Administração de Sistemas

LINUX

Básico

ÍNDICE

1 - INTRODUÇÃO

Este primeiro capítulo se propõe a responder perguntas básicas sobre o Linux, como: sua origem, suas principais características e sua forma de distribuição aos usuários. Além disso, serão introduzidos alguns dos conceitos básicos do mundo Linux e listadas algumas das fontes de informação disponíveis sobre este sistema operacional.

1.1 - O QUE É?

O Linux é um sistema operacional originalmente desenvolvido em 1991 por Linus Torvalds, um finlandês do Departamento de Ciência da Computação da Universidade de Helsinki. A partir do seu lançamento, milhares de programadores espalhados por todo o mundo contribuíram e continuam contribuindo para o seu desenvolvimento, utilizando a Internet como a grande ferramenta que proporciona o intercâmbio de idéias e o gerenciamento de grupos de trabalho.

O Linux foi criado tendo por base o padrão POSIX, que é o mesmo que deu origem aos diversos "sabores" de UNIX. Sendo assim, podemos dizer que o Linux é um UNIX, mas não que ele é UNIX.

O Linux foi inicialmente desenvolvido para PCs baseados em CPUs x86 (como 386/486/Pentium, etc), porém atualmente existem versões para computadores Alpha da DEC, Sparcs da SUN, CPUs M68000 (semelhantes a Atari e Amiga), MIPS, PowerPCs (tais como iMac...), S/390 (IBM), I64 (Intel Itanium) e PDAs.

O sistema operacional em si é chamado kernel (núcleo) do Linux e é o responsável por gerenciar todas as tarefas do sistema. Todos os demais serviços e programas fazem chamadas ao kernel durante sua execução.

Inicialmente o Linux suportava apenas uma interface por linha de comando, com recursos até hoje considerados poderosíssimos. Atualmente, existem servidores gráficos para Linux, como o "Servidor X", que é de livre distribuição, além de outros, comerciais. Para rodar no servidor X, existe uma infinidade de gerenciadores de janelas (window managers), com as mais diversas opções de interfaces gráficas, que imitam as interfaces do Win95, iMac, OS2, NeXT, etc, além daquelas criadas originalmente para o Linux.

Os principais usos do Linux anteriormente eram como servidor de páginas da Web, servidor FTP, servidor de e-mail, servidor de nomes (DNS) ou roteador (gateway) entre LANs e a Internet. Além destes usos, hoje mais e mais sistemas Linux estão sendo utilizados como servidores de bancos de dados, sendo que a maioria dos mais populares pacotes de bancos de dados estão disponíveis nas versões nativas do Linux, incluindo produtos de empresas como Oracle, Informix, Sybase e IBM. Finalmente, o Linux vem sendo usado cada vez mais para estações de trabalho pessoais ou sistema desktop, utilizados para desenvolvimento de software, computação gráfica, acesso à internet, etc.

Atualmente, tem sido desenvolvidos os mais variados aplicativos para a plataforma Linux, desde editores de texto simples até simuladores de circuitos eletrônicos, passando por jogos e aplicativos para internet, tornando cada vez mais popular e amigável este sistema operacional.

1.2 - CARACTERÍSTICAS

A seguir, descreveremos algumas das características mais importantes do Linux, que o tornam um sistema estável, versátil e confiável.

1.2.1 - Multiusuário:

Isto significa que o Linux instalado em uma máquina pode ser utilizado por mais de um usuário, seja na mesma máquina ou através de terminais remotos ligados a esta máquina. Tudo isso com privacidade, já que o acesso a cada arquivo ou diretório pode ser configurado (individualmente ou em grupos, como veremos mais a frente).

Cada usuário tem acesso ao sistema através do login (entrada no sistema), mediante o uso de senha. Por questões de segurança no Linux não é permitido (por vias normais) que alguém tenha acesso à máquina sem possuir autorização, ou seja, é necessário estar cadastrado no sistema para poder acessá-lo, ao contrário de outros sistemas.

1.2.2 - Multitarefa:

O Linux trabalha com multitarefa real, ou seja, ele pode gerenciar diversas tarefas sendo executadas "ao mesmo tempo" pela máquina. Para gerenciar estas tarefas, o Linux trabalha com o conceito de processos, nome dado a cada programa (ou parte dele) que está "rodando" na máquina. Cada vez que executamos um programa, serão criados um ou mais novos processos no kernel, separados dos demais, os quais serão gerenciados pelo Linux.

Quando inicializamos a máquina e o kernel do Linux está carregado, o próprio kernel já cria o primeiro processo, que é chamado *init*. Ao logarmos na máquina, o programa de login cria um processo para o usuário, que é a interface de linha de comando. A partir daí, o usuário pode criar novos processos, isto é, "rodar" programas.

É importante notar que um processo só pode ser criado a partir de outro (com exceção do primeiro). Ao processo de origem dá-se o nome de "processo-pai" e ao processo originado, chamamos "processo-filho".

Outro conceito importante é a "morte" de um processo. Um processo pode ser "morto" por outro processo e existe um mecanismo que mantém o processo-pai informado sobre seus processos-filhos que existam ou deixem de existir.

O principal aspecto que deve ser considerado, portanto, é que cada processo roda em um ambiente independente, de modo que, se por algum motivo qualquer seja necessário "matar" um processo, isto não irá afetar os demais processos (exceto os processos-filhos). Desta forma, os serviços disponibilizados pela máquina em questão continuariam sendo utilizados normalmente, sem maiores problemas.

1.2.3 - Código Aberto (GPL):

O Linux não é um software de domínio público, mas é distribuído sob a GNU General Public License, que preserva a disponibilidade do seu código fonte. Ou seja, o código fonte do Linux deve estar sempre disponível para qualquer um. Alguém pode cobrar pela cópia do Linux se desejar, desde que, com isso, não limite a distribuição do mesmo.

Por ser um software aberto, alertas para qualquer possível problema de segurança ou falha nos programas, são distribuídos imediatamente pela Internet em busca de soluções, não sendo necessário esperar meses para que um fabricante ou desenvolvedor crie uma solução.

O Linux segue o modelo de desenvolvimento aberto e, por isso, cada nova versão disponibilizada ao público é considerada como um "produto de qualidade", pois qualquer um pode examinar e melhorar o código fonte. Para informar às pessoas se elas estão obtendo uma versão estável ou não, o seguinte esquema foi criado:

Versões r.x.y do kernel, onde x é um número par, são versões estáveis, e, enquanto o y é incrementado, apenas reparos de bugs são efetuados.

Versões r.x.y do kernel, onde x é um número ímpar, são versões beta destinadas

apenas a desenvolvedores; podem ser instáveis e falhar, e estarão recebendo novas características o tempo todo.

De tempos em tempos, com o atual desenvolvimento do kernel sendo considerado "estável", x é mudado para um número par e o desenvolvimento continua com uma nova versão (x ímpar).

1.2.4- Custo:

O custo do Linux varia de acordo com o que você espera obter. É possível obter o Linux pela Internet, pagando apenas o acesso (para fazer o download). Se você tiver acesso pela internet gratuita, o custo será zero!!! (se você considerar que tempo não é dinheiro...).

Se você optar por adquirir uma distribuição completa do Linux, receberá os CDs com os programas, manuais impressos e normalmente terá direito a um determinado tempo de suporte técnico gratuito.

Um ponto importante é que nestas distribuições completas incluem não só o sistema operacional, mas também uma infinidade de programas e aplicativos, inclusive pacotes "office", ferramentas gráficas, compiladores, servidores de aplicações, servidores de Internet, servidores de banco de dados, ERP, etc, tornando seu custo irrisório se comparado à quantidade de programas adquiridos.

Existem algumas "versões econômicas" de distribuições, bem mais baratas que as completas, nas quais você recebe apenas alguns CDs, não recebe os manuais impressos (ou nem todos eles) e normalmente não tem direito a suporte técnico.

Vale ressaltar que geralmente os CDs das distribuições Linux trazem os manuais em formato texto, html ou postscript, portanto, nesta situação pode ser uma boa idéia adquirir uma "versão econômica", principalmente se for para uso doméstico.

1.3 - DISTRIBUIÇÕES

Distribuição é o nome dado a um conjunto de programas constituído por um kernel do Linux e uma variedade de outros softwares para esta plataforma, chamados "pacotes", normalmente distribuídos de uma forma personalizada pela empresa ou grupo distribuidor. Existem diversas distribuições, dentre elas podemos citar:

DISTRIBUIÇÃO	***PAÍS DE ORIGEM***	***BASEADA EM:***
Slackware	Alemanha	
Red Hat/Fedora	USA	
Debian	USA	
S.U.S.E	Alemanha	
Mandrake	França	Red Hat
OpenLinux (Caldera)	USA	
Mandriva Linux	Brasil	Red Hat

Cada distribuição tem suas particularidades, umas são mais amigáveis, outras menos, outras são mais fáceis de instalar ou atualizar, outras mais complicadas. Estas distribuições geralmente desenvolvem programas instaladores e configuradores personalizados, de forma a tornar mais fácil e amigável as tarefas no Linux, porém o núcleo do sistema (kernel) é o mesmo para todas elas. Mais adiante veremos as principais diferenças entre as principais distribuições.

2 - INSTALAÇÃO

Neste capítulo serão tratados os pontos básicos do processo de instalação do Linux em máquinas baseadas em Cpus x86 (plataforma PC).

Serão discutidas as ações pré-instalação e os conceitos envolvidos na instalação de um sistema Linux.

2.1 - PRECAUÇÕES ANTES DA INSTALAÇÃO

O Linux pode ser instalado em um HD separado ou pode compartilhar um HD com outros sistemas operacionais (SO). A quantidade de espaço (no HD) para o Linux depende do modo de instalação escolhido, sendo que este assunto será abordado mais adiante, quando serão discutidas as formas de instalação.

É importante salientar que o Linux tem uma convivência amigável com outros sistemas operacionais, apesar do contrário nem sempre ser verdadeiro. Portanto, é comum termos o Linux instalado em uma máquina ou sistema, juntamente com DOS, Windows, OS/2, Novell Netware, etc.

Assim sendo, antes de instalar o Linux, a primeira coisa com que devemos nos ocupar é com relação à preservação dos dados de outros sistemas operacionais que porventura estejam instalados na mesma máquina. Ou seja, devemos definir onde o Linux será instalado na máquina e a necessidade de realizar backup de alguma informação, na hipótese de reparticionamento de HDs.

No caso em que se dispõe de um HD exclusivo para o Linux, o backup dos dados que possam estar em outros HDs na mesma máquina não é obrigatório, porém o bom senso nos diz que esta é uma boa medida preventiva para evitar dores de cabeça...

Caso o Linux vá compartilhar o HD com outro SO, é preciso criar o espaço necessário (partição) para ele, se ainda não existir, criando pelo menos duas partições, sendo uma do tipo Linux Nativa e outra de Swap (arquivos de troca). Para criarmos estas partições temos três alternativas:

a) No HD existe espaço não particionado

Neste caso, o espaço não atribuído a nenhuma partição será utilizado para criar as partições do Linux. Por exemplo, num HD de 8.4GB em que se tem apenas uma partição de 4.2GB com Windows instalado, os 4.2GB restantes (não particionados) poderão ser usados pelo Linux.

b) No HD existe uma ou mais partições sem uso

Se pelo menos uma das partições do HD estiver sem uso ou puder ter seus dados transferidos para outras partições (de modo a torná-la não usada), esta partição sem uso poderá ser excluída para dar lugar às partições do Linux.

c) Existe espaço livre numa partição já utilizada

Para conseguirmos as partições necessárias ao Linux nestas condições, temos duas opções:

- Reparticionamento não-destrutivo: Isto é feito utilizando softwares específicos que conseguem alterar a tabela de alocação de arquivos (FAT) do HD, diminuindo o tamanho da partição e criando uma nova partição apenas com espaço livre.

Antes de usar este método é recomendado fazer BACKUP de todos os dados importantes que houverem no HD, em seguida desfragmentar o HD, de modo que todos os dados fiquem contidos (compactados) em uma área restrita do HD e utilizar o software específico para "encolher" a partição existente e criar a nova partição. Se a nova partição criada não for do tipo apropriada para Linux, deve-se excluí-la e em seguida recriá-la de modo que possa ser utilizada pelo Linux.

Um software para reparticionamento não-destrutivo normalmente encontrado nas distribuições de Linux é o fips.

OBS.: Pode-se encontrar problemas se, mesmo após a desfragmentação do HD, alguns dados não tiverem sido movidos, permanecendo em áreas que reduzam o tamanho da nova partição ou até mesmo impeçam o reparticionamento não-destrutivo.

- Reparticionamento destrutivo: Este é o processo mais frequentemente utilizado, sendo um dos mais radicais, pois é preciso excluir a partição existente e criar as novas partições.

Portanto, antes de excluir a partição existente, é necessário fazer BACKUP de todos os dados contidos no HD e verificar se todos os originais dos programas instalados estão disponíveis, já que deverão ser reinstalados posteriormente, incluindo o sistema operacional.

Deve-se então excluir a partição existente e criar as novas partições, usando utilitários como o fdisk do DOS ou o fdisk do Linux para este fim. Observe que apesar dos nomes iguais, estes dois utilitários são bastante diferentes, sendo que o primeiro roda em DOS e o segundo em Linux. Todo o cuidado é pouco ao usar estes programas, pois o efeito de uma ação impensada pode ser uma enorme dor de cabeça!!!

Além destes dois programas, atualmente as distribuições têm trazido diversas outras opções, inclusive com interfaces gráficas, para o gerenciamento das partições de HDs. Por exemplo: Disk Druid (Conectiva Linux), DiskDrake (Mandrake), entre outros.

Nota: Atualmente, além das opções de instalar o Linux em partições próprias (nativas), diversas distribuições já possuem o recurso de se instalar o Linux em uma partição DOS. Neste caso é criada uma pasta (diretório) dentro do Windows e esta pasta conterá todo o sistema de arquivos do Linux. Desta forma, portanto, não é necessário reparticionar o HD, porém, é esperada uma queda na performance devido à emulação do sistema de arquivos.

Além das precauções discutidas acima, é igualmente importante dispormos das informações sobre o Hardware da máquina em que o Linux será instalado (placas de vídeo, rede, impressora, teclado, mouse, monitor, etc), caso sejam solicitadas durante a instalação / configuração do sistema. Se a máquina tiver Windows instalado, estas informações podem ser obtidas no Painel de Controle, acessando o ícone Sistema.

2.2 - FORMAS DE INSTALAÇÃO

Geralmente as distribuições de Linux costumam disponibilizar formas de instalação pré-definidas, com algumas particularidades e recursos a mais ou a menos em relação às demais.

Além destas, normalmente é disponibilizada uma instalação personalizada, que enfatiza as necessidades específicas do usuário, proporcionando muita flexibilidade. Pode-se ter completo controle sobre os pacotes que serão instalados no sistema, assim como determinar se será ou não usada dupla inicialização.

Esta forma de instalação é recomendada para quem já tem familiaridade com o Linux e com a manipulação de partições. Dependendo de como será feita a seleção dos pacotes a serem instalados, esta forma de instalação pode ser bem mais demorada que as demais.

2.3 - MEIOS DE INSTALAÇÃO

Resumidamente os seguintes meios podem ser utilizados na instalação do Linux:

- CDROM Local: Se você tem um drive de CDROM e o CD de uma distribuição do Linux.

