

- LPI 101 -

**Administração
de Sistemas Linux**

Há mais de 17 anos, a **GREEN Treinamento** vem atuando exclusivamente com treinamento e possui toda a infra-estrutura para oferecer-lhe a melhor experiência no aprendizado de informática.

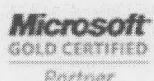
Já treinamos mais de **130.000** alunos de todas as procedências: grandes empresas, particulares, microempresários, estudantes, etc.

Somos um Centro de Treinamento Oficial certificado pelos três principais fabricantes da indústria de softwares: nós somos **CPLS da Microsoft (antigo CTEC)**, **ECIS da IBM** e **Training Provider da Mandriva Conectiva**, ou seja, atendemos aos padrões internacionais de qualidade destas empresas, contando com profissionais certificados e informações técnicas e comerciais sempre atualizadas.

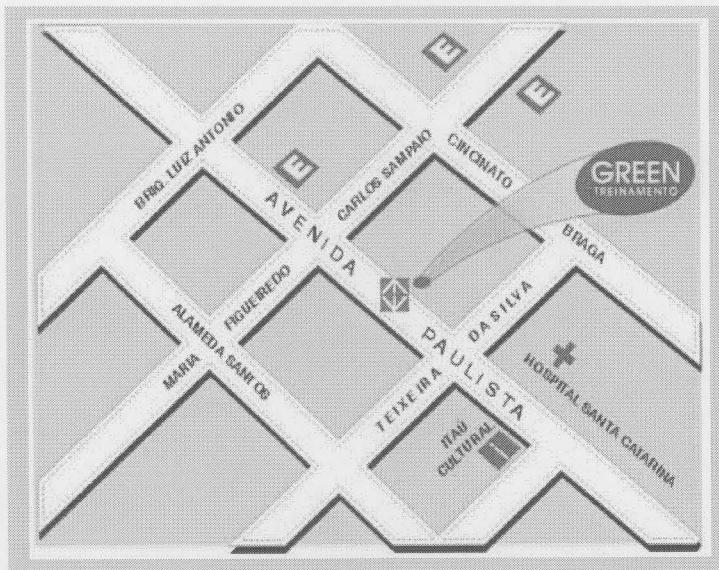
Somos **Centro de Testes da Pearson VUE – Virtual University Enterprises**. Podemos aplicar testes para diversas certificações, entre as quais: Novell, Ericsson, Microsoft, Lucent, LPI (Linux), Informix, CompTIA, Certified Internet Webmaster, Lotus, Avaya, Allaire, Siebel, BROCADE, Cisco, RSA Security, NASD, Check Point, Macromedia, GNU e Intershop Communications.

Possuímos modernos laboratórios equipados com computadores de última geração. Existem também unidades móveis onde instalamos os equipamentos no cliente para ministrar cursos "IN-HOUSE".

"Oferecer a melhor capacitação profissional é o nosso objetivo"



Learning Solutions



GREEN TREINAMENTO

Informações: (11) 3253-5299

Av. Paulista, 326 - 12º Andar
Bela Vista - São Paulo - SP
Metrô Brigadeiro
(Saída Carlos Sampaio)
<http://www.green.com.br>
cursos@green.com.br

Estacionamentos Conveniados

Desconto válido somente com carimbo da recepção no ticket do estacionamento.

Créditos

Marcos Sungaila
autor

Copyright Marcos Sungaila

TODOS OS DIREITOS RESERVADOS. É proibida a reprodução parcial ou total desta obra, por qualquer meio, seja este gráfico, fotográfico ou eletrônico entre outros, bem como a inclusão deste trabalho em qualquer sistema de arquivo de processamento de dados (disquetes, cd's, dvd's, etc), sem prévia autorização por escrito de Marcos Sungaila. Tais vedações se aplicam também à supressão de informações através da alteração das características originais deste trabalho, incluindo alterações de diagramação e de características gráficas da obra.

A violação de direitos autorais é crime, nos termos da lei 9.610/98, e conforme estabelecido pelo artigo 184 do Código Penal. Aplica-se a esta obra e em relação aos nomes empresariais, marcas mistas, figurativas ou nominativas contidas na mesma, as disposições legais da lei 9.279/96.

A reprodução deste material sem autorização escrita de Marcos Sungaila é crime, de acordo com a legislação em vigor. Este material não pode ser fotocopiado.

Se você verificar que esta obra foi fotocopiada ou reproduzida de qualquer outra forma, entre em contato com o autor pelo e-mail marcos@savant.com.br.

Copyright 2006, Marcos Sungaila

Índice

INTRODUÇÃO AO LINUX.....	6
HISTÓRICO	6
SOFTWARE LIVRE	7
LICENÇAS LIVRES	7
DISTRIBUIÇÕES LINUX	8
DEBIAN (WWW.DEBIAN.ORG)	8
FEDORA (FEDORA.REDHAT.COM)	8
MANDRIVA (WWW.MANDRIVA.COM.BR)	8
RED HAT (WWW.REDHAT.COM)	9
SLACKWARE (WWW.SLACKWARE.ORG)	9
SUSe (WWW.NOVELL.COM/LINUX).....	9
TURBOLINUX (WWW.TURBOLINUX.COM).....	9
REFERÊNCIAS.....	9
INSTALAÇÃO DO LINUX	10
LAYOUT DE PARTICIONAMENTO DO DISCO	10
VISÃO GERAL DE UM SISTEMA DE ARQUIVOS (PARTIÇÕES).....	10
DECIDINDO O PARTICIONAMENTO	11
GERENCIADOR DE INICIALIZAÇÃO.....	12
<i>LILO – Linux Loader.....</i>	13
<i>GRUB - GRand Unified Bootloader.....</i>	13
INSTALANDO O LINUX PELA PRIMEIRA VEZ	14
CRIAÇÃO DOS DISQUETES DE INSTALAÇÃO A PARTIR DO WINDOWS®.....	14
CRIAÇÃO DOS DISQUETES DE INSTALAÇÃO A PARTIR DO LINUX.....	15
INSTALANDO O DEBIAN	16
CONHECENDO O LINUX.....	24
USUÁRIOS E GRUPOS	24
MODO TEXTO OU MODO GRÁFICO?.....	25
MODO Gráfico	25
SISTEMAS DE ARQUIVOS E ESTRUTURA DE DIRETÓRIOS.....	26
SISTEMAS DE ARQUIVOS.....	26
ESTRUTURA DE DIRETÓRIOS.....	27
DESCRIPÇÃO DA ÁRVORE DE DIRETÓRIOS DO LINUX.....	28
<i>/bin</i>	28
<i>/boot</i>	28
<i>/dev.....</i>	28
<i>/etc.....</i>	28
<i>/home.....</i>	29
<i>/lib</i>	29
<i>/mnt.....</i>	29
<i>/opt</i>	29
<i>/proc</i>	29
<i>/root.....</i>	29

/sbin.....	30
/sys.....	30
/tmp.....	30
/usr.....	30
/var.....	30
PONTOS DE MONTAGEM	31
TIPOS DE USUÁRIOS.....	31
TIPOS DE ARQUIVOS E PERMISSÕES.....	31
TIPOS DE ARQUIVOS	32
PERMISSÕES	32
ALTERANDO PERMISSÕES.....	34
DEFININDO PERMISSÕES PELO MÉTODO DIFERENCIAL	35
DEFININDO PERMISSÕES PELO MÉTODO DE ATRIBUIÇÃO DIRETA.....	35
DEFININDO PERMISSÕES PELO MÉTODO OCTAL	35
PERMISSÕES ESPECIAIS	36
STICK BIT.....	36
SGID	37
SUID	37
LINKS	38
HARD LINKS	38
SYMLINKS	38
METACARACTERES	39
SÍMBOLO ?	39
SÍMBOLO *	39
AGRUPADOR COLCHETES – []	40
AGRUPADOR CHAVES – { }	41
INSTALAÇÃO DE PROGRAMAS.....	42
RPM	42
GERENCIAMENTO DE USUÁRIOS E GRUPOS	44
COMANDOS PARA GERENCIAMENTO	44
CRIANDO E REMOVENDO USUÁRIOS E GRUPOS	44
COMANDOS E PARÂMETROS MAIS UTILIZADOS.....	45
PLANEJE SEU AMBIENTE.....	47
AMBIENTE DE TRABALHO DO USUÁRIO	47
VARIÁVEIS	47
VARIÁVEIS DE AMBIENTE.....	48
SCRIPTS EXECUTADOS NO LOGIN DO USUÁRIO	49
VARIÁVEIS PRÉ-DEFINIDAS NO AMBIENTE DO USUÁRIO	49
COMANDOS UNIX E GNU	50
LINHA DE COMANDO	50
SINTAXE.....	50

ACESSANDO O SISTEMA OPERACIONAL	51
DESLIGANDO O LINUX	52
PRIMEIROS COMANDOS	53
FILTROS EM ARQUIVOS	63
REDIRECIONAMENTO E CANALIZAÇÃO	64
LOCALIZAÇÃO DE ARQUIVOS.....	66
FIND	66
LOCATE	67
WHEREIS	68
WHICH.....	68
COMANDOS ÚTEIS	68
HEAD.....	68
TAIL	69
SORT	69
EXPAND.....	70
WC	71
CUT	71
TR.....	71
NL	72
DIFF.....	72
GERENCIAMENTO DE PROCESSOS	73
PROGRAMAS E PROCESSOS	73
DAEMONS E ZUMBIS	73
IDENTIFICAÇÃO DE UM PROCESSO	73
VERIFICANDO OS PROCESSOS EM EXECUÇÃO	74
<i>ps</i>	74
<i>pstree</i>	75
<i>top</i>	76
<i>fuser</i>	77
<i>pidof</i>	78
<i>kill</i>	78
<i>killall</i>	79
<i>pgrep – process grep</i>	79
<i>pkill – process kill</i>	80
<i>nice</i>	80
<i>renice</i>	81
<i>jobs, fg bg</i>	81
EXPRESSÕES REGULARES	82
EDIÇÃO DE TEXTOS	83
VIM – VI IMPROVED	83
COMANDOS E AÇÕES DENTRO DO VI/VIM	84
GERENCIAMENTO DE PARTIÇÕES E FORMATAÇÃO.....	86

ENTENDO OS SISTEMAS DE ARQUIVOS	86
INODES.....	86
TIPOS DE SISTEMAS DE ARQUIVOS	87
SISTEMA DE ARQUIVOS EXT2	87
SISTEMA DE ARQUIVOS EXT3	87
DISPOSITIVOS E SUA IDENTIFICAÇÃO	87
PARTICIONAMENTO.....	88
FORMATANDO UM SISTEMA DE ARQUIVOS	89
CHECANDO UM SISTEMA DE ARQUIVOS	90

Introdução ao Linux

Histórico

Durante o período de 1968-1969 um grupo de trabalho formado pelas empresas Bell Telephone Labs da AT&T, General Electric e o MIT trabalhavam em um sistema operacional chamado MULTICS (Multiplexed Information and Computing Service) para criar um sistema operacional para grandes computadores capaz de permitir seu uso por milhares de usuários simultaneamente.

O projeto MULTICS não atingiu seus objetivos e sua equipe foi desmantelada. Com o fim do projeto, os desenvolvedores Ken Thompson e Dennis Ritchie da Bell Labs aproveitaram a experiência adquirida no projeto original rescreveram um sistema operacional com o único intuito de jogar “space war” em um pequeno equipamento DEC PDP-7 com 4k de ram para programas de usuários. O resultado desta “brincadeira” foi um sistema que um de seus colegas de trabalho chamou de UNICS (UNiplexed Information and Computing Service) em contrapartida com o MULTICS original.

Este primeiro sistema foi desenvolvido em assembly e em 1973 surgiu a primeira versão de seu kernel escrito em C para um computador DEC PDP-11.

Nesta época, a AT&T fornecia cópias das fontes de seu sistema operacional às universidades para fins educacionais, sem custo algum. Durante o período entre 1977 e 1982 a AT&T unificou várias versões do Unix em um sistema chamado Unix System III, tendo sido a base de desenvolvimento e evolução até o Unix System V.

O Minix desenvolvido por Andrew S. Tanenbaum, um sistema baseado no padrão POSIX (Portable Operating System Interface) do UNIX foi a base dos estudos de Linus Torvalds, estudante da Universidade de Helsink na Finlândia. Sua intenção era adaptar este sistema para seu equipamento, acrescentando recursos e funcionalidades extras.

Este projeto deu origem a um novo kernel, com os recursos e funcionalidades adicionais em um modelo diferente, tendo sido publicado em uma lista de discussão da USENET em 25 de agosto de 1991, oferecendo uma alternativa ao kernel do MINIX e incentivando qualquer pessoa a modificá-lo e adaptá-lo a novas necessidades. Em pouco tempo o resultado foi percebido na forma de correções e colaborações de novos recursos, dando origem ao kernel do Linux.

As principais características do Linux são:

- Sistema operacional multi-tarefa e multi-usuário, baseado no padrão POSIX oferece compatibilidade com todos os aplicativos desenvolvidos no projeto GNU da FSF (Free Software Foundation)
- Compatibilidade com várias plataformas de processadores, entre elas o padrão x86, PPC, S/390 e Alpha entre outros
- Seu kernel foi publicado por Linus Torvalds sob a licença GPL da FSF garantindo que possa ser utilizado para qualquer fim em qualquer ambiente de trabalho

Abaixo está a transcrição do email original enviado por Torvalds à lista comp.os.minix da USENET.

Software Livre

O Linux é a primeira referência quando falamos em software livre, sendo que este termo muitas vezes gera confusão.

Software Livre, do inglês Free Software, significa a liberdade de uso que seus autores oferecem aos que o escolhem um sistema publicado desta forma. A Free Software Foundation definiu quatro características básicas às quais um software deve atender para ser considerado como LIVRE, são elas:

- Liberdade poder executar o programa para qualquer fim, pessoal ou comercial.
- Liberdade para estudar o programa que deve ter seus fontes disponíveis para que qualquer pessoa possa entender seu funcionamento, e adaptá-lo às suas necessidades.

```
From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID: <1991Aug25.205708.9541@klaava.Helsinki.FI>
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki

Hello everybody out there using minix -
I'm doing a (free) operating system (just a hobby, won't be big and
professional like gnu) for 386(486) AT clones. This has been brewing
since april, and is starting to get ready. I'd like any feedback on
things people like/dislike in minix, as my OS resembles it somewhat
(same physical layout of the file-system(due to practical reasons))
among other things). I've currently ported bash(1.08) and gcc(1.40), and
things seem to work. This implies that I'll get something practical within a
few months, and I'd like to know what features most people would want. Any
suggestions are welcome, but I won't promise I'll implement them :-
Linus (torvalds@kruuna.helsinki.fi)
PS. Yes - it's free of any minix code, and it has a multi-threaded fs.
It is NOT protable (uses 386 task switching etc), and it probably never
will support anything other than AT-harddisks, as that's
all I have :-(.
```

- Liberdade para poder ser modificado e distribuir o programa a qualquer pessoa de forma a beneficiar a todos pelo seu avanço.
- Liberdade para copiar e redistribuir o programa a seus amigos e conhecidos, sem que seja necessário o pagamento de licenças para isto.

Licenças Livres

Todos os softwares atualmente em circulação possuem um método de licenciamento, proprietário ou livre, comercial ou gratuito. Entre os sistemas do software livre a licença mais conhecida é a GPL (General Public License) do projeto GNU (<http://www.gnu.org>) da Free Software Foundation. Todos os projetos criados pela FSF são publicados sob a GPL o que garante as quatro liberdades básicas a eles.

Uma confusão básica que ocorre com o software livre é que, apesar de poder ser modificado e redistribuído, algumas regras devem ser obedecidas para que se possa fazer pleno uso das liberdades básicas, entre elas temos:

- Referência ao autor e ao projeto original.

- Informar que os dados originais foram alterados.
- Exibir informações de copíright e da ausência de garantias no uso ou da intenção do software.
- Não se pode modificar a licença de publicação original de um trabalho, ou seja, um projeto publicado sob uma licença GPL deverá permanecer livre bem como todos os trabalhos derivados dele.

Além das licenças de software livre existem ainda as licenças de documentação que visa garantir que manuais e textos possam ser utilizados, modificados ou redistribuídos de acordo com sua publicação original.

Distribuições Linux

Distribuições Linux são o conjunto de vários softwares agrupados em mídias com instaladores personalizados que visam facilitar o trabalho do usuário ou administrador. O Linux em si é apenas o kernel publicado por Linus Torvalds, sendo que as distribuições devem ser classificadas como GNU/Linux por terem, em sua grande maioria, mais softwares do projeto GNU que apenas o kernel de Torvalds.

Hoje temos várias distribuições que podem ser baixadas da Internet. Esta variedade é devido ao fato de cada uma apresenta uma peculiaridade ou fim específico como uso em Desktops ou em servidores, em sistemas embarcados como celulares ou dispositivos pessoais como PDA's. Entre as mais conhecidas temos:

Debian (www.debian.org)

É a maior distribuição livre existente. Foi criada em 1993 e, até hoje, não é ligada a qualquer empresa. É ainda, uma das maiores distribuições Linux contando com 2 dvd's de instalação e quase 9,2Gb de software. Devido à sua característica livre é uma das distribuições mais utilizadas em Universidades e projetos abertos. Utiliza o formato DEB de pacotes de instalação de programas e possui versões para 10 arquiteturas de processadores. Utilizado o gerenciador de pacotes apt que facilita as tarefas de instalação de novos programas e manutenção do equipamento.

Fedora (fedora.redhat.com)

Distribuição criada em 2001, foi baseada no RedHat 9.0 até hoje é a versão livre do RedHat. Possui grande compatibilidade com sistemas RedHat, chegando a compartilhar vários pacotes entre seus sistemas.

Mandriva (www.mandriva.com.br)

Distribuição que surgiu da união da Mandrake da Conectiva tem como características básicas sua facilidade de uso tanto para usuários finais quanto para servidores. Possui versões específicas para Desktops e servidores e ainda uma versão chamada de Corporate edition. Utiliza o formato RPM de distribuição de pacotes e os gerenciadores urpmi e apt (este último portado para o formato rpm pela Conectiva).

Red Hat (www.redhat.com)

Distribuição criada em 1994 deu origem a várias outras distribuições. Criou o formato de pacotes RPM e atualmente só está disponível em versões comerciais para Desktops e servidores.

Slackware (www.slackware.org)

Criada em 2003, foi uma das distribuições mais utilizadas no Brasil na década de 90. Adepta da simplicidade, oferece poucos recursos administrativos em sua instalação padrão. Entre as distribuições Linux é a que mais se assemelha aos sistemas Unix.

SuSe (www.novell.com/linux)

Distribuição de origem alemão surgida em 1992 foi comprada pela Novell em 2003. Manteve por muito tempo a política de não distribuir seus cd's de instalação de forma aberta, apenas por aquisição de seus pacotes comerciais. Em 2005 surgiu o projeto openSuSe com o intuito de popularizar a distribuição e ampliar seu uso mundialmente. Utiliza o formato RPM de pacotes e possui o gerenciador YAST para cuidar de várias tarefas administrativas do sistema operacional.

TurboLinux (www.turbolinux.com)

Criada em 1992 é a principal distribuição asiática e possui suporte a vários idiomas locais.

