Manual de Sobrevivência

DICAS E COMANDOS DO MUNDO LINUX

Tales Araújo Mendonça

editora viena

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Mendonça, Tales Araújo
Manual de sobrevivência : dicas e comandos do mundo Linux / Tales Araújo Mendonça. -- Santa Cruz do Rio Pardo, SP : Editora Viena, 2005.

ISBN 85-371-0015-3

1. LINUX (Sistema operacional de computador) 2. Redes de computadores 3. UNIX (Sistema operacional de computador) I. Título.

05-8789 CDD-005.43

Índices para catálogo sistemático:

1. LINUX : Sistema operacional : Computadores : Processamento de dados 005.43

Guia Prático

Manual de Sobrevivência
Dicas e Comandos do Mundo Linux

Autor:
Tales Araújo Mendonça

Editora:
EDITORA VIENA
Rua Regente Feijó, 621 - Centro
Santa Cruz do Rio Pardo - SP
CEP 18900-000

Fone (14) 3372-2155
Home-Page: www.editoraviena.com.br
e-mail: editoraviena@editoraviena.com.br

Capa: Marcelo Gino Pereira

ISBN - 85-371-0015-3

1ª Edição - 02/2006 - SCRPardo/SP

Agradecimentos

Primeiramente gostaria de agradecer ao Ricardo Árcega, por me sugerir para que escrevesse um livro sobre uma idéia que tive de publicação de comandos, em uma forma simples, rápida e objetiva. Agradeço também ao Marcos José Setim que me deu apoio e dicas sobre o livro. Muito obrigado ao José Queiroz que além de contribuir com alguns comandos, me deu grandes dicas. As contribuições de: Alex San, Alexandre Daibert, Andrei Drusian, Carlos E. Morimoto, Eduardo C. Silva, Hugo Cisneiros, Luciano Martini, Otávio Rodolfo. Agradeço ao Guilherme Marques e Mauro Xavier por ajudar com algumas traduções. E agradeço principalmente minha mãe que sempre me deu apoio e também aos meus familiares e amigos.

Obrigado a todos, pois vocês colaboraram para que este livro ficasse pronto.

Prefácio

Este manual encontra-se dividido em três partes, para melhor entendimento do mesmo:

Capítulo 1. Dicas e Princípios Básicos do Linux

Essa primeira parte consiste em uma introdução (Conceitos básicos) do que o usuário precisa saber para seguir em frente na leitura do livro. Caso seja leigo, é imprescindível que passe pelos "Conceitos básicos". Também são abordadas algumas "dicas básicas" muito úteis que lhe ajudará no dia a dia para convivência com o Linux.

Capítulo 2. Comandos do Terminal

Para que possamos utilizar com maior facilidade o terminal de comandos do Linux (shell), saber os nomes dos comandos – às vezes, antes de saber o que eles executam – é de extrema necessidade. Também como abordagem, coloquei em prática as noções básicas descritas no Capítulo 1 deste manual.

Complementando este capítulo, no final estão organizados os comandos específicos correspondentes a cada distribuição. É uma forma mais fácil de encontrar o comando que deseja, caso o mesmo só pertença a sua distribuição.

Capítulo 3. Dicas Avançadas

Essa parte é sem dúvida a "salvação" de muitos que utilizam o Linux e querem tirar o máximo de proveito do programa mas não sabem como. Aqui abordaremos assuntos importantes sobre como compilar programas no Linux, como obter uma proteção para sua rede, compartilhar a internet com segurança, checar dispositivos(HD, CD-ROM, Disquete, etc), aprender a utilizar vários comandos em conjunto, e mais...

Sumário

1

Dicas e Princípios Básicos do Linux

1. Dicas e Princípios Básicos do Linux

1.1. Introdução

O **Linux** é um sistema operacional que possui várias funcionalidades que podem ser expressas de diversas formas. Nesse manual procuro citar os comandos e dicas mais usados, que poderão lhes auxiliar no dia-a-dia.

O **Manual de Sobrevivência**, como o próprio nome diz, refere-se a um manual de rápido auxílio, com o objetivo de ajudar e facilitar o usuário nas consultas de dicas e comandos, evitando consultas em locais diversos. Como irão ver, tentei ser o mais simples e objetivo possível para que haja um entendimento rápido e fácil.

O Manual é voltado para todos os níveis de usuários, abrangendo as distribuições mais conhecidas com comandos específicos para as mesmas.

Esse é o primeiro livro que escrevo e espero continuar com a jornada, ajudando ao máximo a comunidade **Linux**, para que haja maior interação e adesão ao software livre.

1.2. Conceitos Básicos

Antes de começar a folhear o manual é imprescindível que conheça alguns conceitos básicos que irão ajudá-lo a compreender melhor o uso do mesmo.

O que significa distribuição (distro)?

R.: É como são conhecidos os diversos "tipos de **Linux**". Como alguns exemplos podemos citar: Slackware, Debian, Gentoo, Mandrake, Fedora, Conectiva; dentre outros.

Qual distribuição devo utilizar?

R.: Quando se fala em **Linux** (Software Livre), fala-se em liberdade de escolha, em democracia. As pessoas que utilizam o **Linux**, não utilizam apenas por ser um software livre, ou mesmo por ser um excelente sistema operacional com muita segurança e poucas falhas, ou por não existir vírus ou trojams que acabam com certos sistemas proprietários, mas o utilizam também pela sua filosofia. É através dessa filosofia que nasceram e continuam nascendo todas essas várias distribuições, nos proporcionando o direito de escolha. Utilize a distro que mais lhe agrade.

O que é um sistema multiusuário?

R.: O **Linux** é um sistema multiusuário, pois permite que mais de um usuário utilize o computador ao mesmo tempo.

O que é Kernel?

R.: É o núcleo – é o cérebro e coração do sistema –, ele que controla todo o hardware. **Kernel** é o centro do sistema, todo o funcionamento da CPU e aplicativos dependem dele, mas o **Kernel** em si, sozinho, não tem utilidade, ele precisa dos aplicativos assim como os aplicativos precisam do **Kernel**, é uma relação de simbiose, um precisa do outro para "sobreviver".

O que é ambiente X?

R.: No **Linux** damos o nome à interface gráfica de "X", ou "ambiente X", ou mesmo "servidor X". Diferente de outros sistemas operacionais proprietários, no **Linux** existem vários ambientes gráficos ou DM (DESKTOP MANAGER) – os mais conhecidos são: kde, gnome e XFCE – e gerenciador de janelas ou WM (WINDOWS MANAGER) – os mais conhecidos são: icewm, windowmaker, fluxbox, dentre outros – que não são acoplados ao **Kernel**, ou seja, você usa o que lhe agrada, de acordo com suas necessidades, e o que sua máquina agüenta. Os DM's, são ambientes mais completos e pesados, necessitam de um hardware melhor; já os WM's, são gerenciadores leves, e necessitam de menos hardware, rodando em máquinas mais fracas.

O que são terminais ou consoles?

R.: O **Linux** é um sistema em que o usuário pode fazer múltiplas tarefas ao mesmo tempo em locais distintos, estes locais são chamados de terminais ou consoles. Para acessar um terminal, basta teclar CTRL+ALT+Fn, sendo "n" o número correspondente ao terminal que deseja abrir, podendo ser de 1 a 6 – terminais em modo texto – e 7 a 12 – terminais gráficos. O 7 é tido como o terminal padrão para o modo gráfico (onde é carregado o ambiente gráfico padrão).

O que é super usuário (root)?

R.: É o usuário que apresenta todo poder sobre a máquina, ele possui todos os direitos como: criar, apagar, executar, dar ou tirar permissão de qualquer arquivo ou diretório. Se você é leigo, tenha muito cuidado ao realizar uma tarefa que exige ser executada pelo root, pois pode danificar todo o funcionamento do sistema.

Dica: *Crie um usuário comum para realizar todos os processos da máquina e, quando precisar executar algo que necessite do root – como instalar programas e configurar o sistema –, faça o login como root, execute as devidas tarefas e retorne para o seu usuário. Assim não terá problemas de segurança, evitando danificar o sistema.*

O que é Case Sensitive?

R.: O **Linux** é um sistema **Case Sensitive**, isso é, ele diferencia letras maiúsculas de minúsculas. O comando 'clear' (usado para limpar a tela do terminal de comandos) é diferente de 'Clear' ou 'CLEAR', sendo que os dois últimos não fazem parte do sistema.

O que são diretórios ocultos e para que servem?

R.: Diretórios ocultos, são todos os diretórios cujo nome começa com um ponto (.), exemplo: "**.kde**" é um diretório oculto que se encontra no diretório "**home**" do usuário. Os diretórios ocultos servem para guardar configurações, arquivos temporários, travas (locks), PIDs, enfim, qualquer coisa que você não queira que o usuário fique mexendo. A mesma idéia vale para os arquivos ocultos.

Qual pacote corresponde a minha distribuição?

R.: Como o **Linux** é um sistema que possui várias distribuições (modelos, vulgarmente dizendo), as principais e mais importantes possuem o seu próprio pacote – é um conjunto de arquivos comprimidos que compõem um programa ou biblioteca, otimizado para uma distribuição – que podem ser instalados facilmente. Veja abaixo como saber qual pacote corresponde a sua distro, para que possam ser instalados:

Pacotes com terminação "**.rpm**" são derivados de distribuições Red Hat, Fedora, Conectiva, Mandrake, Suse, etc, e são instaláveis nas mesmas.

Pacotes com terminação "**.deb**" são derivados de distribuições Debian, Ubuntu, Kurumin, Kalango, ... e são instaláveis nas mesmas.

Pacotes com terminação "**.tgz**" são derivados da distribuição Slackware, e são instaláveis na mesma.

Obs 1.: *Os arquivos que encontram-se com terminações ".tar.gz" e ".tar.bz2" – também conhecidos como arquivos fontes ou sources – estão no formato de compressão, compactados, e podem ser instalados (compilados) em todas as distribuições.*

Obs 2.: *Caso o seu pacote só exista para uma determinada distribuição, você poderá utilizar um programa chamado "alien", que converte o pacote para a distribuição desejada. Para informações de como utilizar o comando, consulte o capítulo 2 do manual que trata dos comandos.*

O que é comentar uma linha?

R.: Na forma mais simples de explicar, seria dizer que é impedir que uma linha de comando seja executada. Isso pode ser feito colocando o sustenido (#) na frente da linha cuja execução se deseja anular.

É muito usado nos arquivos de configuração para adicionar informações que não serão executadas, como por exemplo o nome do autor do arquivo, informações de como proceder para configurar o arquivo, ou mesmo comentar uma linha de comando.

1.2.1. Trabalhando com a Tecla TAB

Tudo que se pensa em facilidade e inovação está presente no **Linux**. Uma dessas facilidades é a utilização da tecla "TAB" no terminal de comandos, além de suas diversas funcionalidades, ela serve também para completar nomes – comandos, arquivos e diretórios.

Abra um terminal de comandos e tecle TAB duas vezes: aparecerá todas as possibilidades de comandos e programas que poderão ser executados. Por exempo, digite "/h" e logo depois tecle TAB.

```
$ /h[tecle TAB]
$ /home/
```

Note que ele completa a palavra para "/home/". Caso não saiba ou tenha esquecido o nome de um comando/programa, pode ser utilizada a tecla TAB para ajudá-lo a lembrar.

1.2.2. Permissões

Existem basicamente três partes que compõem um arquivo/diretório e podem ser reconhecidas como: dono, grupo e outros. Podendo cada um deles obter permissão total (rwx).

```
dono = rwx
grupo = rwx
outros = rwx
r = read (ler)
w = write (gravar)
x = execute (executar)
```

A visualização das permissões podem aparecer de duas formas:

- Quando for um diretório, será apresentado a letra "d" na frente das permissões, como é mostrado abaixo:

```
drwxrwxrwx
```

- Quando for um arquivo, as permissões serão apresentadas sem a letra "d", e no lugar entrará um traço (-), como é mostrado abaixo:

```
-rwxrwxrwx
```

Cada letra significa um número correspondente a sua permissão.

```
r = 4
w = 2
x = 1
```

Se somarmos os três números vamos obter 7, que significa permissão total, ou seja, o usuário poderá ter o direito de ler, gravar e executar.

Exemplo de permissões:

```
-rw-r--r-- (644)
-rwxr-xr-x (755)
-rwxrwxrwx (777)
```

1.2.3. Conhecendo o Prompt de Comandos

Como exemplo vamos pegar a linha que segue abaixo:

```
tales@horus:~$
```

Essa linha aparece toda vez que é aberto um terminal de comandos ou depois de logar como um usuário. Todos os comandos que forem digitados irão aparecer logo na frente do símbolo cifrão ($) – se for um usuário comum –, ou do símbolo sustenido (#) – se for o root. Abrindo a linha acima temos o seguinte:

O primeiro nome (tales) indica o usuário que está logado, caso estivesse logado como root, iria aparecer...

```
horus:~#
```

O símbolo arroba (@) que aparece na frente do nome tales, significa usuário em host – no caso o host é horus.

O segundo nome (horus) indica o nome da máquina.

O símbolo til (~) significa que o diretório onde o usuário se encontra é o home. Ao invés de escrever /home/tales, você pode digitar apenas ~tales.

1.2.4. Conhecendo a Estrutura de Diretórios

No **Linux**, toda estrutura de diretórios começa no barra (/), que significa início. A estrutura de diretórios é muito simples de ser entendida, pois segue o mesmo padrão das páginas na internet, ou seja, um determinado site possui o seguinte endereço que leva até a imagem "**foto.jpg**": www.site.com/imagens/

foto.jpg. No **Linux**, eu poderia criar um diretório no barra (/) chamado "**imagens**" e colocar o arquivo dentro dele que ficaria da seguinte maneira:
/imagens/foto.jpg.

Note que em ambos aparece uma estrutura de diretórios similar.

Irei explicar de forma simples e rápida, os diretórios que procedem o diretório barra (/) e o que significam. Lembrando que o **Linux** possui muito mais subdiretórios do que os relatados aqui. Apenas destaquei os mais importantes e conhecidos.

Diretórios que são padrão em todas as distros:

```
/bin, /etc, /home, /lib, /usr, /dev, /sbin, /tmp e /var
```

Diretórios opcionais:

```
/mnt, /sys, /boot, /proc e /opt
```

Descrição dos diretórios:

/bin ——— *Guarda alguns executáveis do sistema, como por exemplo: ln, mkdir, cat, rm, mount, ...*

/etc ——— *Guarda todos os arquivos de configuração do sistema.*

/home ——— *Por ser padrão, é onde se encontram todos os usuários e suas configurações. Por exemplo, o usuário 'tales' terá todos os seus arquivos e suas configurações gravadas dentro do diretório /home/tales que também pode ser representado como ~tales.*

/lib ——— *Guarda bibliotecas, arquivos não executáveis necessários para iniciar o sistema, e bibliotecas necessárias para rodar outras aplicações além do módulo do **Kernel**.*

/mnt ——— *Geralmente utilizado para montar dispositivos, como CD-ROM, disquete, HD,... O /mnt é como um diretório padrão para montagem de dispositivos, mas nada impede que se monte, por exemplo, o CD-ROM em /cdrom, e assim por diante.*

/sys ——— *Representa os objetos internos do **Kernel**. Contém arquivos que tem ligações com drivers do **Kernel**.*

/usr ——— *Neste diretório encontra-se grande parte do **Linux**, nele estão programas, janelas gráficas, bibliotecas, fontes do **Kernel** etc. Guarda comandos que são de uso dos usuários em geral.*

/usr/bin ——— *Geralmente, guarda os executáveis e links de executáveis de programas que são instalados no sistema.*

/usr/lib ——— *Diretório onde se encontra as bibliotecas utilizadas pelos programas.*

/usr/local ——— *Utilizado para instalar programas que não fazem parte do sistema, ou customizações de programas que fazem parte do sistema.*

/usr/sbin ——— *Contém muitos programas binários que são utilizados pelo sistema.*

/usr/share ——— *Contém arquivos de configuração e gráficos para muitos aplicativos de usuários.*

/usr/src ——— *Contém arquivos de código fonte do sistema, incluindo o **Kernel** do **Linux**.*

/boot ——— *Guarda informações necessárias para carregar o boot do sistema. É o lugar onde o **Kernel** é mantido.*

/dev ——— *Guarda informações de todos os dispositivos que estão disponíveis no sistema. Contém os arquivos desses dispositivos.*

/proc ——— *É o monitor do sistema, além de mostrar o estado dos componentes, serve para ajustes finos no coração do sistema.*

/sbin ——— *Contém ferramentas para administração e configuração do sistema. Geralmente o acesso a esse diretório é de exclusividade do super usuário (root).*

/tmp ——— *Guarda arquivos temporários de programas.*

/var ——— *Guarda arquivos de informação de trabalho, como logs, caches, spoolers e locks.*

/opt ——— *Armazena programas que não utilizam o padrão do sistema e precisam compartilhar arquivos para vários usuários. Programas em que bibliotecas, documentações e binários ficam no mesmo diretório. Algumas distribuições optam por deixar este diretório no modo "leitura/escrita" para todos os usuários, funcionando assim como um diretório compartilhado entre todos.*

Observações importantes: *Sempre que estiver com dúvidas de como utilizar um comando e para saber de todas as opções que dispõem, utilize o manual (man) ou o help – mostra as opções a serem utilizadas de forma simplificada e mais rápida.*

Exemplo:

```
$ mv --help
```

Mostra informações simplificadas de como utilizar o comando "mv".

Para fazer uma busca pelo manual (man), tecle barra (/) dentro do manual + nome a ser procurado e tecle [Enter] para concluir a busca.

Exemplo:

```
$ man mv
/renomeia [Enter]
```

Entra no manual do comando "mv" e procura a palavra "renomeia", caso encontre uma ou mais palavras com este nome, irá marcar todas com uma seleção para destaque do texto.

1.3. Desktop

1.3.1. Reiniciando o X

Quando se executa alguma tarefa no ambiente gráfico que precisa reiniciar o servidor X, o mesmo pode ser feito sem precisar reiniciar o sistema (**Linux**). Reiniciar o X nada mais é do que sair e retornar ao ambiente gráfico. Todos os serviços que foram iniciados durante o boot, não sofrerão qualquer alteração.

Exemplo de utilização:

Tecle **CTRL**+**ALT**+**BACKSPACE** para reiniciar o X.

1.3.2. Copiar e Colar com Seleção do Mouse

Umas das grandes utilidades e facilidades que existem no **Linux** é a opção de selecionar e colocar com o auxílio do mouse. Se você achava fácil usar o **CTRL**+**C** (copiar) e **CTRL**+**V** (colar), no **Linux** isso fica ainda mais fácil.

Exemplo de utilização:

- Para mouse de 2 botões: Selecione o conteúdo que deseja copiar com o botão esquerdo, vá para a área que deseja colar e aperte os 2 botões juntos.
- Para mouse de 3 botões: É utilizado o mesmo processo. Apenas para colar, utilize o botão do meio.