- Disco Rígido Local: Somente se os arquivos do Linux tenham sido copiados para o disco rígido.

- NFS (via Rede): Se a instalação for efetuada pela rede, será necessário montar o CD do Linux em uma máquina que suporte o padrão ISO-9660 para sistemas de arquivos com extensões Rock Ridge. Esse equipamento deverá suportar ainda NFS. O CDROM deverá ser exportado através do NFS, assim como será necessário conhecer o endereço IP e o caminho do CDROM ou ter o servidor de nomes configurado. Este método requer um disquete extra para suporte à inicialização via rede.

- FTP: Este método requer um disquete extra para suporte à inicialização via rede, o nome ou o endereço IP do servidor FTP a ser utilizado e o diretório onde residem os arquivos da distribuição do Linux.

- HTTP: Este método requer um disquete extra para suporte à inicialização via rede, o nome ou o endereço IP do servidor HTTP a ser utilizado e o diretório onde residem os arquivos da distribuição do Linux.

A maioria dos CDs de Linux já são inicializáveis, porém pode ser que a máquina não aceite o boot pelo CDROM por limitação da BIOS e então será necessário criar um disquete de boot, cuja imagem é normalmente distribuída com o Linux. As distribuições de Linux geralmente incluem um disquete de boot em seus pacotes.

Este disquete poderá ser de grande utilidade caso ocorra alguma falha no sistema de dupla inicialização e não seja possível o boot normal pelo HD. Isto é muito comum quando se tem um HD compartilhado entre Windows e Linux e é feita a reinstalação do Windows, pois o mesmo remove a dupla inicialização do MBR (Master Boot Record, ou Registro Mestre de Inicialização).

Para a criação deste disquete, tradicionalmente é usado o utilitário rawrite, que deve ser executado no DOS.

Inicialmente, etiquete um disco formatado de 3 ½ polegadas com o nome de disco de inicialização local ou algo similar e insira na unidade de disco flexível. Após, execute os seguintes comandos (presumindo que o seu CD seja o drive d:):

```
C:> d:
D:> cd dosutils
D:dosutils>rawrite.exe
Enter disk image source file name: ..imagens\boot.img
Enter target diskette drive: a:
Please insert a formatted diskette into drive A: and press <ENTER>

D:dosutils>
```

O utilitário inicialmente solicitará o nome do arquivo do disco imagem (informar por exemplo boot.img). Após, solicitará o dispositivo de gravação, onde deverá ser informado a:. Para gerar um disco adicional, etiquete um segundo disco e execute o rawrite novamente, informando o nome do arquivo imagem desejado.

2.4 - INSTALAÇÃO

Para iniciar a instalação do Linux, insira o disquete de inicialização (ou o CD-ROM, caso a BIOS aceite inicialização do sistema via CD-ROM) no drive, reinicialize o computador e siga os passos do programa de instalação.

3 - PRIMEIRA UTILIZAÇÃO

Neste capítulo trataremos alguns conceitos básicos antes de iniciarmos o uso do Linux, como o gerenciamento de boot, a entrada e saída do sistema e a utilização de consoles.

3.1 - CONCEITOS BÁSICOS

3.1.1- Carregamento do Sistema (lilo/grub):

Como já foi citado anteriormente, o Linux pode conviver amigavelmente com outros sistemas operacionais instalados numa mesma máquina. Para permitir ao usuário escolher qual sistema operacional será usado a cada vez que a máquina é ligada, o Linux possui gerenciadores de boot.

O gerenciador de boot mais tradicionalmente utilizado no Linux é o LILO (Linux Loader), cujas características veremos brevemente nesta seção. Versões mais atualizadas utilizam o Grub como gerenciador de boot.

Normalmente é recomendado que o gerenciador de boot seja carregado no MBR (Master Boot Record ou Registro Mestre de Inicialização) do HD, pois este é o primeiro dado a ser lido pelo BIOS (Basic I/O System) da máquina. Ele portanto informará ao BIOS qual o próximo endereço a ser lido no HD, que será aquele correspondente ao sistema operacional selecionado pelo usuário. Esta opção é recomendada quando não há outros gerenciadores de boot, como o Boot Manager do OS/2 ou System Commander, já instalados no MBR.

Se no MBR já houver um gerenciador de boot instalado, existe a opção de instalação do mesmo no setor de boot de uma partição marcada como "ativa" (ou seja, capaz de dar boot), pois se no MBR não houver nenhum redirecionamento para algum sistema operacional, o próximo setor a ser lido pelo BIOS é o setor de boot de uma partição "ativa". Da mesma forma, o gerenciador de boot redirecionará a leitura para o sistema operacional escolhido pelo usuário.

O LILO está sujeito a algumas limitações impostas pelo BIOS. Geralmente, a maioria dos BIOS não pode acessar mais de dois discos rígidos e eles não podem acessar qualquer dado armazenado além do cilindro 1023 de qualquer dispositivo. Note que alguns BIOS novos não têm estas limitações, mas isto não é universal.

O Grub por sua vez não possui esta limitação, além da característica de ser executado em modo gráfico.

Após instalarmos o Linux, o gerenciador de boot assumirá este sistema operacional como o default, sendo que esta opção pode ser alterada posteriormente.

Portanto, após ligada a máquina e o BIOS ser carregado, se o LILO estiver instalado será apresentada a seguinte mensagem:

lilo boot:

Se pressionarmos a tecla Tab, serão mostradas no vídeo as opções de inicialização (boot) do sistema. Podemos digitar qual a opção desejada ou aguardarmos o tempo definido para que o LILO carregue o sistema operacional definido como padrão.

Caso o gerenciador de boot instalado seja o Grub, aparecerá sua tela gráfica, com as opções disponíveis. Também possui um default que é executado após um tempo predefinido.

Após isto, o sistema operacional escolhido será carregado na máquina.

3.1.2- Entrada no Sistema (login):

Após o sistema operacional Linux ter sido carregado, o primeiro processo (para o usuário, pois diversos outros serviços já estarão inicializados, como impressão, e-mail, etc) estará rodando na máquina, que é o getty. Ele fornece a tela de login e passa a informação digitada ao sistema para autenticação do usuário, que é feita pelo programa login.

Como já mencionamos anteriormente, o Linux só permite acesso do usuário mediante sua identificação ao sistema e a informação de sua respectiva senha.

Após a inicialização aparecerá no vídeo algo como:

Conectiva Linux Versão 9.0
Kernel 2.4.20-9cl

localhost login: _

A primeira vez que se acessa o Linux, o acesso deverá ser realizado com o superusuário root. Este é o nome da conta que tem acesso completo a todos os componentes do sistema.

Normalmente, a conta de superusuário é somente utilizada na execução de tarefas de administração do sistema, como a criação de novas contas, desligar o sistema, etc. Isso se deve ao fato de que o acesso irrestrito do superusuário quando mal utilizado poderá provocar grandes estragos ao sistema.

Então seja cuidadoso ao acessar o sistema como root e use a conta de superusuário somente quando realmente for necessário.

Para o acesso inicial, informe root na linha de comando login: e pressione "Enter". Aparecerá uma linha de comando Password: , como abaixo:

Conectiva Linux Versão 9.0
Kernel 2.4.20-9cl

localhost login: root
Password: _

Digite a mesma senha criada durante a instalação, pressionando "Enter" ao terminar. Deverá então surgir algo como:

[root@localhost /root]#

Chegamos portanto ao bash (Bourne Again Shell), o shell (interpretador de comandos) padrão do Linux. Já podemos então utilizar a máquina digitando comandos que serão interpretados pelo bash e passados ao kernel para execução.

3.1.3- Saída do Sistema (logout ou exit):

A saída do sistema pode ser realizada de duas formas: usando o comando **exit** ou o comando **logout**. A diferença entre os dois é que **exit** encerra o shell de comandos corrente e **logout** encerra a sessão.

Vale lembrar que no Linux existe já definido um atalho para o comando **logout**, bastando pressionar **Ctrl + d**.

3.1.4 - Encerramento do Sistema (shutdown):

Se o sistema deve ser desligado ou reinicializado, devemos utilizar o comando **shutdown** para fazer isto. Este comando se encarrega dos detalhes do desligamento, de modo que tudo ocorra em ordem, sem danos ao sistema. Ele pode inclusive avisar aos demais usuários com antecedência de que o sistema será paralisado e toma automaticamente providências para a finalização.

Sintaxe do comando **shutdown**:

shutdown [-t segundos] [-rkhncfF] tempo [mensagem de alerta]

Opções:

-k Não desliga realmente o sistema, somente envia mensagens de aviso a todos os usuários.

-r Reinicializa após o desligamento do sistema.

-h Desliga o sistema após a execução do comando.

-c Cancela a execução de um programa shutdown. Não necessita do argumento tempo.

tempo Quanto o sistema deverá ser executado, antes da ação do comando shutdown.

msg-de-aviso
Mensagem a ser enviada a todos os usuários.

OBS.: O argumento tempo pode ter diferentes formatos. Primeiro, ele pode ser informado em um formato absoluto no formato hh:mm, na qual hh é a hora (com 1 ou 2 dígitos) e mm são os minutos da hora (com dois dígitos). O segundo formato tem o formato +m, no qual m é o número de minutos a serem aguardados. A palavra **now** é um nome alternativo para +0.

Exemplo:

shutdown -f -h +2 ``Falha na energia elétrica; Sistema sendo desligado''
shutdown -c ``Energia elétrica restaurada; Desligamento Cancelado''

Vale lembrar que no Linux existe um atalho para reinicializar o sistema, bastando pressionar Ctrl + Alt + Del e será executado o comando **shutdown -t3 -r now**.

Outros comandos para desligar/reiniciar o computador:

sync	→ descarregar buffer de disco.	→ root e não root
shutdown -h 0	→ desliga	→ root
halt	→ desliga	→ root e não root
init 0	→ desliga	→ root
shutdown -r 0	→ reboot	→ root
init 6	→ reboot	→ root
reboot	→ reboot	→ root e não root
<CTRL> + <ALT> + <DEL>	→ reboot	→ root e não root

Obs.: Em *shutdown –h 0* e *shutdown –r 0*, onde *0* significa o tempo de espera em minutos. Quando o valor é *0* (zero), a execução é imediata.
Obs.: Quando um usuário *não root* executa os comandos *halt* ou *reboot*, será solicitado sua senha pessoal.

3.1.5 - Alternando Entre Consoles:

Os sistemas Linux permitem que se trabalhe com mais de um console na mesma máquina. No Conectiva Linux, por default temos acesso a seis consoles, além de mais seis sessões do X Window, sendo que estes parâmetros podem ser alterados no sistema.

Ao inicializar a máquina e logar, normalmente utilizamos (por default) o primeiro console. Se quisermos alternar para o segundo console, basta pressionar **Alt** juntamente com uma das seis primeiras teclas de função (F1 a F6).

Portanto, Alt + F1 corresponde ao primeiro console, Alt + F2 ao segundo, até Alt + F6 que corresponde ao sexto console. As combinações de Alt + F7 a Alt + F12 são reservadas para alternar entre sessões do X Window (interface gráfica), como veremos mais adiante.

4 - COMANDOS BÁSICOS I

Neste capítulo, iniciaremos o contato com os comandos mais comuns e úteis do Linux. Nem todas as opções sobre cada comando serão descritas aqui, apenas as de uso mais prático para a maioria dos usuários.

Para conhecer o conjunto completo das opções de um determinado comando, o usuário é encorajado a buscar estas informações nas páginas de manual (man pages), conforme será descrito a seguir.

Os comandos serão tratados em diversos capítulos, para conciliar a introdução de novos conceitos e a sua aplicação na prática pelo usuário.

4.1 - ls

O comando **ls** lista o conteúdo de um diretório. Quando usado sem opções, lista todos os arquivos não ocultos do diretório, em ordem alfabética, preenchendo tantas colunas quantas couber na tela.

Opções:

-a Lista todos os arquivos presentes nos diretórios, inclusive os ocultos.

-k Caso o tamanho do arquivo seja listado, mostra-o em Kbytes.

-l Além do nome de cada arquivo, lista o tipo, permissões, número de ligações diretas, nome do dono, nome do grupo, tamanho em bytes e data (da modificação, a menos que outra data seja selecionada). Para arquivos com uma data anterior a 6 meses ou com mais de 1 hora no futuro, a data conterá o ano ao invés da hora e dia.

-t Ordena o conteúdo dos diretórios pela data ao invés da ordem alfabética, com os arquivos mais recentes listados no início.

-u Ordena o conteúdo dos diretórios de acordo com a data de último acesso ao invés da data de modificação. No formato longo de listagem, apresenta a data de último acesso ao invés da data de modificação.

-R Lista o conteúdo de todos os diretórios recursivamente.

-X Ordena o conteúdo dos diretórios alfabeticamente pelo nome da extensão (caracteres após o último `.'). Arquivos sem extensão são listados no início.

-1 Lista um arquivo por linha.

--color
Colore os nomes dos arquivos dependendo do tipo.

4.2 - | (pipe)

O pipe (ou duto) é utilizado como conexão de utilitários. É uma maneira de redirecionar as entradas e saídas, de modo que a saída de um comando torna-se a entrada do comando seguinte. Pode-se usar vários pipes em uma mesma linha de comando, de maneira que é possível combinar tantos comandos quantos forem necessários.

Veremos exemplos de aplicação do pipe logo em seguida, depois de conhecermos mais alguns outros comandos do Linux.

4.3 - more

O comando **more** consiste de um filtro para uso na visualização de arquivos em terminais. Este comando só pode paginar o texto para frente (do início para o fim).

Opções:

-d O more irá solicitar instruções ao usuário através da mensagem "[Pressione espaço para continuar, 'q' para finalizar.]" e irá apresentar a mensagem "[Pressione 'h' para instruções.]" ao invés de emitir sinal sonoro quando uma tecla ilegal for pressionada.

Exemplos:

```
more .Xdefaults
more -d .Xdefaults
ls -al --color | more -d
```

4.4 - less

O comando **less** é um comando similar ao **more**, porém ele permite paginar para frente e para trás no texto.

Para movimentar dentro do texto, utilize as teclas Page Up, Page Down, Home, End e as setas de direção. Se desejar recorrer à ajuda dentro do **less**, basta pressionar a tecla **h**. Para sair, pressione a tecla **q**.

Exemplos:

```
less .Xdefaults
ls -al | less
```

4.5 - cd

O comando **cd** (abreviatura de change directory) é utilizado para mudar o diretório corrente. Permite mudar do diretório atual para outro especificado pelo usuário. Se for usado sem argumentos, muda para o diretório pessoal do usuário. A opção ``cd -'' volta ao

diretório anterior, enquanto a opção "cd .." sobe um nível no sistema de arquivos (árvore de diretórios).

O argumento do comando, ou seja, a especificação do diretório para onde se quer mudar, pode ser relativo à posição em que se está ou baseado no diretório raiz (/). Neste último caso, o nome do diretório desejado deve ser precedido de uma "/".

Exemplos:

```
cd /etc/X11    (especificação baseada no diretório raiz)
cd ..
cd X11         (especificação relativa à posição em que se está [/etc])
cd -
cd
```

4.6 - COMANDOS DE DATA E HORA

O comando **date** é utilizado para mostrar a data e a hora do sistema. Exibe a data e hora corrente, desde que aplicado sem parâmetros. Somente o superusuário pode, através do uso de parâmetros associados ao comando **date**, alterar a data e hora do sistema.