Referências

História do Linux (<https://netfiles.uiuc.edu/rhasan/linux/>)

Unix History (<http://www.english.uga.edu/hc/unixhistoryrev.html>)

Linux history by Linus Torvalds (<http://www.cs.cmu.edu/~awb/linux.history.html>)

Instalação do Linux

O processo de instalação de uma distribuição Linux é simples, apesar de diferente de um sistema operacional Windows®, por exemplo, possui as mesmas etapas de particionamento e criação de usuários, além de personalização do conjunto de softwares a serem disponibilizados na instalação final.

Nosso curso é voltado para a certificação LPI, onde são aceitas duas distribuições Linux:

- Debian: a maior distribuição totalmente livre, atualmente em sua versão 3.1 incorpora todas as características de um sistema operacional Linux moderno além de ser uma das distribuições Linux mais estáveis. Várias outras distribuições são baseadas em Debian, entre elas: Knoppix, Kurumin e XandrOS.
- RedHat: atualmente conta apenas com versões comerciais para servidores e desktops, sendo que a última versão livre é a versão 9.0 lançada em 2002.

Em função destas características, adotamos o Debian como distribuição para nosso curso, e também por oferecer a maior gama de softwares disponíveis para instalação.

Layout de particionamento do disco

Vamos assumir que você está criando um sistema de arquivos em um equipamento com, ao menos, um disco rígido e que você deseja iniciar o computador a partir deste disco. É necessário decidir o particionamento deste disco durante o processo de instalação tendo em mente o uso do equipamento. Um particionamento inadequado pode prejudicar o funcionamento do sistema instalado. Contudo é bom saber que é possível alterar o layout do particionamento depois do sistema instalado porém isto exige trabalho extra e muita atenção, portanto é importante fazer boas escolhas logo de início.

Visão geral de um sistema de arquivos (partições)

Um sistema de arquivos Linux contém arquivos que são gravados em diretórios. Assim como em outros sistemas operacionais, diretórios no Linux podem conter outros diretórios porém, ao contrário de sistemas como o Microsoft Windows® que utiliza o conceito de sistemas de arquivos em diferentes letras (A:, C:, etc...), um sistema de arquivos Linux é acessado através de diretórios ligados ao diretório principal / (raiz).

Decidir uma estrutura de partições a ser adotada em sua instalação pode influenciar o desempenho de seu sistema bem como o funcionamento dos diferentes serviços rodando em seu equipamento.

Para termos uma idéia de como funciona a estrutura de diretórios do Linux, abaixo temos uma listagem desta estrutura, definida pela norma File Hierarchy Standard (FHS):

Diretório	Descrição
bin	Comandos essenciais
boot	Arquivos estáticos para boot do Linux
dev	Arquivos para dispositivos
etc	Configuração específica do equipamento
lib	Bibliotecas compartilhadas e módulos do kernel
media	Ponto de acesso para mídias removíveis
mnt	Ponto de acesso para sistemas de arquivos temporários
opt	Pacotes de programas adicionais
sbin	Comandos essenciais para administração e boot do Linux
srv	Dados para serviços oferecidos por este equipamento
tmp	Arquivos temporários
usr	Hierarquia secundária para programas adicionais
var	Dados variáveis

Decidindo o particionamento

Inicialmente devemos nos assegurar de que seu computador será capaz de inicializar normalmente. Em sistemas mais antigos há uma limitação na Bios que impede o boot quando os arquivos de inicialização do sistema operacional (neste caso /boot/vmlinuz e /boot/initrd) estão alocados acima de 1024 cilindros no disco rígido. Se você possui um equipamento assim, criar uma partição /boot para alocar estes arquivos é essencial.

Sua próxima preocupação é a criação de uma partição para swap (equivale ao arquivo de troca do Windows®). Inicialmente era comum alocar duas vezes a quantidade de memória RAM do equipamento, mas com os preços dos pentes de memória utilizar valores de 512Mb para estações e 1Gb para servidores é mais do que adequado a menos que você necessite especificamente um servidor para bases de dados ou algo parecido.

Deste ponto em diante a definição do particionamento deve levar em consideração o uso do equipamento. Para estações de trabalho a criação de uma partição / com todo o sistema operacional é indicada para novos usuários, porém recomendamos separar os dados de usuários em uma partição específica /home. O uso do disco em uma instalação deste tipo ocupa, em média de 2Gb a 3Gb de espaço em disco para o sistema operacional, porém é difícil prever como será o crescimento desta instalação e quais programas adicionais serão instalados. Em um disco de 20Gb é recomendável adotar um tamanho mínimo de 10Gb para a partição / e o resto para dados de usuários. Caso seu disco seja maior, aumente o tamanho da partição /, não sendo necessário que ela ocupe 50% de seu disco. Abaixo temos um particionamento recomendado para uma estação de trabalho em um disco com hd de 40Gb.

Partição	tamanho
/boot	64Mb (não ocupará mais do que 30Mb já com atualizações)
/	12Gb (logo após a instalação ocupará cerca de 1,5Gb)
/home	27Gb (vamos deixar os usuários gravarem seus dados)
swap	1Gb

Quando falamos em servidores, falta de espaço em disco pode ser catastrófico. Definir seu particionamento deve levar em consideração quais serviços serão providos pelo equipamento, sendo que abaixo indicamos uma lista dos diretórios e possíveis usos.

Diretório	Descrição
/boot	Arquivos de inicialização
/	Sistema operacional básico
swap	Paginação em disco
/home	Dados de usuários (serv arq) ou caixas postais de email
/var	Logs do sistema ou caixas postais de email
/usr	Ambiente gráfico
/srv	Dados de servidores como web e ftp

Independente do tipo de servidor a ser instalado, a criação de uma partição para armazenamento de logs é essencial. As primeiras três partições são o mínimo recomendado para servidores. Partições adicionais poderão ser criadas de acordo com suas necessidades podendo ser criadas uma ou várias delas.

Gerenciador de inicialização

Após a definição do particionamento, deveremos escolher qual gerenciador de inicialização será utilizado. Um gerenciador de inicialização é um software que lista todos os sistemas operacionais instalados em seu equipamento permitindo que você escolha qual deles que inicializar neste momento, o gerenciamento oferecido pelo Windows® é o NT Loader. Normalmente estão disponíveis o LILO e o GRUB como opções no Linux.

LILO – Linux Loader

Um gerenciador de boot que nasceu junto com o Linux propriamente dito, pode ser instalado no registro mestre de inicialização (Master Boot Record – MBR) ou no setor de inicialização de uma partição (Boot Sector Initializer – BSI). Pode ser instalado, ainda, em dispositivos removíveis como disquetes, cd's ou dispositivos USB.

O LILO permite o uso de até 16 combinações diferentes para inicialização bem como a inicialização de sistemas operacionais diferentes como Linux, Windows®, BSD e outros. Devido à sua construção, o LILO oferece certa flexibilidade de opções na inicialização do Linux, permitindo alterar o nível final de inicialização de texto para gráfico ou vice-versa ou ainda acessar o Linux em modo “mono usuário”. Sua configuração é armazenada no arquivo /etc/lilo.conf e deve ser gravada na MBR ou BSI sempre que for alterada executando-se o comando lilo.

GRUB - GRand Unified Bootloader

Assim como o LILO, o GRUB permite a inicialização de até 16 combinações entre diferentes sistemas operacionais ou configurações do seu Linux como opções de kernel ou modos como texto e gráfico e também pode ser instalado em mídias removíveis. A principal diferença entre os dois gerenciadores reside no fato de que qualquer alteração em seu arquivo de configuração que reside no /boot/grub/menu.lst não precisa ser gravada na MBR ou no BSI pois ela é lida automaticamente quando o sistema inicializa.

Outro ponto importante é que o GRUB permite a inspeção do disco durante o processo de inicialização, permitindo recuperar possíveis falhas em alguns casos sem a necessidade de mídias externas.

Instalando o Linux pela primeira vez

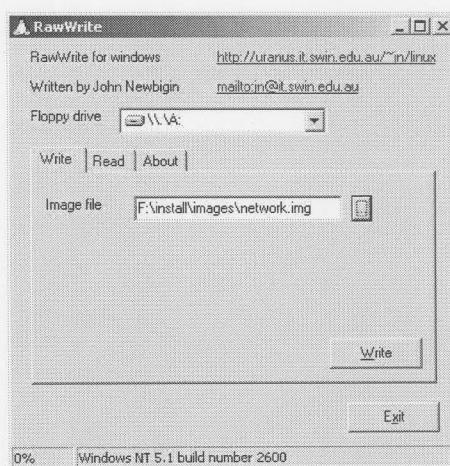
Nosso primeiro contato será com uma instalação básica, voltada para uso pessoal. Aqui temos por objetivo permitir a utilização dos recursos básicos do sistema, desmistificando um pouco aquele velho ditado de que “o Linux é um sistema operacional pouco amigável”. De forma simples costumo dizer que “muito pelo contrário, o Linux é extremamente amigável... o único detalhe é que ele escolhe muito bem seus amigos!”

A distribuição que faremos uso durante o curso será o Debian, versão 3.1 mais conhecida como Sarge. A maioria das distribuições disponibiliza duas versões de instalação: uma voltada para Desktops e outra volta para servidores incorporando todas as funcionalidades do Desktop às necessidades específicas para servidores. O Debian não faz distinção entre versões de instalação, sendo que o seu conjunto de cd's de instalação fornecem todos os aplicativos tanto para Desktop quanto para Servidores.

O processo de instalação do Debian ocorre de forma convencional ao sistemas operacionais atuais, sendo distribuído em cd's ou dvd's inicializáveis. Num processo de instalação local, basta que seu equipamento esteja configurado para inicialização pela unidade de cd/dvd e que você ligue o computador com a mídia de instalação em seu drive. Caso seu equipamento não seja capaz de inicializar pelo cd/dvd ou você queira realizar uma instalação com boot por disquetes ou remotamente através de um compartilhamento de rede ou servidor ftp, em ambos os casos será necessário criar disquetes de inicialização. Estes disquetes podem ser criados em uma estação Windows ou Linux, com os procedimentos abaixo.

Criação dos disquetes de instalação a partir do Windows®

Com a mídia de instalação em sua unidade de cd, normalmente a unidade D:, acesse a pasta D:\tools. Nesta pasta você encontrará os programas necessários para a criação dos disquetes de boot compactados. Caso seu Windows não possa descompactar estes arquivos, nesta pasta você encontrará o pacote unz512x3.exe com o compactador “Info Zip” gratuito. Com ele você poderá descompactar os arquivos necessários. Entre os programas disponíveis, o mais fácil de utilizar é o rwwrtwin. Descompacte-o em uma pasta temporária e dê dois cliques neste programa e será exibida a seguinte tela:



Para instalação via unidade de cd local, são necessários dois disquetes: um de inicialização fornecido pela imagem boot.img e outro de drivers de cd fornecido pelo arquivo cd-driver.img. Estes arquivos estão localizados na pasta \install\floppy da sua unidade de cd. Na caixa de seleção “Floppy drive” escolha a sua unidade de disquete, e na caixa

Instalando o Debian

A tela de inicialização exibe a opção de instalação (Enter por padrão) ou de ajuda que indica alternativas para a carga inicial do instalador. Digite `linux26` e pressione Enter para prosseguir.

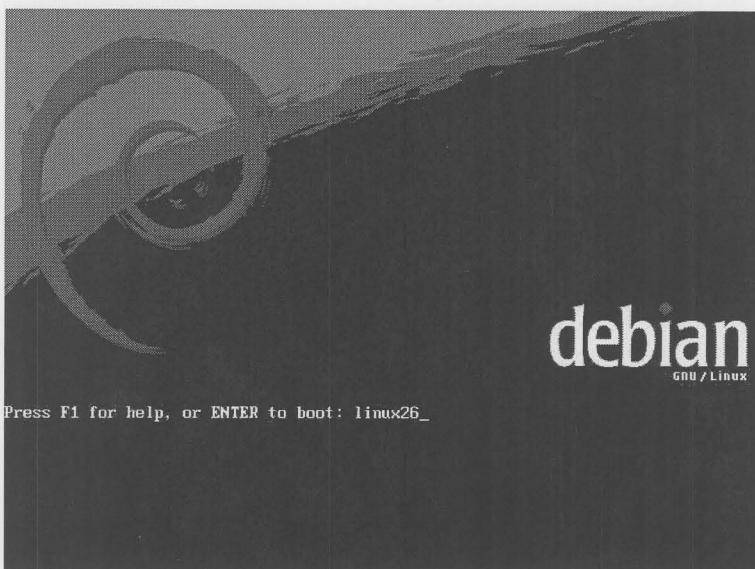


Fig 1 – dando boot pelo cd de instalação do Debian

Nosso próximo passo é identificarmos o idioma da instalação. Ao contrário de outros sistemas operacionais, escolher o idioma local como o português não prejudicará o funcionamento do Linux caso seja utilizado como servidor. As próximas duas telas destinam-se à seleção de idioma e layout de teclado. Selecione-os de acordo com seu equipamento.

No passo seguinte o Debian tenta configurar sua rede através de um servidor DHCP local. Caso isto não seja possível serão solicitados os seguintes dados:

- ip local
- máscara de rede
- default gateway
- dns primário

Após esta etapa será necessário configurar o nome do host. Este é o nome que será exibido em seu terminal toda vez que você utilizar uma linha de comando e ainda é o nome que fará parte de seu DNS identificando seu equipamento. O Debian já sugere um nome. Em nosso curso vamos chamar cada equipamento como “alunoN” onde N é o número do equipamento dentro da sala.

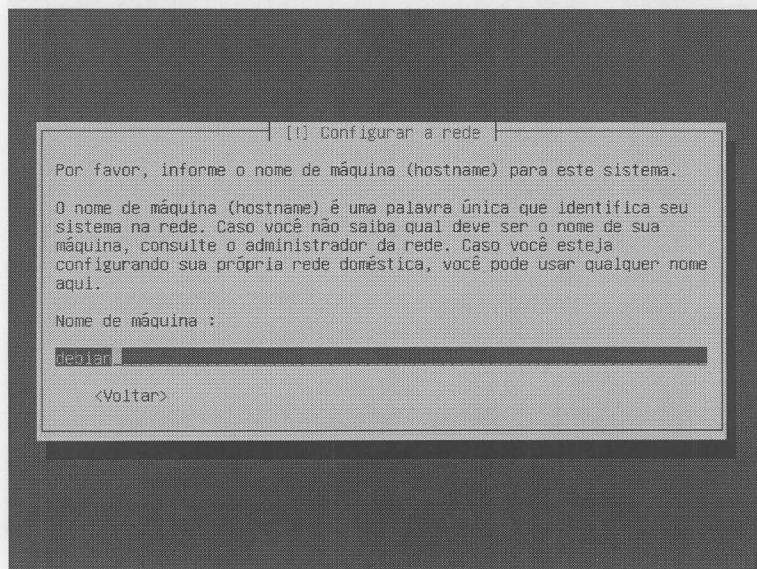


Fig 2 – nome do equipamento

A próxima etapa é a identificação de seu domínio DNS. Em nossa sala vamos substituir o “localdomain” por “sala2”.

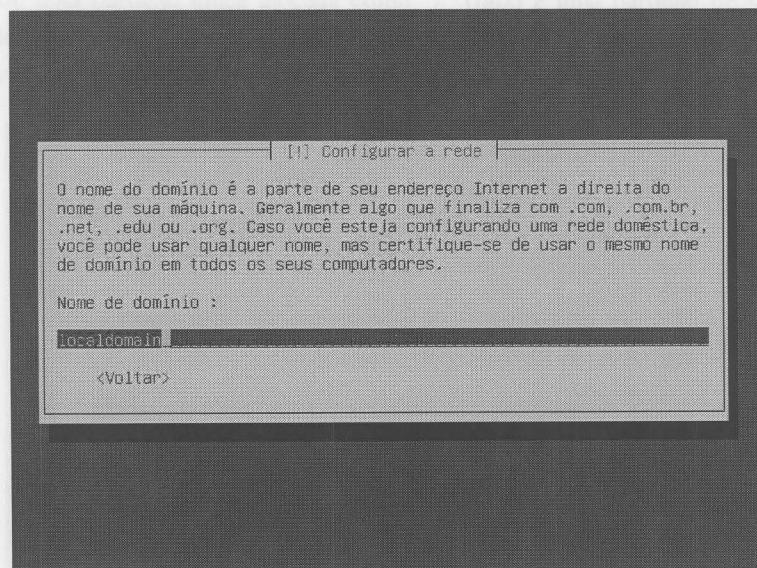


Fig 3 – nome do domínio DNS

Fig 5 – apagar todo o disco

Aqui é exibido o particionamento criado, pedindo sua confirmação para seqüência do processo de instalação.

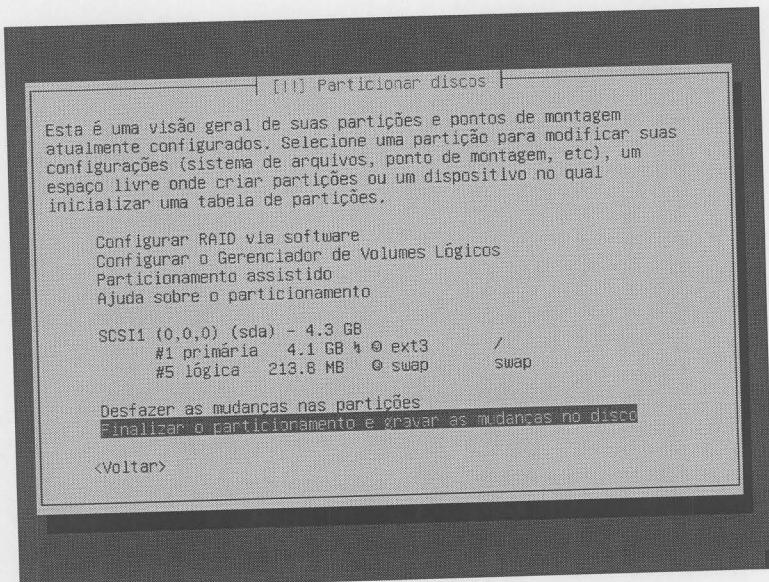


Fig 6 – finalizar particionamento automatico

Escolha “Finalizar particionamento e gravar as mudanças”. Mesmo confirmando o layout acima você ainda deverá confirmar o layout final na tela seguinte informando ao instalador para gravar as mudanças realizadas e que as partições podem ser formatadas.

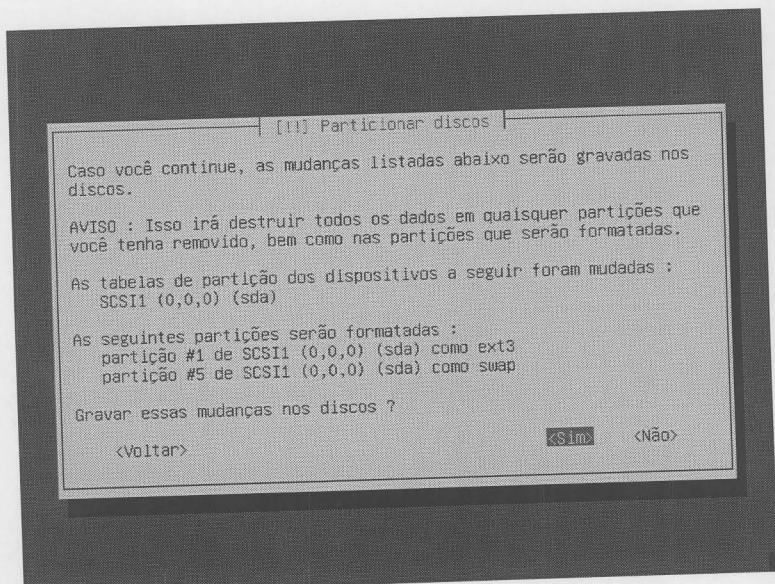


Fig 7 – seleção do perfil de programas

A partir deste momento serão copiados todos os programas necessários à instalação básica do Debian em seu equipamento, sendo necessária uma etapa complementar após o primeiro boot da máquina.