1.3.3. Alternar entre Áreas de Trabalho

Essa dica é válida apenas para quem utiliza o ambiente gráfico **KDE**. Dentro de seu desktop podem ter várias áreas de trabalho virtual – o que ajuda a não acumular muitas janelas em um único local – que podem ser alternadas facilmente sem a utilização do mouse, apenas com 2 teclas, agilizando o trabalho.

(Contribuição, Alexandre Daibert)

Exemplo de utilização:

Segure a tecla **CTRL**, em seguida aperte **TAB** para alternar entre as áreas de trabalho.

1.3.4. Habilitar o Botão PrintScreen

Essa dica é válida apenas para quem utiliza o ambiente gráfico KDE. Facilita muito quando é preciso tirar um screenshot da tela e o mesmo pode ser feito apenas apertando uma tecla.

(Contribuição, Alex Sander C. Moraes)

Exemplo de utilização:

Tecle **ALT**+**F2**, digite "kcontrol" e mande executar. Entre em "Regional & Acessibilidade", "Atalhos de Teclado", "Atalhos de Comando", "Gráficos", "KSnapshot". Na tela, aparecerá abaixo: "Atalho Para Comando Selecionado", marque a opção "Personalizar", aparecerá uma janela. Aperte no teclado o botão "PrintScreen" e a janela se fechará. Para terminar, clique no botão aplicar.

Obs.: *Dentro de "Gráficos", em determinadas distribuições, o Ksnapshot encontra-se em um submenu.*

1.3.5. Executando Comandos pelo Desktop

Dica válida apenas para os ambientes **KDE** e **Gnome**.

Existem atalhos muito úteis que possibilitam a execução de programas através do desktop. Essa é uma dica simples, mas útil para quando precisar executar uma aplicação rapidamente.

Exemplo de utilização:

Tecle **ALT**+**F2**: será aberta uma caixa para você digitar a aplicação que deseja executar.

Exemplos:

xmms ———— *Abre o player de som.*

kedit———— *Abre o editor de texto simples.*

konsole ———— *Abre um terminal de comandos.*

1.4. Terminal

1.4.1. Executando Programas em Background e Deixando o Terminal de Comandos Livre

Essa dica é útil para quando precisar abrir um programa ou vários, ou então precisa digitar vários comandos e ao mesmo tempo deixar o terminal livre para trabalhar.

Para a utilização, basta acrescentar o e-comercial (&) no final do comando.

Exemplo de utilização:

```
$ kwrite &
```

Abre o programa **kwrite** deixando o terminal livre para futuros comandos, possibilitando a abertura de novos programas.

```
$ kwrite & xmms & amsn &
```

Abre os programas **kwrite**, **xmms** e **amsn**. Aqui você pode, por exemplo, programar no kwrite, ouvir música no xmms e conversar com amigos através do amsn, e ainda ter o terminal de comandos livre para trabalhar.

1.4.2. Procurando por Comandos Digitados

Essa dica é simples porém muito útil. Possibilita encontrar comandos que foram digitados no terminal caso não se recorde do comando por completo.

Essa dica pode ser utilizada para evitar a digitação de grandes comandos que não são lembrados por completo.

Exemplo de utilização:

Abra um terminal de comandos e tecle **CTRL**+**R**, irá aparecer o seguinte:

```
(reverse-i-search)`': 
```

Comece teclando, irá aparecer os comandos que já foram executados anteriormente no sistema.

```
(reverse-i-search)`p': history |grep wget
```

No caso acima, foi digitado a letra "p" e apareceu

```
history |grep wget
```

Vejamos o que acontece se continuar digitando mais letras.:

```
(reverse-i-search)`ps': ps aux|grep amsn
```

Posterior a letra "p" foi teclado "s" e apareceu o comando

```
ps aux|grep amsn
```

Para utilizar o comando que aparece na tela, basta teclar [**Enter**].

Obs.: *Lembrando que todos os comandos que aparecem foram executados anteriormente no sistema.*

1.4.3. Executando Arquivos .bin, .run e .sh

Muitas pessoas têm dificuldades para instalar determinados programas, pois não sabem como executá-los. Irei mostrar como executar arquivos "**.bin**", "**.run**" e "**.sh**", pois todos os três tipos seguem o mesmo processo para serem executados, dando início a instalação dos mesmos. Os arquivos com extensões "**.bin**" e "**.run**" estão compactados e a instalação é nada mais do que descompactar os arquivos em determinados diretórios do sistema. Já o arquivo com extensão "**.sh**" é um script e irá executar os comandos para os quais foi programado.

Exemplo de utilização:

```
# chmod +x NVIDIA-Linux-x86-xxx.run
# ./NVIDIA-Linux-x86-xxx.run
```

Primeiro foi fornecido ao arquivo a permissão de execução (x) e logo em seguida o arquivo foi executado com um ponto barra (./) + nome do arquivo.

```
# chmod +x IRPFJavaxxxlinuxv1.1.bin
# ./IRPFJavaxxxlinuxv1.1.bin
```

Foi feito o mesmo processo do exemplo acima, a diferença está apenas no tipo de arquivo que foi executado, no caso um "**arquivo.bin**".

```
$ chmod +x xfck.sh
$ ./xfck.sh
```

Como nos outros dois exemplos, esse procedeu da mesma forma, o script "**xfsck.sh**" recebeu permissão de execução e logo em seguida foi executado.

1.4.4. Tirar Screenshot da Tela

Possibilita tirar um screenshot de seu desktop para que seja mostrado para os seus amigos ou mesmo para trabalhar.

Exemplo de utilização:

```
$ sleep 5 && import -w root imagem.png
```

O comando acima, irá tirar uma "foto" de seu desktop em um intervalo de tempo de 5 segundos e será salvo com o nome de "**imagem.png**".

2
Comandos do Terminal

2. Comandos do Terminal

2.1. adduser

Utilizado para adicionar usuários ou grupos no sistema de acordo com as opções especificadas. Para criação de usuários, cada um terá um diretório particular com o nome especificado dentro de **/home** e todas as configurações do mesmo ficarão guardadas dentro desse diretório: "**~/usuário**".

Sintaxe:

1. usual:

```
# adduser [opções] (usuário)
```

2. grupo:

```
# adduser --group [opções] (grupo)
```

3. grupo/usuário:

```
# adduser [opções] (grupo) (usuário)
```

Opções:

- **--conf arquivo** ——— *Esta opção faz com que o **adduser** utilize outro arquivo ao invés do "**/etc/adduser.conf**".*
- **--group** ——— *Cria grupos similares ao addgroup.*
- **--system** ——— *Faz com que o comando crie apenas usuários e grupos que estejam no sistema local.*

Exemplo de utilização:

- Para criar o usuário "mariana":

```
# adduser mariana
```

- Para criar o grupo "turismo":

```
# adduser --group turismo
```

Obs.: *Depois de criar um novo usuário é importante que se defina uma senha, para isso utilize o comando passwd.*

Para mais informações consulte o manual:

```
$ man adduser
```

2.2. alias

Utilizado para substituir um comando e seus parâmetros por palavras-chave.

Sintaxe:

```
$ alias [nome='comando']
```

Exemplo de utilização:

- Para exibir todos os alias criados para o usuário:

```
$ alias
```

- Para criar um alias chamado "ls", atribuindo os valores "ls --color=auto" (colore arquivos e diretórios na saída do ls).

```
$ alias ls='ls --color=auto'
```

- Para criar um alias chamado "azureus", apontando para o caminho "**/servidor/programas/azureus/./azureus**":

```
$ alias azureus='/servidor/programas/azureus/./azureus'
```

Para mais informações consulte o manual:

```
$ man alias
```

2.3. alien

Conversor de pacotes. O **alien** converte pacotes RPM (Red Hat), DEB (Debian), TGZ (Slackware), PKG (Solaris) e SLP (Stampede Linux) para qualquer formato entre eles. Se você precisou instalar um programa para sua distro e só encontrou o pacote para uma outra, então utilize esse programa que irá converter o pacote para sua distro.

Obs.: *Não é recomendável converter pacotes ligados ao sistema, como pacotes que contém bibliotecas.*

Sintaxe:

```
# alien [opções] [pacote]
```

Exemplo de utilização:

- Para converter o "**pacote.deb**" para pacote.rpm:

```
# alien -r pacote.deb
```

- Para converter o "**pacote.rpm**" para pacote.deb:

```
# alien -d pacote.rpm
```

- Para converter o "**pacote.deb**" para "**pacote.tgz**":

```
# alien -t pacote.deb
```

- Para converter o "**pacote.rpm**" para "**pacote.pkg**":

```
# alien -p pacote.rpm
```

- Para instalar o pacote automaticamente após ser gerado, e remover após a instalação. Se o tipo de pacote não for especificado, será convertido para o padrão no formato DEB:

```
# alien -i pacote.tgz
```

Para mais informações consulte o manual:

```
$ man alien
```

2.4. apt-get

O **apt-get** é um gerenciador de pacotes, com ele você pode instalar, remover e atualizar qualquer programa em seu sistema. De todos os gerenciadores de pacotes o **apt-get** está entre os melhores, pela sua facilidade e eficiência. Basicamente ele é utilizado para baixar e instalar programas da internet, remover e atualizar o sistema. O **apt-get** possui uma grande vantagem sobre os outros sistemas, pois instala os pacotes desejados e também suas dependências.

Obs.: *Os desenvolvedores do Debian recomendam utilizar o comando "aptitude" no lugar de "apt-get" por diversas razões; dentre elas: melhor gerenciamento de pacotes, acompanha a atualização dos pacotes automaticamente, possui uma interface em modo texto amigável e poderosa que pode ser executada por usuários comuns, evitando danificar o sistema, etc.*

Sintaxe:

```
# apt-get [comando] [pacote]
```

Exemplo de utilização:

- Para atualizar a lista de pacotes:

```
# apt-get update
```

- Para atualizar todos os pacotes instalados no sistema:

```
# apt-get upgrade
```

- Para instalar um ou mais pacotes:

```
# apt-get install [nome do pacote]
```

- Para remover um ou mais pacotes:

```
# apt-get remove [nome do pacote]
```

- Para fazer download de um arquivo sem instalá-lo no sistema. Após o download, o arquivo se encontra no diretório **/var/cache/apt/archives**:

```
# apt-get install -d [nome do pacote]
```

- Para apagar arquivos baixados para instalação:

```
# apt-get clean
```

- Para arrumar possíveis problemas, caso o **apt-get** esteja travado, não possibilitando a instalação de novos programas:

```
# apt-get install -f
```

Dica: *Para incluir ou excluir novas fontes, você deve editar o arquivo ("/etc/apt/sources.list") como root e acrescentar as informações de acordo com a sua utilidade. Novas fontes podem ser encontradas no site:* ***http://apt-get.org/***

Para encontrar a fonte do programa desejado, basta clicar no link "**Search for a package**", inserir o nome do programa no campo e escolher o tipo de arquitetura. Caso não saiba a arquitetura utilizada em sua máquina, deixe em "**i386**" e clique em "**Enviar Dado**s". Irá aparecer uma lista com várias fontes disponíveis, escolha a que melhor te satisfaz e inclua dentro do arquivo "**sources.list**". Abaixo segue o exemplo de um "sources.list":

```
# Sarge
# Os endereços abaixo contém endereços dos mirros do Debian
Sarge.
deb http://ftp.br.debian.org/debian sarge main contrib non-free
# Sarge/non-US
# Os mirros abaixo contém pacotes do Sarge que não podem ser-
# distribuídos nos EUA devido às leis do país
deb http://ftp.br.debian.org/debian-non-US sarge/non-US main
contrib non-free
# KDE 3.4.1 - Servidor não oficial
deb http://pkg-kde.alioth.debian.org/kde-3.4.1/ ./
# Pacote oficial do navegador Opera
deb http://deb.opera.com/opera/ testing non-fre
```

Para mais informações consulte o manual:

```
$ man apt-get
```

2.5. arch

Mostra qual a arquitetura de seu PC, ou seja, o tipo de processador de sua máquina.

Alguns tipos de arquiteturas:

- i386
- i586
- i686

Sintaxe:

```
$ arch
```

Exemplo de utilização:

```
tales@horus:~$ arch
i686
```

Foi exibida a arquitetura (**i686**) do sistema.

O "i" que aparece na frente dos números, significa interface.

Para mais informações consulte o manual:

```
$ man arch
```

2.6. cal

Exibe um calendário simples no formato tradicional, oferece vários formatos diferentes de datas. Podem ser utilizados alguns parâmetros para exibição de datas.

Sintaxe:

```
$ cal [opções] [mês] [ano]
```

Exemplo de utilização:

- Para exibir o calendário do mês atual de forma simples:

```
$ cal
```

- Para exibir o calendário em linha única contendo o mês anterior, mês atual e o próximo mês:

```
$ cal -3
```

- Para exibir um calendário do mês de agosto do ano de 2007:

```
$ cal 8 2007
```

- Para exibir o calendário do ano atual:

```
$ cal -y
```

Para mais informações consulte o manual:

```
$ man cal
```

2.7. cat

Mostra o conteúdo de um arquivo, geralmente arquivos de texto. É recomendável usá-lo para arquivos de texto pequeno, pois os arquivos com grande conteúdo são mais difíceis de serem visualizados, o texto rolará dificultando a leitura. O comando também pode ser usado para criação de pequenos arquivos e inserção do conteúdo de texto de um arquivo para dentro de outro arquivo.

Sintaxe:

- Para criar um arquivo:

```
$ cat > [nome do arquivo]
```

- Para visualizar um arquivo:

```
$ cat [nome do arquivo]
```

Exemplo de utilização:

- Para criar o arquivo "**manual.txt**":

```
tales@horus:~$ cat > manual.txt
Aqui você escreve o conteúdo que desejar. Quando acabar de
escrever o seu texto, basta digitar CRTL+C e o conteúdo estará
alvo dentro do arquivo "manual.txt".
```

- Para visualizar o arquivo "**manual**.txt":

```
tales@horus:~$ cat manual.txt
Aqui você visualizará o que foi salvo dentro do arquivo
"manual.txt".
```

- Para inserir o conteúdo do arquivo "**texto1.txt**" no final do arquivo "**texto2.txt**".

```
$ cat texto1.txt >> texto2.txt
```

Para mais informações consulte o manual:

```
$ man cat
```

2.8. cd

Muda a localização na árvore de diretórios. É considerado um dos comandos mais simples e mais essenciais.

Sintaxe:

```
$ cd [diretório que deseja acessar]
```

Exemplo de utilização:

- Para entrar no diretório "**Documentos**":

```
tales@horus:~$ cd Documentos/
tales@horus:~/Documentos$
```

- Para sair do diretório "**Documentos**":

```
tales@horus:~/Documentos$ cd ..
tales@horus:~$
```

Nota: *O comando "cd" sem nenhum parâmetro, retorna para o diretório de origem do usuário.*

```
tales@horus:~/Documentos/Manual$ cd
tales@horus:~$
```

- Para retornar dois diretórios:

```
tales@horus:~/Documentos/Manual$ cd ../..
tales@horus:~$
```

Obs. 1: *Para retornar um diretório utilize: "cd ..", dois diretórios utilize: "cd ../..", três diretórios utilize: "cd ../../..", e assim por diante. Use o comando sem aspas ("").*

Obs. 2: *Sempre que precisar entrar em um diretório que contenha espaço no nome e a tecla **TAB** não der conta, utilize aspas dupla ("") ou a barra invertida (\) entre os espaços para entrar.*

Exemplo 1 – Aspas dupla("")

```
tales@horus:~/Documentos$ cd "Arquivos Compartilhados"
tales@horus:~/Documentos/Arquivos Compartilhados$
```

Exemplo 2 – Barra invertida(\)

```
tales@horus:~/Documentos$ cd Arquivos\ Compartilhados
tales@horus:~/Documentos/Arquivos Compartilhados$
```

Para mais informações consulte o manual:

```
$ man cd
```

2.9. checkinstall

Utilizado para facilitar a instalação e remoção de programas compilados. Pode ser gerado para distribuições que utilizam pacotes(**.tgz, .rpm e .deb**). O comando é utilizado no lugar do "**make install**".

Sintaxe:

```
# checkinstall [opção] [comando]
```

Opções:

-S ——— *Constrói um pacote Slackware (tgz).*

-R ——— *Constrói um pacote RedHad (rpm).*

-D ——— *Constrói um pacote Debian (deb).*

-y ——— *Aceita as respostas padrão para todas as perguntas.*

Exemplo de utilização:

- Após o comando "**make**", entre com o **checkinstall** e gere um pacote ".deb", para a distribuição **Debian** e seus derivados:

```
# checkinstall -D
```

Obs.: *Se nenhuma opção for especificada, o pacote criado será o de utilização do sistema e mais um pacote tgz, ou seja, se o sistema for um* ***Debian****, será criado um pacote.deb e um pacote.tgz.*

Após digitar o comando, irá abrir uma tela pedindo que entre com as informações para a configuração do pacote. Quando terminar, o mesmo estará instalado no sistema e o pacote será criado no mesmo diretório em que foi compilado.

Para mais informações consulte o manual:

```
$ man checkinstall
```

2.10. chmod

Altera a permissão de arquivos e diretórios.

Sintaxe:

```
$ chmod [opções] [nome do arquivo/diretório]
```

Exemplo de utilização:

```
$ chmod +x arquivo.bin
```

No comando acima, o "**arquivo.bin**" está recebendo a permissão de execução.

```
$ chmod 644 arquivo.txt
```

No comando acima, o "**arquivo.txt**" recebe permissão de ler e gravar (6) para o dono, e de leitura (4) para o grupo e outros. A opção acima é muito utilizada em arquivos.

```
$ chmod 755 diretorio3
```

No comando acima, o "**diretório3**" recebe permissão total (7) para o dono, leitura e execução (5) para o grupo e outros.

Para mais informações consulte o manual:

```
$ man chmod
```

2.11. chown

Altera o dono e o grupo de um arquivo/diretório.

Sintaxe:

```
$ chown [nome do usuário]:[nome do grupo]
```

Exemplo de utilização:

```
# chown rag:tales ~rag/
```

No exemplo acima, o diretótio "**~rag/**" está recebendo acesso para o usuário "rag" e o grupo "tales"

```
# chown -R tales:users ~tales/Documentos/
```

No exemplo acima, o diretório "**Documentos**" e todo o seu conteúdo, como arquivos e diretórios, estão recebendo acesso para o dono "tales" e grupo "users". A opção "**-R**" é recursiva, ou seja, altera a permissão do diretório e de seu conteúdo.

Para mais informações consulte o manual:

```
$ man chown
```

2.12. chroot

Faz com que um programa pense que uma determinada pasta é o diretório raiz do sistema, de modo que você possa fazer qualquer tipo de alteração ou instalação, tudo a partir deste diretório como se fosse o raiz.

