UNIVERSIDADE FEDERAL DE SANTA CATARINA

Breve introdução ao GNU/Linux

– Versão 0 –

GUFSC

Grupo de Usuários de Software Livre da UFSC

Sumário

Agradecimentos

Agradecemos a todos aqueles que direta ou indiretamente colaboraram para a elaboração desta apostila. Agradecemos em especial, aos amigos do *GUFSC* pelo apoio e perseverança na luta pelo Software Livre.

GUFSC, 6 de dezembro de 2003.

Capítulo 1

Introdução

Esta apostila foi organizada pelo GUFSC - Grupo de Usuários de Software Livre da Universidade Federal do Estado de Santa Catarina.

Ela visa dar uma noção aos alunos, professores e pessoas da comunidade universitária sobre Software Livre e fornece informações fundamentais para a operação do Sistema Operacional GNU (particularmente a versão baseada no núcleo Linux) que hoje poderíamos dizer que é o Sistema Operacional "livre" que mais cresce em número de usuários no Brasil e no mundo.

Esperamos, a partir dela, passar um pouco da essência deste sistema operacional e talvez transformar você leitor, em um entusiasta como nós. A maior parte dos comandos e aplicativos de sistema referem-se aos utilitários GNU, muitos deles fazem parte dos pacotes 'binutils' e 'bash' largamente adotados em sistemas GNU/Linux. Entretando noutros sistemas tipo UNIX os comandos e utilitários de sistema costumam ser diferentes, mas a documentação 'on-line' do sistema geralmente é suficiente para um rápido esclarecimento das diferenças.

Agradecemos a colaboração dos seguintes coordenadores:

Adriano Afonso Ferreira <adriano@das.ufsc.br>

Ricardo Grützmacher <grutz@das.ufsc.br>

Capítulo 2

Software Livre

2.1 Definição de Software Livre

"Software Livre" é uma questão de liberdade, não de preço. Para entender o conceito, você deve pensar em "liberdade de expressão", não em "cerveja grátis".

"Software livre" se refere à liberdade dos usuários executarem, copiarem, distribuírem, estudarem, modificarem e aperfeiçoarem o software. Mais precisamente, ele se refere a quatro tipos de liberdade, para os usuários do software:

- A liberdade de executar o programa, para qualquer propósito *(Liberdade n° 0)*.
- A liberdade de estudar como o programa funciona, e adaptá-lo para as suas necessidades *(Liberdade n° 1)*. Aceso ao código-fonte é um pré-requisito para esta liberdade.
- A liberdade de redistribuir cópias de modo que você possa ajudar ao seu próximo *(Liberdade n° 2)*.
- A liberdade de aperfeiçoar o programa, e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie *(Liberdade n° 3)*. Acesso ao código-fonte é um pré-requisito para esta liberdade.

Um programa é software livre se os usuários têm todas essas liberdades. Portanto, você deve ser livre para redistribuir cópias, seja com ou sem modificações, seja de graça ou cobrando uma taxa pela distribuição, para qualquer um em qualquer lugar. Ser livre para fazer essas coisas significa (entre outras coisas) que você não tem que pedir ou pagar pela permissão.

Veja mais sobre a definição de software livre em:
http://www.gnu.org/philosophy/free-sw.pt.html

2.2 "Porque Você Não Se Muda Pra Rússia?"

Nos Estados Unidos, qualquer um que defenda outra coisa que não a mais extrema forma de política egoísta de não intervencionismo tem ouvido freqüentemente esta acusação. Por exemplo, é dito isso aos que defendem um sistema de saúde nacional, tal como os encontrados em todas as outras nações industrializadas do mundo livre. Também dizem isso aos que defendem o suporte público para as artes, também universal em nações desenvolvidas. A idéia que cidadãos tenham qualquer obrigação para com o bem público é identificada nos Estados Unidos como comunismo. Mas quão similar são estas idéias?

Comunismo como praticado na União Soviética era um sistema de controle central onde toda atividade era regimentada, supostamente para o bem comum, mas na prática em pról dos membros do partido Comunista. E onde os equipamentos de cópia eram guardados herméticamente para evitar cópias ilegais.

O sistema americano de propriedade intelectual exerce um controle central sobre a distribuição de um programa, e guarda os equipamentos de cópia com esquemas de proteção contra cópia automáticos para evitar a cópia ilegal.

Em contraste, estamos trabalhando para construir um sistema onde pessoas são livres para decidir suas própria ações; em particular, livres para ajudar seus companheiros, e livres para alterar e melhorar as ferramentas que elas usam no seu cotidiano. Um sistema baseado em cooperação voluntária e descentralização.

Assim, se nós temos que julgar pontos de vista pelas suas semelhanças com o comunismo Russo, são os proprietários de software que são os comunistas.

Extraído de http://www.gnu.org/philosophy/shouldbefree.pt.html

Capítulo 3

História do GNU/Linux

3.1 Origem do GNU

"GNU" é acrônimo recursivo para "GNU's Not Unix". Um nome engraçado escolhido por Richard Stallman para seu projeto de sistema operacional compatível com o UNIX. O projeto foi iniciado em 1983. Desde lá todos os componentes do sistema UNIX foram sendo reescritos como livres para formar o que viria a ser o sistema GNU que conhecemos atualmente.

Em 1985 foi fundada a 'Free Software Foundation' (FSF) para fomentar o desenvolvimento do sistema GNU.

O projeto GNU produziu uma quantidade respeitável de software livre. Todo o software desenvolvido no contexto do projeto GNU recebe o prefixo "GNU" como "GNU EMACS", o legendário editor de texto dos hackers.

Entretanto muitos softwares encontrados nos sistemas GNU atuais não foram feitos pelo projeto GNU. A razão era simplesmente poupar trabalho no casos em que alguém já havia feito algum software livre que poderia ser utilizado no projeto GNU. Alguns exemplos são o X Window System, o Apache, o Perl, etc.

O mascote do sistema GNU é um GNU! Animal típico das savanas africanas. GNU pronuncia-se normalmente como 'GNU' e não 'GI-nú', nem mesmo 'nú' como seria normal pronunciar em inglês.

3.2 Origem do Linux

"Linux" é um núcleo, ou 'kernel', de sistema operacional tipo UNIX criado inicialmente como hobby por um estudante, Linus Torvalds, na Universidade de Helsinki - Finlândia. Linus tinha interesse no Minix, um pequeno sistema UNIX, e decidiu criar um sistema que excedia os padrões do Minix. Ele começou seu trabalho em 1991 quando lançou a versão 0.02 e trabalhou constantemente até 1994 quando a versão 1.0 do núcleo Linux foi lançada.

Linux tem um mascote oficial, o Pinguim do Linux, que foi escolhido por Linus Torvalds para representar a imagem que ele associa ao núcleo de sistema UNIX que ele criou.

Embora existam muitas variações da palavra Linux, ela é mais comumente pronunciada com um "i"curto e com ênfase na primeira sílaba, como em LÍ-nucs.

3.3 GNU/Linux hoje

Da combinação do GNU com o Linux surge o sistema "GNU/Linux", ou melhor, o sistema GNU que usa o Linux como núcleo. Por causa da natureza de funcionalidade e da disponibilidade do sistema GNU/Linux, ele se tornou bastante popular no mundo todo e um grande número de programadores pegaram o código do Linux e adaptaram-no para suas necessidades. Atualmente, existem dúzias de projetos em andamento para transportar o Linux para diversas configurações de hardware e propósitos.

Capítulo 4

Organização do Unix

4.1 Características do Unix

4.1.1 Vantagens:

- Filosofia: simplicidade, generalidade, regularidade, portabilidade;
- Idéias simples e muito boas;
- Coerência dos conceitos básicos;
- Documentação de referência concisa e precisa;
- Facilidade de integração;
- Facilidade de geração de aplicativos.

4.1.2 Desvantagens:

- Dificuldade de aprendizado;
- Pouco amigável (dependendo do desktop);
- Sistema muito grande;
- Existência de várias versões e padrões.