Após a tela de “boas vindas”, as etapas complementares incluem definição de senha do root, criação de usuários complementares e escolha do perfil a ser instalado. Nossa primeira etapa é identificar nosso fuso horário e se utilizamos hora local ou GMT.

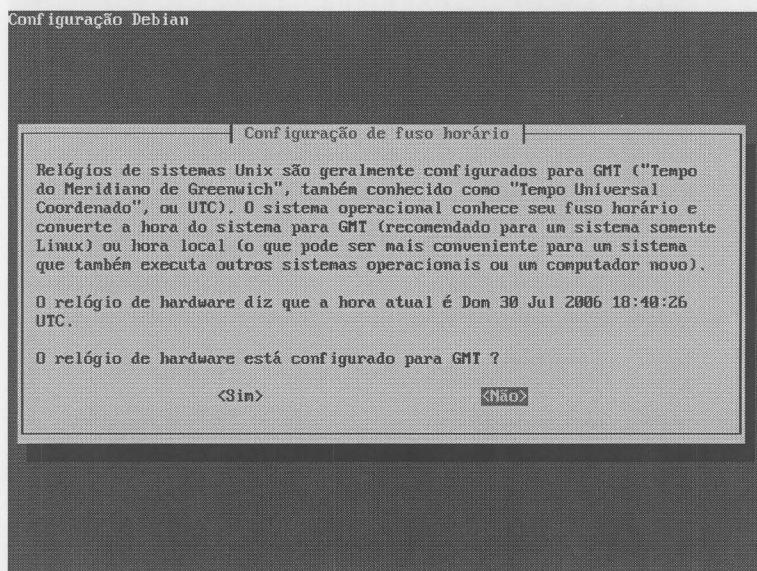


Fig 10 – identificação de horário local

A seguir devemos identificar nosso fuso horário. Como escolhemos idioma “português Brasil” o Debian indica que a escolha padrão deve ser fuso horário Brasil e permite que idiquemos qual a região a ser utilizada. Caso queira, você poderá alterar o fuso horário local selecionando outro e escolhendo uma das várias regiões possíveis.

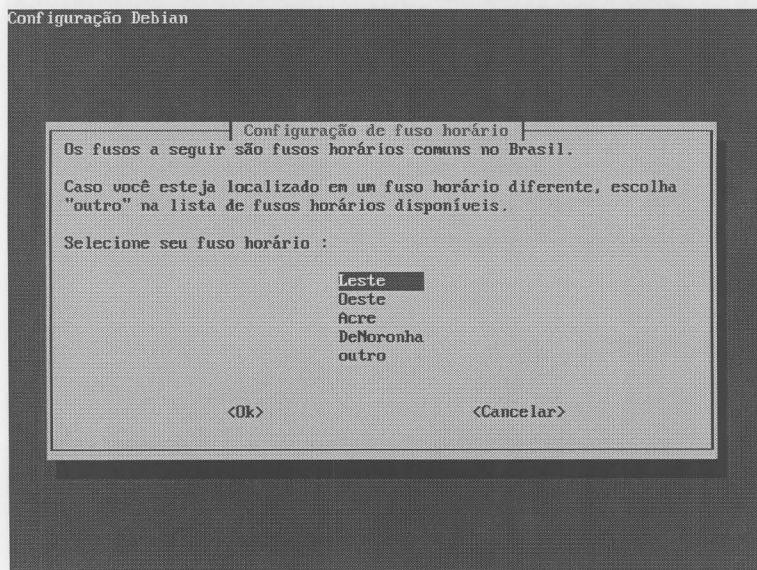


Fig 11 – escolha do fuso horário

Logo após a criação deste usuário o Debian tentará identificar se você possui algum cd de instalação e vai definir de onde serão baixadas as atualizações do sistema. Deixe o cd de instalação em sua unidade e prossiga ao passo seguinte.

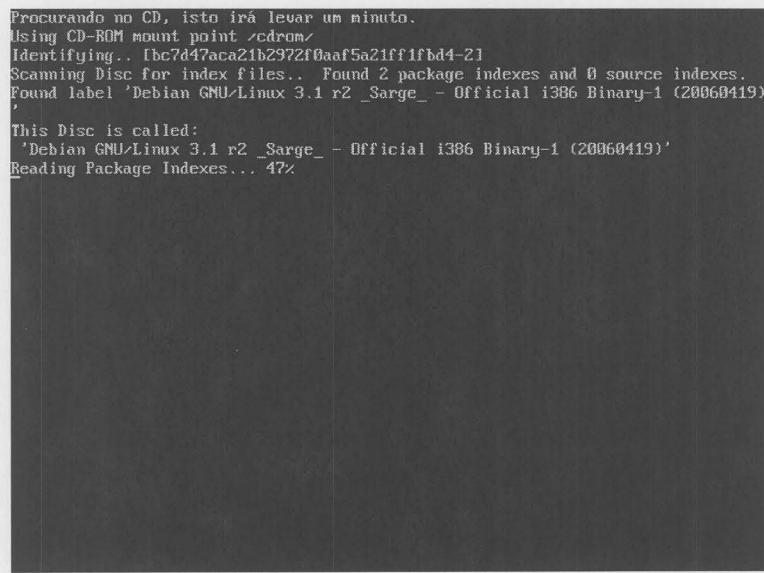


Fig 14 – detecção do cd e importação da lista de programas

Você será questionado se possui outro disco que deva ser escaneado. Responda que não e prossiga ao passo seguinte. O Debian tentará acesso ao repositório “security.debian.org” via internet e, caso consiga, as versões mais recentes dos pacotes selecionados serão instalados.

Agora devemos escolher o perfil que desejamos para nosso equipamento. Em nossas aulas vamos acessar vários recursos, inclusive a Internet, sendo assim vamos escolher apenas o perfil “Ambiente Desktop” e prosseguir com a instalação.

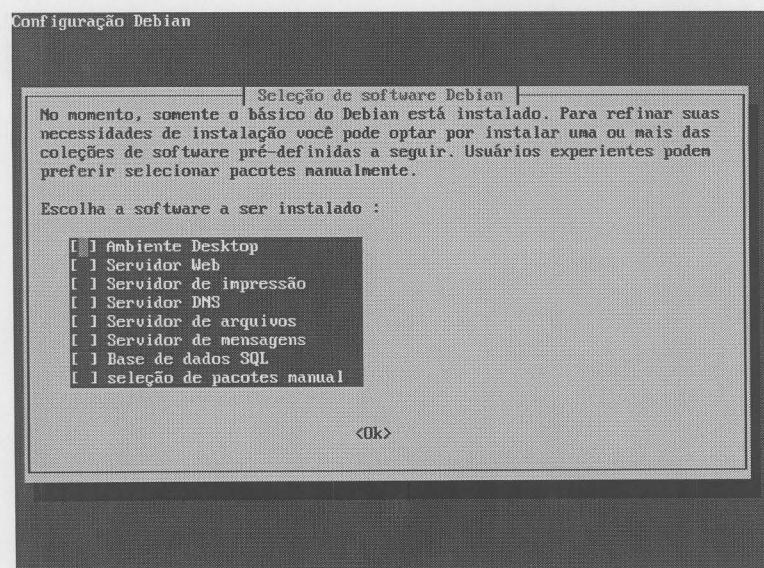


Fig 15 – escolha do perfil da instalação

Por padrão, o Debian envia automaticamente uma série de alertas via email sendo necessário préconfigurar o Exim (servidor de correio) para entrega local.

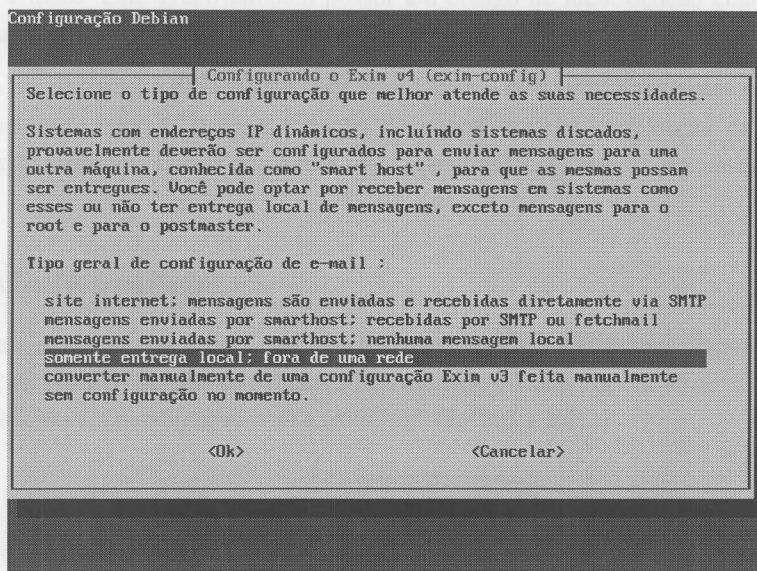


Fig 16 – configurando o Exim

Como criamos um usuário local (aluno) o Debian indicará que as mensagens geradas pelo sistema serão entregues a ele. Apenas aceite a configuração e nossa instalação será finalizada.

Conhecendo o Linux

Quando falamos em Linux, temos a lembrança de uma máquina rodando dos®, com várias linhas de comando e uma sintaxe terrível à nossa espera.

O Linux nada mais é do que um Sistema Operacional Moderno, com tantas funcionalidades e atrativos como qualquer outro S.O. Em suas características mais básicas o Linux é um sistema operacional multi-tarefa permitindo a execução de vários aplicativos e serviços simultaneamente e multi-usuário possibilitando o uso de um mesmo equipamento por vários usuários ao mesmo tempo, através de sessões remotas e locais.

Comparando estas características com outros S.O's, nos lembramos do MS Windows®, mas esta comparação só pode ser feita com servidores que possuam o serviço de terminal disponível, pois em sua concepção o Windows não é um sistema operacional multi-usuário.

Usuários e grupos

Por ter sido criado como um sistema multi-usuário, desde o princípio, o Linux incorpora vários mecanismos de segurança voltados para esta arquitetura. Cada usuário no Linux tem uma pasta pessoal com privilégios bem definidos e limitações claras. Para acesso ao Linux é necessário que o usuário forneça uma identificação (login) e senha de acesso (autenticação) sendo confrontados com os dados cadastrados no sistema e ainda ocorre a verificação se o acesso em questão pode ser realizado (autorização).

O processo de autenticação é necessário para garantir que o usuário não acesse dados aos quais não tem direito.

Sistemas de arquivos e Estrutura de diretórios

Uma das características mais personalizadas de qualquer Sistema Operacional é a sua estrutura de diretórios. Da mesma forma que arrumamos nossas roupas em armários, nos discos é necessário criar compartimentos específicos para cada tipo de arquivo ou por funcionalidades, definindo uma hierarquia que será seguida por todos os aplicativos do sistema.

Sistemas de arquivos

A preparação do equipamento passa inicialmente pela alocação de espaço em disco para a gravação de dados. Neste processo de alocação deve-se levar em consideração o uso final do equipamento a fim de garantir o espaço necessário para cada fim, por exemplo: para servidores de arquivos assim como em equipamentos para uso pessoal normalmente alocamos áreas enormes para dados e arquivos de usuários, em servidores de aplicações a parte destinada à instalação de aplicativos deve ser privilegiada.

O Linux é capaz de acessar, ler e/ou gravar vários tipos de sistemas de arquivos (formatações), sendo os mais comumente utilizados:

SISTEMA DE ARQUIVOS	DESCRIÇÃO
ext3	Sistema de arquivos utilizado como padrão na maioria das distribuições linux. Possui recursos de journaling (recuperação de falhas) que reduz o tempo de checagem do sistema em casos de falhas de energia ou reset do equipamento. É o sucessor do antigo ext2.
reiserfs	Sistema de arquivos padrão de algumas distros, possui recursos de journaling e melhor suporte a diretórios com grandes quantidades de arquivos. Possui ótimo desempenho.
proc	Sistema de arquivos virtual, sendo um espelho dos dados em processamento da memória do equipamento.
xfs	Sistema de arquivos com suporte a journaling desenvolvido pela Silicon e com melhor suporte a diretórios com grandes quantidades de arquivos. Possui ótimo desempenho.
nfs	Sistema de arquivos remoto (compartilhamento de rede) sendo acesso através de mapeamento do servidor na máquina local
vfat	Sistema de arquivos utilizado para acesso a mídias (partições ou disquetes) com formatação fat16/fat32 com suporte a nomes longos.
ntfs	Sistemas de arquivos padrão de equipamentos WNT,W2K e WXP. Somente disponível em modo read-only em praticamente todas as distribuições.

Estrutura de diretórios

A hierarquia de diretórios utilizada em sistemas Linux em nada lembra a estrutura de outros S.O's, tendo sido elaborada no desenvolvimento dos sistemas POSIX na década de 60.

Esta estrutura foi idealizada de forma a agrupar recursos semelhantes ou com mesma utilizada em pontos específicos do sistema, não deixando arquivos perdidos em outros subdiretórios. O esquema a seguir ilustra a estrutura atual do Linux, podendo ter pequenas variações entre as distribuições:

```
/  
|--bin  
|--boot  
|--dev  
|--etc  
|--home  
|--initrd  
|--lib  
|--mnt  
|--opt  
|--proc  
|--root  
|--sbin  
|--sys  
|--tmp  
|--usr  
|--var
```

Diretórios de um sistema
Linux, em árvore

O ponto inicial de nossa estrutura é o diretório “/”, também chamado de diretório raiz. A partir dele todos os outros subdiretórios são acessados.



O Linux não atribui letras a diferentes partições do hd. Para cada partição criada deve ser indicado um diretório de acesso que deverá ser criado na estrutura original do Linux.

Descrição da árvore de diretórios do Linux

Todos devem conhecer a utilização dos diretórios do Linux para podermos administrar nossos sistemas de forma correta. Muitas vezes a instalação de arquivos ou programas em locais diferentes do padrão pode resultar em funcionamento incorreto ou não funcionamento do seu sistema. Vamos ver para que servem os diretórios, com exemplos de seu uso.

/bin

Diretório com os programas e utilitários básicos destinados à inicialização do equipamento, necessários quando não há outros sistemas de arquivos disponíveis. Os aplicativos disponíveis neste diretório podem ser utilizados por qualquer usuário (root ou não) e podem servir também para operação normal do sistema. Seu conteúdo normalmente consiste de:

- Comandos comuns, essenciais à carga do S.O. ou armazenados aqui por compatibilidade com sistemas mais antigos. Exemplos: cat, chmod, cut, date, ls, etc. Além destes, temos ainda comandos destinados a backup e restauração do sistema como gunzip, gzip e tar. Por último temos alguns utilitários de rede, de uso geral como hostname, netstat e ping.

/boot

Neste diretório estão os arquivos destinados à inicialização do computador, o kernel (vmlinuz) propriamente dito e módulos (initrd) necessários à carga da máquina. Em geral este diretório é alocado em uma partição exclusiva para este fim com tamanho variando em 32Mb e 100Mb.

/dev

Todos os dispositivos instalados no seu equipamento necessitam de arquivos especiais que mapeiam a sua existência neste diretório. Exemplos:

- hda1: indica a primeira partição do disco IDE master da controladora primária.
- sda: primeiro disco scsi disponível no equipamento.

/etc

Todas as configurações utilizadas pelo seu equipamento e seus serviços normalmente são armazenadas neste diretório. Não deve ser colocado em partição separada. Seu conteúdo ocupa, em média de 10Mb a 15Mb atualmente. Exemplos:

- Configurações do equipamento: hostname, resolv.conf, inittab
- Configurações de serviços: email: postfix/main.cf; apache: httpd/conf/httpd.conf

/home

Diretórios pessoais de cada usuário. Normalmente todo usuário possui um diretório pessoal com o mesmo nome de seu login no servidor. Em servidores de arquivos é comum alocar esta pasta em uma partição ou disco separado.

/lib

Bibliotecas dinâmicas compartilhadas, necessárias ao funcionamento básico de quase todas as aplicações Linux, além dos módulos do kernel.

/mnt

Utilizada para acessar mídias externas como cd's, disquetes ou flash drives. Em alguns casos é comum encontrar aqui mapeamentos a servidores remotos. Em algumas distros como Suse e Debian pode-se ter estes mapeamentos a partir do diretório /media.

/opt

Destinado a acomodar programas desenvolvidos por terceiros e que normalmente não fazem parte das distribuições Linux, caso do Acrobat Reader®, do StarOffice e alguns outros aplicativos.

/proc

Diretório utilizado para acessar o sistema de arquivos virtual com informações do estado de processamento de sua máquina. Os arquivos aqui não estão fisicamente em seu hd, mas sim em áreas protegidas da memória, sendo alguns destes arquivos podem ser alterados de forma a modificar o funcionamento do seu sistema Linux. Exemplos:

- cpufreq: dados do processador de seu equipamento;
- meminfo: dados da memória como total instalado, total disponível e como está sua utilização;
- ioports: identificação das portas io em uso no seu equipamento.

/root

Diretório pessoal do usuário “root”. Assim como /etc não deve ser colocado em partição separada.

/sbin

Arquivos executáveis destinados à inicialização do equipamento bem como para administração, acrescentando funcionalidades aos aplicativos existentes em /bin. Exemplos:

- ifconfig: configuração das interfaces de redes;
- shutdown: desligamento do sistema;
- route: definição de rotas.

/sys

Assim como /proc é um sistema de arquivos virtual que tende a substitui-lo provendo novos recursos e funcionalidades. Aqui encontramos uma estrutura de diretórios que mostra os recursos instalados em seu equipamento.

/tmp

Arquivos temporários criados por aplicativos iniciados por qualquer usuário ou por serviços disponíveis no equipamento. Normalmente estes arquivos são removidos automaticamente ao término da aplicação porém, caso algum arquivo não seja removido, apenas o usuário criador ou root poderão removê-los.

/usr

Estrutura secundária de diretórios replicando toda a estrutura original. Inicialmente foi utilizada como local de instalação para programas que não faziam parte da distribuição original. A intenção para sua criação é que qualquer novo aplicativo ou serviço que seja instalado não cause conflito com os recursos existentes, podendo ser facilmente removidos sem afetar a estrutura original.

/var

Dados variáveis resultantes de aplicativos em uso como cache de instalação de programas ou do acesso à Internet, arquivos de controle de processos em memória como em /var/run/pid.

Pontos de montagem

Toda e qualquer mídia (fixa ou removível) somente é acessada a partir de diretórios. Caso um disco seja particionado de forma a compartimentar os dados do sistema, o acesso a estas partições deve ser configurado em `/etc/fstab`. Neste arquivo são informados os dados de qual partição será acessada, sua formatação e a partir de qual diretório os dados estarão disponíveis. Um exemplo é particionar um disco em `/` (raiz) com 8Gb e `/home` com 32Gb, teremos as seguintes linhas no arquivo `/etc/fstab`:

```
# dispositivo diret format opções dump seq
/dev/hda1      /      ext3  defaults  0      1
/dev/hda2      /home  ext3  defaults  0      2
```

Identificamos acima que existem duas partições neste disco, a primeira é acessada pela raiz do sistema “`/`”, e a segunda a partir do diretório “`/home`”. Comparando com os sistemas mais comuns no mercado, isto é algo como a letra “`c`” para a primeira partição e a letra “`d`” para a segunda, sendo que esta se destina a guardar as pastas de usuários, digamos assim o “Document & Settings”. As outras opções serão vistas em detalhes posteriormente.

