

Sistema Operacional Linux

Professor: João Albertino Alves

Sumário

O Linux.....	1
Algumas Características do Linux.....	2
Software Livre.....	3
Estrutura básica de diretórios do Sistema Linux.....	4
Comandos.....	5
Opções.....	5
Parâmetros.....	5
Comandos Internos.....	5
Comandos Externos.....	6
Aviso de comando (Prompt).....	6
Interpretador de comandos.....	6
Interativa.....	7
Não-interativa.....	7
Terminal Virtual (console).....	7
Coringas.....	8
Swap.....	9
Definir tamanho do swap.....	10
Comandos básicos do sistema LINUX.....	11
Comandos para iniciar ou terminar uma seção.....	11
login.....	11
logout ou <Ctrl> + <d>.....	11
exit.....	11
Comandos para reinicializar ou desligar o computador.....	12
reboot.....	12
halt.....	12
Comandos de navegação.....	13
cd.....	13
ls.....	13
pwd.....	15
Comandos de manipulação de arquivos.....	16
touch.....	16
rm.....	16

mkdir.....	17
rmdir.....	17
mv.....	18
cp.....	19
Comandos de paginação.....	19
cat.....	19
more.....	20
less.....	20
Comandos para localização de arquivos.....	21
find.....	21
grep.....	21
whereis.....	22
locate.....	23
which.....	23
Comandos de arquivamento, compactação e descompactação.....	24
tar.....	24
gzip.....	24
gunzip.....	25
Comandos de ajuda.....	25
man.....	25
info.....	25
Comandos diversos.....	26
date.....	26
df.....	26
free.....	27
head.....	27
tail.....	27
uptime.....	28
su.....	28
who.....	28
clear.....	28
shutdown.....	29
Gerenciamento de usuários.....	30
useradd ou adduser.....	30

passwd.....	30
finger.....	30
userdel.....	31
users.....	31
w.....	31
Gerenciamento de privilégios.....	32
chmod.....	32
chown.....	33
Gerenciamento de processos.....	34
<Ctrl>+<C>.....	34
<Ctrl>+<Z>.....	34
&.....	34
jobs.....	34
bg.....	34
fg.....	35
ps.....	35
kill.....	35
nohup.....	36
nice.....	36
renice.....	36

O Linux

O Linux é um sistema operacional que foi criado em 1991 por *Linus Torvalds* na universidade de Helsinki na Finlândia. É um sistema Operacional de código aberto distribuído gratuitamente pela Internet. Seu código fonte é liberado como *Free Software* (software livre) o aviso de copyright do kernel feito por Linus descreve detalhadamente isto e mesmo ele está proibido de fazer a comercialização do sistema.

Isto quer dizer que você não precisa pagar nada para usar o Linux, e não é crime fazer cópias para instalar em outros computadores, nós inclusive incentivamos você a fazer isto. Ser um sistema de código aberto pode explicar a performance, estabilidade e velocidade em que novos recursos são adicionados ao sistema.

Para rodar o Linux você precisa, no mínimo, de um computador 386 SX com 2 MB de memória e 40MB disponíveis em seu disco rígido para uma instalação básica e funcional.

O sistema segue o padrão *POSIX* que é o mesmo usado por sistemas *UNIX* e suas variantes. Assim, aprendendo o Linux você não encontrará muita dificuldade em operar um sistema do tipo *UNIX*, *FreeBSD*, *HPUX*, *SunOS*, etc., bastando apenas aprender alguns detalhes encontrados em cada sistema.

O código fonte aberto permite que qualquer pessoa veja como o sistema funciona (útil para aprendizado), corrija algum problema ou faça alguma sugestão sobre sua melhoria, esse é um dos motivos de seu rápido crescimento, do aumento da compatibilidade de periféricos (como novas placas sendo suportadas logo após seu lançamento) e de sua estabilidade.

Outro ponto em que ele se destaca é o suporte que oferece a placas, CD-Roms e outros tipos de dispositivos de última geração e mais antigos (a maioria deles já ultrapassados e sendo completamente suportados pelo sistema operacional). Este é um ponto forte para empresas que desejam manter seus micros em funcionamento e pretendem investir em avanços tecnológicos com as máquinas que possui.

Hoje o Linux é desenvolvido por milhares de pessoas espalhadas pelo mundo, cada uma fazendo sua contribuição ou mantendo alguma parte do kernel gratuitamente. *Linus Torvalds* ainda trabalha em seu desenvolvimento e também ajuda na coordenação entre os desenvolvedores.

O suporte ao sistema também se destaca como sendo o mais eficiente, rápido e barato do que qualquer programa comercial disponível no mercado.

Existem centenas de consultores especializados espalhados ao redor do mundo. Você pode se inscrever em uma lista de discussão e relatar sua dúvida ou alguma falha, e sua mensagem será vista por centenas de usuários na Internet e algum irá te ajudar ou avisará as pessoas responsáveis sobre a falha encontrada para devida correção.

Algumas Características do Linux

- É de graça e desenvolvido voluntariamente por programadores experientes, hackers, e contribuidores espalhados ao redor do mundo que tem como objetivo a contribuição para a melhoria e crescimento deste sistema operacional. Muitos deles estavam cansados do excesso de propaganda (Marketing) e baixa qualidade de sistemas comerciais existentes
- Convivem sem nenhum tipo de conflito com outros sistemas operacionais (com o DOS, Windows, OS/2) no mesmo computador.
- Multitarefa real
- Multiusuário
- Suporte a nomes extensos de arquivos e diretórios (255 caracteres)
- Conectividade com outros tipos de plataformas como *Apple, Sun, Macintosh, Sparc, Alpha, PowerPc, ARM, Unix, Windows, DOS, etc.*
- Proteção entre processos executados na memória RAM
- Suporte há mais de 63 terminais virtuais (consoles)
- Modularização - O GNU/Linux somente carrega para a memória o que é usado durante o processamento, liberando totalmente a memória assim que o programa/dispositivo é finalizado
- Devido a modularização, os drivers dos periféricos e recursos do sistema podem ser carregados e removidos completamente da memória RAM a qualquer momento. Os drivers (módulos) ocupam pouco espaço quando carregados na memória RAM (cerca de 6Kb para a Placa de rede NE 2000, por exemplo)
- Não há a necessidade de se reiniciar o sistema após modificar a configuração de qualquer periférico ou parâmetros de rede. Somente é necessário reiniciar o sistema no caso de uma instalação interna de um novo periférico, falha em algum hardware (queima do processador, placa mãe, etc.).
- Não precisa de um processador potente para funcionar. O sistema roda bem em computadores 386sx 25 com 4MB de memória RAM (sem rodar o sistema gráfico X, que é recomendado 8MB de RAM). Já pensou no seu desempenho em um 486 ou Pentium.
- O crescimento e novas versões do sistema não provocam lentidão, pelo contrário, a cada nova versão os desenvolvedores procuram buscar maior compatibilidade, acrescentar recursos úteis e melhor desempenho do sistema (como o que aconteceu na passagem do kernel 2.0.x para 2.2.x).
- Não é requerida uma licença para seu uso. O GNU/Linux é licenciado de acordo com os termos da GNU.
- Acessa sem problemas discos formatados pelo DOS, Windows, Novell, OS/2, NTFS, SunOS, Amiga, Atari, Mac, etc.
- Utiliza permissões de acesso a arquivos, diretórios e programas em execução na memória RAM.
- **NÃO EXISTEM VÍRUS NO LINUX!** Em 14 anos de existência, nunca foi registrado NENHUM tipo de vírus neste sistema. Isto tudo devido a grande segurança oferecida pelas permissões de acesso do sistema que funcionam inclusive durante a execução de programas.
- Rede TCP/IP mais rápida que no Windows e tem sua pilha constantemente melhorada. O GNU/Linux tem suporte nativo a redes TCP/IP e não depende de uma camada intermediária como o Winsock. Em acessos via modem a Internet, a velocidade de transmissão é 10% maior. Jogadores do Quake ou qualquer outro tipo de jogo via Internet preferem o GNU/Linux por causa da maior velocidade do Jogo em rede. É fácil rodar um servidor Quake em seu computador e assim jogar contra vários adversários via Internet.

