APRESENTAÇÃO

A partir da minha experiência com treinamentos para as certificações LPIC-1 e LPIC-2, recomendo a leitura e prática do material GuiaFocaLinux que segue dentro perspectiva de atrair novos usuários do sistema, explicando funcionalidades triviais de um Sistema Operacional GNU/Linux .

OBJETIVO

Com o objetivo de instrução e apresentação sobre o que é o sistema GNU/Linux, suas distribuições e como operá-lo. Mostrar alguns aspectos de funcionamento para sua operação, como ler, criar, e modificar arquivos e configurações do sistema operacional.
A instrução é voltada para profissionais mesmo com pouco ou nenhuma experiência com o Sistema GNU/Linux. Ensina uma introdução a operação do sistema Linux de uma forma funcional e simples.

O conteúdo abordado faz parte de uma rotina básica de qualquer ambiente, seja um servidor Web, File Server, Proxy, etc. O conteúdo também é abordado como requisitos mínimos para a Certificação LPI Linux Essentials; Que é o programa de certificação de entrada, conhecimento fundamental para aqueles interessados em obter uma certificação profissional, ou operar um sistema Linux. Mais informações sobre certificações LPI podem ser obtidas em lpi.org.

DA TEORIA À PRÁTICA!

Com objetivo de apresentar os conceitos mais abrangentes sobre o Linux e Software Livre para o novato; De uma maneira que seja capaz de realizar trabalhos com a linha de comando e conhecer as principais ferramentas e configurações.
O material está baseado de acordo com os tópicos exigidos na LPI, abordando operação de qualquer distribuição GNU/Linux, como Debian, CentOS, RedHat, Ubuntu, etc..

A prática do comandos e configurações são fundamentais para o aprendizado ***não tendo como pré-requisito uma Máquina Servidor.*** O Conteúdo pode ser praticado em qualquer máquina que contenha um sistema Linux, como uma máquina virtual, instância em cloud, um desktop, ou notebook.

Recomendo utilizar a distribuições matrizes como Debian (www.debian.org) e CentOS (www.centos.org);

Espero que o material seja útil para os interessados; Que corresponda com a proposta de **"Introdução, e fundamentos de operação"**. Todas as ***Dúvidas***, ***Sugestões*** e ***Críticas*** para melhoria do conteúdo podem ser enviadas para: *w.luis.araujo@gmail.com*

Além deste material recomendo outras literaturas e artigos e cursos disponíveis por outros autores conceituados na área de tecnologia; As indicações estão referenciados em ***Fonte***.

Atenciosamente,
Washington Luís Araújo.

CONTEÚDO

w.luis.araujo@gmail.com
wluisaraujo.github.io

w.luis.araujo@gmail.com
wluisaraujo.github.io

Introdução ao Sistema Operacional GNU/Linux

Sistema Operacional

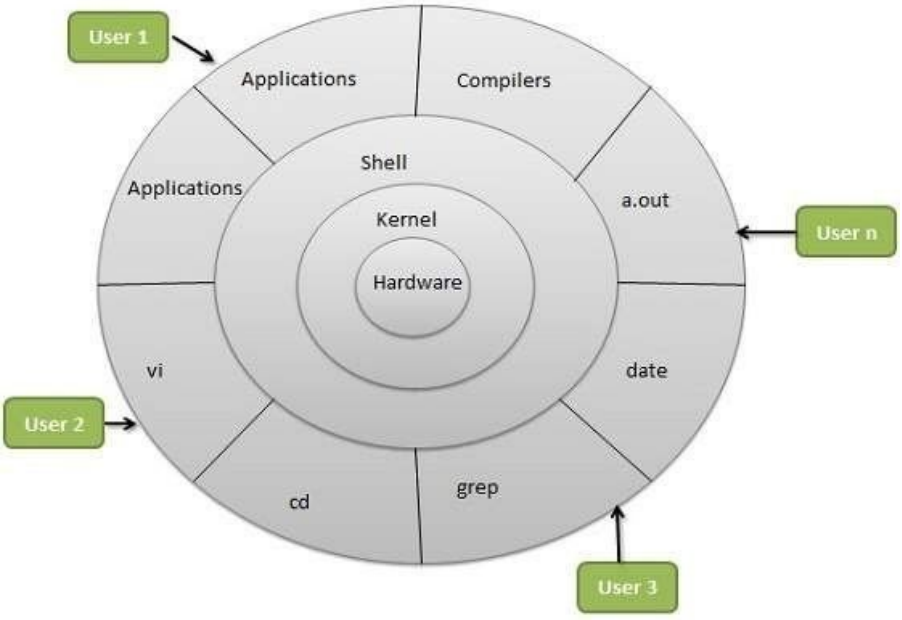
O Sistema Operacional é o conjunto de programas que fazem a interface do usuário e seus programas com o computador. Ele é responsável pelo gerenciamento de recursos e periféricos (como memória, discos, arquivos, impressoras, CD-ROMs, etc.), interpretação de mensagens e a execução de programas. No `Linux' o Kernel mais o conjunto de ferramentas GNU compõem o Sistema Operacional. O kernel (que é a base principal de um sistema operacional), poderá ser construído de acordo com a configuração do seu computador e dos periféricos que possui.

O que é Kernel

“Kernel é o núcleo do Sistema Operacional Linux”

O Sistema Operacional é o conjunto de programas que fazem a interface do usuário e seus programas (Lógico) com o computador (Físico). Ele é responsável pelo gerenciamento de recursos e periféricos (como memória, discos, arquivos, impressoras, CD-ROMs, etc.), interpretação de mensagens e a execução de programas.

No Linux o Kernel é sistema operacional. Quando o Kernel é executado mais o conjunto de ferramentas GNU compõem o Sistema Operacional. O kernel (que é a base principal de um sistema operacional), poderá ser construído de acordo com a configuração do seu computador e dos periféricos que possui.



w.luis.araujo@gmail.com
wluisaraujo.github.io

A origem do Kernel Linux

O Linux é um sistema operacional 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), sob licença GPL, o aviso de copyright do kernel feito por Linus descreve detalhadamente isto e mesmo ele não pode fechar o sistema para que seja usado apenas comercialmente.

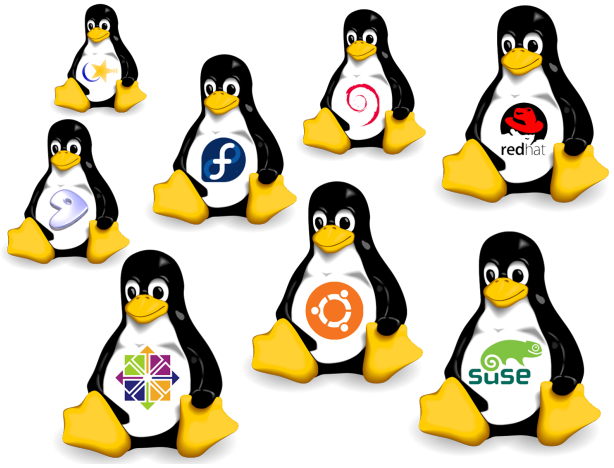
Isto quer dizer que você não precisa pagar nada para usar o Linux, e não é crime fazer cópias para instalar em outros computadores.
Ser um sistema de código aberto permite e incentiva para modificações e melhorias para fins de segurança, estabilidade e velocidade, e adaptar ou construir (desenvolver) novos recursos ao sistema.
O código fonte aberto permite que qualquer pessoa veja como o sistema funciona (útil para aprendizado), corrigir algum problema ou fazer 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.

Algumas Características do Linux

É livre 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.
Também recebe apoio para seu desenvolvimento de grandes empresas como IBM, Sun, HP, etc;
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;
Utiliza permissões de acesso a arquivos, diretórios e programas em execução na memória RAM;
Proteção entre processos executados na memória RAM;
Suporte a mais de 63 terminais virtuais (consoles);
Modularização - O `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);
Suporte nativo a rede e tecnologias avançadas como: balanceamento de carga, ips alias, failover, vlans, bridge, trunking, OSPF, BGP;

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 32MB de RAM);
Suporte nativo a múltiplas CPUs, assim processadores como Dual Core, Core Duo, Athlon Duo, Quad Core tem seu poder de processamento integralmente aproveitado, tanto em 32 ou 64 bits;
Suporte nativo a dispositivos SATA, PATA, Fiber Channel;
Suporte virtualização, onde o `Linux' se destaca como plataforma preferida para execução de múltiplos sistemas operacionais com performance e segurança;
O ***GNU/Linux*** é distribuído livremente e licenciado de acordo com os termos da GPL;
Suporte completo e nativo a diversos dispositivos de comunicação via infravermelho, Bluetooth, Firewire, USB. Basta conectar e o seu dispositivo é automaticamente reconhecido;
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;
Os sistemas de arquivos usados pelo `GNU/Linux' (`Ext2', `Ext3',`Ext4',`reiserfs', `xfs', `jfs') organiza os arquivos de forma inteligente evitando a fragmentação e fazendo-o um poderoso sistema para aplicações multi-usuárias exigentes e gravações intensivas;

O que é Distribuição Linux



É o conjunto de núcleo do sistema mais programas e recursos. Além de conter o Kernel Linux e programas GNU, uma distribuição Linux normalmente agrega outros recursos para tornar sua utilização mais simples “Um Sistema Operacional Completo”. Além de oferecerem um conjunto completo de aplicativos prontos para uso, as distribuições mais populares podem atualizar e instalar novos programas automaticamente. Esse recurso é chamado gestão de pacotes. O gestor de pacotes da distribuição elimina o risco de instalar um programa incompatível ou mal intencionado.

Outra vantagem das distribuições é seu custo. Um usuário experiente pode copiar e instalar ***legalmente*** a distribuição sem precisar pagar por isso. Existem distribuições pagas, mas que pouco diferem daquelas sem custo no que diz respeito a facilidade e recursos. As principais distribuições Linux são:
• ***Debian***. Umas das características do Debian é seu sistema de gestão de pacotes, o dpkg, apt, aptitude. Os pacotes do dpkg tem o sufixo .deb. Instruções para cópia e instalação.

• ***CentOS***. É uma versão gratuita da distribuição comercial **Red Hat Enterprise Linux (RHEL)**. Seu sistema de pacotes chama-se RPM.

Existem muitas outras distribuições importantes além dessas duas, como Ubuntu, Fedora, Linux Mint, openSUSE, etc. Apesar de cada distribuição tem suas peculiaridades, um usuário com alguma experiência em Linux será capaz de trabalhar com todas elas. Muitas utilizam o mesmo sistema de gestão de pacotes. Por exemplo, as distribuições Debian, Ubuntu e Linux Mint utilizam o dpkg (essas duas últimas foram criadas a partir da Debian). Já a Red Hat, CentOS e Fedora utilizam RPM. Outras, como a OpenSUSE e Slackware, tˆem sistemas de gestão de pacotes menos comuns, mas semelhantes ao dpkg e o RPM em sua finalidade.

Operações básicas em Sistemas Linux

Arquivos

É onde gravamos nossos dados. Um arquivo pode conter um texto feito por nós, uma música, programa, planilha, etc.
Cada arquivo deve ser identificado por um ***nome***, assim ele pode ser encontrado facilmente quando desejar usá-lo. Se estiver fazendo um trabalho de história, nada melhor que salvá-lo com o nome **história**.
“ Um arquivo pode ser binário ou texto “

O GNU/Linux é Case Sensitive ou seja, ele diferencia letras maiúsculas e minúsculas nos arquivos. Esta regra também é válido para os comandos e diretórios.

Prefira, sempre que possível, usar letras minúsculas para identificar seus arquivos, pois quase todos os comandos do sistema estão em minúsculas.

Um arquivo oculto no é identificado por um **"."** no início do nome (por exemplo, `.bashrc'). Arquivos ocultos não aparecem em listagens normais de diretórios, deve ser usado o comando **“ls -a”** para também listar arquivos ocultos.

```
~$ ls -a /etc/skel/
.  ..  .bash_logout  .bashrc  .profile  examples.desktop
~$
```

Extensão de arquivos

A extensão serve para identificar o tipo do arquivo. A extensão são as letras após um **"."** no nome de um arquivo, explicando melhor:

relatório.txt	- O **.txt** indica que o conteúdo é um arquivo texto.
script.sh	- Arquivo de Script (interpretado por /bin/sh).
system.log	- Registro de algum programa no sistema.
arquivo.gz	- Arquivo compactado pelo utilitário gzip.
index.html	- Página de Internet (formato Hypertexto).

A extensão de um arquivo também ajuda a saber o que precisamos fazer para abri-lo.
Por exemplo: O arquivo ***relatorio.txt*** é um texto simples. Já o arquivo ***index.html*** contém uma página de Internet e precisaremos de um navegador para poder visualiza-lo (como o lynx, Firefox ou o Konqueror).

A extensão (na maioria dos casos) não é requerida pelo sistema operacional GNU/Linux, mas é conveniente o seu uso para determinarmos facilmente o tipo de arquivo e que programa precisaremos usar para abri-lo.

Em caso de dúvida sobre qual formato é o arquivo, utilize o comando **file <arquivo>** para identificar o seu formato.

```
~$ file /etc/init.d/cron
/etc/init.d/cron: POSIX shell script, ASCII text executable
~$
```

Tamanho de arquivos

A unidade de medida padrão nos computadores é o bit. A um conjunto de 8 bits nós chamamos de byte. Cada arquivo/diretório possui um tamanho, que indica o espaço que ele ocupa no disco e isto é medido em bytes. O byte representa uma letra.
Assim, se você criar um arquivo vazio e escrever o nome ***Linux*** e salvar o arquivo, este terá o tamanho de 5 bytes. Espaços em branco e novas linhas também ocupam bytes.

Além do byte existem as medidas Kbytes, Mbytes, Gbytes.
Os prefixos K (quilo), M (mega), G (giga), T (tera) etc. Da origem matemática.
O "K" significa multiplicar por 10^3, o "M" por 10^6, e assim por diante.

Esta letras servem para facilitar a leitura em arquivos de grande tamanho. Um arquivo de 1K é a mesma coisa de um arquivo de 1024 bytes. Uma forma que pode inicialmente lhe ajudar a lembrar: K vem de Kilo que é igual a 1000 - 1Kilo é igual a 1000 gramas certo?

Da mesma forma 1Mb (ou 1M) é igual a um arquivo de 1024K ou 1.048.576 bytes 1Gb (ou 1G) é igual a um arquivo de 1024Mb ou 1048576Kb ou 1.073.741.824 bytes (1 Gb é igual a 1.073.741.824 bytes, são muitosnúmeros!). Deu pra notar que é mais fácil escrever e entender como 1Gb do que 1.073.741.824 bytes.

A lista completa em ordem progressiva das unidades de medida é a seguinte:

Símbolo	10^	2^	Nome
K	3	10	Quilo
M	6	20	Mega
G	9	30	Giga
T	12	40	Tera
P	15	50	Peta
E	18	60	Eta
Z	21	70	Zetta
Y	24	80	Yotta

Arquivo texto e binário

Quanto ao tipo, um arquivo pode ser de texto ou binário:

Texto - Seu conteúdo é compreendido pelas pessoas (humanamente legível). Um arquivo texto pode ser uma carta, um script, um programa de computador escrito pelo programador, arquivo de configuração, etc.

Binário - Seu conteúdo somente pode ser entendido por computadores (interpretadores de linguagem lógica). Contém caracteres incompreensíveis para pessoas normais.

Um arquivo binário é gerado através de um arquivo de programa (digitado pela pessoa que o criou, o programador) através de um processo chamado de compilação. Compilação é basicamente a conversão de um programa em linguagem humana para a linguagem de máquina.

Diretório

Diretório é o local utilizado para armazenar conjuntos arquivos para melhor organização e localização. O diretório, como o arquivo, também é "***Case Sensitive***" (diretório /teste é completamente diferente do diretório /Teste).

Não podem existir dois arquivos com o mesmo nome em um diretório, ou um sub-diretório com um mesmo nome de um arquivo em um mesmo diretório.