(Contribuição, Eduardo C. Silva)

Sintaxe:

```
# chroot [diretório] [comando]
```

Obs.: *É preciso que a partição esteja montada para que possa ser executado o comando "chroot".*

Exemplo de utilização:

O comando **chroot** pode ser usado para recuperar o boot do **Linux**, caso o tenha perdido instalando outro **Sistema Operacional.** Para gravar o **Lilo** novamente caso o tenha perdido, você pode bootar por um liveCD (**Kalango, Kurumin, Ubuntu**, ...) montar a partição "/" e "**/boot**". Após assumir a partição montada como root, execute o **chroot** para acesso ao sistema.

```
# chroot /mnt/gentoo /bin/bash
```

No comado acima, o ponto de montagem do "/" foi alterado para o "**/mnt/gentoo**". Feito isto, terá assumido a pasta "**/mnt/gentoo**" como diretório raiz, bastando gravar o **Lilo** novamente.

Para mais informações consulte o manual:

```
$ man chroot
```

2.13. clear

Limpa a tela do terminal de comandos.

Exemplo de utilização:

```
$ clear
```

2.14. crontab

O "**cron**" é um programa de "agendamento de tarefas". Com ele você pode programar para executar qualquer coisa numa certa periodicidade ou até mesmo em um exato dia, numa exata hora.

(Descrição, Hugo Cisneiros)

Sintaxe:

```
# crontab [-u usuário] [opção ou arquivo]
```

Exemplo de utilização:

```
$ crontab -e
0 22 * * 0,3,6 ~/script.sh
```

"**crontab -e**" edita o arquivo de configuração do usuário atual. Para entrar no modo de escrita, tecle "**i**", para sair e salvar tecle "**ESC, shift+z+z**".

O comando inserido no **crontab** irá executar o **script** às 22 horas nos dias da semana(domingo -0-, terça -3-, quinta -6-) que seguem os intervalos 1, 2 e 3.

```
$ crontab -l
```

Exibe o conteúdo do **crontab** do usuário atual.

```
$ crontab -r
```

Remove o **crontab** do usuário atual.

Para mais informações consulte o manual:

```
$ man crontab
```

2.15. cp

Copia arquivos e diretórios. Podem ser copiados vários arquivos e/ou diretório para um diretório indicado.

Sintaxe:

```
$ cp [opção] [origem] [destino]
```

Exemplo de utilização:

- Para copiar os arquivos 1,2 e 3 para o diretório "**/tmp**":

```
$ cp arquivo1 arquivo2 arquivo3 /tmp
```

- Para copiar o arquivo 1 e diretórios 1 e 2 para o diretório "**/tmp**" preservando todas as permissões (usuário, data, hora,...):

```
$ cp -a arquivo1 diretorio1 diretorio2 /tmp.
```

- Para forçar a cópia do "**arquivo7**" para o diretório "**/tmp**":

```
$ cp -f arquivo7 /tmp
```

- Para copiar o "diretorio3" recursivamente para o diretório "**/tmp**":

```
$ cp -r diretorio3 /tmp
```

- Para criar um link simbólico do arquivo27.txt dentro do diretório "**/tmp**":

```
$ cp -s ~/arquivo27.txt /tmp
```

- Copia o "**arquivo1.txt**" para o diretório "**/tmp**" somente se o arquivo de origem (**arquivo1.txt**) for mais recente que o arquivo de destino (**arquivo1.txt**, caso haja), ou se não existir o arquivo de destino.

```
$ cp -u arquivo1.txt /tmp
```

Para mais informações consulte o manual:

```
$ man cp
```

2.16. date

Exibe e configura a data e a hora do sistema. Pode-se escolher vários formatos distintos para serem exibidos.

Sintaxe:

```
$ date [opções] [+formato]
```

Exemplo de utilização:

```
$ date -r arquivo.txt
```

O comando acima mostra a data em que o "**arquivo.txt**" sofreu sua última atualização.

```
$ date +%A\ %d/%m/%Y\ %H:%M
quinta 09/12/2004 16:13
```

Explicando o comando acima, temos:

%A ——————— *Exibe o dia da semana.*

%d/%m/%Y ——————— *Exibe na seqüência: dia do mês, mês e ano.*

%H:%M ——————— *Exibe na seqüência: hora e minuto.*

No lugar de **barra invertida+espaço**(\ ), pode ser utilizado **porcentagem+letra t (%t)** que corresponde ao [TAB]. Veja o exemplo abaixo:

```
$ date +%A%t%d/%m/%Y%t%H:%M
quinta 09/12/2004 16:22
```

Para mais informações consulte o manual:

```
$ man date
```

2.17. deluser

Remove um usuário ou grupo do sistema.

(Contribuição, Luciano Martini)

Sintaxes:
1- usual:

```
# deluser [opções] (usuário)
```

2- grupo:

```
# deluser --group [opções] (grupo)
```

3- grupo/usuário:

```
# deluser [opções] (grupo) (usuário)
```

Opções:

--conf arquivo ——— *Esta opção faz com que o "**deluser**" utilize outro arquivo ao invés do **/etc/deluser.conf**.*

--group ——— *Apaga grupos, similar a **delgroup**.*

--system ——— *Faz com que o comando apague apenas usuários e grupos que estejam no sistema local.*

Exemplo de utilização:

```
# deluser --group printer
```

Apaga o grupo printer.

```
# deluser luciano
```

Apaga o usuário luciano.

Para mais informações consulte o manual:

```
$ man deluser
```

2.18. dd

Converte e copia arquivos. Na verdade esse comando é um verdadeiro canivete suíço, pois ele é capaz de gerar imagens de arquivo **.iso**, espelhamento de uma partição, converte arquivos – por exemplo passa todo o conteúdo de um arquivo de letras minúsculas para maiúsculas, ou vice versa - entre outras coisas.

Sintaxe:

```
$ dd [if=origem] [of=destino]
```

Exemplo de utilização:

```
# dd if=/dev/hda of=/dev/hdb
```

No exemplo acima, será criado um espelho idêntico do "**hda**" (HD primário master) para "**hdb**" (HD primário slave). A cópia é feita bit a bit, ou seja, não importa o sistema do arquivo do **hdb** (destino) pois tudo será copiado identicamente ao **hda** (origem).

```
# dd if=/dev/hda of=imagem.img
```

No comando acima, será criado um arquivo no mesmo diretório (**hda**) chamado "**imagem.img**". Dentro deste arquivo haverá uma cópia de todo o conteúdo do "**hda**".

```
# dd if=imagem.img of=/dev/hda
```

O comando acima, restaura a "**imagem.img**" do "**hda**" para o mesmo.

```
$ dd if=/dev/cdrom of=/tmp/arquivo.iso
```

No comando acima, será gerado um "**arquivo.iso**" a partir do conteúdo de um CD e o arquivo será salvo dentro do diretório /tmp.

```
$ dd if=texto1.txt of=texto2.txt conv=ucase
```

O comando acima, irá converter todo o conteúdo do arquivo "**texto1.txt**" para letras maiúsculas, que será gerado no arquivo "**texto2.txt**".

```
$ dd if=texto2.txt of=texto1.1.txt conv=lcase
```

O comando acima é o oposto do comando anterior, ou seja, converte todo o arquivo "**texto2.txt**" para letras minúsculas, que será gerado no arquivo "**texto1.1.txt**".

Para mais informações consulte o manual:

```
$ man dd
```

2.19. df

Mostra o espaço de disco usado pelo sistema de arquivos de todos os tipos, atualmente montados.

Sintaxe:

```
$ df [opções] [arquivo]
```

Exemplo de utilização:

```
$ df -h
```

A opção acima é a mais usada pelos usuários, ela mostra o espaço disponível em todos os discos (**hd***) em megabytes, sistema de arquivo, tamanho do disco, quanto do disco está sendo utilizado, quanto há de espaço disponível, a porcentagem usada e onde o disco está montado.

Veja um exemplo abaixo do comando "**df -h**":

```
Sist. Arq. Tam Usad Disp Uso% Montado em
/dev/hda1 9,8G 2,1G 7,7G 22% /
tmpfs 252M 4,0K 252M 1% /dev/shm
/dev/hda3 43G 42G 1,8G 96% /filmes
/dev/hda4 12G 4,2G 7,6G 36% /home
/dev/hda5 9,1G 6,6G 2,6G 73% /servidor
```

- Para mostrar o mesmo que o "**df -h**", mudando apenas o tamanho dos **hd*** visualizados para kbytes:

```
$ df -k
```

- Para mostrar somente as partições formatadas em "**reiserfs**", em megabytes:

```
$ df -ht reiserfs
```

- Para mostrar somente as partições formatadas em "**vfat**", em megabytes:

```
$ df -ht vfat
```

Para mais informações consulte o manual:

```
$ man df
```

2.20. dmesg

Mostra tudo que é carregado pelo **Kernel** durante o **boot**. Mais utilizado para saber se o **boot** ocorreu bem, sem erros.

Sintaxe:

```
$ dmesg [opção]
```

Exemplo de utilização:

- Para mostrar as informações de carregamento do **Kernel** com paginação:

```
$ dmesg | more
```

- Para apagar o conteúdo do **buffer** rotativo depois de imprimir:

```
# dmesg -c
```

Para mais informações consulte o manual:

```
$ man dmesg
```

2.21. dpkg

Gerenciador de pacotes. Ferramenta utilizada para instalar, criar, remover e gerenciar pacotes **Debian**.

Sintaxe:

```
$ dpkg [opções] ação
```

Exemplo de utilização:

- Para instalar um pacote:

```
# dpkg -i pacote.deb
```

- Para forçar a instalação de um pacote:

```
# dpkg -i --force-all pacote.deb
```

- Para remover um pacote:

```
# dpkg -r pacote.deb
```

- Para extrair e exibir os nomes de arquivos contidos no pacote para dentro do diretório "**outdir**":

```
# dpkg -X pacote.deb outdir/
```

- Para remover um pacote e todas as suas configurações do sistema:

```
# dpkg -P pacote.deb
```

- Para mostrar informações referentes ao pacote:

```
$ dpkg -s pacote.deb
```

- Para buscar todos os pacotes que são referenciados pelo "**nome-do-pacote**" procurado, retornando a versão e uma descrição do mesmo:

```
$ dpkg -l|grep nome-do-pacote
```

- Para listar o conteúdo de um pacote, ou seja, os arquivos contidos no pacote:

```
$ dpkg -c pacote.deb
```

- Para reconfigurar um pacote instalado:

```
# dpkg-reconfigure pacote.deb
```

- Para criar um pacote de um programa instalado no sistema:

```
# dpkg-repack [nome do pacote]
```

- Para apagar as informações existentes sobre os pacotes que estão disponíveis:

```
# dpkg --clear-avail
```

Para mais informações consulte o manual:

```
$ man dpkg
```

2.22. du

Mostra o espaço ocupado em disco por arquivos e diretórios.

Sintaxe:

```
$ du [opções] [arquivo]
```

Exemplo de utilização:

- Para mostrar o total de espaço utilizado pelo diretório "**Documentos**", em megabytes:

```
$ du -hs ~/Documentos
```

- Para mostrar o total de espaço utilizado pelo diretório "**Documentos**" e todos os seus subdiretórios, em megabytes:

```
$ du -hc ~/Documentos
```

Para mais informações consulte o manual:

```
$ man du
```

2.23. emerge

O **Gentoo** implementa um sistema de gerenciamento de pacotes chamado "**Portage**". O Portage foi inteiramente baseado no "**Ports**" dos ***BSDs**. O **portage** implementa, entre outras funcionalidades, o gerenciamento de dependências, falsas instalações, desinstalações seguras, perfis de sistema e o gerenciamento de arquivos de configuração.

(Colaboração, Otávio Rodolfo)

Sintaxe:

```
# emerge [opções] [pacote]
```

Exemplo de utilização:

- Para instalar o pacote "**gftp**" no sistema:

```
# emerge gftp
```

- Para remover o pacote "**gftp**" do sistema:

```
# emerge -C gftp
```

- Para remover os pacotes antigos que não afetarão a funcionalidade nem quebrarão as dependências do sistema:

```
# emerge -c pacote
```

- Para atualizar o Opera para a última versão estável disponível na **portage tree**. Supondo que a última versão estável seja 6.12 e você tenha instalado a versão 7.11, o portage irá fazer **downgrade** para a versão 6.12:

```
# emerge -u opera
```

- Para remover pacotes antigos do seu sistema sem verificar funcionalidades e/ou dependências:

```
# emerge -P pacotes
```

- Para criar os binários/documentação/etc do **unrar** e os instalar no sistema:

```
# emerge -b unrar
```

- Para procurar todos os pacotes que tiverem **kde** no nome (**kde, kde-base, kde-i18n-bs etc**):

```
# emerge -s kde
```

- Para baixar a última versão do **mozilla** e deixar disponível em **$DISTDIR**:

```
# emerge -f mozilla
```

Para mais informações consulte o manual:

```
$ man emerge
```

2.24. file

Determina o tipo de conteúdo do arquivo baseado em padrões encontrados dentro do próprio arquivo.

Com o comando **file**, você pode descobrir informações importantes sobre um arquivo, se é um arquivo texto, binário, imagem, diretório etc.

Sintaxe:

```
$ file [opções] [arquivo]
```

Exemplo de utilização:

- Para mostrar informações sobre que tipo de arquivo é o "**arquivo1.txt**":

```
$ file arquivo1.txt
```

- Para mostrar informações sobre os arquivos: "**arquivo7.txt**", "**arquivo3.sh**", "**arquivo1.png**":

```
$ file arquivo7.txt arquivo3.sh arquivo1.png
```

Para mais informações consulte o manual:

```
$ man file
```

2.25. find

Procura por arquivos pelo sistema em uma hierarquia de diretórios.

Sintaxe:

```
$ find [caminho] [expressão]
```

Exemplo de utilização:

- Para procurar pelo arquivo "**senhas.txt**" no diretório "**/home**" e em todos os seus subdiretórios. O "**-iname**" ignora maiúsculas e minúsculas. Para não ignorar, remova a letra "**i**" e deixa apenas "**-name**":

```
$ find /home -iname senhas.txt
```

- Para fazer uma busca por todos os arquivos "**.txt**" dentro do diretório "**/home**", eliminando buscas indesejáveis (**2>/dev/null**):

```
$ find /home -name "*.txt" 2>/dev/null
```

Para mais informações consulte o manual:

```
$ man find
```

2.26. free

Mostra informações como: memória total, usada, livre, buffers e cache.

Sintaxe:

```
$ free [opção]
```

Exemplo de utilização:

- Para imprimir informações sobre a memória em megabytes:

```
$ free -m
```

- Para mostrar o total para **RAM** + **swap**, em megabytes:

```
$ free -mt
```

Para mais informações consulte o manual:

```
$ man free
```

2.27. fsck

O **fsck** é o programa de checagem de discos. Existem variações do **fsck**, cada uma voltada para um formato de partição. Por exemplo o **fsck.reiserfs** é voltado para checar discos/partições que estejam formatados em **reiserfs.** Por este motivo este tópico foi dividido em partes, cada uma voltada para um programa **fsck**.

(Contribuição, Luciano Martini)

```
fsck.ext2/fsck.ext3
```

Este é o programa responsável por checar sistemas de arquivos **ext2/ext3**. O programa não pode checar uma partição que esteja montada para o modo leitura-escrita, neste modo o programa fica impossibilitado de acessar o dispositivo adequadamente.

```
fsck.reiserfs
```

Este é o programa responsável pela checagem de sistemas de arquivos **reiserfs**. Assim como o **fsck.ext3** o programa não pode checar uma partição que esteja montada para o modo leitura-escrita.

```
fsck.vfat/fsck.msdos
```

Estes programas checam sistemas de arquivos fat e seus derivados usados em sistemas **Dos/Windows**.

Sintaxe:

```
# fsck.[sistema de arquivos] [opções] partição
```

Opções do fsck.ext2/fsck.ext3:

-p ——— *Reparo automático.*

-n ——— *Esta opção faz com que o **fsck.ext2** não faça alterações no sistema de arquivos, apenas o verifique.*

-y ——— *Esta opção força o **fsck.ext2** a assumir a resposta "sim" a todas as questões que serão feitas.*

-c ——— *Esta opção leva o **fsck.ext2** a verificar a presença de danos físicos ou **bad blocks** no **hd**.*

-f ——— *O **fsck.ext2** não verifica uma partição que está marcada como checada. Esta opção força o **fsck** a checar uma partição ou disco, mesmo que ainda não esteja na data de checagem.*

-b [superblock] —— *Esta opção faz com que o **fsck** use o **superblock** alternativo para relacionar os setores aos arquivos.*

-B [blocksize] ——— *Esta opção força o **fsck** a considerar um tamanho de bloco especificado pelo usuário (ajuda quando este não consegue fazer a determinação automaticamente).*

-j external-journal — *Leva o **fsck** a utilizar o journal de recuperação gravado em disco externo.*

Opções do fsck.reiserfs:

--check ——— *Esta é a checagem padrão do programa, quando iniciado sem nenhum parâmetro de opção.*

--fix-fixable ——— *Esta opção leva o **fsck** a arrumar os problemas mais comuns, que podem ser resolvidos sem a reconstrução de arvore **reiser**.*

--rebuild-sb ——— *Esta opção é de alto risco, leva o **fsck** a reescrever o **superblock**, ou seja, o mapa de setores em relação a arquivos e diretórios. Usada em casos críticos em que a partição não é mais reconhecida como **reiserfs**.*

--rebuild-tree ——— ***Rebuild tree** é a opção que leva o **fsck** a remontar a árvore **reiser**. É uma opção de alto-risco usada em últimos casos. Geralmente quando há problemas físicos, ao ser iniciado este processo, o **fsck** não deverá ser cancelado.*

-p ——— *Verificação automática.*

Opções do fsck.vfat:

-a ——— *Reparo automático.*

-t ——— *Testa a superfície.*

-u arquivo ——— *Funciona como o **undelete**. Funcional em partições **fat16**.*

-v ——————— *Aumenta o número de detalhes mostrados ao usuário.*

-w ——————— *Grava no disco imediatamente.*

Exemplos de uso do **fsck.ext3** (ou **fsck.ext2**):

```
# fsck.ext3 -p /dev/fd0
```

O comando acima fará uma checagem automática num disquete em a: (**/dev/fd0**) formatado em ext3.

```
# fsck.ext3 -c /dev/fd1
```

O comando acima fará uma checagem em busca da presença de danos físicos em um disquete em b: (**/dev/fd1**).

```
# fsck.ext3 -pf /dev/hda1
```

O comando acima fará uma checagem em busca de danos físicos, na partição "**hda1**", mesmo que ela esteja fora da data de checagem.

Exemplo de uso do **fsck.reiserfs**:

```
# fsck.reiserfs /dev/fd0
```

O comando acima checa um disquete a: (**/dev/fd0**) formatado em reiserfs (o formato reiserfs não é adequado para disquetes devido à baixa taxa de recuperação de erros físicos), o reiserfs é adequado para sistemas com mais de 30 GB onde ocorrem falhas de energia e onde é necessário um grande desempenho.