- Roda aplicações *DOS* através do DOSEMU. Para se ter uma idéia, é possível dar o boot em um sistema *DOS* qualquer dentro dele e ao mesmo tempo usar a multitarefa deste sistema.
- Suporte a dispositivos infravermelho.
- Suporte a rede via rádio amador.
- Suporte a dispositivos Plug-and-Play.
- Suporte a dispositivos USB.
- Vários tipos de firewalls de alta qualidade e com grande poder de segurança de graça.
- Roteamento estático e dinâmico de pacotes.
- Ponte entre Redes (bridge).
- Proxy Tradicional e Transparente.
- Possui recursos para atender a mais de um endereço IP na mesma placa de rede, sendo muito útil para situações de manutenção em servidores de redes ou para a emulação de "mais computadores" virtualmente. O servidor WEB e FTP podem estar localizados no mesmo computador, mas o usuário que se conecta tem a impressão que a rede possui servidores diferentes.
- O sistema de arquivos usado pelo GNU/Linux (Ext2) organiza os arquivos de forma inteligente evitando a fragmentação e fazendo-o um poderoso sistema para aplicações multi-usuárias exigentes e gravações intensivas.
- Permite a montagem de um servidor Web, E-mail, News, etc. com um baixo custo e alta performance. O melhor servidor Web do mercado, o Apache, é distribuído gratuitamente junto com o Linux.
- Por ser um sistema operacional de código aberto, você pode ver o que o código fonte (o que foi digitado pelo programador) e adaptá-lo as suas necessidades ou de sua empresa. Esta característica é uma segurança a mais para empresas sérias e outros que não querem ter seus dados roubados (você não sabe o que um sistema sem código fonte faz, na realidade, enquanto esta processando o programa).
- Suporte a diversos dispositivos e periféricos disponíveis no mercado, tanto os novos como obsoletos.
- Pode ser executado em várias arquiteturas diferentes (Intel, Macintosh, Alpha, Arm, etc.).
- Consultores técnicos especializados no suporte ao sistema espalhados por todo o mundo.
- Entre muitas outras características que você descobrirá durante o uso do sistema.

Software Livre

Softwares Livres são programas que possuem o código fonte incluído (o código fonte é o que o programador digitou para fazer o programa) e você pode modificar ou distribuí-los livremente. Existem algumas licenças que permitem isso, a mais comum é a *General Public Licence* (ou GPL).

Os softwares livres muitas vezes são chamados de *programas de código aberto* (ou OSS). Muito se acredita no compartilhamento do conhecimento e tendo liberdade de cooperar uns com outros, isto é importante para o aprendizado de como as coisas funcionam e novas técnicas de construção. Existe uma longa teoria desde 1950 valorizando isto, muitas vezes pessoas assim são chamadas de "Hackers Éticos".

Outros procuram aprender mais sobre o funcionamento do computador e seus dispositivos (periféricos) e muitas pessoas estão procurando por meios de evitar o preço absurdo de softwares comerciais através de programas livres que possuem qualidade igual ou superior, devido a cooperação em seu desenvolvimento.

Você pode modificar o código fonte de um software livre a fim de melhorá-lo ou acrescentar mais recursos e o autor do programa pode ser contactado sobre a alteração e os benefícios que sua modificação fez no programa, e esta poderá ser incluída no programa principal. Deste modo, milhares de pessoas que usam o programa se beneficiarão de sua contribuição.

Estrutura básica de diretórios do Sistema Linux

O sistema GNU/Linux possui a seguinte estrutura básica de diretórios:

/bin	Contém arquivos programas do sistema que são usados com frequência pelos usuários.
/boot	Contém arquivos necessários para a inicialização do sistema.
/cdrom	Ponto de montagem da unidade de CD-ROM.
/dev	Contém arquivos usados para acessar dispositivos (periféricos) existentes no computador.
/etc	Arquivos de configuração de seu computador local.
/floppy	Ponto de montagem de unidade de disquetes.
/home	Diretório que contém os diretórios e os arquivos dos usuários.
/lib	Bibliotecas compartilhadas pelos programas do sistema e módulos do kernel.
/lost+found	Local para a gravação de arquivos/diretórios recuperados pelo utilitário fsck.ext2. Cada partição possui seu próprio diretório lost+found.
/mnt	Ponto de montagem temporário.
/proc	Sistema de arquivos do kernel. Este diretório não existe em seu disco rígido, ele é colocado lá pelo kernel e usado por diversos programas que fazem sua leitura, verificam configurações do sistema ou modificar o funcionamento de dispositivos do sistema através da alteração em seus arquivos.
/root	Diretório do usuário root.
/sbin	Diretório de programas usados pelo superusuário (root) para administração e controle do funcionamento do sistema.
/tmp	Diretório para armazenamento de arquivos temporários criados por programas.
/usr	Contém a maior parte de seus programas. Normalmente acessível somente como leitura.
/var	Contém maior parte dos arquivos que são gravados com frequência pelos programas do sistema, e-mails, spool de impressora, cache, logs, etc.

Comandos

Comandos são ordens que passamos ao sistema operacional para executar uma determinada tarefa. Cada comando tem uma função específica, devemos saber a função de cada comando e escolher o mais adequado para fazer o que desejamos, por exemplo:

- ls - Mostra arquivos de diretórios
- cd - Para mudar de diretório

Esta apostila tem uma lista de vários comandos organizados por categoria com a explicação sobre o seu funcionamento e as opções aceitas (incluindo alguns exemplos).

É sempre usado um **espaço** depois do comando para separá-lo de uma opção ou parâmetro que será passado para o processamento.

Um comando pode receber **opções** e **parâmetros**:

Opções

As opções são usadas para controlar como o comando será executado, por exemplo, para fazer uma listagem mostrando o dono, grupo, tamanho dos arquivos você deve digitar ls -l. Opções podem ser passadas ao comando através de um "-" ou "--":

Opção identificada por uma letra: Podem ser usadas mais de uma opção com um único hífen. O comando ls -l -a é a mesma coisa de ls -la

Opção identificada por um nome: O comando ls --all é equivalente a ls -a. Pode ser usado tanto "-" como "--", mas há casos em que somente "-" ou "--" esta disponível.

Parâmetros

Um parâmetro identifica o caminho, origem, destino, entrada padrão ou saída padrão que será passada ao comando. Se você digitar:

ls /usr/doc/copyright, /usr/doc/copyright será o parâmetro passado ao comando ls, neste caso queremos que ele liste os arquivos do diretório /usr/doc/copyright . É normal errar o nome de comandos, mas não se preocupe, quando isto acontecer o sistema mostrará a mensagem command not found (comando não encontrado) e voltará ao aviso de comando. As mensagens de erro não fazem nenhum mal ao seu sistema, somente dizem que algo deu errado para que você possa corrigir e entender o que aconteceu.

Por exemplo: ls -la /usr/doc, ls é o comando, -la é a opção passada ao comando, e /usr/doc é o diretório passado como parâmetro ao comando ls.