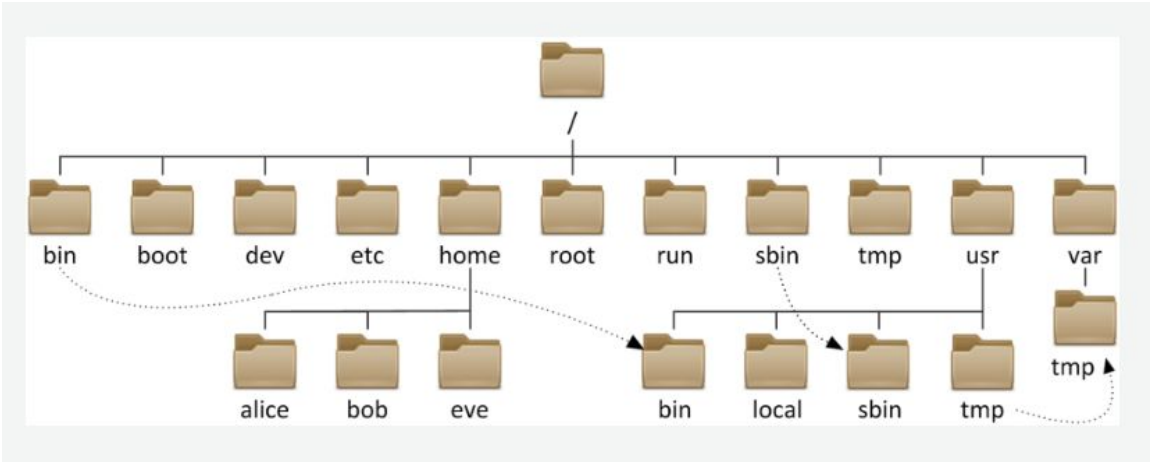
Um diretório nos sistemas Linux/UNIX são especificados por uma **" / "** e não uma **" \ "** como é feito no DOS.

Diretório Raíz

Este é o diretório principal do sistema. Dentro dele estão todos os diretórios do sistema. O diretório Raíz é representado por uma **" /** ", assim se você digitar o comando “cd **/** “ você estará acessando este diretório.

Nele estão localizados outros diretórios como o “/bin, /sbin, /usr, /usr/local, /mnt, /tmp, /var, /home, etc”. Estes são chamados de sub-diretórios pois estão dentro do diretório " / ".

A estrutura de diretórios e subdiretórios pode ser identificada da seguinte maneira:



A estrutura de diretórios também é chamada de “Árvore de Diretórios” porque é parecida com uma árvore de cabeça para baixo. Cada diretório do sistema tem seus respectivos arquivos que são armazenados conforme regras definidas pela **FHS** (FileSystem Hierarchy Standard - Hierarquia Padrão do Sistema de Arquivos), definindo que tipo de arquivo deve ser armazenado em cada diretório.

Diretório atual

É o diretório em que nos encontramos no momento. Você pode digitar “**pwd**” para verificar qual é seu diretório atual. Ou verificar a variável **$PWD,** utilize os comandos.

pwd ou **printenv PWD** ou **echo $PWD**

w.luis.araujo@gmail.com
wluisaraujo.github.io

```
/var/log$ pwd
/var/log
/var/log$ printenv PWD
/var/log
/var/log$ echo $PWD
/var/log
/var/log$
```

O diretório atual também é identificado por um "." (ponto). O comando " **$ ls .** " pode ser usado para listar seus arquivos (é claro que isto é desnecessário porque se não digitar nenhum diretório, o comando "ls" lista o conteúdo do diretório atual).

Diretório home

Também chamado de diretório de usuário. Em sistemas GNU/Linux cada usuário (inclusive o root) possui seu próprio diretório onde poderá armazenar seus programas e arquivos pessoais.

Este diretório está localizado em **/home/[login]** ; Neste caso se o seu login for "joao" o seu diretório home será ***/home/joao***. O diretório home também é identificado por um " **~** " (til), você pode digitar tanto o comando ***"ls /home/joao"*** como **" ls ~ "** para listar os arquivos de seu diretório home.

O diretório home do usuário root (na maioria das distribuições GNU/Linux) está localizado em /root.

Diretório Superior

O diretório superior (Upper Directory) é identificado por **" .. "** (2 pontos). Caso estiver no diretório /usr/local e quiser listar os arquivos do diretório /usr você pode digitar, **"ls .."** Este recurso também pode ser usado para copiar, mover arquivos/diretórios, etc.

Diretório Anterior

O diretório anterior é identificado por **" - "** ; É útil para retornar ao último diretório usado.

Se estiver no diretório /usr/local e digitar cd /lib, você pode retornar facilmente para o diretório /usr/local usando **" cd - "**.

```
/var/log$ cd /usr/local/
/usr/local$ pwd
/usr/local
/usr/local$ cd /lib
/lib$ pwd
/lib
/lib$ cd -
/usr/local
/usr/local$ pwd
/usr/local
/usr/local$
```


w.luis.araujo@gmail.com
wluisaraujo.github.io

Caminhos absoluto e relativo

Caminhos, são os diretórios que teremos que percorrer até chegar no arquivo ou diretório que que procuramos.
Se desejar ver o arquivo /var/log/messages você tem duas opções:

1. Mudar o diretório padrão para /var/log com o comando “cd /var/log” e usar o comando “cat messages”.

2. Usar o comando "cat" especificando o caminho completo na estrutura de diretórios e o nome de arquivo "cat /var/log/messages".

As duas soluções acima permitem que você veja o arquivo "***messages***". A diferença entre as duas é a seguinte:

Na primeira, você muda o diretório padrão para /var/log (confira digitando pwd) e depois o comando cat messages. Você pode ver os arquivos de /var/log com o comando " ls /var/log ".

Na segunda, é digitado o caminho completo para o comando "cat /var/log/messages"' localizar o arquivo messages:

Neste caso, você continuará no diretório padrão (confira digitando pwd). Digitando `ls', os arquivos do diretório atual serão listados.

O caminho de diretórios é necessário para dizer ao sistema operacional onde encontrar um arquivo na "árvore" de diretórios.

Estrutura básica de diretórios do Sistema Linux

O sistema GNU/Linux possui a seguinte estrutura básica de diretórios organizados segundo o FHS (Filesystem Hierarchy Standard):

/bin - Contém arquivos programas do sistema que são usados com freqüência pelos usuários.
/boot - Contém arquivos necessários para a inicialização do sistema.
/cdrom - Ponto de montagem da unidade de CD-ROM.
/media - Ponto de montagem de dispositivos diversos do sistema (rede, pen-drives, CD-ROM em distribuições mais novas).
/dev - Contém arquivos usados para acessar dispositivos (periféricos) existentes no computador.

/etc - Arquivos de configuração de seu computador local.
/home - Diretórios contendo 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.
/sys - 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 pessoal para o 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 maior parte de seus programas. Normalmente acessível somente como leitura.
/var - Contém maior parte dos arquivos que são gravados com freqüência pelos programas do sistema, e-mails, spool de impressora, cache, 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 diretório

“cd” - Para mudar de diretório

Este guia 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:

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. Também chamado de opção extensa. O comando `ls --all' é equivalente a `ls -a'.

Pode ser usado tanto "-" como "--", mas há casos em que somente "-" ou "--" está disponível.

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

Se você digitar: `ls /usr/share/doc/copyright', “/usr/share/doc/copyright' será o parâmetro passado ao comando `ls', neste caso queremos que ele liste os arquivos do diretório “/usr/share/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. No `GNU/Linux', você tem a possibilidade de criar comandos personalizados usando outros comandos mais simples (isto será visto mais adiante).

Os comandos se encaixam em duas categorias: Comandos Internos e Comandos Externos.

Por exemplo: `"ls -la /usr/share/doc"', `ls' é o comando, `-la' é a opção passada ao comando, e ``/usr/share/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. 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 são: “cd, exit, echo, bg, fg, source, help”

Comandos Externos

São comandos que estão localizados no disco. Os comandos são procurados no disco usando o ordem do `PATH' e executados assim que encontrados.

Prompt de Comando (CLI)

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 por uma "#" (tralha), e o aviso de comando de usuários é identificado pelo símbolo "$". Isto é padrão em sistemas `UNIX'.

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:

- Pressione a tecla `Back Space' (<--) 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, ash, csh, tcsh, sh,' etc. Entre eles o mais usado 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 completação 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 completação de nomes encontre mais do que uma expressão que satisfaça a pesquisa, ou nenhuma, é emitido um beep.

Se você apertar novamente a tecla TAB imediatamente depois do beep, o interpretador de comandos irá listar as diversas possibilidades que satisfazem a pesquisa, para que você possa escolher a que lhe interessa. A completação de nomes funciona sem problemas para comandos internos.

Exemplo: `ech' (pressione `TAB'). `ls /vm'(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-usuáriao usando os "terminais virtuais". Um terminal virtual é uma segunda sessão 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>. Para voltar ao modo gráfico, pressione `CTRL'+`ALT'+ <F7>.

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>.

Operações em BASH

Login

Login é a entrada no sistema quando você digita seu nome e senha.

Logout

Logout é a saída do sistema. A saída do sistema é feita pelos comandos `logout', `exit', `CTRL'+`D', ou quando o sistema é reiniciado ou desligado.

Caracteres Coringas

Curingas (ou referência global) é um recurso usado para especificar um ou mais arquivos ou diretórios do sistema de uma só vez. Este é um recurso permite que você faça a filtragem do que será listado, copiado, apagado, etc. São usados 4 tipos de curingas 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 uma faixa de caracteres de um arquivo/diretório. Padrão pode ser:

[a-z][0-9] - Faz referência a caracteres de `a' até `z' seguido de um caracter de `0' até `9'.

[a,z][1,0]' - Faz a referência aos caracteres `a' e `z' seguido de um caracter `1' ou `0' naquela posição.

[a-z,1,0]' - Faz referência a intervalo de caracteres de `a' até `z' ou `1' ou `0' 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 precedida por um `^', faz referência a qualquer caracter exceto o da expressão. Por exemplo `[^abc]' faz referência a qualquer caracter exceto `a', `b' e `c'.

“{padrões}” - Expande e gera strings para pesquisa de padrões de um arquivo/diretório.

X{ab,01}' - Faz referência a seqüencia de caracteres `Xa” ou `X01'

X{a-z,10}' Faz referencia a seqüencia de caracteres X`a-z' e `X10'.

O que diferencia este método de expansão dos demais é que a existência do arquivo/diretório é opcional para geração do resultado. Isto é útil para a criação de diretórios. Lembrando que os 4 tipos de curingas ("*", "?", "[]", "{}") podem ser usados juntos. Para entender melhor vamos a prática:

Vamos dizer que tenha 5 arquivo no diretório `/usr/teste':

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

Caso deseje listar todos os arquivos do diretório `/usr/teste' você pode usar o coringa "*" para especificar todos os arquivos do diretório:
`cd /usr/teste' e `ls *' ou `ls /usr/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 `teste10.txt' este também seria listado.
3. Usando o comando `ls teste[1-3].txt', que 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 ? específica qualquer caracter naquela posição e [] específica 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 números de 4 e 5 naquela posição e terminem com qualquer extensão.

Existem muitas outras 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 curingas 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.

Armazenamento

Discos e Partições

Este capítulo traz explicações de como manipular discos rígidos e partições no sistema `GNU/Linux' e como acessar seus discos de CD-ROM E partições `DOS', `Windows 9X/XP/Vista/Seven' no `GNU/Linux'.

Partições

São divisões existentes no disco rígido que marcam onde começa onde termina um sistema de arquivos. As partições nos permitem usar mais de um sistema operacional no mesmo computador (como o `GNU/Linux',`Windows' e `DOS'), ou dividir o disco rígido em uma ou mais partes para ser usado por um único sistema operacional ou até mesmo por diferentes arquiteturas (32 e 64 bits).

Sistemas de Arquivos

Para formatar um disco e aplicar um sistema de arquivos para serem usados no `GNU/Linux' use o comando:

$ mkfs.ext2 [/dev/sde1]'

Em alguns sistemas você deve usar `mke2fs' no lugar de `mkfs.ext2'. A opção `-c' faz com que o `mkfs.ext2' procure por blocos danificados no pen-drive.
Note que o nome de dispositivo que é conectado varia de acordo com o sistema e quantidade de discos rígidos que sua máquina possui portanto tenha _ATENCÃO_ para não formatar o dispositivo incorreto (que pode ser justamente seu disco disco rígido principal). Para maior segurança, ao identificar o pen-drive, digite `dmesg' ao conectar o pen-drive para visualizar o dispositivo correto ou fique atento as mensagens do console que mostrará o dispositivo que foi associado ao pen-drive.
OBSERVAÇÃO: Este comando cria um sistema de arquivos ext2 no pen-drive e permite usar características como permissões de acesso e outras. Isto também faz com que o pen-drive NÃO possa ser lido pelo `DOS/Windows'.

Formatando com compatibilidade com o Windows

A formatação de pen-drives para serem usados no `Windows' é feita usando o comando `mkfs.msdos' que é geralmente incluído no pacote `dosfstools'. O `mkfs.msdos' permite tanto a criação de sistemas de arquivos FAT16 ou FAT32.

$ mkfs.msdos [opções] [dispositivo]

dispositivo: Pen-drive que será formatado. Normalmente `/dev/sdb1' (dependendo do dispositivo detectado via comando `dmesg').
opções: -F [num] Especifica o tipo de FAT que será usado na formatação. Podem ser usados os valores 12 (para formatação usando FAT12, limitado a 12MB), 16 (para

formatação usando FAT16, limitado a 2Gb) e 32 (para formatação FAT32, limitado a 128Gb).
-n [nome] Atribui o `[nome]' de volume ao dispositivo.
-c Faz uma pesquisa por bad blocks antes da criação do sistema de arquivos no dispositivo. Os setores defeituosos encontrados serão automaticamente marcados para não serem utilizadas.

Note que não se deve montar o pen-drive / disquete para formata-lo.

Segue abaixo exemplos de como formatar seu pen-drive `mkfs.msdos':

$ mkfs.msdos /dev/sdc1
Formata o pen-drive no terceiro dispositivo SCSI Genérico, como FAT32 e usando os valores padrões.
$ mkfs.msdos -F 16 /dev/sdc1
Faz a mesma coisa que o acima, mas formata o pen-drive como FAT16.
$ mkfs.msdos -n teste -F 16 /dev/sdc1
Formata o pen-drive no terceiro dispositivo SCSI genérico, como FAT16 e cria o nome de volume `teste'.

Pontos de Montagem

O `GNU/Linux' acessa as partições existente em seus discos rígidos e disquetes através de diretórios. Os diretórios que são usados para acessar (montar) partições são chamados de Pontos de Montagem. Para detalhes sobre montagem de partições, veja Seção 4.5, `Montando (acessando) uma partição de disco'.

No `DOS' cada letra de unidade (C:, D:, E:) identifica uma partição de disco, no `GNU/Linux' os pontos de montagem fazem parte da grande estrutura do sistema de arquivos raiz.

Identificação de discos e partições em sistemas Linux

No `GNU/Linux', os dispositivos existentes em seu computador (com discos rígidos, pen-drives, flash, disquetes,, tela, portas de impressora, modem, etc) são identificados por um arquivo referente a este dispositivo no diretório `/dev'.

A identificação de discos rígidos no `GNU/Linux' é feita da seguinte forma:

```
/dev/sda1
|   | ||
|   | ||_Número que identifica o número da partição no disco rígido.
|   | |
|   | |_Letra que identifica o disco rígido (a=primeiro, b=segundo, etc...).
|   |
|   |_Sigla que identifica o tipo do disco rígido (sd=SATA/SCSI, md=RAID, xt=MFM).
|
|_Diretório onde são armazenados os dispositivos existentes no sistema.
```

Abaixo algumas identificações de discos e partições em sistemas Linux:

```
/dev/fd0        - `Primeira unidade de disquetes'.
/dev/sda        - `Primeiro disco rígido na primeira controladora SATA ou SCSI'.
/dev/sda1_      - `Primeira partição do primeiro disco rígido SATA
```

/dev/sdb_ - `Segundo disco rígido na primeira controladora SATA ou SCSI'.
/dev/sr0 - `Primeiro CD-ROM SATA ou SCSI'.
/dev/sr1_ - `Segundo CD-ROM SATA ou SCSI'.
/dev/xda - `Primeiro disco rígido XT'.
/dev/xdb_ - `Segundo disco rígido XT'.

As letras de identificação de discos rígidos podem ir além de `sdb', por exemplo, caso utilize pen-drives, memória flash, as unidades serão detectadas como `sdc', `sdd' e assim por diante.

É importante entender como os discos e partições são identificados no sistema, pois será necessário usar os parâmetros corretos para monta-los.

Montando (acessando) uma partição de disco

Você pode acessar uma partição de disco usando o comando `mount'.

$ mount [_dispositivo_] [_ponto de montagem_] [_opções_]'

Onde:
dispositivo: Identificação da unidade de disco/partição que deseja acessar (como `/dev/hda1' (disco rígido) ou `/dev/fd0' (primeira unidade de disquetes).
ponto de montagem: Diretório de onde a _unidade de disco/partição_ será acessado. O diretório deve estar vazio para montagem de um sistema de arquivo. Normalmente é usado o diretório `/mnt' para armazenamento de pontos de montagem temporários.

