.

Módulos do kernel

- São carregados dinamicamente e sob demanda em memória.
- Não fazem parte do código do kernel, mas são executados no kernel-space.
- Localizados em categorias no diretório `/lib/modules/versao_do_kernel`.
- Arquivo `/etc/modules`: relação de módulos que devem ser carregados durante a inicialização do sistema.



São pequenos pedaços do kernel que são compilados e instalados de forma independente. Os módulos do kernel ficam armazenados em subdiretórios localizados no diretório `/lib/modules/versao_do_kernel`, que são divididos em categorias como: drivers, net, sound etc. Os arquivos que representam cada módulo possuem uma nomenclatura característica que termina sempre com `.ko`. O arquivo `/etc/modules` possui os módulos que devem ser carregados durante a inicialização do sistema e mantidos carregados sempre que o sistema estiver em execução.

Comandos para gerenciar módulos:

- **kmod**: carrega em memória módulos automaticamente, sob demanda.
- **modprobe**: carrega ou remove da memória manualmente um módulo e suas dependências.
- **insmod**: carrega em memória um módulo, mas não carrega suas dependências.
- **rmmod**: remove um módulo da memória.
- **lsmod**: lista os módulos carregados em memória.
- **depmod**: verifica dependências entre módulos durante a inicialização do sistema.
- **modconf**: configura um módulo.
- **modinfo**: exibe informações sobre um módulo.



A seguir, serão apresentados os principais comandos para gerenciar os módulos do kernel.

- **kmod:** é utilizado para carregar os módulos do kernel automaticamente quando solicitados por algum processo ou dispositivo. O `kmod` é executado como daemon e fica monitorando o sistema de modo a ativar algum módulo quando necessário;
- **modprobe:** usado para carregar ou remover manualmente um módulo do kernel e suas dependências;
- **insmod:** utilizado para carregar um módulo do kernel manualmente. É importante frisar que o comando `insmod` não carrega as dependências do módulo que está sendo carregado;
- **rmmod:** é usado para remover um módulo do kernel carregado em memória;
- **lsmod:** para listar os módulos do kernel carregados em memória. São exibidos o nome do módulo, seu tamanho e o programa que está utilizando o módulo. O comando `lsmod` usa as informações presentes no arquivo `/proc/modules` para exibir sua saída;
- **depmod:** é utilizado durante a inicialização do sistema para verificar se existe alguma dependência entre os módulos do kernel, de modo que após a inicialização, todas as dependências estejam resolvidas;
- **modconf:** usado para configurar módulos através de uma interface baseada em menus. O comando `modconf` possibilita a passagem de parâmetros de configuração para um módulo, como IRQ, DMA etc.;
- **modinfo:** utilizado para exibir informações a respeito de um módulo, como sua localização em disco, tipo de licença, dependências, parâmetros etc.

Configurando e compilando o kernel

O processo de configuração e compilação do kernel é realizado em várias etapas, que serão descritas em detalhes a seguir.

Configurando os componentes do kernel

- Código-fonte armazenado no diretório `/usr/src/linux`.
- Arquivo `/usr/src/linux/.config`:
 - Variáveis que representam os componentes do kernel.
 - Pode ser alterado através de um editor de textos (`vi`, `emacs` etc.).
 - Também pode ser alterado através do comando `make` (`make config`, `make menuconfig` ou `make xconfig`).

O código-fonte do kernel fica localizado por padrão no diretório `/usr/src/linux`, que é apenas um link simbólico para o diretório `/usr/src/linux-x.y.z`, onde `x.y.z` indica a versão do kernel. Nesse diretório, o arquivo `.config` contém as informações sobre a configuração dos componentes do kernel. Esse arquivo possui diversas variáveis que definem se um determinado componente será estático, quando o valor da variável é `"y"`, ou modular, quando o valor da variável é `"m"`. Um componente pode ser desabilitado se a variável que o representa estiver comentada. O arquivo `/usr/src/linux/.config` pode ser modificado diretamente através de um editor de texto. No entanto, o comando `make` pode ser usado para alterar o arquivo `.config`, simplificando a configuração dos componentes do kernel. O comando `make` pode ser utilizado de três formas para configurar o kernel. A primeira delas é usando o `targetconfig`, conforme é mostrado a seguir:

```
# make config
```



Esse comando executa, em modo texto, um script que solicita a configuração de cada componente do kernel. Os componentes são apresentados linha a linha, obrigando o administrador a configurar todos eles, um de cada vez. O segundo método para configurar o kernel é através do target menu config, conforme apresentado a seguir:

```
# make menu config
```

Esse comando executa, ainda em modo texto, um script baseado em menus, que permite ao administrador configurar os componentes do kernel. Os componentes são apresentados em menus divididos por categorias e podem ser selecionados de acordo com a necessidade do administrador. Os componentes podem ser incluídos como módulos ou integrados ao kernel, ou ainda excluídos, de acordo com a legenda apresentada na interface. Assim, como no método anterior, esse método não requer a existência de um ambiente gráfico. Por fim, um terceiro método para configurar o kernel é através do targetxconfig, conforme apresentado a seguir:

```
# make xconfig
```

Esse método baseia-se no ambiente gráfico X-Window System, tornando a configuração dos componentes bastante intuitiva. Se o administrador possui um ambiente gráfico instalado no seu sistema, esse é o método mais indicado para se configurar o kernel. Nele, os componentes são listados em diferentes níveis de menu e podem ser selecionados através do mouse. Os componentes podem ser incluídos como módulos ou integrados ao kernel, ou ainda excluídos, de acordo com a legenda apresentada na interface. Após a configuração dos componentes, deve-se clicar no botão “Save and Exit” para criar ou atualizar o arquivo de configuração `/usr/src/linux/.config` e sair do programa de configuração do kernel.

Opções de componentes:

- Enable loadable module support
- General setup
- Character devices
- Block devices
- Networking options
- SCSI device support
- Network device support
- Filesystems



Existem diversos componentes que podem ser configurados no kernel. Nos métodos `make menuconfig` e `make xconfig`, a opção `help` pode ser utilizada para mostrar detalhes sobre o significado de cada componente. A seguir, serão comentadas apenas as principais opções de componentes, organizadas de acordo com os menus mostrados nos métodos `make menuconfig` e `make xconfig`.

- **Enable Loadable module support:** permite a configuração de características dos módulos. A opção `Enableloadable module support` habilita a utilização de componentes modulares no kernel. Se essa opção for desabilitada, não será possível habilitar componentes como módulos;
- **General setup:** o conjunto de opções apresentado permite configurar componentes de baixo nível. Por exemplo, a opção `Networking support` habilita a configuração dos módulos de rede, e a opção `Bus options` inclui os módulos que permitem ao kernel funcionar em plataformas com barramento PCI;

- **Character devices:** as opções disponíveis permitem configurar suporte a terminais virtuais, portas seriais, portas paralelas etc.;
- **Block devices:** as opções apresentadas permitem a configuração de diferentes tipos de discos rígidos, CD-ROMs, unidades de fita magnética etc.;
- **Networking options:** O conjunto de opções está relacionado com a habilitação de protocolos de redes e suas funcionalidades. Por exemplo, as opções TCP/IP networking e The IPX protocol habilitam a família de protocolos TCP/IP e o protocolo IPX de redes Novell, respectivamente. Para a família de protocolos TCP/IP, é possível habilitar funcionalidades como: multicast, roteamento avançado etc.;
- **SCSI devicesupport:** as opções apresentadas permitem a configuração de diferentes tipos de dispositivos SCSI, como discos rígidos, CD-ROMs e unidades de fita magnética;
- **Network devicesupport:** permite habilitar suporte a diversas tecnologias de rede, como: ethernet, WLAN, PPP, WiMAX, FDDI, ATM, FibreChannel etc.;
- **Filesystems:** o conjunto de opções permite a configuração do sistema de quotas e o suporte aos diferentes tipos de filesystems (ext2, ext3, ext4, Raiserfs, XFS, NTFS etc.).

Compilando o kernel

Após configurar os componentes do kernel, o administrador deve compilá-lo para que as alterações possam ser efetivadas. Para compilar o kernel, é preciso manter como diretório corrente o “/usr/src/Linux” e em seguida executar os comandos a seguir:

```
# make clean
# make bzImage
```

O comando *make clean* remove arquivos existentes, decorrentes de compilações anteriores do kernel, e o comando *make bzImage* gera a imagem do kernel comprimida no arquivo */usr/src/linux/arch/i386/boot/bzImage*. O comando a seguir também pode ser utilizado para compilar o kernel, só que, nesse caso, a imagem criada é também armazenada em um disquete de boot, que pode ser utilizado para testar o novo kernel.

```
# make bzdisk
```

Em caso de falha na inicialização com o novo kernel, utilizando o disquete de boot, basta retirá-lo e reinicializar o sistema.

Componentes modulares

Compilando o kernel:

- Ir para o diretório “/usr/src/Linux”.
- Comandos para compilação:
 - # make clean
 - # make bzImage
- Compilação e instalação dos módulos:
 - # make modules
 - # make modules_install

Se o novo kernel tem componentes modulares em sua configuração, o administrador precisa, então, compilar e instalar esses módulos usando os comandos *make modules* e *make modules_install*, respectivamente.



```
# make modules  
# make modules_install
```

O tempo gasto no processo de compilação do kernel varia de acordo com o hardware usado e geralmente não dura mais do que alguns minutos. É importante ressaltar que a inclusão de novos componentes modulares em um kernel operacional não requer a sua compilação. O administrador precisa apenas definir a configuração dos componentes utilizando os comandos *make menu config* ou *make xconfig* e, em seguida, executar os comandos *make modules* e *make modules_install*.

Instalando o kernel

```
# make install
```

- Ativa script `/usr/src/linux/arch/x86/boot/install.sh`.
- Copia imagem do novo kernel e arquivos associados.

```
# mkinitramfs 3.2.8 -o /boot/initrd.img-3.2.8
```

- Cria a imagem `initrd` no diretório `"/boot"`.