Se for dado algum argumento que não comece com ´+´, o comando **date** acerta o relógio do sistema com o tempo e data especificados pelo argumento. O argumento deve consistir somente de dígitos, que tenham o seguinte significado:

MM	mês;
DD	dia do mês;
hh	hora;
mm	minuto;
CC	primeiros dois dígitos do ano (opcional);
YY	últimos dois dígitos do ano (opcional);
ss	segundos (opcional).

Exemplos:

```
date
date 05221600200015
date '+Hoje é %A, dia %d de %B'
date '+Agora são %T'
```

4.7 - man

O comando **man** é utilizado para formatar e exibir as páginas de manual on-line, que são textos descrevendo em detalhes como usar um comando especificado.

Exemplos:

```
man ls
man more
```

4.8 - alias

Cria um apelido para um comando. Tem precedência sobre o comando, ou seja, pode-se criar um alias do tipo: alias ls="ls -la". Toda vez que digitarmos ls na verdade ele executará ls -la.

Exemplos:

```
alias park="shutdown -h now"
alias end="shutdown -r now"
alias ls="ls --color"
alias cd..="cd .."
alias cd-on="mount /dev/cdrom /cd"
alias cd-off="umount /dev/cdrom"
alias x="startx"
```

4.9 - clear

Limpa a tela e posiciona o cursor no canto superior esquerdo do vídeo. Similar ao cls do DOS.

5 - EDITANDO TEXTOS

5.1 - nano

Neste capítulo conheceremos um editor de textos básico do Linux, o nano. Ele é um dos mais simples editores, porém satisfaz aos iniciantes justamente pela sua facilidade de uso, sendo ideal para pequenas edições em arquivos de configuração, pequenos textos, etc.

Vale citar que neste campo existem dois poderosíssimos editores rivais: o Emacs e o VI, sendo que ambos possuem versões em modo texto e com front-end gráfico, para serem utilizados sob o X Window. Apesar de toda uma rivalidade entre os simpatizantes de cada um destes dois programas, ambos são excelentes editores, sendo bastante avançados em relação ao nano e aos demais editores encontrados no ambiente Linux.

Sintaxe:

nano [opções] [nome_do_arquivo]

Principais opções:

+*n*

Faz o editor ser iniciado com o cursor localizado *n* linhas dentro do arquivo.

-b

Habilita a opção de substituir textos encontrados pelo comando Ctrl+W (Where is).

-e

Habilita a função de auto-completar nomes de arquivos no manipulador de arquivos.

-j

Habilita o comando "Goto" no manipulador de arquivos.

-k

Faz com que o comando Ctrl+K (Cut Text) remova caracteres da posição do cursor até o fim da linha, ao invés de remover a linha inteira.

-m

Habilita as funcionalidades do mouse. Isto funciona apenas quando o nano é executado em um terminal dentro do X Window.

-r*n*

Especifica a coluna usada para limitar os comandos de justificação (margem direita).

-s *speller*

Especifica um programa alternativo de correção ortográfica, quando for utilizado corretor ortográfico.

-v

Apenas permite visualizar o arquivo, desabilitando qualquer edição do mesmo.

-w

Desabilita quebra de linha (desta forma, permite edição de linhas longas).

-x

Desabilita o menu das teclas de comando no rodapé da tela.

Descrição das funcionalidades do **nano**:

Os comandos são mostrados no rodapé da tela e uma ajuda sensível ao contexto é oferecida. Assim que os caracteres são digitados eles são imediatamente inseridos no texto.

Os comandos de edição são entrados usando combinações de teclas com a tecla CONTROL (Ctrl). O editor tem cinco funções básicas: justificação de parágrafos, pesquisa, cortar / colar blocos, um corretor ortográfico e um manipulador de arquivos.

A justificação de parágrafos utiliza o comando Ctrl+J e ocorre no parágrafo onde está o cursor ou no parágrafo imediatamente abaixo, se o cursor está entre as linhas. Os parágrafos são delimitados por linhas em branco ou por linhas começando com um espaço ou uma tabulação (Tab). O comando de justificação pode ser desfeito imediatamente após a sua execução, utilizando a combinação das teclas Ctrl+U.

As pesquisas de strings são executadas por meio do comando Ctrl+W e não são sensíveis à maiúsculas / minúsculas. A pesquisa é iniciada na posição corrente do cursor e vai até o final do texto. A string mais recentemente pesquisada é oferecida como default na pesquisa seguinte.

Blocos de texto podem ser movidos, copiados ou deletados com o uso criativo dos comandos Ctrl+^ (para marcar), Ctrl+K (para deletar) e Ctrl+U (para desfazer). O comando Ctrl+K removerá o texto que se encontra entre a "marca" feita e a posição corrente do cursor e o colocará no buffer. O comando Ctrl+U faz uma "colagem" do texto do buffer a partir da posição corrente do cursor.

O corretor ortográfico examina todas as palavras do texto. Ele então mostra cada uma das palavras para correção, enquanto a destaca no texto. A correção ortográfica pode ser cancelada a qualquer momento. Alternativamente, o editor poderá substituir a rotina de correção ortográfica padrão por uma outra, definida pela variável de ambiente SPELL.

O manipulador de arquivos é oferecido como uma opção nos prompts dos comandos Ctrl+R (Read File) e Ctrl+O (Write Out). Ele é destinado a ajudar na busca por arquivos

específicos e na navegação de hierarquias de diretórios. Nomes de arquivos com seus tamanhos e nomes de diretórios no diretório de trabalho corrente são apresentados para seleção. O diretório de trabalho corrente é mostrado na linha mais acima na tela, enquanto a lista de comandos disponíveis aparece nas duas linhas de baixo. Algumas funções básicas para manipulação de arquivos são suportadas: renomear, copiar e deletar.

Mais ajuda específica está disponível no "Help OnLine" do nano (em inglês), que pode ser acessada pelo comando Ctrl+G. Para sair do Help, deve ser utilizado o comando Ctrl+X.

Para finalizar o **nano**, é utilizado o comando Ctrl+X, o qual apresentará uma mensagem perguntando se deseja salvar as modificações feitas e ainda não salvas, caso o arquivo sendo editado tenha sido alterado. Caso seja respondido "N", as alterações feitas serão perdidas (não serão salvas).

Quando o nano está sendo executado e é desconectado, ele salvará o trabalho corrente (caso necessário) antes de ser encerrado. O trabalho será salvo com o nome corrente do arquivo, sendo acrescentado .save a este nome. Se o trabalho corrente não possuir nome, ele será salvo como "nano.save".

A maneira que as linhas mais longas que a largura da tela são repartidas não é imediatamente óbvia. Linhas que continuam além da largura da tela são indicadas por um caracter "$" ao fim da linha. Linhas longas são roladas horizontalmente assim que o cursor se move através delas.

Exemplos:

```
nano  nome_do_arquivo
nano  +5  nome_do_arquivo
nano  -b  -k  nome_do_arquivo
nano  -v  nome_do_arquivo
nano  -wx  nome_do_arquivo
```

5.2 - vi

O editor de textos (modo caracter) vi é muito utilizado no mundo Linux para editar arquivos de configuração. O vi apresenta 2 modos de uso:

a) Modo de edição → quando pressionamos a tecla < i >.
b) Modo de comando → quando "chamamos" o vi ou desabilitamos o modo de edição pressionando a tecla <ESC>.

Operações básicas do modo de comando:

Obs.: Conceito de linha: Considerasse linha o texto digitado até o próximo enter.

x	→ apaga um caracter
yy	→ copia a linha corrente
nyy	→ copia n linhas
D	→ remove a linha corrente para posterior "colagem"
nD	→ remove n linhas para posterior "colagem"

Obs.: Notar que é necessário pressionar a tecla <shift> para obter D (maiúsculo)

p → cola o texto copiado ou removido para a memória, após o cursor

Obs.: Notar que é necessário pressionar a tecla <shift> para obter P (maiúsculo)

dd → remove a linha corrente
ndd → remove n linhas
dDD → apaga do cursor até o final da linha

Obs.: Notar que é necessário pressionar a tecla <shift> para obter DD (maiúsculo)

cc → elimina a linha corrente, permitindo a inclusão imediata de uma nova linha
ncc → elimina n linhas, permitindo a inclusão imediata de uma nova linha

o → insere linha em branco abaixo da linha corrente e habilita edição
O → insere linha em branco acima da linha corrente e habilita edição

Obs.: Notar que é necessário pressionar a tecla <shift> para obter O (maiúsculo)

u → desfaz as últimas alterações
. → refaz o que "u" desfez

/ → procura palavra
n → continua (next) a procura da palavra, para frente
N → continua (next) a procura da palavra, para trás

Obs.: Notar que é necessário pressionar a tecla <shift> para obter N (maiúsculo)

e → avança para a próxima palavra (final da palavra) após o cursor
b → retrocede para a palavra (início da palavra) anterior ao cursor

<shift> :% s / termo-antigo / termo-novo → substitui o termo-antigo pelo termo-novo
<shift> :e nome-do-arquivo → edita outro arquivo (novo ou já existente)
<shift> :r nome-do-arquivo → insere na posição do cursor o arquivo especificado
<shift> :q! → sai sem salvar
<shift> :wq (ou x) → sai gravando
<shift> :wq! (ou x!) → sai gravando, forçando
<shift> :w → salva sem sair
<shift> :w! → salva sem sair, forçando

Obs.: Se digitarmos no prompt do Linux apenas vi, o editor entrará em operação com um arquivo em branco e sem nome. Se digitarmos no prompt do Linux vi <nome-do-arquivo>, o editor entrará em operação com um arquivo em branco e com nome. Se digitarmos no prompt do Linux vi +8 <nome-do-arquivo>, abriremos o arquivo especificado na linha 8.

6 - COMANDOS BÁSICOS II

Neste capítulo veremos mais alguns dos principais comandos do Linux, utilizados para criar diretórios, remover, copiar, mover e procurar arquivos.

6.1 - mkdir

Este comando é utilizado para a criação de diretórios. Sua sintaxe é:

mkdir [opções] <caminho>

As opções mais utilizadas são:

-p Cria todos os diretórios especificados no caminho;
-m Especifica as permissões de acesso do novo diretório.

Exemplos:

mkdir meu_diretorio
mkdir -p um dois três

6.2 - rmdir

Este comando é utilizado para a criação de diretórios. Sua sintaxe é:

rmdir [opções] <caminho>

As opções mais utilizadas são:

-p Remove uma árvore de diretórios vazia (sem arquivos).

Exemplos:

rmdir meu_diretorio
rmdir -p um/dois/três

6.3 - rm

Este comando é utilizado para remover arquivos. Pode remover também diretórios.

Sintaxe:

rm [opções] <arquivos>

Opções mais utilizadas:

-f Não solicita confirmação.
-i Solicita confirmação. (Caso sejam informados -f e -i, somente o último terá efeito).
-r Remove as árvores de diretórios recursivamente.
-R Remove as árvores de diretórios recursivamente.

Exemplos:

rm meu_diretorio
rm -ir meu_diretorio

rm -rf um dois tres

6.4 - cp

Comando utilizado para copiar arquivos e diretórios. Pode-se copiar um arquivo para um destino informado, ou copiar arbitrariamente muitos arquivos para o diretório de destino.

Sintaxe:

cp [opções] <origem> <destino>

Principais opções:

-f Remove um arquivo de destino já existente.
-i Pergunta se deve regravar arquivos já existentes.
-R Copia diretórios recursivamente, preservando arquivos que não sejam diretórios.

Caso o último argumento denomine um diretório existente, este comando copiará cada arquivo de destino naquele diretório (mantendo o mesmo nome). Caso dois arquivos sejam informados, ele copiará o primeiro no segundo.

Exemplos:

cp /etc/fstab fstab
cp -R /etc/rc.d /root/etc

6.5 - mv

Comando utilizado para mover e renomear arquivos e diretórios.

Sintaxe:

mv [opções] <origem> <destino>

Principais opções:

-f Remove os arquivos de destino, sem solicitar a confirmação pelo usuário.
-i Solicita confirmação para sobrescrever arquivos de destino.
-v Lista o nome de cada arquivo antes de removê-lo.

Caso o último argumento seja o nome de um diretório existente, este comando moverá cada arquivo informado para o diretório, mantendo o nome original. Por outro lado, caso somente dois arquivos sejam informados, altera o nome do primeiro para o segundo.

Exemplos:

mv /etc/fstab /etc/fstab.bak
mv -v /etc/fstab.bak /etc/fstab

6.6 - find

Comando utilizado para pesquisar arquivos em uma hierarquia de diretórios.

O comando **find** pesquisa a árvore de diretórios com raiz dada pelo nome do arquivo fornecido para avaliação, através de uma expressão avaliada da esquerda para a direita, de acordo com as regras de precedência, até que o resultado seja conhecido. Neste ponto **find**

vai para o próximo nome de arquivo.

Sintaxe:

find [caminho] [expressão]

Algumas opções:

-name pattern O nome do arquivo (o caminho à frente do nome do arquivo não é considerado) deve coincidir com os padrões informados em pattern. Os metacaracteres (`*', `?', e `[]') não combinam com um .' no início do nome do arquivo.

-iname pattern Como -name, mas o teste de padrão não é sensível a maiúsculas e minúsculas. Por exemplo, os padrões `fo*' e `F??' coincidem com os nomes de arquivos `Foo', `FOO', `foo',`fOo', etc.

O primeiro argumento que começar com `-', `(', `)', `,', ou `!' é colocado no início da expressão. Quaisquer argumentos antes disso são caminhos para pesquisa e quaisquer argumentos após constituem o restante da expressão. Caso nenhum argumento seja fornecido, o diretório atual será utilizado.

Exemplos:

find -name login
find /bin -name login
find -iname Netscape
find -iname *Navigator

7 - TIPOS DE ARQUIVOS

Neste capítulo conheceremos os tipos de arquivos permitidos pelo Linux e suas características.

Os arquivos no Linux podem ter nomes com até 255 caracteres e múltiplas extensões (partes separadas por um ponto "."), sendo que não existem padrões de extensão que forcem o arquivo a ser de um determinado tipo (como ".exe", ".com" e ".bak" no DOS).

Alguns caracteres não devem ser utilizados em nomes de arquivos, como "!", "*", "$" e "&". Espaços em branco são permitidos, porém não recomendáveis. Ao utilizar nome de arquivo que contenha espaços, o mesmo deve ser digitado entre aspas duplas: "nome do arquivo", por exemplo.

Além disto, devemos estar atentos para o fato de que o Linux é "case sensitive", ou seja, faz distinção entre maiúsculas e minúsculas. Por exemplo, "nome_do_arquivo" é diferente de "Nome_do_arquivo", que por sua vez é diferente de "Nome_do_Arquivo".

Basicamente, o Linux suporta quatro tipos de arquivos: regulares, de diretório, especiais de caracteres e especiais blocados. A seguir, veremos as particularidades de cada um destes tipos.

Os arquivos regulares são aqueles que contêm informações de usuários, por exemplo, tipo ASCII.

Diretórios são arquivos usados na manutenção do sistema de arquivos.

Arquivos especiais de caracteres estão diretamente ligados à entrada / saída e são usados para dispositivos seriais de entrada / saída, tais como terminais, impressoras e placas de rede.

Os arquivos especiais blocados são usados para modelar dispositivos.