4.2 Kernel Unix

4.2.1 Para que serve?

- Interage diretamente com o hardware;
- Programas utilizam o kernel para todo acesso ao hardware;
- Provê serviços;
- Outros: rede, gerenciamento de arquivos e segurança, serviços de E/S, gerenciamento de processos, gerenciamento de usuários, gerenciamento de memória, interrupções e erros, contabilidade do sistema.

4.3 Nomes de arquivos

Os sistemas tipo UNIX são sensíveis à diferença entre maiúsculas e minúsculas.

Podem existir num mesmo diretório um arquivo "meu_arquivo.txt" e outro "Meu_arquivo.txt" e qualquer outra combinação que não coincidam as mesmas maiúsculas e minúsculas.

Nos sistemas Unix a extensão dos arquivos não é importante pois o sistema tem outras maneiras de identificar o tipo do arquivo. Por exemplo com o comando 'file'.

A extensão nos sistemas UNIX serve apenas para facilidade de identificação do usuário e às vezes para associações nos aplicativos do desktop gráfico.

Programas em geral não possuem extensão quando compilados em linguagem de máquina.

Os programas interpretados como shell scripts, perl, python costuman receber extensões .sh, .pl, py respectivamente.

Outras extensões comuns são:

relatorio.txt - O .txt indica que o conteúdo é um arquivo texto;

script.sh - Arquivo de Script (interpretado por /bin/sh);

system.log - Registro de algum programa no sistema;

arquivo.gz - Arquivo compactado pelo utilitário gzip;

index.html - Página de Internet (formato Hypertexto).

4.4 Permissões de acesso

Modo de acesso	Arquivo comum/especial	Diretório
Leitura	Examinar o conteúdo do arquivo	Listar arquivos dentro do diretório
Escrita	Alterar o conteúdo do arquivo	Criar e remover arquivos
Execução	Executar o arquivo como camando	Acessar diretório e arquivos abaixo dele

Tabela 4.1: Permissões de Acesso

Pode-se especificar a permissão de um arquivo de forma absoluta.

r	w	x	-	u	g	o
4	2	1	-	rwx	rwx	rwx

Tabela 4.2: Tipos de Permissões

4.5 Usuário Real × Usuário Efetivo

1. Cada processo possui usuário/grupo real e usuário/grupo efetivo
2. As permissões de acesso são verificadas considerando-se o usuário/grupo efetivo
3. Para os processos de um usuário, usuário real é equivalente ao seu nome de "login". Isso não muda nunca.
4. Normalmente, o usuário/grupo efetivo é o mesmo que o usuário/grupo real, mas isto pode mudar em alguns processos, dependendo do programa que está sendo executado.

   Se o programa possui set-user-id, o usuário efetivo é o dono do arquivo executável. Caso contrário, o usuário efetivo é o mesmo que o usuário real.

   Se o programa possui set-group-id, o grupo efetivo é o grupo do arquivo executável. Caso contrário, o grupo efetivo é o mesmo que o grupo real.

4.6 Diretório Raiz

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

Nele estão localizados outros diretórios como o /bin, /sbin, /usr, /usr/local, /mnt, /tmp, /var, /home, etc. Estes são chamados de sub-diretórios pois estão dentro do diretório "/". A estrutura de diretórios e sub-diretórios pode ser identificada da seguinte maneira:

/
/bin
/sbin
/usr
/usr/local
/mnt
/tmp
/var
/home

A estrutura de diretórios também é chamada de "Árvore de Diretórios" porque é parecida com uma árvore de cabeça para baixo. Cada diretório do sistema tem seus respectivos arquivos que são armazenados conforme regras definidas pela FHS (FileSystem Hierarchy Standard - Hierarquia Padrão do Sistema de Arquivos) versão 2.0, definindo que tipo de arquivo deve ser armazenado em cada diretório.

4.7 Diretório Padrão

É o diretório em que nos encontramos no momento. Também é chamado de diretório atual. Você pode digitar 'pwd' para verificar qual é seu diretório padrão.

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

4.8 Diretório Home

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

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

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

Dependendo de sua configuração e do número de usuários em seu sistema, o diretório de usuário pode ter a seguinte forma: /home/[1ªletra_do_nome]/[login], neste caso se o seu login for "joao" o seu diretório home será /home/j/joao.

4.9 Camimho de Diretório

São os diretórios que teremos que percorrer até checar no arquivo ou diretório que que procuramos. Se desejar ver o arquivo /usr/doc/copyright/GPL você tem duas opções:

1. Mudar o diretório padrão para /usr/doc/copyright com o comando cd /usr/doc/copyright e usar o comando cat GPL;

2. Usar o comando 'cat' especificando o caminho completo na estrutura de diretórios e o nome de arquivo: cat /usr/doc/copyright/GPL. As duas soluções acima permitem que você veja o arquivo GPL. A diferença entre as duas é a seguinte.

Na *primeira*, você muda o diretório padrão para /usr/doc/copyright (confira digitando 'pwd') e depois o comando 'cat' GPL. Você pode ver os arquivos de /usr/doc/copyright com o comando 'ls'.
"/usr/doc/copyright" é o caminho de diretório que devemos percorrer para chegar até o arquivo GPL.

Na *segunda*, é digitado o caminho completo para o 'cat' localizar o arquivo GPL: cat /usr/doc/copyright/GPL. Neste caso, você continuará no diretório padrão (confira digitando 'pwd'). Digitando ls, os arquivos do diretório atual serão listados.

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

4.10 Estrutura de Diretórios

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

/bin : Contém arquivos programas do sistema que são usados com freqüência pelos usuários.

/boot : Contém arquivos necessários para a inicialização do sistema.

/cdrom : Ponto de montagem da unidade de CD-ROM.

/dev : Contém arquivos usados para acessar dispositivos (periféricos) existentes no computador.

/etc : Arquivos de configuração de seu computador local.

/floppy : Ponto de montagem de unidade de disquetes.

/home : Diretórios contendo os arquivos dos usuários.

/lib : Bibliotecas compartilhadas pelos programas do sistema e módulos do kernel.

/**lost+found** : Local para a gravação de arquivos/diretórios recuperados pelo utilitário 'fsck.ext2'. Cada partição possui seu próprio diretório lost+found.

/**mnt** : Ponto de montagem temporário.

/**proc** : Sistema de arquivos do kernel. Este diretório não existe em seu disco rígido, ele é colocado lá pelo kernel e usado por diversos programas que fazem sua leitura, verificam configurações do sistema ou modificar o funcionamento de dispositivos do sistema através da alteração em seus arquivos.

/**root** : Diretório do superusuário(root).

/**sbin** : Diretório de programas usados pelo superusuário (root) para administração e controle do funcionamento do sistema.

/**tmp** : Diretório para armazenamento de arquivos temporários criados por programas.

/**usr** : Contém maior parte de seus programas. Normalmente acessível somente como leitura.

/**var** : Contém maior parte dos arquivos que são gravados com freqüência pelos programas do sistema, e-mails, spool de impressora, cache, etc.

Capítulo 5

Comandos Básicos

5.1 Linha de Comando

Aprender a operar um sistema tipo Unix a partir da linha de comando é essencial pois além de ser a maneira mais rápida de executar uma grande parte das tarefas, às vezes numa situação problemática talvez a linha de comando seja a única maneira possível de acessar o sistema.

O formato para a maioria dos comandos nos sistemas tipo Unix especialmente em sistemas GNU é
% comando -[opções, argumentos], onde:

- Opções usualmente começam com o caractere "–" seguido de uma letra.
- Algumas opções têm parâmetros. Neste caso o parâmetro vem após a opção.
- Normalmente é possível agrupar opções com um único "–".
- Opções com opções de mais de uma letra usualmente começam com "– –".
- Em geral não é possível agrupar opções de mais de uma letra com apenas um "– –".
- Há freqüêntes exceções por parte de programas que não seguem a padronização por vários motivos.