Tipos de usuários

Como padrão, todo sistema operacional voltado a ambientes corporativos possui separação clara de privilégios para a execução de programas e gravação de dados. Isto é feito com a identificação de usuários que poderão ou não ter acesso a estes recursos mas alguém deve poder ter acesso a todos eles e ainda poder definir que terá estas características.

No Linux o usuário “`root`” é o responsável por estas definições. Ele é o administrador do sistema operacional instalado e pode realizar qualquer tarefa administrativa que seja necessária. Qualquer outro usuário cadastrado será considerado um usuário comum. Há ainda os usuários conhecidos como “usuários de sistema”, um exemplo deles é: `named`, `shutdown`, `bin`, etc... Estes usuários não possuem privilégios adicionais em relação a outro usuário qualquer criado no sistema. Estes são chamados usuários de sistema apenas pelo fato de possuírem um código de identificação em uma faixa restrita, destinado exclusivamente a eles, ou seja, são usuários comuns, sem privilégios, apenas com um código de identificação em uma faixa pré-definida.

Qualquer outro usuário cadastrado no sistema é considerado apenas um usuário. Não possui privilégios adicionais ou `uid's` diferenciados. Seus direitos administrativos se limitam à sua pasta pessoal e, no resto da instalação, ele pode apenas ler ou alterar o que for explicitamente permitido a ele.

Tipos de arquivos e permissões

Cada arquivo e diretório no sistema de arquivos do Linux possui permissões específicas e propriedades que indicam a que usuário e grupo pertence o arquivo. É através destas características que podemos encontrar dados de quem podem acessar esta ou aquela pasta além de tornar inteligível para nós estes dados. Internamente o Linux armazena estas informações através de códigos, que são consultados e convertidos para nomes todas as vezes que realizamos uma consulta.

Tipos de arquivos

O tipo de arquivo é armazenado junto com as permissões que o mesmo tem e normalmente indica para que serve aquele arquivo. Isto pode ser consultado pelo comando `ls`, mas é muito pouco utilizado. Pela listagem abaixo podemos identificar alguns destes tipos:

```
# ls -l ~
-rw-r--r-- 1 root root 145 Jun 13 2005 .bash_completion
-rw----- 1 root root 61 Jun 25 09:55 .bash_history
-rw-r--r-- 1 root root 24 Jun 13 2005 .bash_logout
-rw-r--r-- 1 root root 106 Jun 13 2005 .bash_profile
-rw-r--r-- 1 root root 226 Jun 13 2005 .bashrc
-rw-r--r-- 1 root root 233 Jun 13 2005 .cshrc
drwx----- 2 root root 4096 Mai 17 20:31 drakx/
drwx----- 2 root root 4096 Jun 25 17:48 .ssh/
#
```

A parte desta listagem que nos interessa aqui é a primeira coluna. Aqui vemos os tipos mais comuns, arquivos e diretórios. Os tipos de arquivos estão descritos na seguinte tabela:

TIPO	Descrição
-	Arquivo comum
b	Dispositivo de bloco (armazenamento)
c	Dispositivo de caracter (serial)
d	Diretório
l	Link simbólico
s	Socket de comunicação entre processos
p	Pipe de comunicação entre processos

Permissões

As permissões são as informações que indicam que tipo de acesso pode ser realizado em determinado diretório ou arquivo. No Linux as permissões são atribuídas em três níveis específicos, englobando os usuários da seguinte forma:

- **Usuário dono do arquivo:** Normalmente é aquele que criou o arquivo ou diretório. Pode ser chamado como owner. Se necessário, pode ser alterado apenas pelo root.
- **Grupo dono do arquivo:** Normalmente é o grupo básico do usuário criador. Da mesma forma que o dono, somente pode ser alterado pelo root.
- **Outros:** Usuários que não se enquadram como usuário ou grupo dono do arquivo.

As permissões devem ser interpretadas em três conjuntos, onde cada um possui combinações das permissões “r”, “w” ou “x”. Veja como elas são identificadas para cada grupo:

grupo	
rw- r-- ---	1 root shadow
usuário	
outros	

Vamos ver como avaliar o conjunto de permissões de um arquivo recém criado:

```
# touch /tmp/exemplo
# ls -l /tmp/exemplo
-rw-r--r-- 1 root root 0 2006-05-30 03:15 named.conf
```

Pela saída acima temos a seguinte interpretação:

- exemplo é um arquivo (primeira posição da listagem é um “-”);
- o arquivo pertence ao usuário root e ao grupo named;
- o usuário dono do arquivo (root) pode ler e alterar o arquivo;
- o grupo dono do arquivo (root) apenas pode ler o conteúdo do arquivo;
- outros usuários (que não sejam o usuário root ou pertençam ao grupo root) podem ler o conteúdo do arquivo.

Alterando permissões

Uma vez definidas as permissões de um objeto (arquivo ou diretório), estas podem ser alteradas pelo seu dono ou pelo usuário root.

Há três formas de alterar as permissões, a primeira é pelo método diferencial onde indicamos quais as alterações devem ser realizadas em operações de acréscimo (+) ou retirada (-) com o uso de letras indicando as permissões “r”, “w” e “x”. A segunda forma é pela atribuição direta onde as permissões finais são indicadas, ainda com o uso de letras. O terceiro método é o octal onde as permissões são convertidas em valores binários e atribuídas aos arquivos ou diretórios de forma direta. Todas estas alterações são realizadas pelo comando chmod indicando as permissões e objetos a serem alterados

Definindo permissões pelo método diferencial

Considerando o arquivo /tmp/exemplo listado acima vamos alterar suas permissões de forma que o resultado final seja rw-rw-rw-:

```
# chmod g+w,o+w /tmp/exemplo
# ls -l /tmp/exemplo
-rw-rw-rw- 1 root root 4096 2006-05-30 03:15 exemplo
```

Definindo permissões pelo método de atribuição direta

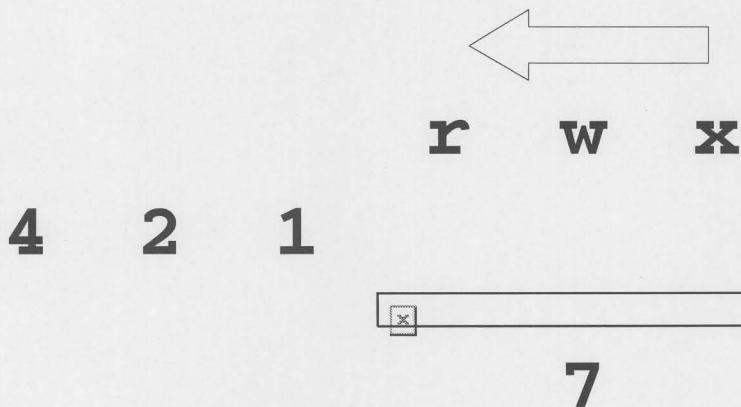
A segunda forma é pelo método de atribuição direta, onde as permissões finais são indicadas ao comando chmod, vamos alterar estas permissões para r-xr-x-x:

```
# chmod u=r,g=r,o=x /tmp/exemplo
# ls -l /tmp/exemplo
-r-xr-x--x 1 root root 4096 2006-05-30 03:15 exemplo
```

Definindo permissões pelo método octal

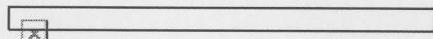
O último método de atribuição, o octal, exige a conversão de cada permissão em valores octais (binários), somando o resultado de cada conjunto para posterior atribuição ao arquivo.

Começando da direita para a esquerda, cada letra recebe um valor: “x” vale 1, “w” vale 2 e “r” vale 4. Para conhecermos a permissão que deve ser atribuída, somamos os valores definidos, chegando ao resultado final.



Caso uma determinada posição não seja definida, o seu valor correspondente não deverá ser somado:

r	w	x
4	-	1



5

Considerando um exemplo mais prático, vamos ver como definir as permissões abaixo a um arquivo:

rwx	r-x	r-x
<hr style="border-top: 1px solid black;"/>	<hr style="border-top: 1px solid black;"/>	<hr style="border-top: 1px solid black;"/>
7	5	5

Utilizando este exemplo, a atribuição da permissão pelo chmod deve ser executada da seguinte forma:

```
# chmod 755 /tmp/exemplo
# ls -l /tmp/exemplo
-rwxr-x--x 1 root root 4096 2006-05-30 03:15 exemplo
```

Permissões especiais

Além das permissões de leitura, escrita e execução temos ainda três permissões especiais, cada uma aplicada a um propósito diferente, que são: “stick bit”, “sgid” e “suid”. As permissões especiais são atribuídas sempre em conjunto com a permissão de execução.

Em modo octal normalmente são atribuídas permissões no estilo NNN, onde cada posição atua sobre um conjunto de permissões, conhecidamente “usuário”, “grupo” e “outros”. As permissões especiais são atribuídas uma posição antes do padrão: ENNN, onde E segue o mesmo padrão de definição de permissões. Veja mais detalhes a seguir.

Stick Bit

Normalmente é aplicado a diretórios compartilhados ou que arazenam dados temporários de usuários em equipamentos compartilhados como servidores de aplicações. Sua função é garantir que qualquer informação gravada nestes diretórios somente sejam removidas pelo proprietário ou pelo usuário root, garantindo que aplicações em funcionamento não sejam finalizadas por terem seus arquivos de controle removidos inadvertidamente por qualquer usuário. Esta permissão é atribuída por padrão ao diretório /tmp, devido ao fato de este armazenar os dados temporários de qualquer aplicação em execução no equipamento.

Stick Bit atua no conjunto de permissões outros por garantir que qualquer arquivo ou diretório de qualquer usuário seja protegido.

4	2	1
rwx	rwx	rwx
421	421	421
7	7	7

Para atribuir esta permissão especial vamos acrescentar antes das permissões comuns o valor que define o **stick bit**, neste caso o valor 1. O comando resultante para esta permissão é:

```
# chmod 1777 diretório
```

SGID

Nesta permissão o intuíto é garantir que o grupo do arquivo ou diretório seja preservado em qualquer operação de criação de dados ou diretórios. Sua utilidade está em ser definida para diretórios ou executáveis quando manter o grupo original for essencial. No caso de diretórios com esta permissão, qualquer novo arquivo ou diretório criado dentro dele também pertencerá ao grupo inicial. No caso de executáveis, qualquer novo dado criado com este programa pertencerá ao grupo inicial.

4	2	1
rwx rwx rwx		
421 421 421		

Para atribuir esta permissão especial vamos acrescentar antes das permissões comuns o valor que define o **SGID**, neste caso o valor 2. O comando resultante para esta permissão é:

```
# chmod 2777 diretório
```

SUID

Aplicado a executáveis, sua função é garantir que, durante a execução do comando, o usuário logado assuma a identidade do dono do programa, dando direitos especiais para leitura ou gravação. Se aplicado a diretórios, qualquer novo arquivo ou diretório pertencerá ao usuário inicial.

4	2	1
rwx rwx rwx		
421 421 421		

Para atribuir esta permissão especial vamos acrescentar antes das permissões comuns o valor que define o **SGID**, neste caso o valor 2. O comando resultante para esta permissão é:

```
# chmod 4777 arquivo
```

Links

Em sistemas POSIX temos dois tipos de links, os físicos, também chamados de “hard links” e os links simbólicos ou “symlinks”.

Hard Links

Hard Links indicam o nome de um arquivo propriamente dito. Ao criar um arquivo qualquer com um editor de textos, ou copiá-los de um diretório para outro estamos criando links físicos. Em situações especiais pode ser necessário criar novos links físicos para arquivos existentes. Isto pode ser feito com o comando `ln`. Vamos criar um arquivo de teste e criar novos links para ele:

```
# cd ~/tmp
# echo "link físico" > testelinkfisico
# ls -li testelinkfisico
980836 -rw-r--r-- 1 root root 12 Jun 2 16:47 testelinkfisico
```

Criando um novo link físico para este arquivo existente:

```
# ln testelinkfisico novolinkfisico
# ls -li *linkfisico
980836 -rw-r--r-- 2 root root 12 Jul 2 16:47 novolinkfisico
980836 -rw-r--r-- 2 root root 12 Jul 2 16:47 testelinkfisico
```

Vemos na primeira coluna o “inode” (unidade física de alocação) onde o arquivo está gravado. Nesta listagem percebemos que os dois arquivos na verdade são o mesmo, pois estão gravados no mesmo inode. Qualquer alteração que seja feita em um dos arquivos será refletida no outro. Percebemos ainda que o número de links também foi incrementado para 2.

De forma simples, um “link físico” nada mais é do que um nome de arquivo que aponta diretamente para uma área em disco. O conteúdo em disco será preservado enquanto houver um hardlink que esteja associado ao inode. Um arquivo com vários links físicos somente será apagado do disco quando todos os links forem removidos.

SymLinks

Links simbólicos ou symlinks são nomes de arquivos que apontam para outros arquivos ou diretórios. Funcionam como apelidos facilitando o acesso a diretórios e programas ou criando nomes reduzidos para arquivos específicos como bibliotecas compartilhadas do sistema.

Ao contrário dos hardlinks, se o arquivo para o qual o symlink aponta for removido, ele perde sua funcionalidade, ficando totalmente sem uso. Um symlink não preserva os dados em disco pois apenas aponta para um nome e não para uma área em disco. Nestas situações, ao rodar um comando `ls`, o symlink aparecerá em branco piscando sobre fundo vermelho, podendo variar de acordo com o terminal em uso.

Para criar links simbólicos, utilizamos o mesmo comando `ln`, como segue:

```
# ln -s testelinkfisico linksimbolico
# ls -li *link*
980847 1rwxrwxrwx 1 root root 15 Jul  2 16:58 linksimbolico -> testelinkfisico
980836 -rw-r--r-- 2 root root 12 Jul  2 16:47 novolinkfisico
980836 -rw-r--r-- 2 root root 12 Jul  2 16:47 testelinkfisico
# cat linksimbolico
link físico
```

Vemos que o inode onde está gravado o arquivo e o symlink são diferentes. São realmente dois arquivos diferentes porém com mesmo conteúdo, como pode ser visto pela saída do comando `cat`.

Metacaracteres

Ao executar comandos como `ls` ou `cp`, muitas vezes utilizamos metacaracteres (caracteres curinga – * e ?) para facilitar nossas tarefas. Os metacaracteres acrescentam funcionalidades a estes comandos, permitindo especificar parâmetros adicionais ou indicar condições para sua execução.

Símbolo ?

O símbolo ?, de forma similar aos sistemas Microsoft®, substitui um único caracter porém aqui este caracter deve existir (em um sistema Windows®, ? representa 0 ou 1 caracter). Abaixo temos a saída do comando `ls` utilizando ?:

```
# ls -d /etc/rc?.d
/etc/rc0.d /etc/rc2.d /etc/rc4.d /etc/rc6.d
/etc/rc1.d /etc/rc3.d /etc/rc5.d /etc/rcs.d
```

Símbolo *

O símbolo * substitui qualquer quantidade de caracteres (0 ou mais). Uma consulta similar à executada anteriormente, porém com o uso de *:

```
# ls -d /etc/rc*.d
/etc/rc.d      /etc/rc1.d   /etc/rc3.d   /etc/rc5.d   /etc/rcS.d
/etc/rc0.d      /etc/rc2.d   /etc/rc4.d   /etc/rc6.d
# ls -d /etc/*conf
/etc/aide.conf          /etc/ld.so.conf        /etc/pwdb.conf
/etc/conf.linuxconf     /etc/ldap.conf        /etc/resolv.conf
/etc/dhcpd.conf         /etc/libuser.conf    /etc/sysctl.conf
/etc/gpm-root.conf      /etc/linuxconf       /etc/syslog.conf
/etc/grub.conf          /etc/logrotate.conf  /etc/updatedb.conf
/etc/host.conf          /etc/mkinitrd.conf   /etc/usb-mount.conf
/etc/inetd.conf          /etc/modprobe.conf   /etc/xinetd.conf
/etc/initlog.conf        /etc/named.conf      /etc/yp.conf
/etc/kernel-postinstall.conf /etc/nsswitch.conf
/etc/krb5.conf          /etc/ntp.conf
```

No primeiro exemplo o * substituiu qualquer caractere, zero (foi o caso de rc.d) ou mais vezes. No segundo exemplo sempre houve mais de um caractere na saída. Outros usos do * são similares a outros sistemas operacionais, como na remoção de arquivos:

```
# rm -f /tmp/*.txt
```

Neste exemplo somente serão removidos os arquivos do diretórios /tmp que terminem com “.txt” em seus nomes.

Agrupador colchetes – []

Permite especificar condições adicionais como faixa de caracteres que devem ser satisfeitas. Por exemplo, solicitando uma listagem de todos os arquivos ou diretórios que comecem com letras minúsculas do diretório /etc:

```
# ls -d /etc/[a-z]*
```

Ou apenas os arquivos ou diretórios que tenham números em seus nomes, neste mesmo diretório:

```
# ls -d /etc/*[0-9]*
```

Ainda é possível negar uma determinada característica, como por exemplo qualquer arquivo ou diretório que não comece com letras maiúsculas. Para isto devemos utilizar o caractere ^ dentro da condição de pesquisa:

```
# ls -d /etc/[^A-Z]*
```

Uma condição importante a observar é que o uso de colchetes permite especificar apenas um caractere que será substituído, ou seja, em cada um destes exemplos apenas um caractere dentro da faixa especificada será combinado com a condição externa (*).

Agrupador chaves – {}

Permite especificar qualquer número de caracteres que serão combinados com o padrão fora das chaves. Ao contrário de colchetes, aqui podemos indicar trechos de nomes que queremos pesquisar ou tratar, por exemplo:

Aqui foram listados todos os objetos que começem com “pass” ou com “print” ou com “gr”. Outro exemplo similar, porém tratando os arquivos que terminem com “conf” ou com “rc”:

```
# ls -d /etc/*{conf,rc}
/etc/aide.conf          /etc/ld.so.conf      /etc/rc
/etc/bashrc              /etc/ldap.conf      /etc/resolv.conf
/etc/conf.linuxconf      /etc/libuser.conf   /etc/screenrc
/etc/dhcpd.conf          /etc/linuxconf     /etc/sysctl.conf
/etc/gpm-root.conf       /etc/logrotate.conf /etc/syslog.conf
/etc/grub.conf            /etc/mail.rc        /etc/updatedb.conf
/etc/host.conf             /etc/mkinitrd.conf /etc/usb-mount.conf
/etc/inetd.conf           /etc/modprobe.conf /etc/wgetrc
/etc/initlog.conf         /etc/named.conf    /etc/xinetd.conf
/etc/inputrc               /etc/nsswitch.conf /etc/yp.conf
/etc/kernel-postinstall.conf /etc/ntp.conf
/etc/krb5.conf             /etc/pwdb.conf
```

Um aspecto interessante deste agrupador é que pode ser utilizado com comandos como touch e mkdir:

```
# touch arq{1,2,3,4,5}
# mkdir aula{1,2,3,4,5}
# ls -dF arq* aula*
arq1 arq2 arq3 arq4 arq5 aula1/ aula2/ aula3/ aula4/ aula5/
```

Instalação de programas

Em todo sistema, a instalação de programas é uma tarefa administrativa e deve ser executada para incluir novos recursos ou para atualizações de segurança. Em outros sistemas normalmente recebemos os programas a serem instalados em um pacote pré-empacotado como um “.msi” ou um executável que descompacta seus arquivos e os copia para os diretórios devidos. Nestes ambientes cada programa é responsável por surpreender todas as suas necessidades de bibliotecas e programas adicionais necessários ao seu pleno funcionamento.