Arquivo de configuração do gerenciador de boot deve ser atualizado com as novas imagens.

A instalação do novo kernel é feita com o comando *make install*. Até certo tempo atrás, cabia ao administrador fazer os arranjos finais para a instalação do novo kernel. Em primeiro lugar, ele deveria copiar o arquivo *vmlinuz*, armazenado no diretório `"/usr/src/linux/arch/i386/boot"`, para o diretório `"/boot"`, colocando no arquivo um nome diferente daquele do kernel original. Por exemplo, supondo que o nome do arquivo do kernel original fosse *vmlinuz-3.2.8*, o nome do arquivo do novo kernel poderia ser *vmlinuz-3.2.8.new*. Além disso, ele teria de fazer as modificações necessárias no arquivo de configuração do carregador de boot. Nas versões mais recentes do Linux, no entanto, é suficiente a execução do comando:

```
# make install
```

Esse comando executa o script `/usr/src/linux/arch/x86/boot/install.sh`, que copia a imagem do novo kernel e os arquivos associados para o diretório indicado, geralmente, o `"/boot"`. Em seguida, é preciso gerar a imagem `initrd` utilizando o comando a seguir:

```
# mkinitramfs 3.2.8 -o /boot/initrd.img-3.2.8
```

Em outras distribuições, a imagem `initrd` pode ser gerada com o comando *mkinitrd*. Após isso, o arquivo de configuração do gerenciador de boot deve ser atualizado com as novas imagens.

Habilitando o kernel

- Verifica a da instalação por parte do administrador:
 - LILO: `/etc/lilo.conf`.
 - GRUB: `/boot/grub/grub.cfg`.
- Manutenção do kernel original para se proteger de erros.
- Se o carregador de boot for o LILO, é necessário atualizar o MBR:
 - Executar o comando *lilo -v*

Embora os comandos executados no passo anterior já tratem de fazer as configurações necessárias para o sistema inicializar utilizando o novo kernel gerado, é importante que o administrador se certifique de que as modificações foram feitas de forma correta.

Na instalação de um novo kernel, é interessante que o original seja mantido como proteção contra possíveis erros que possam ocorrer durante a inicialização com o novo kernel. Essa proteção pode ser obtida na configuração do seu carregador de boot.

Apresentamos a seguir os arquivos de configuração do LILO e do GRUB, antes e após a instalação de um novo kernel. Assim, consideramos que o arquivo */etc/lilo.conf* possui, originalmente, o seguinte conteúdo:

```
prompt
timeout=50
default=Linux
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
message=/boot/message
1ba32
image=/boot/vmlinuz-3.2.8
label=Linux
initrd=/boot/initrd-3.2.8.img
read-only
append="root=LABEL="/>
```

As cinco últimas linhas formam a seção associada à versão original do kernel, que usa o arquivo *vmlinuz-3.2.8* e possui o rótulo "Linux". Esse rótulo é exibido durante a inicialização do sistema para que o administrador possa informar ao LILO o Sistema Operacional que deve ser ativado. Após a instalação, o administrador deve verificar se foi incluída uma nova seção ao final do arquivo para descrever o novo kernel, conforme mostrado a seguir:

```
prompt
timeout=50
default=Linux
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
message=/boot/message
1ba32
# Kernel novo
image=/boot/vmlinuz-3.2.8custom
label=Linux-3.2.8custom
```



```

initrd=/boot/initrd-3.2.8custom.img
read-only
append="root=LABEL=/"
# Kernel antigo
image=/boot/vmlinuz-3.2.8
label=Linux
initrd=/boot/initrd-3.2.8.img
read-only
append="root=LABEL=/"

```

Nota-se que foi criada uma nova entrada no arquivo, mas foi mantida a versão antiga do kernel como padrão do sistema. Isso permite que o administrador teste a nova versão e só a defina como padrão após verificar o seu funcionamento. Há duas formas de se alterar a versão padrão: a primeira delas é modificando o valor da variável default para o label do novo kernel, e a segunda é trocar o valor da variável label dos dois kernels, um pelo outro. Em seguida, é necessário atualizar o LILO, executando o comando a seguir:

```
# lilo -v
```

No caso do GRUB, vamos assumir que o arquivo de configuração, originalmente, tivesse o seguinte conteúdo:

```

default=0
timeout=30
splashimage=(hd0,0)/GRUB/splash.xpm.gz
title Red Hat Linux
root (hd0,0)
kernel /boot/vmlinuz-3.2.8 ro root=/dev/hda3
initrd /boot/initrd-3.2.8.img

```

Após a instalação do novo kernel, o arquivo deve ficar com o seguinte conteúdo:

```

default=1
timeout=30
splashimage=(hd0,0)/GRUB/splash.xpm.gz
title Red Hat Linux (custom)
root (hd0,0)
kernel /boot/vmlinuz-3.2.8custom ro root=/dev/hda3
initrd /boot/initrd-3.2.8custom.img
title Red Hat Linux
root (hd0,0)

```

```
kernel /boot/vmlinuz-3.2.8 ro root=/dev/hda3  
initrd /boot/initrd-3.2.8.img
```

Novamente, o script cria uma entrada para o novo kernel, mas não altera o kernel padrão. Entretanto, diferente do LILO, não é preciso executar nenhum comando após alterar seu arquivo de configuração.

Testando o kernel

- Reinicializar o sistema utilizando o novo kernel.
- Em caso de problemas:
 - ▣ Verificar se o LILO foi atualizado com o comando *lilo -v*.
 - ▣ Reativar kernel antigo.
- Se tudo estiver certo:
 - ▣ Remover entrada do antigo kernel do arquivo de configuração do carregador de boot.
 - ▣ Remover a imagem do antigo kernel e a imagem antiga initrd.

Depois de feitas as devidas verificações, o sistema pode ser reinicializado utilizando o novo kernel. Se o sistema não inicializar ou apresentar problemas enquanto estiver operacional, o administrador poderá reiniciá-lo com o kernel antigo, selecionando-o, durante a inicialização. No momento em que o novo kernel mostrar-se estável, basta remover a sessão de aprendizagem do arquivo */etc/lilo.conf* ou */boot/grub/grub.cfg* associada ao kernel antigo e remover sua imagem do diretório */boot*.



9

Segurança básica e procedimentos operacionais

objetivos

Conhecer os aspectos de segurança básica do sistema; Entender a importância da monitoração do sistema e da antecipação de ações; Compreender como funciona o esquema de autenticação de usuários em um sistema Linux e o princípio de funcionamento dos algoritmos de criptografia do tipo one-way; Entender a utilidade dos bits SUID e SGID e os riscos que trazem à segurança do sistema; Conhecer os principais gerenciadores de pacotes do Linux; Aprender alguns procedimentos operacionais que devem ser adotados, como: estabelecimento de uma política de uso de recursos computacionais, registro de ocorrências, rotina de backup etc.

Serviço de impressão; Segurança básica; Analisadores de senhas; Contas compartilhadas; SUID e SGID; Atualização de software; Procedimentos operacionais; Política de utilização de recursos; Atendimento a usuários; Controle de recursos computacionais; Diagnóstico de falhas; Rotina de backup.

conceitos

Segurança básica

Problemas básicos de segurança:

- Utilizar ferramentas que suportem criptografia.
- Não ativar serviços de rede desnecessários.
- Configurar corretamente os arquivos `/etc/hosts.allow` e `/etc/hosts.deny`.
- Utilizar um firewall para controlar o acesso à rede e um proxy para controlar o acesso à web.

Questões básicas:

- Orientar os usuários a utilizarem senhas não triviais.
- Controlar o acesso físico aos servidores.
- Controlar a criação e a remoção de contas.
- Impedir o uso de contas compartilhadas.

Os aspectos sobre segurança básica e procedimentos operacionais que serão tratados nesta sessão de aprendizagem dizem respeito apenas àqueles que têm alguma correlação com o que foi visto nas sessões de aprendizagem anteriores. Alguns aspectos, como o uso



de senhas não triviais, o uso de contas compartilhadas e a remoção de contas desativadas, por exemplo, já foram discutidos em sessões anteriores. Outros aspectos, no entanto, ainda não foram mencionados, como o uso de criptografia, os serviços de rede desnecessários etc. Esses tópicos serão tratados com mais detalhes nos módulos de administração mais avançados. Ainda nesse tópico, será ressaltada a importância da monitoração do sistema e da antecipação de ações. Algumas questões básicas de segurança, com as quais o administrador de sistemas deve se preocupar sempre, independentemente do tamanho de sua rede, são apresentadas a seguir.

- Orientar os usuários de modo que não utilizem senhas óbvias e configurar os sistemas de modo que solicitem trocas de senhas periodicamente;
- Manter o sistema em um local fisicamente seguro. De nada adianta proteger o sistema contra invasões através da rede e mantê-lo em um ambiente cujo acesso não seja controlado;
- Não criar contas de usuários desnecessárias e sempre excluir as contas que não estejam mais sendo utilizadas. Contas fantasmas podem se transformar em um canal para crackers;
- Evitar compartilhar contas de usuários comuns e, principalmente, a conta do usuário root, facilitando a monitoração das atividades do sistema;
- Sempre que possível, utilizar ferramentas que suportem criptografia para transmitir e receber dados através da rede. O SSH (Secure Shell) é um exemplo de programa que criptografa os dados antes de transmiti-los através da rede;
- Evitar a ativação de serviços de rede desnecessários. Se um determinado sistema não é o servidor de e-mail da instituição, não há motivos para ativar o daemon *postfix*, por exemplo. Cada serviço disponibilizado por um servidor é uma possível porta de invasão para os crackers;
- Configurar corretamente os arquivos */etc/hosts.allow* e */etc/hosts.deny* (discutidos no módulo de introdução à arquitetura TCP/IP para a plataforma Linux);
- Utilizar firewalls para controlar o acesso aos diversos equipamentos da rede e prover o serviço de proxy para controlar o acesso dos usuários à web. Esses dois serviços ajudam a manter a rede mais segura. Nos demais módulos de administração de sistemas Linux, serão discutidos outros aspectos de segurança relativos a redes e serviços.