Comandos Internos

São comandos que estão localizados dentro do interpretador de comandos (normalmente o Bash) e não no disco arquivo. Eles são carregados na memória RAM do computador junto com o

interpretador de comandos. Quando executa um comando, o interpretador de comandos verifica primeiro se ele é um Comando Interno caso não seja é verificado se é um Comando Externo.

Exemplos de comandos internos : cd, exit, echo, bg, fg, source, help

Comandos Externos

São comandos que estão localizados no disco. Os comandos são procurados no disco usando o path e executados assim que encontrados.

Exemplos de comandos externos: find, locate, date

Aviso de comando (Prompt)

Aviso de comando (ou Prompt), é a linha mostrada na tela para *digitação de comandos* que serão passados ao interpretador de comandos para sua execução.

A posição onde o comando será digitado é marcado um traço piscante na tela chamado de *cursor*.

Tanto em shells texto como em gráficos é necessário o uso do cursor para sabermos onde iniciar a digitação de textos e nos orientarmos quanto a posição na tela.

O aviso de comando do usuário root é identificado pelo símbolo "#" (tralha), e o aviso de comando de usuários é identificado pelo símbolo "\$" (dolar) . Isto é padrão em sistemas LINUX.

Você pode retornar comandos já digitados pressionando as teclas Seta para cima / Seta para baixo.

A tela pode ser rolada para baixo ou para cima segurando a tecla SHIFT e pressionando PGUP ou PGDOWN. Isto é útil para ver textos que rolaram rapidamente para cima. Abaixo algumas dicas sobre a edição da linha de comandos (não é necessário se preocupar em decorá-los):

- Pressione a tecla Backspace ("←") para apagar um caracter à esquerda do cursor.
- Pressione a tecla Del para apagar o caracter acima do cursor.
- Pressione CTRL+A para mover o cursor para o início da linha de comandos.
- Pressione CTRL+E para mover o cursor para o fim da linha de comandos.
- Pressione CTRL+U para apagar o que estiver à esquerda do cursor. O conteúdo apagado é copiado para uso com CTRL+Y.
- Pressione CTRL+K para apagar o que estiver à direita do cursor. O conteúdo apagado é copiado para uso com CTRL+Y.
- Pressione CTRL+L para limpar a tela e manter o texto que estiver sendo digitado na linha de comando (parecido com o comando clear).
- Pressione CTRL+Y para colocar o texto que foi apagado na posição atual do cursor.

Interpretador de comandos

Também conhecido como "shell". É o programa responsável em interpretar as instruções enviadas pelo usuário e seus programas ao sistema operacional (o kernel). Ele que executa comandos lidos do dispositivo de entrada padrão (teclado) ou de um arquivo executável. É a principal ligação entre o usuário, os programas e o kernel. O GNU/Linux possui diversos tipos de interpretadores de

comandos, entre eles posso destacar o bash, ksh, csh, sh, etc. O interpretador de comandos (shell) padrão do Linux é o Bash.

O interpretador de comandos do DOS, por exemplo, é o command.com.

Os comandos podem ser enviados de duas maneiras para o interpretador, interativa e não-interativa:

Interativa: Os comandos são digitados no aviso de comando e passados ao interpretador de comandos um a um. Neste modo, o computador depende do usuário para executar uma tarefa, ou próximo comando.

Não-interativa: São usados arquivos de comandos criados pelo usuário (scripts) para o computador executar os comandos na ordem encontrada no arquivo. Neste modo, o computador executa os comandos do arquivo um por um e dependendo do término do comando, o script pode checar qual será o próximo comando que será executado e dar continuidade ao processamento.

Este sistema é útil quando temos que digitar por várias vezes seguidas um mesmo comando ou para compilar algum programa complexo.

O shell Bash possui ainda outra característica interessante: A completção dos nomes. Isto é feito pressionando-se a tecla TAB. Por exemplo, se digitar ls tes e pressionar <tab>, o Bash localizará todos os arquivos que iniciam com tes e completará o restante do nome. Caso a completção de nomes encontre mais do que uma expressão que satisfaça a pesquisa, ou nenhuma, é emitido um beep. A completção de nomes funciona sem problemas para comandos internos.

Exemplo: ech (pressione 'TAB'). ls /ho (pressione 'TAB')

Terminal Virtual (console)

Terminal (ou console) é o teclado e tela conectados em seu computador. O GNU/Linux faz uso de sua característica *multi-usuária* usando os "terminais virtuais". Um terminal virtual é uma segunda seção de trabalho completamente independente de outras, que pode ser acessada no computador local ou remotamente via telnet, rsh, rlogin, etc.

No GNU/Linux, em modo texto, você pode acessar outros terminais virtuais segurando a tecla ALT e pressionando F1 a F6. Cada tecla de função corresponde a um número de terminal do 1 ao 6 (o sétimo é usado por padrão pelo ambiente gráfico X). O GNU/Linux possui mais de 63 terminais virtuais, mas apenas 6 estão disponíveis inicialmente por motivos de economia de memória RAM .

Se estiver usando o modo gráfico, você deve segurar CTRL + ALT enquanto pressiona uma tela de <F1> a <F6>.

Um exemplo prático: Se você estiver usando o sistema no Terminal 1 com o nome "joao" e desejar entrar como "root" para instalar algum programa, segure ALT enquanto pressiona <F2> para abrir o segundo terminal virtual e faça o login como "root".

Será aberta uma nova seção para o usuário "root" e você poderá retornar a hora que quiser para o primeiro terminal pressionando ALT+<F1>.

Coringas

Coringas (ou referência global) é um recurso usado para especificar um ou mais arquivos ou diretórios do sistema de uma só vez.

Este recurso permite que você faça a filtragem do que será listado, copiado, apagado, etc. São usados 3 tipos de coringas no GNU/Linux:

- "*" - Faz referência a um nome completo/restante de um arquivo/diretório.
- "?" - Faz referência a uma letra naquela posição.
- [padrão] - Faz referência a um padrão contido em um nome de arquivo ou diretório. Padrão pode ser:
 - [a-z][1-0] - Faz referência aos caracteres de a até z ou de 1 até 10.
 - [a,z][1,0] - Faz referência aos caracteres de a e z ou 1 e 10 naquela posição.
 - [a-z,1,0] - Faz referência aos caracteres de a até z e 1 e 10 naquela posição.

A procura de caracteres é "Case Sensitive" assim se você deseja que sejam localizados todos os caracteres alfabéticos você deve usar [a-zA-Z].

Caso a expressão seja seguida de um "^", faz referência a qualquer caracter **exceto** o da expressão. Por exemplo [^abc] faz referência a qualquer caracter exceto a, b e c.

Atenção os 3 tipos de coringas ("*", "?" e "[]") podem ser usados juntos. Para entender melhor vamos a prática:

Vamos dizer que tenha 5 arquivo no diretório /home/teste:

```
teste1.txt, teste2.txt, teste3.txt, teste4.new, teste5.new
```

Caso deseje listar *todos* os arquivos do diretório /home/teste você pode usar o coringa "*" para especificar todos os arquivos do diretório:

```
cd /home/teste e ls * ou ls /home/teste/*.
```

Não tem muito sentido usar o comando ls com "*" porque todos os arquivos serão listados se o ls for usado sem nenhum coringa.