Alguns arquivos, apesar de se incluírem num destes tipos, possuem características particulares que os tornam um pouco diferentes, por isto vamos comentá-los aqui:

- Arquivos de backup: terminam com o caracter "~";
- Links: São ponteiros para outro arquivo (ou diretório). Muito semelhantes aos "atalhos" do Windows.
- Arquivos ocultos: são aqueles que têm nomes iniciados por um ponto (".").

Ao executarmos o comando ls -F , um caracter será adicionado ao final do nome de cada arquivo (exceto para arquivos comuns), indicando o seu tipo, conforme abaixo:

`*****' Para arquivos comuns que sejam executáveis;

`**/**' Para diretórios;

`**@**' Para ligações simbólicas (links);

`**|**' Para FIFOs;

`**=**' Para sockets.

7.1 - Links

Há dois conceitos de `links' no Linux, normalmente chamados link direto (hard link) e link simbólico (soft link). Um link direto é somente um nome para um arquivo (e um arquivo pode ter diversos nomes).

Ele será removido do disco quando o último arquivo for removido. Não há algo como um nome original, ou seja, todos os nomes têm o mesmo status.

Normalmente, mas não necessariamente, todos os nomes do arquivo são encontrados no mesmo sistema de arquivos em que o arquivo está.

Já um link simbólico ou soft link, é uma entidade totalmente diferente: é um pequeno arquivo especial que contém um caminho.

Além disto, os links simbólicos podem apontar para arquivos em diferentes sistemas de arquivo (possivelmente arquivos NFS montados a partir de diferentes máquinas), e não necessitam apontar para arquivos realmente existentes.

7.2 - Metacaracteres

Metacaracteres são caracteres que representam o nome de um grupo de arquivos. Vejamos os exemplos a seguir:

```
# ls
doc1    doc2    sessao1     sessao2     sessao3
```

Asterisco (``*''): Substitui por 0 ou mais caracteres quaisquer.

```
ls se*
sessao1     sessao2     sessao3

# ls *1
```

doc1 sessao1

Intervalo de caracteres (``[ ]'')

```
# ls sessao[12]              lista arquivos terminados por 1 e 2
sessao1    sessao2

# ls sessao[1-9]             lista arquivos terminados por 1 até 9
sessao1    sessao2     sessao3
```

Interrogação (``?''): Substitui por um caractere qualquer.

```
# ls doc?
doc1        doc2
```

8 - COMANDOS BÁSICOS III

Neste capítulo conheceremos o comando ln, utilizado para criar links.

8.1 - ln

O comando ln é utilizado para criação de links (ligações) entre arquivos. Se for usado sem opções, por default ele cria links diretos.

Sintaxe:

ln [opções] origem [destino]
ln [opções] origem... diretório

As opções mais utilizadas são:

-f Remove arquivos de destino já existentes.

-i Solicita confirmação antes de remover os arquivos de destino.

-s Cria um link simbólico ao invés de links diretos.

-v Lista o nome de cada arquivo antes de criar a ligação.

Caso somente um arquivo seja informado, ele liga o arquivo no diretório atual, isto é, cria uma ligação para aquele arquivo no diretório atual, com o nome igual ao nome daquele arquivo.

De outra forma, caso o último argumento seja um diretório existente, ln criará uma ligação para cada arquivo mencionado na origem naquele diretório, com o nome igual ao nome do arquivo de origem.

Se somente dois arquivos forem informados, ele cria uma ligação chamada destino para o arquivo origem.

Ocorrerá um erro se o último argumento não for um diretório e mais de dois arquivos forem informados.

Exemplos:

ln /etc/fstab /etc/fstab2

```
ln  /etc/fstab
ln  -s  /etc/fstab  /etc/fstab2
ln  -s  /etc  dir_etc
```

9 - USUÁRIOS E GRUPOS

Veremos a seguir o que são usuários e grupos do sistema, bem como sua utilização no Linux. Mais adiante, aprenderemos como adicionar e remover usuários e grupos do sistema.

9.1 - Porque Criar Usuários?

Num primeiro momento, quem inicia no uso do Linux pode questionar a necessidade de se criar usuários, principalmente se já tem experiências utilizando DOS / WINDOWS.

A pergunta clássica seria: "Se posso fazer tudo como root, porque preciso criar um outro usuário?"

Realmente, como root você pode fazer TUDO mesmo, inclusive danificar o sistema acidentalmente.

É para evitar isto que deve-se criar usuários com menos "poderes" dentro do sistema, de modo a torná-lo menos vulnerável a este tipo de problema. Isto também aumenta consideravelmente a segurança do sistema, pois qualquer invasor, se não estiver como root, pouco dano poderá causar ao sistema.

Um outro objetivo que alcançamos ao criarmos vários usuários, é que cada um deles pode manter sigilo absoluto em relação aos demais, se desejado. Isto inclui todos os arquivos pessoais do usuário, inclusive e-mail, news, etc.

Concluindo, é altamente recomendado que sempre se acesse o sistema como usuário

OBS.: Diz-se que o usuário cadastrado no sistema possui uma "conta", a qual muitas vezes é referenciada como se fosse o usuário. Portanto, é comum dizer "criar uma nova conta" ao invés de "cadastrar um novo usuário".

9.2 - O Conceito de Grupo

Todos os usuários pertencem a um ou mais grupos. Como veremos mais adiante, no Linux cada arquivo tem um dono específico. Por conseqüência, cada arquivo pertence ao mesmo grupo do usuário proprietário.

O grupo pode ser exclusivo do dono do arquivo, ou compartilhado por diversos usuários. A habilidade de ler, gravar ou executar um arquivo pode ser atribuído a um grupo, separadamente das permissões do dono do arquivo. Por exemplo, o dono do arquivo pode ser capaz de gravar um documento, enquanto os membros do grupo somente poderão lê-lo.

10 - PERMISSÕES DE ACESSO

Para cada arquivo ou diretório, consideram-se três categorias de usuários:

- Dono: Quem criou o arquivo.
- Grupo: Grupo ao qual pertence o dono do arquivo.
- Outros: Usuários que não se enquadrem nas categorias anteriores.

Todo arquivo determina quais usuários têm acesso a ele e com que finalidade.

Cada conjunto de permissões de acesso significa presença ou ausência de permissões para: leitura (r); escrita (w); execução (x), conforme a tabela abaixo:

Modo de Acesso	Arquivo comum/especial	Diretório
Leitura ``r''	examinar conteúdo de arquivo	listar arquivos do diretório
Escrita ``w''	alterar o conteúdo do arquivo	escrever no diretório
Execução ``x''	executa o arquivo como comando	pesquisar o diretório

Codificação utilizada para as permissões de acesso:

Tipo	***Proprietário***			***Grupo***			***Sistema***		
	leitura	escrita	execução	leitura	escrita	execução	leitura	escrita	execução
d	r	W	x	r	w	x	r	w	x
l	-	-	-	-	-	-	-	-	-
-									
	1	2	3	4	5	6	7	8	9	10

1 - informa o tipo de arquivo (d para diretório, l para link, - para demais arquivos)
2 - Permissões do Proprietário (r leitura permitida, - não permitida leitura)
3 - Permissões do Proprietário (w escrita permitida, - não permitida escrita)
4 - Permissões do Proprietário (x execução permitida, - não permitida execução)
5 - Permissões do Grupo (r leitura permitida, - não permitida leitura)
6 - Permissões do Grupo (w escrita permitida, - não permitida escrita)
7 - Permissões do Grupo (x execução permitida, - não permitida execução)
8 - Permissões do Sistema (r leitura permitida, - não permitida leitura)
9 - Permissões do Sistema (w escrita permitida, - não permitida escrita)
10 -Permissões do sistema (x execução permitida, - não permitida execução)

11 - COMANDOS BÁSICOS IV

Neste capítulo trabalharemos com os comandos utilizados para adicionar e remover usuários e grupos do sistema, definir e alterar senhas, alterar as permissões de arquivos e alguns outros recursos para a administração de um sistema Linux.

11.1 - useradd

O comando useradd é utilizado para criar um novo usuário no sistema ou para atualizar os dados de um novo usuário.

Quando utilizado sem a opção -D, este comando criará uma nova conta usando os valores especificados na linha de comando e os defaults do sistema. A nova conta passará a fazer parte do sistema de arquivos, um diretório pessoal (home) para o novo usuário será criado e os arquivos iniciais serão copiados, dependendo das demais opções da linha de comando. Em algumas versões deste comando (como da Red Hat e derivados) será criado também um grupo para cada usuário adicionado ao sistema, a menos que a opção -n seja usada.

Quando utilizado com a opção -D, o comando **useradd** mostrará os dados default atuais do sistema ou atualizará estes dados, conforme informado na linha de comando.

Sintaxes:

useradd [opções] *nome_da_conta*

Principais opções:

-d *home_dir*

Criará o diretório *home*_dir para o novo usuário criado. Se esta opção não for utilizada, o nome do diretório criado será o mesmo da conta.

-e *expire_date*

A data na qual a conta do usuário será desabilitada. A data *expire_date* deve ser especificada no formato MM/DD/YY. Para habilitar novamente a conta, deve ser utilizado o comando **usermod -e** *expire_date* , onde *expire_date* é a nova data de expiração da conta do usuário.

-n

Um grupo com o mesmo nome do novo usuário será criado por default.

-u *uid*

O valor numérico de identificação do usuário. Este valor deve ser único e não negativo. Se não usada esta opção, será usado, por default, o menor ID disponível que seja maior que 99 e maior que os demais IDs dos demais usuários. Os valores entre 0 e 99 são tipicamente reservados para contas do sistema (system accounts).

useradd -D [opções]

-b *default_home*

Permite especificar o caminho inicial dos diretórios pessoais (home) dos novos usuários, normalmente /home.

-e *default_expire_date*

Permite especificar a data na qual a conta do usuário será desativada.

-f *default_inactive*

Permite especificar quantos dias após uma senha ter expirado devem ser aguardados, antes da conta ser desativada.

-g *default_group*

Permite especificar o nome do grupo ou ID para ser o grupo inicial de novos usuários.

-s *default_shell*

Permite especificar o nome do shell para os novos usuários.

As informações sobre usuários, senhas, grupos, etc, ficam localizadas nos seguintes

arquivos:

/etc/passwd	Informações de contas de usuário
/etc/shadow	Informações de contas de usuário seguras
/etc/group	Informações de grupos
/etc/default/useradd	Informações default do sistema
/etc/login.defs	Parâmetros do sistema
/etc/skel	Diretório que contém os arquivos default

OBS.: No Conectiva Linux existe por default um link para o comando **useradd**, que é chamado **adduser**. Portanto, neste sistema pode-se utilizar tanto **useradd**, que é o original, como **adduser**.

Exemplos:

useradd -D
useradd *seu_nome*
useradd -d *meu_diretorio meu_nome*
useradd -e *07/28/00* sua*_conta*

11.2 - passwd

O comando **passwd** é utilizado para atualizar a senha de autenticação de um usuário. Normalmente cada usuário pode alterar somente a sua própria senha, exceto o superusuário (root) que pode atualizar a senha de outro usuário, fornecendo o *nome_do_usuário*. Se não for especificado o *nome_do_usuário*, será atualizada a senha do usuário corrente.

Sintaxe:

passwd [opções] [*nome_do_usuário*]

Opções:

-u

Indica que a atualização somente pode ser efetuada para senhas expiradas, mantendo-se a senha atual até a data de sua expiração.

Exemplos:

passwd
passwd *seu_nome*

11.3 - chown

O comando **chown** é utilizado para alterar o dono e/ou grupo de arquivos.

Ele muda o dono e / ou o grupo de um determinado arquivo, de acordo com o primeiro argumento não opcional informado, o qual é interpretado conforme abaixo:

Caso somente um nome de usuário ou identificação numérica de usuário for informada, o usuário é transformado no dono do arquivo informado e o grupo do arquivo não é alterado.

Se o nome do usuário é seguido por dois pontos ou ponto e um nome de grupo ou

identificação numérica de um grupo é fornecido, sem espaços entre eles, o grupo do arquivo também será alterado.

Se os dois pontos ou o ponto e o grupo são informados, mas o nome de usuário é omitido, somente o grupo do arquivo será alterado.

Sintaxe:

chown [opções] [*usuário*] [:] [*grupo*] *arquivo*...

Principais opções:

-v Descreve as mudanças de propriedade realizadas.

-R Altera as propriedades dos diretórios e seus conteúdos de maneira recursiva.

Exemplos:

```
chown -v root /home/meu_diretorio
chown -v :root /home/meu_diretorio
chown -v root:root /home/sua_conta
```

11.4 - chgrp

O comando **chgrp** é usado para alterar o grupo ao qual pertencem os arquivos. Ele muda o grupo de cada arquivo passando-os para um novo grupo, o qual é informado através de seu nome ou de sua identificação numérica.

Sintaxe:

chgrp [opções] *grupo arquivo...*

Principais opções:

-v Descreve as mudanças de propriedade realizadas.

-R Recursivamente muda a propriedade dos diretórios e seus conteúdos.

Exemplo:

```
chgrp -v root /home/seu_nome
```

11.5 - chmod

O comando **chmod** é utilizado para alterar as permissões de acesso aos arquivos para as novas definições informadas.

Pode ser utilizado de dois modos: utilizando uma representação simbólica das mudanças a serem feitas ou utilizando um número em formato octal que represente o padrão de bits das novas permissões.

O formato do modo simbólico é `[ugoa...][[+-=][rwxXstugo...]...][,...]'.

Múltiplas operações simbólicas podem ser informadas, separadas por vírgulas.

A combinação das letras `ugoa' controla quais usuários podem acessar o arquivo que será alterado: o dono do arquivo (u), outros usuários do grupo do arquivo (g), outros usuários não pertencentes ao grupo do arquivo (o), ou todo e qualquer usuário (a). Caso nenhum desses seja informado, o comando assume a opção `a' , porém bits configurados através da opção umask não serão afetados.

O operador `+' causa a adição das permissões informadas às permissões existentes para o arquivo; `-' provoca a sua remoção, e `=' provoca a mudança completa das permissões para as informadas.

As letras `rwxXstugo' selecionam as novas permissões para os usuários definidos:

Letra	***Significado***
r	Leitura.
w	Gravação.
x	Execução (ou acesso a diretórios).
X	Execução somente se o arquivo ou diretório já tem permissão de execução para algum usuário.
s	Configurar o usuário ou identificação do grupo durante a execução.
t	Salvar a área de texto do programa na área de swap.
u	As permissões que o usuário proprietário do arquivo possui atualmente.
g	As permissões que outros usuários do grupo do arquivo têm para acessá-lo.
o	As permissões que outros usuários não pertencentes ao grupo do arquivo têm.

Pode-se utilizar também o modo numérico com um a quatro dígitos da base octal (0-7), derivados da adição dos bits com valores 4, 2, e 1, conforme tabela abaixo:

Decimal	4	2	1
0	-	-	-
1	-	-	¨
2	-	¨	-
3	-	¨	¨
4	¨	-	-
5	¨	-	¨
6	¨	¨	-
7	¨	¨	¨

Qualquer bit omitido é assumido como tendo o valor zero.

O primeiro dígito seleciona a identificação de usuário (4), a seleção do grupo (2) e o salvamento dos atributos da imagem do arquivo (1).

O segundo dígito seleciona as permissões para o dono do arquivo: leitura (4), gravação (2) e execução (1).

O terceiro dígito seleciona as permissões de outros usuários do grupo do arquivo, com os mesmos valores do dono.

O quarto dígito faz o mesmo para outros usuários que não estejam no grupo do arquivo, também como os mesmos valores.