-t [tipo] Tipo do sistema de arquivos usado pelo dispositivo. São aceitos os sistemas de arquivos:
ext2 Para partições `GNU/Linux' usando o Extended File System versão 2
ext3 Para partições `GNU/Linux' usando o Extended File System versão 3, com suporte a journaling.
ext4 Para partições `GNU/Linux' usando o Extended File System versão 4, com suporte a journaling.
reiserfs Para partições reiserfs, com suporte a journaling.
xfs Para partições xfs, com suporte a journaling.
vfat Para partições `Windows 95' que utilizam nomes extensos de arquivos e diretórios.
msdo Para partições `DOS' normais.
iso9660 Para montar unidades de `CD-ROM'. É o padrão.

Na maioria das vezes, caso o sistema de arquivos não seja especificado, o `mount' utilizará a auto-detecção e montará a partição usando o sistema de arquivos correto. Para mais detalhes sobre opções usadas com cada sistema de arquivos, veja a página de manual mount.

-r Caso for especificada, monta a partição somente para leitura.

-w Caso for especificada, monta a partição como leitura/gravação. É o padrão.

Existem muitas outras opções que podem ser usadas com o comando `mount', mas aqui procurei somente mostrar o básico para "montar" seus discos e partições no `GNU/Linux' (para mais opções, veja a página d manual do `mount'). Caso você digitar `mount' sem parâmetros, serão mostrados os sistemas de arquivos atualmente montados no sistema.
Esta mesma listagem pode ser vista em `/etc/mtab'. A remontagem de partição também é muito útil, especialmente após reparos nos sistema de arquivos do disco rígido.

É necessário permissões de root para montar partições, a não ser que tenha especificado a opção `user' no arquivo `/etc/fstab'

Exemplo de Montagem:

Montar uma partição Windows (vfat) de `/dev/sda1' em `/mnt' somente para leitura: `
$ mount /dev/sda1 /mnt -r -t vfat'

Montar um pen-drive detectado em `/dev/sdc1' em `/mnt':
$mount /dev/sdc1 /mnt -t vfat'

Montar uma partição DOS localizada em um segundo disco rígido`/dev/hdb1' em `/mnt':
$mount /dev/hdb1 /mnt -t msdos'.

Remontar a partição raíz como somente leitura:
$mount -o remount,ro /'

Remontar a partição raíz como _leitura/gravação_ (a opção -n é usada porque o `mount' não conseguirá atualizar o arquivo `/etc/mtab' devido ao sistema de arquivos `/' estar montado como somente leitura atualmente:
$`mount -n -o remount,rw /'.

fstab

O arquivo `/etc/fstab' permite que as partições do sistema sejam montadas facilmente especificando somente o dispositivo ou o ponto de montagem. Este arquivo contém parâmetros sobre as partições que são lidos pelo comando `mount'. Cada linha deste arquivo contém a partição que desejamos montar, o ponto de montagem, o sistema de arquivos usado pela partição e outras opções. `fstab' tem a seguinte forma:

Sistema_de_arquivos	Ponto_de_Montagem	Tipo	Opções	dump	ordem
/dev/sda1	/	ext3	defaults	0	1
/dev/sda2	/boot	ext3	defaults	0	2
/dev/sda3	/dos	msdos	defaults,noauto,rw	0	0
/dev/hdg	/cdrom	iso9660	defaults,noauto	0	0

Onde: Sistema de Arquivos é a partição que deseja montar.

Ponto de montagem

Diretório do `GNU/Linux' onde a partição montada será acessada.

Tipo: Tipo de sistema de arquivos usado na partição que será montada.
Para partições `GNU/Linux' use _ext3_, _reiserfs_, _xfs_ (de acordo com o tipo de partição selecionada durante a formatação), para partições `DOS' (sem nomes extensos de arquivos) use _msdos_, para partições `Win 95' (com suporte a nomes extensos de arquivos) use _vfat_, para unidades de CD-ROM use _iso9660_.

Opções: Especifica as opções usadas com o sistema de arquivos. Abaixo, algumas opções de montagem para ext2/3/4 (a lista completa pode ser encontrada na página de manual do `mount'):
defaults' Utiliza valores padrões de montagem.
noauto' Não monta os sistemas de arquivos durante a inicialização (útil para CD-ROMS e disquetes).
ro' Monta como somente leitura.
user' Permite que usuários montem o sistema de arquivos(não recomendado por motivos de segurança).
sync' É recomendado para uso com discos removíveis (disquetes, zip drives, nfs, etc) para que os dados sejam gravados imediatamente na unidade (caso não seja usada, você deve usar o comando
dump Especifica a frequência de backup feita com o programa `dump' no sistema de arquivos. 0 desativa o backup.
ordem Define a ordem que os sistemas de arquivos serão verificados na inicialização do sistema. Se usar 0, o sistema de arquivos não é verificado. O sistema de arquivos raíz que deverá ser verificado primeiro é o raíz "/" .

Após configurar o `/etc/fstab', basta digitar o comando “mount -a” para montar todos os dispositivos indicados no arquivo.

Desmontando uma partição de disco

Utilize o comando `umount' para desmontar um sistema de arquivos que foi montado com o `mount'. Você deve ter permissões de root para desmontar uma partição.

$ umount [_dispositivo_/_ponto de montagem_]'

Você pode tanto usar `umount /dev/sda1' como `umount /mnt' para desmontar um sistema de arquivos `/dev/sda1' montado em `/mnt'.

Observação: O comando `umount' executa o `sync' automaticamente no momento da desmontagem, para garantir que todos os dados ainda em memória RAM sejam salvos.

Execução de Programas

Para executar um comando, é necessário que ele tenha permissões de execução e que esteja no caminho de procura de comandos $PATH
No aviso de comando _#_(root) ou _$_(usuário), digite o nome do comando e tecle Enter. O programa/comando é executado e receberá um número de identificação (chamado de PID - Process Identification), este número é útil para identificar o processo no sistema e assim ter um controle sobre sua execução.

Todo o programa executado no `GNU/Linux' roda sob o controle das permissões de acesso.
Exemplos de comandos: `ls', `df', `pwd'.

path

Path é o caminho de procura dos arquivos/comandos executáveis. O path (caminho) é armazenado na variável de ambiente `PATH'. Você pode ver o conteúdo desta variável com o comando `echo $PATH'.

Por exemplo, o caminho `/usr/local/bin:/usr/bin:/bin:/usr/bin/X11' significa que se você digitar o comando `ls', o interpretador de comandos iniciará a procura do programa `ls' no diretório `/usr/local/bin', caso não encontre o arquivo no diretório `/usr/local/bin' ele inicia a procura em `/usr/bin', até que encontre o arquivo procurado.

Caso o interpretador de comandos chegue até o último diretório do path e não encontre o arquivo/comando digitado, é mostrada a seguinte mensagem:

$ `bash: ls: command not found' (comando não encontrado).

O caminho de diretórios vem configurado na instalação do Linux, mas pode ser alterado no arquivo `/etc/profile'. Caso deseje alterar o caminho para todos os usuários, este arquivo é o melhor lugar, pois ele é lido por todos os usuários no momento do login.

Caso um arquivo/comando não esteja localizado em nenhum dos diretórios do path, você deve executa-lo usando um `./' na frente do comando.

Se deseja alterar o `path' para um único usuário, modifique o arquivo `.bash_profile' em seu diretório de usuário (home).

OBSERVAÇÃO: Por motivos de segurança, não inclua o diretório atual `$PWD' no `path'.

Tipos de Execução de comandos/programas

Um programa pode ser executado de duas formas:

1. `Primeiro Plano' - Também chamado de _foreground_. Quando você deve esperar o término da execução de um programa para executar um novo comando. Somente é mostrado o aviso de comando após o término de execução do comando/programa.

2. `Segundo Plano' - Também chamado de _background_. Quando você não precisa esperar o término da execução de um programa para executar um novo comando. Após iniciar um programa em _background_, é mostrado um número PID (identificação do Processo) e o aviso de comando é novamente mostrado, permitindo o uso normal do sistema.

O programa executado em background continua sendo executado internamente. Após ser concluído, o sistema retorna uma mensagem de pronto acompanhado do número PID do processo que terminou.
Para iniciar um programa em `primeiro plano', basta digitar seu nome normalmente. Para iniciar um programa em `segundo plano', acrescente o caracter `"&"' após o final do comando.

OBS: Mesmo que um usuário execute um programa em segundo plano e saia do sistema, o programa continuará sendo executado até que seja concluído ou finalizado pelo usuário que iniciou a execução (ou pelo usuário root).

Exemplo: `find / -name boot.b &'

O comando será executado em segundo plano e deixará o sistema livre para outras tarefas. Após o comando `find' terminar, será mostrada uma mensagem.

Executando programas em seqüência

Os comandos podem ser executados em seqüência (um após o término do outro) se os separarmos com ";".
Por exemplo: `echo primeiro; echo segundo; echo terceiro'

Comandos podem ser executados em seqüência (um após o término do primeiro comando ter sido com sucesso) se os separarmos com "&&".
Por exemplo: `apt-get update && apt-get upgrade

Comandos podem ser executados em seqüência (um após o término do primeiro comando ter sido com falha) se os separarmos com "||".
Por exemplo: mkdir /backup || echo “/backup falhou”

ps

Algumas vezes é útil ver quais processos estão sendo executados no computador. O comando `ps' faz isto, e também nos mostra qual usuário executou o programa, hora que o processo foi iniciado, etc.

$ps [_opções_]'

Onde [opções]:

a Mostra os processos criados por você e de outros usuários do sistema.

x Mostra processos que não são controlados pelo terminal.
u Mostra o nome de usuário que iniciou o processo e hora em que o processo foi iniciado.
m Mostra a memória ocupada por cada processo em execução.
f Mostra a árvore de execução de comandos (comandos que são chamados por outros comandos).
e Mostra variáveis de ambiente no momento da inicialização do processo.
w Mostra a continuação da linha atual na próxima linha ao invés de cortar o restante que não couber na tela.

As opções acima podem ser combinadas para resultar em uma listagem mais completa. Você também pode usar pipes "|" para `filtrar' a saída do comando `ps'. Para detalhes, veja Seção 12.5, `| (pipe)'.

Ao contrário de outros comandos, o comando `ps' não precisa do hífen "-" para especificar os comandos. Isto porque ele não utiliza opções longas e não usa parâmetros.

Exemplos: `ps', `ps ax|grep inetd', `ps auxf', `ps auxw'.

top

Mostra os programas em execução ativos, parados, tempo usado na CPU, detalhes sobre o uso da memória RAM, Swap, disponibilidade para execução de programas no sistema, etc. É um programa que continua em execução mostrando continuamente os processos que estão rodando em seu computador e os recursos utilizados por eles.
Para sair do `top', pressione a tecla `q'.

`top [_opções_]'

-d [tempo] Atualiza a tela após o [tempo] (em segundos).
-s Diz ao `top' para ser executado em modo seguro.
-i Inicia o `top' ignorando o tempo de processos zumbis.
-c Mostra a linha de comando ao invés do nome do programa.

A ajuda sobre o `top' pode ser obtida dentro do programa pressionando a tecla `h' ou pela página de manual (`man top').
Abaixo algumas teclas úteis:

espaço' Atualiza imediatamente a tela.
CTRL'+`L' Apaga e atualiza a tela.
h' Mostra a tela de ajuda do programa. É mostrado todas as teclas que podem ser usadas com o `top'.
i Ignora o tempo ocioso de processos zumbis.
q Sai do programa.
k Finaliza um processo - semelhante ao comando `kill'. Você será perguntado pelo número de identificação do processo (PID).

Este comando não estará disponível caso esteja usando o `top' com a opção `-s'.

n Muda o número de linhas mostradas na tela. Se 0 for especificado, será usada toda a tela para listagem de processos.

Interrompendo a execução de um processo

Para cancelar a execução de algum processo `rodando em primeiro plano', basta pressionar as teclas `CTRL'+`C'. A execução do programa será cancelada e será mostrado o aviso de comando. Você também pode usar o comando kill para interromper um processo sendo executado.

Parando momentaneamente a execução de um processo

Para parar a execução de um processo rodando em primeiro plano, basta pressionar as teclas `CTRL'+`Z'. O programa em execução será pausado e será mostrado o número de seu job e o aviso de comando.

Para retornar a execução de um comando pausado fg ou bg.

O programa permanece na memória no ponto de processamento em que parou quando ele é interrompido. Você pode usar outros comandos ou rodar outros programas enquanto o programa atual está interrompido.

jobs

O comando `jobs' mostra os processos que estão parados ou rodando em segundo plano_. Processos em segundo plano são iniciados usando o símbolo `"&"' no final da linha de comando.

Exemplo:
$ gzip -9 /var/log/messages &

jobs

O número de identificação de cada processo parado ou em segundo plano (job), é usado com os comando.

Um processo interrompido pode ser finalizado usando-se o comando `kill %[num]', onde `[num]' é o número do processo obtido pelo comando `jobs'.

fg

Permite fazer um programa rodando em segundo plano ou parado, rodar em primeiro plano. Você deve usar o comando `jobs' para pegar o número do processo rodando em segundo plano ou interrompida, este número será passado ao comando `fg' para ativa-lo em primeiro plano.

fg [_número_]:

Onde _número_ é o número obtido através do comando `jobs'.

Caso seja usado sem parâmetros, o `fg' utilizará o último programa interrompido (o maior número obtido com o comando `jobs').
Exemplo:
$ fg 1'

bg

Permite fazer um programa rodando em primeiro plano ou parado, rodar em segundo plano. Para fazer um programa em primeiro plano rodar em segundo, é necessário primeiro interromper a execução do comando com `CTRL'+ `Z', será mostrado o número da tarefa interrompida, use este número com o comando `bg' para iniciar a execução do comando em segundo plano.

`bg [_número_]:

Onde: _número_ número do programa obtido com o pressionamento das teclas `CTRL'+`Z' ou através do comando `jobs'.

kill

Permite enviar um sinal a um comando/programa. Caso seja usado sem parâmetros, o `kill' enviará um sinal de término ao processo sendo executado.

kill [_opções_] [_sinal_] [_número_]'
Onde:

número É o número de identificação do processo obtido com o comando `ps'. Também pode ser o número após o sinal de `%' obtido pelo comando `jobs' para matar uma tarefa interrompida.

sinal Sinal que será enviado ao processo. Se omitido usa `-15' SIGTERM como padrão.
-9 Envia um sinal de destruição ao processo ou programa. Ele é terminado imediatamente sem chances de salvar os dados ou apagar os arquivos temporários criados por ele.
Você precisa ser o dono do processo ou o usuário root para termina-lo ou destruí-lo. Você pode verificar se o processo foi finalizado através do comando `ps'.
-l Lista todos os sinais disponíveis.
Exemplo:
$ kill 500
$ kill -9 500
$ kill %1

killall

Permite finalizar processos através do nome.

killall [_opções_] [_sinal_] [_processo_]'

Onde:

processo	Nome do processo que deseja finalizar
sinal	Sinal que será enviado ao processo (pode ser obtido usando a opção `-i').
-i	Pede confirmação sobre a finalização do processo.
-l	Lista o nome de todos os sinais conhecidos.
-q	Ignora a existência do processo.
-v	Retorna se o sinal foi enviado com sucesso ao processo.
-w	Finaliza a execução do `killall' somente após finalizar todos os processos.

Os tipos de sinais aceitos pelo `GNU/Linux' são explicados em detalhes ‘Sinais do Sistema'.

Exemplo:
$ killall -HUP inetd

killall5

Envia um sinal de finalização para todos os processos sendo executados.
Exemplo:
$ killall5 -9

pgrep

Procura um processo por seu nome, e retorna o numero de PID do processo
Exemplo:
$ pgrep cron

pkill

Procura um processo por seu nome, e encerra os processos encontrados
Exemplo:
$ pkill cron

Sinais do Sistema

Retirado da página de manual `signal'. O `GNU/Linux' suporta os sinais listados abaixo. Alguns números de sinais são dependentes de arquitetura.