Difícultar ao máximo a ação dos crackers.

- Chkrootkit:
 - ▣ Utilizado para detectar rootkits.
 - ▣ Analisa o sistema em busca de executáveis, serviços e processos suspeitos.
- TripWire:
 - ▣ Monitora o sistema para garantir sua integridade.
 - ▣ Utiliza diversas técnicas para encontrar alterações não autorizadas em arquivos.
- OpenVAS:
 - ▣ Utilizado para detectar vulnerabilidades.

Nenhum sistema computacional está livre de ataques de crackers. O melhor que pode ser feito é dificultar ao máximo suas ações e fazer uma monitoração permanente para procurar detectar rapidamente as tentativas de invasão. Os administradores de sistemas devem usar alguns programas para descobrir e corrigir as possíveis falhas de segurança.



- **Chkrootkit:** é um programa utilizado para detectar rootkits em sistemas Unix. Um rootkit tem como finalidade ocultar uma invasão em um sistema. Para isso, ele faz diversas modificações, como substituição de executáveis, remoção de entradas do arquivo *wtmp*, entre outras, sempre com o objetivo de realizar ataques a terceiros. Para combater esse tipo de ameaça, foi desenvolvido o chkrootkit, que faz uma série de verificações, procurando por módulos de kernel maliciosos, arquivos executáveis, serviços e processos suspeitos, entre outros, no intuito de detectar algum indício de invasão ao sistema. A versão atual do chkrootkit detecta cerca de 60 tipos de ameaças;
- **TripWire:** é um programa utilizado para monitorar a integridade de sistemas Unix. O TripWire pode ser configurado para monitorar arquivos e diretórios e enviar alertas caso ocorra alguma modificação não autorizada. São utilizadas diversas rotinas de checksum, message-digest, secure-hash e assinaturas com o intuito de detectar alterações no conteúdo e nas permissões dos arquivos e diretórios monitorados, além de remoção e inclusão de novos arquivos e/ou diretórios;
- **OpenVAS (Open Vulnerability Assessment System):** É um framework para diversos serviços e ferramentas que oferece um poderoso scanner de vulnerabilidade e um gerenciador de soluções que auxilia o administrador a corrigir as vulnerabilidades identificadas. O OpenVAS utiliza a arquitetura cliente-servidor onde o servidor executa uma varredura no cliente, em portas TCP e UDP pré-configuradas, em busca de alguma vulnerabilidade. Mesmo que um serviço não esteja utilizando sua porta padrão ele será examinado pelo OpenVAS, desde que, a porta em que ele esteja sendo executado tenha sido configurada para ser analisada. O administrador pode definir quais clientes serão auditados e após a conclusão da auditoria, é apresentado um relatório com todas as vulnerabilidades encontradas em todos os clientes escaneados. Hoje, o OpenVAS possui uma comunidade crescente, com contribuições de indivíduos e corporações de todo o mundo e está disponível na forma de pacotes binários para a maioria das distribuições Linux e é possível obtê-lo em repositórios de terceiros ou baixá-lo diretamente do site do OpenVAS.

Analísadores de senhas

- Uso da criptografia *one-way*.
- A senha do usuário é criptografada e comparada com a que está armazenada.
- Uso de dicionários próprios para obrigar os usuários a escolherem senhas difíceis.
- Senhas simples podem ser quebradas por crackers com o uso de dicionários.
- Alteração frequente da senha do usuário root.
- Definir os terminais seguros no arquivo */etc/securetty*.

A autenticação do usuário é baseada na comparação da senha fornecida no login com aquela armazenada no arquivo */etc/passwd* ou */etc/shadow*, caso as *shadow passwords* estejam habilitadas. Em qualquer um desses dois arquivos, as senhas são armazenadas criptografadas por algoritmos de criptografia que estão publicamente disponíveis. No entanto, a segurança não é comprometida, uma vez que esses algoritmos são do tipo *one-way* e, por isso, não permitem decifrar a senha criptografada. O Linux utiliza algoritmos *one-way*, já que as senhas nunca precisam ser decifradas. Na prática, a senha fornecida pelo usuário é criptografada e comparada com a senha criptografada armazenada. Se essas senhas são idênticas, o acesso ao sistema é permitido. Assim, as senhas são armazenadas de forma segura e, caso uma senha seja quebrada, não será por falha do algoritmo de criptografia, mas do usuário, que utiliza uma senha fácil de ser quebrada. Esse é o motivo pelo qual não se deve utilizar senhas simples.



O uso de senhas óbvias, como datas de aniversário, nomes de membros da família etc. é uma das principais falhas de segurança a que os sistemas estão sujeitos. Uma senha segura deve conter caracteres maiúsculos e minúsculos, números e ainda símbolos especiais. Isso dificulta muito a tarefa dos crackers, que geralmente utilizam um arquivo contendo diversas senhas, chamado dicionário, para tentar quebrar senhas de usuários.

A melhor maneira de evitar que os usuários do sistema escolham senhas fáceis de serem quebradas é empregar a mesma técnica utilizada pelos crackers, configurando os sistemas de modo que as senhas sejam confrontadas contra um dicionário antes de serem efetivamente armazenadas. Caso a senha informada esteja cadastrada no dicionário, o sistema envia uma mensagem ao usuário informando que a senha escolhida é muito fácil de ser quebrada.

Uma vulnerabilidade é extremamente grave quando sua exploração permite ao cracker invadir o sistema obtendo os privilégios do usuário root. Esse tipo de vulnerabilidade deve ser identificado e resolvido com a máxima prioridade. Para minimizar as chances de crackers descobrirem a senha do usuário root, o administrador precisa mudá-la periodicamente. Outro cuidado adicional é manter no arquivo `/etc/passwd` apenas os nomes dos terminais seguros a partir dos quais o usuário root pode fazer login no sistema. Para os demais terminais, somente será possível ter privilégios de root primeiramente acessando o sistema como um usuário convencional e depois se tornar root através do comando `su`.

Dicionários

Nome dado a arquivos que contêm as senhas mais comuns utilizadas por usuários de sistemas. Os crackers usam esses arquivos como base para tentativas de acesso indevidas a sistemas de terceiros. São utilizados scripts que tentam realizar login em um determinado sistema utilizando para cada conta de usuário do sistema, todas as senhas presentes no dicionário.

Contas compartilhadas

- Utilizadas por diversos usuários.
- Ajudam a poupar tempo e evitam a criação de contas desnecessárias.
- Uso não recomendado.
- Mau uso dificulta a identificação do culpado.
- O mesmo se aplica à conta root.

Contas compartilhadas são aquelas usadas por vários usuários para um determinado fim. Essas contas são criadas para evitar que várias contas sejam criadas desnecessariamente. Por exemplo, uma empresa está fornecendo um curso de 15 dias para 10 pessoas e o uso dos recursos computacionais é essencial. Nesse caso, geralmente, o administrador cria uma única conta, e os alunos compartilham a senha dessa conta. No entanto, esse tipo de compartilhamento não é recomendável, pois caso ocorra uma invasão utilizando essa conta ou qualquer outro problema que comprometa o sistema, o administrador não terá como identificar o usuário que direta ou indiretamente causou o problema e atribuir-lhe as penalidades cabíveis. Uma sugestão de solução para esse caso seria o administrador criar 10 novas contas com prazo de expiração para o final do último dia do curso, apesar de essa alternativa ser mais trabalhosa. O compartilhamento da senha do usuário root é ainda mais perigoso e deve ser restringido ao máximo.



Saiba mais

Em vez de dicionários, o administrador pode também definir políticas de senhas, como exigir que todas as senhas tenham caracteres maiúsculos, minúsculos, números e caracteres especiais.



SUID e SGID



SGID:

- Usuários são organizados em grupos.
- Pode resolver problemas de permissão em arquivos compartilhados criados por usuários de diferentes grupos.

SGID:

- O uso de SUID em arquivos do usuário root é potencialmente perigoso e deve ser evitado.

Serviços devem ser executados por um usuário com os privilégios necessários.

Como foi descrito na sessão de aprendizagem 2, os usuários podem ser distribuídos em grupos e, dessa forma, compartilhar arquivos. Todo usuário possui um grupo primário, o que pode causar problemas caso esse compartilhe arquivos com integrantes de dois ou mais grupos. Por exemplo, os funcionários do departamento de RH de uma determinada empresa pertencem ao grupo “rh” e possuem um diretório no sistema com permissão exclusiva para esse grupo.

Entretanto, o gerente do departamento de RH pertence aos grupos “gerencia” e “rh”, sendo “gerencia” seu grupo primário. Quando o gerente do departamento cria um arquivo no diretório do grupo “rh”, o grupo proprietário do arquivo será o grupo “gerencia”. Assim, dependendo das permissões do arquivo, nenhum outro usuário do grupo “rh” poderá acessá-lo. A permissão SGID pode ser definida nesse diretório para solucionar o problema. Essa permissão é habilitada com o comando *chmod* e, quando habilitada em um diretório, força todos os arquivos e diretórios criados nesse diretório a terem o mesmo grupo proprietário do diretório.

Já o bit SUID é utilizado em programas que precisam ser executados com as permissões de seu dono. Para reduzir as possibilidades de falhas de segurança do sistema, somente devem ser executados com privilégios de root os serviços em que esse requisito for obrigatório. Cada serviço habilitado representa uma possibilidade a mais para a exploração de falhas. Se houver uma falha de segurança em um processo sendo executado com privilégios de root, após a invasão o cracker herdará esses privilégios e terá acesso irrestrito ao sistema. Para resolver esse problema, cada serviço deve ser executado em nome de um usuário que tenha unicamente os privilégios necessários para acessar os recursos utilizados pelo serviço.