5.2 Listando Arquivos e Diretórios

Nome: *'ls'*.

Breve Descrição: O programa *'ls'* lista informações sobre arquivos (de qualquer tipo, incluindo diretórios). Opções adicionais podem ser utilizadas arbitrariamente, como usual.

Para listar os arquivos e/ou diretórios em GNU/Linux usa-se o comando *'ls'*.

Sintaxe Resumida:

ls –[opções] <arquivo(s)>

Opções:

a → (– –all) - Listar todos os arquivos, inclusive ./ e ../
A → (– –almost–all) - Listar todos os arquivos, sem listar ./ e ../
t → Lista or arquivos por ordem de atualização (novos > velhos)
h → (– –human–readable) - Para que os tamanhos dos arquivos sejam convertidos para unidades mais fáceis de se compreender.
l → Listagem detalhada. (permissões, dono, dia/hora/mês de edição ...)
X → Lista os arquivos de acordo com a sua extensão em ordem alfabética.
S → Lista os arquivos de acordo com o seu tamanho.
C → Lista os arquivos organizados em colunas.
– –**color** → Lista os arquivos, mudando a cor de acordo com o que contêm e/ou representam.

Dica:

1. Pode-se usar um alias *" alias ls='ls – –color' "* para que, toda vez que se use o comando *'ls'*, o resultado exibido seja colorido.
2. Em algumas distribuições, já existe um alias para o comendo *'ls -l'*, que é o comando *'ll'*.

Exemplo(s):

$ ls -lah Scilab.pdf.gz

–rw– – – –rw– 1 aaf aaf 295K Jan 27 00:28 Scilab.pdf.gz

Nota(s):

→ Para maiores informações sobre o comando *'ls'*: man ls; info ls, ls – –help.

5.3 Metacaracteres

Metacaracteres são caracteres que representam o nome de um grupo de arquivos: . ? ⋆ + ^ $ | [ ] { } ( ) \ . Cada simbolo acima tem sua função específica, que pode mudar dependendo do contexto no qual está inserido, e podemos agregá-los uns com os outros, combinando suas funções e fazendo construções mais complexas.

Além destes, temos outros metacaracteres extendidos que foram criados posteriormente, pois tarefas mais complexas requisitavam funções mais específicas ainda.

Metacaractere	Mnemônico
.	Ponto
[ ]	Lista
[^]	Lista negada
?	Opcional
$\star$	Asterisco
+	Mais
{ }	Chaves
^	Circunflexo
$	Cifrão
\b	Borda
\	Escape
|	Ou
( )	Grupo
\1	Retrovisor

Tabela 5.1: Metacaracteres

5.3.1 Metacaracteres e suas funções

Eles estão divididos em quatro grupos distintos, de acordo com características comuns entre eles.

Representantes

Metacaractere	Mnemômico	Função
.	Ponto	Um caractere qualquer
[...]	Lista	Lista de caracteres permitidos
[^...]	Lista negada	Lista de caracteres proibidos

Tabela 5.2: Representantes

Quantificadores

Metacaractere	Mnemômico	Função
?	Opcional	Zero ou um
$\star$	Asterisco	Zero, um ou mais
+	Mais	Um ou mais
{n,m}	Chaves	De 'n' até 'm'

Tabela 5.3: Quantificadores

Âncoras

Metacaractere	Mnemômico	Função
^	Circunflexo	Início da linha
$	Cifrão	Fim da linha
\b	Borda	Início ou fim de palavra

Tabela 5.4: Âncoras

Outros

Metacaractere	Mnemômico	Função
\c	Escape	Torna literal o caractere c
|	Ou	Ou um ou outro
(...)	Grupo	Delimita um grupo
\1 ... \ 9	Retrovisor	texto casado nos grupos 1...9

Tabela 5.5: Outros

5.4 Copiando Arquivos e Diretórios

Nome: *'cp'*.

Breve Descrição: O programa *'cp'* copia aquivos (ou, opcionalmente, diretórios). A cópia é completamente independente do original. Pode-se também ou copiar um arquivo para outro, ou copiar arbitrariamente vários arquivos para um diretório de destino.

Para copiar arquivos e/ou diretórios, usa-se o comando *'cp'*.

Sintaxe Resumida:

cp –[opções] <arquivo(s)> <destino>

Opções:

f → (– –force) - Copiar os arquivos mesmo que já haja o arquivo de destino.
p → Copiar o arquivo, mantendo suas carcterísticas (data/hora, permissões, dono).
r → (– –recursive) - Copiar diretórios recursivamente.
u → (– –update) - Copiar os arquivos somente se o arquivos de origem forem mais novos, ou não houverem arquivos de destino.
v → (– –verbose) - Mostra a execução da operação.
i → (– –interactive) - Pergunta antes se realmante deseja efetuar a operação.

Dica: Pode-se usar um alias *"alias cp='cp -iv' "* para que, toda vez que se use o comando *'cp'*, haja uma pergunta, questionando realmente se deseja-se realizar a operação, mostrando a execução da mesma na tela.

Exemplo(s):

$ cp –ivp cv.tex ~/tmp/

'cv.tex' → '~/tmp/cv.tex'

Nota(s):

→ Para maiores informações sobre o comando *'cp'*: man cp ; info cp ; cp –help.

5.5 Movendo e Renomeando Arquivos e Diretórios

Nome: *'mv'*.

Breve Descrição: O programa *'mv'* pode mover qualquer tipo de arquivo de um sistema de arquivos para outro. Também pode-se usar o comando *'mv'* para renomear arquivos e/ou diretórios.

Para mover e renomear arquivos e/ou diretórios usa-se o comando *'mv'*.

Sintaxe Resumida:

- Para mover:
  mv –[opções] <arquivo(s)> <destino>
- Para renomear:
  mv –[opções] <arquivo> <novo_arquivo>

Opções:

b → (– –backup) - Faz um backup de cada arquivo que por ventura seria sobreescrito ou removido.
f → (– –force) - Não pergunta ao usuário antes de remover um arquivo de destino.
v → (– –verbose) - Mostra a execução da operação.
u → (– –update) - Mover os arquivos somente se o arquivos de origem forem mais novos, ou não houverem arquivos de destino.
i → (– –interactive) - Pergunta antes se realmante deseja efetuar a operação.
p → (– –parents) - Cria diretórios pai conforme necessário.

Dica:

- Pode-se usar um alias *" alias mv='mv -iv' "* para que, toda vez que se use o comando *'mv'*, haja uma pergunta, questionando realmente se deseja-se

realizar a operação, mostrando a execução da mesma na tela.

- A opção *'p'* é muito útil para criar árvores de diretório inteiras, por exemplo, supondo que você deseja criar o diretório 'dir1', contendo o subdiretório 'dir2' que contém o subdiretório 'dir3'. Sem o uso da opção *'p'* , você criaria 'dir1' (*mkdir dir1*), criaria 'dir2' (*mkdir dir1/dir2*) e finalmente criaria 'dir3' (*mkdir dir1/dir2/dir3*). Com o uso da opção *'p'*, você faria simplesmente "*mkdir –p dir1/dir2/dir3*" o GNU/Linux criaria a árvore 'dir1/dir2' antes de criar o 'dir3'.

Exemplo(s):

→ Movendo arquivos:

$ mv –i termodinamics.pdf ~/utils/

'termodinamics.pdf' → *'/home/aaf/utils/termodinamics.pdf'*

→ Renomeando arquivos:

$ mv dinamics.pdf termodinamics.pdf

'dinamics.pdf' → *'termodinamics.pdf'*

Nota(s):

→ Para maiores informações sobre o comando *'mv'*: man mv ; info mv ; mv – –help.

5.6 Removendo Arquivos e Diretórios

Nome: *'rm'*.

Breve Descrição: O programa *'rm'* remove cada arquivo dado. Por padrão, ele não remove diretórios, entretanto, existe opção pata tal função.

