

2A. PARTE!

CONHECIMENTOS GERAIS

✓ *Copyright (c) 2002-2008 –Ednei Pacheco de Melo.*

Permission is granted to copy, distribute and/or modify this document under the terms of the *GNU Free Documentation License*, version 1.1 or any later version published by the *Free Software Foundation*; a copy of the license is included in the section entitled “*GNU Free Documentation License*”.

ÍNDICE

VISÃO GERAL.....	8
I. A ESTRUTURA DE ARQUIVOS.....	9
Introdução.....	9
A estrutura.....	9
/bin - Binários essenciais.....	9
/boot - Inicialização do sistema.....	10
/dev - Arquivos de dispositivos.....	11
/etc - Arquivos de configuração.....	12
/home - Dados pessoais.....	13
/lib - Bibliotecas essenciais.....	13
/media e /mnt - Pontos de montagens.....	14
/opt - Compatibilidade entre aplicativos.....	15
/proc - Informações e processos do kernel.....	15
/root - Administrador do sistema.....	16
/sbin - Binários essenciais do sistema.....	17
/tmp - Arquivos temporários.....	17
/srv - Informações de serviços (Internet).....	18
/sys - Suporte ao dispositivos de hardware.....	18
/usr - Recursos dos sistemas Unix.....	19
/var - Variáveis.....	19
Sobre a norma FHS.....	22
Conclusão.....	22
II. OS ARQUIVOS DE DISPOSITIVOS.....	24
Introdução.....	24
A classificação.....	24
Tipo caracter.....	24
Tipo bloco.....	24
A estrutura /dev.....	25
Unidades externas e de armazenamento.....	25
Discos rígidos IDE & CD-ROMs.....	25
Unidades SCSI e SATA.....	26
Disquetes.....	26
Dispositivos de áudio.....	26
Fax-modem (portas seriais).....	27
Console terminal.....	27
Sobre o uDEV.....	28
Conclusão.....	28
III. A LINHA DE COMANDO.....	29
Introdução.....	29
O Bourne Again Shell.....	29

Informações e métodos essenciais.....	30
Complemento da tecla <TAB>.....	30
O uso de expressões.....	31
As cores personalizadas.....	32
Nomenclatura diferenciada.....	32
Acesso à documentação eletrônica.....	33
Equivalências entre o BASH e MS-DOS.....	34
Estrutura de diretórios.....	35
Acesso as unidades do sistema.....	35
Especificações do diretório.....	37
Exibição de caminho.....	37
Case sensitive.....	37
Observações.....	38
Conclusão.....	38
IV. MANIPULAÇÃO DE ARQUIVOS E DIRETÓRIOS.....	39
Introdução.....	39
Operações básicas.....	39
Listagem e navegação.....	39
ls.....	39
cd.....	40
Visualização.....	41
type.....	41
less.....	41
file.....	42
pwd.....	42
more.....	43
du.....	45
Manipulação.....	46
mkdir.....	46
dd.....	46
cp.....	46
mv.....	47
ln.....	47
Editoração.....	48
mcedit.....	48
Exclusão.....	48
rmdir.....	48
rm.....	49
Cópias de segurança e compactação.....	49
Arquivamento.....	49
tar.....	50
Compactação.....	51
gzip / gunzip.....	51
bzip2 / bunzip2.....	52
zip / unzip.....	52
Utilitários.....	53
split.....	53
cat.....	53
Conclusão.....	54

V. UNIDADES, PARTIÇÕES E FORMATOS.....	55
Introdução.....	55
As unidades e as partições.....	55
Os formatos.....	55
Tipos de sistemas de arquivos.....	55
O SWAP.....	56
O ext2 e ext3.....	56
ReiserFS.....	57
MSDOS, FAT32 e NTFS.....	58
ISO9660.....	58
Operações e atividades afins.....	58
Trabalhando com partições e unidades.....	58
mount / umount.....	58
sync.....	60
Formatação e definição do sistema de arquivos.....	60
mkfs.....	60
mkreiserfs.....	61
mkswap / swapon.....	61
Verificando as partições e unidades do sistema.....	61
df.....	61
badblocks.....	62
fsck.....	62
reiserfs.....	63
Realizando a transferência de dados.....	64
dd.....	64
Operações com disquetes.....	64
Utilização.....	64
Formatação.....	64
Operações com os gravadores de CD/DVD.....	65
mkisofs.....	65
cdrecord.....	66
Conclusão.....	69
VI. USUÁRIOS, GRUPOS E PERMISSÕES DE ACESSO.....	70
Introdução.....	70
Considerações básicas.....	70
As contas.....	70
O administrador de sistema.....	70
O usuário comum.....	71
O ID.....	71
Os grupos.....	72
As permissões.....	72
Senha.....	73
Comandos gerais.....	73
Adição de usuários e grupos.....	73
adduser.....	73
groupadd.....	76
Administração de contas.....	76
login / logout / exit.....	76
id.....	76

<i>users / groups</i>	77
<i>passwd</i>	77
<i>finger</i>	78
<i>uptime</i>	79
Eliminando usuários e grupos.....	79
<i>userdel</i>	79
<i>groupdel</i>	79
Obtendo os privilégios de outros usuários.....	80
<i>su</i>	80
Atributos de arquivos e diretórios.....	81
<i>chmod</i>	81
<i>chown</i>	83
<i>chgrp</i>	83
<i>umask</i>	84
<i>Os arquivos de configuração</i>.....	85
<i>/etc/passwd</i>	85
<i>/etc/shadow</i>	85
<i>/etc/groups</i>	86
<i>Conclusão</i>.....	87
VII. O GERENCIAMENTO DE PROCESSOS.....	88
<i>Introdução</i>.....	88
<i>Visão geral</i>.....	88
O que é um processo?.....	88
O identificador PID.....	88
Foreground e background.....	88
<i>Gerenciando os processos</i>.....	89
Visualização.....	89
<i>ps</i>	89
<i>top</i>	90
Segundo plano.....	91
<i>Colocando em segundo plano</i>	91
"Control-zê".....	91
<i>bg</i>	91
<i>jobs</i>	92
<i>fg</i>	92
Exclusão.....	92
<i>kill</i>	92
<i>killall</i>	92
<i>Desligamento do sistema</i>.....	93
<i>halt</i>	93
<i>shutdown</i>	93
<i>Conclusão</i>.....	94
VIII. O SISTEMA DE INICIALIZAÇÃO.....	95
<i>Introdução</i>.....	95
<i>Os métodos de inicialização</i>.....	95
System V.....	95
Estilo BSD.....	95

Os scripts de inicialização.....	96
Os demais scripts.....	96
Sistema & aplicações.....	97
Suporte ao Hardware.....	97
Sistema de impressão.....	97
Redes & Internet.....	98
Configurações locais.....	98
Carregamento de módulos.....	98
Compatibilidade.....	99
A sequência de scripts na inicialização.....	100
Os níveis de execução.....	100
Nível 1 - Manutenção do sistema.....	100
Nível 3 e 4 - Modo multi-usuário.....	101
Nível 0 e 6 - Reinicialização do sistema.....	101
Demais níveis de execução (2 e 5).....	102
Operações e ajustes afins.....	102
Ativar / desativar scripts de inicialização.....	102
Método manual.....	102
Método automatizado.....	102
O arquivo /etc/inittab.....	103
Conclusão.....	105
IX. O GERENCIADOR DE INICIALIZAÇÃO.....	106
Introdução.....	106
O LILO.....	106
/etc/lilo.conf.....	107
Seção global.....	107
Seção de partições.....	109
O liloconfig.....	110
Operações mais freqüentes.....	111
Selecionar o sistema GNU/Linux como padrão.....	111
Mudar a resolução do framebuffer.....	111
Adicionar mais uma entrada no LILO.....	112
Adicionar uma senha extra.....	112
Inicializar o sistema em modo de manutenção.....	113
Criar um disco de recuperação com o lilo.conf.....	113
Problemas mais freqüentes.....	113
Ao invés do sistema inicializar, é exibido.....	113
Remoção das definições do LILO no setor MBR.....	114
Recuperar a senha do superusuário.....	114
Atualizando as alterações desejadas.....	114
Conclusão.....	115
X. GERENCIAMENTO DE PROGRAMAS.....	116
Introdução.....	116
A nomenclatura dos pacotes.....	116
Ferramentas & métodos.....	117
Slackware Package Tools.....	117
Red Hat Package Management.....	118

Compilação do código-fonte.....	119
Outros utilitários.....	119
Conclusão.....	120
XI. VARIÁVEIS DE SISTEMA.....	121
Introdução.....	121
As variáveis.....	121
Path / RootPath.....	121
Home.....	122
OsType.....	122
Shell.....	122
Term.....	123
User.....	123
Comandos relacionados.....	123
echo.....	123
set.....	123
export.....	124
Internacionalização.....	124
Arquivos de configuração.....	125
/etc/profile.....	125
O diretório /etc/profile.d/.....	127
~/bashrc.....	128
Conclusão.....	128

VISÃO GERAL

Chegamos agora, a mais uma nova etapa desta literatura. Trata-se da obtenção de conhecimentos técnicos e do aprendizado das operações básicas e essenciais para a boa manutenção de um sistema *GNU/Linux*.

Como qualquer outro sistema operacional, o *kernel* do *Linux* possui agregado diversas ferramentas, utilitários e aplicações específicas que visam prover aos usuários e administradores um leque de recursos essenciais para a boa administração do sistema como um todo. Atividades simples, básicas e triviais, como copiar arquivos, definir permissões de acesso, acessar unidades, monitorar processos, entre outras (até mesmo mais complexas), são de extrema importância para a garantia de uma boa estabilidade e ótima performance do computador em geral.

Por isto, nesta parte, abordaremos apenas os comandos, parâmetros e funções consideradas indispensáveis aos usuários *desktops*, em especial de nível intermediário, mesmo apesar de constarem algumas instruções para o nível avançado. Existe uma infinidade de ferramentas para as mais variadas finalidades, além de diversos graus de complexidade, onde muitas destas somente serão aplicáveis a tarefas específicas que não se enquadram nas necessidades desta classe de usuários.

A leitura dos próximos capítulos desta parte indispensável para os usuários iniciantes e que pela *1a.* vez estão tendo contato com os sistemas *GNU/Linux*, além de ter grande importância para os usuários de nível médio, pois irão proporcionar todos os conhecimentos necessários para o bom aproveitamento das instruções contidas nas partes seguintes. Já para aqueles usuários mais experimentados, podem tranqüilamente seguir adiante para a próxima parte (ou ainda outras posteriores, dependendo de suas habilidades), ficando esta decisão ao seu critério. Porém deixaremos bem claro que a nossa satisfação será enorme caso os mesmos realizassem uma breve leitura destes capítulos, com o intuito de analisar os pontos negativos e positivos deste material, para que resultem em críticas e sugestões de alta qualidade para a melhoria da obra.

Para aqueles que recentemente vieram de outras distribuições (ou estão pensando com seriedade em adotar o *Slackware*), sugerimos a leitura dos capítulos *A inicialização* e *O gerenciamento de programas*. Neles estarão descritas as particularidades da distribuição-base do livro, além de outras instruções necessárias para a boa manutenção do sistema.

Por mais diferentes que sejam as distribuições, a grande maioria dos comandos descritos nos capítulos anteriores (à salvo aqueles específicos) são aplicáveis na grande maioria dos sistemas *GNU/Linux*. O seu aprendizado, tanto utilizando esta quanto qualquer outra distribuição, será de grande valia para quaisquer intervenções que desejam realiza. Por isso, não se preocupem em pensar que o aprendizado aqui obtido será perdido em caso de mudança de distribuição preferida. &:-D



I. A ESTRUTURA DE ARQUIVOS

INTRODUÇÃO

Como qualquer outro sistema operacional, os sistemas *GNU/Linux* realizam a manipulação de diversos dados e informações, onde para isto é necessária estrutura de arquivos e diretórios bem definida e padronizada. Para cada tipo de arquivo, e de acordo com suas funcionalidades e importância, existe um local específico para seu armazenamento. Além do diretório principal do sistema, existe uma série outros diretórios especificados pela padronização.

Neste capítulo, iremos conhecer a estruturação dos dados e diretórios do sistema de arquivos, como também as suas particularidades e algumas recomendações necessárias para a sua boa manutenção.

A ESTRUTURA...

Os diretórios definidos pela norma *FHS* e que compõem a estrutura do sistema de arquivos no *GNU/Linux* são: */bin*, */boot*, */dev*, */etc*, */home*, */lib*, */media*, */mnt*, */opt*, */proc*, */root*, */sbin*, */sys*, */tmp*, */usr* e */var*.

Segue abaixo a descrição detalhada sobre a estrutura e seus respectivos diretórios, além de algumas dicas, conselhos e recomendações.

/BIN – BINÁRIOS ESSENCIAIS

O diretório */bin* contém todos (ou a maioria) os arquivos binários com os comandos essenciais dos usuários, tais como os programas da linha de comando, entre outros.

```
$ cd /bin
$ ls
Mail@          du*           ln*           readlink*     test*
[*            echo*        loadkeys*    rksh@         touch*
arch*         ed@          login*       rm*           tr*
ash*          egrep@      logname*    rmdir*       true*
awk@          env*         ls*          rpm*          tsort*
base64*       expand*      lsmod@      rzip*         tty*
basename*    expr*       mail@        sed*          umount*
bash*        factor*     md5sum*     seq*          uname*
bunzip2@     false*     mkdir*      setterm*     uncompress@
bzcat@       fgrep@     mkfifo*     sh@           unexpand*
bzip2*       fmt*        mknod*     sha1sum*     uniq*
bzip2recover* fold*       more*      sha224sum*   unlink*
cat*         free*      mount*     sha256sum*   users*
chgrp*       ftp*       mt@         sha384sum*   usleep*
chmod*       gawk@      mt-GNU*    sha512sum*   vdir*
chown*       gawk-3.1.5* mt-st*     shred*       wc*
```



```

chroot*      getopt*      mv*          shuf*        which*
cksum*       getoptprog@ nail@         sleep*       who*
comm*        ginstall@   netstat*    sln*         whoami*
compress@    grep*       nice*       sort*        yes*
cp*          groups*     nisdomainname@ split*       ypdomainname@
cpio*        gunzip*     nl*         stat*        zcat*
csh@         gzexe*     nohup*      stty*       zcmp*
csplit*      gzip*       od*         su*          zdiff*
cut*         head*       paste*      sulogin@    zegrep*
date*        hostid*     pathchk*    sum*         zfgrep*
dd*          hostname*   ping*       sync*        zforce*
df*          id*         ping6*      tac*         zgrep*
dialog*      install*    pinky*      tail*       zless*
dir*         ipmask*     pr*         tar*         zmore*
dircolors*   join*       printenv*   tar-1.13*   znew*
dirname*     kill*       printf*     tar-1.16.1@ zsh*
dmesg*       killall*    ps*         tcsh*       zsh-4.3.2@
dnsdomainname@ ksh*       ptx*        tee*
domainname@  link*       pwd*        telnet*
$ _

```

Os arquivos contidos neste diretório geralmente não são modificados após a instalação, porém quando houverem novas atualizações de pacotes no sistema, estes poderão ser alterados.

/BOOT – INICIALIZAÇÃO DO SISTEMA

O diretório */boot* contém todos os arquivos necessários (estáticos) para a inicialização do sistema (*boot loader*), exceto os arquivos de configuração (*etc*) e o gerenciador de inicialização (*LILO*).

```

$ cd /boot
$ ls -l
total 17103
lrwxrwxrwx 1 root root      37 2007-08-03 10:00 README.initrd ->
/usr/doc/mkinitrd-1.1.2/README.initrd
lrwxrwxrwx 1 root root      27 2007-08-07 23:53 System.map -> System.map-
generic-2.6.21.5
-rw-r--r-- 1 root root 795880 2007-06-19 17:18 System.map-generic-2.6.21.5
-rw-r--r-- 1 root root 813610 2007-06-19 16:53 System.map-generic-
smp-2.6.21.5-smp
-rw-r--r-- 1 root root 1232918 2007-06-19 17:23 System.map-huge-2.6.21.5
-rw-r--r-- 1 root root 1252098 2007-06-19 16:58 System.map-huge-smp-2.6.21.5-
smp
-rw-r--r-- 1 root root      512 2007-08-03 10:17 boot.0300
-rw-r--r-- 1 root root      209 2007-08-03 10:17 boot_message.txt
lrwxrwxrwx 1 root root      23 2007-08-07 23:52 config -> config-
generic-2.6.21.5
-rw-r--r-- 1 root root      72738 2007-06-19 17:18 config-generic-2.6.21.5
-rw-r--r-- 1 root root      72764 2007-06-19 16:53 config-generic-smp-2.6.21.5-
smp
-rw-r--r-- 1 root root      72643 2007-06-19 17:23 config-huge-2.6.21.5
-rw-r--r-- 1 root root      72669 2007-06-19 16:58 config-huge-smp-2.6.21.5-smp
-rw-r--r-- 1 root root      5040 2007-06-10 03:09 diag1.img

```



```

drwxr-xr-x 9 root root      384 2007-08-08 10:46 initrd-tree/
-rw-r--r-- 1 root root    487394 2007-08-08 10:46 initrd.gz
-rw----- 1 root root    42496 2007-08-08 10:48 map
lrwxrwxrwx 1 root root      24 2007-08-07 23:53 vmlinuz -> vmlinuz-
generic-2.6.21.5
-rw-r--r-- 1 root root   1937944 2007-06-19 17:18 vmlinuz-generic-2.6.21.5
-rw-r--r-- 1 root root   2087960 2007-06-19 16:53 vmlinuz-generic-smp-2.6.21.5-
smp
-rw-r--r-- 1 root root   4097784 2007-06-19 17:23 vmlinuz-huge-2.6.21.5
-rw-r--r-- 1 root root   4417112 2007-06-19 16:58 vmlinuz-huge-smp-2.6.21.5-smp
$ _

```

Em distribuições que utilizam o gerenciador *GRUB*, este encontra-se armazenado em um subdiretório dentro deste, chamado */boot/grub*.

Somente em tarefas relacionadas ao processo de compilação e atualização do *kernel*, como também algumas intervenções necessárias na configuração do gerenciador de inicialização (*LILO*), é que este diretório ganha uma certa “*notoriedade*”; fora isto, não teremos maiores preocupações.

/DEV - ARQUIVOS DE DISPOSITIVOS

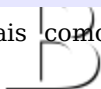
O diretório */dev* contém todos os arquivos de dispositivos necessários para cada dispositivo em que o *kernel* do *Linux* suporta.

```

$ cd /dev
$ ls -l | more
total 0
lrwxrwxrwx 1 root root      10 2007-08-05 08:04 adsp -> sound/adsp
lrwxrwxrwx 1 root root      12 2007-08-05 05:03 appgart -> misc/appgart
lrwxrwxrwx 1 root root      11 2007-08-05 08:04 audio -> sound/audio
drwxr-xr-x 3 root root     60 2007-08-05 05:03 bus/
lrwxrwxrwx 1 root root      3 2007-08-05 05:03 cdr -> hdc
lrwxrwxrwx 1 root root      3 2007-08-05 05:03 cdr1 -> hdc
lrwxrwxrwx 1 root root      3 2007-08-05 05:03 cdrom -> hdd
lrwxrwxrwx 1 root root      3 2007-08-05 05:03 cdrom0 -> hdd
lrwxrwxrwx 1 root root      3 2007-08-05 05:03 cdrom1 -> hdc
lrwxrwxrwx 1 root root      3 2007-08-05 05:03 cdrw -> hdc
lrwxrwxrwx 1 root root      3 2007-08-05 05:03 cdrw1 -> hdc
lrwxrwxrwx 1 root root      3 2007-08-05 05:03 cdwriter -> hdc
lrwxrwxrwx 1 root root      3 2007-08-05 05:03 cdwriter1 -> hdc
crw----- 1 root tty      5, 1 2007-08-05 08:04 console
lrwxrwxrwx 1 root root      11 2007-08-05 05:03 core -> /proc/kcore
crw-rw---- 1 root root    10, 252 2007-08-05 05:03 dac960_gam
drwxr-xr-x 6 root root    120 2007-08-05 05:03 disk/
lrwxrwxrwx 1 root root      9 2007-08-05 08:04 dsp -> sound/dsp
lrwxrwxrwx 1 root root      3 2007-08-05 05:03 dvd -> hdd
lrwxrwxrwx 1 root root      3 2007-08-05 05:03 dvd0 -> hdd
lrwxrwxrwx 1 root root     13 2007-08-05 05:03 fd -> /proc/self/fd/
crw-rw-rw- 1 root root      1, 7 2007-08-05 05:03 full
srwxrwxrwx 1 root root      0 2007-08-05 08:04 gpmctl=
--More--

```

Todo e qualquer dispositivo, tais como portas seriais, discos rígidos,



scanners, mouse, modems, etc., em sistemas baseados em *UNIX* são tratados como arquivos denominados *device node* ou simplesmente *device*. Para ter acesso as funcionalidades de qualquer dispositivo, deveremos recorrer aos seus respectivos arquivos de dispositivos.

Para cada categoria de dispositivo, existe uma certa regra de numeração. Mas, devido a extensa quantidade de *devices*, iremos apresentá-los mais detalhadamente no capítulo seguinte, intitulado *Os arquivos de dispositivos*.

/ETC - ARQUIVOS DE CONFIGURAÇÃO

O diretório */etc* contém todos os arquivos de configuração local para o sistema. Tais arquivos são bem diversificados: a tabela para montagem de partições, as definições do servidor *X.org*, uma série de *scripts*, etc.

```
$ cd /etc
$ ls -l | more
total 1739
-rw-r--r--  1 root root      3458 2007-06-08 22:12 DIR_COLORS
-rw-r--r--  1 root root         21 1999-01-27 23:11 HOSTNAME
drwxr-xr-x 17 root root       592 2007-08-04 13:50 X11/
-rw-r--r--  1 root root      2561 2002-02-24 17:37 a2ps-site.cfg
-rw-r--r--  1 root root     15067 2002-02-24 17:37 a2ps.cfg
drwxr-xr-x  3 root root       104 2004-11-05 06:20 acpi/
-rw-r--r--  1 root root        46 2007-08-04 22:02 adjtime
drwxr-xr-x  3 root root       488 2007-08-03 10:00 asciidoc/
-rw-r--r--  1 root root     9930 2007-08-04 13:23 asound.state
-rw-r----- 1 root daemon    144 2006-08-02 21:55 at.deny
-rw-r--r--  1 root users      95 2007-08-04 14:05 blkid.tab
drwxr-xr-x  3 root root       264 2007-08-03 10:09 bluetooth/
-rw-r--r--  1 root root     1229 2006-06-09 20:35 bootptab
drwxr-xr-x  2 root root       136 2007-08-03 10:10 cron.daily/
drwxr-xr-x  2 root root        72 2007-02-21 19:22 cron.hourly/
drwxr-xr-x  2 root root        48 2002-04-15 23:00 cron.monthly/
drwxr-xr-x  2 root root        48 2002-04-15 23:00 cron.weekly/
-rw-r--r--  1 root root     1799 2007-04-22 16:01 csh.login
drwxr-xr-x  5 root root       328 2006-02-15 21:34 cups/
drwxr-xr-x  3 root root       136 2007-05-19 03:04 dbus-1/
drwxr-xr-x  2 root root       232 2005-07-29 13:17 default/
-rw-r--r--  1 root root        84 2007-06-27 21:56 dhclient.conf
drwxr-xr-x  2 root root        88 2006-07-26 03:09 dhcpc/
--More--
```

No *Windows*, todas as suas definições em termos de configuração ficam armazenadas em seu sistema de registro, que por sua vez é inicializado através do aplicativo *regedit.exe*. Embora funcional e centralizador, infelizmente ele é pouco intuitivo e sem uma documentação eficiente. Já nos sistemas *GNU/Linux*, suas definições ficam registradas em arquivos-textos de configuração, bem mais fácil de ser editado manualmente.