Exemplos de uso do **fsck.vfat**:

```
# fsck.vfat -atw /dev/hda3
```

O comando acima checa a partição "**/dev/hda3**" por **bad blocks** corrigindo erros automaticamente.

```
# fsck.vfat -aw /dev/hda3
```

O comando acima faz uma checagem rápida e automática de uma partição **vfat.**

Para mais informações consulte o manual:

```
$ man fsck
```

2.28. groups

Exibe os grupos os quais o usuário pertence.

Sintaxe:

```
$ groups [usuário]
```

Exemplo de utilização:

- Para exibir todos os grupos que o usuário "tales" pertence:

```
$ groups tales
tales : tales dialout cdrom floppy audio video ntop
```

- Para exibir os grupos aos quais os usuários "tales", "jqueiros" e "mari" pertencem:

```
$ groups tales jqueiros mari
tales : tales dialout cdrom floppy audio video ntop
jqueiros : users
mari : users
```

Para mais informações consulte o manual:

```
$ man groups
```

2.29. halt

Utilizado para desligar o sistema.

Sintaxe:

```
# halt [opção]
```

Exemplo de utilização:

 # halt ———— *Desliga o sistema.*

 # halt -f ———— *Força o desligamento do sistema.*

 # halt -h ———— *Coloca os **Hds** no modo **standby**.*

Para mais informações consulte o manual:

```
$ man halt
```

2.30. hdparm

O comando **hdparm** traz melhorias quanto a velocidade de acesso a arquivos e diretórios em um **HD** ou **CD-ROM**.

Sintaxe:

```
# hdparm [opções] [dispositivo]
```

Exemplo de utilização:

```
# hdparm -i /dev/hdX
```

Identifica informações do **HD** gerando um relatório, com as configurações atuais. Todas as alterações serão feitas com base no relatório adquirido. Essas informações serão importantes para que não se passe parâmetros errados ao disco rígido e o danifique.

Use as informações abaixo de acordo com o seu **HD**, onde o "X" é o número correspondente à partição do **HD**.

hda ——————— *Master primário.*
hdb ——————— *Slave primário.*
hdc ——————— *Master secundário.*
hdd ——————— *Slave secundário.*

- Para ativar o **DMA** do **HD** ou **CD-ROM**:

```
# hdparm -d1 /dev/hdX
```

- Para testar a leitura no **HD** para obter a taxa de transferência:

```
# hdparm -t /dev/hdX
```

- Para ativar o parâmetro "**I/O support**" para o modo de **32-bits**:

```
# hdparm -c1 /dev/hdX
```

Para mais informações consulte o manual:

```
# man hdparm
```

2.31. help

Utilizado para mostrar exemplos rápidos da utilização de comandos, ou mesmo descrever como utilizar um comando de forma rápida e prática.

Sintaxe:

```
$ comando --help
```

Exemplo de utilização:

```
$ cp --help | more
```

Mostra como utilizar o comando "**cp**" de uma forma mais rápida, evitando consultas ao manual para maiores informações. No comando acima, "**| more**" é utilizado para mostrar o comando com paginação, evitando revelar todo o conteúdo de uma única vez, perdendo assim parte do conteúdo se o mesmo for grande.

2.32. history

Utilizado para mostrar o histórico dos comandos digitados no terminal de comandos.

Sintaxe:

```
$ history [opção]
```

Exemplo de utilização:

- Para mostrar todos os comando digitados pelo usuário:

```
$ history
```

- Para mostrar todos os comandos digitados, com paginação:

```
$ history | more
```

- Para mostrar todos os comandos **ls** digitados:

```
$ history | grep ls
```

- Para mostrar os últimos 10 comandos digitados:

```
$ history 10
```

- Para limpar toda a lista de comandos que foram digitados:

```
$ history -c
```

Para mais informações consulte o manual:

```
$ man history
```

2.33. hostname

Mostra ou muda o nome do cliente do sistema. O **hostname** é usado para modificar ou para exibir o cliente atual ou o nome do domínio do sistema. Este nome é usado por vários programas que trabalham em rede para identificar a máquina.

Sintaxe:

```
$ hostname [opções]
```

ou

```
# hostname [Novo nome da máquina]
```

Exemplo de utilização:

- Para exibir o nome da máquina:

```
$ hostname
```

- Para mudar o nome da máquina para "**horus**". Lembrando que, quando o sistema for reiniciado o nome retornará para o anterior. Para isso, basta alterar direto no arquivo de configuração, que pode variar de acordo com a distribuição. "**/etc/hostname**" (**Slackware, Debian, Mandrake, Red Hat**); "**/etc/sysconfig/network**" (**Conectiva**); " **/etc/conf.d/hostname**" (**Gentoo**):

```
# hostname horus
```

- Para exibir o nome alternativo (**alias**) do servidor (**se usado**):

```
$ hostname -a
```

- Para exibir o nome do domínio **DNS**. Não use o comando "**domainname**" para exibir o nome do domínio **DNS** este mostrará o **NIS** e não o **DNS**. Use o comando "**dnsdomainname**":

```
$ hostname -d
```

- Para exibir o(s) endereço(s) de **IP** do(s) cliente(s):

```
$ hostname -i
```

Para mais informações consulte o manual:

```
$ man hostname
```

2.34. ifconfig

Utilizado para configurar e inspecionar interfaces de rede. Seu uso pode ser descrito de forma geral como **ifconfig interface comando,** onde interface é uma das interfaces de rede definidas e conhecidas pelo sistema (eth0, ppp0, usb1)

(Contribuição, José Queiroz)

Sintaxe:

```
# ifconfig [interface]
```

ou

```
# ifconfig [interface] [opção] | [endereço]
```

Exemplo de utilização:

- Para mostrar todas as interfaces ativas no momento, pacotes enviados e recebidos. Pode ser utilizado o comando "**ifconfig interface**" para visualizar só a interface desejada:

```
# ifconfig
```

- Para ativar a interface indicada:

```
# ifconfig interface up
```

- Para desativar a interface indicada:

```
# ifconfig interface down
```

- Para configurar a placa de rede eth0, ajustando o **IP** 10.0.0.1 e **netmask** 255.255.255.0:

```
# ifconfig eth0 10.0.0.1 netmask 255.255.255.0
```

Para mais informações consulte o manual:

```
$ man ifconfig
```

2.35. installpkg

O **installpgk** é uma ferramenta do **pkgtool**, utilizado para instalar pacotes no formato **pacote.tgz** no **slackware**.

Sintaxe:

```
# installpkg [opção] [pacote]
```

Exemplo de utilização:

- Para instalar o "**pacote.tgz**" no sistema:

```
# installpkg pacote.tgz
```

Para mais informações consulte o manual:

```
$ man installpkg
```

2.36. kill

Envia sinais para os processos do sistema, ou seja, envia sinais para o **PID** (**identificação de processo**) do processo. Geralmente utilizado para matar um processo em execução.

Sintaxe:

```
$ kill [sinal] PID
```

Exemplo de utilização:

```
$ ps aux | grep amsn
tales 3698 0.0 0.2 2740 1176 ? S 13:19 0:00 /bin/sh /usr/bin/amsn
tales 3699 0.4 3.0 27972 15920 ? Sl 13:19 2:43 /usr/bin/wish amsn
tales 15726 0.0 0.1 1844 708 pts/3 S+ 22:46 0:00 grep amsn
$ kill -9 3699
$ ps aux | grep amsn
tales 15760 0.0 0.1 1844 704 pts/3 R+ 22:48 0:00 grep amsn
```

Primeiro foi listado o programa que se deseja matar. Com o comando "**ps aux**", foi utilizado o comando "**| grep**" para obter informações apenas do programa **amsn** que se deseja matar. Logo em seguida, utilizando o comando "**kill -9**" + número de identificação (**PID**), o processo (programa **amsn**) é terminado.

- Para matar todos os processos de 1027 a 1051:

```
$ kill -9 1027 1051
```

Para mais informações consulte o manual:

```
$ man kill
```

2.37. last

Exibe o histórico de logins e logouts de usuários efetuados no sistema, baseado no conteúdo do arquivo **/var/log/wtmp**.

Sintaxe:

```
$ last [opções]
```

Exemplo de utilização:

- Para mostrar a lista geral de **logins/logouts**:

```
$ last
```

- Para listar os 10 últimos **logins/logouts** efetuados no sistema:

```
$ last -10
```

- Para exibir as entradas de desligamento do sistema e mudanças de nível de execução:

```
$ last -x
```

- Para listar todos os **reboots** do sistema:

```
$ last reboot
```

Para mais informações consulte o manual:

```
$ man last
```

2.38. ln

Cria uma ligação simbólica entre arquivos. Por padrão são criadas ligações fortes; com a opção **-s** cria-se ligações simbólicas (ou "fracas").

Sintaxe:

```
$ ln [opção] origem [destino]
```

Exemplo de utilização:

- Para criar um link simbólico do executável "**skype**" para o diretório "**/usr/local/bin/**", possibilitando executar o programa de qualquer lugar apenas digitando o nome "**skype**":

```
# ln -s /servidor/programas/skype-xxx/skype /usr/local/bin/
```

- Para remover tanto o **link** simbólico "**skype**" (destino) quanto o arquivo "**skype**" (origem):

```
# ln -f /usr/local/bin/skype
```

- Para perguntar se deseja remover o **link** simbólico e o arquivo de origem:

```
$ ln -i [link simbólico]
```

Para mais informações consulte o manual:

```
$ man ln
```

2.39. locate

Busca por arquivos e diretórios em uma base de dados. O comando **locate** procura pela palavra-chave que foi digitada, ou seja, tudo que contém a palavra digitada, seja no início, meio ou fim. Será retornado o caminho de todos os arquivos encontrados.

Sintaxe:

```
$ locate [opções] [palavra-chave]
```

Exemplo de utilização:

```
$ locate firefox
```

Retorna o caminho de todos os arquivos que contiverem a palavra "**firefox**", indicando aonde se encontram.

```
$ locate -e *.txt
```

Retorna todos os arquivos "**.txt**" que existem no sistema, mostrando apenas os nomes que existem atualmente (ao invés dos nomes que existiam quando a base de dados foi criada).

```
$ locate -E kde
```

Procura no sistema por tudo que tiver "**kde**", mostrando apenas os nomes que não existem atualmente (ao invés dos nomes que existiam quando a base de dados foi criada).

```
$ locate -i xmms
```

Procura por tudo que tiver "**xmms**", ignorando maiúsculas e minúsculas.

Obs.: *Para atualizar a base de dados do comando* ***locate****, digite o seguinte comando, como super usuário:* `# updatedb`

Para mais informações consulte o manual:

```
$ man locate
```

2.40. ls

Lista arquivos e diretórios. Pode-se usar várias opções úteis, como exibir arquivos e diretórios coloridos para facilitar a visualização do mesmo.

Sintaxe:

```
$ ls [opção] [arquivo/diretório]
```

Exemplo de utilização:

- Para exibir arquivos e diretórios, exceto ocultos:

```
$ ls
```

- Para exibir a saída de arquivos e diretórios com cor:

```
$ ls --color=auto
```

- Para exibir arquivo e diretórios, incluindo ocultos (todo arquivo ou diretório oculto começa com um ponto na frente. Exemplo "**.mplayer**"):

```
$ ls -a
```

- Para exibir o arquivo "**teste.txt**" em coluna, na qual a primeira da esquerda para direita significa as permissões que o arquivo possui:

```
$ ls -lh teste.txt
-rw-r--r-- 1 tales tales 7k 2005-02-25 00:34 teste.txt
```

O primeiro nome (tales) é o usuário a quem o arquivo pertence, e o segundo nome (tales) é o grupo ao qual o arquivo pertence. O número 7k é o tamanho do arquivo, seguido pela data que foi criado. No final aparece o nome do arquivo.

- Para listar todos os diretórios encontrados dentro do home (~) do usuário, recursivamente:

```
$ ls -R ~/
```

- Para listar todos os arquivos e diretórios do diretório "**Documentos**" por tempo de alteração:

```
$ ls -t ~/Documentos/
```

- Para mostrar todo o conteúdo do diretório "**tmp**", listando cada diretório com uma barra (/), cada executável com um asterisco (*), cada link simbólico com uma arroba (@), no final do arquivo:

```
$ ls -F /tmp/
```

- Para salvar uma lista de todo o conteúdo do diretório "**Documentos**" no arquivo "**lista-documentos.txt**":

```
$ ls -l ~/Documentos/ > lista-documentos.txt
```

Para mais informações consulte o manual:

```
$ man ls
```

2.41. lsmod

O **lsmod** é um programa trivial que formata o conteúdo do **/proc/modules**, mostrando quais os módulos do **Kernel** estão carregados.

Sintaxe:

```
$ lsmod
```

Exemplo de utilização:

```
$ lsmod
```

Mostra quais os módulos estão carregados no **Kernel** do **Linux**.

Para mais informações consulte o manual:

```
$ man lsmod
```

2.42. lspci

Lista todos os dispositivos **PCI**. O **lspci** é um utilitário que mostra informações sobre o barramento **PCI** e todos os dispositivos conectados a ele.

Sintaxe:

```
$ lspci [opções]
```

Exemplo de utilização:

- Para mostrar todos os números **IRQs** e endereços vistos pelas placas no barramento **PCI**, como nome e modelo da **placa de rede**, **placa de som**, **placa de vídeo**, dispositivos **USB** plugados na **placa mãe**, entre outros:

```
$ lspci
```

- Para mostrar informações detalhadas de tudo o que um dispositivo **PCI** pode dizer:

```
$ lscpi -vv
```

Para mais informações consulte o manual:

```
$ man lspci
```

2.43. MAKEDEV

Utilizado para criar dispositivos. O **MAKEDEV** é um script que irá criar dispositivos no /dev usando a interface com os drivers do **Kernel**.

Sintaxe:

```
# cd /dev
# ./MAKEDEV [opção] [dispositivo]
```

Opções:

-n ——————— *Não irá realmente atualizar os serviços, apenas mostrar as ações que serão executadas.*

-d ——————— *Deleta os serviços. Seu principal uso é pelo próprio **MAKEDEV**.*

Exemplo de utilização:

- Para criar o dispositivo "**ttyS0**" para uso do mouse serial:

```
/dev# ./MAKEDEV ttyS0
```

- Para criar o dispositivo "**dsp**" para utilização do som:

```
/dev# ./MAKEDEV dsp
```

Para mais informações consulte o manual:

```
$ man MAKEDEV
```

2.44. mkdir

Cria um diretório com os nomes especificados.

Sintaxe:

```
$ mkdir [opção] [nome do diretório]
```

Exemplo de utilização:

- Para criar o diretório "**Programas**":

```
$ mkdir Programas
```

- Para criar o diretório "**Programas**" dentro do diretório "**~/Documentos/**":

```
$ mkdir ~/Documentos/Programas
```

- Para criar o diretório "**Imagens**" com os subdiretorios "**JPG**" e "**PNG**", e o diretório "**Jogos**" com o subdiretório "**Ação**".

```
$ mkdir -p Imagens/JPG Imagens/PNG Jogos/Ação
```

- Para criar o diretório "**Vídeos de Humor**". As aspas duplas ("") foram utilizadas porque houve espaço entre os nomes do diretório. Também pode ser utilizado a barra invertida entre os espaços:

```
$ mkdir "Vídeos de Humor"
```

Veja o exemplo abaixo:

```
$ mkdir Vídeos\ de\ Humor
```

Para mais informações consulte o manual:

```
$ man mkdir
```

2.45. modprobe

O comando **modprobe** adiciona ou remove de forma inteligente um módulo no **Kernel** do **Linux**. Não há diferença entre os caracteres underline (_) e traço (-) nos nomes dos módulos.

Sintaxe:

```
# modprobe [opções] [nome do módulo]
```

Exemplo de utilização:

- Para subir com o módulo "**ac97_codec**":

```
# modprobe ac97_codec
```

- Para descarregar na tela as configurações do **modprobe** e sair:

```
# modprobe -c
```

- Para remover o módulo "**ac97_codec**":

```
# modprobe -r ac97_codec
```

Caso haja um outro módulo que dependa do módulo que será removido e não estiver em uso, o **modprobe** tentará removê-lo também. É mais aconselhável utilizar o comando **rmmod**, que é específico para a remoção de módulos.

Obs.: *Quando se insere um módulo no **Kernel** do **Linux** manualmente, o mesmo só ficará salvo enquanto a máquina estiver ligada, ou seja, sempre que reiniciar o sistema será preciso carregar o módulo. Para que isso não aconteça, basta inserir o nome do módulo no final da linha do arquivo de configuração **/etc/modprobe.conf.***

Para mais informações consulte o manual:

```
$ man modprobe
```

2.46. more

Mostra o conteúdo de um arquivo com paginação.

Sintaxe:

```
$ more [opções] [arquivo]
```

Para paginar o arquivo tecle **ESPAÇO**, para descer linha por linha tecle **ENTER**, para editar o arquivo tecle "v" (é chamado o editor padrão) e para sair tecle "q".

Exemplo de utilização:

- Para mostrar o conteúdo do arquivo "**sources.list**":

```
$ more /etc/apt/sources.list
```

- Para mostrar o conteúdo dos arquivos "**sources.list**" e "**.bashrc**":

```
$ more /etc/apt/sources.list ~/.bashrc
```

- Para transferir a saída do comando "**ls**" para a entrada do comando "**more**", mostrando todo o conteúdo do diretório "**/usr/bin/**" com paginação:

```
$ ls -l /usr/bin/ | more
```

Para mais informações consulte o manual:

```
$ man more
```

2.47. mount

Monta dispositivos, sejam eles locais ou remotos.

(Contribuição, Andrei Drusian)

Sintaxe:

```
# mount -t type /dev/hdxy [local de montagem]
```

Onde:

"type" ——— *é o tipo de partição que será montada.*

"x" ——— *é o local que o **HD** está, podendo ser: **Primário Master** (a), **Primário Slave** (b), **Secundário Master** (c) ou **Secundário Slave** (d).*

"y" ——— *é a partição que será montada, podendo ser 1, 2, 3...5 e assim por diante.*

Exemplo de utilização:

- Para montar um disquete formatado em ***FAT***, em "**/mnt/floopy**":

```
# mount -t vfat /dev/fd0 /mnt/floopy
```

- Para montar um **CD-ROM** em "**/mnt/cdrom**":

```
# mount -t iso9660 /dev/hdx /mnt/cdrom
```

- Para montar uma partição **Windows FAT** em "**/mnt/win**":

```
# mount -t vfat /dev/hdxy /mnt/win
```

- Para montar um pendrive em "**/mnt/pendrive**" com o formato **FAT**:

```
# mount -t vfat /dev/sdxy /mnt/pendrive
```

- Para montar uma partição **Windows NTFS** em "**/mnt/win**":

```
# mount -t ntfs /dev/hdaX /mnt/win
```

- Para montar um compartilhamento remoto **Windows/SMB** em "**/mnt/smb**":

```
# mount -t smbfs //192.168.0.1/dados /mnt/smb -o username=xxx
```

- Para montar um compartilhamento remoto **NFS** em "**/mnt/nfs**":

```
# mount -t nfs 192.168.0.1:/dados /mnt/nfs
```

- Para montar uma partição do **FreeBSD** em "**/mnt/bsd**":

```
# mount -t ufs -o ufstype=44bsd /dev/hdaX /mnt/bsd
```

Para mais informações consulte o manual:

```
# man mount
```

2.48. mv

Move ou renomeia arquivos e diretórios. Se o caminho de destino for o mesmo de origem, o arquivo ou diretório será renomeado, caso contrário movido.