O chmod nunca muda as permissões de links simbólicos; a chamada ao sistema chmod não pode fazê-lo. Isso não é um problema desde que as permissões de links simbólicos nunca sejam utilizadas. Porém, para cada link simbólico informado na linha de comando, o chmod altera as informações do arquivo apontado pela ligação. Por outro lado, o chmod ignora links simbólicos encontrados durante a opção recursiva em diretórios.

Sintaxe:

chmod [opções] *modo arquivo...*

Principais opções:

-v Descreve as permissões alteradas.

-R Altera as permissões de diretórios e seus conteúdos de forma recursiva.

Exemplos:

```
chmod -v 777 /home/seu_nome
chmod -v 100 /home/seu_nome
chmod -v 400 /home/seu_nome
chmod -v u+w, u+x /home/seu_nome
chmod -v g+w /home/seu_nome
chmod -v g-w /home/seu_nome
```

11.6 - su

O comando **su** é usado para permitir que um usuário torne-se outro temporariamente. Ele executa um interpretador com a identificação real e efetiva de usuário, identificação de grupo e grupos suplementares do usuário. Caso o usuário não seja informado na linha de comando, o padrão é o superusuário (root). O interpretador executado é o especificado para o usuário no arquivo passwd, ou /bin/sh caso nenhum seja especificado.

Caso o novo usuário tenha senha, su solicita a senha, a menos que se tenha a identificação real de usuário igual a 0 (superusuário).

Sintaxe:

su [opções] [usuário]

Principais opções:

-s interpretador

Executa INTERPRETADOR ao invés do interpretador definido em /etc/passwd, a menos que o usuário que esteja executando su não seja o superusuário e o interpretador do usuário não seja restrito.

OBS.: Para retornar ao usuário original, basta digitar "exit".

Exemplos:

```
su
su seu_nome
su -s /bin/tcsh seu_nome
```

11.7 - userdel

O comando **userdel** é usado para remover uma conta de usuário e os arquivos relacionados a ele. Este comando modifica os arquivos de contas, apagando todas as entradas que se referem ao usuário.

Sintaxe:

userdel [-r] nome_do_usuário

Se a opção -r for utilizada, os arquivos no diretório home do usuário serão removidos junto com o próprio diretório. Os arquivos localizados em outros sistemas de arquivos terão de ser procurados e deletados manualmente.

OBS.: O comando **userdel** não permite exclusão de uma conta se o usuário estiver correntemente logado.

Exemplos:

userdel *seu_nome*
userdel -r *meu_nome*

12 - SISTEMA DE ARQUIVOS

Neste capítulo iremos tratar do sistema de arquivos utilizado pelo Linux e veremos também alguns conceitos necessários para entendermos melhor o sistema de arquivos. Veremos quais os diretórios básicos presentes na maioria dos sistemas Linux e as suas respectivas aplicações.

12.1 - CONCEITOS BÁSICOS

Antes de aprofundarmos sobre o sistema de arquivos do Linux, vamos esclarecer um pouco mais sobre partições. O Linux é um sistema operacional que suporta partições, ou seja, partes de um HD que são tratadas pela máquina como unidades de disco independentes. As partições podem ser criadas segundo diversos padrões, conforme o sistema operacional utilizado, e cada um deles utiliza um tipo de sistema de arquivo. Temos, portanto, um grande número de tipos de partições diferentes, sendo que dentre elas, podemos citar: FAT16 (DOS e WIN95), FAT32 (WIN98), NTFS (WIN NT), HPFS (OS/2).

12.1.1 - O que é?

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

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

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

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

minix
O mais antigo e presumivelmente o mais confiável, mas bastante limitado em características (algumas datas não aparecem, máximo de 30 caracteres para nome de

arquivos, etc...) e restrito em armazenamento (no máximo 64 Mb por sistema de arquivos).

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

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

ext3

Sistema de arquivos *ext3* faz parte da nova geração extended file system do Linux, sendo que seu maior benefício é o suporte a journaling.

O uso deste sistema de arquivos comparado ao *ext2*, na maioria dos casos, melhora o desempenho do sistema de arquivos através da gravação seqüencial dos dados na área de metadados e acesso mhash a sua árvore de diretórios.

A estrutura da partição ext3 é semelhante a ext2, o journaling é feito em um arquivo chamado .journal que fica oculto pelo código *ext3* na partição (desta forma ele não poderá ser apagado, comprometendo o funcionamento do sistema). A estrutura idêntica da partição *ext3* com a *ext2* torna mais fácil a manutenção do sistema, já que todas as ferramentas para recuperação *ext2* funcionarão sem problemas.

Tipos de Journaling no ext3

O ext3 suporta três diferentes modos de trabalho do Journaling. São eles:

Journal: grava todas as mudanças em sistema de arquivos. É o mais lento dos três modos, mas é o que possui maior capacidade de evitar perda de dados;

Ordered: grava somente mudanças em arquivos metadata (arquivos que guardam informações sobre outros arquivos), mas guardas as atualizações no arquivo de dados antes de fazer as mudanças associadas ao sistema de arquivos. Este Journaling é o padrão nos sistemas de arquivos ext3;

Writeback: também só grava mudanças para o sistema de arquivo em metadata, mas utiliza o processo de escrita do sistema de arquivos em uso para gravação. É o mais rápido Journaling ext3, mas o menos confiável.

O modo Ordered é o padrão no ext3, mas é possível especificar qual o modo que você deseja usar, através da atualização do arquivo fstab. Por exemplo, pode ser que a linha /dev/hda1/opt tenha sua opção *data* com o valor ordered. Você pode mudar este valor para writeback ou journal.

msdos
Compatibilidade com MS-DOS (e OS/2 e Windows NT) através de sistemas de arquivos FAT/FAT32.

umsdos
Sistemas de arquivos MS-DOS estendidos para suportar nomes longos, donos, permissões, links e arquivos de dispositivos do Linux. Isso permite que um sistema de arquivos msdos possa ser usado como se fosse um sistema Linux, removendo a necessidade de uma partição distinta para o Linux.

iso9660
O sistema de arquivos padrão do CD-ROM. A extensão Rock Ridge que permite nomes

longos também é suportada automaticamente.

nfs
Sistemas de arquivos em redes que permitem o compartilhamento e o fácil acesso aos arquivos entre diversos computadores da rede.

hpfs
O sistema de arquivos do OS/2.

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

12.1.2 - Ponto de Montagem

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

Como veremos mais à frente, o comando para montagem de um sistema de arquivos possui dois argumentos. O primeiro é o arquivo de dispositivo correspondente ao disco ou partição que contenha o sistema de arquivos. O segundo é o diretório sob o qual ele será montado. Após a execução do comando, dizemos então que ``/dev/hda2 está montado no /home'', por exemplo.

Para examinar estes sistemas de arquivos, pode-se acessar estes diretórios exatamente da mesma forma que qualquer outro, como veremos mais adiante. É importante ressaltar a diferença entre o **dispositivo** /dev/hda2 e o **diretório montado** /home. Enquanto o primeiro dá acesso ao dados brutos do disco, o segundo permite o acesso aos arquivos contidos no mesmo disco. O diretório montado é chamado **ponto de montagem**.

12.2 - DIRETÓRIO RAIZ (/)

O primeiro sistema de arquivos (chamado raiz, por conter o diretório raiz (/)) é montado, não a partir de outros sistemas de arquivos, mas sim, de maneira automática durante a inicialização do sistema operacional, podendo-se estar certo de que ele sempre estará disponível, pois de outra forma o sistema não poderá ser inicializado.

A composição do diretório raiz de um sistema Linux típico pode ser representado pela tabela abaixo:

DIRETÓRIO	***CONTEÚDO***
boot	Arquivos estáticos de boot de inicialização (boot-loader).
bin	Arquivos executáveis (binários) de comandos essenciais pertencentes ao sistema e que são usados com frequência.
sbin	Arquivos de sistema essenciais.
root home	Diretório local do superusuário (root). Diretórios locais (home) dos usuários.

DIRETÓRIO	***CONTEÚDO***
usr	Todos os arquivos de usuários devem estar aqui (segunda maior hierarquia).
proc	Não existe no HD, apenas no kernel. Permite acessar informações sobre a máquina e sobre os processos.
dev	Arquivos de dispositivos de entrada/saída (I/O).
etc	Arquivos de configuração do sistema da máquina local com arquivos diversos para a administração do sistema.
lib	Arquivos das bibliotecas compartilhadas usados com frequência.
tmp	Arquivos temporários gerados por alguns utilitários.
mnt	Ponto de montagem de partição temporária.
var	Informação variável.

Cada diretório listado será discutido em detalhes mais adiante.

O kernel do Linux normalmente está localizado na raiz / ou no /boot. Se estiver localizado em / é recomendado usar o nome vmlinux ou vmlinuz, sendo este último o nome que deverá ser usado em programas fonte do kernel do Linux atualmente.

12.3 - DIRETÓRIO /boot

Este diretório contém tudo que é necessário para carregar o sistema, exceto os arquivos de configuração e o gerenciador de boot.

O /boot é utilizado para qualquer coisa antes do kernel executar /sbin/init. Isto inclui setores master de inicialização (master boot sectors) guardados, arquivos de mapa de setor e qualquer outra coisa que não é editada manualmente.

Como exposto acima, o kernel do Linux pode estar localizado em / ou /boot. Se estiver em /boot, é recomendado usar um nome mais descritivo.

12.4 - DIRETÓRIO /bin

Contém os comandos que podem ser utilizados por todos os usuários e pelo administrador do sistema, os quais são requeridos no modo mono-usuário (single-user mode).

Pode também conter comandos que são utilizados indiretamente por alguns scripts.

Não é recomendado abrir subdiretórios dentro do /bin.

Os arquivos dos comandos que não são suficientemente essenciais para estar em /bin estarão localizados em /usr/bin. Os elementos que são utilizados pelos usuários isoladamente (porém não pelo root, como mail, chsh, etc) não são suficientemente essenciais para estar dentro da partição /bin.

Os seguintes comandos essenciais devem estar em /bin: arch, cat, chgrp, chmod, chown, cp, date, dd, df, dmeg, echo, ed, false, kill, in, login, mkdir, mknod, more, mount, mv, ps, pwd, rm, rmdir, sed, setserial, sh, sfte , seu, sinc, true, umount, uname.

12.5 - DIRETÓRIO /sbin

Tipicamente, /sbin contém arquivos essenciais para dar boot ao sistema, além dos arquivos em /bin. Qualquer coisa que se executar após /usr ter sido montado (quando não há problemas) deveria estar em /usr/sbin. Os arquivos de administração do sistema root local devem estar em /usr/local/sbin.

Decidir que arquivos vão no diretório /sbin é difícil. Se o usuário necessitar executar tais arquivos, estes deverão ir para outro diretório. Se somente o administrador do sistema ou o root necessitarem executar, tais como scripts da administração, então deverão ir para /sbin (não /usr/sbin ou /usr/local/sbin, pois o arquivo não é vital para a operação do sistema).

Arquivos e/ou comandos armazenados em /sbin são dos seguintes tipos :

- Comandos gerais (clock, getty, init, update, mkswap, swapon, swapoff, telinit);
- Comandos de saída(fastboot, fasthalt, halt, reboot, shutdown);
- Comandos de manipular sistema de arquivos (fdisk, fsck, fsck.*, mkfs, mkfs.*, onde * = é um dos seguinte: ext, ext2 minix, msdos, xia, e talvez outros, badblocks, dumpe2fs, e2fsck, mke2fs, mkost+found, tune2fs);
- O gerenciador de boot de inicialização (lilo);
- Comandos de rede (arp, ifconfig, route).

Além destes, existem outros arquivos que são opcionais em /sbin: sln, nc estático e ldconfig.

12.6 - DIRETÓRIO /root

Este diretório é opcional no Linux. Nos sistemas UNIX o diretório local do usuário root é tradicionalmente o diretório /. Em muitos sistemas Linux e em alguns sistemas UNIX utiliza-se o diretório /root.

O diretório local da conta do usuário root pode ser determinada por preferências locais. As possibilidades óbvias incluem /, /root, e /home/root. Se o diretório local do root não está armazenado na partição raiz, será necessário assegurar-se que tome / por default caso não seja localizado.

Não é recomendado o uso da conta root para coisas corriqueiras tal como ler o e-mail e ver as notícias (mail & news), recomenda-se que seja usada somente para a administração do sistema. Por esta razão recomendamos que não apareçam subdiretórios como Mail e News no diretório local da conta do usuário root. É recomendado que o mail para root seja redirecionado a um usuário mais adequado.

12.7 - DIRETÓRIO /home

O diretório /home é claramente um sistema de arquivos específico do diretório local. A regra de criá-lo difere de máquina para máquina. Descreve uma localização sugerida para os diretórios local dos usuários, assim, recomendamos que todas as distribuições LINUX usem este lugar como localização default dos diretórios locais.

Em sistemas pequenos, cada diretório de usuário é um dos subdiretórios debaixo do /home, como por exemplo: /home/dirson, /home/raulison, /home/weslei, etc.

Em sistemas maiores (especialmente quando os diretórios /home são compartilhados entre várias máquinas via rede) é útil subdividir os diretórios locais. A subdivisão pode ser implementada utilizando subdiretórios tais como /home/apoio, /home/docs, /home/cartas, etc.

12.8 - DIRETÓRIO /usr

O diretório /usr é a segunda maior seção do sistema de arquivos. /usr é informação compartilhada, somente de leitura, isto significa que /usr deve ser compartilhada entre várias máquinas que utilizam o Linux e não deve exibir qualquer informação local de uma máquina ou que varia com o tempo.

Nenhum pacote grande (como TeX ou GNUEmacs) deve utilizar o subdiretório direto abaixo de /usr, em vez disso, deve haver um subdiretório dentro de /usr/lib (o /usr/local/lib caso tenha sido instalado localmente). A propósito, para o sistema X Window faz-se uma exceção permitindo um considerável precedente, sendo esta prática amplamente aceita.

Alguns links simbólicos a diretórios podem estar presentes. Esta possibilidade baseia-se na necessidade de preservar a compatibilidade com sistemas anteriores, haja visto que em todas as implementações podemos assumir o uso da hierarquia /var.

Poderão existir os seguintes links :

/usr/adm	---------->	/var/adm
/usr/spool	---------->	/var/spool
/usr/tmp	---------->	/var/tmp
/var/spool/locks	---------->	/var/lock

Deve existir também um link /usr/X11 apontando para a hierarquia do sistema X Window atual.

Veremos a seguir qual o conteúdo de cada um dos diretórios contidos em /usr.

Podem-se usar subdiretórios como desejar.

/usr/include deve conter links aos diretórios /usr/src/linux/include/asm-<arch> e /usr/src/linux/include/linux, cujos dados são necessários ao compilador C. Pelo menos estes arquivos include devem sempre ser distribuídos nas instalações que incluem um compilador C. Devem ser distribuídos no diretório /usr/src/linux de forma a não haver problemas quando os administradores do sistema atualizarem sua versão do kernel pela primeira vez.

/usr/src/linux pode também ser um link simbólico a uma árvore de código fonte do kernel.

12.9 - DIRETÓRIO /proc

O sistema de arquivos /proc é um sistema de arquivos virtual utilizado para manipular informação de processos e de sistema. É recomendada sua utilização para o armazenamento e obtenção de informação de processos, assim como outras informação do kernel ou da memória.