A edição de arquivos de configuração é um aspecto importante na



administração de sistemas *GNU/Linux*¹, pois todos os possíveis parâmetros e variáveis de sistema são armazenados nestes arquivos. Por este motivo, é de extrema importância o conhecimento de suas definições e particularidades.

As definições e particularidades dos arquivos de configuração contidos em */etc* e necessários para o nosso entendimento do funcionamento do sistema, serão apresentados no decorrer desta literatura.

/HOME – DADOS PESSOAIS

Em virtude dos sistemas *Unix-likes* terem sido concebidos para serem sistemas multi-usuários, o diretório */home* é designado exclusivamente para o armazenamento dos arquivos pessoais das contas de usuário do sistema, incluindo personalizações específicas de sua conta.

```
$ cd /home
$ ls -l
total 1
drwx--x--x 19 darkstar users 1248 2007-08-05 08:04 darkstar/
drwxr-xr-x  2 root      root    48 2006-08-06 22:50 ftp/
$ _
```

Para cada conta de usuário criado, é acrescentado neste diretório um subdiretório que utiliza a mesma nomenclatura definida para ser o apelido. Por exemplo, para conta do usuário *darkstar*, teremos um diretório */home/darkstar/* para o armazenamento de todos os arquivos e configurações pessoais desta conta.

/LIB – BIBLIOTECAS ESSENCIAIS

O diretório */lib* contém bibliotecas compartilhadas necessárias para a execução dos arquivos contidos nos diretórios */bin* e */sbin*. Ainda neste diretório são encontrados os módulos do *kernel*.

```
$ cd /lib
$ ls -l | more
total 7746
lrwxrwxrwx 1 root root      12 2007-08-03 10:01 cpp -> /usr/bin/cpp*
-rwxr-xr-x 1 root root    7260 2007-01-25 02:25 e2initrd_helper*
drwxr-xr-x 2 root root      752 2007-04-23 20:00 firmware/
-rwxr-xr-x 1 root root 131484 2007-06-19 17:57 ld-2.5.so*
lrwxrwxrwx 1 root root       9 2007-08-03 10:07 ld-linux.so.2 -> ld-2.5.so*
-rwxr-xr-x 1 root root    7056 2007-06-19 17:57 libBrokenLocale-2.5.so*
lrwxrwxrwx 1 root root      22 2007-08-03 10:07 libBrokenLocale.so.1 ->
libBroke
nLocale-2.5.so*
-rwxr-xr-x 1 root root  16022 2007-06-19 17:57 libSegFault.so*
```

- 1 De acordo com as definições da *FHS*, todos os arquivos de configuração deverão estar armazenados no diretório */etc* - daí a sua importância. Este diretório contém uma estrutura que comporta uma infinidade de arquivos e diretórios, o qual renderia o livro inteiro se todos eles fossem estudados.

```

lrwxrwxrwx 1 root root      15 2007-08-03 09:59 libacl.so.1 ->
libacl.so.1.1.0*
-rwxr-xr-x 1 root root    23512 2006-12-11 22:54 libacl.so.1.1.0*
-rwxr-xr-x 1 root root    13288 2007-06-19 17:57 libanl-2.5.so*
lrwxrwxrwx 1 root root      13 2007-08-03 10:07 libanl.so.1 -> libanl-2.5.so*
lrwxrwxrwx 1 root root      16 2007-08-03 09:59 libattr.so.1 ->
libattr.so.1.1.0
*
-rwxr-xr-x 1 root root    12324 2006-12-11 22:53 libattr.so.1.1.0*
lrwxrwxrwx 1 root root      15 2007-08-03 09:59 libblkid.so.1 ->
libblkid.so.1.0
*
-rwxr-xr-x 1 root root    28712 2007-01-25 02:25 libblkid.so.1.0*
lrwxrwxrwx 1 root root      13 2007-08-03 09:59 libbz2.so.1 -> libbz2.so.1.0*
lrwxrwxrwx 1 root root      15 2007-08-03 09:59 libbz2.so.1.0 ->
libbz2.so.1.0.4
*
-rwxr-xr-x 1 root root    66444 2007-01-24 20:33 libbz2.so.1.0.4*
--More--

```

Estes módulos são armazenados em uma estrutura especial, definida em */lib/modules*. Já as bibliotecas necessárias para as aplicações hospedadas em */usr* não pertencem ao */lib*.

/MEDIA E /MNT – PONTOS DE MONTAGENS

Os diretórios */media* e */mnt* foram definidos para serem utilizados exclusivamente para a montagem de unidades. A diferença entre os dois está justamente no tipo de unidade a ser desmontada.

```

$ cd /mnt
$ ls -l
total 8
-rw-r--r--  1 root    root    376 2006-09-26 00:09 README
drwxr-xr-x  2 root    root     48 2006-09-25 22:02 cdrecorder/
drwxr-xr-x  2 root    root     48 2002-03-16 04:34 cdrom/
drwxr-xr-x  2 root    root     48 2006-09-25 22:02 dvd/
drwxr-xr-x  2 root    root     48 2007-08-04 13:28 flash/
drwxr-xr-x  2 root    root     48 2002-03-16 04:34 floppy/
drwxr-xr-x  2 root    root     48 2002-03-16 04:34 hd/
drwxr-xr-x  2 root    root     48 2006-09-25 22:02 memory/
drwx----- 10 darkstar users  224 2007-08-04 15:08 pkg/
drwxr-xr-x  2 root    root     48 2006-09-25 22:03 tmp/
dr-x-----  1 root    root   4096 2007-08-05 00:59 win/
drwxr-xr-x  2 root    root     48 2006-09-25 22:02 zip/
$ _

```

O */media* deverá ser utilizado exclusivamente para a montagem de mídias removíveis, como *DVDs*, disquetes e memórias eletrônicas em geral.² Já no */mnt* se concentrará a montagem de volumes de uso provisório, como uma partição de um *HD*, por exemplo.

2 Em algumas distribuições, a montagem de determinadas unidades - disquetes e *CD/DVD-ROMs* - são feitas diretamente num diretório situados na raiz - */floppy* e */cdrom*, respectivamente.

/OPT – COMPATIBILIDADE ENTRE APLICATIVOS

O diretório */opt*, apesar de não ser mais especificado na norma *FHS*, foi mantido em virtude da necessidade de manter a compatibilidade com antigos programas que ainda são muito utilizados atualmente.

```
$ cd /opt
$ ls -l
total 0
drwxr-xr-x 8 darkstar users 352 2007-06-17 11:40 openoffice.org.2.2/
$ _
```

Em uma consulta que realizamos na página eletrônica da *Slackware LinuxBR*, obtivemos outras informações muito interessantes:

“Pacotes de software opcional. A idéia atrás do /opt é que cada pacote de software seja instalado para /opt/<software package>, o que facilita para uma desinstalação subsequente. Slackware distribui algumas coisas no /opt (como o KDE em /opt/kde), mas você é livre para colocar o que quiser no /opt.” -> [“Estrutura de diretórios do Slackware LinuxBR”, por r_linux & mistif].

Embora seja citada a manutenção do *KDE* em */opt/kde*, atualmente este ambiente gráfico é mantido na hierarquia de */usr*. Já a suíte *OpenOffice.org* ainda é mantida em */opt*, por se tratar de um *software* instalado à parte.

Num futuro não muito distante, o */opt* será “removido” em definitivo.

/PROC – INFORMAÇÕES E PROCESSOS DO KERNEL

O diretório */proc* contém um sistema de arquivo virtual, com informações gerais do sistema e processo do *kernel*.

```
$ cd /proc
$ ls
1/      212/   2721/  2837/  3/      cmdline  irq/      partitions
1000/   213/   2726/  2839/  3280/   config.gz kallsyms  scsi/
1070/   214/   2728/  2841/  3281/   cpuinfo  kcore     self@
1348/   215/   2761/  2846/  3319/   crypto   key-users slabinfo
173/    216/   2763/  2848/  4/      devices  keys      stat
174/    2229/  2764/  2850/  5/      diskstats kmsg      swaps
175/    2311/  2768/  2852/  6/      dma       loadavg   sys/
178/    2315/  2769/  2856/  83/     driver/   locks     sysrq-trigger
180/    2680/  2774/  2857/  84/     execdomains mdstat    sysvipc/
192/    2688/  2819/  2858/  890/    fb        megaraid/ timer_list
2/      2696/  2820/  2861/  939/    filesystems meminfo   tty/
206/    2701/  2823/  2877/  944/    fs/       misc      uptime
207/    2702/  2825/  2906/  996/    i2o/      modules   version
208/    2708/  2827/  2933/  acpi/   ide/      mounts@   vmstat
209/    2709/  2829/  2944/  asound/ interrupts mpt/      zoneinfo
210/    2712/  2834/  2949/  buddyinfo iomem     mtrr
211/    2719/  2836/  2960/  bus/    ioports   net/
$ _
```



Na verdade, o seu conteúdo não faz parte da estrutura de arquivos do sistema; conforme já dito, ele é apenas um sistema de arquivo virtual para que os administradores do sistema tenham acesso as informações do processamento do *kernel* em forma de arquivos para consulta:

```
$ cat /proc/cpuinfo
processor       : 0
vendor_id     : AuthenticAMD
cpu family    : 6
model        : 8
model name    : AMD Athlon(tm) XP 2000+
stepping     : 1
cpu MHz      : 1668.877
cache size   : 256 KB
fdiv_bug     : no
hlt_bug     : no
f00f_bug    : no
coma_bug    : no
fpu         : yes
fpu_exception : yes
cpuid level  : 1
wp          : yes
flags       : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
cmov pat pse36 mmx fxsr sse syscall mmxext 3dnowext 3dnow ts
bogomips   : 3341.26
clflush size : 32

$ _
```

Estas e muitas outras informações poderão ser obtidas diretamente através da leitura destes “*arquivos-textos*”. Para maiores informações, consultem a *3a. Parte: A Instalação -> Após a instalação...*

/ROOT - ADMINISTRADOR DO SISTEMA

O diretório */root* é definido para ser utilizado exclusivamente no armazenamento de dados e arquivos pessoais do superusuário - o *root*.

```
# cd /root
# ls -l
total 100
-rw-r--r-- 1 root root 1808 2002-04-17 01:21 loadlin16c.txt
-rw-r--r-- 1 root root 97874 2002-04-17 01:20 loadlin16c.zip
# _
```

Ele é mantido na raiz principal e não situado em */home*, em decorrência de uma possibilidade de pane geral do sistema, caso este esteja separado em uma partição. Ao iniciarmos sistema como superusuário para realizar alguma tarefa de manutenção específica, ficaríamos presos a necessidade de ter seus arquivos pessoais disponíveis, e como provavelmente esta partição não se encontrará (pelo fato de utilizar o nível de manutenção), teremos complicações para realizarmos a autenticação do superusuário.

Não é recomendado o uso deste diretório para qualquer finalidades que

B

não seja para a administração e/ou manutenção do sistema, em especial atividades comuns para os usuários tais como leitura do correio eletrônico, armazenamento de dados diversos, etc. Para estas atividades, o administrador deverá ter ou criar para si uma conta de usuário comum.

/sbin - BINÁRIOS ESSENCIAIS DO SISTEMA

O diretório */sbin* somente armazena arquivos binários essenciais para a administração do sistema, onde os mesmos são utilizado somente pelo superusuário ou durante a inicialização do sistema.

```
$ cd /sbin
$ ls -l | more
total 12337
-rwxr-xr-x 1 root bin      4920 2003-02-22 21:47 accton*
-rwxr-xr-x 1 root root    33252 2007-06-24 04:33 adjtimex*
-rwxr-xr-x 1 root root    14476 2007-06-24 04:33 agetty*
-rwxr-xr-x 1 root root    38284 2007-04-30 01:34 arp*
-rwxr-xr-x 1 root root    26600 2006-08-14 22:14 arpd*
-rwxr-xr-x 1 root root    10628 2007-05-09 14:59 arping*
-rwxr-xr-x 1 root root    17604 2007-01-25 02:25 badblocks*
-rwxr-xr-x 1 root root     7744 2007-01-25 02:25 blkid*
-rwxr-xr-x 1 root root     8936 2007-06-24 04:33 blockdev*
-rwxr-xr-x 1 root root    10888 2007-05-14 23:46 bootlogd*
-rwxr-xr-x 1 root root    21120 2007-04-30 01:34 brctl*
lrwxrwxrwx 1 root root      7 2007-08-03 10:00 clock -> hwclock*
-rwxr-xr-x 1 root root     3172 2007-05-14 23:46 consoletype*
-rwxr-xr-x 1 root root    53348 2006-06-14 03:10 convertquota*
-rwxr-xr-x 1 root root   600876 2007-06-20 21:00 cryptsetup.static*
lrwxrwxrwx 1 root root      6 2007-08-03 10:10 ctstat -> lstat*
-rwxr-xr-x 1 root root    65772 2007-01-25 02:25 debugfs*
-rwxr-xr-x 1 root root   199600 2007-05-31 21:28 debugreiserfs*
-rwxr-xr-x 1 root root    98624 2007-02-21 19:22 depmod*
-rwxr-xr-x 1 root root   353276 2007-06-27 21:56 dhclient*
-rwx----- 1 root root     6285 2007-06-27 21:56 dhclient-script*
-rwxr-xr-x 1 root root    43072 2006-07-26 03:09 dhcpcd*
-r-xr-xr-x 1 root root    33828 2007-06-02 18:36 dmsetup*
--More--
```

Todos os executáveis necessários para diversas outras atividades pertinentes estarão disponíveis, como as operações com pacotes, módulos, processos, configurações, partições, etc.

/tmp - ARQUIVOS TEMPORÁRIOS

O diretório */tmp* armazena arquivos temporários gerados pelo sistema. Todos os usuários têm permissão de leitura e escrita nele.

```
$ cd /tmp
$ ls -l
total 1
srwxr-xr-x 1 darkstar users  0 2007-08-05 08:05
OSL_PIPE_1000_SingleOfficeIPC_8
```



```
9e9417147ffe9a2cc78461f15f03871=
```

```
drwx----- 2 darkstar users 120 2007-08-05 08:05 kde-darkstar/  
drwx----- 3 darkstar users 536 2007-08-05 09:58 ksocket-darkstar/  
drwx----- 2 darkstar users 48 2007-08-04 21:36 mc-darkstar/  
drwx----- 2 root root 48 2007-08-04 13:21 mc-root/  
drwxr-xr-x 2 darkstar users 80 2007-08-05 10:02 svbae.tmp/  
$ _
```

Geralmente este diretório é limpo a cada inicialização ou a intervalos relativamente freqüentes. Por este motivo, deveremos evitar a guarda de arquivos por um determinado tempo neste diretório, mesmo aqueles inúteis.

/SRV - INFORMAÇÕES DE SERVIÇOS (INTERNET)

Levemente diferenciado de */opt*, o */srv* armazena dados de aplicações (serviços) direcionados para redes, como o servidor *Web Apache*.

```
$ cd /srv  
$ ls -l  
total 0  
lrwxrwxrwx 1 root root 8 2007-08-03 10:10 httpd -> /var/www/  
lrwxrwxrwx 1 root root 8 2007-08-03 10:10 www -> /var/www/  
$ _
```

A diferença é notória: o */opt* armazena os dados dos programas (binários e componentes), ao passo que o */srv* apenas os dados gerados.

/SYS - SUPORTE AO DISPOSITIVOS DE HARDWARE

O diretório */sys*, tal como o */proc*, é um sistema virtual de arquivos que tem como objetivo, mostrar as informações relacionadas aos *hardware*.

```
$ cd /sys  
$ ls -l  
total 0  
drwxr-xr-x 30 root root 0 2007-08-05 05:03 block/  
drwxr-xr-x 18 root root 0 2007-08-05 05:03 bus/  
drwxr-xr-x 46 root root 0 2007-08-05 08:05 class/  
drwxr-xr-x 10 root root 0 2007-08-05 05:03 devices/  
drwxr-xr-x 3 root root 0 2007-08-05 05:03 firmware/  
drwxr-xr-x 3 root root 0 2007-08-05 05:03 fs/  
drwxr-xr-x 4 root root 0 2007-08-05 05:03 kernel/  
drwxr-xr-x 131 root root 0 2007-08-05 08:05 module/  
drwxr-xr-x 3 root root 0 2007-08-05 05:03 o2cb/  
drwxr-xr-x 2 root root 0 2007-08-05 08:04 power/  
$ _
```

Enquanto que */proc* traz referências mais ligadas ao sistema, o */sys* trata mais especificamente dos dispositivos de hardware em geral.



/usr - RECURSOS DOS SISTEMAS UNIX

O diretório */usr* - abreviação de *Unix Resource System* (recursos dos sistemas *Unix*) - é a segunda maior hierarquia de diretórios do sistema. Todos os aplicativos e utilitários do sistema encontram-se armazenados aqui: ele é como uma espécie de *Arquivos de Programas* do *Windows*.

```
$ cd /usr
$ ls -l
total 280
lrwxrwxrwx 1 root root      5 2007-08-03 10:13 X11 -> X11R6/
drwxr-xr-x 2 root root    192 2007-08-03 10:13 X11R6/
drwxr-xr-x 2 root root    192 2007-08-03 09:59 X11R6.bak/
lrwxrwxrwx 1 root root      8 2007-08-03 09:59 adm -> /var/adm/
drwxr-xr-x 2 root root   89544 2007-08-04 16:03 bin/
drwxr-xr-x 2 root root     48 1993-11-26 01:40 dict/
drwxr-xr-x 536 root root  17640 2007-08-04 15:12 doc/
drwxr-xr-x 2 root root    1112 2006-09-08 21:51 games/
drwxr-xr-x 4 root root     96 2007-06-24 03:57 i486-slackware-linux/
drwxr-xr-x 234 root root  45312 2007-05-10 18:52 include/
drwxr-xr-x 2 root root   11736 2007-02-21 19:58 info/
drwxr-xr-x 100 root root 106280 2007-08-04 15:12 lib/
drwxr-xr-x 13 root root    2680 2006-11-08 18:10 libexec/
drwxr-xr-x 11 root root     264 2007-08-04 15:04 local/
drwxr-xr-x 45 root root    1648 2006-11-08 18:10 man/
drwxr-xr-x 2 root root   9120 2007-06-05 20:28 sbin/
drwxr-xr-x 149 root root   4040 2007-08-04 15:12 share/
lrwxrwxrwx 1 root root     10 2007-08-03 09:59 spool -> /var/spool/
drwxr-xr-x 4 root root    128 2007-08-03 10:04 src/
lrwxrwxrwx 1 root root      8 2007-08-03 09:59 tmp -> /var/tmp/
$ _
```

Embora nas antigas edições tenhamos descrito a função de cada diretório, faremos apenas alguns comentários sobre os principais subdiretórios. Por exemplo, o */usr/X11* define a localização do servidor gráfico *X.org*; */usr/doc* é responsável pela documentação geral; o */usr/include* é a referência padrão dos compiladores *C/C++* para a busca os cabeçalhos (*headers*) no ato da compilação de programas; o */usr/local* - como o próprio nome indica - é uma localização padrão para *softwares* que são instalados localmente (não fazem parte da distribuição); e o */usr/src* armazena o código-fonte das diversas aplicações, inclusive o próprio *kernel-source*!

/var - VARIÁVEIS

O diretório */var* contém informações variáveis, como arquivos e diretórios em fila de execução, arquivos temporários transitórios, etc.

```
$ cd /var
$ ls -l
total 2
lrwxrwxrwx 1 root root      5 2007-08-03 09:59 X11 -> X11R6/
lrwxrwxrwx 1 root root   23 2007-08-03 09:59 X11R6 -> ../../usr/X11R6/lib/X11/
```



```

lrwxrwxrwx 1 root root 3 2007-08-03 09:59 adm -> log/
drwxr-xr-x 8 root root 200 2006-11-08 18:10 cache/
drwxr-xr-x 3 root root 72 2007-06-19 17:27 db/
drwxr-xr-x 2 root root 48 2007-04-03 14:36 empty/
drwxr-xr-x 21 root root 520 2006-11-08 18:10 lib/
drwxrwxrwt 3 root root 104 2007-08-05 08:05 lock/
drwxr-xr-x 14 root root 752 2007-08-05 08:04 log/
lrwxrwxrwx 1 root root 10 2007-08-03 09:59 mail -> spool/mail/
drwxr-xr-x 12 root root 288 1993-11-25 00:29 man/
drwxr-xr-x 3 root root 80 2007-06-08 02:42 named/
drwxr-xr-x 15 root root 664 2007-08-05 08:05 run/
lrwxrwxrwx 1 root root 15 2007-08-03 09:59 rwho -> /var/spool/rwho/
drwxr-xr-x 15 root root 376 2003-06-05 18:43 spool/
drwxr-xr-x 5 root root 128 2007-04-29 19:29 state/
drwxrwxrwt 4 root root 152 2007-08-04 21:44 tmp/
drwxr-xr-x 6 root root 144 2007-07-01 20:12 www/
drwxr-xr-x 2 root root 136 2007-08-03 10:10 yp/
$ _

```

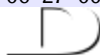
Raramente iremos realizar intervenções manuais nestas estruturas; porém, será interessante que venhamos a conhecer alguns destes diretórios, com o objetivo de coletar informações sobre o funcionamento do sistema. Em determinadas circunstâncias, tais informações podem ser muito úteis para o planejamento e sucesso de certas intervenções, como o */var/log*.