Sintaxe:

```
$ mv [opção] [origem] [destino]
```

Exemplo de utilização:

- Para renomear o "**arquivo-tales.txt**" para "**arquivo-mari.txt**":

```
$ mv arquivo-tales.txt arquivo-mari.txt
```

- Para mover o arquivo "**foto07.png**" de "**~/Documentos/**" para "**~/Fotos**":

```
$ mv ~/Documentos/foto07.png ~/Fotos
```

- Para mover o "**arquivo01.txt**" para o diretório "**Documentos**". Caso haja um "**arquivo01.txt**" dentro do diretório "**Documentos**", o mesmo será substituído pelo arquivo de origem.

```
$ mv -f arquivo01.txt ~/Documentos
```

- Para mover o "**arquivo03.txt**" para o diretório "**Documentos**". Caso haja um "**arquivo03.txt**" dentro do diretório "**Documentos**", serão criadas cópias de segurança dos arquivos que estão para ser sobrescritos ou removidos.

```
$ mv -b arquivo03.txt ~/Documentos
```

Para mais informações consulte o manual:

```
$ man mv
```

2.49. nmap

O comando **nmap** serve para fazer varredura de portas. A varredura de portas é um processo pelo qual se determina quais as portas **TCP** e **UDP** estão abertas (prontas para aceitar conexões) em uma determinada máquina.

(Contribuição, José Queiroz)

Sintaxe:

```
$ nmap [opções] alvos
```

Um alvo é um nome de máquina ou endereço **IP**, o endereço de um bloco **CIDR** ou uma faixa de subrede. O **nmap** pode fazer varreduras em vários alvos numa mesma ativação.

Ao fim da varredura, o **nmap** informa quais são as portas que puderam ser identificadas como abertas (ativas), fechadas (inativas) ou filtradas (protegidas por **firewall**).

O uso de **firewalls** ou **TCP Wrappers** pode alterar os resultados do **nmap**. Além disso, sistemas implementando **DNAT** podem confundir o **nmap**, pois os resultados serão referentes às máquinas de destino, não à máquina vasculhada.

As principais opções do nmap são:

-v ——— *Modo verboso (recomendado). Melhor ainda: use duas vezes, dará ainda mais resultados.*

-s ——— *Indica o tipo de varredura a ser feita. As opções mais comuns são:*

- **-sT** ——— *Varredura **TCP**.*
- **-sS** ——— *Varredura Furtiva **TCP** (Half-Open).*
- **-sU** ——— *Varredura **UDP**.*
- **-sP** ——— *Varredura "**Ping**" (ICMP ECHO).*

-p ——— *Indica as portas a serem vasculhadas para o IP indicado. Ex:*

```
# nmap -p 1-1024,1080,21337 201.19.70.246
```

-P ——— *Indica se antes da varredura, deve se verificar ou não se a máquina existe. Para essa verificação, usa-se um pacote Ping (ICMP ECHO).*

-P0 ——— *desliga a verificação de existência.*

-O — *Faz uma identificação de "impressão digital" (**fingerprint**) para determinar, de maneira aproximada, qual é o sistema operacional do alvo. De acordo com as informações liberadas, no entanto, é possível aprofundar essa identificação, chegando ao ponto de ser possível diferenciar o nível de patch aplicado a um sistema operacional.*

-T — *Define a política de temporização entre as tentativas de acesso às portas. Intervalos curtos, retornam resultados mais rapidamente, porém aumentam a probabilidade de detecção da varredura. Dependendo da política usada, o **nmap** pode ainda realizar testes em paralelo, para aumentar a velocidade da varredura. Em vez de usar os nomes das políticas, podemos usar números; assim, **-T0** equivale a "**-T Paranoid**", e **-T5** equivale a "**-T Insane**".*

-T Paranoid Intervalo máximo.

-T Sneaky.

-T Polite.

-T Normal.

-T Aggressive.

-T Insane Intervalo mínimo.

-n — *Não tenta fazer resolução de endereços em nomes por **DNS** reverso.*

--interactive — *Abre o modo interativo. Neste modo o **nmap** pode receber vários comandos para realizar varreduras. Para maiores informações sobre este modo, ative-o e use "**h**" para obter um texto de auxílio.*

Há várias outras opções possíveis, por exemplo, é possível fazer varreduras diferenciadas por pacotes **FIN**, ou então fazer varreduras de serviços **RPC**. Também é possível alterar parâmetros de temporização, tanto para reduzir a possibilidade de detecção da varredura, quanto para adaptar o funcionamento às características de carga da rede.

Algumas das opções de varredura exigem acesso privilegiado ao sistema. Assim, apenas o superusuário pode realizar varreduras com essas opções. Via de regra, um usuário comum pode fazer apenas varreduras simples.

Obs.: *Ao realizar uma varredura de portas, você está levantando informações sobre a máquina remota. Essa atitude não deixa de ser uma invasão de privacidade; use este recurso com cuidado, pois várias pessoas consideram a varredura de portas como uma tentativa de invasão.*

Para mais informações consulte o manual:

```
$ man nmap
```

2.49.1. xnmap (A Interface Gráfica)

Há uma implementação de GUI para o **nmap** que funciona em Xwindows, chamada **xnmap**. Várias das opções do **nmap** podem ser usadas de forma mais simples com o **xnmap**. Além disso, **xnmap** dá destaque para resultados mais importantes usando cores diferenciadas.

Para mais informações consulte o manual:

```
$ man xnmap
```

2.50. passwd

Muda a senha para usuários e grupos. O usuário pode somente mudar a sua senha, já o **root** tem o direito de mudar a senha de todos os usuários.

Sintaxe:

```
# passwd [opção] [usuário ou grupo]
```

Exemplo de utilização:

- Para mudar a senha do usuário:

```
$ passwd
```

- Para mudar a senha do grupo "tales":

```
$ passwd -g tales
```

- Para mudar informações do usuário "tales":

```
$ passwd -f tales
```

- Para mudar o local do shell de login para "tales":

```
$ passwd -s tales
```

Para mais informações consulte o manual:

```
$ man passwd
```

2.51. ping

O comando **ping** é uma das ferramentas mais simples do técnico de redes. Consiste num programa que envia um pacote especial pela rede, que ao ser recebido pela máquina de destino, faz com que ela envie de volta outro pacote. A partir da chegada da resposta, o computador de origem pode saber que:

- a máquina existe;
- está "viva" (está ligada e ativa, pois responde comandos);
- é alcançável a partir da rede.

De acordo com o sistema em uso, o **ping** pode ter comportamento e parâmetros diferentes. Nas máquinas windows, por padrão, o **ping** envia apenas 4 pacotes. Já nas máquinas **Linux**, por padrão, ele envia um fluxo contínuo de pacotes. Nas máquinas **Sun**, ao contrário, apenas um pacote é enviado.

A saída do comando também se altera de versão para versão. Nas máquinas **Windows** e **Linux**, a cada pacote de resposta recebido, é impressa uma linha com o número desse pacote, o tempo decorrido a partir do envio do pacote e o **TTL** restante. Já nas máquinas **Sun**, um lacônico "<**host**> **is alive**" é impresso caso haja retorno do pacote.

(Contribuição, José Queiroz)

Sintaxe:

```
$ ping [opções] IP
```

Os principais parâmetros do **Ping**, no **Linux**, são:

-c número ——— *Envia no máximo <número> pacotes.*

-f ——— *Inundação de pacotes. Neste modo, o comando envia uma rajada de pacotes sem intervalo, tão rápido quanto o sistema consegue enviar. Esta opção deve ser usada com muita cautela, pois seu uso pode afetar seriamente o desempenho da rede local.*

-n ——— *Não tenta traduzir endereços ip em nomes de máquina.*

-s tamanho ——— *Define o tamanho para os pacotes enviados. Se não for especificado, um tamanho default de 56 bytes é usado, resultando junto com os cabeçalhos num pacote IP de 64 bytes. O valor máximo para o tamanho é definido pelo tipo de rede usado, sendo de 1500 bytes para Ethernet, e 1420 bytes para PPPoE.*

-W timeout ——— *Define um tempo de espera pelos pacotes, antes de se declarar o pacote perdido.*

Exemplo de utilização:

- Para enviar pacotes continuamente para o ip 192.168.254.7 até ser interrompido:

```
$ ping 192.168.254.7
```

- Para enviar 30 pacotes para o ip 192.168.254.7 e terminar:

```
$ ping -c 30 192.168.254.7
```

- Para enviar um pacote com 1420 bytes, quando o normal é enviar com 56:

```
$ ping -s 1420 192.168.254.7
```

Obs.: *É importante perceber que o **ping** não serve para medir a qualidade de uma rede, pois o tempo de resposta dos pacotes pode ser afetado por vários fatores, como a ocupação dos links no meio do caminho e a carga de utilização da cpu da máquina de destino.*

Para mais informações consulte o manual:

```
$ man ping
```

2.52. ps

Informa uma seleção dos processos ativos no sistema.

Sintaxe:

```
$ ps [opções]
```

Exemplo de utilização:

- Para exibir todos os processos do sistema:

```
$ ps aux
```

- Para exibir todos os processos em execução do programa **amsn**:

```
$ ps aux | grep amsn
```

Obs.: *A opção "a" (all) mostra não apenas os seus processos, mas também os de outros usuários. A opção "u" mostra o usuário o qual pertence a cada processo. A opção "x" inclui na lista os processos que não estejam conectados a um terminal.*

Para mais informações consulte o manual:

```
$ man ps
```

2.53. pwd

Exibe o caminho/nome do diretório em que o usuário se encontra.

Sintaxe:

```
$ pwd
```

Exemplo de utilização:

```
tales@horus:~$ pwd
/home/tales
```

O exemplo acima, mostra que o usuário se encontra dentro de "**/home/tales/**", ou "**~tales/**".

Para mais informações consulte o manual:

```
$ man pwd
```

2.54. rar

Compactador de arquivos. Muito utilizado hoje em dia por ser compatível com os **Sistemas Operacionais** mais conhecidos, e por apresentar uma boa performance.

Sintaxe:

```
$ rar [comando][opção] [arquivo.rar] [conteúdo a ser compactado]
```

Exemplo de utilização:

- Para gerar o arquivo "**fotos.rar**" com as fotos "**01**", "**02**" e "**03**" dentro do mesmo:

```
$ rar a fotos.rar foto01.png foto02.png foto03.png
```

- Para extrair o conteúdo do arquivo "**fotos.rar**":

```
$ rar x fotos.rar
```

- Para compactar o diretório "**.kde**" recursivamente gerando o arquivo "**configuracao-kde.rar**":

```
$ rar a -r configuracao-kde.rar ~/.kde
```

Para mais informações consulte o manual:

```
$ man rar
```

2.55. reboot

Reinicia o sistema.

Sintaxe:

```
# reboot [opção]
```

Exemplo de utilização:

- Para "**rebootar**" a máquina:

```
# reboot
```

- Para forçar o reboot do sistema:

```
# reboot -f
```

Para mais informações consulte o manual:

```
$ man reboot
```

2.56. removepkg

O **removepgk** é uma ferramenta do **pkgtool**, utilizada para remover pacotes no formato **pacote.tgz** do slackware.

Sintaxe:

```
# removepkg [opção] [pacote]
```

Exemplo de utilização:

- Para remover o "**pacote.tgz**":

```
# removepkg pacote.tgz
```

Para mais informações consulte o manual:

```
$ man removepkg
```

2.57. rm

Remove arquivos e diretórios.

Sintaxe:

```
$ rm [opções] [arquivo e/ou diretório]
```

Exemplo de utilização:

- Para remover os arquivos 1, 2 e 3:

```
$ rm arquivo1 arquivo2 arquivo3
```

- Para perguntar se deseja remover os arquivos 1, 2 e 3:

```
$ rm -i arquivo1 arquivo2 arquivo3
```

- Para forçar a remoção dos arquivos 1, 2 e 3:

```
$ rm -f arquivo1 arquivo2 arquivo3
```

- Para remover o diretório "**Imagens**" que se encontra dentro de "**~/Documentos/**":

```
$ rm -r ~/Documentos/Imagens
```

- Para forçar a remoção do diretório "**amsn**", encontrado dentro de "**~/Programas/**":

```
$ rm -rf ~/Programas/amsn
```

Obs.: *A opção "f" sempre é acrescentada por último, depois de todas as outras. Quando é utilizada, os diretórios e arquivos são removidos sem nenhuma pergunta. Cuidado ao utilizar essa opção "rm -rf", pode causar sérios danos à máquina caso remova algo indevido.*

Para mais informações consulte o manual:

```
$ man rm
```

2.58. rmmod

Simples programa utilizado para remover módulos no **Kernel** do **Linux**.

Sintaxe:

```
# rmmod [opção] [módulo]
```

Opções:

-v ——— *Mostra mensagens sobre o que o programa está fazendo.*

-f ——— *Esta opção pode ser extremamente perigosa. Com ela, você pode remover módulos que estão em uso, que não foram projetados para serem removidos ou marcados como inseguros.*

-w ——— *Normalmente o **rmmod** irá recusar remover módulos que estão em uso. Com esta opção o **rmmod** irá isolar o módulo e esperar até que o módulo não esteja em uso.*

Exemplo de utilização:

```
# rmmod snd_ac97_codec
```

Remove o módulo de áudio "**snd_ac97_codec**".

Para mais informações consulte o manual:

```
$ man rmmod
```

2.59. rpm

O **RPM** (Redhat Package Manager) é um sistema criado para controlar a instalação e desinstalação de pacotes de software na distribuição **Linux Redhat**. Como várias distribuições se basearam na Redhat, elas aproveitaram esse formato para seus controles de pacotes, tal como a Mandrake e a Conectiva. Com o passar do tempo, estas distribuições adicionaram facilidades ao RPM, como o **urpmi** pelo Mandrake para facilitar a obtenção dos pacotes de instalação. A Conectiva foi além; juntou as facilidades do RPM à flexibilidade do sistema **apt-get** desenvolvido pela distribuição Debian; e criou o ambiente "**Synaptic**" que controla a instalação e a atualização de todos os pacotes que compõem o sistema, além de recuperá-los diretamente dos repositórios na internet, tudo isso num ambiente gráfico.

(Contribuição, José Queiroz)

Sintaxe:

```
# rpm [opções] [pacotes]
```

Exemplo de utilização:

- Para instalar um pacote:

```
# rpm -i pacote.rpm
```

- Para atualizar um pacote já instalado:

```
# rpm -U pacote.rpm
```

- Para instalar e atualizar pacotes nessa ordem, recebendo informações adicionais como o progresso de uma instalação:

```
# rpm -ivh pacote1.rpm pacote2.rpm ...
# rpm -Uvh pacote1.rpm pacote2.rpm ...
```

Um pacote, após a instalação, assume um nome interno. Esse nome fica armazenado na base interna do **rpm**.

- Para ver todos os pacotes instalados, utilize o comando:

```
# rpm -qa pacotes
```

- Para mostrar quais são as dependências de um pacote ou **arquivo.rpm**:

```
# rpm -q Pacote --requires
# rpm -qp pacote.rpm --requires
```

Nota: *Caso deseje remover um ou mais pacotes instalados. Onde Pacote1, Pacote2 etc, são os nomes internos dos pacotes. Caso desinstale algum dos pacotes que seja necessário para outro que não esteja sendo desinstalado, o rpm se recusa a remover o pacote:* `# rpm -e Pacote1 Pacote2 Pacote3...`

- Para mostrar quais outros pacotes dependem de um pacote. Esta opção só funciona com pacotes instalados:

```
# rpm -q Pacote --whatrequires
```

- Para mostrar qual foi o pacote responsável pela instalação de um determinado arquivo:

```
# rpm -qf arquivo
```

- Para mostrar quais os arquivos que um pacote ou **arquivo.rpm** instala:

```
# rpm -q Pacote --list
# rpm -qp pacote.rpm --list
```

- Para obter mais informações sobre um pacote ou **arquivo.rpm**:

```
# rpm -q Pacote --info
# rpm -qp pacote.rpm --info
```

- Para mostrar se algum arquivo instalado por um determinado pacote foi alterado ou apagado:

```
# rpm --verify Pacote
# rpm --verify pacote.rpm
```

Obs.: *Às vezes um pacote para ser instalado depende de outros pacotes. Caso se tente instalar um pacote sem que os outros pacotes que ele necessita estejam instalados, o* ***rpm*** *se recusa a instalá-lo. Nesse caso, é preciso especificar todos os pacotes que faltam na linha de comando. Não é preciso se preocupar com a ordem dos pacotes na lista, o* ***rpm*** *os rearranja de forma que todas as dependências sejam satisfeitas.*

Veja um exemplo abaixo:

```
# rpm -i pacote.rpm pacote2.rpm pacote3.rpm ...
```

Todas as opções acima servem para instalar pacotes pré-compilados. No entanto, o RPM tem a possibilidade de trabalhar também com pacotes fonte (**arquivos.srpm**).

Para compilar e instalar um pacote fonte, usa-se o **rpm** como é mostrado abaixo:

```
# rpm --recompile arquivo.srpm
```

Note que usar a opção "**-i**" com um pacote fonte não irá instalar o pacote, mas sim descompactá-lo e prepará-lo para compilação. O **rpm** tem várias opções que permitem que se crie um pacote binário a partir de fontes, mas essas opções são avançadas e não serão tratadas aqui.

Para mais informações consulte o manual:

```
$ man rpm
```

2.60. scp

Permite fazer cópia de arquivos remotamente, utilizando uma autenticação e transferência de dados criptografados com o mesmo conceito do comando **ssh**.

(Contribuição, Eduardo C. Silva)

Sintaxe:

```
$ scp [opção] [arquivo] [usuário]@[ip da máquina]:[destino]
```

Exemplo de utilização:

Usando a mesma idéia do **ssh**, podemos transferir arquivos para um servidor distante através da internet:

```
$ scp arquivo.txt eduardo@192.168.0.3:/tmp/
```

O comando acima copia o "**arquivo.txt**" da máquina corrente para o diretório "**/tmp**" da máquina com ip "**192.168.0.3**".

Caso seja necessário especificar a porta, utilize o seguinte comando:

```
$ scp -p 22 arquivo.txt eduardo@192.168.0.3:/tmp/
```

Basta autenticar com a senha do usuário escolhido, a conexão e transferência serão iniciadas. Após finalizar a transferência, será desconectado do servidor automaticamente.

```
$ scp -r ~/Imgagens/ eduardo@192.168.0.3:/tmp/
```

O comando acima copia o diretório "**Imagens**" recursivamente da máquina corrente para o diretório "**/tmp**" da máquina com ip "192.168.0.3".