Alguns usuários padrão, como *daemon*, *nobody*, *shutdown*, *ftp*, *games*, *mail* e *news*, são criados na instalação do sistema com esse objetivo. Um servidor web, por exemplo, precisa acessar apenas o conjunto de arquivos e recursos necessários à sua operação. Portanto, seus processos devem ser executados com privilégios de um usuário convencional com acesso somente a esses arquivos e recursos. Geralmente, os processos executados por um servidor web tem o usuário *nobody* como proprietário.

Atualização de software

Os softwares instalados devem ser atualizados regularmente, visando a expansão de funcionalidades e a correção de falhas de funcionamento detectadas nas versões em uso. Os desenvolvedores também utilizam as atualizações para corrigir falhas de segurança encontradas nos seus pacotes de software, que comprometem a segurança do sistema. O procedimento de atualização deve ser realizado com muito cuidado, para evitar mudanças não desejadas, que venham a comprometer o funcionamento do sistema. A maioria das distribuições usa gerenciadores de pacotes, o que facilita bastante o processo de instalação, remoção e atualização de software no sistema. A seguir, serão descritos três dos principais gerenciadores de pacotes do Linux.



RPM: RPM Package Manager

- Gerenciador de pacotes utilizados nas distribuições Red Hat, Fedora, CentOS etc.
- Os pacotes são compostos por binários, arquivos de configuração etc.
- Possui interfaces gráficas e uma interface texto baseada no comando *rpm*.
- Utiliza criptografia e funções de hash para garantir a integridade dos pacotes.
- Verifica dependências entre pacotes, mas não as resolve.
- Não fazem atualizações automáticas de pacotes.



O RPM Package Manager (RPM) é um gerenciador de pacotes que oferece uma maneira fácil e segura de instalar, remover e atualizar pacotes de software. Além disso, o RPM mantém informações sobre a versão atual dos pacotes instalados em uma base de dados própria. Os pacotes RPM são pré-compilados e contêm binários, arquivos de configuração, páginas de manual etc. Red Hat, Fedora e CentOS são exemplos de distribuições que utilizam pacotes RPM. O RPM possui interfaces gráficas como, por exemplo, o Gnome-RPM e uma interface texto baseada no comando *rpm*. Esse curso será baseado na interface texto utilizando o comando *rpm*. São listadas a seguir as principais opções de uso desse comando.

- **-i**: instala um ou mais pacotes;
- **-e**: remove um ou mais pacotes;
- **-U**: atualiza um ou mais pacotes;
- **-q**: faz uma consulta na base de dados criada pelo RPM no sistema.

O RPM possui métodos baseados no algoritmo MD5 e na criptografia de chave pública, utilizando GPG, para validar a integridade de um pacote. Isso é necessário, pois alguns pacotes podem estar corrompidos.

Antes de instalar um pacote, o RPM verifica se este já está instalado no sistema e, caso esteja, verifica se sua versão é mais recente que a versão que está sendo instalada. Em seguida, o RPM confere as dependências, cancelando a instalação, se algum outro pacote requerido não estiver instalado. É importante frisar que o RPM não resolve problemas de dependência entre pacotes. Ele somente indica as dependências, não as instalando automaticamente. Nesse caso, o administrador deve, primeiramente, instalar todas as dependências necessárias. No entanto, se o administrador tentar remover um pacote que é dependente de outro pacote, a remoção não será realizada e uma mensagem será impressa na tela informando que o pacote não pode ser removido porque é utilizado por outro pacote que está instalado no sistema. O RPM também verifica se algum arquivo de configuração será modificado durante o processo de instalação de um pacote. Havendo a necessidade de modificação de um arquivo existente, o RPM salva a versão corrente e imprime uma mensagem na tela informando sobre a alteração.

Periodicamente, novas versões de pacotes RPM são lançadas para corrigir bugs e problemas de segurança. O administrador deve verificar, com frequência, na página da distribuição utilizada, o lançamento de novas versões de pacotes. Na verdade, os sistemas baseados em pacotes RPM, como Red Hat, CentOS e Fedora, possuem um mecanismo ainda mais poderoso para manter o sistema sempre atualizado. Trata-se do YUM, que é um gerenciador de pacotes baseado no RPM, que será descrito a seguir.

YUM: Yellowdog Updater, Modified



- Gerenciador de pacotes usados nas distribuições Red Hat, Fedora, CentOS etc.
- Resolve problemas com dependências, instalando-as automaticamente.
- Utiliza repositórios para fazer o download dos pacotes.
- Possui interfaces gráficas e uma interface texto baseada no comando *YUM*.

O Yellowdog Updater, Modified (YUM) é um gerenciador de pacotes usados em distribuições que são baseadas no RPM. Entre essas distribuições, podemos destacar: Red Hat, Fedora e CentOS. O YUM resolve problemas com dependências entre pacotes, instalando-as automaticamente quando necessário. Essa característica facilita bastante o trabalho do administrador durante o processo de instalação de pacotes. O YUM consulta diversos repositórios que são utilizados para fazer o download dos pacotes que são instalados no sistema. A lista de repositórios fica armazenada em arquivos dentro do diretório */etc/yum.repos.d/*. O YUM possui interfaces gráficas, como, por exemplo, o KYUM e uma interface texto baseada no comando *YUM*. Esse curso será baseado na interface texto utilizando o comando *YUM*. São listadas a seguir as principais opções de uso desse comando.

- **install**: instala um ou mais pacotes;
- **group-install**: faz a instalação de um grupo de programas (exemplo: "System Tools");
- **remove**: remove um ou mais pacotes;
- **update**: atualiza um ou mais pacotes;
- **upgrade**: faz a atualização de todos os pacotes e remove pacotes obsoletos;
- **clean**: limpa o cache do YUM;
- **search**: faz uma busca por um determinado pacote;
- **list**: faz uma busca por um determinado pacote, mas retorna menos informações que a opção **search**.

APT: Advanced Packaging Tool



- Gerenciador de pacotes utilizados nas distribuições Debian, Ubuntu etc.
- Resolve problemas com dependências, instalando-as automaticamente.
- Utiliza repositórios para fazer o download dos pacotes.
- Possui interfaces gráficas e interfaces texto, uma delas, baseada no comando *apt-get*.

É um gerenciador de pacotes usado na distribuição Debian e suas derivadas, como, por exemplo, a distribuição Ubuntu. Assim como o YUM, o APT também resolve problemas com dependências entre pacotes, instalando-as automaticamente quando necessário. O APT consulta uma série de repositórios que são usados para fazer o download dos pacotes que são instalados no sistema. A lista de repositórios fica armazenada no arquivo */etc/apt/sources.list*. O APT possui interfaces gráficas, como, por exemplo, o Synaptic e interfaces texto, com destaque para o comando *apt-get*. Este curso será baseado na interface texto, utilizando o comando *apt-get*. São listadas a seguir as principais opções de uso desse comando.

- **install**: instala ou atualiza um ou mais pacotes;
- **update**: atualiza a lista de repositórios definidos no sistema;
- **upgrade**: atualiza todos os pacotes que estão desatualizados;



- ▣ **dist-upgrade**: atualiza o sistema para uma nova distribuição;
- ▣ **remove**: remove um ou mais pacotes;
- ▣ **clean**: remove todos os pacotes baixados, utilizados em instalações anteriores;
- ▣ **source**: faz o download do código-fonte dos pacotes.

Procedimentos operacionais

Atividades do administrador de sistemas:

- ▣ Manter os sistemas atualizados, seguros e operacionais.
- ▣ Implementar as políticas de uso de recursos.
- ▣ Prestar atendimento aos usuários.
- ▣ Trabalhar de forma proativa.
- ▣ Registrar informações relevantes, independente da forma como é realizado o registro.
- ▣ Etc.



Como foi visto no item sobre segurança básica, o trabalho do administrador de sistemas abrange desde atividades rotineiras e bem definidas, como fazer cópias de segurança (backups), até atividades eventuais e imprevisíveis, como diagnosticar a causa do baixo desempenho de um servidor. Uma característica comum a todas essas atividades é que elas afetam, direta ou indiretamente, a produtividade dos usuários do sistema, e consequentemente a produtividade da instituição a que o sistema atende.

Desse modo, o administrador de sistemas precisa utilizar sempre as práticas de administração mais adequadas para os objetivos propostos pela instituição. Tal necessidade pressupõe que o administrador conheça os objetivos da instituição e de seus usuários, os sistemas que administra e sua área de atuação. Além disso, deve ser capaz de trabalhar em equipe, de forma responsável e organizada.

Focaliza-se, aqui, o último dos atributos relacionados acima, no que diz respeito ao trabalho de forma organizada, reunindo algumas recomendações e propondo modelos de procedimentos, normas e formulários para apoiar o administrador em suas atividades. Administradores que conseguem organizar seu trabalho são muito mais produtivos e efetivos em suas ações. Isso resulta em sistemas com maior tempo de disponibilidade, melhor desempenho e, o que é o mais importante, com usuários satisfeitos. Quando o administrador não consegue se organizar, ele opera de forma reativa e o resultado é um interminável “apagar de incêndios”.

O que está aqui descrito não deve ser entendido como a palavra final sobre as práticas de administração de sistemas, mas como um ponto de partida para ser adaptado às condições locais de cada instituição e aperfeiçoado pelo administrador no seu dia a dia. Inicialmente, o administrador deve utilizar somente o que lhe parecer mais útil e introduzir as demais práticas à medida que perceber sua necessidade. O registro dos dados sugeridos nas práticas aqui descritas pode ser feito tanto em meio eletrônico quanto impresso, com um layout requintado ou em estilo checklist. Esse não é o foco da discussão, pois não importa se o administrador vai usar caneta e papel, editor de textos, planilhas, banco de dados ou alguma ferramenta especializada. O importante é criar o hábito de registrar, de forma organizada e sistemática, as intervenções realizadas no sistema.