Sinal	Valor	Ação	Comentário
HUP	1	A	Travamento detectado no terminal de controle ou finalização do processo controlado
INT	2	A	Interrupção através do teclado
QUIT	3	C	Sair através do teclado
ILL	4	C	Instrução Ilegal
ABRT	6	C	Sinal de abortar enviado pela função abort
FPE	8	C	Exceção de ponto Flutuante
KILL	9	AEF	Sinal de destruição do processo
SEGV	11	C	Referência Inválida de memória

PIPE	13	A	Pipe Quebrado: escreveu para o pipe sem leitores
ALRM	14	A	Sinal do Temporizador da chamada do sistema alarm
TERM	15	A	Sinal de Término
USR1	30,10,16	A	Sinal definido pelo usuário 1
USR2	31,12,17	A	Sinal definido pelo usuário 2
CHLD	20,17,18	B	Processo filho parado ou terminado
CONT	19,18,25		Continuar a execução, se interrompido
STOP	17,19,23	DEF	Interromper processo
TSTP	18,20,24	D	Interromper digitação no terminal
TTIN	21,21,26	D	Entrada do terminal para o processo em segundo plano
TTOU	22,22,27	D	Saída do terminal para o processo em segundo plano

As letras da coluna `Ação' tem o seguinte significado:

A' - A ação padrão é terminar o processo.
B' - A ação padrão é ignorar o sinal.
C' - A ação padrão é terminar o processo e mostrar o core.
D' - A ação padrão é parar o processo.
E' - O sinal não pode ser pego.
F' - O sinal não pode ser ignorado.

Sinais não descritos noPOSIX 1 mas descritos na _SUSv2_:

Sinal	Valor	Ação	Comentário
BUS	10,7,10	C	Erro no Barramento (acesso incorreto da memória)
POLL		A	Evento executado em Pool (Sys V). Sinônimo de IO
PROF	27,27,29	A	Tempo expirado do Profiling
SYS	12,-,12	C	Argumento inválido para a rotina (SVID)
TRAP	5	C	Captura do traço/ponto de interrupção
URG	16,23,21	B	Condição Urgente no soquete (4.2 BSD)
VTALRM	26,26,28	A	Alarme virtual do relógio (4.2 BSD)
XCPU	24,24,30	C	Tempo limite da CPU excedido (4.2 BSD)
XFSZ	25,25,31	C	Limite do tamanho de arquivo excedido (4.2 BSD)

Fechando um programa quando não se sabe como sair

Muitas vezes quando se esta iniciando no `GNU/Linux' você pode executar um programa e talvez não saber como fecha-lo. Este capítulo do guia pretende ajuda-lo a resolver este tipo de problema.

Isto pode também ocorrer com programadores que estão construindo seus programas e por algum motivo não implementam uma opção de saída, ou ela não funciona!

Em nosso exemplo vou supor que executamos um programa em desenvolvimento com o nome `contagem' que conta o tempo em segundos a partir do momento que é executado, mas que o programador esqueceu de colocar uma opção de saída. Siga estas dicas para finaliza-lo:

1. Normalmente todos os programas `UNIX' (o `GNU/Linux' também é um Sistema Operacional baseado no `UNIX') podem ser interrompidos com o pressionamento das teclas `<CTRL>' e `<C>'. Tente isto primeiro para finalizar um programa. Isto provavelmente não vai funcionar se estiver usando um Editor de Texto (ele vai entender como um comando de menu). Isto normalmente funciona para comandos que são executados e terminados sem a intervenção do usuário.

Caso isto não der certo, vamos partir para a força! ;-)

2. Mude para um novo console (pressionando `<ALT>' e `<F2>'), e faça o _login_ como usuário _root_.

3. Localize o PID (número de identificação do processo) usando o comando: `ps ax', aparecerão várias linhas cada uma com o número do processo na primeira coluna, e a linha de comando do programa na última coluna. Caso aparecerem vários processos você pode usar `ps ax|grep contagem', neste caso o `grep' fará uma filtragem da saída do comando `ps ax' mostrando somente as linhas que tem a palavra "contagem".

4. Feche o processo usando o comando `kill _PID_', lembre-se de substituir PID pelo número encontrado pelo comando `ps ax' acima.

O comando acima envia um sinal de término de execução para o processo (neste caso o programa `contagem'). O sinal de término mantém a chance do programa salvar seus dados ou apagar os arquivos temporários que criou e então ser finalizado, isto depende do programa.

5. Alterne para o console onde estava executando o programa `contagem' e verifique se ele ainda está em execução. Se ele estiver parado mas o aviso de comando não está disponível, pressione a tecla <ENTER>. Freqüentemente acontece isto com o comando `kill', você finaliza um programa mas o aviso de comando não é mostrado até que se pressione <ENTER>.

6. Caso o programa ainda não foi finalizado, repita o comando `kill' usando a opção -9: `kill -9 PID'. Este comando envia um sinal de DESTRUIÇÃO do processo, fazendo ele terminar "na marra"!

Uma última dica: todos os programas estáveis (todos que acompanham as boas distribuições `GNU/Linux') tem sua opção de saída. Lembre-se que quando finaliza um processo todos os dados do programa em execução podem ser perdidos (principalmente se estiver em um editor de textos), mesmo usando o `kill' sem o parâmetro `-9'.

Procure a opção de saída de um programa consultando o help on line, as páginas de manual, a documentação que acompanha o programa, info pages.

Eliminando caracteres estranhos

As vezes quando um programa `mal comportado' é finalizado ou quando você visualiza um arquivo binário através do comando `cat', é possível que o aviso de comando (prompt) volte com caracteres estranhos.

Para fazer tudo voltar ao normal, basta digitar `reset' e teclar ENTER'. Não se preocupe, o comando `reset' não reiniciará seu computador (como o botão reset do seu computador faz), ele apenas fará tudo voltar ao normal.

Note que enquanto você digitar `reset' aparecerão caracteres estranhos ao invés das letras. Não se preocupe! Basta digitar corretamente e bater `ENTER' e o aviso de comando voltará ao normal.

Comandos para manipulação de Diretório

ls

Lista os arquivos de um diretório.

ls [_opções_] [_caminho/arquivo_] [_caminho1/arquivo1_] ...'

onde:

caminho/arquivo

Diretório/arquivo que será listado.

caminho1/arquivo1_

Outro Diretório/arquivo que será listado. Podem ser feitas várias listagens de uma só vez.

opções

-a, --all Lista todos os arquivos (inclusive os ocultos) de um diretório.

-A, --almost-all Lista todos os arquivos (inclusive os ocultos) de um diretório, exceto o diretório atual e o de nível anterior.

-B, --ignore-backups Não lista arquivos que terminam com ~ (Backup).

--color=PARAM Mostra os arquivos em cores diferentes, conforme o tipo de arquivo. PARAM pode ser: never (Nunca lista em cores) always (Sempre lista em cores conforme o tipo de arquivo), auto (Somente colore a listagem se estiver em um terminal)

-d, --directory Lista os nomes dos diretórios ao invés do conteúdo.

-f Não classifica a listagem.

-F Insere um caracter após arquivos executáveis ('*'), diretórios ('/'), soquete ('='), link simbólico ('@') e pipe ('|'). Seu uso é útil para identificar de forma fácil tipos de arquivos nas listagens de diretórios.

-G, --no-group Oculta a coluna de grupo do arquivo.

-h, --human-readable Mostra o tamanho dos arquivos em Kbytes, Mbytes, Gbytes.

-H Faz o mesmo que `-h', mas usa unidades de 1000 ao invés de 1024 para especificar Kbytes, Mbytes, Gbytes.

-l Usa o formato longo para listagem de arquivos. Lista as permissões, data de modificação, donos, grupos, etc.

-n Usa a identificação de usuário e grupo numérica ao invés dos nomes.

-L, --dereference Lista o arquivo original e não o link referente ao arquivo.

-o Usa a listagem longa sem os donos dos arquivos (mesma coisa que -lG).

-p Mesma coisa que -F, mas não inclui o símbolo '*' em arquivos executáveis. Esta opção é típica de sistemas `Linux'.

-R Lista diretórios e sub-diretórios recursivamente.

Uma listagem feita com o comando `ls -la' normalmente é mostrada da seguinte maneira:

-rwxr-xr-- 1 gleydson user 8192 nov 4 16:00 teste

Abaixo as explicações de cada parte:
-rwxr-xr-- São as permissões de acesso ao arquivo teste. A primeira letra (da esquerda) identifica o tipo do arquivo, se tiver um `d' é um diretório, se tiver um "-" é um arquivo normal.
1 Se for um diretório, mostra a quantidade de sub-diretórios existentes dentro dele. Caso for um arquivo, será 1.
gleydson Nome do dono do arquivo `teste'.
user Nome do grupo que o arquivo `teste' pertence.
8192 Tamanho do arquivo (em bytes).
nov Mês da criação/ última modificação do arquivo.
4 Dia que o arquivo foi criado.
16:00 Hora em que o arquivo foi criado/modificado. Se o arquivo foi criado há mais de um ano, em seu lugar é mostrado o ano da criação do arquivo.
teste Nome do arquivo.

Exemplos do uso do comando `ls':
$ ls' - Lista os arquivos do diretório atual.
$ ls /bin /sbin' - Lista os arquivos do diretório /bin e /sbin
$ ls -la /bin' - Listagem completa (vertical) dos arquivos do diretório /bin inclusive os ocultos.

cd

Entra em um diretório. Você precisa ter a permissão de execução para entrar no diretório.
$ cd [_diretório_]'

Onde: [diretório] é o diretório que deseja entrar.
Exemplos:

cd retornará ao seu diretório de usuário (diretório home)
cd ~ retornará ao seu diretório de usuário (diretório home).
cd / retornará ao diretório raíz.
cd - retornará ao diretório anteriormente acessado.
cd .. sobe um diretório.
cd ../[_diretório_] sobe um diretório e entra imediatamente no próximo (por exemplo, quando você está em `/usr/sbin', você digita `cd ../bin', o comando `cd' retorna um diretório (`/usr')e entra imediatamente no diretório `bin' (`/usr/bin').

pwd

Mostra o nome e caminho do diretório atual. Você pode usar o comando pwd para verificar em qual diretório se encontra (caso seu aviso de comandos não mostre isso).

mkdir

Cria um diretório no sistema. Um diretório é usado para armazenar arquivos de um determinado tipo. O diretório pode ser entendido como uma _pasta_ onde você guarda seus papeis (arquivos). Como uma pessoa organizada, você utilizará uma pasta para guardar cada tipo de documento, da mesma forma você pode criar um diretório `vendas' para guardar seus arquivos relacionados com vendas naquele local.

$ mkdir [_opções_] [caminho/diretório]

onde:

caminho Caminho onde o diretório será criado.
diretório Nome do diretório que será criado.
-p Caso os diretórios dos níveis acima não existam, eles também serão criados.
--verbose Mostra uma mensagem para cada diretório criado. As mensagens de erro serão mostradas mesmo que esta opção não seja usada.
Para criar um novo diretório, você deve ter permissão de gravação.
Por exemplo, para criar um diretório em /tmp com o nome de `teste' que será usado para gravar arquivos de teste, você deve usar o comando "mkdir /tmp/teste"'.

Podem ser criados mais de um diretório com um único comando (mkdir /tmp/teste /tmp/teste1 /tmp/teste2').

rmdir

Remove um diretório do sistema. Este comando faz exatamente o contrário do `mkdir'. O diretório a ser removido deve estar vazio e você deve ter permissão de gravação para removê-lo.

$rmdir [_caminho/diretório_] [_caminho1/diretório1_]'

São opções:
caminho: Caminho do diretório que será removido.
diretório: Nome do diretório que será removido.

É necessário que esteja um nível acima do diretório(s) que será(ão) removido(s). Para remover diretórios que contenham arquivos, use o comando `rm' com a opção `-r' Por exemplo, para remover o diretório `/tmp/teste' você deve estar no diretório `tmp' e executar o comando `rmdir teste'.

Comandos para manipulação de Arquivos

Abaixo, comandos utilizados para manipulação de arquivos.

cat

Mostra o conteúdo de um arquivo binário ou texto.

$ cat [opções] [_diretório/arquivo_] [_diretório1/arquivo1_]'

diretório/arquivo	Localização do arquivo que deseja visualizar o conteúdo.
opções:	
-n, --number	Mostra o número das linhas enquanto o conteúdo do arquivo é mostrado.
-s, --squeeze-blank	Não mostra mais que uma linha em branco entre um parágrafo e outro.
-	Lê a entrada padrão.

O comando `cat' trabalha com arquivos texto. Use o comando `zcat' para ver diretamente arquivos compactados com `gzip'.
Exemplo:

$ cat /usr/doc/copyright/GPL'

tac

Mostra o conteúdo de um arquivo binário ou texto (como o `cat') só que em ordem inversa.

$tac [opções] [_diretório/arquivo_] [_diretório1/arquivo1_]'

diretório/arquivo	Localização do arquivo que deseja visualizar o conteúdo
opções:	
-s [string]	Usa o [string] como separador de registros.
-	Lê a entrada padrão.

Exemplo:
$ tac /usr/doc/copyright/GPL

rm

Apaga arquivos. Também pode ser usado para apagar diretórios e sub-diretórios vazios ou que contenham arquivos.
$ rm [_opções_][_caminho_][_arquivo/diretório_]

onde:
caminho Localização do arquivo que deseja apagar. Se omitido, assume que o arquivo esteja no diretório atual.
arquivo/diretório Arquivo que será apagado.

opções:
-i, --interactive Pergunta antes de remover, esta é ativada por padrão.
-v, --verbose Mostra os arquivos na medida que são removidos.
-r, --recursive Usado para remover arquivos em sub-diretórios. Esta opção também pode ser usada para remover sub-diretórios.
-f, --force Remove os arquivos sem perguntar.
-- arquivo Remove arquivos/diretórios que contém caracteres especiais. O separador "--" funciona com todos os comandos do shell e permite que os caracteres especiais como "*", "?", "-", etc. sejam interpretados como caracteres comuns.

Use com atenção o comando `rm', uma vez que os arquivos e diretórios forem apagados, eles não poderão ser mais recuperados.

Exemplos:
rm teste.txt - Apaga o arquivo `teste.txt' no diretório atual.
rm *.txt' - Apaga todos os arquivos do diretório atual que terminam com `.txt'.
rm *.txt teste.novo' - Apaga todos os arquivos do diretório atual que terminam com `.txt' e também o arquivo `teste.novo'.
rm -rf /tmp/teste/*' - Apaga todos os arquivos e sub-diretórios do diretório `/tmp/teste' mas mantém o sub-diretório `/tmp/teste'.
rm -rf /tmp/teste' - Apaga todos os arquivos e sub-diretórios do diretório `/tmp/teste', inclusive `/tmp/teste'.
rm -f -- --arquivo--' - Remove o arquivo de nome `--arquivo--'.

cp

Copia arquivos.

$cp [_opções_] [_origem_] [_destino_]'

onde:
origem Arquivo que será copiado. Podem ser especificados mais de um arquivo para ser copiado usando "Curingas"
destino O caminho ou nome de arquivo onde será copiado. Se o destino for um diretório, os arquivos de origem serão copiados para dentro do diretório.
opções

-i, --interactive Pergunta antes de substituir um arquivo existente.
-f, --force Não pergunta, substitui todos os arquivos caso já exista.
-r Copia arquivos dos diretórios e subdiretórios da origem para o destino. É recomendável usar -R ao invés de -r.

-R, --recursive Copia arquivos e subdiretórios (como a opção -r) e também os arquivos especiais FIFO e dispositivos.
-v, --verbose Mostra os arquivos enquanto estão sendo copiados. O comando `cp' copia arquivos da ORIGEM para o DESTINO. Ambos origem e destino terão o mesmo conteúdo após a cópia.

Exemplos:
$cp teste.txt teste1.txt' - Copia o arquivo `teste.txt' para `teste1.txt'.
$cp teste.txt /tmp - Copia o arquivo `teste.txt' para dentro do diretório `/tmp'.
$cp * /tmp - Copia todos os arquivos do diretório atual para `/tmp'.
$cp /bin/* . - Copia todos os arquivos do diretório `/bin' para o diretório em que nos encontramos no momento.
$cp -R /bin /tmp' - Copia o diretório `/bin' e todos os arquivos/sub-diretórios existentes para o diretório `/tmp'.
$cp -R /bin/* /tmp' - Copia todos os arquivos do diretório `/bin' (exceto o diretório`/bin') e todos os arquivos/sub-diretórios existentes dentro dele para `/tmp'.
$cp -R /bin /tmp' - Copia todos os arquivos e o diretório `/bin' para `/tmp'.

mv

Move ou renomeia arquivos e diretórios. O processo é semelhante ao do comando `cp' mas o arquivo de origem é apagado após o término da cópia.

$mv [_opções_] [_origem_] [_destino_]'

Onde:
origem Arquivo/diretório de origem.
destino Local onde será movido ou novo nome do arquivo/diretório.
opções:
-f, --force Substitui o arquivo de destino sem perguntar.
-i, --interactive Pergunta antes de substituir. É o padrão.
-v, --verbose Mostra os arquivos que estão sendo movidos.

O comando `mv' copia um arquivo da _ORIGEM_ para o _DESTINO_ (semelhante ao `cp'), mas após a cópia, o arquivo de _ORIGEM_ é apagado.