Para mais informações consulte o manual:

```
$ man scp
```

2.61. ssh

Comando usado para fazer conexões remotas em sistemas **Linux** via bash. O **ssh** é um serviço de login remoto, um método muito seguro pois permite a autenticação e transmissão de dados criptografados. Ele trabalha como server e client, sendo assim, ao iniciar o serviço, você pode se conectar à uma máquina remota ou permitir uma conexão na sua máquina.

(Contribuição, Eduardo C. Silva)

Sintaxe:

```
$ ssh [opção] [usuário]@[ip da máquina]
```

Exemplo de utilização:

Caso você precise se conectar em uma máquina **Linux** que esteja distante, ou seja, tem algum problema que precisa ser resolvido urgentemente, basta você ter uma conexão à internet e se conectar no servidor desta pessoa, pois a partir dela é possível você se conectar em qualquer outra máquina da rede via **ssh**.

```
$ ssh eduardo@200.199.152.2
```

O comando acima irá conectar o usuário corrente na máquina do "eduardo" que contém o ip "200.199.152.2".

```
$ ssh 200.199.152.2
```

No comando acima não foi definido o usuário, apenas especificado o ip da máquina que deseja se conectar. Será assumido por padrão o usuário root.

Obs.: *Após executar o comando **ssh**, será necessário digitar a senha do usuário escolhido para que seja feita a autenticação. Assim, será estabelecida a conexão no servidor de internet. Mas caso precise se conectar em uma estação da rede, basta executar o **ssh** novamente dentro do servidor acessado.*

Exemplo:

```
$ ssh root@192.168.0.3
```

O comando acima irá conectar-se na máquina interna de ip "**192.168.0.3**".

Caso queira conectar-se em uma porta específica, basta utilizar a opção "**-p**":

```
$ ssh -p 22 root@192.168.0.3
```

Para mais informações consulte o manual:

```
$ man ssh
```

2.62. su

O comando **su** é utilizado quando precisamos rodar um programa como se fossemos outro usuário. O **su** sem parâmetros ativa um interpretador de comandos (prompt/terminal) como root.

(Contribuição, Luciano Martini)

Sintaxe:

```
$ su [usuário] [opções]
```

Obs.: *Quando não fornecemos um usuário, o **su** adota o super usuário (root) como usuário padrão.*

Exemplo de utilização:

- Para logar no terminal como root:

```
$ su
```

- Para logar no terminal como "**777User**":

```
$ su 777Use
```

- Para abrir o "**xf86cfg**" como root (as aspas "" são necessárias):

```
$ su -c "xf86cfg -textmode"
```

- Para rodar o comando "**ls -l**" como "**777User**" :

```
$ su 777Use -c "ls -l"
```

Observações/Dicas:

** Sempre que executamos o **su** para fazer login de qualquer usuário, precisamos fornecer a senha deste usuário, a não ser que estejamos logados como root, dessa forma o **su** nunca pergunta senha de nenhum usuário.*

** Em scripts de distribuições sem o **sudo**, podemos substituí-lo por* **su -c**, *colocando aspas nos respectivos comandos. Exemplo: **sudo kxconfig**, troca-se para* **su -c "*kxconfig*"**.

** Não podemos usar apenas o **su** em um script pois ele abre um novo interpretador de comandos e só termina de ler o script quando o usuário digitar **exit**, nesse caso precisamos executar os comandos usando o* **su -c**.

Para mais informações consulte o manual:

```
$ man su
```

2.63. tail

Serve para visualizar o final de um arquivo. É muito útil para inspecionar arquivos de log onde a informação mais recente, e por isso mais relevante, fica no final.

(Contribuição, José Queiroz)

Há 3 formas de executar o comando **tail**:

- especificando uma quantidade de linhas a serem impressas no final do arquivo;
- especificando uma quantidade de bytes a serem impressos no final do arquivo;
- especificando para ler continuamente o final do arquivo.

Essas formas são selecionadas através de modificadores passados na linha de comando.

Sintaxe:

```
$ tail [opção] [arquivo]
```

Exemplo de utilização:

- Para ler as últimas 100 linhas do arquivo chamado "**arquivo.txt**":

```
$ tail -100 arquivo.txt
```

- Para ler os últimos 100 bytes do arquivo chamado "**arquivo.txt**":

```
$ tail -c 100 arquivo.txt
```

- Para ler as últimas 10 linhas do arquivo, e permanecer lendo continuamente:

```
$ tail -f arquivo.txt
```

Obs.: *Quando se usa a opção "-f", o comando **tail** lê continuamente o arquivo passado. Nesse caso, se algo for gravado no final do arquivo, o comando **tail** irá imprimir esses novos dados também. Essa característica é muito usada para visualizar os arquivos de log do sistema. Por exemplo, o comando "**tail -f /var/log/messages**" é tão comumente usado, que algumas instalações definem um alias "**tm**" para ele.*

Para mais informações consulte o manual:

```
$ man tail
```

2.64. tar

É utilizado para agrupar ou desagrupar arquivos e/ou diretórios. Também utilizado para compactação. Utiliza os compressores gzip (-z) e bzip2 (-j)

Sintaxe:

```
$ tar [opções] [arquivo.tar] [conteúdo a ser compactado]
```

Obs.: *A compactação com bzip2 (tar.bz2) possui uma melhor compressão.*

Exemplo de utilização:

- Para agrupar os arquivos "1", "2" e "3" em um único arquivo chamado "**scripts.tar**":

```
$ tar -cvf scripts.tar script1 script2 script3
```

- Para desagrupar os arquivos "1", "2" e "3" contidos em "**scripts.tar**":

```
$ tar -xvf scripts.tar
```

- Para visualizar o conteúdo do arquivo "**scripts.tar**":

```
$ tar -tvf scripts.tar
```

- Para compactar os arquivos "**musica1**", "**3**" e "**7**" dentro de um único chamado "**musicas.tar.gz**":

```
$ tar -cvzf musicas.tar.gz musica1.mp3 musica3.mp3 musica7.mp3
```

- Para extrair os arquivos "**musica1**", "**3**" e "**7**" contidos dentro de "**musicas.tar.gz**":

```
$ tar -xvzf musicas.tar.gz
```

- Para vizualizar o conteúdo do arquivo "**musicas.tar.gz**":

```
$ tar -tvzf musicas.tar.gz
```

- Para compactar os arquivos "**imagem.jpg**" "**livro.pdf**" e "**apostila.sxw**" no formato tar.bz2 (máxima compressão):

```
$ tar -cvjf documentos.tar.bz2 imagem.jpg livro.pdf apostila.sxw
```

- Para extrair os arquivos "**imagem.jpg**" "**livro.pdf**" e "**apostila.sxw**" do arquivo "**documentos.tar.bz2**":

```
$ tar -xvjf documentos.tar.bz2
```

- Para extrair o arquivo "**imagens.tar.bz2**" para dentro do diretório "**/tmp**". Note a opção "-C" maiúscula acrescentada na frente do diretório, em que se deseja descompactar o arquivo:

```
$ tar -xvjf imagens.tar.bz2 -C /tmp
```

- Para visualizar o conteúdo do arquivo "**documentos.tar.bz2**":

```
$ tar -tvjf documentos.tar.bz2
```

Dica: *Para quebrar um arquivo muito grande em pedaços menores, utilize o seguinte comando:*

```
$ split -d -b 650m arquivo_grande.tar.gz pedacos-
```

O comando acima irá quebrar o "**arquivo_grande.tar.gz**" em vários arquivos de 650MB com os nomes de pedacos-00, pedacos-01 e assim por diante. Para juntar os arquivos "pedacos-" para reconstituir o "**arquivo_grande.tar.gz**", use o comando abaixo:

```
$ cat pedacos-00 pedacos-01 > arquivo.tar.gz
```

Para mais informações consulte o manual:

```
$ man tar
```

2.65. top

Utilizado como monitor do sistema, mostra as atividades do processador e memória em tempo real. Exibe as tarefas que estão sendo executadas na CPU, como: PID (número do processo em execução), USER (usuário pertencente ao processo), %CPU (Porcentagem utilizada do CPU correspondente ao processo), %MEM (Porcentagem utilizada da memória correspondente ao processo) e COMMAND (nome ou linha do processo).

Sintaxe:

```
$ top [opção]
```

Exemplo de utilização:

- Para atualizar a tela de 5 em 5 segundos:

```
$ top -d 5
```

- Para executar o **top** em modo seguro:

```
$ top -s
```

- Para executar o **top** ignorando processos zumbis (processos que não estão mais rodando no sistema):

```
$ top -i
```

- Para mostrar a linha de comando ao invés do nome do programa:

```
$ top -c
```

Opções para serem utilizadas com o **top** em execução:

h ——— *Mostra um help dos comandos que podem ser utilizados no top.*

z ——— *Muda a cor do top. Útil para facilitar a visualização.*

[Espaço] ——— *Atualiza a tela imediatamente.*

i ——— *Ignora os processos ociosos.*

k ——— *Mata um processo.*

N ——— *Classifica os processos por número de PID.*

A ——— *Classifica os processos por período.*

P ——— *Classifica os processos por uso da CPU.*

M ——— *Classifica os processos por uso de memória.*

T ——— *Classifica os processos por tempo.*

u ——— *Mostra os processos de um usuário específico.*

n ——— *Lista um número de processos.*

s ——— *Especifica o tempo em segundos para a atualização da tela.*

W ——— *Cria um arquivo de configuração do top (**~/.toprc**).*

r ——— *Aplica um renice no processo.*

q ——— *Sai do **top**.*

Para mais informações consulte o manual:

```
$ man top
```

2.66. umount

Desmonta dispositivos, sejam eles locais ou remotos

(Contribuição, Luciano Martini)

Sintaxe:

```
# umount [opções] (ponto de montagem) [dispositivo]
```

Opções:

-v ——— *Ativa o modo verbose, ou seja, ativa uma descrição de ações mais completas.*

-n ——— *Faz com que o **umount** desmonte uma partição sem relatar no arquivo **/etc/mtab** (isso pode fazer com que o sistema acredite que a partição ainda está montada).*

-r ——— *Caso ocorra uma falha durante o momento de desmontar, essa opção leva a partição para o modo somente leitura (útil para o **fsck**).*

-a ——— *Esta opção tenta desmontar todos os dispositivos montados que estejam relatados no **/etc/mtab**.*

-t (sistema de arquivos) ——— *Esta opção permite que o usuário forneça manualmente o tipo de partição usada.*

-f ——— *Força a desmontagem do dispositivo.*

-O (opções) ——— *Permite especificar opções como no arquivo **/etc/fstab** para uma partição.*

-d ——— *Se um dispositivo estiver no modo loop, libera este dispositivo.*

Exemplo de utilização:

- Para desmontar a partição que tem como ponto de montagem "**/mnt/hda1**":

```
# umount /mnt/hda1
```

- Para desmontar a partição "hda2":

```
# umount /dev/hda2
```

- Para desmontar uma partição vfat em "**/dev/hda2**" com ponto de montagem "**/mnt/hda2**":

```
#umount -t vfat /dev/hda2 /mnt/hda2
```

Para mais informações consulte o manual:

```
$ man umount
```

2.67. unalias

Remove os nomes (**alias**) criados.

Sintaxe:

```
$ unalias [alias criado]
```

ou

```
$ unalias -a
```

Exemplo de utilização:

- Para remover o **alias** "**fazer-backup**":

```
$ unalias fazer-backup
```

- Para remover todos os **alias** criados pelo usuário:

```
$ unalias -a
```

Para mais informações consulte o manual:

```
$ man unalias
```

2.68. uname

Exibe informações sobre o sistema.

Sintaxe:

```
$ uname [opção]
```

Exemplo de utilização:

- Para exibir toda informação, na seguinte ordem abaixo:

```
$ uname -a
Linux horus 2.6.8-1-686-smp #1 SMP Thu Nov 25 04:55:00 UTC 2004 i686 GNU/Linux
```

- Para exibir o nome do **sistema operacional**:

```
$ uname -s
Linux
```

- Para exibir o nome da máquina:

```
$ uname -n
horus
```

- Para exibir a versão do **Kernel**:

```
$ uname -r
2.6.11-1-k7
```

- Para exibir a data em que o **Kernel** foi compilado:

```
$ uname -v
#1 Thu May 19 18:03:29 JST 2005
```

- Para exibir a **arquitetura do sistema**:

```
$ uname -m
i686
```

- Para exibir o **sistema operacional**:

```
$ uname -o
GNU/Linux
```

Para mais informações consulte o manual:

```
$ man uname
```

2.69. unzip

Descompacta, lista e testa arquivos compactados com **ZIP**.

Sintaxe:

```
$ unzip [opção] arquivo.zip
```

Exemplo de utilização:

- Para extrair o conteúdo do arquivo "**fotos.zip**":

```
$ unzip fotos.zip
```

- Para imprimir informações dos arquivos compactados dentro de "**fotos.zip**", como: nome, tamanho, data, hora, quantidade de arquivos etc:

```
$ unzip -l fotos.zip
```

- Para testar se há algum erro de compressão nos arquivos contidos dentro de "**fotos.zip**":

```
$ unzip -t fotos.zip
```

Para mais informações consulte o manual:

```
$ man unzip
```

2.70. upgradepkg

O **upgradepkg** é uma ferramenta do **pkgtool**, utilizado para atualizar o pacote instalado no sistema pela mais recente versão disponível.

Sintaxe:

```
# upgradepkg [nome do pacote]
```

Exemplo de utilização:

- Para atualizar o "**pacote.tgz**":

```
# upgradepkg pacote.tgz
```

Para mais informações consulte o manual:

```
$ man upgradepkg
```

2.71. uptime

Mostra a hora atual, quanto tempo o sistema está em funcionamento, ou seja, quanto tempo a máquina está ligada, o número de usuários conectados e a média de leitura do sistema sobre os últimos 1, 5, e 15 minutos.

Sintaxe:

```
$ uptime
```

Exemplo de utilização:

```
$ uptime
16:44:48 up 2:58, 1 user, load average: 0.45, 0.67, 0.63
```

Na ordem temos: hora atual, tempo em que a máquina está ligada, número de usuários e média de leitura.

Para mais informações consulte o manual:

```
$ man uptime
```

2.72. urpm

O **urpm** é um sistema de gerenciamento de pacotes similar ao **apt-get**, desenvolvido pela **Mandrake** para sua distribuição. Como o **apt-get** do **Debian**, o **urpm** instala e remove programas e, atualiza todo o sistema gerenciando sua base de dados.

Sintaxe:

```
# urpm [opções] [pacotes]
```

Exemplo de utilização:

- Para instalar o programa **xmms**:

```
# urpmi xmms
```

- Para instalar o **xmms** através da mídia "**contrib**":

```
# urpmi --media "contrib" xmms
```

- Para instalar o **xmms** e resolver as dependências:

```
# urpmi --auto xmms
```

- Para forçar a instalação do **xmms** sem checar por dependências. Use com cuidado esse comando, recomendo só no último caso. Caso não consiga instalar utilizando as opções descritas acima:

```
# urpmi --allow-force xmms
```

- Para selecionar automaticamente os pacotes necessários para atualizar todo o sistema. Este comando é parecido com o "**apt-get upgrade**" do **Debian**:

```
# urpmi --auto-select
```

- Para remover o programa **xmms**:

```
# urpme xmms
```

- Para remover o **xmms** e todos os arquivos que dependem dele:

```
# urpme --auto xmms
```

- Para procurar pelo pacote do **xmms:**

```
# urpmq xmms
```

- Para procurar pelo **xmms** somente na mídia **contrib:**

```
# urpmq --contrib xmms
```

- Para listar os pacotes existentes:

```
# urpmq --list
```

- Para listar os pacotes da mídia **contrib** existentes:

```
# urpmq --list --media contrib
```

- Para procurar por todos os pacotes instalados no sistema, cujo nome apresenta "**xmms**":

```
$ rpm -qa|grep xmms
```

- Para adicionar a mídia "**plf-free**" para a versão **10.2** do **Mandrake**:

```
# urpmi.addmedia plf-free ftp://ftp.planetmirror.com/pub/plf/
mandrake/free/10.2 with hdlist.cz
```

Nota: *O comando acima deverá ser digitado na mesma linha.*

- Para remover todas as mídias existentes:

```
# urpmi.removemedia -a
```

- Para atualizar todas as mídias atribuídas a base de pacote do **Mandrake**:

```
# urpmi.update -a
```

Dica: *Se você utiliza uma banda larga, remova todas as suas mídias (o padrão do* **urpmi** *é buscar somente na mídia "**main**" que contém os pacotes existentes no CD-ROM) e instale novas, buscando no site* ***http://easyurpmi.zarb.org/:***.

Para instalar novas mídias, siga os passos abaixo:

1. Escolha a versão do seu **Mandrake** em "**Mandrake version:**", logo em seguida escolha a arquitetura que utiliza em "**and architecture:**". Se não souber, deixe marcado i586 e clique em **"proceed to step 2"**;
2. Essa é a parte em que você escolhe as mídias que serão instaladas. Você pode selecionar todas sem nenhum problema, mas caso não queira, selecione pelo menos as três primeiras: "**contrib**", "**main**" e "**updates**";
3. Selecione o servidor de onde deseja baixar as fontes, use o que estiver mais próximo, note que existe um servidor do Brasil.
4. Selecionado as mídias e escolhido os servidores, clique em "**proceed to step 3**", note que irá gerar as mídias junto com seus respectivos servidores, basta adicionar uma por uma. Após adicionar, atualize todas as mídias.

Para mais informações consulte o manual:

```
$ man urpm
```

2.73. users

Lista os nomes de usuários (**login**) logados no sistema.

Sintaxe:

```
$ users
```

Exemplo de utilização:

```
$ users
root tales
```

O comando "**users**" exibiu todos os usuários logados no sistema, nesse caso foram o "**root**" e "**tales**".

Para mais informações consulte o manual:

```
$ man users
```

2.74. w

Exibe os usuários que estão utilizando o sistema, e os seus processos.

Sintaxe:

```
$ w [opções] [usuário]
```

Exemplo de utilização:

```
tales@horus:~$ w
18:06:16 up 6:36, 2 users, load average: 0,67, 0,73, 0,83
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
root tty1 - 18:05 51.00s 0.48s 0.48s -bash
tales :0 - 11:31 ?xdm? 1:12m 0.00s -:0
```

Descrição:

- **USER** ——— *Usuários logados no sistema.*
- **TTY** ——— *Nome do terminal em que o usuário se encontra logado.*
- **FROM** ——— *De onde o usuário logou, exemplo: **ssh**, **telnet**, localmente etc.*
- **LOGIN@** ——— *Horário em que o usuário efetuou o **login** no sistema.*
- **IDLE** ——— *Tempo de ociosidade.*
- **JCPU** ——— *Mostra o tempo usado por todos os processos unidos ao **tty**.*
- **PCPU** ——— *Mostra o tempo usado pelo processo atual.*
- **WHAT** ——— *Qual programa o usuário está rodando.*

Para mais informações consulte o manual:

```
$ man w
```

2.75. wget

Gerenciador e restaurador de downloads. Com certeza esse programa é um dos mais utilizados e essencial para o **Linux**.