O */var/log* armazena todas as informações geradas pelo sistema através de arquivos-textos chamados *logs*. Sua estrutura compõe-se tanto de arquivos quando de diretórios, que por sua vez, armazenam outros arquivos...

```

$ cd /var/log
$ ls -l
total 1397
-rw-r--r-- 1 root root 37922 2007-08-08 18:24 Xorg.0.log
-rw-r--r-- 1 root root 38000 2007-08-08 16:56 Xorg.0.log.old
-rw-r----- 1 root root 8047 2007-08-08 18:00 acpid
-rw----- 1 root root 0 2007-08-03 10:00 bttmp
-rw-r----- 1 root root 0 2002-04-06 20:13 cron
drwxr-xr-x 2 root root 48 2007-05-09 18:28 cups
-rw-r----- 1 root root 48260 2007-08-08 18:00 debug
-rw-r--r-- 1 root root 15445 2007-08-08 18:00 dmesg
-rw-r----- 1 root root 24024 2007-08-04 13:50 faillog
drwxr-xr-x 2 root root 48 2007-08-03 10:10 httpd
drwxr-xr-x 2 root root 48 2006-12-27 20:27 iptraf
-rw-r--r-- 1 root root 33598 2007-08-08 18:24 kdm.log
-rw-r--r-- 1 root root 292292 2007-08-08 18:00 lastlog
-rw-r----- 1 root root 0 2002-04-06 20:13 maillog
-rw-r----- 1 root root 513368 2007-08-08 18:20 messages
drwxr-xr-x 2 root root 48 2001-05-15 22:47 nfsd
drwxr-xr-x 2 root root 34416 2007-08-07 19:31 packages
drwxr-xr-x 2 root root 224 2007-08-07 08:56 removed_packages
drwxr-xr-x 2 root root 160 2007-08-07 08:56 removed_scripts
drwxr-xr-x 2 root root 48 2006-08-15 00:15 sa
drwxr-xr-x 2 root root 48 2007-06-27 00:36 samba

```



```

drwxr-xr-x 2 root root 18848 2007-08-07 19:31 scripts
-rw-r----- 1 root root 6563 2007-08-08 18:18 secure
drwxr-xr-x 3 root root 664 2007-05-28 18:04 setup
-rw-r----- 1 root root 0 2002-03-09 01:29 spooler
-rw-r----- 1 root root 343285 2007-08-08 18:00 syslog
drwxr-xr-x 2 uucp root 192 2007-08-03 10:10 uucp
-rw-rw-r-- 1 root utmp 297216 2007-08-08 18:18 wtmp
$ _

```

Dentre os arquivos principais, poderemos destacar o *Xorg.0.log*, *debug*, *dmesg*, *messages* e *syslog*, pois permitem analisar todas as interações do sistema com o *hardware*, registrando além das informações gerais, inconsistências, falhas, erros e anomalias das mais variadas espécies.

```

$ cat dmesg
...
tuner 1-0061: type set to 5 (Philips PAL_BG (FI1216 and compatibles))
tuner 1-0061: type set to 5 (Philips PAL_BG (FI1216 and compatibles))
tuner 1-0063: chip found @ 0xc6 (bt878 #0 [sw])
bttv0: registered device video0
bttv0: registered device vbi0
bttv0: registered device radio0
bttv0: PLL: 28636363 => 35468950 .. ok
input: bttv IR (card=72) as /class/input/input2
Adding 257000k swap on /dev/hda5. Priority:-1 extents:1 across:257000k
input: PC Speaker as /class/input/input3
input: ImPS/2 Generic Wheel Mouse as /class/input/input4
parport: PnPBIOS parport detected.
parport0: PC-style at 0x378, irq 7 [PCSPP,TRISTATE]
lp0: using parport0 (interrupt-driven).
lp0: console ready
Capability LSM initialized
ReiserFS: hda7: found reiserfs format "3.6" with standard journal
ReiserFS: hda7: using ordered data mode
ReiserFS: hda7: journal params: device hda7, size 8192, journal first block
18, max trans len 1024, max batch 900, max commit age 30, max trans age 30
ReiserFS: hda7: checking transaction log (hda7)
ReiserFS: hda7: Using r5 hash to sort names
NTFS driver 2.1.28 [Flags: R/W MODULE].
NTFS volume version 3.1.
$ _

```

Já dentre os diretórios principais, entram em destaque o *packages* e o *removed_packages*, responsáveis por armazenarem dados dos pacotes que se encontram instalados e/ou removidos posteriormente. O mesmo se dá com as os diretórios *script* e *removed_script*, que por sua vez armazenam os *scripts* inicializados após a instalação dos pacotes.

```

$ cd /var/log/packages
$ ls -l
total 13450
-rw-r--r-- 1 root root 12007 2007-08-03 10:00 a2ps-4.13b-i386-2
-rw-r--r-- 1 root root 2151 2007-08-03 09:59 aaa_base-12.0.0-noarch-1
-rw-r--r-- 1 root root 2280 2007-08-03 09:59 aaa_elflibs-12.0.0-i486-3
-rw-r--r-- 1 root root 10845 2007-08-03 09:59 aaa_terminfo-5.6-noarch-1
-rw-r--r-- 1 root root 3919 2007-08-03 10:06 aalib-1.4rc5-i486-2

```

```

-rw-r--r-- 1 root root 1147 2007-08-03 10:00 acct-6.3.2-i386-1
-rw-r--r-- 1 root root 2734 2007-08-03 09:59 acl-2.2.39_1-i486-2
-rw-r--r-- 1 root root 1399 2007-08-03 09:59 acpid-1.0.4-i486-2
-rw-r--r-- 1 root root 5113 2007-08-03 10:06 alsa-lib-1.0.14a-i486-1
-rw-r--r-- 1 root root 1094 2007-08-03 10:06 alsa-oss-1.0.14-i486-1
-rw-r--r-- 1 root root 2645 2007-08-03 10:00 alsa-utils-1.0.14-i486-1
-rw-r--r-- 1 root root 64090 2007-08-03 10:04 amarok-1.4.6-i486-1
-rw-r--r-- 1 root root 829 2007-08-03 10:00 amp-0.7.6-i386-1
-rw-r--r-- 1 root root 1509 2007-08-03 09:59 apmd-3.2.2-i486-1
-rw-r--r-- 1 root root 586 2007-08-03 10:12 appres-1.0.1-i486-1
-rw-r--r-- 1 root root 2223 2007-08-03 10:06 apr-1.2.8-i486-1
-rw-r--r-- 1 root root 1961 2007-08-03 10:06 apr-util-1.2.8-i486-1
-rw-r--r-- 1 root root 5637 2007-08-03 10:06 arts-1.5.7-i486-1
-rw-r--r-- 1 root root 893 2007-08-03 10:00 ash-0.4.0-i386-1
-rw-r--r-- 1 root root 6089 2007-08-03 10:06 aspell-0.60.5-i486-2
-rw-r--r-- 1 root root 2626 2007-08-03 10:06 aspell-en-6.0_0-noarch-4
-rw-r--r-- 1 root root 1045 2007-08-03 10:00 at-3.1.10-i486-1
-rw-r--r-- 1 root root 11696 2007-08-03 10:06 atk-1.18.0-i486-1
--More--

```

Em virtude da existência de antigos programas ainda em vigor, deverão existir alguns subdiretórios para a compatibilidade com os mesmos em */var*. Os diretórios que compõe esta estrutura são: */var/backups*, */var/cron*, */var/lib*, */var/local*, */var/msgs* e */var/preserve*.

SOBRE A NORMA FHS

✓ <<http://www.pathname.com/fhs/>>.

A *FHS - Filesystem Hierarchy Standard* - é um conjunto de requerimentos técnicos que visam estabelecer normas e padrões para a estrutura do sistema de arquivos *Unix*, derivados e clones. É ela quem define quais são os diretórios que deverão existir, a localização dos arquivos de configuração, os atalhos simbólicos, entre outros, com o intuito de promover a compatibilidade dos sistemas *GNU/Linux* e suas aplicações.³

CONCLUSÃO

O conhecimento das características e particularidades do sistema de arquivos de um sistema operacional é fator de suma importância para a sua administração. Saber as funcionalidades de determinados arquivos e diretórios, localização dos arquivos de configuração, permissões de acesso, tudo isso influencia no momento de uma necessidade de intervenção.

3 O *FSSTND - Linux Filesystem Structure* - foi concebido anteriormente e com os mesmos propósitos da *FHS*, porém devido a sua pouca rigidez sobre diversos aspectos, muitas distribuições auto-definiam a localização de diversos arquivos de sistema. Os arquivos de inicialização e configuração do sistema eram os que mais situavam-se fora de uma padronização específica, mesmo que estas distribuições tomassem como base os métodos de inicialização *SystemV* e *BSD*.

Por mais diferentes que sejam as distribuições, todas elas possuem basicamente a mesma estrutura do sistema de dados. Graças à isto, as administrações e intervenções necessárias no sistema de dados poderão ser realizadas em quaisquer sistema *GNU/Linux* sem maiores transtornos. Eis um dos motivos da importância de se utilizar os recursos da linha de comando para interagirmos na manutenção do sistema! &;-D

B

II. OS ARQUIVOS DE DISPOSITIVOS

INTRODUÇÃO

Todos os periféricos e recursos do sistema são acessados pelo *kernel* através de arquivos de dispositivos - conhecidos também por *devices*. Se quisermos formatar um disquete, será necessário a utilização de um desses arquivos; para acessar a *Internet*, será necessário outro arquivo de dispositivo; para utilizar um terminal, mais outro... e assim por diante.

Neste capítulo iremos conhecer os principais arquivos de dispositivos - chamaremos apenas de *devices* para facilitar a pronúncia.

A CLASSIFICAÇÃO

Os *devices* são classificados em 2 categorias, à saber: os *devices* do tipo caracter e os *devices* do tipo de bloco.

TIPO CARACTER

Os *devices* do tipo caracter são aqueles em que a transferência de dados são realizadas de modo *serial*, ou seja, um caracter por vez. Dentre os principais exemplos estão as portas paralelas (impressora), portas seriais (*modems*), *devices* de áudio, terminais, teclado, *mouse*, etc.

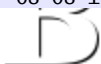
```
$ cd /dev
$ ls -l /dev/lp*
crw-rw-r-- 1 root lp 6, 0 2007-08-08 18:00 /dev/lp0
crw-rw-r-- 1 root lp 6, 0 2007-08-08 18:00 /dev/lp1
crw-rw-r-- 1 root lp 6, 0 2007-08-08 18:00 /dev/lp2
$ _
```

Exemplos de devices do tipo caracter.

TIPO BLOCO

Já nos *devices* do tipo bloco diferenciam-se do tipo caracter no que concerne a transferência de dados - pois como o próprio nome diz - é feita por blocos, oferecendo grande quantidade de dados por vez. Já nesta categoria estão em geral os *devices* de armazenamento tais como disquetes, discos rígidos, *CDs* & *DVDs*, dispositivos de armazenamento *USB* (memória eletrônica), entre outros, devido ao modo de transferência de dados.

```
$ cd /dev
$ ls -l /dev/hd*
brw-rw---- 1 root disk 3, 0 2007-08-08 14:59 /dev/hda
brw-rw---- 1 root disk 3, 1 2007-08-08 14:59 /dev/hda1
```



```
brw-rw---- 1 root disk 3, 2 2007-08-08 14:59 /dev/hda2
brw-rw---- 1 root disk 3, 5 2007-08-08 14:59 /dev/hda5
brw-rw---- 1 root disk 3, 6 2007-08-08 14:59 /dev/hda6
brw-rw---- 1 root disk 3, 7 2007-08-08 18:00 /dev/hda7
brw-rw---- 1 root cdrom 22, 0 2007-08-08 14:59 /dev/hdc
brw-rw---- 1 root cdrom 22, 64 2007-08-08 14:59 /dev/hdd
$ _
```

Exemplos de dispositivos do tipo bloco.

A ESTRUTURA /DEV

Existe uma imensa quantidade de *devices* específicos; porém nesta literatura, somente conheceremos os mais utilizados pelos usuários comuns. Estes por sua vez subdividem-se em diversas categorias e encontram-se armazenados no diretório */dev*.

UNIDADES EXTERNAS E DE ARMAZENAMENTO

DISCOS RÍGIDOS IDE & CD-ROMs

Todos os discos rígidos conectados a interface *IDE* utilizam os seguintes *devices* para serem acessados pelo sistema:

Discos rígidos IDE & CD-ROMs	
<i>hda</i>	1a. unidade na controladora primária - mestre.
<i>hda1, hda2...</i>	Partições da 1a. unidade na controladora primária.
<i>hdb</i>	2a. unidade na controladora primária - escravo.
<i>hdb1, hdb2...</i>	Partições da 2a. unidade na controladora primária - escravo.
<i>hdc</i>	1a. unidade na controladora secundária - mestre.
<i>hdc1, hdc2...</i>	Partições da 1a. unidade na controladora secundária - mestre.
<i>hdd</i>	2a. unidade na controladora secundária - escravo.
<i>hdd1, hdd2...</i>	Partições da 2a. unidade na controladora secundária - escravo.

Quanto aos *CD-ROMs*, na verdade não existem *devices* para acesso as unidades leitoras de *CD-ROMs* atuais, e sim apenas atalhos simbólicos indicando em qual os *devices* reais estes se encontram. Este procedimento é necessário face a utilização dos *devices* para as diferentes localizações (primário, secundário, mestre, escravo, etc.) e antigos *drivers* de *CD-ROMs* que não utilizavam a controladoras *IDE* para serem conectados ao sistema. Para acessá-los, deveremos utilizar o atalho simbólico */dev/cdrom*.



UNIDADES SCSI E SATA

Sem grandes mistérios, estes são os *devices* para as unidades SCSI e SATA:

Unidades SCSI	
<i>sda</i>	1a. unidade nominal (disco rígido, CD-R/W, memória eletrônica, etc.).
<i>sda1, 2, 3.</i>	Indicativo de particionamento das unidades emuladas.
<i>sda4</i>	Indicativo de unidade, utilizado especificamente pelo <i>Zip-drive</i> .
<i>sdb, sdc...</i>	Unidades seqüenciais.

No *kernel 2.4*, os gravadores de CD-R/W e DVD-R/W - são acessados através de emulação SCSI. Por este motivo, eles devem utilizar estes *devices*. Já para o *kernel 2.6*, os *devices* são os mesmos que os utilizados para os discos rígidos padrão IDE.

Em tratando-se de disco rígido, segue-se o mesmo padrão das unidades IDE, somente atentando-se para alterar o caracter *h* para o caracter *s*.

DISQUETES

Os disquetes são acessados através dos *devices* */dev/fd[X]*.

Disquetes	
<i>fd0</i>	1a. unidade de disquete - em DOS corresponde a letra A:.
<i>fd0d360</i>	1a. unid. de disquete, formato baixa densidade, cap. 360 KB.
<i>fd0h720</i>	1a. unid. de disquete, formato alta densidade, cap. 720 KB.
<i>fd0u1200</i>	1a. unid. de disquete, formato alta densidade, cap. 1200 KB.
<i>fd0u1440</i>	1a. unid. de disquete, formato alta densidade, cap. 1440 KB.
<i>fd1...</i>	2a. unidade de disquete - em DOS corresponde a letra B:.

De acordo com as necessidades, existirão circunstâncias em que iremos referir-se a um dos *devices* específicos. Por exemplo, para formatação, iremos defini-lo como */dev/fd[X][D]*, onde *[X]* indica o *device* da unidade em questão e *[D]* a densidade da mesma.

DISPOSITIVOS DE ÁUDIO

Em vista dos diferentes recursos de áudio presentes na maioria das placas de som do mercado, os sistemas GNU/Linux não utilizam somente um, e sim um conjunto de *devices* para o acesso a estes dispositivos.

Segue abaixo, uma simples listagem padrão para uma melhor compreensão:

B

Sistema de áudio (antigo sistema OSS)	
<i>audio</i>	Sintetizador de áudio (<i>wave-table</i>).
<i>dsp</i>	Voz digitalizada.
<i>midi</i>	Sintetizador de instrumentos (<i>MIDI</i>).
<i>mixer</i>	Ajustes e configurações (mixagem).
<i>sequencer</i>	Sequenciador.

Todos estes devices pertencem ao grupo *audio*, onde devemos incluí-lo nas configurações das contas de usuário para que as propriedades de áudio estejam presentes. Para obterem maiores informações de como proceder, consultem na *4a. Parte: Ajustes & Configurações -> Áudio - Placa de som*.

FAX-MODEM (PORTAS SERIAIS)

As tradicionais placas de *fax-modem* - conhecidas popularmente como *hardmodems* - são referidas no sistema através dos *devices /dev/ttyS[X]*.

Portas seriais	
<i>ttyS0</i>	Porta <i>serial 1</i> - equivale ao <i>COM1</i> .
<i>ttyS1</i>	Porta <i>serial 2</i> - equivale ao <i>COM2</i> .
<i>ttyS2</i>	Porta <i>serial 3</i> - equivale ao <i>COM3</i> .
<i>ttyS3</i>	Porta <i>serial 4</i> - equivale ao <i>COM4</i> .

Já no caso dos *softmodems*, a maior parte destes periféricos utilizam um *device* especial, criado pelos *drivers* para a sua instalação.

Para obterem maiores informações sobre estes periféricos, consultem na *4a. Parte: Ajustes & Configurações -> Modem - placas de fax-modem*.

CONSOLE TERMINAL

Um terminal é uma interface entre o usuário e o sistema. Ao serem inicializados, utiliza o *device /dev/tty[X]*.

Console terminal	
<i>ttyX</i>	Um terminal propriamente dito.
<i>ttypX</i>	Terminais <i>SSH/Telnet</i> .

Somente teremos à disponibilidade diversos terminais desde que o *kernel* tenha o suporte aos terminais virtuais. Felizmente, todos os *kernels* são pré-compilados com este recurso habilitado.



SOBRE O UDEV

✓ <<http://www.kernel.org/pub/linux/utils/kernel/hotplug/udev.html>>.

O *udev* subsistema desenvolvido para o *kernel*, que tem como objetivo atuar como gerenciador de arquivos de dispositivos (*devices*) dinâmicos. Ele é o responsável pela criação automática dos *devices* relacionados aos dispositivos que se encontram disponíveis, gerando assim um único *device*, ao invés de uma estrutura complexa e infindável de *devices*.

Não existem muitas intervenções necessárias para serem feitas nas configurações do *udev*; porém, suas regras relacionadas ao disparo de ações, que por sua vez são causadas quando um novo dispositivo é conectado (p. ex. *pendrives*) podem ser perfeitamente customizadas. Tais regras se encontram armazenadas em */etc/udev/rules.d*.

```
$ cd /etc/udev/  
$ ls -l  
total 4  
drwxr-xr-x 2 root root 440 2007-08-12 19:29 rules.d/  
-rw-r--r-- 1 root root 576 2007-05-17 16:30 udev.conf
```

(...)

```
$ cd rules.d/  
$ ls -l  
total 120  
-rw-r--r-- 1 root root 16054 2007-06-28 21:27 50-udev.rules  
-rw-r--r-- 1 root root 2199 2007-05-19 03:04 60-bluetooth.rules  
-rw-r--r-- 1 root root 926 2007-03-16 17:00 60-pcmcia.rules  
-rw-r--r-- 1 root root 1289 2007-06-02 18:36 64-device-mapper.rules  
-rw-r--r-- 1 root root 511 2007-08-12 16:29 75-network-devices.rules  
-rw-r--r-- 1 root root 1414 2007-08-04 10:20 75-optical-devices.rules  
-rw-r--r-- 1 root root 8825 2007-05-27 01:52 80-libmtp.rules  
-rw-r--r-- 1 root root 1704 2007-05-27 01:59 80-libnjb.rules  
-rw-r--r-- 1 root root 2505 2007-06-24 03:39 80-libpisock.rules  
-rw-r--r-- 1 root root 61148 2007-05-27 02:12 80-libsane.rules  
-rw-r--r-- 1 root root 82 2007-06-27 21:48 90-hal.rules  
$ _
```

CONCLUSÃO

Acreditamos que um simples passeio pelos *devices* mais utilizados pelo sistema possa deixar os usuários mais familiarizados com a administração e manutenção do sistema operacional em geral. Lembrem-se: os sistemas *Unix* em geral referem-se a qualquer *device* do sistema como arquivo. Quaisquer intervenção necessária, sempre tenha consciência de que poderá intervir nos arquivos da estrutura */dev*. &;-D



III. A LINHA DE COMANDO

INTRODUÇÃO

A disponibilidade de diversos utilitários gráficos, facilitam muito a administração de sistemas *GNU/Linux*. Porém, de acordo com as necessidades, facilidades de uso e recursos (ou ausência destes), existirão diversas circunstâncias em que teremos de usar tais recursos através de um aplicativo especial chamado interpretador de comando, também conhecido popularmente como a linha de comando ou *terminal*.⁴

Neste capítulo, iremos conhecer alguns recursos e funcionalidades da linha de comando. Em especial, destacaremos o *BASH*, o interpretador de comando oficial dos sistemas *GNU/Linux*, segundo as especificação *LSB*.

O BOURNE AGAIN SHELL

✓ <<http://www.gnu.org/software/bash/>>.

O interpretador de comandos é um programa especial que permite a interação do usuário com o sistema operacional através da utilização de comandos especiais via teclado. É nele em que coordenamos muitas das atividades administrativas pertinentes as funções do *kernel*, como a manipulação de arquivos, a edição de configurações, o gerenciamento de processos, entre outras atividades. Nos sistemas *GNU/Linux*, o *BASH* - *Bourne Again Shell* - é o interpretador de comandos oficial.

4 Infelizmente muitos usuários têm conceitos errôneos sobre o uso da linha de comando. Já escutamos diversos comentários específicos e sem fundamentos, tais como “*isso é coisa do passado...*”, “*voltar aos tempos do DOS...*”, “*ninguém mais usa isto...*”, “*pra quê mexer nessa !%#&*...*”, etc., mas em sistemas *GNU/Linux* ela é essencial e indispensável para o bom funcionamento do sistema.

```
Shell - Konsole
Session Edit View Bookmarks Settings Help

$ ls
total 89
drwxr-xr-x 2 root root 3904 2006-10-11 21:19 bin
drwxr-xr-x 2 root root 400 2006-10-11 21:29 boot
drwxr-xr-x 17 root root 63456 2006-11-03 22:23 dev
drwxr-xr-x 52 root root 5112 2006-11-05 08:33 etc
drwxr-xr-x 6 root root 120 2006-10-29 21:38 home
drwxr-xr-x 6 root root 3400 2006-10-11 21:19 lib
drwxr-xr-x 16 root root 600 2006-10-11 21:12 media
drwxr-xr-x 14 root root 368 2006-10-21 23:21 mnt
drwxr-xr-x 5 root root 136 2006-10-29 21:26 opt
dr-xr-xr-x 85 root root 0 2006-11-03 20:22 proc
drwx-x-x 13 root root 576 2006-11-05 09:01 root
drwxr-xr-x 2 root root 6824 2006-08-16 17:20/sbin
drwxr-xr-x 2 root root 48 2004-05-12 01:03 sys
drwxrwxrwt 23 root root 1000 2006-11-05 09:33 tmp
drwxr-xr-x 19 root root 544 2006-10-29 21:26 usr
drwxr-xr-x 19 root root 544 2006-10-25 14:51 var
$
```

Terminal virtual (Konsole) do KDE.

Desenvolvido pelo *Projeto GNU*, o *BASH* é uma derivação do antigo *Shell Bourne*, que leva o nome de seu criador original: *Steven Bourne*. O *BASH* é um trocadinho, pois o *Again* significa “*novamente*”, ou seja: “*Novamente um Shell Bourne*”. O *Shell Bourne* e o *BASH* são compatíveis, embora diversas melhorias tenham sido feitas neste último...

À primeira vista, o *BASH* lembra-se muito o *MS-DOS*, por ele disponibilizar uma tela de texto preta e uma linha de comando. Nesta literatura o utilizaremos apenas as instruções básicas e essenciais, necessárias para garantirmos a boa manutenção de um sistema para o uso em *desktops*.

Por se tratar de um requerimento das especificações do padrão *LSB*, torna-se essencial conhecer as suas características e as funcionalidades básicas.

INFORMAÇÕES E MÉTODOS ESSENCIAIS

São inúmeros os recursos disponibilizados pelo *BASH*; por isto, iremos omitir as instruções detalhadas e desnecessárias. Seguem abaixo apenas algumas informações e métodos básicos e essenciais para obtermos um excelente rendimento nas intervenções que iremos realizar mais à frente.

COMPLEMENTO DA TECLA <TAB>

Em muitas circunstâncias, teremos a necessidade de digitar toda a nomenclatura de um comando, arquivo ou diretório na linha de comando. Ao invés de digitarmos cada caracter, poderemos apenas digitar as iniciais e complementar com a tecla <TAB>, onde o sistema se encarregará de localizar tal nomenclatura e preencher com os dados restantes.



Por exemplo, ao digitarmos...

```
$ mce<TAB>
```

... o resultado final será...

```
$ mcedit _
```

Simple e prático, evitando o inconveniente de ter que redigitar toda a nomenclatura caso ocorram simples erros de digitação.

Dependendo das circunstâncias, poderão existir mais de uma ocorrência para a complementação do comando a ser digitado:

```
$ xorg<TAB>
```

... resultará em...

```
$ xorg<TAB>
xorgcfg      xorgconfig  xorgsetup
$ xorg_
```

Nestes casos, bastará complementar aos poucos os caracteres restantes e teclar novamente <TAB>, ou ainda, digitar o comando por inteiro.