Para remover arquivos e/ou diretórios, usa-se o comando *'rm'*.

Sintaxe Resumida:

rm –[opções] <arquivo(s)>

Opções:

r → (– –recursive) - Remove o conteúdo dos diretórios recursivamente.
f → (– –force) - Não pergunta ao usuário antes de remover um arquivo de destino.

v → (– –verbose) - Mostra a execução da operação.
i → (– –interactive) - Pergunta antes se realmante deseja efetuar a operação.

Dica: Pode-se usar um alias *" alias rm='rm -iv' "* para que, toda vez que se use o comando *'rm'*, haja uma pergunta, questionando realmente se deseja-se realizar a operação, exibindo o resultado da mesma na tela.

Exemplo(s):

$ rm –iv gnu.png

rm: remove regular empty file 'gnu.png'? y
removed 'gnu.png'

→ Para maiores informações sobre o comando *'rm'*: man rm ; info rm ; rm – –help.

→ Os arquivos removidos com o *'rm'* podem ser recuperados, essa operação requer um conhecimento avançado do sistema de arquivos ativo e é bastente perigosa. Portanto, uma idéia bastante interessante é utilizar a dica citada acima.

→ Uma maneira curiosa de remover um arquivo é mandá-lo para /dev/null ou seja *'mv foo.bar /dev/null'*.

Em casos onde temos diretórios vazios, podemos usar também, o seguinte comando:

Nome: *'rmdir'*.

Breve Descrição: O programa *'rmdir'* remove diretório(s) vazio(s).

Para remover especificamente diretórios vazios em GNU/Linux, usa-se o comando *'rmdir'*.

Sintaxe Resumida:

$ rmdir –[opções] <diretório(s)>

Opções:

v → (– –verbose) - Mostra a execução da operação.

Exemplo(s):

$ rmdir –v utils/

rmdir: removing directory, utils/

→ Para maiores informações sobre o comando *'rmdir'*: man rmdir ; info rmdir ; rmdir – –help.

5.7 Visualizando o Conteúdo de Arquivos e Diretórios

5.7.1 O comando *'more'*

Nome: *'more'*.

Breve Descrição: O programa *'more'* é um filtro que permite a visualização do conteúdo de arquivos por páginas.

Para visualizar um arquivo por página em GNU/Linux, usa-se o comando *'more'*.

Sintaxe Resumida:

$ more –[opções] <arquivo>

Opções:

d - Pergunta ao usuário com a mensagem "Pressione espaço para continuar, 'q' para sair".
s - Resumir várias linhas em branco na visualização por uma única.

Dica: Usa-se o comando *'q'* para encerrar a execução do *'more'*, *'/'* para se procurar alguma palavra específica no texto em exibição, e *'h'* para exibir o sumário.

Exemplo(s):

$ more -d abnt.sumario.tex

\documentclass{report}
% \documentclass{abnt}
\usepackage[brazil]{babel}
\usepackage[latin1]{inputenc}

\begin{document}
\autor{Fulano de Tal}
\titulo{Minha tese fenomenal}
\orientador{Ciclano de Tal}
\instituicao{UFSC}
\local{Florianópolis}
– –More– –(45%)[Press space to continue, 'q' to quit.]

→ Para maiores informações sobre o comando *'more'*: man more ; info more ; more – –help.

5.7.2 O comando *'less'*

Nome: *'less'*.

Breve Descrição: O *'less'* é um programa similar ao *'more'*, mas que permite manipular a visualização do arquivo de forma mais flexível.
Para visualizar um arquivo em GNU/Linux, usa-se o comando *'less'*.
Sintaxe Resumida:

$ less –[opções] <arquivo>

Opções:

f → (– –force) - Força arquivos não-regulares serem abertos para visualização.(Um arquivo não-regular pode ser um diretório ou um dispositivo de arquivo especial).
m → (– –long-prompt) - Exibição verbosa como o *'more'*.
N → (– –LINE_NUMBERS) - Exibe a numeração das linhas.
s - Resumir várias linhas em branco na visualização por uma única.

Dica: Usa-se o comando *'q'* para encerrar a execução do *'less'*, *'/'* para se procurar alguma palavra específica no texto em exibição, e *'h'* para exibir o sumário.

Exemplo(s):

$ less –Ns ~/.bashrc

```
1 # .bashrc
2
3 # User specific aliases and functions
4
5 alias rm='rm -iv'
6 alias cp='cp -iv'
7 alias mv='mv -iv'
8 alias su='su -l'
9
10 # Source global definitions
11 if [ -f /etc/bashrc ]; then
12 . /etc/bashrc
13 fi
.bashrc (END)
```

→ Para maiores informações sobre o comando *'less'*: man less ; info less ; less – –help.

5.7.3 O comando *'tail'*

Nome: *'tail'*.

Breve Descrição: O *'tail'* é um programa similar aos anterioores , porém por padrão, imprime na tela somente as últimas 10 linhas do arquivo a ser visualizados.

Para visualizar as últimas linhas de um arquivo em GNU/Linux, usa-se o comando *'tail'*.

Sintaxe Resumida:

$ tail –[opções] <arquivo>

Opções:

n → (– –lines=N) - Exibe a saída com as últimas 'n' linhas.
v → (–verbose) - Saída verbosa.

Exemplo(s):

$ tail -4 ~/.bashrc

10 # Source global definitions
11 if [–f /etc/bashrc]; then
12 . /etc/bashrc
13 fi

→ Para maiores informações sobre o comando *'tail'*: man tail ; info tail ; less – –tail.

Observação: O camando *'cat'*, também pode ser utilizado na visualização do conteúdo de arquivos → ver *'cat'*.

5.8 Criando Ponteiros Simbólicos

5.8.1 O comando *'ln'*

Nome: *'ln'*.

Breve Descrição: O programa *'ln'* cria ponteiros ('links') simbólicos entre arquivos. Por padrão, ele cria "hard links". Entretanto, com a opção '–s', cria-se ponteiros simbólicos ('soft links').

Para criar ponteiros simbólicos em GNU/Linux, usa-se o comando *'ln'*.

Sintaxe Resumida:

ln <alvo> <nome_do_ponteiro(s)>

ln <alvo> <diretório(s)>

Opções:

d → (– –directory) - Cria 'hard links' entre diretórios (somente para super-usuário).
f → (– –force) - Remove os arquivos de destino existentes.
i → (– –interactive) - Pergunta antes de remover destinos.
s → (– –symbolic) - Cria ponteiros simbólicos ao invés de "hard links".
v → (– –verbose) - Mostra a execução da operação.

Dica: Ponteiros simbólicos são indicados por uma '→' após o nome do arquivo/dispositivo que indica para onde o mesmo aponta; por ex. /dev/cdrom → /dev/hdc.

Exemplo(s):

$ ln –siv /dev/hdc /dev/cdrom

create symbolic link '/dev/cdrom' to '/dev/hdc'
... 3 May 10 10:02 /dev/hdc → /dev/cdrom

→ Para maiores informações sobre o comando *'ln'*: man ln ; info ln ; ln –help.

5.9 Armazenando e Compactando Arquivos

Usando o *'gzip'*, *'bzip2'* e o *'tar'*.

Em GNU/Linux, os utilitários mais usados para compactar e armazenar arquivos são o *'gzip'* ou *'bzip2'* e o *'tar'*, sendo os dois primeiros usados para compactar e o último para armazenar, ou empacotar vários arquivos em um só. Se você procurar por softwares para GNU/Linux disponíveis pela Internet, na maioria das vezes eles serão distribuídos nos formatos *'tar'*, *'gzip'*, *'bz2'*, ou, na grande maioria das vezes como uma combinação do *'tar'* com o *'bz2'* ou *'gzip'*. E se você pretende armazenar e/ou compactar os seus arquivos pessoais, é recomendável o uso destes utilitários.

5.9.1 O comando *'bzip2'*

Nome: *'bzip2'*.