Exemplos:
$mv teste.txt teste1.txt'
Muda o nome do arquivo `teste.txt' para `teste1.txt'.

$mv teste.txt /tmp'
Move o arquivo teste.txt para `/tmp'. Lembre-se que o arquivo de origem é apagado após ser movido.

$mv teste.txt teste.new' (supondo que `teste.new' já exista) Copia o arquivo `teste.txt' por cima de `teste.new' e apaga

$teste.txt' após terminar a cópia.

Comandos Diversos

Comandos de uso diversos no sistema.

clear

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

$ clear

date

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

$ date MesDiaHoraMinuto[AnoSegundos]'

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

Onde [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).
* `%m' - Mês do Ano (00-12).
* `%y' - Ano (dois dígitos).
* `%Y' - Ano (quatro dígitos).
* `%H' - Hora (00-24).
* `%I' - Hora (00-12).
* `%M' - Minuto (00-59).
* `%j' - Dia do ano (1-366).
* `%p' - AM/PM (útil se utilizado com %d).
* `%r' - Formato de 12 horas completo (hh:mm:ss AM/PM).
* `%T' - Formato de 24 horas completo (hh:mm:ss).
* `%w' - Dia da semana (0-6).

Para ver a data atual digite: `date'

Exemplo, se quiser mudar a Data para 25/12 e a hora para 08:15 digite:

$date 12250815

Exemplo, para mostrar somente a data no formato dia/mês/ano:
$ date +%d/%m/%Y

df

Mostra o espaço livre/ocupado de cada partição.

$ df [_opções_]'

São opções

-a	Inclui sistemas de arquivos com 0 blocos.
-h, --human-readable	Mostra o espaço livre/ocupado em MB, KB, GB ao invés de blocos.
-H	Idêntico a `-h' mas usa 1000 ao invés de 1024 como unidade de cálculo.
-k	Lista em Kbytes.
-l	Somente lista sistema de arquivos locais.
-m	Lista em Mbytes (equivalente a --block-size=1048576).

Exemplos:
$df'
$df -h
$df -t vfat.

ln

Cria links para arquivos e diretórios no sistema. O link é um mecanismo que faz referência a outro arquivo ou diretório em outra localização. O link em sistemas `GNU/Linux' faz referência reais ao arquivo/diretório podendo ser feita cópia do link (será copiado o arquivo alvo), entrar no diretório (caso o link faça referência a um diretório), etc.

$ ln [_opções_] [_origem_] [_link_]'

Onde:

origem	Diretório ou arquivo de onde será feito o link.
link	Nome do link que será criado.

São Opções:

-s	Cria um link simbólico. Usado para criar ligações com o arquivo/diretório de destino.
-v	Mostra o nome de cada arquivo antes de fazer o link.
-d	Cria um hard link para diretórios. Somente o root pode usar esta opção.

Existem 2 tipos de links, simbólicos e hardlinks.

link simbólico: cria um arquivo especial no disco (do tipo link) que tem como conteúdo o caminho para chegar até o arquivo alvo (isto pode ser verificado pelo tamanho do arquivo do link). Use a opção `-s' para criar links simbólicos.

hardlink: faz referência ao mesmo inodo do arquivo original, desta forma ele será perfeitamente idêntico, inclusive nas permissões de acesso, ao arquivo original.
Ao contrário dos links simbólicos, não é possível fazer um hardlink para um diretório ou fazer referência a arquivos que estejam em partições diferentes.

Observações:
Se for usado o comando `rm' com um link, somente o link será removido.
Se for usado o comando `cp' com um link, o arquivo original será copiado ao invés do link.
Se for usado o comando `mv' com um link, a modificação será feita no link.
Se for usado um comando de visualização (como o `cat'), o arquivo original será visualizado.

Exemplos:
$ln -s /dev/ttyS1 /dev/modem' - Cria o link `/dev/modem' para o arquivo `/dev/ttyS1'.
$ln -s /tmp ~/tmp' - Cria um link `~/tmp' para o diretório /tmp.

du

Mostra o espaço ocupado por arquivos e subdiretórios do diretório atual.

$ du [_opções_]'

São opções:
-a, --all Mostra o espaço ocupado por todos os arquivos.
-b, --bytes Mostra o espaço ocupado em bytes.
-c, --total Faz uma totalização de todo espaço listado.
-D Não conta links simbólicos.
-h, --human Mostra o espaço ocupado em formato legível por humanos (Kb, Mb) ao invés de usar blocos.
-H Como o anterior mas usa 1000 e não 1024 como unidade de cálculo.
-k Mostra o espaço ocupado em Kbytes.
-m Mostra o espaço ocupado em Mbytes.
-S, --separate-dirs Não calcula o espaço ocupado por sub-diretórios.

Exemplo:
$du -h
$du -hc

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 `'-'".

$ find [_diretório_] [_opções/expressão_]'

Onde:

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

São 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.

-mount, -xdev Não faz a pesquisa em sistemas de arquivos diferentes daquele de onde o comando `find' foi executado.

-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]. A opção -size pode ser seguida de:
b Especifica o tamanho em blocos de 512 bytes. É o padrão caso [num] não seja acompanhado de nenhuma letra.

c Especifica o tamanho em bytes.

k Especifica o tamanho em Kbytes.

-type [tipo] Procura por arquivos do [tipo] especificado. Os seguintes tipos são aceitos:
b' - bloco
c' - caracter
d' - diretório
p' - pipe
f' - arquivo regular
l' - link simbólico
s' - sockete
A maior parte dos argumentos numéricos podem ser precedidos por "+" ou "-". Para detalhes sobre outras opções e argumentos, consulte a página de manual.

Exemplo:

$ find / -name grep - Procura no diretório raíz e sub-diretórios um arquivo/diretório chamado `grep'.

$find / -name grep -maxdepth 3 - Procura no diretório raíz 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).

free

Mostra detalhes sobre a utilização da memória RAM do sistema.

$free [_opções_]'

Onde:
São opções:

-b	Mostra o resultado em bytes.
-k	Mostra o resultado em Kbytes.
-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'.

grep

Procura por um texto dentro de um arquivo(s) ou no dispositivo de entrada padrão.

$grep [_expressão_] [_arquivo_] [_opções_]'

Onde:

expressão	Palavra ou frase que será procurada no texto. Se tiver mais de 2 palavras você deve identificá-la com aspas "" caso contrário o `grep' assumirá que a segunda palavra é o arquivo!
arquivo	Arquivo onde será feita a procura.

opções

-A [número]	Mostra o [número] de linhas após a linha encontrada pelo `grep'.
-B [número]	Mostra o [número] de linhas antes da linha encontrada pelo `grep'.
-f [arquivo]	Especifica que o texto que será localizado, está no arquivo [arquivo].
-h, --no-filename	Não mostra os nomes dos arquivos durante a procura.
-i, --ignore-case	Ignora diferença entre maiúsculas e minúsculas no texto procurado e arquivo.

-n, --line-number Mostra o nome de cada linha encontrada pelo `grep'.
-E Ativa o uso de expressões regulares.
-U, --binary Trata o arquivo que será procurado como binário.

Se não for especificado o nome de um arquivo ou se for usado um hífen "-", `grep' procurará a string no dispositivo de entrada padrão. O grep' faz sua pesquisa em arquivos texto. Use o comando `zgrep' para pesquisar diretamente em arquivos compactados com `gzip', os comandos e opções são as mesmas.

Exemplos:
$grep "capitulo" texto.txt', `ps ax|grep inetd',
$grep"capitulo" texto.txt -A 2 -B 2'.

head

Mostra as linhas iniciais de um arquivo texto.

$ head [_opções_]

Onde:
-c [numero] Mostra o [numero] de bytes do inicio do arquivo.
-n [numero] Mostra o [numero] de linhas do inicio do arquivo. Caso não for especificado, o `head' mostra as 10 primeiras linhas.

Exemplos:
$head teste.txt',
$head -n 20 teste.txt'.

nl

Mostra o número de linhas junto com o conteúdo de um arquivo.

$nl [_opções_] [_arquivo_]'

Onde:
opções
-f [opc] Faz a filtragem de saída de acordo com [opc]:
a Numera todas as linhas.
t Não numera linhas vazias.
n Numera linhas vazias.
arquivo/texto Numera somente linhas que contém o [texto].
-v [num] Número inicial (o padrão é 1).
-i [num] Número de linhas adicionadas a cada linha do arquivo (o padrão é

Exemplo:

$ nl /etc/passwd', `nl -i 2 /etc/passwd'.

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' ou `espaço' para continuar avançando no arquivo sendo visualizado. Para sair do `more' pressione `q'.

Exemplo:

$more /var/log/messages

Para visualizar diretamente arquivos texto compactados pelo `gzip' ‘.gz' use o comando `zmore'.

Exemplos:
$ more /var/log/dmesg.gz

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'.

$ 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'

sort

Organiza as linhas de um arquivo texto ou da entrada padrão.

sort [_opções_] [_arquivo_]'

Onde:
arquivo É o nome do arquivo que será organizado. Caso não for especificado, será usado o dispositivo de entrada padrão (normalmente o teclado ou um "|").

opções
-b Ignora linhas em branco.
-d Somente usa letras, dígitos e espaços durante a organização.

-f Ignora a diferença entre maiúsculas e minúsculas.
-r Inverte o resultado da comparação.
-n Caso estiver organizando um campo que contém números, os números serão organizados na ordem aritmética. Por exemplo, se você tiver um arquivo com os números
100
10
50
Usando a opção `-n', o arquivo será organizado desta maneira:
10
50
100
Caso esta opção _não_ for usada com o `sort', ele organizará como uma listagem alfabética (que começam de `a' até `z' e do `0' até `9')
10
100
50
-c Verifica se o arquivo já esta organizado. Caso não estiver, retorna a mensagem "disorder on _arquivo_".
-o _arquivo_ Grava a saída do comando `sort' no _arquivo_.

Abaixo, exemplos de uso do comando `sort':
$`sort `texto.txt'' - Organiza o arquivo `texto.txt' em ordem crescente.
$ sort `texto.txt' -r' - Organiza o conteúdo do arquivo `texto.txt' em ordem decrescente.
$ cat `texto.txt'|sort' - Faz a mesma coisa que o primeiro exemplo, só que neste caso a saída do comando `cat' é redirecionado a entrada padrão do comando `sort'.
$ sort -f `texto.txt'' - Ignora diferenças entre letras maiúsculas e minúsculas durante a organização.

tail

Mostra as linhas finais de um arquivo texto.

`tail [_opções_]'

Onde:
-c [numero] Mostra o [numero] de bytes do final do arquivo.
-n [numero] Mostra o [numero] de linhas do final do arquivo.
-f Mostra continuamente linhas adicionadas no final do arquivo.

Exemplos:
$ tail teste.txt'
$`tail -n 20 teste.txt'.

time

Mede o tempo gasto para executar um processo (programa).

`time [_comando_]'

Onde: _comando_ é o comando/programa que deseja medir o tempo gasto para ser concluído.

Exemplo:
$ `time ls', `time find / -name crontab'.

touch

Muda a data e hora que um arquivo foi criado. Também pode ser usado para criar arquivos vazios. Caso o `touch' seja usado com arquivos que não existam, por padrão ele criará estes arquivos.

touch [_opções_] [_arquivos_]'

Onde: [arquivos] São arquivos que terão sua data/hora modificados.

São opções:

-t MMDDhhmm[ANO.segundos] Usa Mês (MM), Dias (DD), Horas (hh), minutos (mm) e opcionalmente o ANO e segundos para modificação do(s) arquivos ao invés da data e hora atual.
-a, --time=atime Faz o `touch' mudar somente a data e hora do acesso ao arquivo.
-c, --no-create Não cria arquivos vazios, caso os _arquivos_ não existam.
-m, --time=mtime Faz o `touch' mudar somente a data e hora da modificação.
-r [arquivo] Usa as horas no [arquivo] como referência ao invés da hora atual.

Exemplos:

$touch teste' - Cria o arquivo `teste' caso ele não existir.

$touch -t 10011230 teste' - Altera da data e hora do arquivo para 01/10 e 12:30.

$ touch -t 120112301999.30 teste' - Altera da data, hora ano, e segundos do arquivo para 01/12/1999 e 12:30:30.

$touch -t 12011200 *' - Altera a data e hora do arquivo para 01/12 e 12:00.

uptime

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

```
~$ uptime
 15:35:25 up 0 min,  0 users,  load average: 0.52, 0.58, 0.59
~$
```

dmesg

Mostra as mensagens de inicialização do kernel. São mostradas as mensagens da última inicialização do sistema.

$ dmesg | less

echo

Mostra mensagens. Este comando é útil na construção de scripts para mostrar mensagens na tela para o usuário acompanhar sua execução.

$ echo [_mensagem_]'

A opção `-n' pode ser usada para que não ocorra o salto de linha após a mensagem ser mostrada.

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.

$su [_usuário_] [_-c comando_]'

Onde: [usuário] é o nome do usuário que deseja usar para acessar o sistema. Se não digitado, é assumido o usuário `root'. Caso seja especificado _-c comando_, executa o comando sob o usuário specificado.

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

sync

Grava os dados do cache de disco na memória RAM para todos os discos rígidos e flexíveis do sistema. O cache um mecanismo de aceleração que permite que um arquivo seja armazenado na memória ao invés de ser imediatamente gravado no disco, quando o sistema estiver ocioso, o arquivo é gravado para o disco. O `GNU/Linux' procura utilizar toda memória RAM disponível para o cache de programas acelerando seu desempenho de leitura/gravação.

$ sync

O uso do `sync' é útil em disquetes quando gravamos um programa e precisamos que os dados sejam gravados imediatamente para retirar o disquete da unidade. Mas o método recomendado é especificar a opção `sync' durante a montagem da unidade de disquetes.

uname

Retorna o nome e versão do kernel atual.

$ uname

reboot

Reinicia o computador.

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 quando é usada a opção `-a' pelos usuários cadastrados no arquivo `/etc/shutdown.allow' que estejam logados no console virtual do sistema.

$ 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 segundos, 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 reinicio/desligamento do sistema.

São 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 reinicio/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'.

wc

Conta o número de palavras, bytes e linhas em um arquivo ou entrada padrão. Se as opções forem omitidas, o `wc' mostra a quantidade de linhas, palavras, e bytes.

wc [_opções_] [_arquivo_]'

Onde:
arquivo - Arquivo que será verificado pelo comando `wc'.

São opções -
-c, --bytes - Mostra os bytes do arquivo.
-w, --words - Mostra a quantidade de palavras do arquivo.
-l, --lines - Mostra a quantidade de linhas do arquivo.

A ordem da listagem dos parâmetros é única, e modificando a posição das opções não modifica a ordem que os parâmetros são listados.

Exemplo:
$ wc /etc/passwd - Mostra a quantidade de linhas, palas letras (bytes) no arquivo `/etc/passwd'.
$ wc -w /etc/passwd' - Mostra a quantidade de palavras.
$ wc -l /etc/passwd' - Mostra a quantidade de linhas.
$ wc -l -w /etc/passwd' - Mostra a quantidade de linhas e palavras no arquivo `/etc/passwd'.

seq

Imprime uma seqüência de números começando em [primeiro] e terminando em [último], utilizando [incremento] para avançar.

seq [_opções_] [_primeiro_] [_incremento_] [_último_]'

Onde:
primeiro - Número inicial da seqüência.
incremento - Número utilizado para avançar na seqüência.
último - Número final da seqüência.

São opções:

-f, --format=[formato] - Formato de saída dos números da seqüência. Utilize o estilo do printf para ponto flutuante (valor padrão: %g).
-s, --separator=[string] - Usa [string] para separar a seqüência de números (valor padrão: \n).
-w, --equal-width - Insere zeros na frente dos números mantendo a seqüência alinhada.

Observações:
Se [primeiro] ou [incremento] forem omitidos, o valor padrão 1 será utilizado.
Os números recebidos são interpretados como números em ponto flutuante.
[incremento] deve ser positivo se [primeiro] for menor do que o último, e negativo caso contrário.
Quando utilizarmos a opção --format, o argumento deve ser exatamente %e, %f ou %g.
Exemplos:
$seq 0 2 10'
$seq -w 0 10'
$seq -f%f 0 10'
$seq -s", " 0 10'

Comandos de Rede

Este capítulo traz alguns comandos úteis para uso em rede e ambientes multiusuário.