Em sistemas Linux isto é bem diferente. Primeiro não é comum a distribuição de programas de instalação executáveis. Eles existem, porém são fornecidos por fabricantes específicos para programas que não estão disponíveis em nenhuma distribuição, que é o caso do servidor Domino da Lotus ou o plugin FlashPlayer para navegador. Em 98% dos casos os programas a serem instalados em seu Linux são previamente empacotados em formatos como “rpm”, desenvolvido pela RedHat e utilizado pelo Mandriva e várias outras distribuições, ou “deb” desenvolvido pelo Debian e também utilizado por várias distribuições.

O padrão rpm adotado pelo Mandriva permite a instalação de programas e a consulta de várias informações sobre os pacotes instalados.

rpm

O rpm (RedHat Package Manager) define uma série de regras para a instalação de novos programas ou para a sua atualização. Dentro de cada pacote de instalação são definidas informações como pré-requisitos ou tarefas a serem realizadas durante a instalação dos pacotes. Suas principais opções são:

rpm [opções] pacote1 pacote2 ... pacoteN	
OPÇÃO	Descrição
-e	Remove um pacote instalado
-h	Exibe um caractere “#” para acompanhamento da instalação
-i	Instala pacotes ainda não instalados
-q	Questiona se o pacote está instalado, ou questiona informações adicionais do pacote instalado: -qa: lista todos os pacotes instalados -qi: descrição do pacote instalado -qf: quem instalou determinado arquivo -ql: lista todos os arquivos instalados pelo pacote -qlv: lista detalhada dos arquivos instalados pelo pacote
-U	Instala novos pacotes ou atualiza pacotes já instalados. Deve ser utilizado com cuidado no caso de pacotes como o kernel.
-v	Exibe mais detalhes durante a instalação de um pacote
-V	Verifica um pacote instalado

Exemplos:

Verifica se o pacote coreutils está instalado:

```
# rpm -q coreutils  
coreutils-5.2.1-8mdk
```

Qual o pacote responsável pela instalação do programa /sbin/shutdown:

```
# rpm -qf /sbin/shutdown  
SysVinit-2.86-3mdk
```

Quais os arquivos instalados pelo pacote less:

```
# rpm -ql less
```

Instalando um novo pacote:

```
# rpm -ivh /exemplos/tree-1.5.0-1mdk.i586.rpm  
A preparar...  
 1:tree          ##### [100%]  
 1:tree          ##### [100%]
```

Para a instalação acima tivemos de informar o nome completo do arquivo a ser instalado, porém depois de instalado podemos realizar todas as operações apenas seu nome principal, sem nos importarmos com versão ou arquitetura.

```
# rpm -q tree  
tree-1.5.0-1mdk
```

Removendo um pacote instalado:

```
# rpm -e tree
```

Listando uma série de pacotes instalados:

```
# rpm -qa | grep vim  
vim-enhanced-6.3-21mdk  
vim-common-6.3-21mdk  
vim-minimal-6.3-21mdk
```

O mesmo resultado pode ser obtido com:

```
# rpm -qa "vim*"
```

Gerenciamento de usuários e grupos

Por ter sido desenvolvido como um sistema multi-usuário desde o início, o Linux já traz em suas características básicas o controle de acesso simultâneo, definindo de forma clara os privilégios e restrições a que qualquer usuário está sujeito no momento do seu acesso. Tais características são definidas no momento da criação do usuário ou posteriormente durante a administração cotidiana do servidor, na movimentação de usuários entre departamentos da empresa. Entre as muitas tarefas do sistema, o gerenciamento de usuários envolve:

- Definição de políticas de senhas, complexidade e prazos de validade;
- Criação de contas para usuários e grupos;
- Remoção ou bloqueio de contas sem uso;
- Testes de segurança para validar as políticas adotadas.

Como em, praticamente, todas as tarefas do Linux há várias interfaces de gerenciamento que visam facilitar estas tarefas, mas há muitas situações onde as ferramentas por linha de comando são mais eficazes ou podem ser as únicas disponíveis.

Comandos para gerenciamento

Os comandos utilizados no gerenciamento de usuários e grupos são:

COMANDO	DESCRIÇÃO
<code>useradd</code>	Cria usuário
<code>userdel</code>	Remove usuário
<code>usermod</code>	Modifica dados de uma conta de usuário
<code>passwd</code>	Define/troca a senha de usuários
<code>groupadd</code>	Cria contas de grupo
<code>groupdel</code>	Remove grupo
<code>groupmod</code>	Modifica dados do grupo
<code>gpasswd</code>	Gerência membros e senhas de grupos

Criando e removendo usuários e grupos

O processo de criação de usuários pode combinar vários parâmetros como diretório pessoal, comentários e outros itens mas os dados essenciais para a criação de uma conta é o login a ser utilizado pelo usuário e a definição de uma senha de acesso. Isto é feito com os comandos `useradd` e `passwd`:

```
# useradd usuario1
# passwd usuario1
Changing password for user usuario1.
New UNIX password:
Retype new UNIX password:
```

Para que um usuário possa ter acesso ao Linux é obrigatória a definição de uma senha inicial, mesmo que forcemos a troca no primeiro login.

A criação de um usuário força a criação de seu diretório pessoal dentro de /home e de um grupo pessoal, que pode ser consultado abaixo:

```
# grep usuario1 /etc/passwd /etc/shadow /etc/group
/etc/passwd:usuario1:x:500:500::/home/usuario1:/bin/bash
/etc/shadow:usuario1:$1$x3hSN/IB$/mBsf7/es7WKZyLZPa7s01:13332:0:99999:7:::
/etc/group:usuario1:x:500:
```

Para remover este usuário, basta rodar o comando userdel?

```
# userdel -r usuario1
```

De forma similar, a criação de grupos é realizada com o comando groupadd, como segue:

```
# groupadd informatica
```

E sua remoção pode ser feita com groupdel.

Comandos e parâmetros mais utilizados

Vários comandos de gerenciamento de usuários e grupos compartilham as mesmas opções, tornando o seu uso mais simples. Veja as opções mais utilizadas:

COMANDO	DESCRIÇÃO
useradd usermod	-c Define comentários como nome completo e outros
	-d Define o diretório pessoal
	-g Define o grupo básico do usuário
	-G Define os grupos adicionais do usuário
	-l Renomeia uma conta de usuário
	-s Define o shell do usuário
userdel	-r Remove usuário e seu diretório pessoal
passwd	-d Define a senha de usuário como vazia

COMANDO	DESCRIÇÃO
	<ul style="list-style-type: none"> -l Trava a senha do usuário -u Destrava a senha do usuário -S Exibe o status da senha do usuário
groupmod	<ul style="list-style-type: none"> -n Renomeia um grupo
gpasswd	<ul style="list-style-type: none"> -a Adiciona usuário ao grupo -d Remove usuário do grupo

Exemplos:

Criando um usuário com diretório pessoal em /usr/local/usuario

```
# useradd -d /usr/local/usuario usuario
```

Criando usuário com grupo básico informática:

```
# useradd -g informatica usuario1
```

Criando usuário com dados comentários:

```
# useradd -c "Jose da Silva" usuario2
```

Criando um usuário e definindo que ele deverá trocar sua senha no primeiro login. Isto só é possível quando a política de troca periódica de senhas estiver ativa:

```
# useradd usuario3
# passwd usuario3
# chage -M 90 -d 2000-01-01 usuario3
```

Os passos para a criação do usuário forçando a troca de senhas deve ser seguindo à risca. Logo após a criação do usuário, definimos sua senha inicial e em seguida informamos que as senhas devem ser trocadas a, no máximo, cada 90 dias e que foi trocada pela última vez em 01.jan.2000.

Um último exemplo é o de usuários que serão cadastrados no Linux, porém não terão acesso à linha de comandos:

```
# useradd -s /bin/false usuario4
```

Planeje seu ambiente

Antes de iniciar a implantação de um servidor é importante ter em mente qual será o seu uso e quais recursos devem estar disponíveis. Ter em mente o volume de usuários que será cadastrado pode evitar mudanças drásticas em um curto espaço de tempo.

Definir se todos os usuários cadastrados devem ter acesso ao sistema é outro ponto importante.

Adotar um padrão para a criação de logins de usuários e grupos ajuda a evitar problemas futuros com logins duplicados ou mal interpretados.

Defina se seus servidores terão controle de cotas de disco para usuário e planeje desde o início sua implementação.

Ambiente de trabalho do usuário

Assim que tem acesso ao sistema operacional (login) uma série de configurações pré-definidas são disponibilizadas ao usuário, montando o seu ambiente de trabalho. Este ambiente é personalizável e, uma vez que a maioria do trabalho do administrador é realizada em modo texto, é altamente recomendável torná-lo mais amigável acrescentando funcionalidades ou modificando seus recursos.

Todo o ambiente de trabalho no Linux é controlado por variáveis de ambiente. Estas variáveis controlam desde quais diretórios são pesquisados à procura de programas (PATH) até o tamanho da tela (número de linhas e colunas visíveis), modificando o comportamento dos programas que adequam suas telas ao novo tamanho do terminal.

Variáveis

Um variável é um nome, definido pelo usuário (sem acentos, espaços ou símbolos) que armazena um valor em memória RAM. Seu conteúdo pode ser uma palavra, uma string (texto) ou um número. Criar variáveis pode ser realizado assim:

```
# MENSAGEM="olá"
```

Este comando associa a string “olá” à área de memória identificada como MENSAGEM. Esta área de memória pode ser consultada a qualquer momento no shell corrente porém, da forma como criada, não poderá ser utilizada em outros programas.

A consulta a uma variável é realizada pelo comando echo, precedendo o nome da variável com \$, assim:

```
# echo $MENSAGEM
olá
```

Uma variável pode ser removida da memória tão facilmente quanto pode ser criada, com o comando unset:

```
# unset MENSAGEM
```

Alguns pontos a observar na criação de variáveis é que na sua atribuição NÃO devem existir espaços antes ou depois do sinal de igual e seu conteúdo, quando for um texto, deve ser envolto em “” (aspas).

Variáveis de ambiente

Variáveis de ambiente são aquelas que já estão disponíveis quando o usuário efetua login em um terminal. Elas podem conter vários tipos de informações. Abaixo vemos alguns exemplos:

As variáveis de ambiente podem ser utilizadas por qualquer programa que as consulte, sendo definidas como variáveis de abrangência global (variáveis exportadas). Uma variável que seja criada com os procedimentos vistos acima são consideradas de abrangência local, não podendo ser consultadas por qualquer outro programa além do shell onde foram criadas.

Exportar uma variável é um procedimento tão simples quanto a sua criação e é essencial para que ela seja útil ao sistema. Vamos, inicialmente, criar uma variável simples e consultá-la em um novo shell:

```
# MENSAGEM="posso ver esta mensagem"
# echo $MENSAGEM
posso ver esta mensagem
# bash
# echo $MENSAGEM
<- criação da variável
<- consulta da variável local
<- iniciando um novo shell
<- consulta da variável local
<- conteúdo não está disponível

# exit

# env
LESSKEY=/etc/.less
LC_PAPER=pt_BR
LC_ADDRESS=pt_BR
LC_MONETARY=pt_BR
HOSTNAME=localhost
TERM=xterm
SHELL=/bin/bash
(...)
```

Definindo a variável e exportando-a permite que outros programas como o novo shell tenham acesso a ela:

```
# MENSAGEM="posso ver esta mensagem"
# export MENSAGEM
# echo $MENSAGEM
posso ver esta mensagem
# bash
# echo $MENSAGEM
posso ver esta mensagem
# exit
# <- criação da variável
<- tornando a variável global
<- consulta da variável local
<- iniciando um novo shell
<- consulta da variável local
<- conteúdo está disponível
<- voltando ao shell anterior
```

A partir da execução do comando `export`, todo e qualquer novo programa que consulte o ambiente do usuário poderá utilizar esta variável e seu conteúdo.

Scripts executados no login do usuário

Todas as variáveis são definidas no momento do acesso do usuário ao shell do sistema. Estas definições são carregadas a partir de login-scripts executados automaticamente. Há dois tipos de scripts: os válidos para todos os usuários, definidos no `/etc` e os pessoais, localizados no home do usuário.

Estes scripts possuem vários comandos do shell que podem ser encadeados de acordo com as necessidades de cada equipamento/sistema. No Linux temos os seguintes login-scripts:

ARQUIVO	USO
<code>/etc/profile</code>	Arquivo global, define variáveis exportadas para todos os usuários do sistema como o PATH e ainda define o comportamento de vários aplicativos
<code>/etc/bashrc</code>	Arquivo global, define macros disponíveis para todos os usuários como “ls” alterado para saída colorida
<code>~/.bash_profile</code>	Arquivo pessoal que permite a personalização de variáveis como o PATH, o prompt de comando do usuário
<code>~/.bashrc</code>	Arquivo pessoal que permite a criação de macros personalizadas como a listagem automática dos logs do sistema ou atalhos para comandos complexos repetitivos
<code>~/.bash_logout</code>	Arquivo pessoal executado no logout do usuário, normalmente tem um comando clear, mas pode ser alterado para incluir o backup de dados importantes.

Variáveis pré-definidas no ambiente do usuário

Entre as muitas variáveis disponíveis, algumas merecem destaque e devem ser citadas devido à sua importância e uso no Linux:

VARIÁVEL	DESCRIÇÃO
<code>PATH</code>	Caminho de pesquisa à procura de comandos para executar
<code>HOME</code>	Diretório pessoal do usuário logado
<code>USER</code>	Nome de login do usuário logado, pode ser utilizado ainda <code>LOGNAME</code> e <code>USERNAME</code> disponíveis por compatibilidade
<code>LANG</code>	Idioma preferencial
<code>PWD</code>	Localização atual, mesma saída do comando <code>pwd</code>
<code>LC_ALL</code>	Localidade definida na instalação, alterando a visualização de números, data e hora e outros.
<code>TMP</code>	Indica o diretório temporário
<code>PS1</code>	Define o comportamento do prompt de comandos

VARIÁVEL	DESCRIÇÃO
HISTSIZE	Número de comandos armazenados no histórico do Linux

Comandos UNIX e GNU

Em todo equipamento Linux a utilização da linha de comando é a sua característica mais forte e é onde encontramos todo o poder deste Sistema Operacional. Para melhor aproveitar os recursos disponíveis é essencial que se possa navegar entre diretórios e manipular arquivos em seu equipamento. Aqui vamos abordar desde um acesso ao terminal até o tratamento e edição de arquivos.

Linha de comando

Usar comandos simples ou concatenados é uma tarefa básica para a administração de um sistema Linux, bem como o ambiente ou definir novas variáveis de forma a personalizar a utilização do shell também fazem parte desta administração.

Muitas tarefas podem ser realizadas em um sistema Linux a partir de ferramentas gráficas, contudo várias opções de configuração acabam não estando disponíveis nestas interfaces. É comum que, num primeiro contato, novos administradores, especialmente os originados de sistemas Windows tenham resistência ao Linux e à sua interface texto. Isto é devido principalmente ao fato de que, apesar de poderem estar familiarizados com sistemas MS-DOS, os comandos de operação do Linux são muito diferentes e em alguns casos possuem conceitos diferentes.

Basicamente todo comando no Linux se comporta de forma similar a um sistema DOS ou Windows. Cada comando possui uma sintaxe própria e parâmetros específicos que variam de comando para comando. Uma comparação entre alguns comandos básicos que podem servir para um primeiro contato são:

Dos	Linux	Dos	Linux
dir	ls	type	cat
cls	clear	cd\	cd /
copy	cp	rd	Rmdir
move/rename	mv	del	rm

Sintaxe

Todo comando segue uma estrutura básica para execução. Comparando a estrutura de comandos em sistemas Dos e Linux, percebemos que não há diferenças entre eles. Todo comando obedece à seguinte construção:

<comando> <opções> <argumentos>

ls -l /var/log

Em sistemas Linux qualquer opção a ser passada a um comando é precedida de um hífen “-” enquanto em sistemas Dos normalmente é utilizada uma barra “/”, contudo podem existir comandos onde a forma de informar opções seja diferente. Para saber sobre o comportamento de cada comando em específico, é aconselhável consultar a sua página de manual online (man page). Abaixo temos um trecho da execução do comando acima temos a seguinte saída:

Acessando o sistema operacional

Antes de podermos acessar nossa instalação e realizar nossos primeiros comandos há um conhecimento básico que deve ser passado: como acessar (login), sair (logout) e desligar seu equipamento tudo isto através do modo texto, pois pela interface gráfica estes procedimentos são intuitivos e quase que padronizados.

Apesar de ser uma tarefa banal, pode causar alguns contra-templos caso não estejamos habituados a elas.

Acessando o primeiro terminal em modo texto (<ctrl>+<alt>+<F1>) temos uma tela como a que segue:

```
Debian GNU/Linux 3.1 debian tty1
aluno1 login: _

debian:~# ls -l /var/log
total 528
-rw-r--r-- 1 root      root   2872 2006-07-30 19:08 aptitude
-rw-r----- 1 root      adm    2292 2006-07-30 21:04 auth.log
-rw-r--r-- 1 root      root  172217 2006-07-30 18:51 base-config.log
-rw-r--r-- 1 root      root  10497 2006-07-30 18:51 base-config.timings
-rw-rw-r-- 1 root      utmp     0 2006-07-30 15:34 btmp
-rw-r--r-- 1 root      root   2654 2006-07-30 21:11 daemon.log
-rw-r--r-- 1 root      root   4096 2006-07-30 15:37 debian-installer
drwxr-xr-x 3 root      root    15975 2006-07-30 19:08 debug
-rw-r--r-- 1 root      root  13274 2006-07-30 18:56 dmesg
```

Vamos efetuar login como usuário root:

```
Debian GNU/Linux 3.1 debian tty1
aluno1 login: root
Password:
Last login: Sun Jul 30 21:04:18 2006 from tty1
Linux debian 2.6.8-2-386 #1 Tue Aug 16 12:46:35 UTC 2005 i686 GNU/Linux
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
aluno1:~# _
```

Caso esta não seja a primeira vez que este usuário efetua login, teremos uma mensagem similar a esta, indicando quando foi o último login (Sun Jul 30 21:04:18) e de onde partiu (tty1). O acesso ocorreu sem problemas e assim deve ser normalmente.

Caso eu queira apenas efetuar logoff, basta digitar “exit”, “logout” ou a combinação de teclas “<ctrl>+<d>” e o terminal será fechado.

Antes de continuarmos é bom sabermos como interpretar este prompt.