Agora para listar todos os arquivos teste1.txt, teste2.txt, teste3.txt com excessão de teste4.new, teste5.new, podemos usar inicialmente 3 métodos:

1. Usando o comando ls *.txt que pega todos os arquivos que começam com qualquer nome e terminam com .txt.
2. Usando o comando ls teste?.txt, que pega todos os arquivos que começam com o nome teste, tenham qualquer caracter no lugar do coringa "?" e terminem com .txt. Com o exemplo acima teste*.txt também faria a mesma coisa, mas se também tivéssemos um arquivo chamado teste4.txt este também seria listado.

3. Usando o comando `ls teste[1-3].txt`, pega todos os arquivos que começam com o nome teste, tenham qualquer caracter entre o número 1-3 no lugar da 6a letra e terminem com `.txt`. Neste caso se obtém uma filtragem mais exata, pois o coringa “?” especifica qualquer caracter naquela posição e `[]` especifica números, letras ou intervalo que será usado.

Agora para listar somente `teste4.new` e `teste5.new` podemos usar os seguintes métodos:

1. `ls *.new` que lista todos os arquivos que terminam com `.new`
2. `ls teste?.new` que lista todos os arquivos que começam com teste, contenham qualquer caracter na posição do coringa “?” e terminem com `.new`.
3. `ls teste[4,5].*` que lista todos os arquivos que começam com teste contenham os números de 4 ou 5 naquela posição e terminem com qualquer extensão.

Existem muitas formas de se fazer a mesma coisa, isto depende do gosto de cada um. O que pretendi fazer aqui foi mostrar como especificar mais de um arquivo de uma só vez. O uso de coringas será útil ao copiar arquivos, apagar, mover, renomear, e nas mais diversas partes do sistema. Aliás esta é uma característica do GNU/Linux: permitir que a mesma coisa possa ser feita com liberdade de várias maneiras diferentes.

Swap

A principal função do swap é permitir a execução e manipulação de programas maiores que a capacidade da RAM principal.

Neste tópico vou comentar sobre a partição linux-swap, ela será utilizada como extensão da memória RAM, sempre que a memória RAM for insuficiente o sistema fará uso deste swap, serve somente para isto.

Esta partição é necessária para evitar travamentos, erros ou falhas no sistema, mas o sistema deve ser administrado de forma a não utilizá-la com frequência, o motivo é simples, o acesso a esta partição é muito mais lento que à memória RAM, portanto quando for usada fatalmente o sistema ficará mais lento.

Monitore o uso da RAM e do linux-swap através do comando `top`, observe que o Linux administra muito bem o uso da RAM, dificilmente vai encontrar memória livre, mesmo aumentando a memória vai parecer como toda em uso, isto é muito bom, indica que o sistema está otimizando o uso de toda a memória RAM.

O problema surge quando aparece muito uso do swap, isto é ruim porque a execução de programas ficará muito lento. Tendo possibilidade e disponibilidade tente aumentar a memória RAM ao ponto do sistema não utilizar este espaço de swap, ou se possível administrar o uso de programas, por exemplo não executar vários programas que consome muita memória ao mesmo tempo, ou tentar outros equivalentes que sejam mais leves, por exemplo o KDE é um ambiente gráfico pesado, consome muita memória e CPU, tem outros ambientes mais leves, faça testes com outros ambientes gráficos e tente avaliar isto e descobrir qual deles é mais indicado para a máquina em uso.

Definir tamanho do swap

Em quase toda a documentação sobre este assunto você vai encontrar a seguinte definição:

swap = o dobro da memória RAM.

Em resumo quer dizer que se a máquina tem 64 MB de RAM a partição linux-swap deve ter o tamanho de 128MB, seguindo esta esta lógica, para uma máquina com 1GB de RAM precisaria de um swap de 2GB, isto é um erro, dificilmente uma máquina com 1 GB de RAM precisaria de 2GB de swap e também dificilmente utilizaria o swap, ao passo que a máquina com 64 MB de RAM com certeza vai utilizar o swap e os 128 MB pode ser pouco.

Existem casos especiais que se recomenda uma partição de swap fora desta regra, é o caso do banco de dados da Oracle, mesmo assim tem uma relação com a quantidade de registros e outros requisitos.

Nos micros atuais é comum os HD ter grande capacidade e muitas vezes não faz falta 1 ou 2 GB, sendo este o seu caso, então pode usar uma partição swap até maior que o recomendado, mas não esqueça de monitorar o seu uso, em micros mais antigos geralmente com HD pequeno e pouca RAM requer uma avaliação mais cuidadosa para definir esta partição, mas tem meios de aumentar o tamanho mesmo depois da instalação.

Para o dia a dia, as máquinas que tenham até 128 MB de RAM, recomendo criar a partição swap com espaço um pouco maior do que o dobro, tenho adotado um tamanho fixo de 300 MB.

Em resumo, o uso e tamanho da partição swap está relacionado ao total de memória e a quantidade requerida pelo sistema pra rodar os aplicativos, quando faltar memória RAM o sistema recorre ao swap, você pode monitorar o uso do swap para saber quando precisa de mais RAM ou aumentar o swap.

A partição de swap nunca deverá ser totalmente usada, caso isto ocorra pode indicar que para teu uso ela está pequena e deverá preferencialmente aumentar a memória RAM, ou então aumentar o tamanho do swap.

COMANDOS BÁSICOS DO SISTEMA LINUX

Os comandos em Linux possuem algumas características particulares. Eles podem ser controlados por opções e devem ser digitados em letras minúsculas.

Comandos para iniciar ou terminar uma seção

Login - cancela a sessão atual e inicia uma nova sessão de usuário.

Exemplo:

```
# login
```

logout ou <Ctrl> + <d> - Encerra a sessão do usuário.

Exemplo:

```
# logout  
# <Ctrl> + <d>
```

exit - encerra o shell de comandos corrente.

Exemplo:

```
# exit
```

Comandos para reinicializar ou desligar o computador

reboot - reinicializa o computador

Exemplo:

```
# reboot
```

halt - desliga o computador.

Exemplo:

```
# halt
```

Comandos de navegação

cd – muda o diretório de trabalho.

Sintaxe: `cd <diretório>`

Diretório - é o nome do diretório para o qual você deseja mudar. O símbolo "." refere-se ao diretório corrente onde o usuário se encontra e o símbolo ".." refere-se ao "diretório-pai", imediatamente acima do diretório corrente. Para mover para um "diretório-pai", ou seja, um diretório acima do que você está, use o comando :

```
# cd .. (note o espaço entre "cd" e "..")
```

Você também pode usar nomes-de-caminho (pathnames) como argumento para o comando `cd`.

Exemplo :

```
# cd /var/log
```

Posicionará diretamente em "log". O uso de "cd" sem nenhum argumento fará com que você retorne para o seu "home-directory" .

ls - Exibe informações sobre arquivos e diretórios, é usado para visualizar o conteúdo de um diretório.

Sintaxe: `ls <diretório> [opções]`

Quando executado sem qualquer parâmetro, mostra o conteúdo do diretório corrente.

Exemplo:

```
# ls
```

Mostra o conteúdo do diretório corrente naquele momento. Como na maioria dos comandos LINUX, "ls" pode ser controlado por opções que começam com um hífen (-). Tenha sempre o cuidado de deixar um espaço antes do hífen. Uma opção bastante útil é -a (que vem do inglês 'all', tudo), e irá mostrar detalhes que você nunca imaginou sobre o seu diretório.