Sintaxe:

```
$ wget [opções] [URL]
```

Exemplo de utilização:

- Para baixar o arquivo "**pacote3.deb**" para o diretório atual:

```
$ wget http://linuxhard.org/downloads/pacote3.deb
```

- Para baixar o arquivo "**kalango.iso**". A opção "**-c**" é utilizada para reiniciar o download de onde parou. Muito utilizado para baixar arquivos grandes:

```
$ wget -c http://www.linorg.usp.br/iso/kalango.iso
```

- Para baixar o site inteiro para a máquina, ou seja, todos arquivos contidos no site:

```
$ wget -m http://tales.linuxhard.org/
```

- Para baixar somente as imagens no formato "**img.jpg**" que se encontram no site. A opção "-**nd**" é para não baixar diretórios e a "-**A**" para selecionar o tipo de arquivo a ser baixado:

```
$ wget -m -nd -A ".jpg" http://www.wallpapers.com/
```

Para mais informações consulte o manual:

```
$ man wget
```

2.76. whereis

Procura por binários, arquivos fonte, arquivos de configuração e páginas do manual.

Sintaxe:

```
$ whereis [opção] [comando ou nome do programa]
```

Exemplo de utilização:

- Para procurar apenas por binários. No exemplo abaixo o comando **whereis** procura por binários do programa **mplayer**:

```
$ whereis -b mplayer
```

- Para procurar somente nas seções do manual. No exemplo abaixo o comando **whereis** irá procurar por manuais do programa **xmms**:

```
$ whereis -m xmms
```

- Para procurar apenas por sources (arquivos de código fonte). No exemplo abaixo o comando **whereis** irá procurar pelos arquivos fonte do **Kernel**:

```
$ whereis -s kernel
```

Para mais informações consulte o manual:

```
$ man whereis
```

2.77. who

Mostra quem está acessando o sistema. Exibe informações sobre os usuários logados, incluindo data, hora e quem entrou e saiu do sistema.

Sintaxe:

```
$ who [opção] [Arquivo]
```

Exemplo de utilização:

- Para exibir quais usuários estão acessando o sistema:

```
$ who
```

- Para exibir a hora do último boot do sistema:

```
$ who -b
```

- Para exibir processos mortos:

```
$ who -d
```

- Para exibir os processos de login do sistema:

```
$ who -l
```

- Para exibir todos os nomes e o número de usuários logados no sistema:

```
$ who -q
```

Para mais informações consulte o manual:

```
$ man who
```

2.78. whoami

Exibe o nome de login do usuário no sistema.

Sintaxe:

```
$ whoami
```

Exemplo de utilização:

```
tales@horus:~$ whoami
tales
```

O comando acima exibiu o nome de usuário "tales" que é quem está logado no sistema.

Para mais informações consulte o manual:

```
$ man whoami
```

2.79. yum

O **yum** é um ótimo gerenciador de pacotes, muito utilizado nas distribuições **Red Hat** e **Fedora**, com ele você pode instalar, remover e atualizar pacotes e atualizar todo o sistema com um simples comando. Os comandos são simples de guardar, pois se assemelham muito aos comandos do **atp-get**.

Sintaxe:

```
# yum [comando] [pacote]
```

Exemplo de utilização:

- Para atualizar a lista de pacotes:

```
# yum update
```

- Para atualizar a lista de pacotes e mostrar quais pacotes serão atualizados:

```
# yum check-update
```

- Para atualizar um ou mais pacotes:

```
# yum update [nome do pacote]
```

- Para atualizar todos os pacotes instalados no sistema:

```
# yum upgrade
```

- Para instalar o pacote especificado:

```
# yum install [nome do pacote]
```

- Para remover o pacote especificado:

```
# yum remove [nome do pacote]
```

- Para mostrar uma lista de programas disponíveis:

```
# yum list available
```

- Para procurar por todos os pacotes que apresentam a "palavra chave":

```
# yum search [palavra chave]
```

- Para mostrar as informações sobre um determinado pacote:

```
# yum info [nome do pacote]
```

Obs.: *Para manter sempre o* ***yum.conf*** *(lista de mirros) atualizados, proceda da seguinte maneira:*

```
# rm -f /etc/yum.conf
# wget /etc/http://www.fedorafaq.org/samples/yum.conf
```

Para mais informações consulte o manual:

```
$ man yum
```

2.80. zip

Compactador de arquivos. O compactador zip é considerado um ótimo programa pois além de ser compatível com todos os sistemas operacionais apresenta uma garantia com o arquivo. Caso o "**arquivo.zip**" copiado esteja corrompido, o mesmo não irá abrir, acusando erro.

Sintaxe:

```
$ zip [opção] [arquivo.zip] [conteúdo]
```

Exemplo de utilização:

- Para compactar os arquivos 1, 2 e 3 dentro do arquivo "**documentos.zip**".

```
$ zip documentos.zip arquivo1 arquivo2 arquivo3
```

- Para compactar recursivamente os diretórios de configuração "**.kde**" e "**.mozilla**", dentro de "**backup.zip**".

```
$ zip -r backup.zip ~/.kde ~/.mozilla
```

Para mais informações consulte o manual:

```
$ man zip
```

2.81. Comandos Específicos das Distros

Conectiva

Debian

Fedora

Gentoo

Mandrake

Red Hat, Suse

Slackware

3
Dicas Avançadas

3. Dicas Avançadas

3.1. Ambiente Gráfico

O **Linux**, por ser um sistema multiusuário, possibilita que o usuário trabalhe com vários terminais. Para acessar os terminais, basta teclar simultaneamente "**CTRL+ALT+Fn**" sendo "n" o número do ambiente desejado.

Sintaxe:

```
$ startx -- :n
```

ou

```
$ xinit -- :n
```

Sendo "n" um número de 1 a 6.

Exemplo de utilização:

Abra um terminal em modo texto, faça o login com o user que desejar, tecle **CTRL+ALT+F2** e digite:

```
tales@horus:~$ startx -- :2
```

Você também pode abrir terminais gráficos utilizando o comando "**xinit**":

```
tales@horus:~$ xinit -- :2
```

3.1.1. Vários Ambientes X

A diferença do **xinit** para o **startx** é que, no **xinit** é aberto um **servidor X**, apenas com um terminal de comandos, possibilitando abrir o gerenciador de janelas desejável como **KDE**, **GNOME**, **FLUXBOX**, **ICEWM** etc. Já no **startx** é aberto o gerenciador de janelas que estiver já configurado sem que se possa escolher qual.

Para retornar ao terminal gráfico (padrão) que se encontrava, basta teclar **CTRL+ALT+F7**.

Para acessar o terminal gráfico que foi aberto, tecle **CTRL+ALT+F8** (caso tenha sido executado o comando no terminal **F2**), ou no **F9** se for executado no terminal **F3** e assim em diante.

Para mais informações consulte os manuais:

```
$ man xinit
```

```
$ man startx
```

3.2. Terminal

3.2.1. Compilando Programas

Muitos já devem ter ouvido falar em "compilar um programa no **Linux**". Para realizar essa façanha, não é preciso ser nenhum expert, ou mesmo saber programar. Basta apenas ter perseverança e paciência, pois nem sempre consegue-se compilar um programa facilmente. Isso acontece por vários motivos. Os mais prováveis são que, você não possui os compiladores necessários, como **gcc**, **make**, entre outros, para compilação, ou falta alguma **biblioteca** necessária, ou existe um **conflito** entre as versões dos arquivos, entre outras coisas. Na verdade "**compilar um programa**" depende muito do programa que irá ser compilado, pois muitos programas não seguem um "padrão".

Geralmente os **códigos fontes** (**source**) dos programas para serem compilados vem nos formatos:

```
arquivo.tar.gz
arquivo.tar.bz2
```

Como exemplo de compilação será utilizado o programa "**mplayer**".

Obs.: *A compilação que será mostrada é válida para compilar a maioria dos programa.*

O primeiro passo é baixar o arquivo para a máquina e descompactá-lo.

```
$ tar -xvjf Mplayer-*.tar.bz2
```

Quando você pega um source para ser compilado, o mesmo vem compactado. Após descompactá-lo, é extraído um diretório onde se encontram os arquivos para que o programa possa ser compilado. Alguns arquivos importantes são: **README**, **INSTALL** e **configure**.

Após descompactar o arquivo "**Mplayer-*.tar.bz2**", será extraído o diretório "**Mplayer-***", onde encontram-se os arquivos para compilação. Entre no diretório para começar a compilação.

```
$ cd Mplayer-*
```

Antes de sair compilando, é importante ler o **README** ou **INSTALL** para obter mais informações sobre como instalar o programa. Após obter essas informações vá para a execução do primeiro comando e comece a compilação.

```
$ ./configure
```

Este comando é utilizado para configurar o programa antes que seja compilado. Por exemplo, no caso do **mplayer**, caso queira compilar o programa na linguagem português do Brasil, é utilizado o parâmetro:

```
$ ./configure --language=pt_BR
```

Para saber de todas as informações que seu programa dispõe para ser utilizada a compilação, utilize o seguinte comando:

```
$ ./configure --help
```

Irão aparecer várias opções. Para utilizar mais de uma coloque espaço ( ) entre uma opção e outra.

Após executar o comando "**./configure**", verifique se não aparece nenhum erro no final da execução, caso não apareça, comece a compilar o programa.

Obs.: *Se aparecer algum erro, você terá que resolvê-lo. Não poderá passar para o próximo passo que é a compilação. Isso também vale para o comando* ***make****, você só poderá prosseguir para o próximo passo, caso não haja nenhum erro. No caso do comando "****./configure****" é gerado um log (****configure.log****) com todas as informações de configuração que poderão lhe auxiliar para resolver o erro.*

Dependendo do desempenho de sua máquina e do programa a ser compilado, esse processo pode demorar alguns minutos.

```
$ make
```

Espere a compilação terminar, caso não haja erros, vá para o próximo e último passo. Apresentando erros, resolva-os primeiro antes de seguir em frente.

```
# make install
```

Note que este último comando requer privilégios do **root** (**#**), pois irá instalar o programa que foi compilado no sistema.

Caso não apareça nenhum erro, você conseguiu compilar o programa e já pode utilizá-lo.

3.2.2. Reorganizando o Seu Home

Quando você faz uma instalação do **Linux**, configura todo o sistema, nota que a máquina apresenta uma certa performance. Com o passar do tempo, instalando, removendo programas e configurando arquivos, essa performance pode cair e apresentar o famoso "**pau**" (problemas que atrapalham o desempenho da máquina). Pode começar a surgir problemas com o ambiente

gráfico, por exemplo, se sumir com a barra de tarefas, o navegador não abrir mais, ou você configurar um programa indevidamente, como por exemplo, uma senha que não se recorda e outras milhares de coisas que podem aparecer ligadas ao que acontece em seu home. Isso pode ser facilmente refeito retornando os programas para suas configurações padrões. Veja os exemplos abaixo:

Suponhamos que você tenha instalado o navegador Opera, e mexendo nas configurações desaparece com a barra de menus e não consegue retorná-la. Uma das coisas simples para se fazer é retornar com as configurações padrões do navegador. Isso pode ser feito removendo o diretório oculto do Opera onde encontram-se todas as configurações do navegador. Para a remoção do diretório, primeiro feche o navegador e logo em seguida execute o comando em um terminal de comandos:

```
$ rm -rf ~/.opera
```

Abra o navegador novamente e a barra de menus voltará.

Obs.: *Geralmente o nome do diretório oculto é o nome do programa.*

Esse procedimento pode ser realizado em qualquer diretório oculto que se encontre no seu home. Para localizar todos os diretório ocultos existentes no seu home, utilize o comando:

```
$ ls -la ~/
```

Olhe na primeira fileira da esquerda, onde aparecem as permissões. Tudo o que estiver com o "d" na frente significa que é um diretório, basta olhar se o diretório começa com um ponto (.) na frente do nome, se for, é um diretório de configuração e pode ser removido caso aconteça problemas referentes ao programa utilizado.

Obs.: *É importante fazer backup de certos diretórios antes de removê-los, por exemplo, o diretório "***.kde***" apresenta configurações de vários programas, como* ***konqueror***, ***sim***, ***k3b***, ***knotes*** *etc. Os mesmos podem ser encontrados em "***~/.kde/share/apps/***".*

3.2.3. Permissão

Como foi descrito rapidamente sobre as permissões em "**Conceitos Básicos**", irei descrever mais detalhadamente como trabalhar com as permissões nos dois modos – literal e numérico. O primeiro (literal) é pouco utilizado por ser mais extenso de se aplicar e ser preciso digitar mais linhas. O segundo (numérico) é utilizado com muita freqüência por sua facilidade,

praticidade e apresentar apenas uma seqüência de três números – em sua forma básica – que constituem a permissão.

Primeiramente o comando que dá ou retira a permissão de um arquivo/diretório é o "**chmod**".

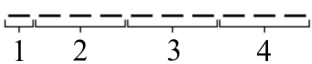
As três partes que compõem a permissão de um arquivo são:

Dono ——— *Quem criou e a quem pertence o arquivo.*

Grupo ——— *Usuários no mesmo grupo compartilham as mesmas autorizações de acesso.*

Outros ——— *Não são donos nem pertencem aos grupos.*

Composição das permissões:



1. Identifica se trata de um arquivo, caso apareça com um traço(-), ou diretório caso apareça com a letra "d".
2. Dono.
3. Grupo.
4. Outros.

r ——— *Read (permite apenas leitura).*

w ——— *Write (permite apenas gravação).*

x ——— *Execute (permite apenas execução de arquivos e acesso a diretórios).*

3.2.3.1. Modo Literal

Opções:

u ——— *dono.*

g ——— *grupo.*

o ——— *outros.*

a ——— *todos.*

+ ——— *adiciona permissão.*

- ——— *retira permissão.*

Exemplos de permissão literal:
Exemplo 1:

```
$ chmod u+rw,g+r,o+r senhas.txt
```

O arquivo ficará da seguinte maneira:

```
-rw-r--r--
```

No exemplo acima, o arquivo "**senhas.txt**" recebeu permissão do dono (u+rw) de ler e gravar, do grupo (g+r) e outros (o+r) de ler.

Exemplo 2:

```
$ chmod a+rx senhas.txt
```

O arquivo ficará da seguinte maneira:

```
-r-xr-xr-x
```

No exemplo acima, o arquivo "**senhas.txt**" recebeu permissão do dono (a+rx), grupo (a+rx) e outros (a+rx) de ler e executar.

Exemplo 3:

Com base na permissão do "Exemplo 1" será aplicada a seguinte permissão:

```
$ chmod u-w senhas.txt
```

O arquivo ficará da seguinte maneira:

```
-r--r--r--
```

Note que foi retirado do arquivo "**senhas.txt**" a opção de gravar para o dono.

Exemplo 4:

Com base na permissão do "Exemplo 3" será aplicada a seguinte permissão:

```
$ chmod +x senhas.txt
```

O arquivo ficará da seguinte maneira:

```
-r-xr-xr-x
```

Foi adicionado o direito de execução para o dono, grupo e outros.

3.2.3.2. Modo Numérico

A parte de permissões gerenciadas por números é muito mais prática, fácil e usual, pois utiliza uma seqüência de apenas três números para aplicar uma permissão. Veja o esquema abaixo que relaciona números com letras.

Obs.: *Para facilitar a leitura, use sempre a seqüência mostrada abaixo, que é mais simples, portanto mais fácil de memorizar qual número corresponde a qual letra.*

r ——————— *4 (Ler).*

w ——————— *2 (Gravar).*

x ——————— *1 (Executar).*

Por que utilizar a seqüência acima? Muito simples, a leitura é sempre feita da esquerda para a direita, como usamos normalmente para escrever. Veja a permissão abaixo:

```
-rwxrwxrwx
```

Note que a seqüência não muda, ou seja, sempre aparecerá as letras (**rwx**) na mesma ordem. Para os números, basta apenas decrescer (4, 2, 1). Com esses três números é possível formar qualquer tipo de permissão. Aplicando-os individualmente ou combinando-os com uma soma entre os mesmos. Veja a seguir todas as seqüências numéricas que podem ser aplicadas para formar as permissões:

0 ——————— *Sem permissões.*
1 ——————— *Permissão de executar.*
2 ——————— *Permissão de gravar.*
3 ——————— *Permissão de gravar/executar (2+1=3).*
4 ——————— *Permissão de ler.*
5 ——————— *Permissão de ler/executar (4+1=5).*
6 ——————— *Permissão de ler/gravar (4+2=6).*
7 ——————— *Permissão de ler/gravar/executar (4+2+1=7).*

Exemplos de permissão numérica:

- Exemplo 1:

```
$ chmod 644 senhas.txt
```

O arquivo ficará da seguinte maneira:

```
-rw-r--r--
```

No exemplo acima, o arquivo "**senhas.txt**" recebeu permissão do dono (6) de ler e gravar, do grupo (4) e outros (4) de ler. Note que este exemplo é o mesmo utilizado no "Exemplo 1" literal, apenas foi aplicado com números.

- Exemplo 2:

```
$ chmod 555 senhas.txt
```

O arquivo ficará da seguinte maneira:

```
-r-xr-xr-x
```

No exemplo acima, o arquivo "senhas.txt" recebeu permissão do dono (5), grupo (5) e outros (5) de ler e executar. Note que este exemplo é o mesmo utilizado no "Exemplo 2" literal, apenas foi aplicado com números.

Obs.: *Dependendo do tipo de permissão que será aplicada, pode ser que seja mais vantajoso utilizar o modo literal, como nos exemplos 3 e 4 literais. No exemplo 3 foi retirada a permissão apenas do dono e no exemplo 4 foi adicionada a permissão de execução para todas as partes.*

3.2.4. Partições no Linux

O estudo sobre partições é um pouco mais extenso do que o descrito aqui, mas será útil para a grande parte dos casos.

3.2.4.1. Tipos de Partições

Para **HD's** do tipo **IDE**, o **Linux** identifica as partições como **hdxy**. Para **HD's** do tipo **SCSI**, o **Linux** identifica as partições como **sdxy**.

3.2.4.2. Conhecendo as Letras

O "**hd**" indica que o **HD** é um **IDE**, e o "**sd**" indica que o **HD** é um **SCSI**.
O "**x**" pode assumir o valor das letras (a, b, c, d), já o "**y**" assume um número começando pelo 1 e aumentando de acordo com o número de partições.