Breve Descrição: O *'bzip2'* é um programa de compressão usado para gerar uma cópia compactada de um determinado arquivo, o *'bzip2'* não une vários arquivos em um só. Se você passar uma lista de arquivos para ele, cada arquivo será compactado individualmente.

Sintaxe Resumida:

bzip2 –[Opções] <arquivo(s)>

Após isto o comando criará um arquivo terminado em *.bz2* que poderá ser descompactado através do comando: *'bunzip2'*.

bunzip2 <arquivo.bz2>

Opções:

d → (– –descompress) - Expande arquivos comprimidos (equivalente ao *'bunzip2'*).
z → (– –compress) - Compacta os arquivos comprimidos.
t → (– –test) - Verifica a integridade de um arquivo compactado.
f → (– –force) - Força a sobreescrita dos arquivos de saída.
k → (– –keep) - Mantém os arquivos originais. Normalmente o *'bzip2'* troca o arquivo original pelo destino.
v → (– –verbose) - Modo verboso. Exibe saída detalhada.
1 → (– –fast) - Compressão rápida.
9 → (– –best) - Melhor compressão.

Exemplo(s):

$ bzip2 meu_arquivo.ext

Compacta o arquivo *meu_arquivo.ext*, trocando ele por *meu_arquivo.ext.bz2*.

$ bzip2 –k meu_arquivo.bz2

Cria o *meu_arquivo.ext.bz2*, mas não remove o original.

$ bzip2 –d meu_arquivo.ext.bz2

Decompacta *meu_arquivo.ext.bz2* para *meu_arquivo.ext* se este não existir.

$ bunzip2 meu_arquivo.ext.bz2

Mesmo que o anterior.

→ Para maiores informações sobre o comando *'bzip2'*: man bzip2, info bgzip2, bzip2 – –help.

→ Pelas características de implementação do *'bzip2'* a compressão dos arquivos normalmente é melhor do que utilizando o *'gzip'*.

5.9.2 O comando *'gzip'*

Nome: *'gzip'*.

Breve Descrição: O *'gzip'* (GNU-zip) é um programa de compressão usado para gerar uma cópia compactada de um determinado arquivo, o que o *'gzip'* **não** faz é unir vários arquivos em um único arquivo.

Sintaxe Resumida:

gzip –[Opções] <arquivo(s)>

Após isto o comando criará um arquivo terminado em *'.gz'* que poderá ser descompactado através do comando: *'gunzip'*.

gunzip <arquivo.gz>

Opções:

c → (– –stdout) - Mantém os arquivos originais.
d → (– –descompress ou – –uncompress) - Expande arquivos comprimidos (equivalente ao *'gunzip'*).
l → (– –list) - Lista o conteúdo de arquivos comprimidos.
v → (– –verbose) - Modo verboso. Exibe saída detalhada.
t → (– –test) - Checa a integridade do arquivo compactado.
1 → (– –fast) - Compressão rápida.
9 → (– –best) - Melhor compressão.

Exemplo(s):

$ gzip nome_do_arquivo.ext

Compacta removendo o arquivo original e criando o arquivo *'nome_do_arquivo.ext.gz'*.

$ gzip –c nome_do_arquivo.ext

Compacta mantendo o arquivo original e criando o arquivo *'nome_do_arquivo.ext.gz'*.

$ gzip –9 nome_do_arquivo.ext

Alta compactação removendo o arquivo original e criando o arquivo *'nome_do_arquivo.ext.gz'*.

$ gzip –cv1 arq1.ext arq2.ext

Compactação baixa mantendo o arquivo original e criando os arquivos *'arq1.ext.gz'* e *'arq2.ext.gz'*, exibindo uma saída detalhada.

$ gzip –l nome_do_arquivo.gz

Lista o conteúdo do arquivo.

$ gzip –d nome_do_arquivo.ext.gz

Descomprime o arquivo (o mesmo que *gunzip nome_do_arquivo.ext.gz*).
→ Para maiores informações sobre o comando *'gzip'*: man gzip, info gzip, gzip – –help.

5.9.3 O comando *'tar'*

Nome: *'tar'*.

Breve Descrição: O *'tar'* é um programa capaz de armazenar um ou mais arquivos. Por sua vez, o *'tar'* não é capaz de compactar os arquivos armazenados; para contornar isto, foi adicionado ao *'tar'*, um parâmetro para os dois utilitários (*'tar'* e *'gzip'* ou *'tar'* e *'bzip2'*) se interagirem, assim o *'tar'* pode criar o armazenamento e logo em seguida compactar o arquivo resultante.

Uma outra capacidade do *'tar'* é a de gravar a propriedade e as permissões dos arquivos, além de manter a estrutura completa de diretórios e as ligações diretas e simbólicas.

Sintaxe Resumida:

tar –[Opções] –f <arquivo> –C <diretório> <arquivo(s)>

Opções:

c → (– –create) - Cria um novo arquivo *'.tar'*.
M → (– –multi–volume) - Cria, lista ou extrai um arquivo multivolume. Não funciona com a opção *z*.
p → (– –preserve–permissions) - Preserva as permissões de acesso originais dos arquivos.
r → (– –append) - Acrescenta arquivos a um arquivo *'.tar'*.
t → (– –list) - Lista o conteúdo de um arquivo *'.tar'*.
v → (– –verbose) - Exibe saída detalhada.
w → (– –interactive) - Solicita confirmação antes de cada ação.
x → (– –extract ou – –get) - Extrai arquivos de um arquivo *'.tar'*.
z → Comprime o arquivo tar resultante com o *'gzip'*.
j → (– –bzip) - Comprime o arquivo tar resultante com o *'bzip2'*.
I → (– –bzip) - Em algumas versões do tar o comando *j* é substituído pelo *I*.
f → (– –file) - Especifica o arquivo tar a ser usado.
C → (– –directory) - Especifica o diretório dos arquivos a serem armazenados.

Dica(s): Em alguns parâmetros o '–' (hífem) não é necessário.

Exemplo(s):

$ tar xzvf /meudir/subdir/arq.tar.gz

Extraindo o conteúdo de um arquivo *'tar.gz'*.

$ tar jxvf /meudir/subdir/arq.tar.bz2

Extraindo o conteúdo de um arquivo *'tar.bz2'*.

→ Para criar arquivos *'.tar.gz'* ou *'tar.bz2'* proceda da seguinte forma:

$ tar -cf arquivo.tar pasta/
$ gzip -9 arquivo.tar
arquivo.tar.gz
$ bzip2 -9 arquivo.tar
arquivo.tar.bz2
→ Para maiores informações sobre o comando *'tar'*: man tar, info tar, tar – –help.

5.9.4 O comando *'zip'*

Nome: *'zip'*.

Breve Descrição: O programa *'zip'* também pode ser utilizado para compactar um arquivo com extensão em GNU/Linux, gerando um novo arquivo com extensão *'.zip'*.

Sintaxe Resumida:

zip –[Opções] <arquivo_a_ser_criado> <arquivo_a_ser_compactado>

Após isto o comando criará um arquivo terminado em *'.zip'* que poderá ser descompactado através do comando: *'unzip'*.

unzip <arquivo.zip>

Opções:

v → (– –verbose) - Modo verboso. Exibe saída detalhada.
1 → (– –fast) - Compressão rápida.
9 → (– –best) - Melhor compressão.

→ Para maiores informações sobre o comando *'zip'*: man zip, info zip, zip – –help.

5.10 Redirecionamento E/S

Alguns comandos possuem uma saída muito extensa para ser visualizada em tela, e seria interessante armazená-la em um arquivo. Para isso existe o redirecionamento de entrada e saída padrões.