Exemplo:

```
# cd - retorna para o diretório do user root  
# ls -a
```

Digitando estes comandos em sequência, o sistema vai para o seu home directory do usuário corrente, através do comando `cd` e em seguida mostra o conteúdo do mesmo, que será exibido da seguinte forma:

```
.          .bacshrc  .fvwmrc  
..         .emacs   .xinitrc  
.bash_history .exerc
```

Aqui, o ponto simples refere-se ao diretório corrente, e o ponto duplo refere-se ao diretório imediatamente acima dele. Mas o que são estes outros arquivos que se iniciam com um ponto? Eles são chamados arquivos escondidos (ocultos). A colocação do ponto na frente de seus nomes os impede de serem mostrados durante um comando "ls" normal.

Outra opção bastante utilizada é -l (que vem do inglês "long"). Ela apresenta informação extra sobre os arquivos .

Exemplo:

```
# ls -l
```

Apresenta, além do conteúdo do diretório, todas os detalhes sobre cada arquivo pertencente a ele. Por exemplo, suponha que você tenha executado este comando e na tela apareceu algo assim:

```
-rw-r--r--  1 xyz users 2321  Mar 15 1994  Fontmap  
-rw-r--r--  1 xyz users 14567 Feb  3 1995  file003  
drwxr-xr-x  2 xyz users 1024  Apr 23 1995  Programs  
drwxr-xr-x  3 xyz users 1024  Apr 30 1995  bitmaps
```

Lendo da esquerda para direita, este primeiro caracter indica se o arquivo é um diretório (d) ou um arquivo comum (-). Em seguida temos as permissões de acesso ao arquivo, sendo as três primeiras referentes ao proprietário, as três seguintes ao grupo e, por último, aos demais usuários.

A segunda coluna desta listagem mostra o número de links que o arquivo possui.

A terceira coluna mostra o proprietário do referido arquivo, neste caso, o usuário cujo user name é "xyz".

Na próxima coluna é mostrado o grupo ao qual pertence o proprietário do arquivo (no exemplo temos o grupo users). Na quinta coluna temos o tamanho do arquivo em bytes.

Por fim, na sexta e sétima colunas, temos a data da última modificação feita no arquivo e o nome do mesmo, respectivamente. Vale lembrar que várias opções podem ser usadas de forma composta. Por exemplo, podemos executar o comando:

```
# ls -la
```

E este mostrará todos os detalhes que as opções -l e -a dispõem.

pwd - Exibe a estrutura de o diretório corrente.

Sintaxe: pwd

Este comando é utilizado para exibir o diretório corrente no sistema de arquivos.

Exemplo:

```
# pwd  
/home/usuario
```

Comandos de manipulação de arquivos

touch - atualiza a última data de acesso ao arquivo. Caso o arquivo não exista, será criado vazio por padrão.

Um arquivo vazio será criado se o nome especificado ainda não existir.

Sintaxe: touch [opções] [mmddhhMM[yy]] nome-do-arquivo

sendo :

mm	Mês
dd	Dia
hh	Hora
MM	Minuto
yy	ano (últimos dois dígitos)

Se não for especificada nenhuma data, a data atual será utilizada.

Opções:

- a atualiza somente o tempo de acesso respectivo.
- m atualiza somente o tempo de modificação.
- c previne a criação de um arquivo se ele não existia anteriormente.

As opções default são : -am

rm - Este comando é utilizado para apagar arquivos.

Sintaxe: rm (arquivo 1) (arquivo 2) ... (arquivo n)

onde (arquivo 1) até (arquivo n) são os arquivos a serem apagados.

Se um arquivo não possuir permissão de escrita e a saída-padrão for um terminal, todo o conjunto de permissões do arquivo será exibido, seguido por um ponto de interrogação. É um pedido de confirmação. Se a resposta começar com "y" ("yes" = sim), o arquivo será apagado, caso contrário ele será mantido no sistema.

Quando você apaga um arquivo com o comando "rm", você está apagando somente um link (ligação ou entrada) para um arquivo. Um arquivo somente será apagado verdadeiramente do sistema quando ele não possuir mais nenhuma ligação para ele, isto é, nenhum link referenciando-o. Geralmente, arquivos possuem somente um link, portanto o uso do comando "rm" irá apagar o(s) arquivo(s). No entanto, se um arquivo possuir muitos links, o uso de "rm" irá apagar somente uma ligação; neste caso, para apagar o arquivo, é necessário que você apague todos os links para este arquivo.

Você pode verificar o número de links que um arquivo possui utilizando o comando ls, com a opção "-l".

Opções:

- f remove todos os arquivos (mesmo se estiverem com proteção de escrita) em um diretório sem pedir confirmação do usuário.
- i esta opção pedirá uma confirmação do usuário antes de apagar o(s) arquivo(s) especificado(s).
- r opção recursiva para remover um diretório e todo o seu conteúdo, incluindo quaisquer subdiretórios e seus arquivos.

CUIDADO : diretórios e seus conteúdos removidos com o comando "rm -r" não podem ser recuperados.

mkdir - Cria diretórios.

Sintaxe: mkdir <diretório 1> <diretório 2> ...<diretório n>

onde (diretório 1) até (diretório n) são os diretórios a serem criados.

As entradas padrão em um diretório (por exemplo, os arquivos ".", para o próprio diretório, e ".." para o diretório pai) são criadas automaticamente. A criação de um diretório requer permissão de escrita no diretório pai.

O identificador de proprietário (owner id), e o identificador de grupo (group id) dos novos diretórios são configurados para os identificadores de proprietário e de grupo do usuário efetivo, respectivamente.

Opções:

- m (mode) esta opção permite aos usuários especificar o modo a ser usado para os novos diretórios.
- p com esta opção, mkdir cria o nome do diretório através da criação de todos os diretórios-pai não existentes primeiro.

Exemplo:

```
mkdir -p /home/joao/docs
```

cria a estrutura de subdiretórios "/home/arqs/texto".

rmdir - Utilizado para apagar diretórios vazios.

Sintaxe: rmdir (diretório 1) (diretório 2) ... (diretório n)

onde (diretório 1) até (diretório n) são os diretórios a serem apagados.

O comando "rmdir" se recusa a apagar um diretório inexistente, exibindo a mensagem:

```
rmdir : (nome-do-diretório) : No such file or directory
```

Quando usar "rmdir", lembre-se que o seu diretório de trabalho corrente não pode estar contido no(s) diretório(s) a ser(em) apagado(s). Se você tentar remover seu próprio diretório corrente, será exibida a seguinte mensagem:

```
rmdir : . : Operation not permitted
```

Se o diretório o qual você deseja remover não estiver vazio, utilize o comando "cd" para acessar os arquivos dentro do diretório, e então remova estes arquivos utilizando o comando "rm".

Opções:

-p permite aos usuários remover o diretório e seu diretório pai, o qual se torna vazio. Uma mensagem será exibida na saída padrão informando se o caminho ("path") inteiro foi removido ou se parte do caminho persiste por algum motivo.

CUIDADO : diretórios removidos com o comando "rmdir" não podem ser recuperados!

mv - Move arquivos para um outro arquivo ou diretório.

O comando "mv" é utilizado para mover arquivo(s) para outro arquivo ou diretório. Este comando faz o equivalente a uma cópia seguida pela deleção do arquivo original. Pode ser usado para renomear arquivos.

Sintaxe: mv (arquivo 1) (arquivo 2) ... (arquivo n) (destino)

onde (arquivo 1) até (arquivo n) são os arquivos a serem movidos, e (destino) é o arquivo ou o diretório para onde os arquivos serão movidos.

Se (destino) não for um diretório, somente um arquivo deverá ser especificado como fonte. Se for um diretório, mais de um arquivo poderá ser especificado.

Se (destino) não existir, "mv" criará um arquivo com o nome especificado. Se (destino) existir e não for um diretório, seu conteúdo será apagado e o novo conteúdo será escrito no lugar do antigo. Se (destino) for um diretório, o(s) arquivo(s) será(ão) movido(s) para este diretório.