OPÇÃO	DESCRIÇÃO
aluno1	Nome do equipamento (hostname)
~	Diretório atual, neste caso “~” representa o diretório pessoal do usuário. Se você estiver em outro diretório como, por exemplo, /etc/X11 seu prompt serial apenas “/etc/X11”
#	Indicativo do tipo de usuário conectado (root ou não root). Neste caso o usuário é root, para um usuário não root o indicativo assume o caractere “\$”

Desligando o Linux

Uma procedimento simples, que pelo modo gráfico representa apenas escolher uma opção do menu, no modo texto pode ser uma tarefa ingrata. Vamos ver o comando shutdown, suas opções e variações, além de compará-lo com outros comandos existentes. A estrutura do comando shutdown é a seguinte:

shutdown [opções] tempo [mensagem broadcast]	
OPÇÃO	DESCRIÇÃO
-h	Desligar o equipamento
-r	Reiniciar o equipamento
-t	Tempo, em segundos, até o envio de SIGTERM às aplicações em execução
-c	Aborta a execução do comando shutdown

As opções e tempo são obrigatórios, sem um destes parâmetros o comando não funciona. A condição de tempo pode ser específica em minutos ou com algumas palavras chave como “now” indicando imediatamente. Vamos pedir ao Linux que desligue o computador agora.

```
aluno1:~# shutdown -h now
```

Nosso equipamento deve ter desligado completamente. Após reiniciar o computador, vamos pedir uma reinicialização com tempo dfe 1min.

```
aluno1:~# shutdown -r 1
```

Durante este período nosso prompt de comandos ficou preso, pela execução do “shutdown”. Se algo ocorresse de anormal, ou um simples “**<ctrl>+<c>**” fosse pressionado, o desligamento teria sido abortado. Podemos rodar este comando, colocando-o para execução em segundo plano (background).

```
aluno1:~# shutdown -r 10 &
```

Para cancelar sua execução, uma vez que não temos mais acesso ao comando, devemos utilizar o próprio “shutdown” para esta tarefa.

```
aluno1:~# shutdown -c  
shutdown cancelled  
aluno1:~#
```

Um exemplo de uso com a mensagem de broadcast:

```
aluno1:~# shutdown -h 10 "Servidor ficará fora do ar por 45 min"
```

Embora plenamente funcional, o shutdown não é a única opção para desligar/reiniciar o Linux. Veja a seguinte tabela de comparação:

DESLIGAR	REINICIAR
shutdown -h	shutdown -r
halt	reboot
init 0	init 6
	<ctrl>+<alt>+

A combinação de tecla “<ctrl>+<alt>+” não ativa a interrupção de bios. Ela é interceptada pelo Linux que converte corretamente no comando “shutdown”.

Primeiros comandos

Após o primeiro acesso, já sabemos como ligar ou desligar nosso equipamento, é hora de começarmos a nos mover na estrutura do Linux e a manipular arquivos e diretórios.

pwd

```
aluno1:~# pwd  
/root
```

Exibe a sua localização atual. Qual o caminho completo do diretório onde você está neste momento.

cd

Muda de diretório. De forma similar ao comportamento do Windows®, este comando muda para o diretório especificado, porém não existe o uso de “letras de unidade”, apenas diretórios
Mudar para o diretório “/etc”:

```
aluno1:~# cd /etc  
aluno1:/etc#
```

Mudar para o diretório “/usr/share/doc”:

```
aluno1:/etc# cd /usr/share/doc  
aluno1:/usr/share/doc# pwd  
/usr/share/doc
```

Mudar para o diretório “/usr/local/bin”:

```
aluno1:/usr/share/doc# cd /usr/local/bin  
aluno1:/usr/local/bin#
```

Voltar para o diretório anteriormente visitado:

```
aluno1:/usr/local/bin# cd -  
aluno1:/usr/share/doc#
```

Descer (voltar) um diretório:

```
aluno1:/usr/share/doc# cd ..  
aluno1:/usr/share# pwd  
/usr/share
```

Voltar ao diretório pessoal “cd” ou “cd ~”:

```
aluno1:/usr/share# cd  
aluno1:~# pwd  
/root
```

Caminho absoluto e caminho relativo

Um conceito necessário à navegação em diretórios via linha de comando é o de caminho absoluto e caminho relativo. Caminho absoluto é todo aquele que se inicia a partir do diretório raiz (“/”) e caminho relativo é aquele que só existe em relação à sua localização atual, por exemplo, todos os

```
cd /etc/x11  
cd /usr/share/doc  
cd /usr/local/bin
```

comandos de troca de diretório realizados acima foram feitos com caminhos absolutos:

Todos estes exemplos iniciam na raíz do Linux. Agora, de forma prática, vamos testar a idéia de caminho relativo. Acesse o diretório “/var”:

```
aluno1:/usr/local/bin]# cd /var  
aluno1:/var#
```

Estando dentro dele acesse o subdiretório tmp:

```
aluno1:/var# cd tmp
aluno1:/var/tmp]# pwd
/var/tmp
```

Se tivessemos incluído uma “/” antes de tmp teríamos acessado o diretório “/tmp” e não “/var/tmp” como solicitado.

Tenha sempre muito cuidado ao acessar diretórios, pois alguns comandos realizados no local errado podem invalidar todo o seu sistema instalado.

clear

Limpa a tela do computador, equivale a um “cls” no Windows®

```
aluno1:/tmp# clear
```

tree

Exibe a estrutura de diretórios e arquivos do diretório corrente ou do diretório solicitado. Por exemplo,

```
aluno1:~# tree
|-- drakx
|   |-- README
|   |-- auto_inst.cfg.pl
|   |-- ddebug.log
|   |-- install.log
|   |-- replay_install.img
|   |-- report.bug.gz
|   |-- stage1.log
-- tmp
2 directories, 7 files
aluno1:~#
```

estando em seu diretório pessoal, digite o seguinte comando:
Omitindo a exibição do conteúdo (arquivos) dos diretórios:

```
aluno1:~# tree -d /boot
/boot
-- grub
1 directory
aluno1:~#
```

ls

Lista o conteúdo de um diretório indicado na linha de comando ou do diretório atual. Possui parâmetros que alteram a forma de exibição da listagem, como mostram as opções abaixo:

ls [opções] arquivo	
OPÇÃO	Descrição
-l	Exibe o conteúdo do diretório em formato detalhado

ls [opções] arquivo

-a	Exibe todos os objetos dentro de um diretório, incluindo arquivos e pastas ocultas (iniciados por ".")
-F	Acrescenta ao nome do objeto um caracter para indicar o seu tipo, por exemplo: / para diretórios e * para executáveis
-r	Ondem inversa (decrescente) de exibição.
-R	Listagem em modo recursivo ou seja, exibe o conteúdo do diretório corrente e de todas as subpastas
-l	Exibe uma saída simples (apenas o nome) em uma coluna
-S	Ordena a saída pelo tamanho dos objetos
-t	Ordena a saída pela data/hora da criação
--color	Saída colorida de acordo com as definições da variável LS_COLORS. As cores normalmente utilizadas são: diretórios em azul escuro, links simbólicos em azul claro (cyan), dispositivos em amarelo e arquivos compactados em vermelho

Exemplos:

Listagem simples do diretório /etc

```
# ls /etc
acpi/          hosts           personal        mtab      rpc
adjtime        hosts.allow    personal.conf   mtools.conf  rpm/
alternatives/ hosts.deny     personal.log    nail.rc    RPM-GPG-KEY/
asound.names   ifplugd/       netplug/        netplug/    samba/
at.deny        iftab/        netplug.d/      netprofile/ sane.d/
bash_completion.d/ info-dir   noprofile/    nsswitch.conf screenrc
bashrc         init.d@      netprofile/    openldap/   scsi_id.config
cron.d/        initlog.conf  nsswitch.conf  openldap/  securetty
(...)
```

Listagem em modo detalhado exibindo arquivos ocultos com caractere de controle:

```
# ls -laF /var/log
total 536
drwxr-xr-x  6 root      root  4096 2006-07-30 18:48 ./
drwxr-xr-x 13 root      root  4096 2006-07-30 15:34 ../
-rw-r--r--  1 root      root  2872 2006-07-30 19:08 aptitude
-rw-r-----  1 root      adm   2615 2006-07-30 21:18 auth.log
-rw-r-----  1 root      root 172217 2006-07-30 18:51 base-config.log
-rw-r-----  1 root      root 10497 2006-07-30 18:51 base-config.timings
-rw-rw-r--  1 root      utmp     0 2006-07-30 15:34 btmp
-rw-r-----  1 root      root  2887 2006-07-30 21:26 daemon.log
(...)
```

Mesmo tipo de listagem do diretório pessoal adicionando a saída em cores:

```
# rmdir aula1 aula2 aula3 aula4 novo-dir
```

Apagando uma estrutura de diretórios:

```
# rmdir -p ~/Linux/Debian/Aula2/Exemplos
```

cat

Envia o conteúdo de um arquivo para a saída padrão. Em seu funcionamento básico equivale a visualizar o conteúdo de um arquivo. Pode ser utilizado para concatenar arquivos ou ainda, se nenhum arquivo ou parâmetro for fornecido captura os dados digitados pelo teclado e os envia à saída padrão.

Sua sintaxe e principais opções:

cat [opções] arquivo	
OPÇÃO	Descrição
-n	Numera todas as linhas do arquivo
-b	Numera todas as linhas com conteúdo
-E	Exibe “\$” como fim de linha
-T	Exibe “^I” como caractere de tabulação

Exemplos:

Exibe o conteúdo do arquivo “/etc/passwd”:

```
# cat /etc/passwd
```

Exibe o arquivo “/etc/profile”, enumerando suas linhas:

```
# cat -n /etc/profile
```

Exibe o arquivo “/etc/profile”, enumerando suas linhas que não estejam em branco:

```
# cat -b /etc/profile
```

tac

Com funcionalidade parecida ao **cat**, este comando que exibe o conteúdo do arquivo em ordem inversa.

```
# tac /etc/profile
```

more

Exibe o conteúdo de um arquivo em tela, paginando o seu conteúdo.

more [opções] arquivo	
OPÇÃO	Descrição
-d	Exibe informações adicionais no rodapé da tela
-s	Suprime linhas em branco consecutivas.
+NN	Exibe o conteúdo do arquivo a partir da linha especificada

Exemplos:

Visualizar o conteúdo de vários arquivos, com orientações no rodapé da tela:

```
# more -d /etc/sudoers /etc/services
```

Visualiza o conteúdo do arquivo a partir da linha 200:

```
# more +200 /etc/services
```

O comando `more` finaliza automaticamente ao atingir o fim do arquivo ou pressionando a letra “q”.

less

Utilizado como paginador de textos, substitui o comando `more` com muita eficiência, acrescentando recursos de navegação e pesquisa avançada no texto através de expressões regulares.

A navegação no texto pode ser feita com as setas do teclado, as teclas “page dow” e “page up” além de “home” e “end”

`less` é utilizado como paginador padrão em vários comandos do Linux como o `man`.

```
# less /etc/services
```

Exemplo:

As principais ações dentro do `less` podem ser vistas a seguir:

OPÇÃO	Descrição
setas	Navega pelo texto no sentido escolhido
h, j, k, l	Seta a esquerda, seta abaixo, seta acima e seta a direita respectivamente, da mesma forma que utilizado no <code>vi</code> .
z, f ou espaço	Mesmo que “page down”
b	Mesmo que “page up”
/	Abre um prompt de pesquisa dentro do texto
:n	Próximo arquivo

OPÇÃO	DESCRIÇÃO
:p	Arquivo anterior
q	Finaliza a aplicação

Indicando vários arquivos para serem visualizados:

```
# less /etc/passwd /etc/group
```

Neste exemplo os arquivos são visualizados um a um, e pressionando :p ou :n podemos mudar entre eles.

cp

Copia arquivos para o destino informado sendo que, alguns pontos devem ser levados em consideração:

- quando mais de um arquivo for copiado, o destino somente poderá ser um diretório, como neste exemplo:

```
# cp /etc/* /tmp
```

- no funcionamento padrão, copiar um arquivo para um diretório onde já exista um outro arquivo com o mesmo nome irá sobrescrever o arquivo existente;
- se o nome de destino do arquivo a ser copiado for diferente do nome de origem, o arquivo será renomeado no destino, mantendo o arquivo original.

As principais opções do comando cp são:

OPÇÃO	DESCRIÇÃO
-a	Preserva todas as características do arquivo/diretório original na cópia, mantendo dados como proprietário/grupo, data e hora de criação além de ser executado de forma recursiva. O mesmo que -dpR
-b	Caso o arquivo de destino exista, faz um backup antes de sobrescrevê-lo.
-f	Força a gravação da cópia do arquivo, mesmo que o parâmetro interativo -i tenha sido especificado.
-d	Realiza a cópia de links simbólicos e não de arquivos.
-i	Modo interativo, solicita confirmação antes de sobrescrever um arquivo.
-p	Mantém os dados originais do arquivo origem na cópia, quando possível. Os dados preservados são: proprietário/grupo, data e hora de modificação e acesso e permissões.
-R	Copia diretórios recursivamente, sendo que o destino neste caso deve, obrigatoriamente, ser um diretório.
-v	Modo detalhado onde cada arquivo copiado é listado.

Exemplos:

Copia um arquivo com outro nome:

```
# cp /etc/services ~/serviços
```

Copia um arquivo para outro diretório mantendo o nome original:

```
# cp /etc/services /tmp
```

Copia um arquivo de forma interativa:

```
# cp -i /etc/services /tmp
```

Copia um arquivo efetuando um backup antes de sobrescrever o destino:

```
# cp -b /etc/services /tmp
```

Copia vários arquivos para outro diretório:

```
# cp /etc/* /tmp
```

Copia uma estrutura de diretórios para outro diretório:

```
# cp -r /etc/* /tmp
```

mv

Move ou renomeia arquivos / diretórios. As opções mais comuns são

OPÇÃO	DESCRIÇÃO
-f	Força a gravação do destino, caso este já exista.
-i	Modo interativo.
-b	Efetua backup antes da gravação do destino.
-v	Modo detalhado.

Exemplos:

```
# cd /tmp
# mv services serviços
```

Renomeando um arquivo:

Movendo um arquivo para outro local:

```
# mv serviços ~/tmp
```

Movendo um arquivo forçando a gravação do destino:

```
# ln -s mandriva-release versao-mandriva
```

Criando um apelido para um comando:

```
# ln -s /usr/bin/clear /bin/cls
```

Filtros em arquivos

Vimos que, com o comando `less`, é possível localizar dados dentro de arquivos de forma simples e eficaz, porém em alguns casos não queremos abrir o arquivo para localizar determinada ocorrência, gostaríamos de visualizar em nosso terminal. Isto pode ser feito com o comando `grep`. Este comando tem a seguinte sintaxe:

grep [opções] “filtro-a-aplicar” arquivo	
OPÇÃO	DESCRIÇÃO
<code>-i</code>	Ignora diferenças entre maiúsculas e minúsculas
<code>-l</code>	Quando indicar * no lugar de arquivos, retorna apenas o nome dos arquivo que contém a expressão pesquisada.
<code>-E</code>	Ativa o modo de expressões regulares

Um exemplo desta aplicação é localizar a ocorrência “ldap” dentro do arquivo `/etc/services`:

```
# grep "ldap" /etc/services
ldap          389/tcp          # Lightweight Directory Access Protocol
ldap          389/udp
ldaps         636/tcp          # LDAP over SSL
ldaps         636/udp
```

Uma opção pode ser a pesquisa por ocorrências em letras maiúsculas ou minúsculas, ou qualquer combinação destas:

```
# grep -i "name" /etc/services
nameserver    42/tcp           name        # IEN 116
whois         43/tcp           nickname
domain        53/tcp           nameserver # name-domain server
domain        53/udp
hostnames     101/tcp          hostname
csnet-ns      105/tcp          cso-ns     # usually from sri-nic
netbios-ns    137/tcp          # also used by CSO name server
at-nbp        202/tcp          # NETBIOS Name Service
at-nbp        202/udp          # AppleTalk name binding
#> Ports are used in the TCP [45,106] to name the ends of logical
nbp          2/ddp            # Name Binding Protocol
```

Algumas condições especiais de pesquisa podem ser definidas com o uso de expressões regulares. Não vamos nos aprofundar nisto agora, apenas vamos ver duas condições fornecidas por “^” que, neste caso, representa início de linha e “\$” que representa fim de linha. A mesma pesquisa anterior, forçando as ocorrências em início de linha:

```
# grep -i "^name" /etc/services
nameserver      42/tcp        name          # IEN 116
```

Agora um exemplo pesquisando por uma condição de final de linha:

```
grep -i "mail$" /etc/services
smtp           25/tcp        mail
```

Redirecionamento e canalização

Um conceito a ser absorvido quando se inicia o uso do Linux é que em sistemas POSIX os aplicativos são desenvolvidos para efetuarem uma única tarefa, porém que ela seja muito bem realizada. Este conceito nasceu em uma época onde os computadores possuíam, nos melhores casos, 640kb de memória RAM total. Em um ambiente assim é impossível a execução de uma aplicação que consuma muita memória ou que, para tratar seus dados, aloque grandes áreas em RAM.

Um exemplo deste conceito é que, ao tentar visualizar um arquivo com o comando `cat`, normalmente juntamos um comando adicional que permite paginá-lo em tela:

```
# cat /etc/services | more
```

A junção destes comandos foi feita com o uso do caractere “|” (pipe-line ou simplesmente pipe – cano – daí o nome de canalização). O efeito produzido com este comando é que a saída do comando `cat` que seria jogada em tela foi canalizada para a entrada do comando `more` e este exibiu o arquivo tela a tela de acordo com suas opções disponíveis.

Um outro exemplo deste uso é a impressão de um trecho de um arquivo, por exemplo as primeiras 240 linhas do arquivo `/etc/services`:

```
# head -n 240 /etc/services | lpr
```

Canalizar significa que os dados que seriam enviados à saída padrão são encaminhados a outro programa para processamento adicional.

Já o conceito de redirecionamento é muito semelhante à canalização, porém o seu efeito é que normalmente resulta em arquivos adicionais. Por exemplo, para obter uma listagem, em arquivo, de todos os arquivos e diretórios dentro do `/etc` executamos o seguinte comando:

```
# ls -lR /etc > ~/listagem/etc.txt
```

Apesar de, na prática, o efeito ter sido o mesmo da canalização: a saída foi redirecionada para outro lugar, o efeito é que foi gerado um arquivo adicional com este comando. As possibilidades de redirecionamento são:

REDIRECIONADOR	SIGNIFICADO
>	Redireciona a saída para um arquivo novo.
>>	Redireciona a saída para um arquivo existente, anexando os dados ao final do arquivo. Caso o arquivo não exista, será automaticamente criado.
<	Redireciona a entrada padrão, lendo dados de um arquivo
<<	Redireciona a entrada padrão, lendo dados de um arquivo ou teclado até que uma condição seja satisfeita.