Os aplicativos GNU/Linux (em modo texto), de um modo geral, possuem 3 "dispositivos" para interação com o usuário, que são a entrada padrão (stdin), a saída padrão (stdout) e saída de erros (stderr), que estão previamente configuradas para: teclado na "stdin" e terminal ativo para a "stderr" e "stdout". Para redefinir a saída ou entrada padrão para, digamos, um arquivo, basta executar o programa colocando o caractere > após o comando, seguido do arquivo de destino. Por exemplo, deseja-se redirecionar a saída do comando 'ls –al' para o arquivo "/temp/res.ls". Bastaria executar no prompt de comando: *ls –la > /temp/res.ls*. Para redirecionar a entrada de dados, o princípio é o mesmo, exceto que o caractere utilizado é o sinal <.

Note que utilizando o > para redirecionar a saída, altera tando a "stdout" quanto a "stderr", se for necessário redirecionar uma delas apenas, basta inserir imediatamente antes do sinal de > (sem espaços) o número 1 (um) para redirecionar "stdout" e 2 (dois) para a "stder". O uso do '2> arquivo.err' é bastante útil para determinar erros de configuração de diversos aplicativos.

5.11 Dutos

Os dutos (ou "pipes") têm um funcionamento bastante semelhante ao redirecionamento de E/S (operadores > e <), porém o duto não redireciona a saída para um arquivo, mas sim para um outro programa.

Digamos que você deseja executar o programa **A** e o programa **B**, mas o programa **B** deverá ser executado com os resultados do programa **A**. Nada impediria você de executar **A** e redirecionar sua saída para um arquivo (/tmp/saida.A) e depois redirecionar a entrada de **B** para esse mesmo arquivo, fazendo os comandos: "A > /tmp/saida.A" e "B < /tmp/saida.A". Neste caso você executaria dois comandos. Se usasse um duto, o próprio interpretador de comandos (shell) se encarregaria de fazer essas duas operações. Em vez de você digitar os dois comandos acima, você faria "A | B" (**A** duto **B**).

O símbolo para duto são duas barrinhas verticais uma em cima da outra, como um dois pontos esticado. Os dutos são muito utilizados para visualizar a listagem de diretórios com muitos arquivos: "ls -la | less".

5.12 Foreground & Background

O uso do Primeiro Plano ("Foreground") e do Segundo Plano ("Background") serve, essencialmente para quando precisamos "destrancar" um terminal onde está-se rodando algum processo.

Ele funciona basicamente desta maneira:

Quando você está rodando um programa e você aperta *CTRL–Z*, nesse momento aparecerá na sua tela:

[1]+ Stopped programa

Onde o número 1 indica a ordem em que ele foi parado, por exemplo, se no mesmo terminal você está usando outro programa e quer parar ele vai aparecer assim:

[2]+ Stopped programa_2

O comando *'fg'* ("Foreground") serve para você voltar ao programa que estava em execução, para você usar ele, basta digitar 'fg num', onde 'num' é o número da ordem que foi dado quando o programa foi parado, ou então somente *'fg'*, caso for somente um processo, ou então se deseja voltar ao último processo parado.

Da mesma maneira que o comando *'fg'* recoloca um processo parado em execução em primeiro plano, os sistemas tipo Unix podem colocar um processo em execução no segundo plano, com o comando *'bg'*. A diferença entre um processo rodando em primeiro ou segundo plano é semelhante a uma situação em que dois alunos solicitam a atenção de um professor, o professor atende o primeiro aluno, deixa ele fazendo algum exercício e em seguida passa a atender o segundo aluno, enquanto o primeiro continua realizando sua tarefa. Algumas vezes é interessante iniciar um processo diretamente no segundo plano, apesar de se poder realizar isso com a seqüência de comandos:

$ comando <return>

<CTRL+Z>

[1]+ Stopped comando

$ bg <retunr>

É muito mais cômodo poder fazer isso com um só comando:

$ comando &

Colocando o caractere '&' (Ampersand) ao final de um comando indica ao interpretador que aquele processo deve ser executado em segundo plano, ou seja, o processo é chamado e colocado em execução e imediatamente depois o interpretador retoma o controle do sistema.

5.13 Comandos em Sequência

Durante o uso de GNU/Linux podem acontencer situações em que é necessário executar uma sequência de comandos, como para compilar e instalar um programa, por exemplo. Para isso, dispomos de algumas "ferramentas", conhecidas como listas de comandos. Em uma lista, cada comando vem seguido de um

delimitador, que podem ser: "**;**", "**&&**", "||", "**&**". Quando os comandos são separados por "**;**", eles serão executados em sequência direta, ou seja, uma linha da forma:

$ cd /usr/src/prg ; make ; make install
O comando acima srá interpretado da seguinte forma:

$ cd /usr/src/prg

$ make

$ make install

Que é a sequência necessária para instalar a maioria dos aplicativos GNU. Neste caso, o comando *make* vai ser executado assim que o comando *cd* terminar sua execução, e assim que *make* for completado, o *make install* será executado.

Algumas vezes, como no caso da compilação de um programa, alguns comandos dependem do estado de saída do outro, ou seja, só devem ser executados se um outro comando tiver sucesso ou falhar, nesses casos podemos utilizar os delimitadores "||" e "**&&**", que representam a lógica *OU* e *E*, respectivamente. Por exemplo, a sequência executada acima pode ser melhor escrita da seguinte forma:

$ cd /usr/src/prg && make && make install

Que será interepretada como:

$ cd /usr/src/prg

Se o comando *cd* não retornar erro,

$ make

Se o *make* não retornar erro,

$ make install

Que é uma forma muito mais segura, ou seja, ele só vai compilar (*make*) o programa se o diretório */usr/src/prg* existir e puder ser acessado, e depois só vai instalar (*make install*) o programa se ele tiver sido compilado.

O delimitador *OU* funciona de maneira semelhante ao *E*, porém o comando só será executado se o anterior der erro, ou seja, poderíamos fazer algo como:

$ cp arq1 arq2 || rm arq1

Que será interpretado como: copie 'arq1' em 'arq2', se não conseguir, apague o 'arq1'.

O delimitador "**&**" (não confundir com o "**&&**") tem um efeito semelhante ao "**;**", exceto que o interpretador não espera o primeiro comando terminar para depois chamar o segundo, ou seja, os comandos são executados simultâneamente.

Em geral, o último delimitador é o caractere de nova linha, ou simplesmente a telca <*Return*>, que indica para o interpretador que a linha de comandos está pronta e já pode ser executada.

Os vários delimitadores podem ser combinados em uma mesma linha de comando, por exemplo:

$ cd /usr/src/linux ; (make && make install || echo "não consegui instalar")

Os operadores **"&&"** e **"||"** tem igual precedência, seguidos por **"&"** e **;** que têm a mesma precedência.

5.14 Espaço Livre/Ocupado no Disco

Para verificarmos o espaço livre em disco e o espaço ocupado por um arquivo qualquer, utilizamos respectivamente o comando *'df'* e o *'du'*.

O comando *'df'* ("disc free"), lista o espaço livre em cada um dos sistemas de arquivos montados.
Uma opção útil é a *'h'* (– –human-readable), que possibilita que a saída seja visualizada em megabytes (Mb).

Já o comando *'du'* ("disc usage"), lista o espaço ocipado pelos arquivos em disco.
Uma opção útil é a *'h'* (– –human-readable), que possibilita que a saída seja visualizada em megabytes (Mb).

Capítulo 6

Comandos Avançados

6.1 Agendar Tarefas

O GNU/Linux possui um utilitário chamado *"CRON"*. Este utilitário serve para agendar tarefas.

Qualquer execução pode ter data e hora para ser feita.

Para agendar uma tarefa siga os seguintes passos:

1. Logue com superusuario.

2. Execute o comando: *crontab -e* . Isso fará com que voce entre em modo de edição do arquivo "crontab"(com o editor "vi").

3. Insira uma linha assim para agendar a tarefa desejada:

   1. MIN-HORA-DIA DO MES-MES-DIA SEMANA-AÇÃO

   00 02 * * * TESTE

4. Grave o "crontab" ao sair e pronto!

OBS: use " * " para "todos" e use "0" a "7" para dia da semana (0 ou 7 = domingo).