Os arquivos "fonte" e "destino" não precisam compartilhar o mesmo diretório pai.

Opções:

-i com esta opção, "mv" irá perguntar a você se é permitido escrever por cima do conteúdo de um arquivo destino existente.

Uma resposta "y" (yes = sim) significa que a operação poderá ser executada. Qualquer outra resposta impedirá que "mv" escreva por cima do conteúdo de um arquivo já existente.

Exemplo:

```
# pwd  
/home/arqs/texto
```

```
# ls  
teste
```

```
# mv teste ../temp1  
# ls ../temp1  
teste
```

Neste exemplo, o diretório teste foi movido de temp para temp1 com o comando "mv".

cp - Copia arquivos para um outro arquivo ou diretório.

Sintaxe: cp (arquivo1) (arquivo2) ... (arquivo n) (destino)

onde (arquivo1) até (arquivo n) são os arquivos a serem copiados, e (destino) é o arquivo ou o diretório para onde os arquivos serão copiados. O(s) arquivo(s) fonte(s) e o (destino) não podem ter o mesmo nome. Se o arquivo-destino não existe, "cp" criará um arquivo com o nome especificado em . Se o arquivo-destino já existia antes e não for um diretório, "cp" escreverá o novo conteúdo por cima do antigo.

Exemplo:

```
# cp -r temp temp1
```

Este comando copia todos os arquivos e subdiretórios dentro do diretório temp para um novo diretório temp1. Esta é uma cópia recursiva, como designado pela opção -r. Se você tentar copiar um diretório sem utilizar esta opção, você verá uma mensagem de erro.

Comandos de paginação

cat - Usado para concatenar arquivos. Também usado para exibir todo o conteúdo de um arquivo de uma só vez, sem pausa.

Sintaxe: cat < arquivo1 > < arquivo2 >... < arquivo n > ,

onde (arquivo1) até (arquivo n) são os arquivos a serem mostrados. "cat" lê cada arquivo em sequência e exibe-o na saída padrão.

Exemplo:

```
# cat texto1.txt  
# cat texto.txt  
# cat texto1.txt texto2.txt
```

exibirá o arquivo em seu terminal; e a linha de comando abaixo:

Sintaxe:

```
cat < arquivo1 > < arquivo2 > > < arquivo3 >
```

concatenará "arquivo1" e "arquivo2", e escreverá o resultado em "arquivo3". O símbolo ">", usado para redirecionar a saída para um arquivo, tem caráter destrutivo; em outras palavras, o comando acima escreverá por cima do conteúdo de <arquivo3>. Se, ao invés disto, você redirecionar com o símbolo ">>", a saída será adicionada a <arquivo3>, ao invés de escrever por cima de seu conteúdo.

more - Permite fazer a paginação de arquivos ou da entrada padrão. O comando more pode ser usado como comando para leitura de arquivos que ocupem mais de uma tela. Quando toda a tela é ocupada, o more efetua uma pausa e permite que você pressione Enter para continuar avançando o número de páginas. Para sair do more pressione q.

Sintaxe: more <arquivo>

Onde:

<arquivo> é o arquivo que será paginado.

O more somente permite avançar o conteúdo do arquivo linha por linha, para um melhor controle de paginação, use o comando less.

less - Permite fazer a paginação de arquivos ou da entrada padrão. O comando less pode ser usado como comando para leitura de arquivos que ocupem mais de uma tela. Quando toda a tela é ocupada, o less efetua uma pausa (semelhante ao more) e permite que você pressione Seta para Cima e Seta para Baixo ou PgUP/PgDown para fazer o rolamento da página. Para sair do less pressione q.

Sintaxe: less <arquivo>

Onde:

<Arquivo> é o arquivo que será paginado.

Para visualizar diretamente arquivos texto compactados pelo utilitário gzip (arquivos .gz), use o comando zless.

Exemplos:

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

```
# cat /etc/passwd | less
```

Comandos para localização de arquivos

find - Procura por arquivos/diretórios no disco. find pode procurar arquivos através de sua data de modificação, tamanho, etc através do uso de opções. 'find', ao contrário de outros programas, usa opções longas através de um "-".

Sintaxe: find < diretório > [opções/expressão]

Onde:

diretório Inicia a procura neste diretório, percorrendo seu sub-diretórios.

Opções/expressão

- name [expressão] Procura pelo nome [expressão] nos nomes de arquivos e diretórios processados.
- depth Processa os sub-diretórios primeiro antes de processar os arquivos do diretório principal.
- maxdepth [num] Faz a procura até [num] sub-diretórios dentro do diretório que está sendo pesquisado.
- mindepth [num] Não faz nenhuma procura em diretórios menores que [num] níveis.
- size [num] Procura por arquivos que tiverem o tamanho [num]. [num] pode ser antecedido de "+" ou "-" para especificar um arquivo maior ou menor que [num].

Exemplo:

```
# find / -name grep
```

Procura no diretório raiz e sub-diretórios um arquivo/diretório chamado grep

```
# find / -name grep -maxdepth 3
```

Procura no diretório raiz e sub-diretórios até o 3o. nível, um arquivo/diretório chamado grep

```
find / -size +1000k
```

Procura no diretório atual e sub-diretórios um arquivo com tamanho maior que 1000 kbytes (1Mbyte).

grep - Procura em um ou mais arquivos por linhas que contém um padrão de busca (expressão regular simples ou estendida).

Sintaxe: grep [opções] <padrão> <arquivos>

Opções:

- e <expr> Procura pela expressão regular expr.
- n Exibe o número de linha que contém padrão.
- c Exibe apenas o número de ocorrências, do padrão de busca, no arquivo.
- i Não diferencia maiúsculas de minúsculas na procura.
- l Exibe os nomes de arquivos que contém padrão.

Obs.: Recomenda-se que o padrão esteja entre apóstrofos (‘), pois alguns caracteres (notadamente \$, *, [,], (,) e \) têm significado especial para o shell e podem ser interpretados erroneamente.

Exemplos:

```
# grep conf /etc/*
```

```
# grep -n conf /etc/*
```

```
# grep -c conf /etc/*
```

```
# grep -i conf /etc/*
```

```
# grep -l conf /etc/*
```

whereis - Localiza o arquivo binário, o código-fonte e a página do manual para um comando.

Sintaxe: whereis [opções] <comando>

Opção:

- b Localiza apenas arquivos binários.
- s Localiza apenas códigos-fonte.
- m Localiza apenas páginas de manual.

Exemplos:

```
# whereis rpm  
rpm: /bin/rpm /usr/include/rpm /usr/man/man8/rpm.8
```

```
# whereis -b rpm  
rpm: /bin/rpm
```

```
# whereis -s rpm  
rpm: /usr /include /rpm
```

```
# whereis -m rpm  
rpm: /usr /man /man8 /rpm.8.
```

Não retorna nenhuma informação se o comando especificado não for encontrado.

locate - Localiza arquivos a partir de um banco de dados. Esse banco de dados deve ser atualizado periodicamente, com o comando updatedb (executado pelo superusuário root).

Sintaxe: locate <padrão>

Exemplo:

```
# locate *.doc  
Localiza os arquivos que terminam com a extensão *.doc
```

which - Procura por um comando em diretórios e na variável de ambiente PATH.

Sintaxe: which <comando>

Exemplo:

```
# which clear
```

Comandos de arquivamento, compactação e descompactação

tar - Armazena ou extrai vários arquivos e diretórios dentro de um único arquivo ou dispositivo.