Nos exemplos acima, a saída padrão foi alterada, tendo sido enviada para um arquivo ou para um outro programa. Esta saída padrão, na maioria dos programas é a tela, e é um dos descritores de seu equipamento. Em todos os sistemas computacionais existem 3 descritores:

DESCRITOR	SIGNIFICADO
0	Entrada padrão: normalmente é o teclado
1	Saída padrão: normalmente é a tela
2	Saída de erro padrão: normalmente é a tela

O uso destes descritores adicionais é, por exemplo, redirecionar as saídas de erro para um arquivo de forma a termos um log de ocorrências.

```
# ls -lR /proc > ~/listagem-proc.txt 2> ~/listagem-proc.erro
```

Uma aplicação para o redirecionamento da entrada de dados (<) é, por exemplo, a ordenação de um arquivo:

```
# sort < /etc/services | less
```

E o redirecionamento da entrada padrão de forma condicional (<<), conhecido como “here document” pode ser a criação de um arquivo texto via console:

```
# cat << FIM > ~/arquivo
exemplo de uso do "here document"
todo este teste fará parte de arquivo
com exceção da linha FIM
FIM
```

Consultando este arquivo temos:

```
# cat ~/arquivo
exemplo de uso do "here document"
todo este teste fará parte de arquivo
com exceção da linha FIM
```

Localização de arquivos

Em qualquer sistema instalado, localizar arquivos é uma tarefa essencial à administração de qualquer servidor. No Linux há vários métodos para localização de arquivos, sendo alguns extremamente rápidos porém pouco precisos e outros extremamente precisos porém mais morosos.

find

A localização de arquivos com o comando **find** é extremamente precisa pois ele varre o hd em busca do objeto que satisfaça as condições estabelecidas pelo usuário. Em seu funcionamento padrão, o **find** aceita os seguintes argumentos:

find caminho [opções] [comando-complementar]	
OPÇÃO	Descrição
-name	Localiza objetos baseado no nome
-iname	Localiza objetos baseado no nome ignorando diferenças entre maiúsculas e minúsculas
-type	Localiza um objeto de acordo com seu tipo: d: diretório f: l: link simbólico
-user	Localiza objetos pertencentes ao usuário especificado
-group	Localiza objetos pertencentes ao grupo especificado
-perm	Localiza objetos com a permissão especificada: find / -perm -4000
-empty	Localiza objetos com tamanho 0 bytes
-atime	Localiza objetos com acesso em N dias
-ctime	Localiza objetos que tenham tido suas permissões ou usuário/grupo alterados em N dias
-mtime	Localiza objetos que tenham sido alterados em N dias

Exemplos:

Localizando um arquivo específico:

```
# find / -iname "bashrc"  
/etc/bashrc
```

Localizando objetos baseado em sua data de acesso há menos de 1 dia:

```
# find / -atime -1
```

Localizando objetos usando metacaracteres:

```
# find / -iname "*profile*"
```

Executando um comando complementar após a localização do objeto:

```
# find / -iname "bashrc" -exec ls -l {} \;  
-rw-r--r-- 1 root root 913 Ago 31 2005 /etc/bashrc
```

Localizando objetos baseado em suas permissões:

```
# find /usr -perm -4000
```

Utilizando mais de uma condição de pesquisa:

```
# find /etc \(( -type f -a -group disk \)  
/etc/dumpdates
```

Acima a condição de pesquisa foi a localização de um arquivo e que pertence ao grupo disk. A tag “-a” indica que as duas condições devem ser satisfeitas. Caso a intenção seja de uma ou outra condição pode ser utilizada a chave “-o”.

locate

O comando locate utiliza uma base indexada para localização de arquivos. Antes de sua utilização é necessário construir esta base com o comando updatedb.

```
# updatedb
```

Na primeira execução deste comando é criada a base e todo o hd é varrido para a construção da base e de sua indexação. Por questões de performance e segurança, este comando não indexa diretórios temporários, diretórios pessoais (/home, /root) e sistemas de arquivos remotos (mapeamentos). Qualquer pesquisa realizada por qualquer usuário somente retornará os dados a que o usuário tiver permissão de acesso. Exemplos de uso:

```
# locate baserc  
# locate profile.d
```

Devido à sua característica de pesquisa “off-line” (em uma base previamente construída), devemos criar uma tarefa agendada para atualizar nossa base periodicamente. A regularidade desta atualização deve ser estabelecida de acordo com o uso de nosso equipamento.

whereis

Localiza arquivos executáveis e páginas de manual, precedendo a saída com os dados informados para pesquisa. Este comando utiliza as variáveis de ambiente PATH e MANPATH para realizar as pesquisas. Exemplo:

```
# whereis shutdown  
shutdown: /usr/bin/shutdown /sbin/shutdown /usr/share/man/man8/shutdown.8.bz2  
/usr/share/man/man2/shutdown.2.bz2 /usr/share/man/man3p/shutdown.3p.bz2
```

Pode-se localizar apenas os executáveis especificando a opção “-b” (binários) ou as páginas de manual com a opção “-m”.

which

Outra alternativa para localização de executáveis é o comando which. Sua saída é simples, indicando apenas o caminho completo e o programa informado. Exemplo:

```
# which passwd  
/usr/bin/passwd
```

Assim como whereis, which utiliza a variável de ambiente PATH para pesquisas. Por padrão, este comando para na primeira ocorrência encontrada. Caso queira que a pesquisa seja realizada em todos os diretórios do PATH, utilize a opção “-a”.

Comandos úteis

Além dos primeiros comandos vistos anteriormente, a administração de um sistema Linux requer, muitas vezes, o uso de alguns comandos complementares.

head

Exibe as primeiras 10 linhas de um arquivo. Exemplo:

```
# head /etc/services
```

Pode receber como parâmetros um número diferente para exibição em tela:

```
# head -n 55 /etc/services
```

Pode ser utilizado em canalizações recebendo os dados a filtrar pelo redirecionamento da entrada padrão:

```
# cat /etc/profile | head -n 5
```

tail

Exibe as últimas 10 linhas do arquivo informado. Exemplo:

```
# tail /etc/passwd
```

Pode ser utilizado com canalização. Filtrando todas as linhas com “tcp” no arquivo /etc/services e exibindo as últimas 15 ocorrências:

```
# grep -i "tcp" /etc/services | tail -n 15
```

Os comandos head e tail podem ser utilizados em conjunto para exibir um trecho de um arquivo, por exemplo, das linhas 311 à 315 do arquivo /etc/services:

```
# head -n 315 /etc/services | tail -n 5
```

sort

Ordena o conteúdo de um arquivo enviando o resultado a saída padrão (tela). Suas principais opções são:

sort [opções] arquivo	
OPÇÃO	DESCRIÇÃO
-b	Ignora espaços em branco ou tabulações no início de linha
-d	Efetua ordenação em ordem de dicionário
-f	Ignora maiúsculas e minúsculas
-k	Campo a ser utilizado para ordenação
-n	Força ordenação numérica (por valor)

sort [opções] arquivo	
-r	Ordenação inversa (decrescente)
-t padrão	Utiliza o caractere informado como padrão de separação de campos, utilizado em conjunto com -k
-u	Elimina linhas duplicadas

Exemplos:

Ordenando a agenda pessoal:

```
# sort -bdf agenda.dat
```

Ordenando os usuários que têm acesso à linha de comando em ordem inversa:

```
# grep bash /etc/passwd | sort -r
```

Ordenando os usuários cadastrados no equipamentos pelo seu uid (terceiro campo):

```
# sort -t: -k3 -n /etc/passwd
```

expand

Substitui tabulações por espaços em branco, de acordo com o valor informado. Suas opções são:

expand [opções] arquivo	
OPÇÃO	Descrição
-i	Substitui apenas as ocorrências em início de linha
-t N	Valor a ser utilizado para o tamanho da tabulação

Exemplos:

Expandindo todas as tabulações de um arquivo para seu valor padrão (8 posições):

```
# expand /exemplos/usuarios
```

Expandindo as tabulações para um valor determinado:

```
# expand -t 15 /exemplos/usuarios
```

wc

Conta o número de linhas, palavras e caracteres de um arquivo. Suas opções são:

wc [opções] arquivo	
OPÇÃO	Descrição
-c	Conta o número de caracteres do arquivo
-l	Conta o número de linhas do arquivo
-w	Conta o número de palavras do arquivo

Exemplos:

Contando o número de linhas do arquivo /etc/services:

```
# wc -l /etc/services
584 /etc/services
```

cut

Recorta colunas de arquivos e as exibe em tela. As colunas podem ser posições fixas do texto, letra a letra, ou campos definidos por separadores específicos. Suas opções são:

cut [opções] arquivo	
OPÇÃO	Descrição
-c	Lista de colunas a exibir
-d	Delimitador de campo a ser utilizado
-f	Campo a ser exibido, de acordo com o delimitador especificado

Exemplos:

Exibindo os nomes de usuários cadastrados no arquivo /etc/passwd:

```
# cut -d : -f 1 /etc/passwd
```

Exibindo as colunas de 1 a 8 deste mesmo arquivo:

```
# cut -c 1-8 /etc/passwd
```

tr

Troca caracteres do arquivo informado, exibindo o resultado em tela. Suas principais opções são:

tr [opções] expr1 expr2	
OPÇÃO	DESCRIÇÃO
-d car	Apaga o caracter informado
-s	Substitui ocorrências duplicas por apenas uma

Exemplos:

Trocando todas as letras maiúsculas por minúsculas:

```
# w | tr "A-Z" "a-z"
```

Trocando ":" do arquivo /etc/passwd por um tab:

```
# w | tr ":" "\t"
```

nl

Enumera as linhas de um arquivo, similar ao comando “cat -n”. Exemplo:

```
# nl /etc/services
```

diff

Exibe as diferenças entre dois arquivos. Suas principais opções são:

tr [opções] expr1 expr2	
OPÇÃO	DESCRIÇÃO
-i	Ignora diferenças entre maiúsculas e minúsculas
-u	Mostra uma saída simplificada das diferenças entre arquivos

Exemplos:

```
# diff /etc/passwd /etc/passwd-
```

Durante a instalação do seu equipamento ocorrem alterações no cadastro de usuários. Veja as diferenças:

Gerenciamento de processos

Programas e processos

Um programa é um arquivo contendo comandos que são executados em uma ordem específica a fim de realizar uma tarefa. Normalmente estes programas possuem instruções em código de máquina adequado ao tipo de processador de seu equipamento. Exemplos de programas são:

- /bin/bash
- /sbin/shutdown

Cada programa sendo executado gera tarefas que são realizadas pelo sistema operacional. Estas tarefas alocam recursos de processamento e memória. Para poder gerenciar de forma adequada os recursos consumidos por um programa, o sistema operacional define um código numérico que recebe o nome de processo sob o qual são definidos vários itens como prioridade de execução, tamanho máximo de memória disponível para uso entre outros.

Um simples comando “ls” gera um processo que ocupa recursos do S.O. e acessa periféricos de seu equipamento. Os recursos de memória e processamento são alocados para o comando apenas durante a sua execução, sendo liberados ao sistema operacional quando este é finalizado.

Gerenciar processos envolve verificar quais estão em atividade, alterar sua prioridade ou mesmo interromper sua execução.

Daemons e zumbis

Um daemon disponibiliza serviços para o equipamento ou para a rede como um todo. Normalmente os daemons executam em segundo plano (background) e as saídas de tela são redirecionadas para os logs do sistema operacional.

Um zumbi é um processo terminado. Ao efetuar um comando “ls” é exibido o conteúdo do diretório em tela. Ao finalizar o comando “ls”, seu processo se torna zumbi e é removido da memória pelo comando que o criou (neste caso o shell do usuário) e seus recursos são disponibilizados ao sistema operacional novamente.

Identificação de um processo

Todo processo é identificado por um número único chamado de “process ID” (PID) e cada processo pertence a um usuário e a um grupo, que definem o nível de acesso do processo aos recursos do equipamento.

Como qualquer tarefa no Linux, um processo aloca duas áreas de memória distintas:

- área de código *read-only), onde reside as instruções que serão executadas;
- área de dados (gravável) onde são armazenados os dados sendo manipulados pelo processo.

Entre as várias informações sobre cada processo armazenada pelo kernel, as mais importantes são:

- Process ID – PID: um número inteiro e único que identifica o processo no equipamento. São alocados de forma seqüencial e mesmo que a contagem reinicie, nunca existirão dois processos com um mesmo número.
- Segmentos de texto, dados e pilha: detalha a utilização de recursos de memória para cada segmento

específico, além de mapear recursos virtuais para recursos físicos.

- User ID e Group ID: identificação numérica de usuário e grupo que originou o processo.
- Status: um processo pode estar aguardando novos eventos (standby), em execução (running) ou parado (stopped).

Verificando os processos em execução

Há vários métodos para identificar os processos em atividade em um equipamento, entre os mais populares temos o comando “ps” (process status), o top, pstree, fuser e pidof. A seguir vamos verificar o funcionamento destes e de mais alguns programas para verificação e listagem de processos.

ps

O comando ps lista, por padrão apenas os processos executados pelo usuário atual e de forma simplificada. Podemos alterar este funcionamento com parâmetros específicos para aumentar o nível de informação ou ainda mudar a forma de exibição dos dados. Os principais parâmetros são:

ps [opções]	
OPÇÃO	Descrição
a	Exibe processos ligados a terminais (ignora daemons e shells de login)
e	Exibe o ambiente (variáveis) que o processo utiliza depois da linha de comando
f	Exibe em forma de árvore os processos gerados
l	Listagem detalhada
t##	Exibe os processos em execução em no terminal t##
u	Exibe informações de usuário e hora de início
w	Acrescenta mais colunas à exibição dos dados em tela
x	Exibe processos não ligados a terminais como servidores
-o	Permite definir o formato de exibição em tela (campos)
-C	Exibe os processos originados pelo comando informado
--user	Exibe os processos do usuário informado

As identificações de cada coluna na saída do comando “ps” são:

- PID: Identificação do processo
- TIME: tempo de utilização de cpu
- UID: usuário dono do processo
- TTY: terminal onde o processo está em execução

- COMMAND: comando executado
- STAT: status do processo, podendo ser:

R	em execução	(runnable)
S	sem atividade	(sleeping)
Z	finalizado porém ainda não removido da memória	(zombie)
D	dispositivo sem atividade	por muito tempo
<	processo com alta prioridade	
N	processo com baixa prioridade	

```
# ps
 PID TTY      TIME CMD
 2181 pts/0    00:00:00 bash
 2195 pts/0    00:00:00 ps
```

Processos do usuário atual:

Todos os processos em execução na máquina (comandos e serviços):

```
# ps ax
 PID TTY      STAT   TIME COMMAND
 1 ?        S      0:00 init [2]
 2 ?        SN     0:00 [ksoftirqd/0]
 3 ?        S<    0:08 [events/0]
 4 ?        S<    0:00 [khelper]
 (...)
```

Todos os processos, com identificação de usuário e informações adicionais:

```
# ps axuw
USER   PID %CPU %MEM   VSZ   RSS TTY      STAT START   TIME COMMAND
root    1  0.1  0.5  1504  512 ?        S 21:19  0:00 init [2]
root    2  0.0  0.0     0     0 ?        SN 21:19  0:00 [ksoftirqd/0]
root    3  1.7  0.0     0     0 ?        S< 21:19  0:08 [events/0]
root    4  0.0  0.0     0     0 ?        S< 21:19  0:00 [khelper]
(...)
```

Processos em formato detalhado alternativo:

```
# ps axlw
F   UID   PID  PPID PRI NI   VSZ   RSS WCHAN STAT TTY      TIME COMMAND
4   0     1     0  16  0  1504  512 -      S   ? 0:00 init [2]
1   0     2     1  34  19     0     0 ksofti SN   ? 0:00 [ksoftirqd/0]
1   0     3     1  5 -10     0     0 worker S<  ? 0:08 [events/0]
1   0     4     3  6 -10     0     0 worker S<  ? 0:00 [khelper]
(...)
```

Listando os processos de um usuário exibindo apenas o usuário, pid e comando:

```
# ps -o user,pid,command --user aluno
```

pstree

Mostra os processos em execução em formato hierárquico, identificando quais processos foram originados por quem.

Em seu formato padrão são exibidos todos os processos em execução no equipamento, podendo ser filtrados dados como usuário que iniciou o processo ou processos originados por outro processo específico. Seus principais parâmetros são:

pstree [opções] [pid usuário]	
OPÇÃO	DESCRIÇÃO
-a	Mostra os argumentos da linha de comando
-A	Exibe a árvore de processos com caracteres ASCII
-h	Destaca o processo corrente e sua ascendência
-H	Destaca um processo informado

Árvore de processos em saída ASCII exibindo todos os argumentos utilizados:

```
# ps -Aah
init
(..)
|-sshd
|  `-sshd
|    `--bash
`--syslogd
           `--pstree -Aah
```

Árvore de processos de um determinado usuário:

```
# ps aluno
```

top

Um comando interativo muito popular para acompanhar o funcionamento do equipamento por algum tempo. Este programa exibe em tela os processos em atividade, efetuando uma atualização dos processos a cada 3seg, por padrão.

Além dos dados de processos como PID e COMMAND são exibidos em tempo real a taxa de utilização de CPU, MEMORIA, PRIORIDADE de execução e outros. O top permite a passagem de parâmetros em linha de comando e a utilização de comandos dentro de sua interface.

Parâmetros:

top [opções]	
OPÇÃO	DESCRIÇÃO
-b	Modo não interativo (executa e sai)
-d	Tempo entre as atualizações de tela
-i	Mostra apenas os processos ativos (running)
-u	Mostra os processos do usuário informado
-p	Mostra os processos informados pelo pid (lista separada por vírgulas)

Comandos dentro da tela do top:

COMANDO	DESCRIÇÃO
F	Escolher o campo para ordenação
M	Ordena a exibição pela utilização de memória
P	Ordena a exibição pela utilização de CPU
R	Inverte a exibição
<Enter>	Atualiza a exibição de processos
d	Altera o intervalo entre exibições, em seg
k	Finaliza um processo
q	Encerra o top
r	Altera o nível de prioridade do processo
u <user>	Exibe apenas os processos do usuário informado

fuser

Lista o número dos processos que estão acessando determinado arquivo ou diretório. Muito utilizado para identificar o bloqueio de dispositivos ou mapeamentos remotos.

OPÇÃO	DESCRIÇÃO
-u	Mostra o PID e o usuário que roda o comando
-v	Altera o formato de exibição para a saída do comando ps, com os seguintes campos: PID, USER, ACCES e COMMAND
-k	Finaliza os processos que estão acessando o arquivo

Fecha todos os processos que foram iniciados na console pelo terminal tty1

```
# fuser -k /dev/tty1
```

Lista os processos que estão acessando o diretório /etc

```
# fuser /etc
```

pidof

Mostra o PID de um processo relativo ao comando informado. Caso existam vários processos ativos para o comando informado, todos são listados em uma mesma linha.

pidof [opções] comando	
OPÇÃO	Descrição
-s	Exibe apenas o PID do primeiro processo localizado
-x	Exibe o PID de scripts também

Todos os processos relativos ao comando getty:

```
# pidof getty
13923 2378 2341 2340 2339 2338
```

Todos os processos relativos ao script brute.sh

```
# pidof -x brute.sh
6455 6443
```

kill

Normalmente é utilizado quando se perde o controle sobre um programa ou serviço, permitindo alterar o status de funcionamento de um processo, parando-o ou mesmo finalizando de forma segura ou de forma radical.