6.2 Alterar Largura de Linhas

O comando *"fold"* permite alterar a largura das linhas de um arquivo. Em sistemas GNU/Linux, o comando *fold* faz parte do pacote "textutils".
Esta alteração pode ser feita contando-se os bytes, palavras, ou através de um

valor pré-definido para o comprimento de linha que se desenha.

O comportamento padrão do comando *fold* é quebrar as linhas em 80 colunas.

Um exemplo seria:

$ FOLD –S –W 60 ARTIGO.TXT

Com estas opções as linhas serão quebradas na 60^a coluna, após um espaço em branco (opção –s), impedindo que palavras sejam partidas.

6.3 Comandos do Shell & Atalhos do teclado

6.3.1 Comandos do Shell

- *history* : lista os últimos comandos.
- *cd –* : volta para o último diretório.
- *pushd* : coloca o diretório atual na pilha e vai para o diretório especificado.
- *popd* : volta para o último diretório empilhado.
- *env* : mostra as variáveis de ambiente.
- *stty* : ajusta algumas opções de entrada e saída para o terminal.

6.3.2 Atalhos de teclado

- *CTRL+S* : Congela o fluxo de caracteres no terminal.
- *CTRL+Q* : Continua o fluxo de caracteres no terminal.
- *CTRL+L* : Limpa/redesenha o terminal.
- *CTRL+A* : Desloca o cursor até o início da linha.
- *CTRL+E* : Desloca o cursor até o fim da linha.
- *CTRL+K* : Apaga desde o cursor até o fim da linha.
- *CTRL+U* : Apaga desde o cursor até o início da linha.
- *CTRL+D* : EOF (End Of File), sai de programas ou dá logout no shell.
- *CTRL+Z* : Pausa processo atual.
- *CTRL+C* : Interrompe processo atual.
- *CTRL+H* : Apaga o último caracter digitado.

6.4 Manipulando CD's

Veremos agora como manipular cd's em GNU/Linux usando a linha de comando. Existem muitos programas gráficos que podem fazer as mesmas tarefas. Entretanto, eles oferecem menos opções de controle que os comando abaixo.

6.4.1 Ripando um CD de Audio

NICE −18 CDPARANOIA -X -B -W −FORCE-READ-SPEED 1

Este comando irá ripar o cd de audio e depositar os arquivos ".wav" no diretório atual.

Programa usado: *"cdparanoia"*

6.4.2 Criando um CD de Audio

NICE −18 CDRECORD -V -DUMMY -DAO -USEINFO -PAD -AUDIO *.WAV

Este comando irá escrever todos os arquivos ".wav" no cd vazio, criando um cd de audio padrão.

Programa usado: *"cdrecord"*

6.4.3 Criando uma Imagem de CD

MKISOFS -F -ISO-LEVEL=1 -J -R -T -PAD -V -O $OUTFILE -V "$TITLE"$DIR

Este comando criará um arquivo-ISO $outfile contendo o sistema de arquivos do CD do diretório $dir com o título $title. O sistema de arquivos contido no arquivo ISO, deverá ser legível por qualquer SO.

Programa usado: *"mkisofs"*

6.4.4 Escrevendo um arquivo ISO (Imagem de CD) em um CD

NICE −18 CDRECORD -V -DUMMY -DATA XXX.ISO

Este comando escreverá o arquivo ".iso" especificado no CD, criando um cd de dados.

Programa usado: *"cdrecord"*

6.4.5 Fazendo uma cópia 1-para-1 de CD's de dados

NICE −18 DD IF=/DEV/CDROM OF=/TMP/CD.ISO

Este comando criará uma cópia 100% do cd de dados e o escreverá em um arquivo iso /tmp/cd.iso

NICE −18 CDRECORD -V -DUMMY -DATA /TMP/CD.ISO

Este comando escreverá o arquivo ISO no CD, criando uma cópia perfeita da original.

Programas usados: *"dd"* e *"cdrecord"*

6.4.6 Convertendo MP3's para WAV's para criar CD's de audio

MPG123 -S $SRC | SOX -C2 -S -W -T RAW -R 44100 - -T WAV - > $DEST

Este comando converterá os arquivos ".mp3" $src para arquivos ".wav" $dest.

Programas usados: *"mpg123"* ou *"mpg321"*, *"sox"*

6.4.7 Convertendo WAV's para MP3's

LAME −PRESET EXTREME INFILE.WAV OUTFILE.MP3

Este comando irá converter os aquivos ".wav" infile.wav para arquivos ".mp3" outfile.mp3, usando as opções de melhor qualidade.

Programa usado: *"lame"*.

6.4.8 Notas sobre o programa *"cdrecord"*

- Ao usar pela primeira vez o gravador,teste como se dá sua perfomance com a opção "-dummy". Isso não gravará nada em seu cd, testará a compatibilidade da velocidade escolhida.

- A configuração do comando está em /etc/cdrecord.conf. Faça "man cdrecord" para aprender como ajustá-lo corretamente.

- Ao criar cd's de audio, use somente cd's do tipo CD-R e não CD-RW. CD-RW não pode ser tocado pela maioria dos tocadores de audio disponíveis.

6.4.9 Capas de CD's

Para criar capa para seus CD's, veja em "http://home.wanadoo.nl/jano/disccover.html." Geralemte esse programa reconhece o cd de audio e baixa a lista dos nomes das músicas automaticamente da internet. Ele pode, até mesmo achar e incluir a capa original. RPM's para esse software são providos.

6.5 Construindo Sistemas de Arquivos

Nome: *'mkfs'.*

Breve Descrição: O programa *'mkfs'* é usado para contruir um sistema de arquivos Linux em um dispositivo, usualmente uma partição de disco rígido. Os sistemas de arquivos são ou o nome de dispositivo (p.ex. /dev/hda1, /dev/sdb2) ou o ponto de montagem (p.ex. /, /usr, /home, /mnt) para o sistema de arquivos.

Para construir um sistema de arquivos ("filesystem") em GNU/Linux, usa-se o comando 'mkfs'.

Sintaxe Resumida:

mkfs –[opções] [–t tipo_de_sistema_de_arquivos] <Sistema_de_Arquivos>

Opções:

v - Produz saída verbosa, ou seja, mostra a execução da operação.
t - Especifica o tipo de sistema de arquivos a ser construido. Se não especificado, o tipo de sistema de arquivos padrão "ext2" é usado. Os sistemas de aruivos que são atualmente suportados são: adfs, affs, autofs, coda, coherent, cramfs, devpts, efs, ext, ext2, ext3, hfs, hpfs, iso9660, jfs, minix, msdos, ncpfs, nfs, ntfs, proc, qnx4, reiserfs, romfs, smbfs, sysv, tmpfs, udf, ufs, umsdos, vfat, xenix, xfs, xiafs.
c - Checa o dispositivo por blocos ruíns antes de executar a operação.

Exemplo(s):

 # mkfs –V –t ext3 /dev/hda2

mkfs version 2.11r (Aug 30 2002)
mkfs.ext3 /dev/hda2
mke2fs 1.27 (8–Mar–2002)
(Aqui se processa o andamento da operação...)

Nota(s):

→ A execução do comando *'mkfs'* só é permitida para super-usuário (ver comando *'su'.*

→ Para maiores informações sobre o comando *'mkfs'*: man mkfs ; info mkfs ; mkfs – –help.

6.6 Montando um sistema de Arquivos

Nome: *'mount'*.

Breve Descrição: Todos os arquivos acessíveis em sistemas tipo Unix são arranjados em uma grande árvore, uma hierarquia de arquivos, enraizada em /. Esses arquivos podem ser divididos em diversos dispositivos. O programa 'mount' serve para montar um sistema de arquivos dado em um dispositivo à grande árvore de arquivos. Esse comando também pode ser utilizado para visualizar a tabela de sistema de arquivos ("fstab"), localizada no /etc.

Para montar um sistema de arquivos ou visualizar a tabela do sistema de arquivos ("fstab") em GNU/Linux, usa-se o comando *'mount'*.