O mesmo se dá para a digitação dos nomes de arquivos, pois...

```
# rpm -ivh quake-1<TAB>
```

... resultará em...

```
# rpm -ivh quake-1.1-6cl.i386.rpm _
```

O USO DE EXPRESSÕES

Expressões são conjuntos de símbolos e caracteres que visam especificar uma determinada informação ou o conjunto delas. Estes são conhecidos popularmente como *curingas*, que podem representar um valor, uma definição, um conjunto destes e até mesmo um comando.

Descreveremos aqui os principais caracteres utilizados em sistemas *GNU/Linux* e suas principais funcionalidades:

- **Asterisco (*):** popularmente representa “tudo”, ou seja, qualquer campo ou instrução que estiver sendo representado por um asterisco, indicará todos os possíveis caracteres;
- **Ponto (.) e ponto-ponto (..):** O ponto representa entrada de diretórios, ao passo que o ponto-ponto - .. - representa o diretório-pai (diretório anterior);
- **Interrogação (?):** similar ao caracter “*”, porém somente representa os caracteres que se situarem na posição onde este se encontra;
- **Pipes (!):** processa a saída de um comando para que seja usado como dados ou parâmetros em outro comando, onde dois pipes servem para executar 2 comandos seqüenciais, independente de haver erro no 1o. comando, como também utilizado para

representar a expressão lógica *OR* (ou);

- *E-comercial (&):* execução de aplicações em 2o. plano (*background*), onde duas *&&* servem para executar 2 comandos seqüenciais, desde que o primeiro não retorne nenhum erro, como também utilizado para representar a expressão lógica *AND* (e).

Ao contrário do que muitos pensam, existem diferenças entre eles: os símbolos especiais possuem funções específicas; já os caracteres coringas representam e/ou substituem outros caracteres e/ou conjunto de letras.

AS CORES PERSONALIZADAS

Outra ponto a ser observado são as cores dos arquivos e diretórios a serem exibidos em modo texto, pois conforme já dito anteriormente, os sistemas *GNU/Linux* utilizam como padrão o interpretador de comandos *BASH*. Nele, temos padronizado o seguinte perfil de cores:

- *Amarelo:* dispositivos do sistema (*devices*);
- *Azul:* diretórios (seguidos pelo caracter *"/*");
- *Azul ciano:* atalhos simbólicos (*links*);
- *Cinza:* arquivos diversos ou desconhecidos;
- *Magenta:* arquivos de imagens bitmaps (*JPEG, GIF, PNG, etc.*);
- *Verde:* arquivos executáveis (arquivos de lote e binários);⁵
- *Vermelho:* arquivos compactados (inclusive pacotes de instalação).

Vale lembrar que, dependendo tanto dos atributos específicos dos arquivos, quanto das configurações utilizadas no terminal, muitos poderão ter cores diferentes dos padrões acima citados. É o caso de arquivos com atributos para execução (*flag x*), que aparecem com a cor verde no vídeo, seja uma imagem, um arquivo compactado, etc.

NOMENCLATURA DIFERENCIADA

Da mesma forma que os arquivos e diretórios apresentam um sistema de cores para a sua identificação, os mesmos possuem sinais especiais para informar determinados atributos:

<i>Sinal / Significado / Exemplo</i>		
.	Arquivo oculto.	.Confidencial
*	Arquivo executável.	Programa*
@	Atalho.	Programa@ -> Programa-1.0.2

Lembre-se que de acordo com o interpretador de comando utilizado,

⁵ Estes indicam arquivos com permissões para execução, ainda que sejam um texto, uma imagem, um pacote compactado e outros quaisquer.



poderão ser exibidos ou não, tais atributos.

ACESSO À DOCUMENTAÇÃO ELETRÔNICA

Uma das características interessantes dos sistemas *GNU/Linux* está na forte documentação eletrônica dos utilitários e comandos disponíveis, onde uma simples consulta poderá resolver a maioria das dúvidas existentes. Tais informações podem ser acessados no *BASH* através da execução do próprio comando, porém com a adição do parâmetro *--help*:

```
$ ps --help
***** simple selection *****          ***** selection by list *****
-A all processes                          -C by command name
-N negate selection                       -G by real group ID (supports names)
-a all w/ tty except session leaders      -U by real user ID (supports names)
-d all except session leaders             -g by session leader OR by group name
-e all processes                          -p by process ID
T all processes on this terminal           -s processes in the sessions given
a all w/ tty, including other users        -t by tty
g all, even group leaders!                 -u by effective user ID (supports names)
r only running processes                  U processes for specified users
x processes w/o controlling ttys          t by tty
***** output format *****             ***** long options *****
-o,o user-defined  -f full                 --Group --User --pid --cols
-j,j job control  s signal                 --group --user --sid --rows
-O,O preloaded -o v virtual memory         --cumulative --format --deselect
-l,l long          u user-oriented         --sort --tty --forest --version
                                X registers --heading --no-heading
                                ***** misc options *****
-V,V show version  L list format codes  f ASCII art forest
-m,m show threads  S children in sum     -y change -l format
-n,N set namelist file c true command name n numeric WCHAN,UID
-w,w wide output   e show environment    -H process heirarchy
$ _
```

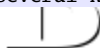
Se for necessária a obtenção de informações mais detalhadas, o *man...*

```
$ man ps
...
PS(1)                                Linux User's Manual                                PS(1)
,
NAME
  ps - report process status

SYNOPSIS
  ps [options]

DESCRIPTION
  ps(1) gives a snapshot of the current processes. If you
  want a repetitive update of this status, use top.

COMMAND-LINE OPTIONS
  This version of ps accepts several kinds of options.
```



```
Unix98 options may be grouped and must be preceded
by a dash.
BSD options may be grouped and must not be used
with a dash.
GNU long options are preceded by two dashes.
Options of different types may be freely mixed.
```

```
Set the I_WANT_A_BROKEN_PS environment variable to force
lines 1-28
...
```

... e o *info*...

```
$ info ps
...
File: a2ps.info, Node: psmandup, Next: psset, Prev: pdiff, Up: Contribution\
s
`psmandup'
=====
```

I personally hate to print documents of hundreds of pages on a single sided printer. Too bad, here there are no Duplex printers. The idea is then simply first to print the odd pages, then the even in reversed order. To make sure one flips the page in the meanwhile, the second half should be printed from the manual feed tray.

Make a shell script that automates this, and you get ``psmandup'`.

* Menu:

* Invoking psmandup:: Command Line Interface

```
--zz-Info: (a2ps.info.gz)psmandup, 18 lines --All-- Subfile: a2ps.info-5.gz-----
Welcome to Info version 4.8. Type ? for help, m for menu item.
```

... certamente atenderão perfeitamente bem a estes propósitos.

EQUIVALÊNCIAS ENTRE O BASH E MS-DOS

Apesar de se encontrar praticamente em desuso por usuários do *Windows*, muitos usuários sentem-se confortáveis ao realizar diversas intervenções com a utilização da linha de comando do *MS-DOS*, especialmente quando não há possibilidade de inicializar o *Windows* para estes eventos. Já em sistemas *GNU/Linux*, a linha de comando não só se encontra presente, como também é fundamental para a manutenção geral do sistema e ainda possui uma eficiência muito superior à que encontramos no velho *MS-DOS*.



ESTRUTURA DE DIRETÓRIOS

A principal diferença entre os sistemas *GNU/Linux* e o *MS-DOS* encontra-se na estrutura de diretórios do sistema. Enquanto o *MS-DOS* suporta somente a formato de nomes 8.3 (até a versão 6.22) e dispõe somente de permissões de acesso para somente leitura e ocultação (atributos)...

```
C:\>dir
O volume na unidade C é DARKSTAR
O número de série do volume é 0816-A972

Pasta de C:\

12/05/2004 20:03 <DIR>          WINDOWS
24/05/2004 23:23 <DIR>          Documents and Settings
12/05/2004 20:18 <DIR>          Arquivos de programas
12/05/2004 20:19             0 CONFIG.SYS
12/05/2004 20:19             0 AUTOEXEC.BAT
                2 arquivo(s)          0 bytes
                3 pasta(s) 3.165.028.352 bytes disponíveis

C:\>_
```

... o *BASH* suporta nomes com até 255 caracteres, além do simples e eficiente sistema permissões de acesso de leitura, escrita e execução.

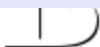
```
$ ls -l
total 82
drwxr-xr-x  2 root  bin           2304 Jun 28 17:09 bin/
drwxr-xr-x  2 root  root           336 Set  5 22:14 boot/
drwxr-xr-x 15 root  root        61072 Set  6 09:58 dev/
drwxr-xr-x 45 root  root         4520 Set  6 10:36 etc/
drwxr-xr-x  5 root  root          128 Ago 20 23:11 home/
drwxr-xr-x  4 root  root         2520 Jun 28 17:08 lib/
drwxr-xr-x  6 root  root          144 Jun 28 17:16 mnt/
drwxr-xr-x  4 root  root           96 Ago 23 18:48 opt/
dr-xr-xr-x 73 root  root            0 Set  6 06:57 proc/
drwx--x--- 25 root  root         1064 Set  5 13:55 root/
drwxr-xr-x  2 root  bin           5456 Jun  1 2002 sbin/
drwxrwxrwt 31 root  root         1352 Set  6 10:09 tmp/
drwxr-xr-x 21 root  root           592 Mar  6 2003 usr/
drwxr-xr-x 17 root  root           456 Mar  2 2003 var/
$ _
```

Além destes, existem outros recursos presentes em sua linha de comando.

ACESSO AS UNIDADES DO SISTEMA

O *MS-DOS* atribui letras para a unidade de armazenamento de dados (A:, C:, D:, etc.), onde para acessarmos as demais unidades do sistema, bastaria apenas indicá-las na linha de comando acrescido de <DOIS_PONTOS> (":"), teclando <ENTER> em seguida:

```
C:\> A:
A:\> D:
```



```
D:\> C:
```

```
C:\> _
```

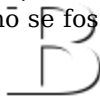
Já os sistemas *GNU/Linux* possui apenas um único diretório raiz, do qual as demais unidades encontram-se previamente montadas em seus respectivos subdiretórios. No *BASH*, para acessarmos as demais unidades do sistema, deveremos montar os dispositivos e acessá-los através do ponto de montagem, situados no diretório */mnt/<UNIDADE>* conforme a norma *FHS*. O ponto de montagem */mnt* possui basicamente a seguinte estrutura:

```
$ cd /mnt
$ ls -l
total 8
-rw-r--r--  1 root    root    376 2006-09-26 00:09 README
drwxr-xr-x  2 root    root     48 2006-09-25 22:02 cdrecorder
drwxr-xr-x  2 root    root     48 2002-03-16 04:34 cdrom
drwxr-xr-x  2 root    root     48 2006-09-25 22:02 dvd
drwxr-xr-x  2 root    root     48 2002-03-16 04:34 floppy
drwxr-xr-x  2 root    root     48 2002-03-16 04:34 hd
drwxr-xr-x  2 root    root     48 2006-09-25 22:02 memory
drwxr-xr-x 10 root    root    224 2007-08-04 15:08 pkg
drwxr-xr-x  2 root    root     48 2006-09-25 22:03 tmp
dr-x-----  1 root    root   4096 2007-08-05 00:59 win
drwxr-xr-x  2 root    root     48 2006-09-25 22:02 zip
$ _
```

Por padrão existe apenas estes diretórios, porém caso necessitem trabalhar com outros dispositivos do sistema, basta criá-los conforme a necessidade. Lembrem-se de que precisarão estar com os poderes de superusuário.

```
$ su
Password:
# mkdir flash
# mkdir tape
# ls -l
total 8
-rw-r--r--  1 root    root    376 2006-09-26 00:09 README
drwxr-xr-x  2 root    root     48 2006-09-25 22:02 cdrecorder
drwxr-xr-x  2 root    root     48 2002-03-16 04:34 cdrom
drwxr-xr-x  2 root    root     48 2006-09-25 22:02 dvd
drwxr-xr-x  2 root    root     48 2007-08-05 19:07 flash
drwxr-xr-x  2 root    root     48 2002-03-16 04:34 floppy
drwxr-xr-x  2 root    root     48 2002-03-16 04:34 hd
drwxr-xr-x  2 root    root     48 2006-09-25 22:02 memory
drwxr-xr-x 10 root    root    224 2007-08-04 15:08 pkg
drwxr-xr-x  2 root    root     48 2007-08-05 19:07 tape
drwxr-xr-x  2 root    root     48 2006-09-25 22:03 tmp
dr-x-----  1 root    root   4096 2007-08-05 00:59 win
drwxr-xr-x  2 root    root     48 2006-09-25 22:02 zip
# _
```

Após a montagem das partições e/ou unidades, seus respectivos acessos são realizados normalmente, como se fossem simples subdiretórios.



ESPECIFICAÇÕES DO DIRETÓRIO

Nos sistemas *GNU/Linux*, a especificação dos caminhos de diretórios é feita de forma similar ao *MS-DOS*...

```
$ cd /<DIRETÓRIO>
```

... porém outra diferença simples está no uso da <BARRA> - / - ao invés da <BARRA_INVERTIDA> - \ - para a navegação entre os diretórios.

```
C:\> cd <UNIDADE>\<DIRETÓRIO>
```

Em ambos, para navegar até o diretório raiz:

```
C:\WINDOWS> cd \
```

...

```
$ cd /
```

EXIBIÇÃO DE CAMINHO

Em ambos os interpretadores - e de acordo com a distribuição utilizada - o *prompt* da linha de comando indicam em que diretório estamos...

```
C:\WINDOWS> _
```

... tendo o *BASH* a vantagem de informar a conta autenticada no momento.

```
darkstar@darkstar:/usr/doc$ _
```

As definições do prompt podem ser alteradas através da customização do arquivo de configuração */etc/profile*, seção *default shell prompt*.

CASE SENSITIVE

O sistemas de nomenclatura dos arquivos e diretórios do *GNU/Linux* é *case sensitive*, ou seja, o tratamento de caracteres maiúsculos e minúsculos é diferente em comparação ao *MS-DOS*, onde neste tanto faz utilizar caracteres maiúsculos quanto minúsculos:

```
C:\> CD WINDOWS
```

```
C:\WINDOWS> CD ..
```

```
C:\> cd windows
```

```
C:\WINDOWS> _
```

Já no *BASH*, o sistema não reconhecerá o comando utilizado caso utilizem caracteres maiúsculos ao invés de maiúsculos, e vice-versa.

```
darkstar@darkstar:/$ cd /usr
```

```
darkstar@darkstar:/usr$ CD ..
```

```
-bash: CD: command not found
```

```
darkstar@darkstar:/usr$ _
```



OBSERVAÇÕES

As aplicações disponibilizadas em interfaces gráficas geralmente tem a simpatia dos usuários menos habituados ao modo texto, graças as facilidades condicionadas, como a intuitividade e o apelo visual, tornando bem mais atrativo e agradável realizar as operações necessárias no sistema. Porém, existirão circunstâncias específicas e variadas em que a utilização de uma linha de comando será praticamente insubstituível. Mesmo apesar do vastos recursos disponibilizados nas atuais aplicações gráficas, em muitos aspectos teremos maior eficiência, prática e agilidade em realizar as operações necessárias na linha de comando.

CONCLUSÃO

Apesar da existência de diversas diferenças em comparação a linha de comando do *MS-DOS*, o objetivo o *BASH* é o mesmo: fornecer ao usuário um conjunto de recursos que poderão ser habilitados pela linha de comando. Porém são inúmeras as diferenças e características, além da disponibilidade de recursos extras que o tornam muito superior ao *MS-DOS* e quaisquer outros aplicativos gráficos existentes. &:-D

B

IV. MANIPULAÇÃO DE ARQUIVOS E DIRETÓRIOS

INTRODUÇÃO

Nas mais variadas circunstâncias, a manipulação direta de arquivos e diretórios será necessária para a realização de determinadas intervenções. A checagem de conteúdo, a organização de documentos, a edição direta, a definição de permissões, entre outras, são simples e bons exemplos de operações realizadas em imensas possibilidades.

Neste capítulo, apresentaremos as principais operações de manipulação, que por sua vez se encontram subdivididas por categorias, onde as instruções necessárias estarão estruturadas de uma forma bem simples e organizada para uma melhor compreensão destes processos.

OPERAÇÕES BÁSICAS

Entende-se como operações básicas: a listagem, visualização, edição e manipulação, dos arquivos e diretórios gerais que compõe o sistema, além de dados e informações adicionais.

LISTAGEM E NAVEGAÇÃO

LS

Lista os arquivos e diretórios do diretório corrente ou especificado.

Sintaxe:

```
ls [PARÂMETROS] [ARQUIVOS/DIRETÓRIOS]
```

Onde:

- *-a*: lista todos os arquivos e diretórios ocultos;
- *-R*: lista todos os arquivos e diretórios de forma recursiva;
- *-i*: lista o conteúdo, porém exibindo o tamanho em blocos;
- *-l*: lista o conteúdo, exibindo-o em somente uma simples coluna;
- *-l*: tal como *-l*, lista o conteúdo, exibindo-o em somente uma, mas contendo informações gerais acerca de seus atributos;

Observem que os parâmetros exemplificados podem ser utilizados em conjunto (combinados). Experimentem utilizar estes parâmetros combinado com outros parâmetros e verifiquem sua saída:

```
$ ls -li
```

```
total 83
 11 drwxr-xr-x  2 root   bin           2416 2004-01-31 07:43 bin/
 20 drwxr-xr-x  2 root   root          336 2004-01-31 07:55 boot/
  9 drwxr-xr-x 16 root   root        62352 2004-01-31 12:09 dev/
 12 drwxr-xr-x 47 root   root         4808 2004-01-31 15:06 etc/
  2 drwxr-xr-x  5 root   root          96 2004-01-31 10:27 home/
 15 drwxr-xr-x  4 root   root        2592 2004-01-31 07:42 lib/
 16 drwxr-xr-x  5 root   root         120 2004-01-31 10:08 mnt/
11750 drwxr-xr-x  4 root   root          96 2002-12-13 19:01 opt/
  1 dr-xr-xr-x 86 root   root           0 2004-01-31 10:09 proc/
  7 drwx--x--- 11 root   root         656 2004-01-31 14:31 root/
 22 drwxr-xr-x  2 root   bin        5704 2003-09-01 19:29/sbin/
  2 drwxrwxrwt 19 root   root         648 2004-01-31 15:17 tmp/
  2 drwxr-xr-x 19 root   root         544 2004-01-05 18:32 usr/
  2 drwxr-xr-x 17 root   root         456 2003-08-15 00:17 var/
$ _
```

Além do comando `ls`, podemos também utilizar os comandos `dir...`

`$ dir`

```
bin/  dev/  home/  mnt/  proc/ /sbin/  usr/
boot/ etc/  lib/   opt/  root/  tmp/   var/
$ _
```

... e `vdir`, que correspondem respectivamente a `ls` e `ls -l`.

`$ vdir`

```
total 83
drwxr-xr-x  2 root   bin           2416 2004-01-31 07:43 bin/
drwxr-xr-x  2 root   root          336 2004-01-31 07:55 boot/
drwxr-xr-x 16 root   root        62352 2004-01-31 12:09 dev/
drwxr-xr-x 47 root   root         4808 2004-01-31 15:06 etc/
drwxr-xr-x  5 root   root          96 2004-01-31 10:27 home/
drwxr-xr-x  4 root   root        2592 2004-01-31 07:42 lib/
drwxr-xr-x  5 root   root         120 2004-01-31 10:08 mnt/
drwxr-xr-x  4 root   root          96 2002-12-13 19:01 opt/
dr-xr-xr-x 93 root   root           0 2004-01-31 10:09 proc/
drwx--x--- 11 root   root         656 2004-01-31 14:31 root/
drwxr-xr-x  2 root   bin        5704 2003-09-01 19:29/sbin/
drwxrwxrwt 19 root   root         648 2004-01-31 15:17 tmp/
drwxr-xr-x 19 root   root         544 2004-01-05 18:32 usr/
drwxr-xr-x 17 root   root         456 2003-08-15 00:17 var/
$ _
```

Os comandos `ls -l` e `vdir` são os mais indicados para verificar todas as informações disponíveis sobre a estrutura de arquivos e diretórios.

CD

Possibilita navegar (“*entrar e sair*”) entre os diretórios desejados.

Sintaxe:

```
$ cd [DIRETÓRIO]
```

Exemplo:



```
darkstar@darkstar:~$ cd /usr
darkstar@darkstar:/usr$ ls -l
...
darkstar@darkstar:/usr$ cd lib
darkstar@darkstar:/usr/lib$ ls -l
...
darkstar@darkstar:/usr/lib$ cd X11
darkstar@darkstar:/usr/lib/X11$ _
```

Observe que, para entrarmos no subdiretório *lib* e *X11*, não foram especificados os caminhos completos, pois estes pertencem ao diretório do qual se encontra localizado no momento (referência local).

Como podem ver, seu uso é bastante simples, não existindo qualquer grau de dificuldade. Ainda assim podemos utilizar outros recursos, tais como:

```
darkstar@darkstar:/usr/lib/X11$ cd ..
darkstar@darkstar:/usr/lib$ cd /
darkstar@darkstar:/$ _
```

É bem mais simples que digitar o caminho completo para acessar o diretório anterior. Observem ainda que podemos “desfazer” a ação anterior, ou seja, retornar para o diretório onde estávamos anteriormente com o uso do parâmetro - (hífen).

```
darkstar@darkstar:/usr/lib$ cd /
darkstar@darkstar:/$ cd -
/usr/lib
darkstar@darkstar:/usr/lib$ _
```

Para retornarem ao diretório *\$HOME* específico do usuário (neste caso, */home/darkstar/*), utilizem o parâmetro ~ (til) ou ainda apenas *cd*.

```
darkstar@darkstar:/usr/lib$ cd ~
darkstar@darkstar~$ _
```

...ao invés de utilizar o comando e definir todo o caminho */home/darkstar/*.

VISUALIZAÇÃO

TYPE

Permite visualizar o conteúdo de arquivos-textos.

Sintaxe:

```
$ type [ARQUIVO]
```

As mesmas regras do *MS-DOS* também são aplicadas aqui: os arquivos textos serão exibidos normalmente, ao contrário dos arquivos binários.

LESS

Permite também visualizar o conteúdo de arquivos-textos, tal como *type*.

Sintaxe:

```
$ less [PARÂMETROS] [ARQUIVO]
```

Difere de *type*, o *less* possibilita realizar a paginação como a utilização das teclas <SETA_ACIMA>, <SETA_ABAIXO>, <PÁGINA_ACIMA> e <PÁGINA_ABAIXO> para rolar o texto a ser visualizado.

O recurso de paginação é muito útil para visualizar o conteúdo de arquivos com grande quantidade de textos. Existem diversos parâmetros, mas caso queiram realizar uma consulta mais refinada, experimentem digitar...

```
$ less -help
```

... e analisem os resultados exibidos.

FILE

Exibe informações gerais sobre o tipo de arquivo consultado.

Sintaxe:

```
$ file [ARQUIVO]
```

Observem atentamente os exemplos abaixo para uma melhor compreensão.

```
$ file fotografia
```

```
fotografia: JPEG image data, JFIF standard 1.01, resolution (DPI), 72 x 72
$_
```

O comando *file* detectou que o arquivo *fotografia* trata-se de uma imagem gravada no formato *JPEG*, fornecendo ainda outras informações adicionais.

```
$ file tutorial
```

```
tutorial: HTML document text
$_
```

Novamente o comando *file* detectou a existência do arquivo-texto *tutorial* informando ainda que trata-se de um documento *HTML*.