Sintaxe: tar [opções] <arquivos_ou_diretórios>

Opções:

- c Cria um novo arquivo .tar e adiciona a ele os arquivos especificados.
- x Retira os arquivos agrupados no arquivo .tar.
- f Indica que o destino é um arquivo em disco e não uma fita magnética.
- t Lista o conteúdo do arquivo .tar.
- v Exibe o nome de cada arquivo processado.

Exemplos:

```
# cd /home
# mkdir doc1
# cd doc1
# touch arq11.doc arq12.doc arq13.doc
# mkdir doc2
# cd doc2
# touch arq21.doc arq22.doc arq23.doc
# cd ..
# cd ..
# tar -cvf documentos.tar doc1
# rm -ri doc1
# tar -tvf documentos.tar
# tar -xvf documentos.tar
# tar -cvzf textos.tar.gz doc1
# rm -ri doc1
# tar -xvzf textos.tar.gz doc1

# mount /dev/fd0
# tar -cvf /dev/fd0 -1440 -M arquivo.tar
# cd /mnt/fd0
# tar -xvf arquivo.tar -C /diretório
```

gzip - Compacta um ou mais arquivos

Sintaxe: gzip [opções] <arquivos>

Opções:

- d Descompacta o arquivo.
- h Exibe uma mensagem de ajuda.

gunzip - Descompacta arquivos compactados pelos comandos gzip e compress. Utiliza as mesmas opções de gzip.

Sintaxe: gunzip [opções] <arquivos>

Exemplo:

```
# gunzip documentos.gz
```

Comandos de ajuda

man - Exibe uma página do manual interno do LINUX, para um dado comando ou recurso (isto é, qualquer utilitário do sistema que não seja comando, por exemplo, uma função de biblioteca). É como um "help" interno ao sistema.

Sintaxe : man <comando>

onde "comando" e o nome do comando ou recurso que se deseja obter a ajuda.

Exemplo:

```
# man ls  
# man less
```

info - Exibe informações de um comando do sistema.

Sintaxe: info <comando>

Exemplo:

```
# info
```

```
# info ls
```

Para sair do info, pressione <Q>.

Comandos Diversos

Date - Permite ver/modificar a Data e Hora do Sistema. Você precisa estar como usuário root para modificar a data e hora.

Sintaxe

```
# date MesDiaHoraMinuto[Ano]
```

Onde:

MesDiaHoraMinuto[Ano] São respectivamente os números do mês, dia, hora e minutos sem espaços. Opcionalmente você pode especificar o Ano (com 2 ou 4 dígitos).

Sintaxe

```
# date +[FORMATO] - Define o formato da listagem que será usada pelo comando date.
```

Os seguintes formatos são os mais usados:

%d - Dia do Mês (00-31).

%d - Mês do Ano (00-12).

%y - Ano (dois dígitos).

%Y - Ano (quatro dígitos).

%H - Hora (00-24).

%T - Formato de 24 horas completo (hh:mm:ss).

Para ver a data atual digite: date

Se quiser mudar a data para 25/12 e a hora para 08:15 digite: date 12250815

Para mostrar somente a data no formato dia/mês/ano: date +%d/%m/%Y

df - Mostra o espaço livre/ocupado de cada partição.

Sintaxe: df [*opções*]

Opção

-h Mostra o espaço livre/ocupado em KB, MB, GB ao invés de blocos.

Exemplos:

```
# df
```

```
# df -h
```

free - Mostra detalhes sobre a utilização da memória RAM do sistema.

Sintaxe: free [opções]

Opções:

- m Mostra o resultado em Mbytes.
- o Oculta a linha de buffers.
- t Mostra uma linha contendo o total.
- s [num] Mostra a utilização da memória a cada [num] segundos.

O free é uma interface ao arquivo /proc/meminfo.

head - Mostra as linhas iniciais de um arquivo texto.

Sintaxe: head [opções]

Opções

- n [número] Mostra o [número] de linhas do início do arquivo. Caso não for especificado, o head mostra as 10 primeiras linhas.

Exemplos:

```
# head teste.txt
```

```
# head -n 20 teste.txt
```

tail - Mostra as linhas finais de um arquivo texto.

Sintaxe: tail [opções]

Opções:

- n [número] Mostra o [número] de linhas do final do arquivo.

Exemplos:

```
# tail teste.txt
```

```
# tail -n 20 teste.txt
```

uptime - Mostra o tempo de execução do sistema desde que o computador foi ligado.

Sintaxe: uptime

Exemplo:

```
# uptime
```

su - Permite o usuário mudar sua identidade para outro usuário sem fazer o logout. Útil para executar um programa ou comando como root sem ter que abandonar a seção atual.

Sintaxe: su [usuário]

Onde: *usuário* é o nome do usuário que deseja usar para acessar o sistema. Se não digitado, é assumido o usuário root.

Será pedida a senha do superusuário para autenticação. Digite exit quando desejar retornar a identificação de usuário anterior.

who - Mostra quem está atualmente conectado no computador. Este comando lista os nomes de usuários que estão conectados em seu computador, o terminal e data da conexão.

Sintaxe: who [opções]

Opções:

- H Mostra o cabeçalho das colunas.
- i, -u Mostra o tempo que o usuário está parado em Horas:Minutos.
- q Mostra o total de usuários conectados aos terminais.

clear - Limpa a tela e posiciona o cursor no canto superior esquerdo do vídeo.

Sintaxe: clear

shutdown - Desliga/reinicia o computador imediatamente ou após determinado tempo (programável) de forma segura. Todos os usuários do sistema são avisados que o computador será desligado . Este comando somente pode ser executado pelo usuário root ou usuário autorizado no arquivo /etc/shutdown.allow.

Sintaxe: shutdown [opções] [hora] [mensagem]

hora Momento que o computador será desligado. Você pode usar HH:MM para definir a hora e minuto, MM para definir minutos, +SS para definir após quantos minutos, ou now para imediatamente (equivalente a +0).

O shutdown criará o arquivo /etc/nologin para não permitir que novos usuários façam login no sistema (com excessão do root).

Este arquivo é removido caso a execução do shutdown seja cancelada (opção -c) ou após o sistema ser reiniciado.

mensagem Mensagem que será mostrada a todos os usuários alertando sobre o reinício/desligamento do sistema.

Opções:

- h Inicia o processo para desligamento do computador.
- r Reinicia o sistema.
- c Cancela a execução do shutdown. Você pode acrescentar uma mensagem avisando aos usuários sobre o fato.

O shutdown envia uma mensagem a todos os usuários do sistema alertando sobre o desligamento durante os 15 minutos restantes e assim permite que finalizem suas tarefas. Após isto, o shutdown muda o nível de execução através do comando init para 0 (desligamento), 1 (modo monousuário), 6 (reinicialização). É recomendado utilizar o símbolo "&" no final da linha de comando para que o shutdown seja executado em segundo plano.

Quando restarem apenas 5 minutos para o reinício/desligamento do sistema, o programa login será desativado, impedindo a entrada de novos usuários no sistema.

O programa shutdown pode ser chamado pelo init através do pressionamento da combinação das teclas de reinicialização CTRL+ALT+DEL alterando-se o arquivo /etc/inittab. Isto permite que somente os usuários autorizados (ou o root) possam reinicializar o sistema.