Sintaxe Resumida:

- Montar um sistema de arquivos:

  mount –[opções] [–t tipo_de_sistema_de_arquivos] <Sistema_de_Arquivos> <Ponto_de_Montagem>

- Visualizar a tabela de sistema de arquivos:

  mount

Opções:

v - Produz saída verbosa, ou seja, mostra a execução da operação.
a - Monta todos os sistemas de arquivos (do tipo dado) mencionados em "fstab".
n - Monta sem escrever em /etc/mtab. Isso é necessário p.ex. quando /etc é um sistema de arquivos somente-leitura ("read-only").
t - Especifica o tipo de sistema de arquivos a ser construido. Se não especificado, o tipo de sistema de arquivos padrão "ext2" é usado. Os sistemas de aruivos que são atualmente suportados são: adfs, affs, autofs, coda, coherent, cramfs, devpts, efs, ext, ext2, ext3, hfs, hpfs, iso9660, jfs, minix, msdos, ncpfs, nfs, ntfs, proc, qnx4, reiserfs, romfs, smbfs, sysv, tmpfs, udf, ufs, umsdos, vfat, xenix, xfs, xiafs.
r - Monta p sistema de arquivos somente-leitura ("ready-only").
w - Monta o sistema de arquivos ler/escrever ("read/write"). Esse é o padrão.

Exemplo(s):

 # mount –v –t ext3 /dev/hda3 /mnt/hawk2

/dev/hda3 on /mnt/hawk2 type ext3 (rw)

Nota(s):

→ A execução do comando 'mount' só é permitida para super-usuário (ver comando *'su'*).

→ Para maiores informações sobre o comando *'mount'*: man mount ; info mount ; mount – –help.

6.7 Desmontando um sistema de Arquivos

Nome: *'umount'*.

Breve Descrição: O programa *'umount'* serve para desmontar um sistema de arquivos dado em um dispositivo à grande árvore de arquivos.

Para desmontar um sistema de arquivos em GNU/Linux, usa-se o comando *'umount'*.

Sintaxe Resumida:

umount –[opções] [–t tipo_de_sistema_de_arquivos] <Sistema_a_Desmontar>

Opções:

v - Produz saída verbosa, ou seja, mostra a execução da operação.
a - Desmontar todos os sistemas de arquivos mencionados em "mttab, menos o sistema de arquivos "proc".
n - Desmontar sem escrever em /etc/mtab.
r - Em caso de falha na desmontagem, tentar montar novamente como "ready-only".
r - Indica que as ações devem ser tomadas somente com sistemas de arquivos do tipo especificado.

Exemplo(s):

 # umount -v /mnt/hawk2

/dev/hawk2 umounted

Nota(s):

→ A execução do comando 'umount' só é permitida para super-usuário (ver comando *'su'*).

→ Para maiores informações sobre o comando *'umount'*: man umount ; info umount ; umount – –help.

→ Note que um sistema de arquivos não pode ser desmontado quando ela está ocupado, p.ex., quando existem arquivos abertos nele, ou quando algum processo nele localizado está em operação.

6.8 Redimensionando um Sistema de Arquivos

Utilizando *'resize2fs'* para redimensionar um sistema de arquivos "ext2".

Um dos principais motivos para se redimensionar um sistema de arquivos é quando queremos redimensionar uma partição que contém um sistema de arquivos sem perda de dados e sem danos ao mesmo. Antes de redimensionar um sistema de arquivos, deve-se observar o seguinte:

- O sistema de arquivos não pode estar montado. Caso esteja redimensionando o sistema de arquivos 'root' (/), será necessário utilizar um disco de boot ou então conectar o disco rígido a uma outra máquina para que o sistema de arquivos seja redimensionado.

- O sistema de arquivos não deve conter erros.

6.8.1 Expandindo um sistema de arquivos ext2

Primeiramente é necessário rodar o *'e2fsck'* para checar se não existem erros no sistema de arquivos. A checagem é obrigatória mesmo que você tenha certeza de que o sistema de arquivos não possui erros (o *'resize2fs'* irá reclamar caso você não rode o *'e2fsck'* primeiro e não irá rodar). Para checar o sistema de arquivos, digite o seguinte na linha de comando:

$ e2fsck –f /dev/hdXX

Onde 'XX' irá depender de qual disco rígido e partição contêm o sistema de arquivos que está sendo redimensionado.

Após isto, supondo que seu sistema de arquivos ocupa inteiramente a partição na qual ele está localizado (este é o caso geral), é necessário expandir a partição que contém o sistema de arquivos, de forma que ele possa ser expandido. Para isto deve-se utilizar um aplicativo adequado (*'fdisk'*, *'cfdisk'*, etc.) para remover a partição existente e recriá-la de forma que ela inicie no mesmo bloco que iniciava anteriormente, mas com um tamanho maior. É importante que o bloco inicial da nova partição seja o mesmo da partição anterior, caso contrário o *'resize2fs'* pode não encontrar o superbloco ou ainda causar algum dano ao sistema de arquivos.

Depois de expandida a partição, basta digitar na linha de comando:
$ resize2fs –p /dev/hdXX

Onde 'XX' irá depender de qual disco rígido e partição contêm o sistema de arquivos que está sendo redimensionado. Dessa forma, o sistema de arquivos passará a ter o mesmo tamanho da partição.

6.8.2 Reduzindo um sistema de arquivos ext2

Primeiramente é necessário rodar o *'e2fsck'* para checar se não existem erros no sistema de arquivos. A checagem é obrigatória mesmo que você tenha certeza de que o sistema de arquivos não possui erros (o *'resize2fs'* irá reclamar caso você não rode o *'e2fsck'* primeiro e não irá rodar). Para checar o sistema de arquivos, digite o seguinte na linha de comando:

$ e2fsck –f /dev/hdXX

Onde 'XX' irá depender de qual disco rígido e partição contêm o sistema de arquivos que está sendo redimensionado.

Após isto, pode-se reduzir o sistema de arquivos digitando-se o seguinte na linha de comando:

$ resize2fs –p /dev/hdXX <tamanho>

Onde 'XX' irá depender de qual disco rígido e partição contêm o sistema de arquivos que está sendo redimensionado e <tamanho> deve ser substituido pelo número de blocos que o sistema de arquivos deve ocupar (em geral, 1 bloco = 1 KB). O <tamanho> não pode ser menor que o atual número de blocos ocupados no sistema de arquivos (em outras palavras, não tente reduzir um sistema de arquivos para um espaço menor do que o que ele necessita para comportar todos os arquivos!).

Depois de redimensionado o sistema de arquivos, é muito provável que você irá querer reduzir também a partição que contém o sistema de arquivos para que ela passe a ter o mesmo tamanho do novo sistema de arquivos (a menos que você tenha algum motivo especial para não fazer isso, mas lembre-se: a partição terá um espaço significativo que não estará sendo utilizado para o armazenamento de arquivos). Para isto, é preciso utilizar um aplicativo adequado (*'fdisk'*, *'cfdisk'*, etc.) para remover a partição e recriá-la de forma que o bloco inicial seja o mesmo anterior, mas a partição tenha tamanho reduzido. É importante que o bloco inicial seja o mesmo que era anteriormente, pois caso contrário pode não ser possível montar a partição.

Como saber o bloco exato onde termina o sistema de arquivos pode ser complicado, o mais simples é proceder da seguinte maneira: recriar a partição de forma que ela seja um pouco maior que o sistema de arquivos e rodar novamente o *'resize2fs'*, desta vez sem o parâmetro <tamanho>:

$ resize2fs –p /dev/hdXX

Onde 'XX' irá depender de qual disco rígido e partição contêm o sistema de arquivos que está sendo redimensionado. Assim, o sistema de arquivos passará a ter o tamanho exato da partição.

Algumas observações

Apesar de estarmos falando especificamente de sistemas de arquivos ext2, o procedimento funcionará para partições ext3.