PWD

Este comando apenas mostra em qual é o diretório corrente.

```
$ pwd
```

```
/home/darkstar
```

```
$_
```

Para as definições padrão no */etc/profile* do *Slackware*, este comando é desnecessário, pois o próprio sinal de prontidão já exhibe a sua localização.

```
darkstar@darkstar:~/usr/local$ _
```

Porém existirão circunstâncias em que este, ao invés de exibir o caminho o qual se encontra, exibirá apenas o sinal *~*. Isto significa que nos encontramos no diretório raiz do usuário autenticado.

```
darkstar@darkstar:~$ pwd
```



/home/darkstar

darkstar@darkstar:~\$ _

MORE

Atua como um filtro paginador das informações exibidas no vídeo.

Sintaxe:

```
$ [COMANDO] [PARÂMETROS] | more
```

Ou...

```
$ more [ARQUIVO-TEXTO]
```

Existem diversas situações em que o uso deste comando será de bastante utilidade, mas basicamente o utilizaremos apenas para realizar uma listagem pausada. Vejam o exemplo a seguir:

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

```
total 520
drwxr-xr-x  7 root root  296 2004-05-07 22:32 AbiSuite-2.0
drwxr-xr-x  2 root root   48 2004-08-07 20:16 ImageMagick
drwxr-xr-x  5 root root  264 2004-08-07 20:16 ImageMagick-6.0.4
-rw-r--r--  1 root root  600 2004-04-23 18:59 Ssh.bin
drwxr-xr-x  2 root root  176 2003-02-11 22:41 WINGs
drwxr-xr-x  8 root root 1600 2003-02-11 22:41 WindowMaker
drwxr-xr-x  8 root root  216 2002-02-24 17:38 a2ps
drwxr-xr-x  2 root root 2208 2004-09-19 21:07 acllocal
drwxr-xr-x  2 root root  968 2004-05-21 03:38 acllocal-1.8
drwxr-xr-x  4 root root  160 2004-05-29 17:06 alsa
drwxr-xr-x  2 root root  536 2004-06-07 19:36 application-registry
drwxr-xr-x  2 root root 2320 2004-10-22 20:42 applications
drwxr-xr-x  6 root root  168 2003-01-07 20:15 apsfiler
drwxr-xr-x  2 root root 1112 2003-01-13 22:27 aspell
drwxr-xr-x  2 root root  216 2003-08-29 03:17 aumix
drwxr-xr-x  7 root root  224 2004-02-15 23:44 autoconf
drwxr-xr-x  4 root root  648 2004-05-21 03:38 automake-1.8
drwxr-xr-x  2 root root  608 2003-07-28 18:30 awk
drwxr-xr-x  2 root root   88 2004-05-13 17:27 battstat_applet
drwxr-xr-x  2 root root  112 2002-10-15 22:19 bison
drwxr-xr-x  2 root root  200 2004-05-16 20:47 blackjack
--Mais--
```

O diretório `/usr/share` possui uma infinidade de diretório em sua estrutura. Uma simples listagem iria exibir todo o seu conteúdo, porém não realizaria uma pausa para a verificação, de forma que apenas conseguiríamos ver as informações da saída de vídeo do final da operação. Para continuar com a exibição dos dados, basta apenas pressionar `<BARRA_DE_ESPAÇO>`:

```
drwxr-xr-x  2 root root  200 2004-05-16 20:47 blackjack
drwxr-xr-x  3 root root  224 2004-05-07 16:02 bug-buddy
drwxr-xr-x  2 root root   72 2004-06-08 16:57 cdrdao
drwxr-xr-x  5 root root  128 2004-05-13 03:47 control-center
drwxr-xr-x  7 root root  176 2004-06-19 21:09 control-center-2.0
```



```

drwxr-xr-x  8 root root    232 2004-09-19 20:50 cups
drwxr-xr-x  2 root root     88 2004-06-06 22:48 curl
drwxr-xr-x  3 root root     72 2004-06-09 15:28 cvs
drwxr-xr-x  4 root root    544 2004-11-15 10:24 d4x
drwxr-xr-x  2 root root     72 2003-03-13 22:00 dict
drwxr-xr-x  3 root root     72 2004-05-02 19:22 distcc
lrwxrwxrwx  1 root root      6 2004-11-01 16:27 doc -> ../doc
drwxr-xr-x  2 root root   3120 2004-06-19 20:52 eazel-engine
drwxr-xr-x  7 root root    888 2004-02-22 03:25 elvis-2.2_0
drwxr-xr-x  4 root root    104 2003-05-23 02:28 emacs
drwxr-xr-x  4 root root    936 2002-02-26 02:05 enscript
drwxr-xr-x  3 root root    104 2004-05-07 16:40 eog
drwxr-xr-x  4 root root     96 2004-02-21 23:24 epic
drwxr-xr-x  4 root root   496 2004-06-19 17:27 epiphany
drwxr-xr-x  4 root root    136 2004-06-19 18:50 epiphany-extensions
drwxr-xr-x  2 root root     48 2004-06-19 19:52 faces
drwxr-xr-x  3 root root     72 2004-05-07 17:22 file-roller
drwxr-xr-x  3 root root    104 2004-11-01 16:40 fonts
--Mais--

```

Ao pressionarmos <ENTER>, apenas será exibida a linha seguinte:

```

drwxr-xr-x  3 root root     72 2004-05-07 17:22 file-roller
drwxr-xr-x  3 root root    104 2004-11-01 16:40 fonts
drwxr-xr-x  2 root root   2272 2004-06-19 02:08 galeon
--Mais--

```

Já na leitura de um arquivo texto, ele se comportará da mesma forma:

```
$ more /mnt/cdrom/COPYRIGHT.TXT
```

```

The Linux(R) kernel is Copyright 1991, 1992, 1993, 1994, 1995,
1996, 1997, 1998 Linus Torvalds (others hold copyrights on some
of the drivers, filesystems, and other parts of the kernel) and
is licensed under the terms of the GNU General Public License.

```

```
LINUX is a registered trademark of Linus Torvalds.
```

```
(see COPYING in /usr/src/linux)
```

```
-----
```

```

Many other software packages included in Slackware are licensed under the GNU
General Public License, which is included in the file COPYING.

```

```
-----
```

```

This product includes software developed by the University of
California, Berkeley and its contributors:

```

```

Copyright (c) 1980,1983,1986,1988,1990,1991 The Regents of the University
of California. All rights reserved.

```

```
--Mais--(8%)
```

Observem a existência dos caracteres `--Mais--`, indicando a existência de mais informações após as exibidas, diferenciando-se apenas a presença de



um valor percentual na exibição do conteúdo de arquivos-textos, indicando a proporção das informações já visualizadas durante todo o processo.

Para realizarmos uma pausa durante a listagem de arquivos e diretórios, o comando *more* auxilia bastante; porém para a exibição do conteúdo de arquivos-textos, o comando *less* proporciona melhor conforto, pois diferente deste último, o comando *more* não permite que se possa retornar as informações passadas com as teclas <SETA_ACIMA> e <PÁGINA_ACIMA>.

DU

Mostra o espaço ocupado por um arquivo ou conjunto de arquivos.

Sintaxe:

```
$ du [PARÂMETROS] [ARQUIVO/DIRETÓRIO]
```

Onde:

- *-a*: mostra não só apenas o tamanho dos diretórios, como também dos arquivos encontrados;
- *-b, -k, -m*: mostra o tamanho dos arquivos e diretórios em *bytes, KB e MB* respectivamente (valor padrão: *byte*);
- *-c*: acrescenta uma linha mostrando o tamanho total de todos os arquivos e diretórios.

Existem diversos outros parâmetros úteis de acordo com as circunstâncias, mas para nós, nos interessa apenas conhecer sua utilização básica, que por sua vez mostrará apenas o tamanho do arquivo e diretórios neles aplicados:

```
$ du texto.txt
104  texto.txt
$ _
```

Neste caso, o arquivo *texto.txt* possui apenas *104 KB*. Já neste outro...

```
$ du Prova 3o. Bimestre/
149  Prova 3o. Bimestre/Português
245  Prova 3o. Bimestre/Matemática
394  Prova 3o. Bimestre/
$ _
```

... é mostrado os tamanhos de cada subdiretórios, além do total do diretório principal consultado, contando com o espaço ocupado por todos eles.



MANIPULAÇÃO

MKDIR

Este comando é utilizado unicamente para criar diretórios.

Sintaxe:

```
$ mkdir [PARÂMETROS] [DIRETÓRIO]
```

Exemplo:

```
$ mkdir /home/darkstar/teste
$ ls -l /home/darkstar/
total 3
drwx-----  3 darkstar users      128 Ago 16 00:25 Desktop/
drwxr-xr-x   26 darkstar users    1008 Ago  9 22:12 docs/
drwxr-xr-x   3 darkstar users      320 Jul 11 18:54 OpenOffice.org1.0.0/
drwxr-xr-x   2 darkstar users       48 Ago 16 20:25 teste/
drwxr-xr-x   5 darkstar users     256 Ago 16 20:16 z/
$ _
```

Para criar um conjunto de diretórios e respectivos subdiretórios diretamente com um único comando, utilizem o parâmetro *-p*.

```
$ mkdir -p /[DIRETÓRIO]/[SUBDIRETÓRIO]/
```

Exemplo:

```
$ mkdir -p teste/subteste/
$ _
```

DD

Abreviatura de *direct copy*, possui a finalidade de copiar e transferir dados utilizando as especificações de bloco de entrada e saída. Útil para realizar cópia de arquivos e transferência de dados conforme sua estrutura.

Sintaxe:

```
$ dd if=[ORIGEM] of=[DESTINO]
```

Onde estes dados serão formatados de acordo com as definições dos parâmetros de entrada (*if*) e saída (*of*).

Observem este simples exemplo para a cópia de um arquivo:

```
$ dd if=/dev/hda9 of=/dev/sda1
```

Este comando copiará todo o conteúdo da partição *hda9* (o diretório */home*) para um dispositivo de memória eletrônica (um *pendrive*).

CP

Realiza a cópia de arquivos e/ou o conteúdo de um diretório.

Sintaxe:



```
$ cp [PARÂMETROS] [ORIGEM] [DESTINO]
```

Onde:

- *-f*: sobrescreve o arquivo, caso já exista no local de destino;
- *-p*: preserva os atributos de permissões, usuários e grupos de acesso originais do arquivo copiado;
- *-r*: realiza a cópia de diretórios recursivamente.

Exemplo:

```
$ cp /home/darkstar/docs/texto.txt /mnt/floppy
```

O exemplo acima copia o arquivo *texto.txt* para o ponto de montagem */mnt/floppy* (um disquete que deverá estar previamente montado).

MV

Abreviatura de *move*, movimenta o(s) arquivo(s) e/ou diretório(s) para o local desejado ou ainda, os renomeia conforme desejado.

Sintaxe:

```
$ mv [PARÂMETROS] [ORIGEM] [DESTINO]
```

Exemplo:

```
$ mv /home/darkstar/teste/* /home/darkstar/
```

O comando *mv* também pode ser utilizado para renomear diretórios...

```
$ mv /home/darkstar/teste/ /home/darkstar/ok
```

... e arquivos...

```
$ mv /home/darkstar/rasura /home/darkstar/texto.txt
```

LN

Cria atalhos para apontar determinados arquivos ou diretórios do sistema.

Sintaxe:

```
$ ln [PARÂMETROS] [ORIGEM] [DESTINO]
```

Onde:

- *-s*: atalho simbólico;
- *-d*: atalho físico (somente disponível para o superusuário).

A diferença entre atalhos simbólicos para atalhos físicos (ou *hardlinks*) está na manipulação direta do mesmo. O atalho simbólico apenas fornece um caminho apontado por ele; já o atalho físico faz uma referência direta, sendo perfeitamente idêntico ao arquivo original em tamanho e permissões de acesso. Sua única limitação está no fato de não fazer referências aos diretórios ou arquivos de outras partições.

Uma das principais vantagens da utilização do atalho é a possibilidade de

—B

fazer referências a determinados arquivos, ao invés de realizar outras intervenções mais elaboradas. Por exemplo, na necessidade de dispor do navegador *firefox* para todos os usuários, basta simplesmente...

```
# ln -s /usr/local/firefox/firefox /usr/bin/firefox
```

... ao invés de atualizar a variável *\$PATH* para o caminho do *Firefox*.

EDITORÇÃO

MCEDIT

Componente indispensável das ferramentas *GNU Midnight Commander*, o *MCedit* é um excelente editor de textos *ASCII*, com uma aparência muito similar ao já conhecido *Edit*, do *MS-DOS*.

```
COPYING [----] 0 L:[_1+0_1/348] *(0 /18454b)= . 9 0x09
GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
 59 Temple Place, Suite 330, Boston, MA 02111-1307, USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

The Free Software Foundation has exempted Bash from the requirement of
Paragraph 2c of the General Public License. This is to say, there is
no requirement for Bash to print a notice when it is started
interactively in the usual way. We made this exception because users
and standards expect shells not to print such messages. This
exception applies to any program that serves as a shell and that is
based primarily on Bash as opposed to other GNU software.

Preamble

The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users. This
1aJuda 2Salvan 3Mancar 4Substit 5Copiar 6Mover 7Procura 8Excluir 9Levar p10Sair x
```

Dentre seus comandos, para acionar o menu principal, basta teclarmos *<F9>*; para que o programaseja encerrado, basta pressionarem *<F10>*.

Para os iniciantes, recomendamos a utilização deste editor de textos. Mas como estamos num mundo livre, fica em aberto a livre-escolha!

EXCLUSÃO

RMDIR

Remover um diretório já existente. O uso de parâmetros é opcional.

Sintaxe:

```
$ rmdir [DIRETÓRIO]
```

Exemplo:

```
$ rmdir /home/darkstar/teste/
```



Porém este comando não pode remover nenhum diretório que não esteja vazio. Veja o exemplo abaixo:

```
$ rmdir /home/darkstar/teste/  
rmdir: `/home/darkstar/teste': Diretório não vazio  
$ _
```

Para estas circunstâncias, o comando *rm* possui o parâmetro *-r* (recursivo), que apaga todos os diretórios e seus respectivos arquivos. Veja à seguir.

RM

Remove um arquivo já existente.

Onde:

- *-i*: exclusão interativa (permite definir se o arquivo em questão vai ser excluído ou não);
- *-r*: remoção recursiva (utilizado para excluir diretório não vazios).

Sintaxe:

```
$ rmdir [PARÂMETROS] [NOME_DO_ARQUIVO]
```

Exemplo:

```
$ rm /home/darkstar/texto.txt
```

Ao estudarmos o comando *rmdir*, vimos que ele não pode remover diretórios que possuem conteúdo. Para esta situação, podemos utilizar o comando *rm* junto com o parâmetro *-r* para resolver este problema.

```
$ rmdir teste/  
rmdir: `teste/': Diretório não vazio  
$ rm -r teste/  
$ _
```

CÓPIAS DE SEGURANÇA E COMPACTAÇÃO

A cópia de segurança é uma das operações essenciais para a manutenção dos dados e informações contidas nas unidades de armazenamento. Basicamente é dividida em 2 etapas: arquivamento e compactação.

ARQUIVAMENTO

O ato de arquivar consiste basicamente em reunir um conjunto de arquivos, diretórios ou ainda, uma estrutura de arquivos e diretórios, em apenas um único arquivo. Para esta operação, temos um ótimo utilitário de linha de comando, chamado *TAR*.



TAR

O *TAR* - *Type ARchive* - é um eficiente arquivador de dados.

Sintaxe:

```
$ tar [PARÂMETROS] [ARQUIVO_OU_DIRETÓRIO_OU ESTRUTURA]
```

Pelo fato de possuir vastos recursos, existe uma imensidade de parâmetros relacionados. Descreveremos aqui apenas os mais utilizados:

- *-v*: exibe o andamento da operação no vídeo;
- *-f*: tratamento de arquivo, *device ARQ*;
- *-w*: solicita a confirmação para cada operação a realizar;
- *-M*: criação / listagem / extração de múltiplos volumes.

Além dos parâmetros gerais, encontram-se também outros para o arquivamento propriamente dito. Segue a listagem abaixo:

- *-c*: criação de um novo arquivo (o destino);
- *-r*: acrescenta arquivos a um pacote já criado.

Com o *TAR* podemos também visualizar o conteúdo de pacotes gerados, além de realizar alguns processos de comparação:

- *-t*: lista o conteúdo de um arquivo gerado;
- *-d*: compara o arquivo gerado com os arquivos atuais.

Ainda poderemos utilizar parâmetros específicos para compactar ou descompactar os pacotes:

- *-j*: compressão / descompressão com *Bzip2*;
- *-z*: compressão / descompressão com *GZIP*.

Por último, também existem alguns parâmetros extras para facilitar as atividades de extração dos pacotes criados, tendo destaque a descompactação e extração simultânea:

- *-x*: extrai os dados arquivados;
- *-p*: mantém as permissões originais dos dados arquivados.

Exemplos:

Para empacotar toda uma estrutura de arquivos de diretório...

```
$ tar -cvf BACKUP.tar *
```

Para anexar o arquivo *TEXTO.txt* ao pacote *BACKUP.tar...*

```
$ tar -rf TEXTO.txt BACKUP.tar
```

Para desempacotar o pacote *BACKUP.tar...*

```
$ tar -xvf BACKUP.tar
```

Será descomprimir e desempacotar o pacote *BACKUP.tar.gz...*



```
$ tar -xzf BACKUP.tar.gz
```

Será descompactar e desempacotar o pacote *BACKUP.tar.bz...*

```
$ tar -xjf BACKUP.tar.bz
```

Apesar de ser apenas um empacotador, o *TAR* também pode gerar pacotes compactados com o auxílio dos compactadores *gzip* e *bzip2*, conforme demonstramos acima com os dois últimos exemplos, onde utilizamos os parâmetros *z* (zê) e *j* (jota) para descompactar volumes compactados com o *GZIP* e o *BZIP2*. (pacotes *BACKUP.tar.gz* e *BACKUP.tar.bz2*).

Já os utilitários de compactação descritos apenas geram um único arquivo por vez, sendo necessário a utilização dos empacotadores para criar um único arquivo compactado com uma estrutura de arquivos e diretórios.

Por último, em algumas circunstâncias serão necessárias ferramentas como *cat* e *split* para o manuseio de arquivos e pacotes com grandes volumes.

COMPACTAÇÃO

A compactação de arquivos nos sistemas *GNU/Linux* é feita tradicionalmente através dos seus próprios utilitários de linha de comando disponíveis na distribuição, não necessitando de quaisquer outro para as funcionalidades básicas. Os arquivos ou estrutura de arquivos e diretórios também podem ser manipulados por outros utilitários gráficos, conforme o interesse do usuário. Existem diversos compactadores para o sistema, mas os principais utilizados são *gzip*, *bzip2* e - em alguns casos - *zip*.

GZIP / GUNZIP

Atualmente é o compactador mais utilizado entre os disponíveis.

Sintaxe:

```
$ gzip [PARÂMETROS] [ARQUIVO_ORIGEM]
```

Onde:

- *-c*: mantém o arquivo original (não apaga);
- *-d*: descompacta o arquivo (o mesmo que utilizar apenas *gunzip*);
- *-l*: listagem de conteúdo do arquivo compactado;
- *-v*: exibe as informações do processo;
- *- (1 até 9)*: variação da taxa de compressão, onde *1* = compressão rápida (baixa) e *9* = compressão lenta (alta).

Para simplesmente compactar um arquivo...

```
$ gzip [ARQUIVO]
```

Para compactar um arquivo com alta taxa de compressão...



```
$ gzip -9 [ARQUIVO]
```

Para descompactar um arquivo comprimido.

```
$ gzip -d [ARQUIVO]
```

... ou...

```
$ gunzip [ARQUIVO]
```

BZIP2 / BUNZIP2

O 2o. compactador mais utilizado pelos *linuxers*, já que possibilita uma boa vantagem em comparação *gzip*: consegue obter taxas maiores de compressão, conseguindo ganhos de espaço em até 20%. Porém, exige uma maior carga de processamento, requerendo maior capacidade de processamento para reduzir o tempo gasto nesta operação.

Sintaxe:

```
$ bzip2 [PARÂMETROS] [ARQUIVO]
```

Onde:

- *-d*: descompacta o arquivo (o mesmo que utilizar apenas *bunzip2*);
- *-f*: força o modo sobre-escrita (grava sobre um arquivo existente);
- *-s*: reduz o consumo de memória durante a compactação (ideal para máquinas com pouca memória);
- *-v*: exibe o andamento da operação no vídeo.

Para realizar a compactação de um arquivo, bastará...

```
$ bzip2 [ARQUIVOS]
```

Para descompactar...

```
$ bunzip2 [ARQUIVO].bz2
```

ZIP / UNZIP

Os comandos *zip* e *unzip* são respectivamente compactador e descompactador de volumes para o uso do formato *ZIP* (.zip).

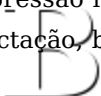
Sintaxe:

```
$ zip [PARÂMETROS] [DESTINO] [ORIGEM]
```

Onde:

- *-e*: permite a utilização de senha de proteção, esta solicitada no ato da compactação / descompactação;
- *-r*: compacta arquivos e diretórios de modo recursivo;
- *-(1 até 9)*: variação da taxa de compressão, onde 1 = compressão rápida (baixa) e 9 = compressão lenta (alta);

Para realizar uma simples compactação, basta utilizar.



```
$ zip BACKUP.tar NOVO-BACKUP.zip
```

... ou...

```
$ zip -r SOURCE PROGRAMA.zip
```

Para descompactar um arquivo para um diretório específico...

```
$ unzip -d /tmp BACKUP.zip
```

UTILITÁRIOS

SPLIT

Divide um arquivo em várias partes.

Sintaxe:

```
$ split [PARÂMETROS] [TAMANHO][UNIDADE_DE_MEDIDA] [ARQUIVO]
```

Onde:

- *-b*: define a unidade de medida - *byte (b)*, *KB (kb)* e *MB (mb)*.

Exemplo:

```
$ split -b 1440kb BACKUP.tar.gz
```

Serão gerados diversos arquivos com o tamanho especificado com a nomenclatura *xaa*, *xab*, *xac*, etc. No exemplo acima citado, o tamanho definido é ideal para gravá-los em disquetes.

Para reuni-los novamente em um só arquivo, utilizem o comando *cat*.

CAT

O comando *cat* possui diversas opções e funcionalidades, como poderemos ver ao checar sua documentação. Mas para operações básicas, ele é bastante utilizado em operações de visualização e concatenação.

Sintaxe:

```
$ cat [OPTION] [ARQUIVO]
```

Para realizar uma visualização do arquivo *texto.txt*, basta utilizar...

```
$ cat texto.txt
```

Para realizar uma concatenação, basta utilizar a seguinte sintaxe:

```
$ cat [TEXT01] [TEXT02] [TEXT03] > [ARQUIVO_TEXTO_FINAL]
```

Um detalhe importante que deverá ser observado está na ordem correta dos arquivos a serem concatenados.

Segue um exemplo simples e prático:

```
$ cat texto.txt mais_info.txt > texto1.txt
```

Para mesclar arquivos gerados pelo *split* (veja seção anterior)...



```
$ cat xaa xab xac > BACKUP.tar.gz
```

Observação: não devemos utilizar o mesmo arquivo para receber o conteúdo dos arquivos mesclados; teremos o risco de sofrer ocorrências imprevisíveis, onde a perda dos dados é certamente indesejada.