Política de utilização de recursos

Entre as atribuições do administrador de sistemas, encontra-se a elaboração de regras de utilização dos recursos computacionais disponibilizados para os funcionários de uma instituição. Esse ponto é especialmente crítico em grandes instituições, com unidades fisicamente distribuídas, mas que compartilham o uso de recursos. As regras de uso dependem fundamentalmente do tipo de instituição, do seu grau de informatização e da sua política de utilização de recursos computacionais.

Quanto à utilização de sistemas corporativos, cabe ao administrador levantar as informações relativas ao item “Caracterização do uso”, mostradas na tabela 9.1. Com base nessas condições, o administrador de sistemas deve definir os requisitos técnicos a serem observados para que os funcionários obtenham acesso aos recursos computacionais da instituição. É ideal que o administrador de sistemas proceda à inclusão do funcionário no sistema somente se essas condições forem atendidas. Na prática, nem sempre as instituições possuem tal grau de organização, e o resultado é a degradação do desempenho do sistema e a insatisfação generalizada, tanto dos usuários como da equipe técnica.

O uso dos recursos computacionais disponibilizados pela instituição para seus funcionários e colaboradores deve ser feito única e exclusivamente com o objetivo de realizar atividades pertinentes ao cargo e à função do usuário, em acordo com os objetivos e a missão da instituição, respeitando as leis de direitos autorais e de propriedade intelectual aplicáveis aos softwares utilizados, dados e informações armazenados ou acessíveis por meio daqueles recursos.

Entende-se por recursos computacionais o conjunto de ativos de TI, formados por servidores, desktops, laptops, equipamentos de conectividade, dispositivos de hardware, cabeamento, softwares diversos, links de comunicação de dados, entre outros, que são responsáveis por proverem os serviços de computação, comunicação e informação da instituição.

Tabela 9.1
Caracterização do
uso de recursos
computacionais.

Campo	Caracterização do uso
1	Tipo (administrativo, desenvolvimento, pesquisa, produção etc.). Regime (uso eventual ou contínuo, processamento em tempo real ou batch etc.). Carga (uso marginal ou intensivo de recursos do sistema). Período do uso (data inicial, data final).
2	Experiência do usuário em informática, relativa ao uso pretendido.
3	Parecer do administrador sobre a disponibilidade dos recursos para o uso pretendido (incluindo insumos, suporte técnico e treinamento).

Acompanhamento da utilização dos recursos computacionais:

- Manutenção preventiva.
- Tuning.
- Substituição, atualização e expansão de dispositivos saturados.
- Aquisição de novos equipamentos.
- Treinamento de pessoal de suporte.

A inclusão de um usuário no sistema deve ser documentada com o levantamento referido anteriormente, acrescido das informações presentes na tabela 9.2. Essas informações são resultantes da inclusão do usuário e devem ser registradas pelo administrador juntamente com as informações da tabela 9.1. O acompanhamento do uso dos recursos computacionais



da empresa também é uma atribuição do administrador do sistema e visa coletar dados para antecipar deficiências no sistema como um todo e subsidiar ações de:

- Manutenção preventiva;
- Ajustes de parâmetros para melhoria de desempenho (tuning);
- Substituição, atualização e expansão de dispositivos saturados;
- Planejamento para aquisição de novos equipamentos;
- Treinamento de operadores, monitores e equipe de suporte.

A coleta dessas informações nem sempre é fácil. Existem ferramentas comerciais especializadas, mas o administrador de sistemas pode, inicialmente, utilizar os recursos do próprio Sistema Operacional e de softwares específicos para coletar estatísticas simples sobre o uso de CPUs, memória, espaço em disco, links de comunicação de dados etc. À medida que os dados coletados se mostrarem importantes para o planejamento de ações futuras, o processo de coleta pode ser aperfeiçoado para incluir mais detalhes e, eventualmente, poderá ser adquirido um software específico para essa função. Os relatórios gerados a partir dos dados consolidados devem ser claros, objetivos e concisos. O formato, caso ainda não exista, deve ser definido em conjunto com os demais integrantes da equipe e com o gerente de TI da instituição.

Informações sobre os novos usuários

- Documentação da inclusão do usuário no sistema.
- Assinatura do Termo de Responsabilidade.
- Registro das informações da conta.
- Termo de responsabilidade:
 - De acordo/Data/Assinatura do usuário.



Para ser efetivada, a inclusão de um novo usuário deve ser acompanhada de um termo de responsabilidade, no qual o futuro usuário declara sua concordância com os termos e condições de uso estabelecidos pela instituição e da assinatura de seu superior imediato. Um exemplo simples de termo de responsabilidade é apresentado na tabela 9.2.

Campo	Dados cadastrais
1	Nome completo do usuário
2	Documento de identificação (matrícula na empresa ou RG)
3	Departamento/Setor
4	Cargo/Função
5	Nome do chefe imediato
7	Telefones de contato
8	Endereço eletrônico
9	Datas de recebimento/processamento da solicitação de inclusão
10	Identificação do técnico responsável pela inclusão da conta

Tabela 9.2
Dados cadastrais dos usuários.

Atendimento aos usuários




- Disponibilizar as informações básicas sobre características e condições de utilização dos sistemas.
 - Formato simples (eletrônico) e de fácil acesso.
- Descobrir as necessidades dos usuários.
- Tratar de problemas de inadequação de recursos.
- Promover treinamento por meio da equipe de suporte.
- Respeitar o usuário e suas necessidades.
- Recursos computacionais são meios, e não fins.
- Fazer verificação periódica do índice de satisfação dos usuários.
- Identificação de falhas no atendimento.
- Não comprometimento das atividades fim da instituição.

Visando facilitar o trabalho do novo usuário, o administrador de sistemas deve disponibilizar, em forma impressa e eletrônica, as informações básicas sobre as características e condições de uso dos sistemas sob sua responsabilidade. De nada adianta ter um sistema muito bem organizado se os usuários não estão sendo bem atendidos nas suas necessidades. Os usuários são os clientes e, como clientes, precisam ser ouvidos e respeitados, mesmo quando suas expectativas sejam inadequadas para a realidade da instituição. Quando se trata de instituições com uma alta taxa de rotatividade de usuários, como é o caso de instituições de ensino e pesquisa, essas informações devem possuir formato simples e resumido e serem disponibilizadas em local de fácil acesso.

Cabe ao administrador do sistema, em última instância, tentar descobrir as reais necessidades do usuário e decidir como pode melhor atendê-los. Se o problema é de desconhecimento dos recursos existentes e de como utilizá-los, o usuário deve ser treinado, o que pode ser feito pela própria equipe de suporte, ou com a ajuda de outros usuários experientes. Caso o problema seja de inadequação dos recursos computacionais existentes para o fim desejado, o administrador pode levar o problema ao seu superior e tentar resolvê-lo da melhor maneira possível.

Eventualmente, se o trabalho do usuário for importante para a instituição, o recurso poderá ser adquirido. Qualquer que seja a solução encontrada para as necessidades dos usuários, o administrador de sistemas precisa se colocar sempre na posição de quem ajuda a viabilizar os projetos da instituição, e não como a pessoa interessada unicamente em manter os sistemas operando de maneira adequada.

 O importante é que o usuário do sistema seja respeitado em suas necessidades.

Afinal, os recursos computacionais são meios, e não fins para a empresa. Eles têm de ser utilizados com o objetivo de melhorar a produtividade dos funcionários e aumentar a competitividade da instituição como um todo, e não como vitrine do estágio tecnológico por ela alcançado. O administrador de sistemas necessita manter contato permanente com os usuários e estimulá-los a opinarem sobre o funcionamento dos sistemas sob sua responsabilidade e sobre o atendimento que estão recebendo da equipe técnica. Essa postura é fundamental para que o administrador identifique, em primeira mão, as falhas de atendimento da sua equipe, ou as necessidades não atendidas dos usuários, e possa corrigi-las antes que



comprometam as atividades finais da instituição. Se toda a equipe sob sua coordenação estiver imbuída desse mesmo espírito, o administrador de sistemas terá mostrado possuir não só competência técnica, mas também organizacional.

Controle de recursos computacionais

- De uma pequena rede com estações de trabalho a uma grande rede geograficamente distribuída.
- Cadastro com informações de identificação e localização de cada equipamento.
- Histórico de manutenções preventivas ou corretivas e atualizações de hardware e software.
- Inventário de hardware e software instalados.
- Checklist em papel para ser utilizado pela equipe de suporte local.



Os recursos computacionais sob responsabilidade de um administrador de sistemas podem variar desde uma pequena rede local com estações individuais de trabalho, para uso dedicado de um pequeno grupo de usuários, até uma grande rede com diversos servidores, equipamentos de conectividade, estações de trabalho etc. Qualquer que seja o caso, é importante que o administrador mantenha informações atualizadas sobre os itens que compõem a infraestrutura de TI. A manutenção de um cadastro com informações de identificação e de localização de cada equipamento é uma atividade importante em organizações com uma infraestrutura complexa e distribuída por diversas unidades separadas geograficamente. A tabela 9.3 mostra os dados essenciais que devem ser coletados para a montagem desse cadastro.

É importante, também, manter um histórico das ocorrências de cada equipamento, incluindo atividades de manutenção preventiva e corretiva e de atualização de hardware e software. Vale observar a importância de serem registradas as versões e os respectivos números de licenças de software comercial instalado nos equipamentos, para facilitar o controle e evitar problemas com cópias ilegais. Essas informações devem estar, preferencialmente, em formato eletrônico. Aplicativos de planilha e de bancos de dados, ou mesmo softwares de inventário, são uma boa alternativa para armazenamento desses dados, uma vez que facilitam a geração de relatórios. Mas é bom existir um checklist em papel para ser utilizado quando a equipe de suporte sair a campo para diagnosticar algum problema.