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.

who [_opções_]

São Opções

-H, --heading	Mostra o cabeçalho das colunas.
-b, --boot	Mostra o horário do último boot do sistema.
-d, --dead	Mostra processos mortos no sistema.
-i, -u, --idle	Mostra o tempo que o usuário está parado em Horas:Minutos.
-m, i am	Mostra o nome do computador e usuário associado ao nome. É equivalente a digitar `who i am' ou `who am i'.
-q, --count	Mostra o total de usuários conectados aos terminais.
-r, --runlevel	Mostra o nível de execução atual do sistema e desde quando ele está ativo.
-T, -w, --mesg	Mostra se o usuário pode receber mensagens via `talk' (conversação).
+	O usuário recebe mensagens via talk
-	O usuário não recebe mensagens via talk.
?	Não foi possível determinar o dispositivo de terminal onde o usuário está conectado.

telnet

Permite acesso a um computador remoto. É mostrada uma tela de acesso correspondente ao computador local onde deve ser feita a autenticação do usuário para entrar no sistema. Muito útil, mas deve ser tomado cuidados ao disponibilizar este serviço para evitar riscos de segurança e usado o `ssh' sempre que possível por ser um protocolo criptografado e com recursos avançados de segurança.

telnet [_opções_] [_ip/dns_] [_porta_]'

onde:
[ip/dns] Endereço IP do computador de destino ou nome DNS.

[porta] Porta onde será feita a conexão. Por padrão, a conexão é feita na porta _23_.

São opções
-8 Requisita uma operação binária de 8 bits. Isto força a operação em modo binário para envio e recebimento. Por padrão, `telnet' não usa 8 bits.
-a Tenta um login automático, enviando o nome do usuário lido da variável de ambiente `USER'.
-d Ativa o modo de debug.
-r Ativa a emulação de rlogin.
-l [usuário] Faz a conexão usando [usuário] como nome de usuário.

Exemplo:
$ telnet 192.168.1.1
$ telnet 192.168.1.1 22

finger

Mostra detalhes sobre os usuários de um sistema. Algumas versões do `finger' possuem bugs e podem significar um risco para a segurança do sistema. É recomendado desativar este serviço na máquina local.

finger [_usuário_] [_usuário@host_]

Onde: [usuário] Nome do usuário que deseja obter detalhes do sistema. Se não for digitado o nome de usuário, o sistema mostra detalhes de todos os usuários conectados no momento.
[usuário@host] Nome do usuário e endereço do computador que deseja obter detalhes.
-l Mostra os detalhes de todos os usuários conectados no momento. Entre os detalhes, estão incluídos o _nome do interpretador de comandos_ (shell) do usuário, _diretório home_, _nome do usuário_, _endereço_, etc.
-p Não exibe o conteúdo dos arquivos `.plan' e `.project; Se for usado sem parâmetros, mostra os dados de todos os usuários conectados atualmente ao seu sistema.

Exemplo:
$finger', `finger root'.

ftp

Permite a transferência de arquivos do computador remoto/local e vice versa. O file transfer protocol é o sistema de transmissão de arquivos mais usado na Internet. É requerida a autenticação do usuário para que seja permitida a conexão. Muitos servidores ftp disponibilizam acesso anônimo aos usuários, com acesso restrito.

Uma vez conectado a um servidor `ftp', você pode usar a maioria dos comandos do `GNU/Linux' para operá-lo.

ftp [_ip/dns_]'

Abaixo alguns dos comandos mais usados no FTP:
ls Lista arquivos do diretório atual.
cd [diretório] Entra em um diretório.
get [arquivo] Copia um arquivo do servidor ftp para o computador local. O arquivo é gravado, por padrão, no diretório onde o programa ftp foi executado.
hash [on/off] Por padrão esta opção está desligada. Quando ligada, faz com que o caracter "#" seja impresso na tela indicando o progresso do download.
mget [arquivos] Semelhante ao get, mas pode copiar diversos arquivos e permite o uso de curingas.
send [arquivo] Envia um arquivo para o diretório atual do servidor FTP (você precisa de uma conta com acesso a gravação para fazer isto).
prompt [on/off] Ativa ou desativa a pergunta para a cópia de arquivo. Se estiver como `off' assume sim para qualquer pergunta.

Exemplo: `ftp ftp.debian.org'.

whoami

Mostra o nome que usou para se conectar ao sistema. É útil quando você usa várias contas e não sabe com qual nome entrou no sistema :-)

$ whoami

hostname

Mostra ou muda o nome de seu computador na rede.

São opções:

-i Exibe endereço ip associado na interface
-I Exibe todos endereços ip de todas interfaces
-s Exibe o nome sem o domínio
-d Exibe o nome do domínio

Contas de Usuário

Este capítulo traz comandos usados para manipulação de conta de usuários e grupos em sistemas `GNU/Linux'. Entre os assuntos descritos aqui estão adicionar usuários ao sistema, adicionar grupos, incluir usuários em grupos existentes, etc.

adduser

Adiciona um usuário ou grupo no sistema. Por padrão, quando um novo usuário é adicionado, é criado um grupo com o mesmo nome do usuário.

Opcionalmente o `adduser' também pode ser usado para adicionar um usuário a um grupo (veja Seção 10.9, `Adicionando o usuário a um grupo extra'). Será criado um diretório home com o nome do usuário (a não ser que o novo usuário criado seja um usuário do sistema) e este receberá uma identificação. A identificação do usuário (UID) escolhida será a primeira disponível no sistema especificada de acordo com a faixa de UIDS de usuários permitidas no arquivo de configuração `/etc/adduser.conf'. Este é o arquivo que contém os padrões para a criação de novos usuários no sistema.

adduser [_opções_] [_usuário/grupo_]'

Onde:
[usuário/grupo] Nome do novo usuário que será adicionado ao sistema.

[opções]
-disable-passwd Não executa o programa `passwd' para escolher a senha e somente permite o uso da conta após o usuário escolher uma senha.
--force-badname Desativa a checagem de senhas ruins durante a adição do novo usuário. Por padrão o `adduser' checa se a senha pode ser facilmente adivinhada.
--group Cria um novo grupo ao invés de um novo usuário. A criação de grupos também pode ser feita pelo comando `addgroup'.
-uid [num] Cria um novo usuário com a identificação [num] ao invés de procurar o próximo UID disponível.
-gid [num] Faz com que o usuário seja parte do grupo [gid] ao invés de pertencer a um novo grupo que será criado com seu nome. Isto é útil caso deseje permitir que grupos de usuários possam ter acesso a arquivos comuns. Caso estiver criando um novo grupo com `adduser', a identificação do novo grupo será [num].
--home [dir] Usa o diretório [dir] para a criação do diretório home do usuário ao invés de usar o especificado no arquivo de configuração `/etc/adduser.conf'.
--ingroup [nome] Quando adicionar um novo usuário no sistema, coloca o usuário no grupo [nome] ao invés de criar um novo grupo.
--quiet Não mostra mensagens durante a operação.
--system Cria um usuário de sistema ao invés de um usuário normal.

Os dados do usuário são colocados no arquivo `/etc/passwd' após sua criação e os dados do grupo são colocados no arquivo `/etc/group'.
OBSERVAÇÃO: Caso esteja usando senhas ocultas (shadow passwords), as senhas dos usuários serão colocadas no arquivo `/etc/shadow' e as senhas dos grupos no arquivo `/etc/gshadow'. Isto aumenta mais a segurança do sistema porque somente o usuário `root' pode ter acesso a estes arquivos, ao contrário do arquivo `/etc/passwd' que possui os dados de usuários e devem ser lidos por todos.

addgroup

Adiciona um novo grupo de usuários no sistema. As opções usadas são as mesmas do `adduser'.

addgroup [_usuário/grupo_] [_opções_]'

passwd

Modifica a parâmetros e senha de usuário. Um usuário somente pode alterar a senha de sua conta, mas o superusuário (`root') pode alterar a senha de qualquer conta de usuário, inclusive a data de validade da conta, etc. Os donos de grupos também podem alterar a senha do grupo com este comando.
Os dados da conta do usuário como nome, endereço, telefone, também podem ser alterados com este comando.

passwd [_usuário_] [_opções_]'

Onde:
[usuário] Nome do usuário que terá sua senha alterada.

São opções
-e Força a expiração de senha para a conta especificada.
-k Somente altera a senha se a conta estiver expirada.

Procure sempre combinar letras maiúsculas, minúsculas, e números ao escolher suas senhas. Não é recomendado escolher palavras normais como sua senha pois podem ser vulneráveis a ataques de dicionários cracker. Outra recomendação é utilizar _senhas ocultas_ em seu sistema (_shadow password_).
Você deve ser o dono da conta para poder modificar a senhas. O usuário root pode modificar/apagar a senha de qualquer usuário.
Exemplo:
$passwd root'

gpasswd

Modifica parâmetros e senha de grupo. Um usuário somente pode alterar a senha de seu grupo, mas o superusuário (`root') pode alterar a senha de qualquer grupo de usuário, inclusive definir o administrador do grupo.

$gpasswd [_opções_] [_usuario_] [_grupo_]'

Onde:

[usuário]	Nome do usuário/grupo que terá sua senha alterada.

São opções:

-r	Remove a senha de grupo.
-R	Desativa o acesso do grupo usando o comando `newgrp'.
-a	Adiciona o usuário no grupo especificado.
-d	Apaga o usuário do grupo especificado.

Quando o grupo não possui senha, somente quem faz parte do grupo pode utilizar o comando new-grp.
Você deve ser o dono da conta para poder modificar a senhas. O usuário root pode modificar/apagar a senha de qualquer usuário.

Exemplo:
$ gpasswd grupo
$ gpasswd -a gleydson grupo

newgrp

Altera a identificação de grupo do usuário. Para retornar a identificação anterior, digite `exit' e tecle `Enter'. Para executar um comando com outra identificação de grupo de usuário, use o comando `sg'.

newgrp [_-_] [_grupo_]

Onde:
[_-_] Se usado, inicia um novo ambiente após o uso do comando `newgrp' (semelhante a um novo login no sistema), caso contrário, o ambiente atual do usuário é mantido.

[_grupo_] Nome do grupo ou número do grupo que será incluído. Quando este comando é usado, é pedida a senha do grupo que deseja acessar. Caso a senha do grupo esteja incorreta ou não exista senha definida, a execução do comando é negada. A listagem dos grupos que pertence atualmente pode ser feita usando o comando `id'.

userdel

Apaga um usuário do sistema. Quando é usado, este comando apaga todos os dados da conta especificado dos arquivos de contas do sistema.

`userdel [_-r_] [_usuário_]'

Onde: -r apaga também o diretório HOME do usuário.
OBS: Note que uma conta de usuário não poderá ser removida caso ele estiver no sistema, pois os programas podem precisar ter acesso aos dados dele (como UID, GID) no `/etc/passwd'.

groupdel

Apaga um grupo do sistema. Quando é usado, este comando apaga todos os dados do grupo especificado dos arquivos de contas do sistema.

groupdel [_grupo_]'

Tenha certeza que não existem arquivos/diretórios criados com o grupo apagado através do comando `find'.
OBS: Você não pode remover o grupo primário de um usuário. Remova o usuário primeiro.

sg

Executa um comando com outra identificação de grupo. A identificação do grupo de usuário é modificada somente durante a execução do comando. Para alterar a identificação de grupo durante sua seção shell, use o comando `newgrp'.

sg [_-_] [_grupo_] [_comando_]'

Onde:
[_-_] Se usado, inicia um novo ambiente durante o uso do comando (semelhante a um novo login e execução do comando), caso contrário, o ambiente atual do usuário é mantido.
`grupo'

Nome do grupo que o comando será executado [comando]; Comando que será pelo bash.
Quando este comando é usado, é pedida a senha do grupo que deseja acessar. Caso a senha do grupo esteja incorreta ou não exista senha definida, a execução do comando é negada.

Exemplo:
$ sg root ls /root'

Adicionando o usuário a um grupo extra

Para adicionar um usuário em um novo grupo e assim permitir que ele acesse os arquivos/diretórios que pertencem àquele grupo, você deve estar como root e editar o arquivo `/etc/group' com o comando `vigr'.

Este arquivo possui o seguinte formato:

NomedoGrupo:senha:GID:usuários

Onde:
NomedoGrupo É o nome daquele grupo de usuários.
senha Senha para ter acesso ao grupo. Caso esteja utilizando senhas ocultas para grupos, as senhas estarão em `/etc/gshadow'.
GID Identificação numérica do grupo de usuário.
usuarios Lista de usuários que também fazem parte daquele grupo. Caso exista mais de um nome de usuário, eles devem estar separados por vírgula.

Deste modo para acrescentar o usuário "joao" ao grupo `audio' para ter acesso aos dispositivos de som do Linux, acrescente o nome no final da linha: "audio:x:100:joao". Pronto, basta digitar `logout' e entrar novamente com seu nome e senha, você estará fazendo parte do grupo `audio' (confira digitando `groups' ou `id').
Outros nomes de usuários podem ser acrescentados ao grupo `audio' bastando separar os nomes com vírgula. Você também pode usar o comando `adduser' da seguinte forma para adicionar automaticamente um usuário a um grupo:

$ adduser joao audio

Isto adicionaria o usuário "joao" ao grupo `audio' da mesma forma que fazendo-se a edição manualmente.

chfn

Muda os dados usados pelo comando `finger'.

chfn [_usuário_] [_opções_]'

Onde:
usuário Nome do usuário.

São opções:
-f [nome] Adiciona/altera o nome completo do usuário.
-r [nome] Adiciona/altera o número da sala do usuário.
-w [tel] Adiciona/altera o telefone de trabalho do usuário.
-h [tel] Adiciona/altera o telefone residencial do usuário.
-o [outros] Adiciona/altera outros dados do usuário.

Caso o nome que acompanha as opções (como o nome completo) contenha espaços, use " " para identifica-lo.

Exemplo:
$chfn -f "Nome do Usuário root" root'

id

Mostra a identificação atual do usuário, grupo primário e outros grupos que pertence.

id [_opções_] [_usuário_]'

Onde:
usuário É o usuário que desejamos ver a identificação, grupos primários e omplementares.

São opções:
-g, --group Mostra somente a identificação do grupo primário.
-G, --groups Mostra a identificação de outros grupos que pertence.
-n, --name Mostra o nome do usuário e grupo ao invés da identificação numérica.
-u, --user Mostra somente a identificação do usuário (user ID).
-r, --real Mostra a identificação real de usuário e grupo, ao invés da efetiva. Esta opção deve ser usada junto com uma das opções: -u,
-g, ou -G. Caso não sejam especificadas opções, `id' mostrará todos os dados do usuário.

Exemplo:
$ id
$, id --user
$ id -r -u.

logname

Mostra seu login (username).

$ logname

users

Mostra os nomes de usuários usando atualmente o sistema. Os nomes de usuários são mostrados através de espaços sem detalhes adicionais, para ver maiores detalhes sobre os usuários, id e who.

users Os nomes de usuários atualmente conectados ao sistema são obtidos do arquivo `/va/log/wtmp'.

groups

Mostra os grupos que o usuário pertence.

groups [_usuário_]'

Exemplo:
$ groups',
$ groups root

As permissões de acesso protegem o sistema de arquivos Linux do acesso indevido de pessoas ou programas não autorizados.

Permissões de acesso a arquivos e diretórios

A permissão de acesso do `GNU/Linux' também impede que um programa mal intencionado, por exemplo, apague um arquivo que não deve, envie arquivos especiais para outra pessoa ou forneça acesso da rede para que outros usuários invadam o sistema. O sistema `GNU/Linux' é muito seguro e como qualquer outro sistema seguro e confiável impede que usuários mal intencionados (ou iniciantes que foram enganados) instalem programas enviados por terceiros sem saber para que eles realmente servem e causem danos irreversíveis em seus arquivos, seu micro ou sua empresa.

Esta seção do guia, de início, pode ser um pouco difícil de se entender, então recomendo ler e ao mesmo tempo prática-la para uma ótima compreensão. Não se preocupe, também coloquei exemplos para ajuda-lo a entender o sistema de permissões de acesso do ambiente `GNU/Linux'.

Donos, Grupos e outros usuários

A idéia básica da segurança no sistema `GNU/Linux' é definir o acesso aos arquivos por donos, grupos e outros usuários:

Dono

É a pessoa que criou o arquivo ou o diretório. O nome do dono do arquivo/diretório é o mesmo do usuário usado para entrar no sistema `GNU/Linux'. Somente o dono pode modificar as permissões de acesso do arquivo.
As permissões de acesso do dono de um arquivo somente se aplicam ao dono do arquivo/diretório. A identificação do dono também é chamada de `user id' (UID).
A identificação de usuário ao qual o arquivo pertence é armazenada no arquivo `/etc/passwd' e do grupo no arquivo `/etc/group'. Estes são arquivos textos comuns e podem ser editados em qualquer editor de texto, mas utilize preferencialmente os comandos `vipw' e `vigr' que executa procedimentos adicionais de checagem de uids e grupos após a alteração. Tenha cuidado para não modificar o campo que contém a senha do usuário encriptada (que pode estar armazenada no arquivo `/etc/passwd' caso não estiver usando senhas ocultas).