12.10 - DIRETÓRIO /dev

Este é o diretório dos dispositivos de entrada/saída. Contém um arquivo para cada dispositivo que o kernel do Linux pode suportar.

Também contém um script chamado MAKEDEV, o qual pode criar dispositivos quando necessário. Pode conter um MAKEDEV local para dispositivos locais.

Os links simbólicos não devem ser distribuídos nos sistemas Linux, somente como previsto na lista de dispositivos de Linux. Isto é porque as instalações locais diferem daquelas da máquina do administrador. Além disso, se um script de instalação configurar links simbólicos na instalação, estes links seguramente não se atualizarão se houverem trocas locais no hardware. Quando utilizados responsavelmente são de bom uso.

12.11 - DIRETÓRIO /etc

Contém arquivos e diretórios de configuração do sistema da máquina local com arquivos diversos para a administração de sistema.

Tipicamente /etc possui dois subdiretórios:

X11	Arquivos de configuração para o X11
skel	Esqueletos da configuração de usuários

O /etc/skel é a localização para os chamados arquivos esqueletos de usuários, que são os dados por default quando um novo usuário recebe uma conta. Este diretório pode conter subdiretórios para diferentes grupos de usuários (/etc/skel/apoio, /etc/skel/usuários).

O /etc/X11 é o lugar recomendado para todos os arquivos de configuração X11 locais da máquina. Este diretório é necessário para permitir o controle local se /usr for colocado somente para leitura. Os arquivos que devem ir neste diretório incluem Xconfig (e/ou XF86Config) e Xmodmap. O /etc/X11/xdm contém os arquivos de configuração xdm. Estes são a maioria dos arquivos normalmente gravados em /usr/lib/X11/xdm.

Esta descrição do conteúdo é genérica, portanto não está totalmente completa, pode haver algumas variações dependendo do distribuidor do Linux ou do administrador de sistema. Os arquivos /etc são composto de:

Arquivos gerais, necessários na maioria dos sistemas LINUX, tais como: adjtime, fstab, group, inittab, issue, ld.so.conf, lilo.conf, motd, mtab, mtools, passwd, profile, shells, termcap.

Arquivos de rede: exports, ftpusers, gateway, hosts, host.conf, host.equiv, host.lpd, inetd.conf, networks, printcap, protocols, resolv.conf, rpc, service.

Os sistemas com senhas sombreadas (shadow password) terão arquivos de configuração adicionais, em /etc (/etc/shadow e outros) e /usr/bin (useradd, usermod, e outros).

12.12 - DIRETÓRIO /lib

O diretório /lib contém aquelas bibliotecas compartilhadas que são necessárias para carregar o sistema e executar os comandos do sistema de arquivos raiz, além de módulos essenciais do kernel.

As bibliotecas que são necessárias somente pelos arquivos /usr (como qualquer arquivo X Window) não pertencem a /lib. Só as bibliotecas compartilhadas requeridas para executar os arquivos dentro de /bin e /sbin devem estar ali. A biblioteca libm.so.* poderia

estar localizada em /usr/lib se não é utilizada de nenhuma forma em /bin ou /sbin.

Por razão de compatibilidade, /lib/cpp necessita existir como uma referência ao pre-processador C instalado no sistema. A localização usual do arquivo é /usr/lib/gcc-lib/<target>/<versão>/cpp. Podem existir links /lib/cpp apontando para estes arquivos ou a qualquer outra referência a ele que exista no sistema de arquivos. (Por exemplo, /usr/bin/cpp é utilizado frequentemente). A especificação para /lib/module ainda não foi definida, pois ainda não há um consenso na comunidade Linux.

12.13 - DIRETÓRIO /tmp

O /tmp é utilizado para arquivos temporários gerados por alguns arquivos utilitários, preferencialmente em dispositivo rápido (um sistema de arquivos baseado em memória, por exemplo).

A "permanência" da informação que é armazenada em /tmp é diferente daquela que é armazenada em /var/tmp. /tmp pode ser limpo em cada inicialização ou a intervalos relativamente frequentes. Portanto, não se deve operar a informação armazenada em /tmp permanecendo por algum período determinado de tempo.

Os programas devem utilizar /tmp ou /var/tmp (que era originalmente /usr/tmp) de acordo os requisitos esperados da informação, pois não devem colocar nenhum arquivo particular em qualquer diretório de armazenamento temporário.

Os administradores de sistemas podem querer ajuntar /tmp a algum outro diretório, tal como /var/tmp. Isto é útil, por exemplo, para conservar espaço na partição raiz. Se isto for feito, então a permanência de arquivos em /var/tmp deve ser mesmo tão grande como a de /tmp.

O subdiretório /tmp pode estar na memória RAM.

12.14 - DIRETÓRIO /mnt

Este diretório foi previsto para o administrador poder montar temporariamente sistemas de arquivos quando necessitar. O conteúdo deste diretório é um assunto local e não deve afetar a maneira que executamos nenhum programa. É recomendável a não utilização deste diretório para programas de instalação.

12.15 - DIRETÓRIO /var

O diretório /var contém arquivos com informação variável. Inclui arquivos e diretórios em fila de execução, informação de ordem administrativa e arquivos temporários e transitórios.

/var é especificada aqui para fazer possível montar /usr somente para leitura. Tudo aquilo que alguma vez ficou em /usr e é escrito durante a operação normal do sistema (não durante a instalação e manutenção do software) deve ir em /var.

13 - ACESSANDO HD, CDROM E DISQUETE

Neste capítulo trataremos da forma como o Linux faz acesso aos dados contidos em Hds, CDROMs e disquetes.

13.1 - Nomes dos Dispositivos

O Linux possui uma identificação bastante diferente do DOS / Windows quanto às unidades de disquete, Hds e CDROMs, bem como aos demais dispositivos.

A seguir, veremos como são identificados alguns dos dispositivos mais comuns no Linux:

Descrição	***Identificação***
Winchesters (HDs) e CDROMs (IDE)	/dev/hda, /dev/hdb, ... ou /dev/ide0/..., /dev/ide1/...
Floppy drives	/dev/fd0H1440, /dev/fd0h1200, /dev/fd1H1440, ...
Dispositivos SCSI	/dev/sda, /dev/sdb, /dev/sdc. ...
Impressoras paralelas Portas seriais (COM 1, COM 2, etc)	/dev/lp0, /dev/lp1, /dev/lp2, ... /dev/cua0, /dev/cua, ...1 ou /dev/ttyS0, /dev/ttyS1, ...

OBS.:

1)No kernel 2.4 os nomes dos dispositivos foram alterados. Por exemplo: /dev/hda passou a ser /dev/ide0/...

2) Normalmente no diretório /dev existem diversos arquivos de dispositivos, não significando que todos estes dispositivos estejam instalados. Isto facilita a instalação de novos componentes de hardware no sistema, pois não há necessidade de encontrar os parâmetros corretos para o dispositivo sendo instalado.

13.2 - Montagem de Dispositivo

No Linux, os arquivos em qualquer dispositivo de armazenamento (disquete, CDROM ou HD), devem fazer parte do sistema de arquivos para poderem ser acessados. Ou seja, é diferente do DOS / Windows, que trata cada unidade (a:, c:, etc) como um sistema de arquivos isolado.

Para que os arquivos armazenados em um disquete, CDROM ou HD possam fazer parte do sistema de arquivos, é preciso realizar uma operação chamada "montagem". Basicamente a montagem consiste em informar ao sistema os parâmetros necessários para o acesso aos dados, como o ponto de montagem, o dispositivo utilizado, o tipo do sistema de arquivo, entre outros.

A operação de montagem pode ser realizada manualmente ou pode ser automatizada para que seja realizada em determinadas circunstâncias, especialmente na inicialização do sistema.

Da mesma forma, quando um sistema de arquivos em um dispositivo não é mais necessário, ele deve ser "desmontado" antes de ser fisicamente removido, de modo a não fazer mais parte do sistema de arquivos raiz (/).

O arquivo /etc/fstab contém informações que permitem sintetizar os comandos de montagem de dispositivos, já que as informações contidas nele não precisarão ser informadas ao sistema durante a operação de montagem dos dispositivos nele relacionados. Mais adiante veremos como isto funciona.

13.3 - Ponto de Montagem

Como vimos anteriormente, "ponto de montagem" é a denominação do local onde um sistema de arquivos está montado.

No Linux, o diretório destinado à montagem de sistemas de arquivos temporários é o /mnt. Porém nada impede que os mesmos sejam montados em outros diretórios, atentando apenas para o fato de estarmos saindo um pouco fora dos padrões adotados em consenso pela comunidade Linux.

Veremos a seguir os principais pontos de montagem normalmente utilizados nos diversos sistemas Linux:

Dispositivo	***Ponto de Montagem***
Floppy drive	/mnt/floppy
CDROM	/mnt/cdrom

Desta forma, se quisermos acessar o conteúdo de um disquete, devemos montá-lo e teremos acesso a seus dados através o diretório /mnt/floppy. Da mesma forma com os dados de um CDROM, porém neste caso o diretório que nos dará o acesso aos dados é /mnt/cdrom.

14 - COMANDOS BÁSICOS V

Neste capítulo veremos alguns comandos utilizados para montar e desmontar sistemas de arquivos a partir de um dispositivo, bem como para formatar disquetes e criar sistemas de arquivos.

14.1 - mount

Todos os arquivos acessíveis em um sistema Linux estão organizados em uma grande árvore, a hierarquia de arquivos, iniciada pelo raiz simbolizado como /. Estes arquivos podem estar distribuídos por diversos dispositivos. O comando mount destina-se a incluir o sistema de arquivos encontrado em algum dispositivo à grande árvore de arquivos raiz (/).

O formato padrão do comando mount é **mount** -t *tipo dispositivo dir.* Isso indica ao kernel para incluir o sistema de arquivos encontrado em *dispositivo* (o qual é do tipo *tipo*) nominando-o como diretório *dir*.

Sintaxe:

mount -a [-fnrvw] [-t tipo]

mount [-fnrvw] [-o opções [,...]] dispositivo | dir

mount [-fnrvw] [-t tipo] [-o opções] dispositivo dir

Opções disponíveis para o comando mount :

-v

Modo de mensagens ativas.

-a

Monta todos os sistemas de arquivos (ou aqueles com os tipos mencionados) descritos em fstab.

-n

Montagem sem gravação de /etc/mtab. Isso é necessário por exemplo quando o sistema de arquivos /etc está somente com permissões de leitura.
-f

Faz com que tudo seja executado exceto a montagem efetiva em si. Apesar de não ser tão óbvia, esta opção permite que falsas montagens sejam realizadas, e é útil quando em conjunto com -v permite determinar o que o comando mount está tentando fazer. Pode ainda ser usado para adicionar entradas para dispositivos que foram montados anteriormente com a opção -n.

-r

Monta o sistema de arquivos somente com permissões de leitura. Um sinônimo é -o ro.

-w

Monta o sistema de arquivos com permissões de leitura e gravação. Este é o padrão. É um sinônimo de -o rw.

-t tipo

O argumento seguinte a -t é usado para indicar o tipo do sistema de arquivo. Uma relação dos tipos suportados pelo Linux pode ser encontrada em linux/fs/filesystems.c , quais sejam: minix, ext, ext2, xiafs, hpfs, msdos, umsdos, vfat, proc, nfs, iso9660, smbfs, ncpfs, affs, ufs, romfs, sysv, xenix, coherent. Note que os últimos três são equivalentes e que xenix e coherent serão descontinuados em algum momento no futuro. Sugere-se o uso de sysv em seu lugar. Desde o kernel 2.1.21 os tipos ext e xiafs foram descontinuados.

O tipo iso9660 é o padrão. Se nenhuma opção -t for apresentada, ou se o tipo auto for especificado, o superbloco será testado para verificação do tipo do sistema de arquivos (minix, ext, ext2, xiafs, iso9660, romfs são suportados). Caso este teste falhe e /proc/filesystems exista, então todos os sistemas de arquivos listados serão testados, exceto aqueles que estejam marcados como "nodev" (por exemplo proc e nfs). Note que o tipo auto pode ser útil para unidades de disquetes montadas pelos usuários. Porém atente que o teste usa um método heurístico (a presença de um número mágico) e pode reconhecer de forma equivocada o tipo do sistema de arquivos).

Mais que um tipo pode ser especificado com uma vírgula como separador. A lista dos tipos de sistema de arquivos pode ser precedida pela palavra **no** para especificar tipos de sistemas que não devem ser utilizados nos testes. (Isso pode não ter sentido com a opção -a *option*.)

-o

Opções são especificadas com um indicador -o seguido por vírgula como separador. Algumas dessas opções são úteis somente quando aparecem no arquivo /etc/fstab. As opções a seguir aplicam-se a qualquer sistema de arquivos que esteja sendo montado:

auto Pode ser montado com a opção -a .

defaults Usa as opções padrão: rw, suid, dev, exec, auto, nouser e async.

dev Interpreta dispositivos especiais de blocos ou caracter no sistema de arquivos.

exec Permite a execução de binários.

noauto O arquivo somente pode ser montado explicitamente.
nodev Dispositivos especiais de blocos ou caracter não devem ser interpretados.

noexec Não permite a execução de qualquer binário no sistema de arquivos montado.

nosuid Não permite usar os bits de configuração de identificação de usuário ou grupo.

nouser Proíbe que um usuário comum monte o sistema de arquivos. Este é o padrão.

remount Tenta remontar um sistema de arquivos já montado.

ro Monta o sistema de arquivos somente para leitura.

rw Monta o sistema de arquivos com permissão de leitura e gravação.

suid Permite o uso dos bits de configuração de identificação do usuário e do grupo.

user Permite que um usuário normal possa montar o sistema de arquivos.

Além das citadas, existem várias opções especiais (que se seguem ao parâmetro -o) que se aplicam aos diversos tipos de sistemas de arquivos. Para mais informações, deve-se consultar man mount.

Na tabela abaixo estão os principais arquivos relacionados ao comando mount:

Arquivo	***Descrição***
/etc/fstab	Tabela de sistemas de arquivos
/etc/mtab /etc/mtab~	Tabela de sistemas de arquivos montados Arquivo de lock
/etc/mtab.tmp	Arquivo temporário

O arquivo /etc/fstab pode conter linhas descrevendo quais dispositivos são usualmente montados, e com quais opções. O arquivo é usado de três formas:

a) O comando mount -a [-t tipo] faz com que todos os sistemas de arquivos indicados em fstab (de tipo apropriado) sejam montados conforme indicado, exceto para aqueles cujas linhas contenham a palavra chave noauto.

b) Quando estiver montando um sistema de arquivos mencionado em fstab, é suficiente fornecer somente o dispositivo, ou somente o ponto de montagem.

c) Quando fstab contém a opção user na linha, qualquer usuário poderá montar este sistema.

Exemplos:

```
mount  /dev/fd0
mount  /mnt/floppy
mount  -a  -t  nomsdos,ext
```

14.2 - umount

O comando umount retira o sistema de arquivos indicado da hierarquia de arquivos. O sistema de arquivos a ser "desmontado" pode ser especificado tanto através da informação do diretório onde ele foi montado, quanto pelo nome do dispositivo onde ele

reside.

Devemos notar que um sistema de arquivos não pode ser desmontado quando ele está em uso - por exemplo quando há arquivos abertos ou quando alguns processos tenham seu diretório de trabalho nele, ou quando um arquivo de swap esteja em uso.

Sintaxes:

umount -a [-nrv] [-t tipo]

umount [-nrv] dispositivo | dir [...]