CONCLUSÃO

Existe uma infinidade de possíveis operações onde é necessária a manipulação direta de arquivos e diretórios. As operações descritas neste capítulo são apenas as mais comuns, necessárias para a grande maioria dos usuários em *desktops*. Caso queiram se aprofundar, consultem as páginas de manual disponíveis na distribuição. &;-D

B

V. UNIDADES, PARTIÇÕES E FORMATOS

INTRODUÇÃO

Como já sabemos, todos os sistemas operacionais alocam suas informações em sistemas de armazenamento de dados de vários tipos (unidades), que podem ser subdivididos em várias partes (partições) e utilizarem um método de escrita específico (sistema de arquivos).

Neste capítulo iremos conhecer as particularidades das distribuições *GNU/Linux* quanto a este aspectos, além de obter instruções para a manipulação através das ferramentas disponíveis na linha de comando.

AS UNIDADES E AS PARTIÇÕES

As unidades, são como o próprio nome diz, dispositivos físicos especiais utilizados para o armazenamento de dados. Já as partições são divisões criadas nos dispositivos que podem ser utilizadas para diversas finalidades.

Entre estas finalidades, a mais importante é a organização e otimização dos dados nela arquivados e conseqüentemente do sistema como um todo. Por exemplo, um disco rígido poderá ter diferentes tipos de partições e com isto poderemos até mesmo alocar diferentes sistemas operacionais em uma única unidade, conforme veremos mais adiante.

Existem diversos tipos de unidades, das quais as principais são os disquetes, as memórias eletrônicas, os discos rígidos e a unidade leitoras de mídia óptica (*CDs* e *DVDs*). Além destas, existem outras menos comuns, como o *Zip-drive*, o gravador de mídia óptica, entre muitos outros. Todas elas são suportadas e acessíveis pelos sistemas *GNU/Linux*, mas isto no momento não é o mais importante. O que deveremos realmente dar importância são os sistemas de arquivos utilizados nelas.

OS FORMATOS

Sistemas de arquivos são métodos (formatos) de representação utilizados pelo sistema operacional para a organização dos arquivos (dados) em um determinado meio de armazenamento. Ao realizarmos a formatação de uma unidade qualquer (seja disco rígido, disquete, *zips*, etc.), estaremos condicionando a sua estrutura para que esteja pronto para receber dados.

TIPOS DE SISTEMAS DE ARQUIVOS

Cada sistema operacional normalmente suporta um pequeno conjunto de formatos específicos. Os sistemas *GNU/Linux* suportam uma infinidade de



tipos de sistemas de arquivos, onde as mais importantes são:

O SWAP

O *SWAP* é um sistema de arquivos utilizado em uma partição especial chamada *SWAP*, que por sua vez é um espaço do disco rígido reservado para o uso do sistema operacional como “*complemento*” da memória *RAM*.

Quando a memória principal do sistema operacional está completamente “*cheia*” e existe a necessidade de executar alguma tarefa que exija mais consumo de memória, as informações que estão contida na memória principal são gravadas nesta partição em separado enquanto o sistema carrega para a memória principal as informações requeridas por esta tarefa. Assim que é encerrada, a memória principal é “*esvaziada*” e novamente recarregada com as informações contidas na partição *SWAP*.

O EXT2 E EXT3

O sistema de arquivos *ext2* já foi o padrão dos sistemas *GNU/Linux*. Possui muitas similaridades em comparação ao sistema *FAT32* utilizado no *Windows*, como o suporte a nomenclatura de arquivos com 256 caracteres e tamanho de *clusters* fixo em 4 KB, além da limitar o tamanho de arquivos para 4 GB. Dentre outras características, possibilita atribuir *flags* especiais aos arquivos criados neste sistema. Estes atributos são a leitura, a escrita e a gravação. Somente iremos operar nestes arquivos de acordo com os atributos definidos pelo usuário que o criou (dono) ou o administrador.

No sistema de arquivos *ext2*, foram implementadas melhorias que resultaram no surgimento do seu substituto: o sistema de arquivos *ext3*, que por sua vez não difere muito estruturalmente do sistema de arquivos *ext2*. Eis porque a conversão de partições do formato *ext2* para o formato *ext3* pode ser feita sem dificuldades, desde que o processo seja feito de forma correta e com as ferramentas (nativas) adequadas. Em todo o caso, cópias de segurança são recomendáveis nestas circunstâncias.

O sistema de arquivos *ext3* tem como acréscimo o recurso *journaling*: este - diferente do formato *ReiserFS* - se caracteriza por manter arquivadas no disco rígido informações do estado deste sistema de arquivos atualizada constantemente. Este arquivo, localizado na raiz da partição em uso, é chamado de *journal*. Com o uso do *journaling*, toda vez que houver algum erro ou falha abrupta no sistema que necessite de uma reinicialização, o estado do sistema de arquivos é automaticamente restaurado baseando-se apenas em suas informações. Na prática, isto substitui a necessidade da utilização de ferramentas especiais para a sua manutenção como o *fsck*, em que ao realizar a checagem da integridade do sistema de arquivos, consome muito tempo e indisponibiliza o sistema por longos intervalos.

A vantagem do *Ext3* em comparação com o *ReiserFS* está na possibilidade de realizar a manutenção dos dados gravados no exato momento da queda



do sistema, pois devido as características do *journaling*, as possibilidades de recuperação destes arquivos são infinitamente maiores. Porém, é praticamente certa a degradação de desempenho por causa das constantes gravações em seus registros (*logs*). Além disso, existe a possibilidade do próprio *journal* se corromper na queda do sistema, resultando na utilização do demorado processo de recuperação com o *fsck*.

REISERFS

✓ <<http://www.namesys.com/>>.

Criado pela empresa *Namesys*, o sistema de arquivos *ReiserFS* foi desenvolvido visando adequar-se ao conceito de *Alta Disponibilidade*.

Por exemplo, quando ocorre uma queda do sistema resultante de falhas ou falta de energia, ao reinicializá-lo, será feita na partição *ext2* (e em alguns casos, o *ext3*), a verificação de sua integridade (de forma automática) pelo programa *e2fsck*. Mas dependendo do tamanho e da quantidade de partições, este processo pode requerer um tempo considerável, o que é indesejável devido a necessidade de *Alta Disponibilidade*. No sistema de arquivos *ReiserFS*, ao invés de ser feita a checagem total, ele apenas consulta o arquivo *journal*, onde o mesmo apenas informa as ocorrências inesperadas para que seja feita a restauração.

Dentre outras características importante do *ReiserFS* está nos seus pequenos *clusters*, que possuem tamanho máximo de *512 bytes*, o que por sua vez é o ideal para a utilização em armazenamento de inúmeros arquivos de pequeno volume, acarretando assim pouca perda de espaço. Além disso, não há a limitação de *2 GB* para arquivos que ultrapassem este tamanho.⁶

A principal diferença do formato *ReiserFS* em comparação ao *ext3* está exatamente no funcionamento do recurso *journaling*, conforme dito anteriormente. No *ReiserFs*, ele apenas armazena informações sobre o espaço dos arquivos e permissões, ao passo que no *Ext3*, além dele executar estas funções ainda salvaguarda o próprio conteúdo dos arquivos afetados durante uma queda do sistema.

Em vista disso, a grande vantagem do *ReiserFS* está na facilidade de recuperar a consistência do sistema de arquivos em um tempo mínimo (décimos de segundos). Praticamente torna-se inexistente a possibilidade de uma pane em alguma pasta ou até mesmo nas partições do disco rígido. Em contrapartida, caso o sistema esteja sofrendo gravações de dados no exato momento da queda, estas arquivos infelizmente não poderão ser recuperados, pois seu conteúdo estará truncado ou incompleto.

⁶ Isto torna este sistema de arquivos uma excelente opção para trabalharmos com geração de imagens e gravação de *DVD-RW*, o que não quer dizer que não seja possível realizar estas atividades nos sistemas *ext2* e *ext3*.

MSDOS, FAT32 E NTFS

Estes três são os tradicionais formatos de sistema de arquivos do *MS-DOS* e *Windows*. Apesar de não serem utilizados pelos sistemas *GNU/Linux* e em virtude de sua popularização, estes sistemas de arquivos são plenamente bem suportados por praticamente todas as distribuições existentes.

Na utilização de disquetes e dispositivos de memória eletrônica (*flash memory*), é recomendável a utilização destes formatos para que os mesmos possam ser lidos em outros computadores providos do sistema operacional *Windows*. Em especial, o sistema de arquivos *FAT32* possibilita visualizar os arquivos com nomes longos (mais de 8.3 caracteres), o qual é recomendada a sua utilização no acesso a estes dispositivos.

Já o *NTFS*, infelizmente ainda não há suporte maduro, visto que a *Microsoft* não libera (e provavelmente nem irá liberar) as especificações deste sistema de arquivos para que os desenvolvedores do *kernel* possam escrever *drivers* adequados para a sua leitura e escrita. Porém, existem projetos interessantes (embora em estágio experimental) que possibilita a realização destas operações, como o *NTFS-NG* e o *Captive*.

ISO9660

O sistema de arquivos *ISO9660* é somente utilizado para os *CD-ROMs*, devido a natureza de seu armazenamento de dados permanente, impossibilitando o sistema operacional de definir um sistema de arquivos.

A imagem *ISO* nada mais é um arquivo especial que contém as informações sobre todos os arquivos que serão gravados em uma mídia de *CD-R/RW*, utilizando-se o formato *ISO9660*.

OPERAÇÕES E ATIVIDADES AFINS

TRABALHANDO COM PARTIÇÕES E UNIDADES

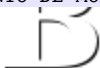
Existem uma infinidade de operações que poderão ser realizadas nas unidades e partições, mas antes, será necessário obter acesso a estes dispositivos para realizar as operações mais cotidianas...

MOUNT / UMOUNT

Os comandos *mount* e *umount* são utilizados respectivamente para “montar” (ter acesso) e “desmontar” (retirar acesso) unidades e partições.

Sintaxe:

```
$ mount [OPÇÕES] [DISPOSITIVO] [PONTO DE MONTAGEM]
```



Onde:

- *OPÇÕES*: parâmetros a serem definidos;
- *DISPOSITIVO*: device da unidade ou partição;
- *PONTO DE MONTAGEM*: diretório de acesso.

Dentre os principais parâmetros existentes, destaca-se:

- *-a (auto)*: montagem automática;
- *-f (force)*: montagem de modo forçado;
- *-r (read-only)*: permissão somente para leitura;
- *-t (type)*: pré-define o sistema de arquivos o qual a partição se encontra para ser montado (*ext2, ext3, iso9660, reiserfs, vfat...*);
- *-v (verbose)*: exhibe informações adicionais do processo;
- *-w (write)*: permissão para escrita.

Segue alguns exemplos para melhor ilustrar:

```
$ mount /dev/hda5 /mnt/hd
$ mount -t vfat /dev/hda1 /mnt/win
```

Os disquetes são acessados tradicionalmente utilizando seus respectivos *devices*: */dev/fd0* para o *drive A*: e */dev/fd1* par o *drive B*:. Para acessá-los, bastará utilizar a seguinte sintaxe...

```
$ mount /dev/fd0 /mnt/floppy
```

... onde o ponto de montagem */mnt/floppy* poderá ser omitido, caso a unidade em questão já esteja especificada em */etc/fstab*.

Tanto para os disquetes como quaisquer outras unidades que utilizam o sistema *VFAT*, deveremos incrementar este comando o parâmetro *-t vfat*:

```
$ mount -t vfat /dev/fd0 /mnt/floppy
```

Já os *CDs* e *DVDs* são acessados com a utilização do comando...

```
$ mount /dev/cdrom /mnt/cdrom
```

..., onde */mnt/cdrom* poderá ser omitida caso a unidade esteja especificado em */etc/fstab*. Caso contrário...

```
$ mount -t iso9660 /dev/cdrom /mnt/cdrom
```

Da mesma forma que no disquete, será desnecessária a utilização do parâmetro *-t iso9660* se ele estiver previamente especificado em */etc/fstab*.

Lembre-se de que */dev/cdrom* é apenas um atalho apontado para o verdadeiro dispositivo (*CDs* e *DVDs*), onde provavelmente serão */dev/hdb* ou */dev/hdc*, de acordo com as configurações da máquina em uso.

Para montar uma unidade de memória eletrônica (ou memória *flash*), deveremos recorrer ao *device* responsável pela emulação *SCSI*.

```
$ mount /dev/sda1 /mnt/[PONTO_DE_MONTAGEM]
```

Como não existem diretórios pré-definidos para a memória *flash*, poderemos criar um específico (p. ex.: */mnt/flash*).

Enfim, para desmontar quaisquer dessas unidades...

```
$ umount /dev/[UNIDADE]
```

... OU...

```
$ umount [PONTO_DE_MONTAGEM]
```

Um detalhe importante é que o comando *umount* checa se os dados a serem gravados nas unidades a ser desmontadas foram realizados, para depois efetivar a desmontagem propriamente dita dos dispositivos.

SYNC

Realiza toda a transferência de dados da *cache* do sistema (arquivos e diretórios) para a unidade montada (disquete, memória eletrônica, CD/DVD-ROM, etc.), para que possamos desmontar a unidade imediatamente.

```
$ sync
```

É muito útil em circunstâncias em que não sabemos porque o dispositivo não desmonta, mesmo que todas as operações estejam concluídas.

FORMATAÇÃO E DEFINIÇÃO DO SISTEMA DE ARQUIVOS

Os respectivos utilitários que utilizaremos em linha de comando para estas atividades são *mkfs*, *mkreiserfs* e *mkswap*.

MKFS

O *mkfs* é o utilitário usado para criar um sistema de arquivos.

Sintaxe:

```
# mkfs.ext2 [PARÂMETROS] /dev/[PARTIÇÃO]
```

Onde:

- *-b*: definição do tamanho do bloco (*cluster*);
- *-c*: checagem de blocos danificados;
- *-L [NOME]*: define um nome para o sistema de arquivos.

O *mkfs* possui “*extensões*”, das quais cada uma possui a finalidade de criar sistemas de arquivos específicos: o *mkfs.ext2*, *mkfs.ext3* e *mkfs.msdos*. Estas opções poderão ser omitidas, desde que utilizemos o parâmetro *-t*, acompanhado do sistema que se queira criar (*msdos*, *ext2*, *ext3*, etc.).

```
# mkfs -t [SIST._ARQUIVOS] [PARÂMETROS] /dev/[PARTIÇÃO]
```

Observem também que o *mkfs* não suporta o sistema de arquivos *ReiserFS*, sendo necessário então utilizar outra ferramenta, o *mkreiserfs*.

MKREISERFS

Conforme dito na seção anterior, o utilitário *mkreiserfs* é utilizado para definir um sistema de arquivos *ReiserFS* em uma partição.

Sintaxe:

```
# mkreiserfs [PARÂMETROS] /dev/[PARTIÇÃO]
```

Onde:

- *-b*: definição do tamanho do bloco (*cluster*);
- *-L [NOME]*: define um nome para o sistema de arquivos;
- *-s [VALOR]*: define o tamanho do arquivo de *journal* em blocos.

MKSWAP / SWAPON

Formata e ativa uma partição *SWAP*.

Sua utilização é simples, bastando digitar na linha de comando...

```
# mkswap /dev/[PARTIÇÃO]
```

Para ativá-la ao sistema, deveremos utilizar...

```
# swapon /dev/[PARTIÇÃO]
```

VERIFICANDO AS PARTIÇÕES E UNIDADES DO SISTEMA

Existem diversas ferramentas para a checagem das partições e unidades do sistema, dentre as quais, as principais usadas são:

DF

Verifica o espaço ocupado das unidades montadas.

Sintaxe:

```
$ df [PARÂMETROS]
```

Onde:

- *-h*: exibe as dimensões em *MB (hm)* e *GB (h)*;
- *-T*: exibe o sistema de arquivos utilizado.

Ao utilizar somente o *df*, teremos apenas esta simples avaliação:

```
# df
Filesystem          1k-blocks      Used Available Use% Mounted on
/dev/hda5            2104408        339684   1764724   17% /
/dev/hda6            7341440       1841004   5500436   26% /usr
/dev/hda7            2104408         51984   2052424    3% /var
/dev/hda8            1052184         39240   1012944    4% /tmp
/dev/hda9            1052184        177076    875108   17% /home
/dev/hda10           24879804       282344  24597460    2% /usr/pkg
# _
```



Definindo o parâmetro `-T`, veremos o sistema de arquivos utilizado...

```
# df -T
Filesystem      Type      1k-blocks      Used Available Use% Mounted on
/dev/hda5 reiserfs    2104408        339808   1764600   17% /
/dev/hda6 reiserfs    7341440       1841004   5500436   26% /usr
/dev/hda7 reiserfs    2104408        51984    2052424    3% /var
/dev/hda8 reiserfs    1052184        39240    1012944    4% /tmp
/dev/hda9 reiserfs    1052184       177076    875108    17% /home
/dev/hda10 reiserfs   24879804       282344   24597460    2% /usr/pkg
# _
```

Com a utilização do parâmetro `-h`, teremos as informações desejadas em GB.

```
# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/hda5      2.0G  332M  1.6G   17% /
/dev/hda6      7.0G  1.8G  5.2G   26% /usr
/dev/hda7      2.0G   51M  1.9G    3% /var
/dev/hda8      1.0G   39M  989M    4% /tmp
/dev/hda9      1.0G  173M  854M   17% /home
/dev/hda10     23G  276M  22G    2% /usr/pkg
# _
```

Os parâmetros também podem ser utilizados de forma combinada.

BADBLOCKS

Utilizado para checar blocos defeituosos na unidade de disco rígido.

Sintaxe:

```
# badblocks [PARÂMETRO] /dev/[PARTIÇÃO]
```

Onde:

- `-b`: definição do tamanho do bloco a ser checado (*cluster*);
- `-v`: mostra as operações em andamento.

Para realizarem uma checagem básica, utilizem...

```
# badblocks -b 4096 /dev/[PARTIÇÃO]
```

Normalmente os tamanhos de blocos utilizados pelos atuais sistemas de arquivos são de *4.096 bytes*.

FSCK

Realiza uma checagem da unidade em questão, procurando por áreas e blocos danificados que porventura possam existir no disco rígido.

Sintaxe:

```
# fsck.[FS] [PARÂMETRO] /dev/[PARTIÇÃO]
```

Onde:

- `[FS]`: define o sistema de arquivos (*ext2, ext3, xiafs, minix, etc.*);



- *-a*: modo automático, pois realiza as operações sem realizar qualquer pergunta (inverso de *-r*);⁷
- *-c*: verifica os blocos defeituosos e atualiza a tabela da unidade, marcando-os como inválidos;
- *-f*: realiza a checagem em modo forçado;
- *-r*: modo interativo, pois realiza algumas perguntas antes de efetuar as operações (inverso de *-a*);
- *-t*: define o sistema de arquivos utilizado;
- *-v*: visualiza as operações em execução;
- *-y*: requer confirmação para aceitar todas as perguntas realizadas no modo interativo.

Para uma simples e básica checagem em uma partição *ext3*:

```
# fsck.ext3 /dev/[PARTIÇÃO]
```

Para uma partição *ext2* também poderemos utilizar...

```
# e2fsck /dev/[PARTIÇÃO]
```

Onde *e2fsck* é um apelido para *fsck.ext2*.

Em um disquete formatado para *DOS* basta utilizar...

```
# fsck -t msdos /dev/fd0
```

É importante observar que as unidades avaliadas deverão estar desmontadas. Além disso, o *fsck* somente funciona em sistemas de arquivos *ext2* e *ext3*, não sendo útil em partições *ReiserFS* e *VFAT*.

REISERFS

Também realiza uma checagem da unidade em questão, procurando por áreas e blocos danificados que porventura possam existir na unidade de disco rígido; sendo que esta ferramenta foi desenvolvida para ser utilizada unicamente em sistemas de arquivos *ReiserFS*.

Sintaxe:

```
# reiserfs [PARÂMETRO] /dev/[PARTIÇÃO]
```

Onde:

- *-a*: exibe detalhes sobre o sistema de arquivos;
- *-j*: especifica um arquivo *journaling* alternativo para ser utilizado;
- *-q*: não exibe nenhuma mensagem de *status*.

Para uma simples e básica checagem, utilizem...

```
# reiserfs /dev/[PARTIÇÃO]
```

⁷ Para o *ext2*, deveremos utilizar a opção *-p* para obter a mesma funcionalidade, visto que a opção *-a* somente encontra-se por questões de compatibilidade.

Lembrem-se: as partições devem estar previamente desmontadas.

REALIZANDO A TRANSFERÊNCIA DE DADOS

DD

Além da cópia direta de arquivos e dados, o comando *dd* também poderá ser utilizado para criar cópias de dados das partições desejadas.

Para obterem maiores informações, consultem nesta parte o capítulo *Manipulação de arquivos e diretórios*.

OPERAÇÕES COM DISQUETES

Em virtude da imensa utilização desta unidades de armazenamento, resolvemos descrever um conjunto de instruções para facilitar ao máximo sua utilização em sistemas *GNU/Linux*.

UTILIZAÇÃO

No *MS-DOS*, o processo de utilização de disquetes é algo relativamente simples, onde basta apenas inserir a mídia no *drive* leitor e acessar diretamente o dispositivo. Em sistemas *GNU/Linux*, este dispositivo deverá ser antes montado como qualquer outro, porém lembre-se que as unidades são referenciadas de forma diferente.

FORMATAÇÃO

Diferente dos sistemas da *Microsoft*, onde seus sistemas operacionais realizam uma única operação - formatação - para preparar os disquetes para o uso, nos sistemas *GNU/Linux*, por padrão estes processos se dividem em 2 fases distintas:

- Formatação de baixo nível;
- Criação do sistema de arquivos.

Na formatação de baixo nível serão recriados todos o setores e trilhas para posteriormente ser definido um sistema de arquivo. Este processo é de suma importância, pois só assim teremos certeza de que armazenaremos nossos dados em uma unidade isenta de defeitos, pois a mídia de armazenamento magnética é muito sensível.

```
# fdformat /dev/fd0H1440
```

Observe que para realizar a formatação de um disquete, o mesmo não poderá estar montado previamente no sistema.

Após a baixa formatação, teremos então a possibilidade de definir o sistema de arquivos que desejarmos, diferente dos utilitários do *MS-DOS* que nos permite apenas a utilização de um único sistema de arquivos.

Dentre os formatos suportados, destacam-se o *msdos* e o *ext2*.

Para criar um sistema de arquivos com as opções desejadas, basta utilizar o comando *mkfs* e definir o formato desejado:

```
$ mkfs.msdos /dev/fd0
```

... ou...

```
$ mkfs -t ext2 /dev/fd0
```

Outro bom utilitário para esta atividade é o *Superformat*, que por sua vez já realiza a formatação e define o sistema de arquivos da unidade em questão.

Sintaxe:

```
$ superformat [PARÂMETROS] /dev/[UNIDADE]
```

Segue um exemplo simples e básico:

```
$ superformat /dev/fd0
Measuring drive 0's raw capacity
warmup cycle: 7 200150 200148
```

Não irá demorar para que todo o processo seja concluído.

```
Measuring drive 0's raw capacity
In order to avoid this time consuming measurement in the future,
add the following line to /etc/driveprm:
drive0: deviation=720
CAUTION: The line is drive and controller specific, so it should be
removed before installing a new drive 0 or floppy controller.
```

```
Verifying cylinder 79, head 1
mformat -s18 -t80 -h2 -S2 -M512 a:
```

```
$ _
```

Este comando utiliza apenas seus parâmetros padrões.