Cadastro de hardware

- Guardar informações.
- Manter histórico das ocorrências de cada equipamento.
- Possuir registro de versões e de licenças de software.
- Guardar informações, preferencialmente em formato eletrônico.



O cadastro dos equipamentos de TI deve ser feito independentemente do software utilizado para esse propósito. Em grandes organizações, essa atividade é fundamental para que o administrador não perca o controle sobre os ativos pelos quais é responsável. A tabela 9.3 mostra uma relação de informações que devem fazer parte do cadastro de um item de hardware, mesmo que em alguns casos, alguns dos itens relacionados não sejam aplicáveis:

Tabela 9.3
Cadastro de
hardware.

Campo	Dados Cadastrais
1	Tipo do equipamento/Dispositivo
2	Marca/Modelo
3	Configuração de hardware (dados principais)
4	Identificação patrimonial
5	Número de série
6	Localização física
7	Operador/usuário responsável (nome completo, departamento e telefone)
8	Softwares instalados (título, versão e número da licença)
9	Endereço IP
10	Nome do computador na rede

Também é importante que se faça o cadastro de todos os softwares utilizados pela instituição, visando melhor controle sobre licenças e necessidades de uso, usando formato semelhante ao utilizado na tabela 9.3.

Diagnóstico de falhas

- Uma das atividades mais desafiantes do administrador.
- Tempo gasto no diagnóstico pode atrasar a execução de outras tarefas.
- Utilização do Livro de Registro de Ocorrências.



Por mais responsável e dedicado que seja, o administrador de sistemas não vai conseguir evitar que ocorram falhas no funcionamento dos sistemas sob sua responsabilidade. Essas falhas podem ter as causas mais diversas, incluindo:

- Bugs de software;
- Mau funcionamento de dispositivos e equipamentos;
- Incompatibilidade de versões de hardware e software;
- Mau uso intencional;
- Erro de operação;
- Condições físicas inadequadas.

O diagnóstico de falhas, devido ao seu caráter investigativo, é uma das atividades mais desafiantes entre as atribuições do administrador de sistemas. O problema é que o tempo gasto no diagnóstico de uma falha pode atrasar outras tarefas também importantes e comprometer toda a programação de atividades, tornando o administrador de sistemas um verdadeiro “apagador de incêndios”. Além da experiência, um dos instrumentos mais efetivos com que o administrador de sistemas pode contar ao diagnosticar uma falha é o “Livro de Registro de Ocorrências” do sistema.

Registro de ocorrências

O registro de ocorrências é uma atividade importante no dia a dia de um administrador de sistemas e constitui uma excelente referência para consultas, identificação de problemas decorrentes de associações entre problemas (que sugerem efeitos colaterais), de soluções diferentes para o mesmo problema (que sugerem diagnósticos incorretos) etc.

O livro de registro de ocorrências é utilizado para registrar, cronologicamente, as principais ações realizadas pelo administrador nos diversos sistemas que administra, bem como acontecimentos relevantes que tenham afetado algum componente da infraestrutura de TI da organização. A tabela 9.4 mostra uma relação de informações que devem fazer parte de um registro de ocorrências.

Campo	Informações
1	Número sequencial de identificação
2	Data e hora do registro da ocorrência
3	Descrição resumida do problema
4	Hardware/Software/Serviço afetado
5	Autor do registro
6	Categoria da ocorrência (exemplo: parada total, parada parcial, atualização de software, manutenção de hardware, instalação de novo hardware etc.)
7	Diagnóstico provável
8	Solução adotada
9	Data/hora da solução da ocorrência
10	Autor da solução
11	Observações (para referência a outras ocorrências etc.)

Tabela 9.4
Registro de ocorrências.

Esse tipo de documento, além de contribuir para elevar o nível de qualidade da administração dos sistemas de uma instituição, facilita o trabalho da equipe de TI durante a resolução de problemas, liberando o administrador para atividades mais complexas. O campo 6 da tabela 9.4, por exemplo, é útil para agrupar ocorrências de um mesmo tipo, facilitando a produção de relatórios consolidados. O administrador pode criar categorias específicas para a sua instituição.

Rotina de backup

- Lembrar que para diferentes instituições há diferentes necessidades de backup.
- Elaborar o plano de backup.
- Promover o rodízio dos conjuntos de mídia.
- Cuidar do correto acondicionamento físico das mídias.
- Controlar o acesso ao local de armazenamento das mídias de backup.
- Dependendo da necessidade, o backup deve ser duplicado e armazenado em locais distintos.



A atividade de backup é uma das atribuições mais críticas do administrador de sistemas. Deve ser planejada de acordo com as necessidades da instituição e ser de amplo conhecimento de todos os usuários. Avaliadas as necessidades da empresa, com base nos critérios discutidos na sessão de aprendizagem 6, o administrador deve elaborar o plano de backup e cumpri-lo rigorosamente.

No que diz respeito à execução dessa atividade, deve ser dada especial atenção à documentação da mídia utilizada, com o objetivo de facilitar a recuperação da informação salva, em

caso de necessidade. Diferentes instituições possuem diferentes necessidades de backup. Uma instituição de ensino e pesquisa é totalmente distinta de uma instituição financeira, no que diz respeito a essas necessidades. As mídias de backup devem estar devidamente identificadas por algum rótulo que caracterize o seu uso. O administrador deve ainda programar o rodízio dos conjuntos de mídias de backup (diários, semanais e mensais) para assegurar que o desgaste pelo uso seja proporcionalmente equivalente entre os vários conjuntos e que os backups totais sempre sejam feitos sobre conjuntos de mídia mais novos e, portanto, mais confiáveis.

O acondicionamento físico da mídia de backup precisa levar em conta os requisitos ambientais de armazenamento de cada tipo, seguindo as recomendações do fabricante. Além disso, o controle de acesso ao local de armazenamento das mídias de backup deve ser rigoroso. No caso de sistemas críticos, o backup pode ser duplicado e armazenado em locais distintos, para reduzir o risco de perda por catástrofes como incêndios, inundações etc.

10

Webmin

objetivos

Conhecer o Webmin, uma ferramenta para administração de servidores Unix por intermédio da web; Aprender os pré-requisitos necessários à instalação do Webmin; Identificar as principais vantagens do Webmin; Entender o conceito de módulo utilizado pelo Webmin; Conhecer o Usermin, uma ferramenta que pode ser usada pelos usuários de um sistema para gerenciarem seus perfis.

conceitos

Características gerais; Instalação; Interface de administração; Módulos; Usermin.

Características gerais

- Interface web para administração de sistemas Unix-like.
- Aplicação composta por um servidor web e por scripts Perl.
- Possui suporte à criptografia de dados, utilizando o protocolo SSL.
- Pode ter seu acesso restringido a determinados IPs ou redes.
- Possui um esquema de autenticação de usuários completamente independente do sistema.
- Controle de acesso:
 - ▣ Possui excelente flexibilidade em relação ao controle de acesso aos diversos módulos.
 - ▣ Permite registrar as alterações feitas por meio da interface web em arquivos de log.
 - ▣ As alterações feitas por meio do Webmin são executadas diretamente nos arquivos de configuração do sistema.
 - ▣ Deve ser executado pelo superusuário.



Nesta sessão de aprendizagem, serão abordados tópicos relacionados ao Webmin, uma interface web para administração de sistemas Unix-like. Trata-se de um pacote que possui um servidor web, que roda em uma porta TCP, definida durante a sua instalação, e alguns scripts CGI que podem atualizar diretamente os arquivos de configuração do sistema e de basicamente qualquer programa que esteja instalado. O servidor web e todos os outros scripts CGI são escritos em linguagem Perl. Serão apresentadas as principais características do Webmin e seus aspectos de segurança e controle de acesso, que o tornam uma alternativa interessante na rotina de trabalho de um administrador de sistemas.



Algumas características importantes do Webmin serão apresentadas a seguir:

- **Segurança:** seu sistema de autenticação de usuários é completamente independente do esquema utilizado pelo sistema. Isso significa que o administrador pode atribuir direitos a um usuário, por meio do Webmin, sem que esse usuário possua acesso a qualquer tipo de configuração por intermédio do sistema. Para possibilitar uma comunicação segura entre o servidor web e o browser cliente, pode ser utilizada uma conexão criptografada, por meio do protocolo SSL, principalmente em se tratando de atividades com permissãoamento de superusuário. O acesso ao Webmin também pode ser restringido a determinadas redes ou IPs específicos;
- **Funcionalidade:** o Webmin pode atuar como um servidor proxy. É possível estabelecer uma conexão segura a um firewall e, por intermédio dessa conexão, gerenciar computadores em uma intranet, atrás desse firewall. Essa função revela-se muito útil nos casos em que se deseja fornecer serviços de gerenciamento a clientes externos. Outra funcionalidade interessante é a oferecida pelo módulo “Custom Commands”, que permite ao administrador executar os seus próprios scripts e ter a saída deles redirecionada a uma página web. O Webmin deve ser executado pelo usuário root, já que a maioria das tarefas de administração que realiza, como editar arquivos de configuração ou reiniciar serviços, só podem ser executadas com os privilégios desse usuário;
- **Configuração:** o Webmin também pode ser configurado para registrar todas as alterações feitas por meio dele em arquivos de log. Essa característica é extremamente útil para a solução de problemas e para o controle, por parte do administrador, das mudanças que estão sendo feitas no sistema por meio dessa interface. O Webmin trabalha diretamente com os arquivos de configuração dos serviços, o que significa que ele não utiliza uma base de dados própria, ou algum outro meio para armazenar qualquer tipo de informação. É possível editar diretamente, com um editor de textos, o arquivo de configuração do Apache, por exemplo, sem preocupação com possíveis problemas que poderiam ser causados ao Webmin.