Principais opções:

-v Modo de apresentação de mensagens.

-n Desmontar sem escrever em /etc/mtab.

-r No caso da desmontagem falhar, tenta remontar somente para leitura.

-a Todos os sistemas de arquivos descritos em /etc/mtab são desmontados.

-t tipo Indica que as ações podem ser realizadas nos sistemas de arquivos do tipo especificado. Mais de um tipo pode ser especificado, separados por vírgulas. A lista de tipos de sistemas de arquivos pode ter um prefixo **no** para especificar os tipos de sistemas de arquivos nos quais as ações não podem ser exercidas.

Exemplos:

```
umount /dev/fd0
umount -v /mnt/floppy
umount -a -t noext2
```

14.3 - fdformat

O comando **fdformat** executa uma formatação de baixo nível em um disquete. O parâmetro *dispositivo* é normalmente um dos seguintes: /dev/fd0d360, /dev/fd0h1200, /dev/fd0D360, /dev/fd0H360, /dev/fd0D720, /dev/fd0H720, /dev/fd0h360, /dev/fd0h720, /dev/fd0H1440, /dev/fd1d360, /dev/fd1h1200, /dev/fd1D360, /dev/fd1H360, /dev/fd1D720, /dev/fd1H720, /dev/fd1h360, /dev/fd1h720 ou /dev/fd1H1440.

Os dispositivos de disquetes genéricos, /dev/fd0 e /dev/fd1, não funcionarão com fdformat quando um formato não padrão estiver sendo usado, ou o formato não seja auto detectado.

Sintaxe:

fdformat [-n] dispositivo

Opções:

-n Não verificar. Esta opção desabilitará a verificação que é realizada após a formatação.

14.4 - mkfs

O comando mkfs constrói um sistema de arquivos Linux em um dispositivo, geralmente uma partição de um disco rígido ou disquete. O parâmetro *sistema-arq* pode ser o nome do dispositivo (por exemplo /dev/hda1, /dev/fd0H1440) ou o ponto de montagem (por exemplo /, /usr, /home) para o sistema de arquivos. O parâmetro opcional [blocos] é a quantidade de blocos a ser utilizada pelo sistema de arquivos.

Os códigos de retorno do comando mkfs são: 0 em caso de sucesso e 1 em caso de erro.

Sintaxe:

mkfs [-V] [-t tipo] [opções] *sistema-arq* [blocos]

Principais Opções:

-V

Exibe informações detalhadas sobre os comandos executados, incluindo os comandos específicos de cada sistema de arquivos. Passando esta opção mais de uma vez inibe-se a execução de comandos específicos ao sistema de arquivos. Isso é útil durante a realização de testes.

-t tipo

Especifica o tipo de sistema de arquivos a ser criado. Se não especificado o tipo padrão de sistema de arquivo (atualmente ext2) é criado.

Opções

Opções específicas do sistema de arquivos a ser passado ao construtor de sistemas de arquivos. Embora não seja garantido, geralmente as opções seguintes são suportadas por muitos construtores de sistemas de arquivos.

-c

Checa o dispositivo por blocos defeituosos durante a criação do sistema de arquivos.

-l *nomearq*

Lê a lista de blocos defeituoso a partir de *nomearq.*

-v

Mostra o que está sendo feito (modo detalhado, do inglês "verbose").

Exemplos:

```
mkfs -t ext2 /dev/fd0H1440
mkfs -t msdos /dev/fd0H1440
```

15 - FORMAS DE ACESSO A DISQUETES

Neste capítulo veremos duas formas diferentes para acessarmos os dados contidos em disquetes, quando trabalhamos com dois ou mais sistemas de arquivos de tipos diferentes nestes disquetes.

15.1 - Usando Dois Pontos de Montagem

A primeira forma, consiste em usar mais de um ponto de montagem para o mesmo dispositivo, sendo que cada um dos pontos de montagem será utilizado para acessar um sistema de arquivos de tipo diferente.

Para realizarmos isto, devemos:

1. Criar os pontos de montagem, o que significa criar dois diretórios diferentes, através dos quais os dados serão acessados. Normalmente estes diretórios devem ser criados dentro de /mnt.

2. Editar o arquivo /etc/fstab e informar os dois pontos de montagem, correspondendo ao mesmo dispositivo, porém com tipos de sistemas de arquivos diferentes.

Exemplo:

```
cd  /mnt
mkdir  disk
mkdir  floppy
nano  /etc/fstab
```

Dentro de fstab:

```
/dev/fd0            /mnt/floppy          ext2    noauto     0 0
/dev/fd0            /mnt/disk             vfat    noauto     0 0
```

Desta forma, portanto, temos um mesmo dispositivo (/dev/fd0) que pode ser montado em dois pontos diferentes, sendo /mnt/floppy o ponto de montagem correspondente ao sistema de arquivos ext2 e /mnt/disk o correspondente a vfat.

Assim, ao executarmos o comando mount /mnt/floppy o sistema tentará encontrar no dispositivo /dev/fd0 um sistema de arquivos do tipo ext2. Ao executarmos mount /mnt/disk o sistema tentará encontrar no dispositivo /dev/fd0 um sistema de arquivos do tipo vfat.

É bom observarmos que neste caso não podemos usar o comando mount informando apenas o dispositivo (/dev/fd0), pois o sistema não saberá a qual dos dois pontos de montagem estamos nos referindo, especialmente porque cada um deles foi definido para corresponder a um determinado tipo de sistema de arquivos.

15.2 - Usando Apenas um Ponto de Montagem

Esta segunda forma de acesso a disquetes tende a ser mais simples que a primeira, pois usa apenas um ponto de montagem, com identificação automática do tipo de sistema de arquivo que está sendo utilizado.

Para isto, o arquivo /etc/fstab deve conter uma linha como abaixo, sendo obrigatório o uso do parâmetro *auto*:

```
/dev/fd0        /mnt/floppy    auto    noauto          0  0
```

Para montar o dispositivo, o comando será: mount /mnt/floppy

16 - ACESSANDO WINDOWS

É bastante freqüente termos na mesma máquina o Linux e o DOS ou Windows, normalmente cada um em uma partição ou mesmo em HDs separados. Portanto, é comum a necessidade de acessar pelo Linux os dados que estão na partição DOS / Windows.

Para possibilitarmos isto é muito simples, basta criar o ponto de montagem e em seguida montar o dispositivo correspondente à partição onde o acesso é desejado.

Exemplo:

```
mkdir  /mnt/win
mount  -t  vfat  /dev/hda1  /mnt/win
```

Se quisermos facilitar ainda mais, devemos editar o arquivo /etc/fstab e adicionar uma linha semelhante à descrita abaixo, para informarmos ao sistema os parâmetros que devem ser utilizados pelo comando mount:

```
/dev/hda1          /mnt/win          vfat    noauto,user     0 0
```

Note que as opções *noauto,user* indicam que este sistema de arquivos não é montado automaticamente (*noauto*) e que ele pode ser montado pelos usuários (*user*) e não apenas pelo root. Se desejar que este sistema de arquivos seja montado automaticamente, substitua as opções *noauto,user* por *defaults*. Para saber mais sobre o arquivo fstab e as opções que podem ser utilizadas, utilize o comando man fstab.

Utilizando a linha descrita acima no arquivo /etc/fstab, para montarmos o sistema de arquivos basta informarmos o dispositivo ou o ponto de montagem:

```
mount  /dev/hda1          ou          mount  /mnt/win
```

Se utilizarmos a opção *defaults* em lugar de *noauto,user* o sistema de arquivos será montado automaticamente durante a inicialização do sistema e poderemos acessar seus dados simplesmente adentrando no ponto de montagem.

17 - INSTALAÇÃO DE PROGRAMAS

Existem diversas maneiras de se instalar programas no Linux. Podemos dizer basicamente que elas se dividem em duas formas: baseadas em código fonte e baseadas em pacotes.

A primeira delas, mais tradicional e ausente na maior parte dos demais sistemas operacionais, é aquela onde se dispõe do código fonte do programa e a instalação consiste em compilar este código, gerando o código objeto (também chamado de binário ou executável). Muitas vezes, junto com o código fonte é fornecido um script (semelhante a um arquivo de lote .bat do DOS) para facilitar a compilação do programa. Geralmente, isto é feito utilizando-se a seguinte sequência de comandos:

```
./configure
make
make install
```

A segunda forma é baseada em pacotes, ou seja, o programa já compilado (código objeto ou executável) é "empacotado", junto com os demais arquivos que sejam necessários para seu funcionamento. Existem diversos tipos de pacotes, como rpm (usado pela Red Hat e derivados), deb (Debian e derivados) e tgz (Slackware).

Aqui vale uma observação: existe uma certa confusão entre os formatos tgz e tar.gz. O primeiro é o formato dos pacotes de programas utilizado pelo Slackware, enquanto o segundo são apenas arquivos agrupados pelo comando *tar* e compactados com *gzip*.

Praticamente todas as principais distribuições de Linux utilizam algum programa gerenciador de pacotes, existindo também programas para conversão de um tipo de pacote em outro. Exemplos de gerenciadores de pacotes: RPMDrake, GnoRPM, pkgtool, etc.
Exemplos de conversores de pacotes: alien, rpmtotgz, etc.

18 - ARQUIVAMENTO

A cópia de segurança de arquivos (backup) é uma tarefa muito importante em redes e também nas aplicações onde máquinas individuais necessitam preservar informações (dados) de valor.

O processo de backup mais comum é o que utiliza cópias simples, a qual consiste em copiar tudo uma única vez e nas próximas vezes, copiar apenas os arquivos que foram alterados após a cópia inicial. A primeira cópia é chamada cópia total ou completa, enquanto as seguintes são chamadas cópias incrementais.

Além deste, existe o processo de backup que utiliza cópias em diversos níveis (multinível), sendo mais adequado para tarefas mais complexas.

O que deve ser arquivado? Esta pergunta deve ser respondida pelo usuário da máquina ou pelo administrador da rede, baseado em seu conhecimento sobre o conteúdo armazenado em cada máquina. Porém, é bastante claro que devem ser copiados os arquivos de usuários (normalmente em /home) e os arquivos de configuração do sistema (normalmente em /etc, podendo haver outros arquivos espalhados por todo o sistema de arquivos).

O Linux dispõe de alguns comandos que podem ser utilizados para gerar e restaurar cópias de arquivos (backup), sendo os mais comuns o *tar*, *cpio* e *dump*. Eles permitem criar arquivos de backup em fitas, disquetes, discos rígidos e outros formatos de mídia.

Uma forma de diminuir o espaço gasto na mídia com os arquivos de backup é a compactação dos mesmos. Porém a compactação do backup, além de tornar o processo mais lento, pode contribuir para que todo o arquivo seja inutilizado, caso haja ocorrência de um grande número de erros no backup compactado.

A restauração da cópia de segurança pode ser feita de maneira completa (restaura todos os arquivos do backup) ou parcial, bastando especificar os nomes dos arquivos desejados ao executar o comando para restauração.

19 - COMANDOS BÁSICOS VI

Neste capítulo veremos alguns comandos utilizados para a instalação, atualização, desinstalação e consulta de pacotes e também o comando utilizado para backup do sistema.

19.1 - rpm

O rpm é um poderoso gerenciador de pacotes, que pode ser utilizado para instalar, consultar, atualizar e desinstalar pacotes de software. Além destas funções, que serão

abordadas aqui, o rpm também pode fazer verificação, validação de assinatura, construção, reconstrução do banco de dados, ajustar permissões, ajustar donos e grupos e exibir configuração. Cada um destes modos básicos de operação aceita um conjunto diferente de opções.

19.1.1 - Consulta:

A sintaxe geral para o modo de consulta é:

rpm -q [opções-de-consulta]

Há dois subconjuntos de opções de consulta: seleção de pacotes e seleção de informações.

Opções de seleção de pacotes:

<nome_do_pacote> Consulta o pacote instalado de nome <nome_do_pacote>.

-a Consulta todos os pacotes instalados.

-f <arquivo> Consulta o pacote do qual <arquivo> faz parte.

-p <arquivo_pacote> Consulta um arquivo de pacote (desinstalado) de nome <arquivo_pacote>.

Opções de seleção de informações:

-i Exibe informações sobre o pacote, incluindo nome, versão e descrição.

-R Lista os pacotes dos quais este depende (o mesmo que --requires).

--provides Lista as capacidades que este pacote fornece.

--changelog Exibe informações sobre as mudanças neste pacote.

-l Lista os arquivos contidos no pacote.

-s Exibe os estados dos arquivos no pacote (implica -l). O estado de cada arquivo é normal, não instalado (uninstalled), ou substituído (replaced).

-d Lista apenas os arquivos de documentação (implica -l).

-c Lista apenas os arquivos de configuração (implica -l).

--scripts Lista os scripts de shell do pacote, que são usados no processo de instalação e desinstalação, se existirem.

Exemplos:

```
rpm -qa | less
rpm -qa | grep less
rpm -qf /usr/bin/less
rpm -qi less-332-9cl
rpm -q less-332-9cl -R
rpm -q glibc-2.1.1-11cl --provides | more
rpm -q less-332-9cl -l
rpm -q less-332-9cl -ld
```

rpm -q less-332-9cl -lc

19.1.2 - Instalação:

A sintaxe geral para o modo de instalação é:

rpm -i [opções-de-instalação] <arquivo_pacote>

Opções de instalação:

--force O mesmo que usar --replacepkgs, --replacefiles e --oldpackage.

-h, --hash Exibe 50 caracteres # (hash) à medida que o arquivo é desempacotado. Usar em conjunto com -v para uma exibição interessante.

--oldpackage Permite que uma atualização substitua um pacote por uma versão anterior.

--percent Exibe porcentagens à medida que os arquivos são desempacotados.

--replacefiles Instala os pacotes mesmo que eles substituam arquivos de outros pacotes, já instalados.

--replacepkgs Instala os pacotes mesmo que alguns deles já estejam instalados no sistema.

--allfiles Instala ou atualiza todos os arquivos do pacote que estão faltando, independente deles existirem ou não.

--nodeps Não verifica as dependências antes de instalar ou atualizar um pacote.

--noscripts Não executa os scripts de pré ou pós instalação.

--excludedocs Não instala nenhum arquivo marcado como documentação (o que inclui as páginas de manual e documentos texinfo).

--test Não instala o pacote, apenas verifica e avisa sobre possíveis conflitos.

Exemplos:

rpm -ivh --test tree-1.2-8cl.rpm
rpm -i --percent tree-1.2-8cl.rpm
rpm -ivh nt-1.06-1.rpm
rpm -ivh xcircuit-2.0a11-1.i386.rpm

19.1.3 - Atualização:

Quando usada a sintaxe de atualização do rpm, ele instala ou atualiza o pacote atualmente instalado para a versão do novo RPM. Isso é o mesmo que instalar, exceto que todas as versões anteriores dos pacotes serão removidas do sistema após a atualização.

A sintaxe geral para o modo de atualização é:

rpm -U [opções-de-instalação] <arquivo_pacote>

As opções de instalação são as mesmas descritas acima, no item instalação.

Exemplos:

rpm -Uvh man-1.5g-7cl.i386.rpm

19.1.4 - Desinstalação:

A sintaxe geral utilizada para desinstalação é:

rpm -e <nome_do_pacote>

Opções de desinstalação:

--allmatches Remove todas as versões do pacote que casarem com <nome_do_pacote>. Normalmente um erro é exibido se <nome_do_pacote> casar com múltiplos pacotes.