OPERAÇÕES COM OS GRAVADORES DE CD/DVD

Os seguintes utilitários são essenciais para a manipulação da unidade de *CD-R/CD-RW* do sistema:

MKISOFS

O *mkisofs* - abreviação de *mk* (*make* = fazer) *iso* (imagem *ISO*) e *fs* (*filesystem* = sistema de arquivos) - permite construir imagens *ISO* à partir de dados disponíveis de uma unidade.

Sintaxe:

```
$ mkisofs [PARÂMETROS] -o [IMAGEM].iso [DIRETÓRIO_ORIGEM]
```

Onde:

- *-C*: para a criação de imagens com múltiplas sessões;



- *-J*: habilita as extensões *Joilet*, requerimento indispensável para manter compatibilidade com os demais sistemas operacionais (como o *Windows*, por exemplo);
- *-l*: habilita o suporte aos longos nomes de até 31 caracteres, pois o padrão *ISO9660* suporta apenas o formato 8.3, que por sua vez é compatível com o *MS-DOS*;
- *-L*: permite a gravação de arquivos ocultos, ou seja, os que iniciam com a utilização do ponto (.);
- *-o*: especifica o nome da imagem a ser gerada (*output*);
- *-r*: abreviação de *Rock Ridge*, um protocolo que evita a truncagem dos nomes de arquivos longos para a gravação de mídias;
- *-v*: mostra todo o andamento do processo em execução;
- *-V*: define o nome do volume (*label*).

Como exemplo, criaremos uma imagem simples apenas com uma pequena estrutura de diretórios para teste.

```
$ mkisofs -J -r -o TESTE.iso /home/darkstar/teste
Total translation table size: 0
Total rockridge attributes bytes: 1693
Total directory bytes: 0
Path table size(bytes): 10
Max brk space used 8284
464 extents written (0 Mb)
$ _
```

Estes são os mínimos comandos necessários para se criar uma imagem para serem gravadas posteriormente em *CD-ROM*.

CDRECORD

O *cdrecord* é o utilitário padrão para a realização de atividades relacionadas ao processo de gravação de mídias (*CDs* e *DVDs*).

Sintaxe para o uso geral:

```
$ cdrecord [PARÂMETROS]
```

Sintaxe para gravação:

```
$ cdrecord -fs=[BUFFER]M speed=[VELOCIDADE] dev=[LOCALIZAÇÃO] -[ESPECIFICAÇÃO]
[IMAGEM].iso
```

Onde:

- *-blank=[TIPO]*: apaga os dados armazenados em uma mídia regragável. Os subparâmetros (tipos) específicos deste são: *all* -> Apaga tudo, *session* -> apaga uma sessão (para gravações multisessão) e *track* -> faixa de áudio;
- *-dev[N,N,N.]*: localização da unidade *SCSI* em questão (veja *-scanbus*);

B

- *-dummy*: realiza apenas um teste ao invés de realizar a gravação;
- *-eject*: ejeta o disco no final da operação;
- *[ESPECIFIC.]*: define qual tipo de imagem será criado: *-data* (para dados), *-audio* (para CDs de áudio);
- *-fs=[BUFFER]*: especifica o tamanho do *buffer* de memória para a gravação (quando omitido, o valor padrão é 4 MB);
- *-multi*: utiliza o recurso de multi-sessão;
- *-scanbus*: exibe os dados da unidade do sistema.
- *-speed=[VEL]*: define a velocidade de gravação (a máxima deverá ser a suportada pelo aparelho e mídia);
- *-v*: exibe mensagens sobre o andamento do processo.

Segue simples exemplos para...

- Gravação de dados (*ISO*):

```
$ cdrecord -v -fs=8M speed=16 dev=0,0,0 -data backup.iso
```

- Gravação de áudio (faixas):

```
$ cdrecord -v -fs=8M speed=16 dev=0,0,0 -audio [F1], [F2], [F3], ...
```

- Gravação de dados + áudio:

```
$ cdrecord -v -fs=8M speed=16 dev=0,0,0 -data backup.iso -audio [F1], [F2], [F3], ...
```

Lembrem-se de que, apesar de improvável, de acordo com o equipamento o *device* correspondente ao do usuário poderá ser diferente ao acima especificado (*dev=0,0,0*). Para descobrir o *dev* correspondente, utilizem...

```
$ cdrecord -scanbus
```

```
Cdrecord 2.00.3 (i686-pc-linux-gnu) Copyright (C) 1995-2002 Jörg Schilling
Linux sg driver version: 3.1.25
Using libscg version 'schily-0.7'
scsibus0:
```

```
 0,0,0    0) 'HL-DT-ST' 'CD-RW GCE-8525B ' '1.03' Removable CD-ROM
 0,1,0    1) *
 0,2,0    2) *
 0,3,0    3) *
 0,4,0    4) *
 0,5,0    5) *
 0,6,0    6) *
 0,7,0    7) *
```

```
$ _
```

Já as faixas de áudio deverão ser arquivos nos formatos de *.wav* e/ou *.cdr*.

Um recurso interessante do *cdrecord* está na possibilidade de se descobrir todos os dados de uma determinada mídia de *CD-R/RW*. Para isto, basta inseri-lo na bandeja e (sem montar a unidade) digitar o seguinte comando:

```
# cdrecord -atip dev=0,0,0
```



A linha que nos importa será a seguinte:

Manufacturer: CMC Magnetics Corporation

Além do fabricante, observe que também são exibidas outras informações gerais, tanto da unidade gravadora quanto da mídia utilizada.

```
$ cdrecord -atip dev=0,0,0
```

```
Cdrecord 2.00.3 (i686-pc-linux-gnu) Copyright (C) 1995-2002 Jörg Schilling  
scsidev: '0,0,0'
```

```
scsibus: 0 target: 0 lun: 0
```

```
Linux sg driver version: 3.1.25
```

```
Using libscg version 'schily-0.7'
```

```
Device type      : Removable CD-ROM
```

```
Version          : 0
```

```
Response Format: 2
```

```
Capabilities     :
```

```
Vendor_info      : 'HL-DT-ST'
```

```
Identifikation   : 'CD-RW GCE-8525B '
```

```
Revision         : '1.03'
```

```
Device seems to be: Generic mmc CD-RW.
```

```
Using generic SCSI-3/mmc CD-R driver (mmc_cdr).
```

```
Driver flags     : MMC-2 SWABAUDIO BURNFREE
```

```
Supported modes: TAO PACKET SAO SAO/R96P SAO/R96R RAW/R16 RAW/R96P RAW/R96R
```

```
cdrecord: Input/output error. test unit ready: scsi sendcmd: no error
```

```
CDB: 00 00 00 00 00 00
```

```
status: 0x2 (CHECK CONDITION)
```

```
Sense Bytes: 70 00 02 00 00 00 00 0A 00 00 00 00 3A 01 00 00
```

```
Sense Key: 0x2 Not Ready, Segment 0
```

```
Sense Code: 0x3A Qual 0x01 (medium not present - tray closed) Fru 0x0
```

```
Sense flags: Blk 0 (not valid)
```

```
cmd finished after 0.000s timeout 40s
```

```
cdrecord: No disk / Wrong disk!
```

```
bash-2.05b$ cdrecord -atip dev=0,0,0
```

```
Cdrecord 2.00.3 (i686-pc-linux-gnu) Copyright (C) 1995-2002 Jörg Schilling  
scsidev: '0,0,0'
```

```
scsibus: 0 target: 0 lun: 0
```

```
Linux sg driver version: 3.1.25
```

```
Using libscg version 'schily-0.7'
```

```
Device type      : Removable CD-ROM
```

```
Version          : 0
```

```
Response Format: 2
```

```
Capabilities     :
```

```
Vendor_info      : 'HL-DT-ST'
```

```
Identifikation   : 'CD-RW GCE-8525B '
```

```
Revision         : '1.03'
```

```
Device seems to be: Generic mmc CD-RW.
```

```
Using generic SCSI-3/mmc CD-R driver (mmc_cdr).
```

```
Driver flags     : MMC-2 SWABAUDIO BURNFREE
```

```
Supported modes:
```

```
ATIP info from disk:
```

```
  Indicated writing power: 5
```

```
  Is not unrestricted
```

```
  Is not erasable
```

```
  Disk sub type: Medium Type A, high Beta category (A+) (3)
```

```
  ATIP start of lead in: -11634 (97:26/66)
```



```
ATIP start of lead out: 359846 (79:59/71)
Disk type:    Short strategy type (Phthalocyanine or similar)
Manuf. index: 3
Manufacturer: CMC Magnetics Corporation
$ _
```

Lembrem-se de que - dependendo da versão utilizada - o *cdrecord* apenas grava em unidades *SCSI* (os quais as unidades *IDE* são emuladas pelo *kernel* através do módulo *ide-scsi*). Já na versão atual do *kernel* (série 2.6.x), esta restrição não existe...

CONCLUSÃO

A necessidade de ter conhecimentos para a manipulação de unidades, partições e formatos de sistemas de arquivos em sistemas *GNU/Linux* em comparação ao *Windows* pode tornar sua utilização cansativa nestas atividades (em alguns casos, complicados), porém serão vitais para realizar atividades de manutenção em situações que não tenhamos disponíveis ambientes gráficos e suas respectivas ferramentas. &;-D

B

VI. USUÁRIOS, GRUPOS E PERMISSÕES DE ACESSO

INTRODUÇÃO

O conhecimento em administração de contas usuários, grupos e permissões de acesso, além das atividades de edição de arquivos de configuração pertinentes, são de extrema importância para assegurar a boa coexistência entre diferentes utilizadores do sistema.

Neste capítulo, iremos conhecer os principais recursos disponíveis pelos sistemas *GNU/Linux* para o gerenciamento de contas, grupos e permissões.

CONSIDERAÇÕES BÁSICAS

AS CONTAS

Conforme já dito anteriormente, os sistemas *GNU/Linux* são ambientes multi-usuários. Apesar disto - e de acordo com sua categoria - nem todos possuem um mesmo perfil, necessidades e/ou responsabilidades. Para isto existem as contas de autenticação - ou contas de acesso.

Por mais diferentes que tais perfis sejam, estes se enquadram em duas classes distintas: Superusuário - também conhecido como administrador do sistema -, e usuário comum - ou simplesmente usuários.

O ADMINISTRADOR DE SISTEMA

Conforme já comentado diversas vezes, o *root* é o administrador do sistema - muitas vezes também chamado de superusuário.

De acordo com a distribuição utilizada, existirá um momento na instalação do sistema em que será solicitado a senha para acesso do superusuário.

```
Changing password for root
Enter the new password (minimum of 5, maximum of 127 characters)
Please use a combination of upper and lower case letters and numbers.
New password: _
```

Instante final do processo de instalação da distribuição em que é requerida uma senha para o administrador (root).

Somente o *root* possui acesso total ao sistema. É ele quem define as permissões gerais para acesso aos recursos e dados gerais do sistema para o usuário. Em algumas circunstâncias (de acordo com a distribuição utilizada), a sua única limitação existente está no acesso a *Internet*.



Por tratar-se de um usuários com acesso total ao sistema, muitas distribuições definiram as permissões de acesso referentes a conexão com a *Internet* somente disponíveis para os usuários, que por sua vez, caso sejam acessados involuntariamente por invasores da rede, estes não terão as permissões necessárias para ocasionar males ao sistema. Mas em algumas distribuições *expert-user* (como o *Slackware*), por ser destinada a usuários de médio e alto nível técnico, esta limitação não existe, deixando a responsabilidade à cargo dos administradores do sistema.

Após autenticado, o símbolo de indicador na linha de comando é:

```
Login: root
Passwd:
# _
```

O USUÁRIO COMUM

A conta de usuário - ou conta comum -, diferente do superusuário, pois possui apenas permissões básicas para a realização de atividades gerais no sistema e algumas configurações locais, onde não é possível a realização de intervenções de encargo do superusuário, como a instalação / atribuição de permissão / configuração / remoção de programas, arquivos e dispositivos do sistema, entre outros.

Porém, de acordo com as definições do superusuário, as configurações gerais de permissão do usuário poderão ser editadas para que possam atender a determinadas circunstâncias. Por exemplo, podemos atribuir grupos para que estes tenham permissão para acessar a *Internet*, ao sistema de impressão, etc.

Após autenticado, o símbolo de indicador na linha de comando é:

```
Login: darkstar
Passwd:
$ _
```

O ID

O *ID* é um número de identificação para qualquer um dos elemento do sistema - usuário e grupo. Este por sua vez se subdivide em:

- *GID*: definido como a numeração de *ID* específica para os grupos de acesso do sistema. Todos os *IDs* de contas de grupos de acesso pertencem ao intervalo de 1 a 499.
- *UID*: definido como a numeração de *ID* específica para as contas de usuário do sistema. À exceção do superusuário (*UID* = 0), todos os *IDs* de contas de usuário iniciam a partir de 500.

Para obter informações das *IDs* relacionadas uma conta específica...

```
$ id
uid=1000(darkstar) gid=100(users)
groups=100(users),0(root),1(bin),2(daemon),3(sys),5(tty),6(disk),7(lp),8(mem),
```



```
9(kmem),10(wheel),11(floppy),12(mail),13(news),14(uucp),15(man),17(audio),18(v
ideo),19(cdrom),20(games),21(slocate),22(utmp),25(smmsp),27(mysql),32(rpc),33(
sshd),42(gdm),43(shadow),50(ftp),90(pop),93(scanner),98(nobody)
$ _
```

Os GRUPOS

Os grupos de acesso são definições especiais de permissões para serem utilizadas quando houver a necessidade de disponibilizar o acesso a um sistema de dados (arquivos e diretórios) ou um determinado serviço para uma categoria distinta. Por exemplo, num escritório de recursos humanos, onde as planilhas de contabilidade somente poderão ser acessadas pelos profissionais de contabilidade, poderão ser definidos um grupo específico chamado “*contabilidade*”, onde também as permissões de acesso para cada arquivo criado por este grupo deverá estar previamente definido.

As PERMISSÕES

Ao listarmos o conteúdo de um determinado diretório com o comando *ls -l*, poderemos observar a existência de várias colunas contendo informações referentes a: *tipo e permissão de acesso, número de atalhos, proprietário, grupo, tamanho e nome*, respectivamente.

```
lrwxrwxrwx 1 darkstar users 11 2005-02-18 22:03 cdrom -> /mnt/cdrom/
drwxr-xr-x 17 darkstar users 464 2005-03-10 23:57 docs
lrwxrwxrwx 1 darkstar users 11 2005-02-18 22:03 flash -> /mnt/flash/
lrwxrwxrwx 1 darkstar users 12 2005-02-18 22:03 floppy -> /mnt/floppy/
lrwxrwxrwx 1 darkstar users 9 2005-02-18 22:03 pkg -> /mnt/pkg/
drwxr-xr-x 4 darkstar users 208 2005-03-12 21:17 z
```

No modo texto, existem 10 seqüências de caracteres chamados *flag*.

```
lrwxrwxrwx 1 darkstar users
drwxr-xr-x 17 darkstar users
lrwxrwxrwx 1 darkstar users
lrwxrwxrwx 1 darkstar users
lrwxrwxrwx 1 darkstar users
drwxr-xr-x 4 darkstar users
```

O primeiro caracter indica o tipo do elemento:

- *d*: diretório;
- *l*: atalho (*link*);
- *c*: dispositivo do tipo *character*;
- *b*: dispositivo do tipo bloco.

Aos demais caracteres da coluna de permissões de acesso refere-se as *flags* de permissões definidas para os arquivos e/ou diretórios presentes. No modo gráfico, a 1ª. coluna é omitida, visto que os recursos visuais facilitam a exibição do tipo do item.

B

Name	Size	Modified	Permissions	Owner	Group
bin	2,5 KB	24.09.2005 17:42	rwxf-xf-x	root	root
boot	400 B	24.09.2005 10:35	rwxf-xf-x	root	root
dev	61,1 KB	25.09.2005 09:23	rwxf-xf-x	root	root
etc	4,6 KB	25.09.2005 09:23	rwxf-xf-x	root	root
home	96 B	14.07.2005 05:14	rwxf-xf-x	root	root
lib	3,3 KB	24.09.2005 17:43	rwxf-xf-x	root	root
mnt	216 B	24.09.2005 11:16	rwxf-xf-x	root	root
opt	96 B	24.09.2005 17:54	rwxf-xf-x	root	root
proc	0 B	25.09.2005 09:22	r-xf-xf-x	root	root
root	528 B	24.09.2005 16:38	rw-x---	root	root
sbin	6,6 KB	25.07.2005 07:19	rwxf-xf-x	root	bin
share	288 B	24.09.2005 17:42	rwxf-xf-x	root	root
sys	48 B	12.05.2004 04:03	rwxf-xf-x	root	root
tmp	520 B	25.09.2005 10:22	rw-rw-rwt	root	root
usr	544 B	24.09.2005 22:11	rwxf-xf-x	root	root
var	456 B	15.09.2003 20:11	rwxf-xf-x	root	root

Diretório raiz do sistema através do Konqueror, onde são exibidos suas respectivas permissões de acesso, definições de usuário e grupo.

Estas 9 seqüências de letras subdividem-se nos seguintes grupos:

- *Dono:* 1a. a 3a. letras da seqüência, refere-se as permissões de acesso do dono do arquivo em questão - neste caso, *darkstar*;
- *Grupo:* 4a. a 6a. letras da seqüência, refere-se as permissões de acesso do grupo o qual o dono do arquivo pertence - *users*;
- *Todos:* 7a. a 9a. letras da seqüência, refere-se as permissões de acesso dadas àqueles que não sejam donos e nem pertencem ao grupo de acesso do qual o arquivo pertence.

SENHA

Sem maiores dúvidas, os usuários dos sistemas *GNU/Linux* somente poderão ter acesso as suas contas de acesso e tudo o que nela estiver após definirem uma senha de autenticação. Durante a criação de uma nova conta de acesso, será solicitado ao usuário a definição de uma senha, o qual bastará o usuário utilizar um conjunto de caracteres.

COMANDOS GERAIS

ADIÇÃO DE USUÁRIOS E GRUPOS

ADDUSER

O comando *adduser* é utilizado para criar contas de usuários e é somente utilizado pelo administrador do sistema.

Sintaxe:

```
# adduser [APELIDO]
```

...ou simplesmente...

```
# adduser
```



Exemplificaremos detalhadamente o processo de criação da conta de um usuário. Para começar, digitem apenas o comando com administrador do sistema, onde será solicitado um apelido (*nick*) para o usuário.

```
Login name for new user []: darkstar
- User 'Darkstar' contains illegal characters (uppercase); please choose another
```

```
Login name for new user []: _
```

Observe que o comando não aceita o uso de caracteres iniciais maiúsculas (caixa alta), solicitando novamente a inclusão do apelido desejado.

```
Login name for new user []: darkstar
```

```
User ID ('UID') [ defaults to next available ]: _
```

Nesta seção deverá ser definido o *UID* do usuário. Apenas teclem <ENTER> para que o sistema possa definir uma *UID* disponível.

```
Initial group [ users ]: _
```

Aqui deverá ser definido o grupo padrão do usuário ou o grupo inicial, conforme a descrição acima. Digitem o grupo desejado ou apenas teclem <ENTER> para defini-lo no grupo padrão do sistema (*users*).

```
Additional groups (comma separated) []: _
```

Nesta próxima etapa, o comando solicita a informação de outros grupos para a conta a ser criada. O usuário terá acesso aos recursos de acordo com os grupos incluídos para este. Para obterem maiores informações, consultem a seção *Considerações básicas -> Grupos de acesso*.

Após definir os grupos de acesso, teclem <ENTER>.

```
Home directory [ /home/darkstar ] _
```

O sistema definirá o diretório do usuário utilizando o próprio nome da conta. Caso haja necessidade de definir outro caminho, digitem-no na linha de comando ou simplesmente teclem <ENTER> para manter o padrão.

```
Shell [ /bin/bash ] _
```

Definição padrão do interpretador da linha de comando é o *Bash*, indicado acima na opção *Shell*. Caso queiram utilizar outro que não seja o *Bash*, bastará especificá-lo aqui. Por exemplo, para o interpretador *Ash*, */bin/ash*; para o *Csh*, */bin/csh*. Prefiram manter o interpretador padrão, apenas teclando <ENTER>.

```
Expiry date (YYYY-MM-DD) []: _
```

A data de validade poderá ser definida nesta seção, bastando apenas digitá-la no formato *ANO-MÊS-DIA*. Caso queiram que esta não tenha prazo de expiração definido, apenas teclem <ENTER>.

```
New account will be created as follows:
```

```
-----
```

```
Login name.....: darkstar
```



```
UID.....: [ Next available ]
Initial group....: users
Additional groups: [ None ]
Home directory...: /home/darkstar
Shell.....: /bin/bash
Expiry date.....: [ Never ]
```

This is it... if you want to bail out, hit Control-C. Otherwise, press ENTER to go ahead and make the account.

Enfim, tendo a conta criada, será mostrado conforme acima todos os dados digitados pelo superusuário para a criação da nova conta. Logo em seguida, será solicitado pelo sistema os dados adicionais para a nova conta criada. Tecle <ENTER> e iniciaremos o cadastro das informações adicionais.

Creating new account...

```
Changing the user information for darkstar
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
```

Preencha os dados solicitados como *nome completo, endereço, telefone do trabalho, telefone do lar*, além de *outras* informações que considerarem necessárias, se assim desejarem (estas informações são opcionais). Por último, deverá ser definido a senha de acesso.

```
Changing password for darkstar
Enter the new password (minimum of 5, maximum of 127 characters)
Please use a combination of upper and lower case letters and numbers.
New password: _
```

Ao utilizarem um conjunto de caracteres curtos - com 5 ou menos - o comando rejeitará a cadeia definida devido a sua política de segurança para a definição de senhas, onde há o requerimento mínimo de 6 caracteres.

```
Bad password: too short.
Warning: weak password (enter it again to use it anyway).
New password: _
```

Novamente defina uma nova senha de acesso respeitando esta política. Se a cadeia de caracteres estiver de acordo como o exigido, o comando solicitará para que seja repetida a seqüência utilizada para confirmar. Este procedimento é ideal para aquelas circunstâncias em que digitamos algum número ou caracter diferente do desejado, possibilitando com isto corrigir a senha definida.

```
New password:
Re-enter new password:
```

No final da operação, serão exibidas as seguintes mensagens:



```
Password changed.
```

```
Account setup complete.
```

```
# _
```

GROUPADD

Adiciona grupos ao sistema operacional.

Sintaxe:

```
# groupadd [PARÂMETROS] [GRUPO]
```

Onde:

- *-g*: define manualmente um *ID* específico para o grupo;
- *-r*: adiciona uma conta do sistema;
- *-f (force)*: mantém um grupo já existente no sistema.

Exemplo:

```
# groupadd suporte
```

... para que seja criado um novo grupo chamado *suporte*.

ADMINISTRAÇÃO DE CONTAS

LOGIN / LOGOUT / EXIT

Toda vez que o sistema é inicializado, você deverá notar a existência de uma linha de comando com a seguinte mensagem:

```
login: _
```

Sua finalidade é a autenticação do usuário: para que possamos acessar o sistema e usufruir de seus recursos, teremos que nos tornar “*autênticos*”.

```
login: darkstar
```

```
Password: [SENHA]
```

```
$ _
```

E como fazer para nos “*deslogarmos*”? Para isto, existe o comando *logout*.

```
$ logout
```

Outra forma de finalizar uma sessão é utilizando o comando *exit*:

```
$ exit
```

Na utilização destes comandos nas interfaces gráficas, como o *Konsole* ou o *XTerm*, estes apenas têm suas caixas de diálogos finalizadas.

ID

O comando *id* é utilizado basicamente para obter informações dos *IDs* dos



usuários autenticados no sistema, como o seu próprio e também o dos grupos os quais este se encontra.