Exemplos:

```
# shutdown -h now  
Desligar o computador imediatamente.
```

```
# shutdown -r now  
Reinicia o computador imediatamente.
```

```
# shutdown 19:00 A manutenção do servidor será iniciada às 19:00  
Faz o computador entrar em modo monousuário (init 1) às 19:00 enviando a mensagem "A  
manutenção do servidor será iniciada às 19:00" a todos os usuários conectados ao sistema.
```

```
# shutdown -r 15:00 O sistema será reiniciado às 15:00 horas  
Faz o computador ser reiniciado (init 6) às 15:00 horas enviando a mensagem "O sistema será  
reiniciado às 15:00 horas" a todos os usuários conectados ao sistema.
```

```
# shutdown -r 20  
Faz o sistema ser reiniciado após 20 minutos.
```

```
# shutdown -c  
Cancela a execução do shutdown.
```

GERENCIAMENTO DE USUÁRIOS

O linux é multiusuário e possui ferramentas para gerenciamento dos usuários e seus privilégios de acesso a arquivos e diretórios.

useradd ou adduser - Adiciona usuários aos sistema.

Sintaxe: `useradd <usuário> [opções]` ou `adduser <usuário> [opções]`

Opção:

`-d <dir_home>` Diretório home do usuário que está sendo criado.
`-c <comentário>` Comentário.
`-s <programa>` Programa que o usuário utilizará ao entrar no sistema (normalmente um shell).

Exemplos:

```
# adduser user1
# useradd user2
# adduser user3 -d /home/temp -c "linus torvalds" -s /bin/bash
```

passwd - Define uma senha para um usuário.

Sintaxe: `passwd <usuário>`

Exemplo:

```
# passwd user1
```

finger - Exibe informações sobre usuários locais ou remotos.

Sintaxe: `finger [opções] [usuário]`

Opção

`-l` Saída em formato detalhado.
`-s` Saída em formato simples.

Exemplos:

```
# finger root
# finger user1
# finger user3
```

userdel - Elimina um usuário do sistema.

Sintaxe: userdel <opção> <usuário>

Opção:

-r Remove todos os arquivos do usuário, incluindo o seu diretório home.

Exemplo:

```
# userdel -r user1  
# userdel user2
```

users - Exibe os usuários ativos do sistema.

Sintaxe: users

Exemplo:

```
# users
```

w - Exibe os usuários conectados ao sistema e o que estão executando.

Sintaxe: w [opções]

Opção

-h Não mostra o cabeçalho na saída.
-l Saída em formato detalhado.
-s Saída em formato simples.

Exemplo:

```
# w
```

GERENCIAMENTO DE PRIVILÉGIOS

O gerenciamento de privilégios permite ao administrador do sistema definir políticas para acesso dos usuários e grupos aos arquivos, diretórios e programas executáveis do sistema.

chmod - Altera permissões de acesso a arquivos.

Sintaxe: `chmod [opção] <modo_arquivo> <arquivo>`

Opção

-R Recursivo. Muda o modo de acesso de todos os arquivos e subdiretórios abaixo do especificado.

É definido na forma `<dono> <grupo> <outros>`.

`<modo_arquivo>`

`<dono>`

r - representa o número 4.

w - representa o número 2.

x - representa o número 1.

`<grupo>`

r - representa o número 4.

w - representa o número 2.

x - representa o número 1.

`<outros>`

r - representa o número 4.

w - representa o número 2.

x - representa o número 1.

Permissão de leitura (r).

Permissão de escrita (w).

Permissão de execução (x).

Permissão de leitura e escrita (rw).

Permissão de leitura e execução (rx).

Permissão de escrita e execução (wx).

Permissão de leitura, escrita e execução (rwx).

Nenhuma permissão.

Exemplos:

```
# chmod 750 teste
```

Permissão de leitura, escrita e execução para o dono, leitura e execução para o grupo e nenhuma permissão aos outros usuários.

```
# chmod 644 teste
```

Permissão de leitura, escrita o dono, leitura para o grupo e leitura aos outros usuários.

```
# chmod 777 /home/samba/share
```

Permissão de leitura, escrita e execução para todos os usuários.

chown - Muda o dono de um arquivo.

Sintaxe: `chown [opção] <novo_dono[.novo_grupo]> <arquivo>`

Opção

-R Recursivo. Muda o grupo de todos os arquivos e subdiretórios abaixo do especificado.

Exemplos:

```
# chown claudio monografia.doc
```

Só muda o proprietário do arquivo.

```
# chown claudio. *
```

Mudam o proprietário e o grupo de todos os arquivos do diretório corrente.

GERENCIAMENTO DE PROCESSOS

Todo sistema operacional fornece uma interface para que os usuários tenham algum controle sobre os programas que este está executando. No linux ocorre algo similar.

<Ctrl>+<C> - Aborta um processo. Finaliza um processo.

<Ctrl>+<Z> - Suspende um processo. O processo passa a ser executado em background.

& - Executa um comando (processo) em segundo plano (background).

Sintaxe: **Comando &**

Exemplos:

```
# vi &  
# find / -name dmesg &
```

jobs - Exibe os jobs em execução pelo shell.

Sintaxe: **jobs [opções]**

Opções

- l Exibe o nome e o número de cada processo.
- s Exibe o nome de cada processo.
- p Exibe o número de cada processo.

Exemplo:

```
# jobs  
# jobs -l
```

bg - Coloca um processo em segundo plano (background).

Sintaxe: **bg [%id]**

Opção

%id Número do job a ser colocado em segundo plano.

Exemplo:

```
# vi  
<ctrl> + <Z>  
[1] + stopped vi
```

```
# bg %1  
[1] + vi
```

fg - Coloca um processo em primeiro plano (foreground).

Sintaxe: fg [%id]

Opção

%id Número do job a ser colocado em primeiro plano.

Exemplo:

```
# vi &  
[1] xxx (xxx é o número do processo)  
[1] + stopped (tty output) vi
```

```
# fg %1
```

ps - Exibe informações sobre os processos ativos.

Sintaxe: ps [opções]

Opção

- a Exibe também informações de outros usuários.
- u Exibe o nome do usuário e a hora de início do processo.
- x Exibe também os processos não associados a um terminal de controle.

Exemplo:

```
# ps -aux
```

kill - Finaliza um processo pelo PID.

Sintaxe: kill [opções][sinal]<PID_processo>

Opção

- n Sinal aplicado ao processo.
- l Lista todos os nomes e números de sinais.

Exemplo:

```
# kill -9 213  
encerra o processo 213.
```

nohup - Continua a execução de um comando mesmo que o usuário que o iniciou saia do sistema.

Sintaxe: nohup <comando>

Exemplo:

```
# nohup find / -name xinitrc &
```

nice - Executa um processo com uma prioridade diferente do padrão.

Sintaxe: nice [opções]<prioridade><comando>

Opção

-help	Exibe uma mensagem de ajuda.
<prioridade>	Valor a ser adicionado à prioridade do processo. Quanto maior este valor, menor será a prioridade de escalonamento do processo. Os valores-padrão do nice variam entre as várias versões do Unix. No Linux variam de -20 a +19, sendo -10 o valor padrão.

Exemplo:

```
# nice -4 find / -name *.log
```

renice - Modifica a prioridade de um processo em execução. Pode ser aplicado a um processo, usuário ou grupo de usuários.

Sintaxe: renice <prioridade>[opções]

<prioridade> - Valor que deve ser atribuído à prioridade do processo.

Opção:

-p <número>	Processo.
-u <usuário>	Nome de usuário.

Exemplo:

```
# renice +15 10570  
Modifica a prioridade corrente do processo 10570 para +15.
```

```
# renice -1 -p 26786
```

Modifica a prioridade corrente do processo 26786 para -1.

```
# renice +4 -u rubem
```

Modifica para +4 a prioridade de todos os processos do usuário rubem.