--noscripts Não executa os scripts de pré e pós desinstalação.

--nodeps Não verifica se dependências serão quebradas antes de desinstalar o pacote.

--test Não desinstala nada, apenas simula todos os movimentos.

Exemplos:

rpm -evv tree-1.2-8cl --test
rpm -evv tree-1.2-8cl
rpm -e xcircuit-2.0a11-1.i386.rpm

19.2 - tar

O **tar** é um programa de arquivamento desenvolvido para armazenar e extrair arquivos de um arquivo **tar** (que contém os demais) conhecido como tarfile. O tarfile pode ser construído em uma fita magnética, ou também, o que é comum, gravar-se um tarfile em um arquivo normal.

O primeiro argumento para **tar** deve ser uma das seguintes opções: Acdrtux, seguido por uma das seguintes funções adicionais.

Os argumento finais do **tar** são os nomes dos arquivos ou diretórios nos quais eles podem ser arquivados. O uso de um nome de diretório implica sempre que os subdiretórios sob ele serão incluídos no arquivo.

Sintaxe:

tar [opções] arquivo1 [arquivo2, ... arquivoN] diretório1 [diretório2, ...diretórioN]

Uma das seguintes opções deve ser usada:

-A Anexar os arquivos tar a um arquivo

-c Criar um novo arquivo tar

-d Encontrar as diferenças entre um arquivo tar e um sistema de arquivos

--delete Apagar do arquivo tar (não pode ser usado para fitas magnéticas!)

-r Anexar arquivos ao final do arquivo tar

-t Lista o conteúdo de um arquivo tar

-u Somente anexa arquivos mais novos que a cópia presente no arquivo tar

-x Extrai arquivos de um arquivo tar

Opções adicionais:

--atime-preserve Não altera a data de acesso dos arquivos copiados

-b, --block-size N Tamanho do bloco Nx512 bytes (padrão N=20)

-B, --read-full-blocks Redefine o tamanho do bloco enquanto lê (para leitura de pipes 4.2.BSD)

-C, --directory DIR Mudar para o diretório DIR

--checkpoint Imprimir os nomes dos diretórios enquanto lê o arquivo tar

-f, --file [HOSTNAME:]F Usar o arquivo file ou o dispositivo F (padrão /dev/rmt0)

--force-local Arquivo tar será local mesmo que tenhas vírgulas

-G, --incremental Cria/lista/extrai no formato GNU antigo de cópia de segurança incremental.

-g, --listed-incremental F Cria/lista/extrai no formato GNU novo de cópia de segurança incremental.

-h, --dereference Não copia ligações simbólicas, mas sim os arquivos que elas apontam.

-i, --ignore-zeros Ignorar blocos com zeros no arquivo tar (normalmente significam fim de arquivo)

--ignore-failed-read Não finalizar com status diferente de zeros quando houver arquivos que não possam ser lidos

-k, --keep-old-files Mantém os arquivos existentes, não regravando a partir do arquivo tar

-K, --starting-file F Começa no arquivo F do arquivo tar

-l, --one-file-system Manter-se no sistema de arquivos local ao criar um arquivo tar

-L, --tape-length N Muda a fita após gravar N*1024 bytes

-m, --modification-time Não extrair a data de modificação dos arquivos

-M, --multi-volume Cria / lista / extrai arquivos multivolumes

-N, --after-date DATA, --newer DATA Somente armazena arquivos mais recentes que DATA

-o, --old-archive, --portability Grava o arquivo no formato V7, ao invés do

formato ANSI

-O, --to-stdout	Extrai arquivos para a saída padrão
-p, --same-permissions, --preserve-permissions	Extrai todas as informações de proteção
-P, --absolute-paths	Não retirar os caracteres `/' do início do nome dos arquivos
--preserve	Similar -p -s
-R, --record-number	Mostra o número do registro dentro do arquivo tar em cada mensagem
--remove-files	Remover os arquivos após adicioná-los ao arquivo tar
-s, --same-order, --preserve-order	Ordena a lista de nomes a serem extraídos para comparar com o arquivo tar
--same-owner	Cria os arquivos extraídos com a mesma propriedade.
-S, --sparse	Manuseia arquivos segmentados eficientemente
-T, --files-from F	Obtém os nomes a serem extraídos ou criados no arquivo F
--null	-T lê nomes terminados com caracter nulo, desabilita -C
--totals	Lista o total de bytes gravados com --create
-v, --verbose	Mostra a lista dos arquivos processados
-V, --label NOME	Cria um arquivo tar com o nome de volume igual a NOME
-w, --interactive, --confirmation	Solicita uma confirmação para cada ação
-W, --verify	Tenta verificar um arquivo após gravá-lo
--exclude ARQUIVO	Excluir arquivo ARQUIVO
-X, --exclude-from ARQUIVO	Excluir arquivos listados em ARQUIVO
-Z, --compress, --uncompress	Filtra o arquivo através de compactação
-z, --gzip, --ungzip	Filtra os arquivos através de gzip
--use-compress-program PROG	Filtra os arquivos através de PROG (o qual deve aceitar -d)
--block-compress	Bloco de saída de programas de compactação de fitas magnéticas
-[0-7][lmh]	Especifica o dispositivo e a densidade.

Exemplos:

```
tar  -cf  arquivo.tar  /home
tar  -czf arquivo.tar.gz  /home
tar  -tf  arquivo.tar
tar  -tzf  arquivo.tar.gz
```

tar xvzf xcircuit-2.0b1.tar.gz
tar xvf arquivo.tar --directory backup arquivo_a_extrair

20 - COMANDOS BÁSICOS VII

Neste capítulo veremos dois comandos utilizados para gerenciamento de processos no Linux, o ps e o kill.

20.1 - ps

O comando **ps** relata a situação dos processos, fornecendo uma imagem dos processos atuais. Caso se deseje uma atualização repetitiva da situação, deve-se usar o comando top.

Sintaxe:

ps [-] [lujsvmaxScewhrnu] [txx] [O[+|-]k1[[+|-]k2...]] [pids]

Descrição das opções longas no estilo GNU:

Opção	***Descrição***
l	Formato longo
u	Formato de usuário: fornece o nome do usuário e o horário de início do processo
j	Formato de processos: pgid sid
s v	Formato de sinal Formato vm
m	Lista informações de memória (combinado com o indicador p pode informar o número de páginas).
f	Formato de árvores da família na linha de comando
a	Mostra também processos de outros usuários
x	Mostra processos sem controle de terminal
S	Adiciona o tempo de processamento e falhas nas páginas dos processos filhos.
c	Nome do comando a partir de task_struct
e	Mostra o ambiente após a linha de comando e ` + '
w	Saída larga: não trunca as linhas de comandos para que caibam em uma única linha. Para ser exato, cada w que é especificado irá adicionar outra linha possível na saída. Caso o espaço não seja necessário, ele não será utilizado. Pode-se ter até 100 w's.
h	Sem cabeçalhos
s	Somente os processos em execução
n	Saída em formato numérico para USER e WCHAN.
txx	Somente processos controlando terminais tty xx; para xx pode-se usar ou o nome do dispositivo em "/dev" ou o nome com o qual tty é dividido. Este é o reverso heurístico do processo usado por ps para listar o nome abreviado no campo TT, e.g. ps -t1.

Opção	***Descrição***
O[+|-]k1[,[+|-]k2[,...]]	Ordena a lista de processos de acordo com a ordem multinível especificada na seqüência de chaves curtas em SORT KEYS, k1, k2,... Especificações padrão de ordem existem para cada um dos vários formatos de ps. Eles podem ser sobrepostos por uma ordem especificada pelo usuário. O sinal `+' é opcional, e significa somente a direção padrão de uma chave. O sinal `-' reverte a direção da chave que seja precedida por ele. Assim como t e pids, a opção O deve ser a última

Opção	*Descrição*
	opção em um argumento simples de comando, mas especificações em argumentos sucessivos podem ser concatenados.
pids	Lista somente os processos especificados; separados por vírgulas. A lista deve ser informada imediatamente após a última opção em um argumento de linha de comando, sem espaços, como por exemplo ps -j1,4,5. Listas especificadas em argumentos subseqüentes serão tratadas de forma diferenciada, por exemplo ps -l 1,2,3,4,5,6 irá listar todos os processos de 1 a 6 no formato longo. Caso as identificações dos processos sejam informadas, eles serão listados, não importa de que forma. Caso um terminal tty seja informado, todos os processos associados a ele serão listados. Estas opções sobrepõem-se aos indicadores 'a' e 'x'.

O campo **STAT** , que aparece na lista gerada pelo comando ps, possui o seguinte significado: O primeiro campo será R para em execução, S para aguardando, D para aguardando sem interrupção, T para parado ou em depuração, ou Z para um processo zumbi. O segundo campo contém W caso o processo não tenha páginas residentes. O terceiro campo será N caso o processo tenha um valor de prioridade positivo.

NOTA:

%CPU mostra o percentual de cputime/realtime. Não deve ser superior a 100%. É igual ao tempo usado dividido pelos processos que estão sendo executados.

Exemplos:

ps ux
ps fx
ps aux
ps awx

20.2 - kill

O comando **kill** permite finalizar (matar) um processo em execução. **kill** envia um sinal específico para um determinado processo. Caso nenhum sinal seja especificado, o sinal TERM é enviado. Este sinal irá finalizar processos que esperam este tipo de mensagem. Para outros processos, pode ser necessário usar o sinal KILL (9), uma vez que este sinal não pode ser ignorado.

Sintaxe:

kill [-s sinal | -p] pid ...

kill -l [sinal]

Opções:

Opção	*Descrição*
pid ...	Especifica a lista de processos para os quais kill deve sinalizar. Cada pid pode ser um entre quatro opções: Um nome de processo no qual o processo nomeado receberá o sinal. n onde n é maior que 0. O processo com o pid (número de identificação) n receberá o sinal. -1 onde todos os processos de MAX_INT a 2 receberão o sinal, se permitido pelo dono do processo.

Opção	*Descrição*
	-n onde n é maior que 1, e todos os processos do grupo n receberão o sinal. Caso um sinal negativo seja informado, o sinal obrigatoriamente deve ser especificado antes, de outra forma será interpretado como o sinal a ser enviado.
-s	Especifica o sinal a ser enviado. O sinal pode ser informado como um dígito ou como um número.
-p	Especifica que kill pode somente listar a identificação do processo (pid) do processo nomeado, e não deve enviar-lhe um sinal.
-l	Lista uma relação dos nomes de sinais. Eles podem ser encontrados em /usr/include/linux/signal.h

Exemplos:

```
kill  -l
kill  -p  572
kill  -9  572
```

21 - COMANDOS BÁSICOS VIII

Neste capítulo trataremos dos comandos utilizados para impressão de arquivos e seu gerenciamento no Linux, além de alguns outros comandos variados.

21.1 - >

Este comando é um redirecionador, ou seja, redireciona a saída de um comando para um arquivo.

Exemplo:

```
ls  -al  /sbin/*  >  ~/meu_arq
```

O comando acima criará um arquivo contendo o resultado do comando ls.

21.2 - >>

Este comando também é um redirecionador, porém acrescenta a saída de um comando no fim de um arquivo.

Exemplo:

```
ls  -al  /bin/*  >>  ~/meu_arq
```

21.3 - &

Este caracter, acrescentado no final de uma linha de comando, informa ao sistema que aquela linha de comando deve ser executada em segundo plano (background). É interessante seu uso, pois libera o prompt, permitindo que outros comandos sejam entrados, enquanto os comandos anteriores são processados em segundo plano pelo sistema.

Exemplo:

Para este exemplo, deve-se estar no modo gráfico e executar o comando abaixo em um terminal virtual (xterm ou outro):

```
/usr/bin/netscape  &
```

21.4 - cat

Este comando concatena um arquivo e lista o resultado na saída padrão (normalmente o vídeo). Pode ser utilizado na visualização, cópia e impressão de arquivos como veremos nos exemplos a seguir.

Exemplos:

```
cat  meu_arq  |  more              Lista o conteúdo de meu_arq na tela.
cat  meu_arq  >  seu_arq    Copia o conteúdo de meu_arq para seu_arq.
cat  meu_arq  >  /dev/lp0   Copia meu_arq para a impressora lp0 (impressão direta).
```

21.5 - lpr

lpr é a interface entre a fila de impressão e os demais processos da máquina. Geralmente uma tarefa de impressão é iniciada com o comando:

```
lpr  [-P  fila]  arquivo_texto
```

Caso se omita o parâmetro -P que indica o nome da fila, o padrão será obtido através da variável de ambiente $PRINTER. Caso ela não esteja configurada, o nome padrão lp será utilizado.

Exemplo:

```
lpr  -P  lp0  meu_arq
```

21.6 - lpq

O comando **lpq** mostra as tarefas de impressão dos usuários.

Exemplo:

```
lpq  -P  lp0
lpq  -P  lp0  marcos
```

A listagem resultante será algo como:

```
lp0 está pronta e imprimindo
Ordem Dono      Tarefa  Arquivo       Tamanho Total
ativo           marcos     678     texto.txt      428934 bytes
1st             marcos     679     texto2.txt     859345 bytes
2nd             marcos     684     texto3.txt     985903 bytes
```

21.7 - lprm

O comando **lprm** remove as tarefas de impressão do usuário na fila. Caso nenhum número de tarefa seja informado, a tarefa ativa ou em impressão no momento será cancelada.

Exemplo:

lprm -P lp0 679
lprm -P lp0 marcos

21.8 - lpc

O comando **lpc** inicia uma interação com o superusuário, disponibilizando comandos de administração da impressora, como habilitar, suspender, alterar a ordem de impressão, etc. Mais informações podem ser obtidas com *man lpc*.

Exemplos:

#lpc
lpc> status lp0
lpc> topq lp0 684
lpc> topq lp0 marcos
lpc> clean all
lpc> exit

21.9 - pwd

O comando **pwd** informa o caminho completo do diretório corrente (atual).

21.10 - who

O comando **who** informa quem está conectado.

Exemplos:

who	Informa todos os usuários conectados.
whoami	Informa sob qual usuário você está conectado.
who -q	Informa quem são os usuários conectados e quantos são.
who -i	Informa quem são os usuários conectados e o tempo que estão ociosos no sistema. "." significa que está ativo e "old", que está ocioso há mais de 24 horas.

21.11 - df

O comando **df** informa quais são os sistemas de arquivos montados e qual a porcentagem de utilização do espaço em disco correspondente a cada um deles.

Exemplo:

df	
df -k	Informa o tamanho em blocos de 1k, se este não for o default.
df -a	Informa inclusive os sistemas de arquivos virtuais (que não ocupam o disco).

21.12 - du

O comando **du** informa o espaço ocupado pelos arquivos ou diretórios. Se não for informado o local, serão exibidas informações do diretório corrente.

Exemplos:

du -b /etc/fstab	Informa o tamanho do arquivo /etc/fstab em Bytes.
du -k /etc/fstab	Informa o tamanho do arquivo /etc/fstab em KBytes.
du -m /etc/fstab	Informa o tamanho do arquivo /etc/fstab em MBytes.
du -h /etc/fstab	Informa o tamanho do arquivo /etc/fstab na unidade mais adequada.
du /etc | more	Informa os tamanhos dos arquivos e diretórios localizados em /etc.
du	Informa os tamanhos dos arquivos e diretórios do diretório corrente.