Grupo

Permite que vários usuários diferentes tenham acesso a um mesmo arquivo (já que somente o dono poderia ter acesso ao arquivo).
Cada usuário pode fazer parte de um ou mais grupos e então acessar arquivos que pertençam ao mesmo grupo que o seu (mesmo que estes arquivos tenham outro _dono_).
Por padrão, quando um novo usuário é criado e não especificar nenhum grupo, ele pertencerá ao grupo de mesmo nome do seu grupo primário (este comportamento é controlado pelo parametro USERGROUPS=yes' do arquivo `/etc/adduser.conf'.

A identificação do grupo é chamada de `GID (_group id_)'.
Um usuário pode pertencer a um ou mais grupos.
É a categoria de usuários que não são donos ou não pertencem ao grupo do arquivo.
Cada um dos tipos acima possuem três tipos básicos de permissões de acesso que serão vistas na próxima seção.

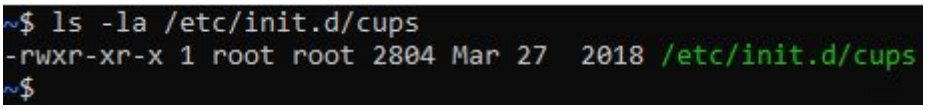
Tipos de Permissões de Acesso

Quanto aos tipos de permissões que se aplicam ao _dono_, _grupo_ e _outros usuários_, temos 3 permissões básicas:

`r' - Permissão de leitura para arquivos. Caso for um diretório, permite listar seu conteúdo (através do comando `ls', por exemplo).

`w' - Permissão de gravação para arquivos. Caso for um diretório, permite a gravação de arquivos ou outros diretórios dentro dele.
Para que um arquivo/diretório possa ser apagado, é necessário o acesso a gravação.
`x' - Permite executar um arquivo (caso seja um programa executável). Caso seja um diretório, permite que seja acessado através do comando `cd'.

As permissões de acesso a um arquivo/diretório podem ser visualizadas com o uso do comando `ls -la'. As 3 letras (rwx) são agrupadas da seguinte forma:



Virou uma bagunça não? Vou explicar cada parte para entender o que quer dizer as 10 letras acima (da esquerda para a direita):

A primeira letra diz qual é o tipo do arquivo. Caso tiver um "d" é um diretório, um "l" um link a um arquivo no sistema, um "-" quer dizer que é um arquivo comum, etc.

Da segunda a quarta letra (rwx) dizem qual é a permissão de acesso ao _dono_ do arquivo. Neste caso root ele tem a permissão de ler (r - read), gravar (w - write) e executar (x - execute) o arquivo /etc/init.d/cups.

Da quinta a sétima letra (r-x) diz qual é a permissão de acesso ao _grupo_ do arquivo. Neste caso todos os usuários que pertencem ao grupo _users_ tem a permissão de ler (r), e também executar (x) o arquivo `/etc/init.d/cups.

Da oitava a décima letra (r--) diz qual é a permissão de acesso para os _outros usuários_. Neste caso todos os usuários que não são donos do arquivo `teste' tem a permissão somente para ler o programa.

Utilize o comando `chmod' para detalhes sobre a mudança das permissões de acesso de arquivos/diretórios.

Etapas para acesso a um arquivo/diretório

O acesso a um arquivo/diretório é feito verificando primeiro se o usuário que acessará o arquivo é o seu _dono_, caso seja, as permissões de dono do arquivo são aplicadas. Caso não seja o _dono_ do arquivo/diretório, é verificado se ele pertence ao grupo correspondente, caso pertença, as permissões do _grupo_ são aplicadas.

Caso não pertença ao _grupo_, são verificadas as permissões de acesso para os outros usuários que não são _donos_ e não pertencem ao _grupo_ correspondente ao arquivo/diretório.

Após verificar aonde o usuário se encaixa nas permissões de acesso do arquivo (se ele é o _dono_, pertence ao _grupo_, ou _outros usuários_), é verificado se ele terá permissão acesso para o que deseja fazer (ler, gravar ou executar o arquivo), caso não tenha, o acesso é negado, mostrando uma mensagem do tipo: "Permission denied" (permissão negada).

O que isto que dizer é que mesmo que você seja o dono do arquivo e definir o acesso do _dono_ (através do comando `chmod') como somente leitura (r) mas o acesso dos _outros usuários_ como leitura e gravação, você somente poderá ler este arquivo mas os outros usuários poderão ler/grava-lo.

As permissões de acesso (leitura, gravação, execução) para donos, grupos e outros usuários são independentes, permitindo assim um nível de acesso diferenciado. Para maiores detalhes veja `Tipos de Permissões de Acesso'.

Lembre-se: Somente o dono pode modificar as permissões de um arquivo/diretório!

Exemplo de permissão de um diretório

Abaixo um exemplo com explicações das permissões de acesso a um diretório no `GNU/Linux'

```
~$ ls -ld /var/log/
drwxrwxr-x 1 root syslog 4096 Oct 25 14:09 /var/log/
```

drwxrwxr-x
Permissões de acesso ao diretório `exemplo'. É um conjunto de 10 letras que especificam o tipo de arquivo, permissão do dono do diretório, grupo que o diretório pertence e permissão de acesso a outros usuários. Veja as explicações abaixo:

A primeira letra (do conjunto das 10) determina o tipo do arquivo. Neste caso é um diretório porque tem a letra “ d “.

“rwx” Estas 3 letras (da segunda a quarta) são as permissões de acesso do dono do diretório `log. O dono do diretório (neste caso root) tem a permissão para listar arquivos do diretório (r), gravar arquivos no diretório (w) e entrar no diretório (x).

“rwx” Estas 3 letras (da quinta a sétima) são as permissões de acesso dos usuários que pertencem ao grupo syslog. Os usuários que pertencem ao grupo syslog tem a permissão para listar arquivos do diretório (r) e entrar no, gravar (w), acessar (x) log.

“r-x” Estes 3 simbolos (do oitavo ao décimo) são as permissões de acesso para usuários que não são donos do diretório; Exemplo e que não pertencem ao grupo syslog. Com as permissões acima, nenhum usuário que se encaixe nas condições de _dono_ e _grupo_ do diretório tem a permissão de acessa-lo.
“root” Nome do dono do diretório log.
“syslog” Nome do grupo que diretório `log pertence.
“/var/log” Nome do diretório.

OBSERVAÇÕES:
O usuário `root' não tem nenhuma restrição de acesso ao sistema.
Se você tem permissões de gravação no diretório e tentar apagar um arquivo que você não tem permissão de gravação, o sistema perguntará se você confirma a exclusão do arquivo apesar do modo leitura. Caso você tenha permissões de gravação no arquivo, o arquivo será apagado por padrão sem mostrar nenhuma mensagem de erro (a não ser que seja especificada a opção -i com o comando `rm').
Por outro lado, mesmo que você tenha permissões de gravação em um arquivo mas não tenha permissões de gravação em um diretório, a exclusão do arquivo será negada.
Isto mostra que é levado mais em consideração a permissão de acesso do diretório do que as permissões dos arquivos e sub-diretórios que ele contém. Este ponto é muitas vezes ignorado por muitas pessoas e expõem seu sistema a riscos de segurança. Imagine o problema que algum usuário que não tenha permissão de gravação em um arquivo mas que a tenha no diretório pode causar em um sistema mal administrado.

Permissões de Acesso Especiais

Em adição as três permissões básicas (rwx), existem permissões de acesso especiais (stX) que afetam os arquivos e diretórios:

```
~$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 59640 Jan 25  2018 /usr/bin/passwd
```

s - Quando é usado na permissão de acesso do _Dono_, ajusta a identificação efetiva do usuário do processo durante a execução de um programa, também chamado de _bit setuid_. Não tem efeito em diretórios.

Quando `s' é usado na permissão de acesso do _Grupo_, ajusta a identificação efetiva do grupo do processo durante a execução de um programa, chamado de _bit setgid_. É identificado pela letra `s' no lugar da permissão de execução do grupo do arquivo/diretório. Em diretórios, força que os arquivos criados dentro dele pertençam ao mesmo grupo do diretório, ao invés do grupo primário que o usuário pertence.

Ambos _setgid_ e _setuid_ podem aparecer ao mesmo tempo no mesmo arquivo/diretório. A permissão de acesso especial `s' somente pode aparecer no campo _Dono_ e _Grupo_.

S - Idêntico a "s". Significa que não existe a permissão "x" (execução ou entrar no diretório) naquela posição. Um exemplo é o chmod 2760 em um diretório.

```
~$ ls -ld  /tmp/
drwxrwxrwt 1 root root 4096 Nov  1 19:23 /tmp/
```

t - Salva a imagem do texto do programa no dispositivo swap, assim ele será carregado mais rapidamente quando executado, também chamado de _stick bit_.

Em diretórios, impede que outros usuários removam arquivos dos quais não são donos. Isto é chamado de colocar o diretório em modo `append-only'. Um exemplo de diretório que se encaixa perfeitamente nesta condição é o `/tmp', todos os usuários devem ter acesso para que seus programas possam criar os arquivos temporários lá, mas nenhum pode apagar arquivos dos outros. A permissão especial `t', pode ser especificada somente no campo outros usuários das permissões de acesso.

T - Idêntico a "t". Significa que não existe a permissão "x" naquela posição (por exemplo, em um chmod 1776 em um diretório).

X - Se você usar `X' ao invés de `x', a permissão de execução somente é aplicada se o arquivo já tiver permissões de execução. Em diretórios ela tem o mesmo efeito que a permissão de execução `x'.

Exemplo da permissão de acesso especial `X':

1. Crie um arquivo `teste' (digitando `touch teste') e defina sua permissão para `rw-rw-r--' (`chmod ug=rw,o=r teste' ou `chmod 664 teste').
2. Agora use o comando `chmod a+X teste'
3. digite `ls -l'
4. Veja que as permissões do arquivo não foram afetadas.
5. agora digite `chmod o+x teste'
6. digite `ls -l', você colocou a permissão de execução para os outros usuários.
7. Agora use novamente o comando `chmod a+X teste'
8. digite `ls -l'
9. Veja que agora a permissão de execução foi concedida a todos os usuários, pois foi verificado que o arquivo era executável (tinha permissão de execução para outros usuários).
10. Agora use o comando `chmod a-X teste'
11. Ele também funcionará e removerá as permissões de execução de todos os usuários, porque o arquivo `teste' tem permissão de execução (confira digitando `ls -l').
12. Agora tente novamente o `chmod a+X teste'
13. Você deve ter reparado que a permissão de acesso especial `X' é semelhante a `x', mas somente faz efeito quanto o arquivo já tem permissão de execução para o dono, grupo ou outros usuários.

Em diretórios, a permissão de acesso especial `X' funciona da mesma forma que `x', até mesmo se o diretório não tiver nenhuma permissão de acesso (`x').

A conta root

‘*Esta seção foi retirada do Manual de Instalação da Debian.’*

A conta root é também chamada de super usuário, este é um login que não possui restrições de segurança. A conta root somente deve ser usada para fazer a administração do sistema, e usada o menor tempo possível.

Qualquer senha que criar deverá conter de 6 a 8 caracteres (em sistemas usando crypto) ou até frases inteiras (caso esteja usando MD5, que garante maior segurança), e também poderá conter letras maiúsculas e minúsculas, e também caracteres de pontuação. Tenha um cuidado especial quando escolher sua senha root, porque ela é a conta mais poderosa. Evite palavras de dicionário ou o uso de qualquer outros dados pessoais que podem ser adivinhados.

Se qualquer um lhe pedir senha root, seja extremamente cuidadoso.
Você normalmente nunca deve distribuir sua conta root, a não ser que esteja administrando um computador com mais de um administrador do sistema.
Utilize uma conta de usuário normal ao invés da conta root para operar seu sistema. Porque não usar a conta root? Bem, uma razão para evitar usar privilégios root é por causa da facilidade de se cometer danos irreparáveis como root. Outra razão é que você pode ser enganado e rodar um programa _Cavalo de Tróia_ -- que é um programa que obtém poderes do _super usuário_ para comprometer a segurança do eu sistema sem que você saiba.

chmod

Muda a permissão de acesso a um arquivo ou diretório. Com este comando você pode escolher se usuário ou grupo terá permissões para ler, gravar, executar um arquivo ou arquivos. Sempre que um arquivo é criado, seu dono é o usuário que o criou e seu grupo é o grupo do usuário (exceto para diretórios configurados com a permissão de grupo `'"s"', será visto adiante).

$ chmod [_opções_] [_permissões_] [_diretório/arquivo_]'

Onde:
diretório/arquivo - Diretório ou arquivo que terá sua permissão mudada.

opções
-v, --verbose - Mostra todos os arquivos que estão sendo processados.
-f, --silent - Não mostra a maior parte das mensagens de erro.
-c, --change - Semelhante a opção -v, mas só mostra os arquivos que tiveram as permissões alteradas.

w.luis.araujo@gmail.com
wluisaraujo.github.io

-R, --recursive - Muda permissões de acesso do _diretório/arquivo_ no diretório atual e sub-diretórios.
ugoa+-=rwxXst - Controla que nível de acesso será mudado. Especificam, em ordem, usuário (u), grupo (g), outros (o), todos (a).
* rwx - _r_ permissão de leitura do arquivo. _w_ permissão de gravação. _x_ permissão de execução (ou acesso a diretórios). `chmod' não muda permissões de links simbólicos, as permissões devem ser mudadas no arquivo alvo do link. Também podem ser usados códigos numéricos octais para a mudança das permissões de acesso a arquivos/diretórios. Para detalhes veja Seção 11.10, `Modo de permissão octal'.

DICA: É possível copiar permissões de acesso do arquivo/diretório, por exemplo, se o arquivo `teste.txt' tiver a permissão de acesso r-xr-----' e você digitar `chmod o=u', as permissões de acesso dos outros usuários (o) serão idênticas ao do dono (u). Então a nova permissão de acesso do arquivo `teste.txt' será `r-xr--r-x'

Exemplos de permissões de acesso:
chmod g+r * - Permite que todos os usuários que pertençam ao grupo dos arquivos (g) tenham (+) permissões de leitura (r) em todos os arquivos do diretório atual. `chmod o-r teste.txt'
Retira (-) a permissão de leitura (r) do arquivo `teste.txt' para os outros usuários (usuários que não são donos e não pertencem ao grupo do arquivo `teste.txt').

chmod uo+x teste.txt' - Inclui (+) a permissão de execução do arquivo `teste.txt' para o dono e outros usuários do arquivo.

chmod a+x teste.txt' - Inclui (+) a permissão de execução do arquivo `teste.txt' para o dono, grupo e outros usuários.

chmod a=rw teste.txt' - Define a permissão de todos os usuários exatamente (=) para leitura e gravação do arquivo `teste.txt'.

chgrp

Muda o grupo de um arquivo/diretório.

chgrp [_opções_] [grupo] [arquivo/diretório]'

Onde:
grupo - Novo grupo do _arquivo/diretório_.
arquivo/diretório - Arquivo/diretório que terá o grupo alterado.

opções
-c, --changes - Somente mostra os arquivos/grupos que forem alterados.
-f, --silent - Não mostra mensagens de erro para arquivos/diretórios que não puderam ser alterados.
-v, --verbose - Mostra todas as mensagens e arquivos sendo modificados.

-R, --recursive - Altera os grupos de arquivos/sub-diretórios do diretório atual.

chown

Muda dono de um arquivo/diretório. Opcionalmente pode também ser usado para mudar o grupo.

$ chown [_opções_] [dono.grupo] [diretório/arquivo]'

Onde:
dono.grupo - Nome do _dono.grupo_ que será atribuído ao _diretório/arquivo_. O grupo é opcional.

diretório/arquivo - Diretório/arquivo que o dono.grupo será modificado.

opções
-v, --verbose - Mostra os arquivos enquanto são alterados.
-f, --supress - Não mostra mensagens de erro durante a execução do programa.
-c, --changes - Mostra somente arquivos que forem alterados.
-R, --recursive - Altera dono e grupo de arquivos no diretório atual e sub-diretórios.

O _dono.grupo_ pode ser especificado usando o nome de grupo ou o código numérico correspondente ao grupo (GID).

Você deve ter permissões de gravação no diretório/arquivo para alterar seu dono/grupo.