O comando kill pode enviar vários sinais aos processos em atividade no sistema. Em seu funcionamento padrão é enviado um sinal chamado SIGTERM de código 15 que solicita ao processo que se finalize de forma convencional, fechando seus canais de leitura/gravação e removendo-se da memória do sistema. Um método mais radical é realizado com o sinal SIGKILL de código 9 onde o sistema operacional realiza a tarefa de remover o processo da memória e encerrar seus canais de leitura/gravação.

kill [opções] processo	
OPÇÃO	Descrição
-l	Lista todos os sinais possíveis para o comando kill
-s SINAL	Envia o sinal informado ao processo especificado

Localizando e pedindo o encerramento um processo do ssh (acesso remoto):

```
# ps ax | grep ssh
2309 ?      Ss    0:00 /usr/sbin/sshd
5568 ?      Ss    0:00 sshd: marcos [priv]
5571 ?      S    0:00 sshd: marcos@pts/0
# kill -s 15 5571
```

Forçando o encerramento de vários processos de uma só vez:

```
# kill -9 5568 5571
```

Forçando que o servidor squid releia seus arquivos de configuração e ative as mudanças:

```
# ps ax | grep squid
1958 ?    Ss      0:00 /usr/sbin/squid
# kill -1 1958
```

killall

Permite enviar sinais a processos informando o nome do comando em execução ao invés de seu PID. Isto facilita muito o gerenciamento de processos uma vez que não é necessário localizar seu PID antes de encerrar o comando ou de reconfigurá-lo.

Da mesma forma que o comando kill, o sinal padrão é o SIGTERM (15) que solicita o encerramento do processo. As principais opções do comando killall são:

killall [opções] comando	
OPÇÃO	Descrição
-i	Solicita uma confirmação para o envio do sinal especificado
-s SINAL	Envia o sinal informado ao processo especificado
-v	Modo detalhado informando quais processos foram finalizados.
-w	Envia o sinal aos processos e aguarda até que todos sejam executados

Finalizando todos os processos do servidor spamd:

```
# killall -9 spamd
```

Finalizando todos os processos do spamd solicitando confirmação:

```
# killall -9 spamd
Kill spamd(2156) ? (y/n) y
Kill spamd(4854) ? (y/n) y
Kill spamd(4993) ? (y/n) y
Kill spamd(5287) ? (y/n) y
Kill spamd(5306) ? (y/n) y
Kill spamd(5350) ? (y/n) y
```

Finalizando os processos do spamd aguardando até que sejam finalizados:

```
# killall -v -w spamd
Killed spamd(8068) with signal 15
```

pgrep – process grep

Executa um filtro sobre a saída do comando ps, exibindo apenas os PID's do comando informado. Permite ainda que sejam executados filtros adicionais como apenas os processos de um comando executado por um usuário específico.

pgrep [opções] filtro	
OPÇÃO	Descrição
-l	Lista o PID e o comando localizado com base no filtro
-u user	Lista os processos relativos ao usuário informado
-x	Somente exibe os processos que casem exatamente com o padrão informado

Lista os processos e os comando correspondentes:

```
# pgrep -l ssh
2427 sshd
7367 sshd
7422 sshd
```

Lista os processos do usuário root que combinem com apache:

```
# pgrep -l -u root apache
2508 apache2
```

pkill – process kill

Permite a finalização de processos baseado no padrão de pesquisa informado. Funciona de forma análoga ao comando pgrep e aceita uma lista reduzida de parâmetros.

Finalizando todos os processos do servidor spamd:

```
# pkill -9 -x spamd
```

nice

Todos os comandos em um sistema operacional são executados em níveis de prioridade padrão, ou seja: nem muito rápido, nem muito lento. O Linux possui uma escala de prioridades que vai de -20 (prioridade máxima) até +19 (prioridade mínima). Qualquer comando ou serviço executado, por padrão, entra em nível zero (meio termo).

Há situações onde é necessário executar um comando com uma prioridade maior ou deseja-se reduzir a utilização de recursos do sistema para uma determinada atividade. O comando nice permite iniciar o comando com um nível de prioridade diferente do padrão.

nice [-n PRIO] comando	
OPÇÃO	Descrição
-n PRIO	Define a prioridade para execução do comando

Executando um backup com baixa prioridade:

```
# nice -n +10 tar -czf /backup.tar.gz /dados
```

Recriando a base do rpm com alta prioridade:

```
# nice -n -10 rpm --rebuiddb
```

renice

Permite que um processo em execução tenha sua prioridade alterada.

Este comando atua sobre os processos em execução, não podendo ser utilizado para iniciar uma tarefa com nível diferente do padrão. Renice pode ser utilizado para a prioridade de uma tarefa ou dos processos de um usuário.

renice PRIO [-p PID] [-u USER]	
OPÇÃO	DESCRIÇÃO
PRIO	Define a prioridade para execução do comando
-p PID	Número do processo a ser alterado
-u USER	Alterar os processos do usuário informado

Redefinindo a prioridade de um processo em execução:

```
# renice -10 -p 795
```

```
# renice +10 -u aluno
```

Alterando a prioridade de todos os processos do usuário aluno:

jobs, fg bg

Sendo um sistema operacional multi-tarefa, o Linux permite que vários processos sejam executados ao mesmo tempo. Isto é possível executando-se alguns processos em primeiro plano e outros processos em segundo plano. Os comandos jobs, fg e bg permitem consultar os processos em execução, colocá-los em primeiro plano ou em segundo plano, respectivamente.

O gerenciamento de processos executados em primeiro plano ou em segundo plano pode ser feito ao iniciar o processo, ou quando este já estiver em execução.

Para iniciar um processo em segundo plano acrescentamos ao final do comando um “&”, por exemplo:

```
# shutdown -r 5 &
[1] 2165
debian:~#
Broadcast message from root (pts/0) (Mon Aug 21 21:41:54 2006):
The system is going DOWN for reboot in 5 minutes!
debian:~#
```

Nesta saída são exibidas várias informações: a primeira é “[1]” que identifica ser este o primeiro processo em segundo plano neste terminal; “**2165**” que é o número do PID deste processo e por último a saída em tela do próprio comando executado.

Poderemos consultar esta tarefa com o comando jobs, cuja função é exatamente esta, consultar as tarefas que estão executando em segundo plano no terminal corrente, aliás, todo processo suspenso ou executando em segundo plano é chamado de job.

```
# jobs  
[1]+  Running           shutdown -r 5 &
```

Ao consultar os processos com o comando jobs, são exibidos seu status e o comando executado. Pode-se utilizar o parâmetro **-l** para listar os PID junto a cada job:

```
# jobs -l  
[1]+  2165 Running       shutdown -r 5 &
```

Expressões regulares

Edição de textos

Editar textos em sistemas Linux é a forma mais comum de configurar o sistema operacional. Devido às suas origens (sistemas Unix®) praticamente todos os serviços de rede ou configurações locais são realizadas a partir de arquivos texto. Em todos os sistemas derivados do padrão POSIX (Unix®, HPUX®, AIX®, Solaris® e sistemas Linux) a edição de textos é uma tarefa constante. Em todas estas implementações é possível encontrar um editor de textos que segue os mesmos padrões e funcionalidades em todas elas: o VI (pronuncia-se vi-ai).

Desenvolvido nos primórdios dos sistemas Unix®, o vi teve sua implementação baseada no bloco alfa-numérico do teclado. Bem mais tarde, muito tempo após o surgimento dos teclados extendidos (setas, movimentação, bloco numérico), em uma outra implementação do vi é que foi possível fazer uso destes recursos. Ainda hoje o vi trabalha em modo de compatibilidade com a versão original, ignorando muitas destas funções.

Apesar de sua simplicidade, o vi, ainda hoje, é utilizado pela grande maioria dos administradores, implementando recursos como “identificação de sintaxe” pelo reconhecimento automático do tipo de arquivo e pesquisa incremental entre outros. Um um arquivo aberto no vi:

```
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local
~
~
~/etc/rc.local" 9L, 222C
```

1,1

All

vim – vi improved

Os recursos extras como sintaxe, identificação de arquivos, pesquisa avançada e uso das teclas extendidas só foram implementados por uma versão melhorada do vi original. Esta versão leva o nome de vim. Com a implementação de um arquivo de configurações bem flexível, é possível habilitar várias funções avançadas não disponíveis normalmente.

O conceito de edição de textos desenvolvido pelo vi se baseia em dois modos de trabalho:

- Modo comando onde é possível instruir o vi a salvar arquivos ou efetuar tarefas
- Modo edição onde o vi aceita os dados digitados pelo teclado como sendo o conteúdo do arquivo

Ao ser iniciado o vi sempre abre em “modo comando” independente de ter sido instruído a carregar um arquivo ou não. Neste modo podemos instruí-lo a efetuar pesquisas, inserir mais dados no arquivo em tela, copiar, colar, mover, executar comandos em um terminal ou simplesmente finalizar a aplicação.

Para poder digitar um texto devemos acessar o “modo edição” do vi. Isto é feito pressionando a tecla “i” (inserir). A partir desta ação, é exibido no canto inferior esquerdo da tela a palavra --INSERT--, indicando que o vi está aceitando a digitação de conteúdo para o arquivo. O modo edição é possível funcionar de duas formas: --INSERT-- onde todo o texto é acrescentado ao texto existente e --REPLACE-- onde o texto digitado sobrescreve o existente caso seja digitado sobre linhas pré-existentes. O acesso ao modo de edição --REPLACE-- só é possível a partir do modo comando pressionando a tecla “R”.

Para realizar um exemplo prático de utilização do vi, siga os passos abaixo:

```
# vi
i
Curso Linux
Edição de textos
<ESC>
:w primeiro- arquivo
:q
```

Comandos e ações dentro do vi/vim

As principais ações dentro do vi/vim envolvem a movimentação dentro do texto, localização, salvamento e edição. A tabela abaixo representa uma pequena parcela das possibilidades que o vim oferece.

MOVIMENTAÇÃO NO TEXTO	
0, \$	Início / Fim da linha
w, b	Próxima palavra, palavra anterior
e	Fim da próxima palavra
h, j, k, l	À esquerda, abaixo, acima e à direita
gg, G	Início do arquivo, fim do arquivo
60gg	Posiciona o cursor na linha 60
ctrl+e	Move a tela uma linha acima
ctrl+y	Move a tela uma linha acima

MANIPULAÇÃO DE ARQUIVOS	
:q, :q!	Sai do vi, força a saída do vi sem salvar alterações
:w, :w!	Grava o arquivo, força a gravação do arquivo
:w arquivo	Grava o texto com o nome arquivo
:wq, ZZ, :x	Salva o arquivo e fecha o vi
:sav novoarquivo	Salva como novoarquivo

TROCA DE MODO DE OPERAÇÃO	
i, a	Insere texto antes do cursor, insere texto após o cursor
I, A	Insere texto partir do ínicio da linha, ou do final da linha
O, o	Nova linha acima do cursor, nova linha abaixo do cursor
<ESC>	Saí do modo edição, retornando ao modo comando

COPIAR E COLAR	
YY, S, DD	Copia uma linha, altera uma linha, deleta uma linha
P, p	Cola antes do cursor, cola depois do cursor
V, v, ctrl+v	Seleciona uma linha, seleciona caracteres, seleciona colunas

DIVERSOS	
/palavra	Pesquisa palavra dentro do texto
n, N	Repete a pesquisa abaixo, repete a pesquisa acima
:%s/velho/novo/g	Pesquisa velho no texto e substitui todas as ocorrências por novo
u, ctrl+r	Desfaz, refaz última ação

Alguns comportamentos do vim podem ser modificados de acordo com várias opções. O Mandriva já traz várias delas pré-definidas de forma global para todos os usuários em /etc/vim/vimrc e algumas personalizações para o usuário root em ~/.vimrc. Algumas opções que podem ser incluídas ao arquivo pessoal:

```
" permite mover o cursor por toda a tela
set ve=all

" define cores apropriadas ao terminal padrão do modo texto
set bg=dark

" marca todos os textos que casarem com o padrão pesquisado em vídeo reverso
set hlsearch

" permite pesquisa incremental
set incsearch

" permite que a tecla backspace remova fins de linha
set backspace=indent,eol,start

" retira o beep do vi
set visualbell

" permite pesquisar um texto ignorando caixa alta/baixa
set ignorecase

" corrige erros comuns de digitação no modo de comandos
cab Wq wq
cab WQ wq
cab W! w!
cab Q! q!
```

Gerenciamento de partições e formatação

Entendo os sistemas de arquivos

Um sistema de arquivos nada mais é do que uma partição do disco e que deve ser acessada através de um diretório. Ao contrário de outros sistemas operacionais, não são alocadas letras de unidades para cada partição ou novo disco instalado em um equipamento Linux.

Toda partição deve ser montada na estrutura de diretórios do Linux para que possa ser acessada, sendo que estas partições podem ser mapeadas em qualquer diretório.

Todo diretório que dá acesso a uma partição é chamado de “ponto de montagem”.

Um sistema de arquivos define com um partição é estruturada sendo que diferentes partições podem ser formatadas com diferentes sistemas de arquivos. Apesar disto, um ponto em comum é o fato de todo sistema de arquivos possuir um “superbloco”.

Um superbloco é uma área especial que armazena informações do disco, localização das tabelas de inodes, uma lista de blocos livres e qual o diretório raiz do sistema. A estrutura exata do superbloco varia de um sistema de arquivos para outro, sendo que sempre há cópias do superbloco pois, se este for danificado, o sistema de arquivos será perdido.

Inodes

Em sistemas Linux, os inodes armazenam informações sobre cada arquivo gravado em disco. Em sistemas de arquivos formatados com EXT2 ou EXT3 o número de inodes é fixo, indicando o número máximo de arquivos que podem ser gravados no disco.

Todo arquivo possui um “inode number” (i-number) que pode ser consultado com um comando “ls -i”.

As informações armazenadas no inode chamadas meta dados são:

- tipo
- usuário dono e grupo
- permissões
- horários
 - mtime (hora de modificação do arquivo)
 - ctime (hora de mudança de permissões ou proprietário/grupo)
 - atime (hora de acesso – leitura)
- número de links físicos
- ponteiros para área de dados

Tipos de sistemas de arquivos

O Linux suporta, normalmente, 30 sistemas de arquivos diferentes sendo que algumas distribuições suportam mais do que isto. Entre os sistemas de arquivos que o Linux pode utilizar estão FAT, VFAT e FAT32 para compatibilidade com mídias removíveis ou partições em disco. Em muitos casos também está disponível suporte a NTFS em modo somente-leitura.

O principal sistema de arquivos utilizado pelo Linux durante muitos anos foi o ext2, porém este não oferece os recursos exigidos pelos sistemas atuais pois não oferecia tolerância a falhas ou recuperação de erros de forma inteligente. Atualmente todas as distribuições adotam algum sistema de arquivos com suporte a journaling como ext3 (uma evolução natural do ext2) ou o reiserfs. Há ainda sistemas como xfs desenvolvido pela Silicon Graphics e jfs da IBM.

Sistema de arquivos EXT2

O sistema de arquivos EXT2 foi criado com uma extensão do sistema EXTfs para oferecer melhor suporte aos arquivos do padrão UNIX: arquivos regulares, links simbólicos e dispositivos, além de ser o primeiro sistema de arquivos a permitir partições grandes com até 4Tb.

Além das características básicas, o EXT2 trouxe recursos avançados como a definição de atributos de arquivos e, se definido em diretórios, a herança destes mesmos atributos, permitindo que o usuário altere o comportamento do kernel do Linux. Recursos como sincronismo de gravação de dados em partições, além de melhorias no tempo de acesso.

Sistema de arquivos EXT3

O sistema de arquivos EXT3 implementa recursos de journaling ao antigo sistema EXT2, reduzindo drasticamente as possibilidades de perda de meta-dados ou a obrigatoriedade de checagem do sistema de arquivos em caso de uma queda de energia ou desligamento incorreto do equipamento.

O journaling permite a recuperação de erros e verificação do sistema de arquivos por meio de logs de transação que registram todas as alterações que devem ser realizadas no disco. Caso alguma tarefa não seja completada devido a uma queda de energia o sistema de arquivos é recuperado com base nas informações armazenadas no journaling.

Dispositivos e sua identificação

Toda unidade de armazenamento é conhecida como um dispositivo de bloco, cada um com características especiais como discos rígidos ou mídias removíveis como cd's ou disquetes e, no Linux, são identificados através de arquivos especiais localizados no diretório /dev. Os principais dispositivos que podem ser identificados no Linux são:

- Dispositivos IDE
 - /dev/hda: disco master da controladora primária
 - /dev/hdb: disco slave da controladora primária
 - /dev/hdc: disco master da controladora secundária
 - /dev/hdd: disco slave da controladora secundária

- Dispositivos SCSI ou SATA
 - /dev/sda: disco SCSI 1 ou SATA
 - /dev/sdb: disco SCSI 2 e assim por diante

De acordo com a unidade utilizada, IDE ou SCSI/SATA, as partições alocadas em cada um disco são identificadas com o prefixo da unidade, por exemplo:

- /dev/hda1: primeira partição do disco master da controladora primária
- /dev/sda1: primeira partição do disco 1 SCSI ou SATA

Particionamento

O gerenciamento de sistemas de arquivos envolvem a criação de partições, formatação e checagem destas para garantir sua integridade.

Para a criação de uma partição o comando fdisk deve receber como parâmetro a unidade a ser manipulada.

Acessando o disco IDE master da controladora primária

```
# fdisk /dev/hda
The number of cylinders for this disk is set to 1044.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
 1) software that runs at boot time (e.g., old versions of LILO)
 2) boot and partitioning software from other OSs
    (e.g., DOS FDISK, OS/2 FDISK)
Command (m for help):
```

Caso se quisesse acessar o primeiro disco SATA de um equipamento utilizariamos /dev/sda como identificação da unidade.

O fdisk possui opções que podem ser consultadas digitando-se m (menu) dentro de sua interface. Entre as principais opções do fdisk temos:

Opção	Descrição
a	Ativa uma partição (flag de inicialização)
d	Deleta uma partição existente
l	Lista os tipos de partições suportadas pelo fdisk
m	Exibe o menu de opções
n	Nova partição
t	Altera a identificação do tipo de partição

Formatando um sistema de arquivos

O processo de formatação no Linux é bem mais rápido que em outros sistemas. Por padrão, este processo não verifica a superfície do disco, limitando-se a criar as trilhas e definir as áreas de dados, superblocos e tabelas de inodes. O comando utilizado para formatar partições é o mkfs, sendo suas principais opções:

Opção	Descrição
-t TYPE	Tipo de sistema de arquivos a ser criado: <ul style="list-style-type: none"> • ext2 • ext3
-C	Força a verificação da superfície do disco
-b SIZE	Tamanho do bloco a ser criado, por padrão 4096, sendo possível utilizar 1024 ou 2048 bytes.
-L LABEL	Rótulo a ser atribuído à partição

Formatando a quarta partição do disco IDE master da controladora primária

```
# mkfs -t ext2 /dev/hda4
```

Criando um sistema de arquivos ext3 (com journaling)

```
# mkfs -t ext3 /dev/hda4
```

Alternativa ao comando mkfs acima

```
# mke2fs -j /dev/hda4
```

Checando um sistema de arquivos

Eventualmente mesmo os sistemas com journaling devem ser testados quanto à sua integridade e corrigidos, se necessário. Este processo é realizado com o comando fsck. As opções de verificação do fsck são:

Opção	Descrição
-p	Recuperação automática (sem perguntas)
-y	Exibe todas as perguntas e assume “yes” como padrão
-c	Força a checagem da superfície do disco
-f	Força a checagem do disco, mesmo que esteja marcado como íntegro (sem falhas).

Forçando a checagem de uma partição

Forçando a checagem de uma partição com correção automática

```
# fsck -p /dev/hda3
```

Como alternativa ao comando fsck, também pode ser utilizado o comando e2fsck.

```
# fsck -f /dev/hda3
```



www.green.com.br