O Webmin também possui um excelente mecanismo de controle de acesso aos vários módulos disponíveis. O administrador pode, por exemplo, permitir a um determinado usuário o controle total sobre as configurações do servidor DNS, sem que esse usuário tenha acesso a quaisquer configurações do servidor Apache, ou mesmo permitir o acesso somente às configurações de determinadas zonas no servidor DNS. Essa característica torna-se bastante proveitosa quando é necessário delegar tarefas a vários administradores.

Instalação

Pacotes necessários:

- Perl.
- OpenSSL.
- Net_SSLeay.pm.
- Webmin.

As principais distribuições do Linux possuem um pacote pré-compilado do Webmin, que pode ser instalado usando um gerenciador de pacotes, que já instala automaticamente todas as dependências necessárias. No entanto, nesta sessão de aprendizagem, será apresentado o modo de instalação manual, usando o código-fonte do Webmin e de suas dependências. Para acessar o Webmin de forma segura, é necessário a instalação de três dependências que são exibidas a seguir:



- ▣ Perl;
- ▣ OpenSSL;
- ▣ Net_SSLeay.pm.

Perl

Instalação do Perl:

- ▣ # cd /usr/local/src
- ▣ # wget http://www.cpan.org/src/5.0/perl-5.14.2.tar.gz
- ▣ # tar -zxvf perl-5.14.2.tar.gz
- ▣ # cd perl-5.14.2
- ▣ # ./configure
- ▣ # make
- ▣ # make test
- ▣ # make install



O Webmin, conforme mencionado anteriormente, é escrito na linguagem de programação Perl. Sendo assim, esta deve estar instalada no sistema para que o Webmin possa ser executado. O Perl já vem instalado por padrão, na maioria das distribuições Linux, mas caso seja necessário instalá-lo, basta executar a sequência de comandos a seguir para instalar a versão 5.14.2:

```
# cd /usr/local/src
# wget http://www.cpan.org/src/5.0/perl-5.14.2.tar.gz
# tar -zxvf perl-5.14.2.tar.gz
# cd perl-5.14.2
# ./configure
# make
# make test
# make install
```

Se não ocorrer nenhum erro, passa-se à próxima seção. Caso contrário, o administrador pode consultar a seção de documentação no site oficial do Perl, em <http://www.perl.org>. Feito isso, o diretório “perl-5.14.2” pode ser excluído, assim como o pacote de instalação. A instalação também pode ser feita através dos gerenciadores de pacotes como yum e apt.

OpenSSL

Instalação do OpenSSL:

- ▣ # cd /usr/local/src
- ▣ # wget http://www.openssl.org/source/openssl-1.0.0g.tar.gz
- ▣ # tar -zxvf openssl-1.0.0g.tar.gz
- ▣ # cd openssl-1.0.0g
- ▣ # ./configure



- # make
- # make test
- # make install



O OpenSSL é um pacote que implementa, entre outras características, os protocolos SSL e TLS, que permitem a troca de informações criptografadas entre dois computadores através da rede. Na instalação padrão, o binário do OpenSSL é instalado no diretório "/usr/local/ssl/bin". É recomendado manter essa localização, pois outros programas que usam esse binário geralmente o procuram nesse diretório. Para a instalação da versão 1.0.0g do OpenSSL, deverá ser executada a seguinte sequência de comandos:

```
# cd /usr/local/src
# wget http://www.openssl.org/source/openssl-1.0.0g.tar.gz
# tar -zxvf openssl-1.0.0g.tar.gz
# cd openssl-1.0.0g
# ./configure
# make
# make test
# make install
```

Se não ocorrer nenhum erro, passa-se à próxima seção. Caso contrário, o administrador pode consultar a seção de documentação no site oficial do OpenSSL, em <http://www.openssl.org>. Feito isso, o diretório openssl-1.0.0g pode ser excluído, assim como o pacote de instalação. A instalação também pode ser feita através dos gerenciadores de pacotes como yum e apt.

Net_SSLeay.pm

Instalação do Net_SSLeay:

- # cd /usr/local/src
- # wget http://www.cpan.org/modules/by-module/Net/Net-SSLeay-1.45.tar.gz
- # tar -zxvf Net-SSLeay-1.45.tar.gz
- # cd Net-SSLeay-1.45
- # perl Makefile.PL
- # make
- # make install



O Net_SSLeay.pm nada mais é do que um módulo necessário para a implementação de aplicações Perl que utilizem funções da biblioteca OpenSSL. É o Net_SSLeay que ficará responsável pela geração do certificado digital requerido para o site seguro do Webmin. Para a instalação da versão 1.45 do Net_SSLeay, deverá ser executada a seguinte sequência de comandos:

```
# cd /usr/local/src
# wget http://www.cpan.org/modules/by-module/Net/Net-SSLeay-1.45.tar.gz
# tar -zxvf Net-SSLeay-1.45.tar.gz
# cd Net-SSLeay-1.45
```



```
# perl Makefile.PL  
# make  
# make install
```

Se não tivermos nenhum erro, passa-se à próxima seção. Do contrário, o administrador pode consultar o site <http://www.cpan.org>. Feito isso, o diretório “Net-SSLeay-1.45” pode ser excluído, assim como o pacote de instalação. A instalação também pode ser feita através dos gerenciadores de pacotes como yum e apt ou pelo repositório do CPAN.

Webmin

Instalação do Webmin:

- # cd /usr/local/src
- # wget <http://ufpr.dl.sourceforge.net/project/webadmin/Webmin/1.580/Webmin-1.580.tar.gz>
- # tar -zxvf Webmin-1.580.tar.gz
- # cd Webmin-1.580
- # ./setup.sh /usr/libexec/Webmin

Nesse tópico será abordada a instalação do programa Webmin propriamente dito. As instruções apresentadas nesse documento foram baseadas nas versões indicadas a seguir, mas devem ser semelhantes para versões futuras. Para a instalação da versão 1.580 do Webmin, deverá ser executada a sequência de comandos:

```
# cd /usr/local/src  
# wget http://ufpr.dl.sourceforge.net/project/webadmin/Webmin/1.580/  
Webmin-1.580.tar.gz  
# tar -zxvfWebmin-1.580.tar.gz  
# cd Webmin-1.580  
# ./setup.sh /usr/libexec/Webmin
```

Quando for executado o script `setup.sh`, que é o programa de instalação do Webmin, serão feitas algumas perguntas que, em sua maioria, devem ser mantidas com suas respostas padrão. O administrador deve escolher uma senha de acesso para a aplicação nesse passo. O diretório “/usr/libexec/Webmin” armazenará todos os scripts, imagens e páginas HTML. A seguir podem ser vistas as perguntas feitas pelo script `setup.sh` e as respostas utilizadas:

```
Config file directory [/etc/Webmin]:  
Log file directory [/var/Webmin]: /var/log/Webmin  
Full path to perl (default /usr/bin/perl):  
web server port (default 10000):  
Login name (default admin):  
Login password:  
Password again:  
Use SSL (y/n): y
```



```
Start Webmin at boot time (y/n): y
```

Feito isso, o diretório “/usr/local/src/Webmin-1.580” pode ser excluído, assim como o pacote de instalação. Se o administrador quiser desinstalar o Webmin, ele deve executar o script de desinstalação, /etc/Webmin/uninstall.sh, que, quando executado, remove todos os diretórios e arquivos adicionados por ele durante o processo de instalação.

Interface de administração

O acesso à interface do Webmin pode ser feito por meio de qualquer navegador web:

URL de acesso: <https://servidor:10000>

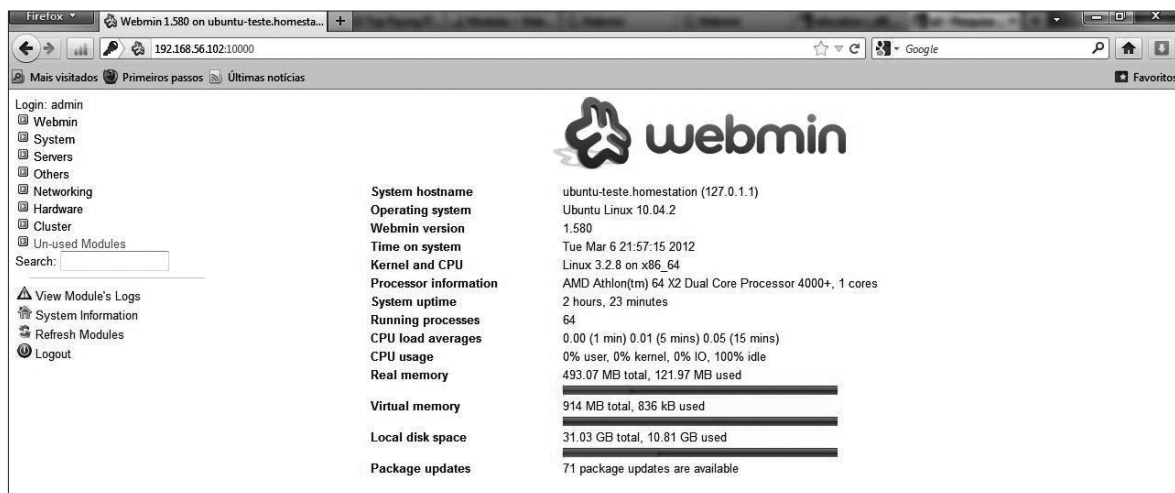
- Na página inicial, devem ser informados o usuário e a senha definidos durante a instalação.
- Apresenta um menu com as categorias principais de configuração do sistema.
- Cada categoria possui uma série de módulos que representam os diversos serviços disponíveis.



Finalizado o processo de instalação, deve-se testar o acesso ao Webmin. Para acessar sua interface, basta digitar a URL <https://servidor:10000> em qualquer navegador web, onde servidor é o nome do servidor onde o Webmin está instalado. Se a instalação tiver sido executada de maneira correta, a página inicial do Webmin deverá ser carregada.