* `chown gleydson teste.txt' - Muda o dono do arquivo `teste.txt' para `gleydson'.
* `chown gleydson.foca teste.txt' - Muda o dono do arquivo
`teste.txt' para `gleydson' e seu grupo para `foca'.
* `chown -R gleydson.focalinux *' - Muda o dono/grupo dos arquivos do diretório atual e sub-diretórios para `gleydson/focalinux' (desde que você tenha permissões de gravação no diretórios e sub-diretórios).

Modo de permissão octal

Ao invés de utilizar os modos de permissão `+r', `-r', etc, pode ser usado o modo octal para se alterar a permissão de acesso a um arquivo.
O modo octal é um conjunto de oito números onde cada número define um tipo de acesso diferente.

É mais flexível gerenciar permissões de acesso usando o modo octal ao invés do comum, pois você especifica diretamente a permissão do dono, grupo, outros ao invés de gerenciar as permissões de cada um separadamente. Abaixo a lista de permissões de acesso octal:
* `0' - Nenhuma permissão de acesso. Equivalente a -rwx.
* `1' - Permissão de execução (x).
* `2' - Permissão de gravação (w).

* `3' - Permissão de gravação e execução (wx). Equivalente a permissão 2+1
* `4' - Permissão de leitura (r).
* `5' - Permissão de leitura e execução (rx). Equivalente a permissão 4+1
* `6' - Permissão de leitura e gravação (rw). Equivalente a permissão 4+2
* `7' - Permissão de leitura, gravação e execução. Equivalente a +rwx (4+2+1).

O uso de um deste números define a permissão de acesso do _dono_, _grupo_ ou _outros usuários_. Um modo fácil de entender como as permissões de acesso octais funcionam, é através da seguinte tabela:

1 = Executar
2 = Gravar
4 = Ler

* Para Dono e Grupo, multiplique as permissões acima por x100 e x10.

Basta agora fazer o seguinte:

* Somente permissão de execução, use 1.
* Somente a permissão de leitura, use 4.
* Somente permissão de gravação, use 2.
* Permissão de leitura/gravação, use 6 (equivale a 2+4 / Gravar+Ler).
* Permissão de leitura/execução, use 5 (equivale a 1+4 / Executar+Ler).
* Permissão de execução/gravação, use 3 (equivale a 1+2 / Executar+Gravar).
* Permissão de leitura/gravação/execução, use 7 (equivale a 1+2+4 / Executar+Gravar+Ler).

Vamos a prática com alguns exemplos:
$ chmod 764 teste

Os números são interpretados da _direita para a esquerda_ como permissão de acesso aos _outros usuários_ (4), _grupo_ (6), e _dono_ (7). O exemplo acima faz os _outros usuários_ (4) terem acesso somente leitura (r) ao arquivo `teste', o _grupo_ (6) ter a permissão de leitura e gravação (w), e o _dono_ (7) ter permissão de leitura, gravação e execução (rwx) ao arquivo `teste'.

Outro exemplo:

$ chmod 40 teste

O exemplo acima define a permissão de acesso dos _outros usuários_ (0) como nenhuma, e define a permissão de acesso do _grupo_ (4) como somente leitura (r). Note usei somente dois números e então a permissão de acesso do _dono_ do arquivo `não' é modificada (leia as permissões de acesso da direita para a esquerda!).

$ chmod 751 teste

O exemplo acima define a permissão de acesso dos _outros usuários_ (1) para somente execução (x), o acesso do _grupo_ (5) como leitura e execução (rx) e o acesso do _dono_ (7) como leitura, gravação e execução (rwx).

umask

A umask (_user mask_) são 3 números que definem as permissões iniciais do `dono', `grupo' e `outros usuários' que o arquivo/diretório receberá quando for criado ou copiado para um novo local. Digite `umask' sem parâmetros para retornar o valor de sua umask atual.

A umask tem efeitos diferentes caso o arquivo que estiver sendo criado for _binário_ (um programa executável) ou _texto. Veja a tabela a seguir para ver qual é a mais adequada a sua situação:

	ARQUIVO		DIRETÓRIO
UMASK	Binário	Texto	
0	r-x	rw-	rwx
1	r--	rw-	rw-
2	r-x	r--	r-x
3	r--	r--	r--
4	--x	-w-	-wx
5	---	-w-	-w-
6	--x	---	--x
7	---	---	---

Um _arquivo texto_ criado com o comando `umask 012;touch texto.txt' receberá as permissões `-rw-rw-r--', pois 0 (dono) terá permissões `rw-', 1 (grupo), terá permissões `rw-' e 2 (outros usuários) terão permissões `r--'. Um _arquivo binário_ copiado com o comando

$ umask 012 ; cp /bin/ls /tmp/ls'

Receberá as permissões `-r-xr--r-x' (confira com a tabela acima).

Por este motivo é preciso atenção antes de escolher a umask, um valor mal escolhido poderia causar problemas de acesso a arquivos, diretórios ou programas não sendo executados. O valor padrão da umask na maioria das distribuições atuais é 022. A umask padrão no sistema Debian é a 022 .

A umask é de grande utilidade para programas que criam arquivos/diretórios temporários, desta forma pode-se bloquear o acesso de outros usuários desde a criação do arquivo, evitando recorrer ao `chmod'.

Redirecionamentos e Pipe

Esta seção explica o funcionamento dos recursos de direcionamento de entrada e saída do sistema `GNU/Linux'.

Insert >

Redireciona a saída padrão de um programa/comando/script para algum dispositivo ou arquivo ao invés do dispositivo de saída padrão (tela).

Quando é usado com arquivos, este redirecionamento cria ou substitui o conteúdo do arquivo.

Por exemplo, você pode usar o comando `ls' para listar arquivos e usar `ls >listagem' para enviar a saída do comando para o arquivo `listagem'. Use o comando `cat' para visualizar o conteúdo do arquivo `listagem'.

O mesmo comando pode ser redirecionado para o segundo console `/dev/tty2' usando: `ls >/dev/tty2', o resultado do comando `ls' será mostrado no segundo console (pressione `ALT' e `F2' para mudar para o segundo console e `ALT' e `F1' para retornar ao primeiro). O mesmo resultado pode ser obtido com o comando `ls 1>/dev/tty2', sendo que o número `1' indica que será capturada a saída padrão do comando.

Para redirecionar somente a saída de erros do comando `ls', use a sintaxe: `ls 2>/tmp/erros-do-ls'

Append >>

Redireciona a saída padrão de um programa/comando/script para algum dispositivo ou adiciona as linhas ao final de arquivo ao invés do dispositivo de saída padrão (tela). A diferença entre este redirecionamento duplo e o simples, é se caso for usado com arquivos, adiciona a saída do comando ao final do arquivo existente ao invés de substituir seu conteúdo.

Por exemplo, você pode acrescentar a saída do comando `ls' ao arquivo `listagem' do capítulo anterior usando `ls / >>listagem'. Use o comando `cat' para visualizar o conteúdo do arquivo `listagem'.

< input

Direciona a entrada padrão de arquivo/dispositivo para um comando. Este comando faz o contrário do anterior, ele envia dados ao comando.

Você pode usar o comando `cat <teste.txt' para enviar o conteúdo do arquivo `teste.txt' ao comando `cat' que mostrará seu conteúdo (é claro que o mesmo resultado

pode ser obtido com `cat teste.txt' mas este exemplo serviu para mostrar a funcionalidade do <).

$ tr [:upper:] [:lower:] < /etc/resolv.conf

<< input

Este redirecionamento serve principalmente para marcar o fim de exibição de um bloco. Este é especialmente usado em conjunto com o comando `cat', mas também tem outras aplicações. Por exemplo:

```
$ cat << final
este arquivo
será mostrado
até que a palavra final seja
localizada no inicio da linha
final
```

| (pipe)

Envia a saída de um comando para a entrada do próximo comando para continuidade do processamento. Os dados enviados são processados pelo próximo comando que mostrará o resultado do processamento.

Por exemplo: `ls -la | more', este comando faz a listagem longa de arquivos que é enviado ao comando `more' (que tem a função de efetuar uma pausa a cada 25 linhas do arquivo).

Outro exemplo é o comando `locate find | grep "bin/"', neste comando todos os caminhos/arquivos que contém _find_ na listagem serão mostrados (inclusive man pages, bibliotecas, etc.), então enviamos a saída deste comando para `grep "bin/"' para mostrar somente os diretórios que contém binários. Mesmo assim a listagem ocupe mais de uma tela, podemos acrescentar o `more':

$ locate find | grep "bin/" | more

Podem ser usados mais de um comando de redirecionamento (<, >, |) em um mesmo comando.

Diferença entre o "|" e o ">"

A principal diferença entre o "|" e o ">", é que o Pipe envolve processamento entre comandos, ou seja, a saída de um comando é enviado a entrada do próximo e o ">" redireciona a saída de um comando para um arquivo/dispositivo.

Você pode notar pelo exemplo acima (`ls -la | more') que ambos `ls' e `more' são comandos porque estão separados por um "|". Se um deles não existir ou estiver digitado incorretamente, será mostrada uma mensagem de erro.

Um resultado diferente seria obtido usando um `">"' no lugar do `"|"'; A saída do comando `ls -la > more' seria gravada em um arquivo chamado `more'.

tee

Envia ao mesmo tempo o resultado do programa para a saída padrão (tela) e para um arquivo. Este comando deve ser usado com o pipe "|".

$ comando | tee [_arquivo_]

Exemplo:
$ ls -la | tee listagem.txt

A saída do comando será mostrada normalmente na tela e ao mesmo tempo gravada no arquivo `listagem.txt'.

Como obter ajuda no sistema

Como obter ajuda no sistema

Dúvidas são comuns durante o uso do `GNU/Linux' e existem várias maneiras de se obter ajuda e encontrar a resposta para algum problema.
O `GNU/Linux' é um sistema bem documentado, provavelmente tudo o que imaginar fazer ou aprender já esta disponível para leitura e aprendizado. Abaixo segue algumas formas úteis para encontrar a solução de sua dúvida, vale a pena conhecê-las.

Páginas de Manual

As _páginas de manual_ acompanham quase todos os programas `GNU/Linux'. Elas trazem uma descrição básica do comando/programa e detalhes sobre o funcionamento de opção. Uma página de manual é visualizada na forma de texto único com rolagem vertical. Também documenta parâmetros usados em alguns arquivos de configuração.

A utilização da página de manual é simples, digite:

$ man [_seção_] [_comando/arquivo_]'

Onde:

seção - É a seção de manual que será aberta, se omitido, mostra a _primeira_ seção sobre o comando encontrada (em ordem crescente).

comando/arquivo - Comando/arquivo que deseja pesquisar. A navegação dentro das páginas de manual é feita usando-se as teclas:

* q - Sai da página de manual
* PageDown ou f - Rola 25 linhas abaixo
* PageUP ou w - Rola 25 linhas acima
* SetaAcima ou k - Rola 1 linha acima
* SetaAbaixo ou e - Rola 1 linha abaixo
* r - Redesenha a tela (refresh)
* p ou g - Inicio da página
* h - Ajuda sobre as opções da página de manual
* s - Salva a página de manual em formato texto no arquivo especificado (por exemplo: `/tmp/ls').

Exemplo:
$ man ls
$ man 5 hosts_access

Info Pages

Idêntico as páginas de manual, mas é usada navegação entre as páginas.
Se pressionarmos <Enter> em cima de uma palavra destacada, a `info pages' nos levará a seção correspondente. A _info pages_ é útil quando sabemos o nome do comando e queremos saber para o que ele serve. Também traz explicações detalhadas sobre uso, opções e comandos.
Para usar a info pages, digite:

$ info [_comando/programa_]'

Se o nome do _comando/programa_ não for digitado, a info pages mostra a lista de todos os manuais de _comandos/programas_ disponíveis. A navegação da info pages é feita através de nomes marcados com um "*" (hipertextos) que se pressionarmos <Enter>, nos levará até a seção correspondente. A _info pages_ possui algumas teclas de navegação úteis:

* q - Sai da info pages
* ? - Mostra a tela de ajuda (que contém a lista completa de teclas de navegação e muitos outras opções).
* n - Avança para a próxima página
* p - Volta uma página
* u - Sobre um nível do conteúdo (até checar ao índice de documentos)
* m - Permite usar a localização para encontrar uma página do `info'. Pressione `m', digite o comando e tecle <Enter> que será levado automaticamente a página correspondente.
* d - Volta ao índice de documentos.

Existem muitos outras teclas de navegação úteis na info pages, mas estas são as mais usadas. Para mais detalhes, entre no programa `info' e pressione `?'.

Exemplo:
$ info cvs

Help on line

Ajuda rápida, é útil para sabermos quais opções podem ser usadas com o comando/programa. Quase todos os comandos/programas `GNU/Linux' oferecem este recurso que é útil para consultas rápidas (e quando não precisamos dos detalhes das páginas de manual). É útil quando se sabe o nome do programa mas deseja saber quais são as opções disponíveis e para o que cada uma serve. Para acionar o _help on line_, digite:

$ [_comando_] --help

comando - é o comando/programa que desejamos ter uma explicação rápida.

O Help on Line não funciona com comandos internos (embutidos no Bash), para ter uma ajuda rápida sobre os comandos internos.

Por exemplo:

$ ls --help

Ajuda rápida, útil para saber que opções podem ser usadas com os comandos internos_ do interpretador de comandos. O comando `help' somente mostra a ajuda para comandos internos, para ter uma ajuda similar para comandos externos, veja Seção 15.3, `Help on line'. Para usar o `help' digite:

$ help [_comando_]'

Por exemplo, `help echo', `help exit'

apropos/whatis

Apropos procura por _programas/comandos_ através da descrição. É útil quando precisamos fazer alguma coisa mas não sabemos qual comando usar. Ele faz sua pesquisa nas páginas de manual existentes no sistema e lista os comandos/programas que atendem a consulta. Para usar o comando `apropos' digite:

$ apropos [_descrição_]'

Digitando `apropos copy', será mostrado todos os comandos que tem a palavra `copy' em sua descrição (provavelmente os programas que copiam arquivos, mas podem ser mostrados outros também).

locate

Localiza uma palavra na estrutura de arquivos/diretórios do sistema.
É útil quando queremos localizar onde um comando ou programa se encontra (para copia-lo, curiosidade, etc). A pesquisa é feita em um banco de dados construído com o comando `updatedb' sendo feita a partir do diretório raíz `/' e sub-diretórios. Para fazer uma consulta com o `locate' usamos:

$ locate [_expressão_]'

A _expressão_ deve ser o nome de um arquivo diretório ou ambos que serão procurados na estrutura de diretórios do sistema. Como a consulta por um programa costuma localizar também sua página de manual, é recomendável usar _"pipes"_ para filtrar a saída do comando.

Por exemplo, para listar os diretórios que contém o nome "_cp_": `locate cp'. Agora mostrar somente arquivos binários, usamos:

$ locate cp|grep bin/

which

Localiza um programa na estrutura de diretórios do path. É muito semelhante ao locate, mas a busca é feita no `path' do sistema e somente são mostrados arquivos executáveis .

$ which [_programa/comando_]

FONTE

“ Sábio aquele que aprende com tudo e com todos ”

Este material foi baseado em literaturas e artigos, manuais, etc. Disponíveis por outros autores ou corporações conceituados na área de tecnologia e GNU/Linux.

Fontes e Recomendações de Leitura

Guia Foca GNU/Linux (guiafoca.org)
Recomendação de Leitura:
http://www.guiafoca.org

LinuxTips (linuxtips.com.br)
Recomendação de Leitura:
https://www.linuxtips.com.br/

Aurelio Jargas (aurelio.net)
Recomendação de Leitura:
aurelio.net/shell/

Luciano Siqueira (lcnsqr.com)
Recomendação da Leitura:
https://lcnsqr.com/curso-linux-essentials

Fontes e Recomendações de Cursos

The Linux Foundation (linuxfoundation.org)
Recomendação de Curso **(**Gratuito**)**:
https://www.edx.org/course/introduction-to-linux

Cisco (cisco.com)
Recomendação de Curso **(**Gratuito**)**:
https://www.netacad.com/campaign/linux-unhatched3

RedHat (redhat.com)
Recomendação de Curso **(**Gratuito**)**:
https://www.edx.org/course/fundamentals-red-hat-enterprise-linux-red-hat-rh066x

4Linux (4linux.com.br)
Recomendação de Curso **(**Gratuito**)**:
https://www.4linux.com.br/curso/linux-gratis

GLOSSÁRIO

CLI	Command Line Interface (Interface de Linha de Comando)
GNU	GNU é um sistema operacional tipo Unix.
LFC	Linux Foundation Certified
LPIC	Linux Professional Institute Certified
NIC	Network Interface Card
RH	Red Hat
RHEL	Red Hat Enterprise Linux