3.2.4.3. Conhecendo as Partições

hda ou sda ———— *Primária Master.*
hdb ou sdb ———— *Primária Slave.*
hdc ou sdc ———— *Secundária Master.*
hdd ou sdd ———— *Secundária Slave.*

O **Linux** pode ter 4 partições primárias, ou 3 primárias e uma partição extendida – nesta extendida pode-se criar até 256 partições lógicas.

3.2.4.4. Partição Swap (Troca)

A partição **swap** é utilizada para suprir a memória **RAM** sempre que a memória física se esgota. Para memórias **RAM** de até **256MB**, é recomendável criar uma partição com o dobro desse tamanho. Com memórias acima de 256MB, poderá ser criada uma partição **swap** com esse mesmo tamanho.

3.2.5. Criando um Firewall

Um **firewall** simples com o **Iptables**.

Todo administrador de redes aprende logo que uma das coisas mais importantes para qualquer rede é um bom **firewall** (proteção). Embora existam

muitos mitos em torno disso, os **firewalls** não fazem milagres, apenas adicionam uma camada extra de proteção, escondendo as vulnerabilidades das máquinas.

(Contribuição, Carlos E. Morimoto)

Você pode ter um servidor **IIS** ativo com todas as vulnerabilidades possíveis dentro da sua rede, mas ninguém poderá fazer nada se não conseguir se conectar à ele. Este é o papel do **firewall**, limitar e filtrar os acessos aos servidores e estações de trabalho da sua rede.

Existem vários tipos de **firewall**, de todos os preços. Os tipos mais simples e ao mesmo tempo mais eficazes para PCs domésticos são os **firewalls** de bloqueio, onde você simplesmente fecha todas as portas do micro (ou deixa aberta apenas as portas de que você realmente precisa). Se ninguém consegue se conectar ao seu PC, 90% das brechas de segurança são anuladas.

Outro ponto comum é a necessidade de compartilhar a conexão com a **Web**. Isto permite que dois ou mais micros usem a mesma conexão, sem a necessidade de contratar um serviço adicional para isso.

Isso pode ser feito facilmente através do **Iptables**. A receita funciona em qualquer distribuição que utilize um **Kernel** acima do **2.2**. Basicamente qualquer coisa que você ainda possa querer usar hoje em dia.

Existem vários programas gráficos para a configuração de **firewalls**, como por exemplo o **GuardDog** e o **Shorewall** (usando no **RedHat** e **Mandrake**). Estes programas também trabalham com o **Iptables**, servindo apenas para facilitar a configuração, criando as regras a partir das escolhas feitas pelo usuário.

A configuração do **Iptables** também pode ser feita diretamente via terminal, bastando que você insira as regras uma a uma. Como as regras se perdem ao reiniciar o micro, deve-se criar um **script** para que elas sejam recriadas automaticamente a cada **reboot**.

O **Iptables** é tão versátil que pode ser usado para quase tudo relacionado a inspeção, encaminhamento e até mesmo alteração de pacotes. Se ele não fizer algo é possível criar um módulo que o faça, já que as suas possibilidades são infinitas, mas seu tempo não.

Fique com algumas regras simples que resolvem a maior parte dos problemas do dia a dia. A partir daí você pode ir se aperfeiçoando e desenvolvendo soluções mais sofisticadas.

Antes de mais nada, você precisa verificar se o pacote do **iptables** está instalado. Se você estiver no **Mandrake** basta dar um "**urpmi iptables**". Se você estiver no **Debian**, **Kurumin** ou **Conectiva**, um "**apt-get install iptables**" resolve.

Vamos então à criação das regras que determinam o que entra e o que não entra na máquina. Se o seu micro está ligado apenas à internet, sem uma rede

local, então são necessárias apenas duas regras para resolver o problema. Abra um terminal, faça o login como root e digite os comandos:

```
# iptables -A INPUT -p tcp --syn -j DROP
# iptables -A INPUT -i ppp0 -p udp --dport 0:30000 -j DROP
```

Isso fará com que o micro passe a ignorar conexões vindas em qualquer porta **TCP**, sem enviar sequer uma confirmação de que o pacote foi recebido. Você continuará conseguindo acessar a internet normalmente, mas ninguém conseguirá se conectar diretamente ao seu PC, um servidor **Web** ou **SSH** que você esqueça de desativar passaria despercebido. Apenas as conexões iniciadas por você são aceitas, o que permite que alguns programas de compartilhamento como o **gtkgnutella** e o **Kazza** continuem funcionando normalmente.

A segunda regra é opcional *(dica do Fabrício Carvalho)*, ela bloqueia também parte das portas **UDP** adicionando uma camada extra de segurança.

O efeito colateral é que alguns programas que abrem servidores podem deixar de funcionar. Você não conseguirá mais receber arquivos pelo **ICQ**, por exemplo, se estivesse acessando através de uma conexão compartilhada via **NAT**.

O interessante é que você pode desativar o **firewall** a qualquer momento, para isso basta um único comando:

```
# iptables -F
```

Isso elimina todas as regras do **Iptables**, fazendo com que seu micro volte a aceitar todas as conexões. Você pode usá-la para permitir que alguém se conecte rapidamente via **ssh** na sua máquina, por exemplo, e depois fechar tudo novamente reinserindo as regras anteriores.

Se você tiver uma rede local e quiser que os micros da rede interna sejam capazes de se conectar normalmente, mas mantendo o bloqueio a tudo que vem da internet, basta dar um "**iptables -F**" e começar de novo, desta vez adicionando primeiro a regra que permite os pacotes vindos da rede local:

```
# iptables -A INPUT -p tcp --syn -s 192.168.0.0/255.255.255.0 -j ACCEPT
```

Em seguida vem os comandos anteriores:

```
# iptables -A INPUT -p tcp --syn -j DROP
# iptables -A INPUT -i ppp0 -p udp --dport 0:30000 -j DROP
```

Altere o "**192.168.0.0/255.255.255.0**" para a faixa de endereços e máscara de subrede que estiver utilizando em sua rede. Este exemplo serve para redes que utilizam a faixa de **IP 192.168.0.1** até **192.168.0.254.**

O **Iptables** processa os comandos em seqüência. Então todos os pacotes passam pela primeira instrução antes de ir para a segunda. Quando um pacote vem de um dos endereços da rede local é imediatamente aceito. Os demais

vão para as duas últimas linhas e acabam recusados. É uma simples questão de sim ou não. A primeira linha diz sim para os pacotes da rede local, enquanto as duas últimas dizem não para todos os demais.

Imagine agora que você queira permitir ao mesmo tempo pacotes vindos da rede local e uma certa porta vinda da Internet, como por exemplo a **porta 22** do **SSH**. Neste caso você adicionaria mais uma regra, mantendo as regras anteriores:

```
# iptables -A INPUT -p tcp --destination-port 22 -j ACCEPT
# iptables -A INPUT -p tcp --syn -s 192.168.0.0/255.255.255.0 -j ACCEPT
# iptables -A INPUT -p tcp --syn -j DROP
# iptables -A INPUT -p udp -j DROP
```

Agora tudo o que vem na **porta 22** (tanto da **Internet** quanto da **rede local**) é aceito, tudo o que vem da rede local é aceito e todo o resto é rejeitado. Você pode adicionar mais linhas para abrir outras portas. Se você quiser abrir também as portas **1021** e **1080**, a lista ficaria assim:

```
# iptables -A INPUT -p tcp --destination-port 22 -j ACCEPT
# iptables -A INPUT -p tcp --destination-port 1021 -j ACCEPT
# iptables -A INPUT -p tcp --destination-port 1080 -j ACCEPT
# iptables -A INPUT -p tcp --syn -s 192.168.0.0/255.255.255.0 -j ACCEPT
# iptables -A INPUT -p tcp --syn -j DROP
```

Isso permite que você mantenha disponíveis apenas os servidores que você realmente quer disponibilizar e nos momentos que quiser. A qualquer tempo você pode dar um "**iptables -F**" e readicionar apenas as regras para fechar tudo.

Para abrir de uma determinada porta a outra, ou seja, uma seqüência de portas, usa-se os dois pontos (**:**) para separar a seqüência. Veja um exemplo abaixo:

```
# iptables -A INPUT -p tcp --destination-port 6891:6893 -j ACCEPT
```

O comando acima irá abrir as portas **6891**, **6892** e **6893**, que são as portas de recebimento de arquivos do **Amsn**.

3.2.6. Compartilhando a Conexão

Vamos então à segunda receita para compartilhar a conexão. Ela é ainda mais simples e também permite ativar ou desativar o compartilhamento a qualquer momento.

Em primeiro lugar, você deve configurar as suas **placas de rede** e **modem e**, verificar se tanto a conexão com a Internet quanto a conexão com os micros da rede local estão funcionando normalmente. O compartilhamento da

conexão em si pode ser feito com apenas três comandos. Para compartilhar a conexão do **modem** com a **rede local**:

```
# modprobe iptable_nat
# iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Para compartilhar uma conexão via **ADSL** ou cabo instalada na **eth0**:

```
# modprobe iptable_nat
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Para desativar o compartilhamento, você deve usar o comando:

```
# iptables -t nat -F
```

Isso mesmo, é só isso...!!

O compartilhamento é ativado ou desativado imediatamente sem que seja necessário reiniciar a conexão. Rápido, prático e confiável.

As três linhas respectivamente ativam o módulo **nat** do **iptables**, responsável pela tradução de endereços, avisam para o **iptables** que ele deve direcionar todas as conexões recebidas para a interface **ppp0** (**o modem**) ou **eth0** (**a primeira placa de rede**) e devolver as respostas para os clientes e confirmar a ativação no arquivo de configuração do **TCP/IP**.

Não faz mal se você acessa via modem e não fica permanentemente conectado. A regra mantém o compartilhamento ativo mesmo que você desconecte e reconecte várias vezes.

Se os clientes da rede já estiverem configurados para acessar a **web** através do endereço **IP** usado pelo **servidor** (**192.168.0.1** se você quiser substituir uma máquina **Windows** compartilhando através do **ICS**), você já deve ser capaz de acessar a web automaticamente nos demais PCs da rede.

Uma observação, estas regras não incluem um servidor **DHCP**, você deve configurar os clientes com endereço **IP** fixo ou então ativar o serviço **DHCPD** na sua distribuição. No **Mandrake** ou **RedkHat** basta ativar o serviço no painel de controle e o **DHCP** irá funcionar automaticamente.

A configuração nos clientes fica:

Endereço IP: ——— *Qualquer endereço dentro da faixa de endereços usada pelo servidor.*
Ex: **192.168.0.3**

Servidor DNS: —— *Os endereços dos servidores* **DNS** *do seu provedor.*
Ex: **200.177.250.10**

Gateway Padrão: —— *O endereço do servidor.*
Ex: **192.168.0.1**

Domínio:———— *O domínio do seu provedor.*

Ex: **terra.com.br**

As linhas de compartilhamento da conexão não conflitam com as regras de **firewall** que vimos anteriormente, você deve apenas ter o cuidado de colocá-las no início da seqüência. Neste caso nosso script completo ficaria assim:

```
# Carrega o módulo para compartilhamento
modprobe iptable_nat
# Compartilha a conexão
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
echo 1 > /proc/sys/net/ipv4/ip_forward
# Abre algumas portas (opcional)
iptables -A INPUT -p tcp --destination-port 22 -j ACCEPT
iptables -A INPUT -p tcp --destination-port 1021 -j ACCEPT
iptables -A INPUT -p tcp --destination-port 1080 -j ACCEPT
# Abre para a rede local
iptables -A INPUT -p tcp --syn -s 192.168.0.0/255.255.255.0 -j ACCEPT
# Fecha o resto
iptables -A INPUT -p tcp --syn -j DROP
```

Se você quiser que o PC também não responda a **pings**, adicione a linha:

```
# echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_all
```

Mais uma linha interessante de se adicionar, que protege contra pacotes danificados (usados em ataques **DoS** por exemplo) é:

```
# iptables -A FORWARD -m unclean -j DROP
```

Esta linha deve ser adicionada antes das demais.

Obs.: *Nem todas as distribuições implementam o módulo* ***unclean****; assim, antes de usar esta linha num script, teste-a em linha de comando, para garantir que ela é aceita pelo* **iptables**.

3.2.7. Automatizando a Inicialização do Firewall

Agora já temos 10 comandos, fora os utilizados para abrir portas específicas. Não seria muito prático ficar digitando tudo isso cada vez que for preciso reiniciar o micro. Para automatizar isso, basta colar todos os comandos dentro de um arquivo de texto. Você pode salvá-lo como por exemplo:

```
/usr/local/bin/meu_firewall
```

Em seguida, dê permissão de execução para o arquivo (**chmod +x /usr/local/bin/meu_firewall**) e você terá um shell script que pode ser chamado a qualquer momento, bastando digitar: **meu_firewall**

Para tornar a inicialização realmente automática, você precisa apenas colocar o comando em um dos arquivos de inicialização do sistema. Abra o arquivo **/etc/rc.d/rc.local** e adicione a linha:

```
/usr/local/bin/meu_firewall
```

No **Debian** e **Kurumin** você pode usar o arquivo:

```
/etc/init.d/bootmisc.sh.
```

As regras que vimos acima funcionam como um **firewall** de bloqueio, ou seja, o servidor não deixa que ninguém acesse os compartilhamentos de arquivos ou conectem o **backorifice** instalado na máquina com o **Windows 98**. Mas não impedem que os usuários baixem e-mails com vírus ou que acessem uma página **web** que explore alguma das vulnerabilidades do **IE** por exemplo.

Ao usar clientes **Windows**, o ideal é complementar o **firewall** com um bom **anti-vírus**.

Observações Importantes:

1. *Esse **firewall** é muito simples, e só protege a própria máquina que faz o compartilhamento.*
2. *Esse firewall não inclui recursos como **nat** reverso (redirecionamento de portas).*

Sugiro que esse código não seja utilizado em ambientes de produção. O intuito aqui é mostrar como fazer um **firewall** para uso doméstico e não empresarial, que requer **firewall's** mais complexos.

Glossário

Árvore de Diretório
Organizada a estrutura dos arquivos, dividindo-os em: raiz, diretórios e subdiretórios.

Bash
O bash é uma espécie de tradutor entre o sistema operacional e o usuário. Permite a execução de seqüências de comandos escritas em arquivos de texto. São os scripts de shell ou shell scripts.

Comando
Dá-se este nome ao texto que executa um ou vários programas. Ex.:

```
$ man xinit
```

Este é o comando que executa o programa que abre o manual do "**xinit**"

Desktop
Termo utilizado para designar o computador de mesa, em geral um IBM-PC ou Mac, rodando os sistemas Windows, MacOS, Linux ou Unix. Este termo também pode ser empregado para o ambiente gráfico de um programa onde estão presentes uma ou mais janelas de uma área de trabalho.

Diretório
Local (também chamado de pastas) onde ficam armazenados os arquivos.

DNAT
É a sigla usada para Destination Network Address Translation (Troca de Endereço de Rede de Destino), são linhas de comando ou scripts que permitem a conexão de uma rede local a um link (internet) ou outra rede.

Espelhamento
O espelhamento, também chamado RAID, é o método pelo qual o conteúdo de um disco rígido (HD), ou no caso, uma parte do conteúdo (partição do HD) seja inteiramente copiado para outro HD ou partição, de forma automática. Na falha de um o outro entra em ação automaticamente.

Firewall
É um mecanismo de filtragem de dados que somente permite a transmissão e recepção de dados autorizados. O **Firewall** é um mecanismo extremamente importante para a segurança das redes de computadores, principalmente na internet, pois evita, ou diminui a possibilidade de invasão dos computadores e conseqüentemente do sistema.

Host
É o nome dado a qualquer computador ou usuário conectado a uma rede.

IRQ
Uma IRQ (abreviação para Interrupt Request) é a forma pela qual componentes de hardware requisitam tempo computacional da CPU. Uma IRQ é a sinalização de um pedido de interrupção de hardware.

Kernel
Kernel, que na tradução literal significa **cerne**, é o núcleo do programa, sendo este responsável por todo o gerenciamento dos recursos do sistema, além de possibilitar o ambiente para a execução dos aplicativos - ambientes gráficos; editores de texto, imagem e som etc.

Login
É o nome dado ao ato de acessar (ou pela tradução literal **in** que significa entrar em) um computador, sistema ou rede. Login também é o nome dado ao usuário, sendo correspondido por um conjunto de caracteres específicos determinados pelo adminstrador do sistema. Geralmente é necessário o uso de senha para que seja efetuado.

Logout
É o ato contrário ao login, ou seja, o ato de sair de um computador, sistema ou rede. A palavra **out** em sua tradução literal significa exatamente sair.

Pacote
Pacote ou trama é a estrutura de dados unitária que circula numa rede de computadores. A informação a transmitir poderá ser fragmentada (normalmente quando excedem o tamanho limite do pacote) e os fragmentos podem seguir caminhos diferentes. Este termo pode também significar um grupo ou conjunto de programas que possibilitam a instalação de um sistema operacional, aplicativo etc.

Pacotes FIN
São pacotes que finalizam a ligação de uma seção TCP.

Partição
São subdivisões dentro de um mesmo disco rígido (HD). Ou seja, dentro de um mesmo HD podemos ter várias partições, também chamadas de drivers, que são indicados por letras. Ex.: Dentro do HD-A podemos ter os drivers C, D, E ... etc, respeitando a capacidade do HD, ou seja um HD de 60 Gibabytes pode ser dividido em 3 drivers de 20 Gigabytes, ou 6 de 10 Gigabytes, e assim por diante.

PCI
O barramento PCI (Peripheral Component Interconnect - Interconector de Componentes Periféricos) é um elemento utilizado para conectar periféricos em computadores baseados na arquitetura IBM PC.

Screenshot
Captura da tela do computador transformando-a em um arquivo de imagem. Nome também utilizado para indicar a imagem no plano de fundo de um ambiente gráfico.

Script
Scripts são arquivos contendo linguagem interpretada. São normalmente chamados de arquivos de script ou scripts em vez de executáveis. Os Scripts precisam de softwares interpretadores porque são blocos de códigos não compilados, que são interpretados durante a execução.

Serviços RPC
A Chamada de Procedimento Remoto ou RPC (Remote Procedure Call) é o tipo de protocolo para chamada remota de procedimentos em qualquer lugar da rede ou uma chamada de função para o método de transferência de controle de parte de um processo para outro. Permite a divisão de um software em várias partes, compartilhamento de arquivos e diretórios.

TCP Wrappers
O TCP wrappers também é uma ferramenta de segurança. São pacotes utilizados para controlar o acesso a serviços.

USB
Universal Serial Bus (USB) é um tipo de conexão Plug and Play que permite a conexão de periféricos sem a necessidade de desligar o computador.

Editora Associada à:

Câmara Brasileira do Livro

ABIGRAF

ASSOCIAÇÃO BRASILEIRA DE TECNOLOGIA GRÁFICA

FIESP CIESP

editora viena

Impressão e acabamento:
Tel.: (14) 3372.2155

PRESERVE A NATUREZA

IMPRESSO EM PAPEL RECICLÁVEL