Nessa página, o administrador deve informar o usuário e a senha de acesso à aplicação, que foram criados durante o processo de instalação do Webmin. Se o usuário e a senha fornecidos estiverem corretos, o acesso será autorizado e a página inicial do Webmin será exibida. A figura 10.1 mostra a página inicial do Webmin, que apresenta um menu na parte superior esquerda da tela, com as categorias de configuração do sistema. Cada categoria é composta por diversos módulos, que podem ser configurados de acordo com as necessidades do administrador.

Figura 10.1
Página inicial do Webmin.



Para cada categoria presente no menu, existe uma página principal correspondente, com os diversos itens de configuração do sistema ou de programas instalados pelo administrador. Navegando pelo Webmin, é possível notar que, apesar de existirem diversos módulos disponíveis para

serem administrados através de sua interface, nem sempre esses módulos estão habilitados, pois, em determinados casos, pode acontecer de determinado programa não estar devidamente instalado no servidor. Navegando pelas várias opções disponíveis no Webmin, pode-se ver, em poucos instantes, como essa ferramenta é extremamente poderosa e de fácil utilização, simplificando o trabalho tanto dos administradores novatos, como dos mais experientes.

Módulos



- São incorporados ao Webmin, permitindo que os mais diversos serviços sejam controlados por meio de sua interface web.
- Possui mais de 100 módulos em sua instalação padrão.
- Existem mais de 300 módulos desenvolvidos por terceiros para o Webmin.
- É possível utilizar a API do Webmin para o desenvolvimento de novos módulos.
- Devem ser escritos preferencialmente em linguagem Perl e seguir algumas regras.

O Webmin possui um design modular, onde todo e qualquer serviço ou característica é representado por meio de um conjunto de módulos. O Webmin possui mais de 100 módulos em sua instalação padrão, e muitos outros estão disponíveis na internet. Existem mais de 300 módulos escritos por terceiros, disponíveis para integração com o Webmin. Esses módulos cobrem desde LDAP até IDSs, como o Snort etc. A maioria deles é livre e regida pela GPL, porém alguns módulos são comerciais. A qualidade desses módulos varia, mas pode-se dizer que, para qualquer serviço que se queira gerenciar, existe um módulo disponível. Os módulos do Webmin ficam armazenados por padrão em diretórios dentro do diretório `"/usr/libexec/Webmin"`, definido durante a instalação. Para desenvolver um módulo externo, é necessário seguir algumas regras:

- Um módulo, a princípio, pode ser escrito em qualquer linguagem, mas para evitar problemas de incompatibilidade, é recomendável que seja escrito em linguagem Perl, versão 5.8 ou superior, e não utilize programas externos;
- Para ser exibido na interface do Webmin, um módulo deve conter o arquivo *module.info*, que contém as informações necessárias sobre o módulo: o arquivo *images/icon.gif*, que contém o ícone que representa o módulo, e o arquivo *lang/en*, que possui as mensagens de texto utilizadas pelo módulo, nesse caso, na língua inglesa.
- Evitar o uso de frames, javascript, DHTML e Flash.

Categorias de configuração:



- Webmin: módulos de configuração do Webmin.
- System: módulos de configuração de serviços de sistema.
- Servers: módulos de configuração de servidores.
- Others: módulos de configurações diversas.
- Networking: módulos de configuração de rede.
- Hardware: módulos de configuração de hardware.
- Cluster: módulos de configurações relativas a clusters.



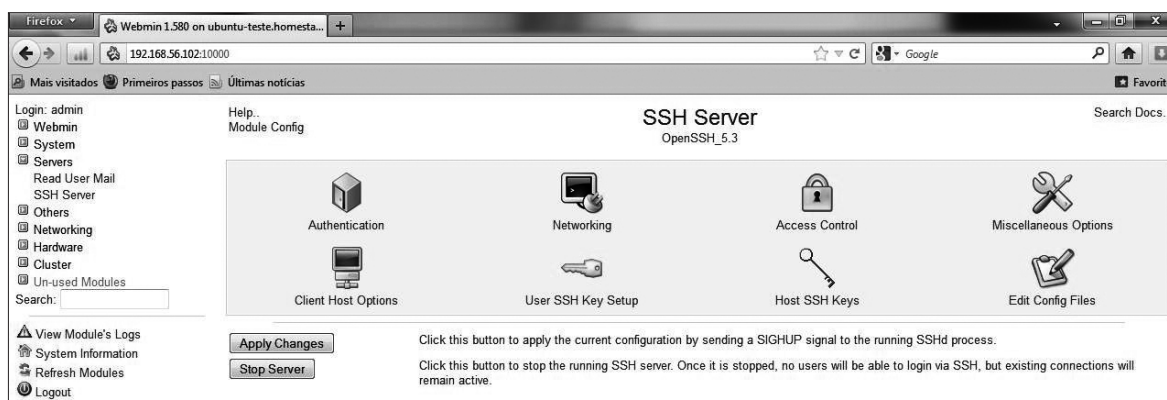
A seguir, são descritas as categorias presentes na interface do Webmin:

- **Webmin:** possui módulos de configuração do Webmin, como criação de usuários, backup dos arquivos de configuração etc.;
- **System:** contém módulos de configuração de serviços do sistema, como boot e shutdown, gerenciamento de file systems, backup etc.;
- **Servers:** tem módulos de configuração de servidores, como Apache, SSH etc.;
- **Others:** possui módulos de configurações diversas, como comandos do shell, status do sistema, módulos Perl etc.;
- **Networking:** contém módulos de configuração de rede, como firewall, monitoramento de banda etc.;
- **Hardware:** tem módulos de configuração de hardware, como gerenciamento de volumes lógicos, partições, impressoras etc.
- **Cluster:** possui módulos de configurações relativas à administração de clusters.

Um novo módulo pode ser instalado tanto através da linha de comando, quanto através da própria interface de administração do Webmin. Essa última opção pode ser seguida navegando em sua interface através das opções: Webmin, WebminConfiguration e Webmin Modules. As opções de instalação incluem: um módulo que está localizado no próprio servidor, um módulo que está localizado no computador que está acessando a interface do Webmin, um módulo que está disponível através de uma URL ou um módulo padrão do Webmin, presente na categoria “Un-used Modules”.

A figura 10.2 mostra a página principal de administração do serviço SSH, que está presente na categoria Servers. Como pode ser visto, é possível realizar uma série de configurações no serviço SSH utilizando o Webmin. Cada um dos outros módulos possui uma página inicial equivalente a esta, com diversas opções de configuração disponíveis. Devido à grande quantidade de módulos, suas páginas não serão exibidas nessa sessão, cabendo ao administrador navegar pela interface do Webmin para explorar todas as suas funcionalidades.

Figura 10.2
Página de administração do serviço SSH.



Usermin



- Interface web desenvolvida para auxiliar os usuários de sistemas Unix em tarefas cotidianas.
- Possibilita a alteração de senhas, edição de scripts de login, agendamento de tarefas etc.
- O Webmin possui um módulo que permite o gerenciamento do Usermin.



O Usermin pode ser acessado através do endereço `https://servidor:20000`.

Devido às inúmeras facilidades introduzidas pelo Webmin, seu desenvolvedor resolveu criar outro programa de gerenciamento, chamado Usermin, desta vez voltado exclusivamente para a administração de tarefas relativas a usuários.

Entre as características do Usermin, podemos citar que ele provê uma interface web, por meio da qual usuários de um sistema podem alterar suas senhas, editar seus scripts de login, agendar tarefas no cron etc. O Usermin pode ser gerenciado através do Webmin. Sendo assim, é possível restringir o nível de autonomia de cada usuário. Durante esse curso, não trataremos do aprendizado e da utilização do Usermin. Cabe ao aluno o estudo mais aprofundado dessa ferramenta de grande utilidade nas tarefas realizadas pelos administradores de sistemas. O Usermin possui os seguintes módulos:

- **Usermin:** permite a troca de linguagem e de tema;
- **Others:** acesso a várias outras aplicações, como file manager, submissão de tarefas utilizando o cron, documentação do sistema etc.;
- **Mail:** leitura de mensagens, criação de redirecionamentos, filtros etc.;
- **Login:** possibilita ao usuário administrar sua conta, podendo fazer mudança de senha, visualizar processos, executar comandos etc.;
- **Applications:** iniciar aplicações como, por exemplo, GPGencryption, fazer upload e download de arquivos etc.



Ari Frazão Jr. é bacharel em Ciência da Computação pela Universidade Federal da Paraíba (UFPB) e mestre em Ciência da Computação, na área de redes de computadores, pela Universidade Federal de Pernambuco (UFPE). Atualmente é responsável pelas áreas de engenharia e operações da Rede Nacional de Ensino e Pesquisa (RNP), onde atua desde 1993.

Marcelo Castellan Braga possui graduação em Engenharia Eletrônica pelo CEFET-RJ, pós-graduação em Análise, Projeto e Gerência de Sistemas pela PUC-RJ e mestrado em informática pela UNIRIO. Atualmente é sócio diretor da MCB Tech, empresa que presta consultoria em redes de computadores, serviços de internet, segurança de dados e desenvolvimento de software. Atuou durante mais de 10 anos na área de TI em empresas como Rede Nacional de Ensino e Pesquisa (RNP) e Embratel.

Este curso é indicado para analistas de suporte e responsáveis pela manutenção de servidores e especialistas que desejem aprofundar os conhecimentos para se tornarem administradores de sistemas Linux. Administradores de sistemas que precisam gerenciar sistemas Linux também se beneficiarão deste curso, assim como os profissionais que pretendem iniciar os estudos para a certificação LPI, do Linux Professional Institute. Este livro inclui os roteiros das atividades práticas e o conteúdo dos slides apresentados em sala de aula, apoiando profissionais na disseminação deste conhecimento em suas organizações ou localidades de origem.

ISBN 978-85-63630-52-0



9 788563 630520