Para a sua utilização, basta apenas digitar o comando com as contas desejadas autenticadas. Observe nos exemplos abaixo que as informações são exibidas em duas categorias: *UID* (*ID* do usuário) e *gid* (*ID* do grupo).

Superusuário:

```
# id
uid=0(root) gid=0(root)
grupos=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel),11(floppy)
# _
```

Usuário comum:

```
$ id
uid=1000(darkstar) gid=100(users) grupos=100(users),14(uucp)
$ _
```

USERS / GROUPS

Exibem respectivamente os usuários autenticados naquele momento e seus grupos de acesso.

Superusuário:

```
$ users
root
$ groups
root bin daemon sys adm disk wheel floppy
$ _
```

Usuário comum:

```
$ users
darkstar
$ groups
users disk floppy uucp audio video cdrom
$ _
```

PASSWD

O comando *passwd* é utilizado para especificar uma nova senha ou alterar a senha atual de conta de usuário. Basta digitarem...

```
# passwd [USUÁRIO]
```

... ou apenas...

```
$ passwd
```

... para alterarmos a senha do usuário desejado. Segue um exemplo simples para a alteração de uma senha feita pelo administrador, para melhor entendimento.

```
# passwd darkstar
Changing password for darkstar
Old password: _
```

Observem que o comando não aceita de forma alguma a utilização de novas senhas que não possuam um mínimo de 5 caracteres.

```
Enter the new password (minimum of 5, maximum of 127 characters)
Please use a combination of upper and lower case letters and numbers.
New password:
passwd: all authentication tokens update sucessfully
# _
```

Ao tentar utilizarmos uma senha fora dos requerimentos necessários...

```
New password: [SENHA]
Bad password: no change. Try again.
```

Dentro de uma seção de usuário, bastará utilizarem apenas...

```
$ passwd
```

... e o sistema solicitará que o mesmo atualize a senha desta conta.

```
$ passwd
Changing password for darkstar
Old password: [SENHA ANTIGA]
Enter the new password (minimum of 5, maximum of 127 characters)
Please use a combination of upper and lower case letters and numbers.
New password: [SENHA NOVA]
passwd: all authentication tokens update sucessfully
$ _
```

FINGER

Exibe informações gerais de autenticação das contas do sistema.

Sintaxe:

```
$ finger [PARÂMETROS] [USUÁRIO]
```

Onde:

- *-s*: informações adicionais (nome completo, telefone, etc., ou seja, todas as informações solicitadas no ato da criação da conta);
- *-l*: modo de exibição das informações em diversas linhas;

A conta *darkstar* esta conectada no momento da utilização deste comando...

```
$ finger darkstar
Login: darkstar                Name: (null)
Directory: /home/darkstar      Shell: /bin/bash
On since Tue Apr  6 09:27 (UTC) on :0 (messages off)
On since Tue Apr  6 09:27 (UTC) on pts/0  2 hours 25 minutes idle
On since Tue Apr  6 09:42 (UTC) on pts/2 (messages off)
No mail.
No Plan.
$ _
```

Ainda autenticado como *darkstar*...

```
$ finger root
Login: root                    Name: (null)
Directory: /root              Shell: /bin/bash
```



```
Last login Fri Apr 2 18:00 (UTC) on tty1
Mail last read Tue Mar 30 23:23 2004 (UTC)
No Plan.
$ _
```

Observem que, pelo fato do superusuário não estar autenticado, o comando apenas exibe a última vez em que este autenticou-se no sistema.

UPTIME

Exibe o tempo de autenticação do usuário no sistema.

Sintaxe:

```
$ uptime
```

Ao digitar o comando acima descrito...

```
$ uptime
12:29:37 up 3:13, 3 users, load average: 0.03, 0.05, 0.00
$ _
```

ELIMINANDO USUÁRIOS E GRUPOS

USERDEL

Elimina a conta de usuário do sistema.

Sintaxe:

```
# userdel [PARÂMETROS] [USUÁRIO]
```

Onde:

- *-r*: elimina todos os dados gravados em seu diretório pessoal.

Segue um exemplo simples e prático:

```
# userdel darkstar
```

Caso não sejam mais necessários os dados arquivados em seu diretório pessoal, utilizem o parâmetro *-r* desta forma:

```
# userdel -r darkstar
```

GROUPDEL

Elimina o grupo de acesso do sistema.

Sintaxe:

```
# groupdel [GRUPO]
```

Uma dica importante encontrada na documentação eletrônica do comando está na restrição da remoção de um grupo primário de um grupo existente, onde a prévia remoção dos usuários faz-se necessária.

B

OBTENDO OS PRIVILÉGIOS DE OUTROS USUÁRIOS

Apesar da pouca utilização das acessibilidades dos demais usuários - e especialmente o superusuário -, existirão circunstâncias em que haverá a necessidade da obtenção de específicas permissões.

SU

Apesar possuírem plenos poderes, os superusuários somente devem utilizar sua senha de acesso apenas para a administração do sistema, onde sua utilização para tarefas triviais de usuários, além de desnecessária, poderá acarretar riscos para a integridade do sistema. Na maioria das vezes estes necessitam apenas de uma simples conta de acesso para fazer uso dos recursos do sistema em si, porém poderão surgir algumas circunstâncias em que será necessário realizar intervenções ao sistema. Mas como fazer isto, sem ter que estar autenticando-se a todo o momento?

Para obtermos os privilégios de administrador do sistema, sem ter que nos "*deslogarmos*", deveremos utilizar o comando *su*. Sua sintaxe é simples e básica, onde bastará apenas digitar na linha de comando...

```
$ su
Password: [SENHA DO SUPERUSUÁRIO]
# _
```

E fornecer a senha de superusuário para ter as permissões de acesso desejadas. Note que indicador também mudou (\$ -> #).

A utilidade deste comando basicamente é esta: ter acesso aos poderes de superusuário apenas para realizar intervenções básicas e rápidas, retornando logo em seguida às suas atividades comuns.

Dentre algumas limitações deste recurso está a impossibilidade de executar algumas aplicações que façam a utilização das bibliotecas gráficas - estes sequer se encontrarão disponíveis, ao digitar as *las*. Iniciais do executável e teclar <TAB>. Por exemplo, ao utilizar o comando *su* para obter os privilégios de superusuário e tentar executar...

```
# kppp
bash: kppp: command not found
# _
```

... o interpretador retorna uma mensagem de que não se encontra o programa desejado, mesmo estando ciente de que se encontra instalado no sistema e que o superusuário tem plenos poderes para a sua execução.

Dependendo destas limitações, talvez será até mesmo necessário autenticar-se novamente ao sistema como superusuário e iniciar o ambiente gráfico para realizarmos as atividades administrativas desejadas.

Para retornar ao estado anterior, bastará apenas digitarmos...



ATRIBUTOS DE ARQUIVOS E DIRETÓRIOS

Além da manipulação básica dos arquivos em sistemas *GNU/Linux*, poderemos também definir suas permissões e atributos específicos, além das definições dos usuários donos e grupos de acessos destes.

CHMOD

Em modo texto, poderemos utilizar o comando *chmod* para alterar as permissões de acesso destes elementos de acordo com sua necessidade.

Sintaxe:

```
# chmod [ugoa] {+-} [rwx] [ARQUIVO_OU_DIRETÓRIO]
```

...OU...

```
# chmod [NNN] [ARQUIVO_OU_DIRETÓRIO]
```

Onde a categoria *[ugoa]* possui as seguintes definições:

Caracter		Definição de categoria
<i>u</i>	<i>users</i>	... apenas para usuário (dono) do arquivo.
<i>g</i>	<i>group</i>	... apenas para o grupo o qual o usuário se encontra.
<i>o</i>	<i>other</i>	... para outros que não pertençam ao grupo do usuário.
<i>a</i>	<i>all</i>	... para todos.

Já os sinais e atribuição *{+-}* possuem as seguintes definições:

Sinal	Significado / Função
+	Habilita os parâmetros indicados.
-	Desabilita os parâmetros indicados.

Já as *flags [rwx]* possuem as seguintes definições:

flags	Arquivo	Diretório	
<i>r</i>	<i>read</i>	Leitura.	Acesso ao dados de seu conteúdo
<i>w</i>	<i>write</i>	Escrita.	Gravação de dados em seu conteúdo.
<i>x</i>	<i>execute</i>	Execução.	Visualização de dados conteúdo de seu conteúdo.

Por último, *[NNN]* também representa as definições de categorias...

[NNN]	Categoria (Equivalente de...)
<i>1o. número</i>	... <i>2a.</i> a <i>4a.</i> seqüência de letras (dono).
<i>2o. número</i>	... de <i>5a.</i> a <i>Za.</i> seqüência de letras (grupo).

B

[NNN]	Categoria (Equivale de...)
3o. número	... de 8a. a 10a. seqüência de letras (outros).

... e as permissões de acesso, em valor numérico:

N	Funcionalidade básica (permissão...)
0	Nenhum.
1	... apenas para executar.
2	... apenas para gravar.
4	... apenas para ler.

Outro recurso interessante da atribuição dos valores numéricos é a sua combinação para a definição de múltiplas *flags*. Observe a tabela abaixo:

N	Combinação	Funcionalidade combinada
3	1 + 2	Permissão para executar (1) e gravar (2).
5	1 + 4	Permissão para executar (1) e ler (4).
6	2 + 4	Permissão para gravar (2) e ler (4).
7	1 + 2 + 4	Permissão para executar (1), gravar (2) e ler (4).

No geral fica assim:

N	Funcionalidades totais
0	sem permissão.
1	Permissão apenas para executar.
2	Permissão apenas para gravar.
3	Permissão para gravar e executar.
4	Permissão apenas para ler.
5	Permissão para executar e ler.
6	Permissão para gravar e ler.
7	Permissão para executar, gravar e ler.

Segue alguns exemplos práticos para melhor compreensão...

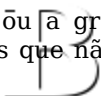
Desabilita a execução do arquivo ou a visualização do conteúdo do diretório especificado apenas para o usuário:

```
$ chmod u-x [ARQ/DIR]
```

Habilita a leitura do arquivo ou o acesso ao diretório especificado apenas para o grupo do qual o usuário se situa:

```
$ chmod g+r [ARQ/DIR]
```

Desabilita a escrita do arquivo ou a gravação de arquivos no diretório especificado somente para outros que não seja o usuário, nem pertençam



ao grupo deste:

```
$ chmod o-w [ARQ/DIR]
```

Desabilita a execução do arquivo ou a visualização do conteúdo do diretório especificado para todos os usuários:

```
$ chmod a+x [ARQ/DIR]
```

Permissão para executar, gravar e ler pelo usuário (7); ler e executar pelo grupo o qual pertença o usuário (5); apenas leitura para outros que não pertençam ao grupo (4):

```
$ chmod 754 [ARQ/DIR]
```

Parece um pouco complicado; mas com a prática, iremos nos familiarizar.

CHOWN

Utilizado para mudar dono e grupo de um arquivo ou diretório.

Sintaxe:

```
# chown [USUÁRIO].[GRUPO] [ARQUIVO_OU_DIRETÓRIO]
```

Onde *[USUÁRIO].[GRUPO]* são as especificações do novo usuário e grupo para o arquivo e/ou diretório desejado. Segue um simples exemplo:

```
# chown darkstar.users /home/darkstar/texto.txt
```

... ou...

```
# chown root.root /usr
```

Para definir os mesmos valores nos arquivos e diretórios de um determinado diretório, deveremos então utilizar o parâmetro *-R* (recursivo).

```
# chown -R root.root /usr
```

CHGRP

Possui a mesma finalidade que o comando *chown*, porém apenas atua na modificação do grupo.

Sintaxe:

```
# chgrp [GRUPO] [ARQUIVO_OU_DIRETÓRIO]
```

Exemplo:

```
# chgrp root /usr
```

Da mesma maneira, podemos definir o conteúdo do diretório recursivamente com o parâmetro *-R*.

```
# chgrp -R root /usr
```



UMASK

Define as permissões padrões que os arquivos e diretórios deverão ter no momento em que serão criados.

Sintaxe:

```
# umask [NNN]
```

Apesar de atribuir as permissões de acesso utilizando a mesma metodologia do comando *chmod*, o comando *umask* utiliza valores diferenciados para os números os quais utiliza. Segue sua tabela de permissões de acesso, equivalente ao *chmod*, porém com valores invertidos, conforme vemos:

- 6: sem permissão;
- 5: permissão apenas para executar;
- 4: permissão apenas para gravar;
- 3: permissão para gravar e executar;
- 2: permissão apenas para ler;
- 1: permissão para executar e ler;
- 0: completo - leitura, execução e escrita.

Por padrão, o valor das permissões praticadas pelo *umask* é *022*, que corresponde ao *644* utilizado pelo *chmod*. Ou seja, apesar de apresentarem as mesmas definições, os valores numéricos são exatamente opostos.

Para alterar o valor *umask*, basta utilizar o comando...

```
$ umask [VALOR]
```

Onde [VALOR] deverá ser o novo perfil de valores das permissões à serem adotadas. Para torná-lo padrão à todos usuários, basta editar o arquivo */etc/profile* e alterar suas definições...

```
# Default umask. A umask of 022 prevents new files from being created group  
# and world writable.  
umask 022
```

... para...

```
umask [VALOR]
```

Uma questão importante está no uso da permissão para execução. Mesmo que na criação da grande maioria dos arquivos não necessitem, os diretórios precisam dela para que possamos acessá-los.




```
games:!!:9797:0:::::
ftp:!!:9797:0:::::
smmsp:!!:9797:0:::::
mysql:!!:9797:0:::::
rpc:!!:9797:0:::::
sshd:!!:9797:0:::::
gdm:!!:9797:0:::::
apache:!!:9797:0:::::
messagebus:!!:9797:0:::::
haldaemon:!!:9797:0:::::
pop:!!:9797:0:::::
nobody:!!:9797:0:::::
darkstar:$1$XQe/m3Bh$s1/1sEdNMEnUKVIWFCaF4/:13729:0:99999:7:::
```

/ETC/GROUPS

No arquivo */etc/groups* estão todas as definições de grupos de autenticação. Todos os grupos padrões do sistema, mais os grupos específicos das aplicações e ainda os grupos criados pelo administrador possuem suas especificações aqui descritas.

```
root::0:root
bin::1:root,bin,daemon
daemon::2:root,bin,daemon
sys::3:root,bin,adm,darkstar
adm::4:root,adm,daemon
tty::5:
disk::6:root,adm
lp::7:lp
mem::8:
kmem::9:
wheel::10:root
floppy::11:root
mail::12:mail
news::13:news
uucp::14:uucp,darkstar
man::15:
games::20:
slocate::21:
utmp::22:
smmsp::25:smmsp
mysql::27:
rpc::32:
sshd::33:sshd
gdm::42:
shadow::43:
ftp::50:
pop::90:pop
nobody::98:nobody
nogroup::99:
users::100:darkstar
console::101:
```



CONCLUSÃO

Apesar da existência de diversos comandos e recursos para a administração de contas, grupos e outras propriedades de um sistema multi-usuário, o bom conhecimento das atividades de gerenciamento de usuários é muito importante, visto que além de manter todo o sistema de dados em segurança, não teremos ocorrências e inconvenientes desagradáveis como a perda de privacidade ou reclamações deste tipo do próprio usuário ou de outras pessoas que fizeram uso da máquina. Além disso, nos dará liberdade de personalizarmos o sistema de acordo com nossas preferências. &:-D

B

VII. O GERENCIAMENTO DE PROCESSOS

INTRODUÇÃO

Conforme enfatizamos diversas vezes, o *kernel* é o principal responsável pelo gerenciamento das atividades do sistema, que por sua vez são caracterizadas como processo. Pelo fato do sistema operacional ser multitarefa, existem diversos processos em andamento.

Neste capítulo iremos conhecer como os sistemas *GNU/Linux* (mais precisamente o *kernel*) administram os processos em execução, bem como todas as instruções e comandos para um bom gerenciamento destes.

VISÃO GERAL

O QUE É UM PROCESSO?

O processo é qualquer atividade executada pelo sistema que envolve uma rotina de instruções com seus respectivos dados e informações. Um processo poderá ser um programa em execução, um comando sendo acionado, uma chamada do sistema, etc.

Existem 3 tipos de processos distintos:

- *Interativos*: são processos executados a partir e controlados por um terminal. Ex.: Comandos gerais da linha de comando.
- *Lotes*: são todos os processos agendados pelo usuário e/ou sistema. Ex.: Programação de tarefas e atividades com o *cron*.
- *Servidores*: são processos executados na inicialização do sistema e que necessitam estar em execução permanente para a utilização de seus serviços por outros processos que o necessitem. Ex.: O banco de dados *MySQL* e o servidor *Apache*.

O IDENTIFICADOR PID

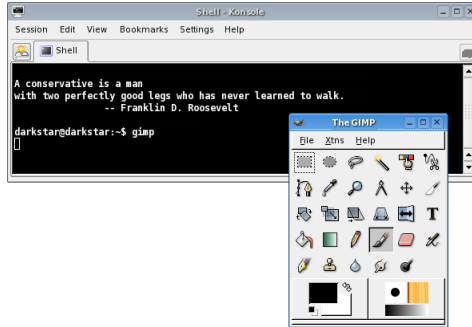
O *PID* é nada mais que uma identificação numérica de um determinado processo existente, que por sua vez, quando criado, passa a ser identificado e manipulado por este número. Já o *PPID* de um processo é nada mais que a informação do processo-pai que o gerou. Seria algo análogo como “a aplicação que executou outra aplicação”.

BACKGROUND E FOREGROUND

Muitas vezes, ao executarmos um aplicativo na linha de comando, esta fica indisponível até o encerramento do mesmo. Apesar da disponibilidade de



vários terminais virtuais, seria deselegante toda vez que ao executarmos um programa...



Observem que, quando o GIMP é executado, o sinal de prontidão fica indisponível.

... ter que abrir um sessão do terminal, que por sua vez deverá ficar aberto até o encerramento deste, poluindo e dificultando o gerenciamento das janelas abertas em seu ambiente gráfico preferido. A este modo de execução, chamamos de inicialização em primeiro plano (*foreground*).

Por isto, temos a possibilidade de executar um determinado programa ou atividade na linha de comando e ainda mantê-la disponível conforme nossas necessidades. O processo gerado para a execução do programa fica em segundo plano (*background*), o qual poderá ser manipulado através de outros comandos pela própria linha de comando, desta vez liberada para outras atividades.

GERENCIANDO OS PROCESSOS

Os principais comandos para o gerenciamento de processos são:

VISUALIZAÇÃO

PS

Exibe os processos os quais estão sendo rodados no sistema.

Sintaxe:

```
$ ps [PARÂMETROS]
```

Onde:

- `-A`: exibe todos os processos;
- `-a`: exibe os processos somente da seção corrente;
- `-p [N]`: verifica a existência de um determinado processo, onde `[N]` é o número do processo em questão;
- `-u`: mostra os processos inicializados pelo usuário.

Experimentem os seguintes comandos e analisem os resultados obtidos de acordo com a tabela apresentada:

```
$ ps
$ ps -a
$ ps -A
$ ps -p [NÚMERO DE UM PROCESSO PRÉ-VISUALIZADO COM OS COMANDOS ANTERIORES]
$ ps -u
$ _
```

Para obterem maiores detalhes, consultem a sua documentação eletrônica.

TOP

Exibe todos os processos em execução, além de outros fatores importantes, como a utilização geral da *CPU* e ocupação da memória.

Sintaxe:

```
$ top [PARÂMETROS]
```

Exemplo:

```
$ top
top - 18:40:14 up 37 min, 1 user, load average: 0.07, 0.03, 0.00
Tasks: 46 total, 2 running, 44 sleeping, 0 stopped, 0 zombie
Cpu(s): 0.7% user, 0.3% system, 0.0% nice, 99.0% idle
Mem: 515444k total, 295952k used, 219492k free, 40912k buffers
Swap: 506008k total, 0k used, 506008k free, 170472k cached
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
159	root	16	0	79428	12m	2244	S	0.3	2.6	0:33.61	X
538	darkstar	12	0	15948	15m	13m	R	0.3	3.1	0:00.45	kdeinit
551	darkstar	12	0	988	988	800	R	0.3	0.2	0:00.08	top
1	root	8	0	236	236	208	S	0.0	0.0	0:04.77	init
2	root	9	0	0	0	0	S	0.0	0.0	0:00.02	keventd
3	root	19	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd_CPU0
4	root	9	0	0	0	0	S	0.0	0.0	0:00.00	kswapd
5	root	9	0	0	0	0	S	0.0	0.0	0:00.00	bdflush
6	root	9	0	0	0	0	S	0.0	0.0	0:00.10	kupdated
10	root	-1	-20	0	0	0	S	0.0	0.0	0:00.00	mdrecoveryd
11	root	9	0	0	0	0	S	0.0	0.0	0:00.00	kreiserfsd
35	root	9	0	0	0	0	S	0.0	0.0	0:00.00	kapmd
81	root	9	0	592	592	512	S	0.0	0.1	0:00.02	syslogd
84	root	9	0	444	444	388	S	0.0	0.1	0:00.01	klogd
124	root	9	0	520	520	464	S	0.0	0.1	0:00.00	inetd
134	root	9	0	528	528	464	S	0.0	0.1	0:00.00	crond
136	daemon	9	0	616	616	548	S	0.0	0.1	0:00.00	atd
140	root	9	0	500	500	448	S	0.0	0.1	0:00.00	apmd

Bastam analisar as informações gerais disponíveis na saída do *monitor*.

O comando também possui umas teclas especiais para funcionalidades específicas, as quais seguem:

- `<ESPAÇO>`: atualiza a tela;



- `<CTRL>+<L>`: atualiza a tela, porém apagando o conteúdo atual;
- `<H>` ou `<?>`: inicializa a ajuda eletrônica;
- `<I>`: ignora processos “zumbis” (ociosos);
- `<K>`: finaliza um determinado processo, onde o usuário será questionado qual o número referente;
- `<Q>`: encerra o programa.

Para sairmos desta tela, bastará apenas pressionarmos a tecla `<Q>`.

SEGUNDO PLANO

COLOCANDO EM SEGUNDO PLANO

Para colocarmos qualquer processo em segundo plano na linha de comando, bastará utilizarmos o caracter `&` (“*e-comercial*”) no final do comando invocado.

```
$ gimp &
[1] 1212
$ _
```

O programa será executado normalmente, porém com a prontidão da linha de comando. Mas se nós esquecermos este detalhe desejarmos colocá-lo em segundo plano?

“CONTROL-ZÊ”

Conforme sabemos, a linha de comando ficará indisponível toda vez que necessitarmos executar outro programa (e conseqüentemente gerar um novo processo).

```
$ gimp
_
```

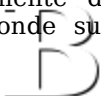
Para que possamos retornar à linha de comando, bastará pressionar `<CTRL> + <Z>`, do qual retornaremos imediatamente ao sinal de prontidão.

```
$ gimp
[1]+  Stopped                  gimp
$ _
```

Porém o aplicativo simplesmente ficará inoperante. Como fazer para reverter este quadro?

BG

Para tornar o aplicativo novamente disponível, deveremos utilizar o comando `bg` - *BackGround* -, onde sua finalidade é colocar qualquer



programa interrompido em segundo plano. Digitem na linha de comando...

```
$ bg
[1]+  gimp &
$ _
```

... para dar continuidade às atividades relacionadas ao aplicativo em execução (no exemplo mencionado, *Gimp*).

JOBS

Apenas exibe os programas que se encontram em segundo plano:

```
$ jobs
[1]+  Running                  gimp &
$ _
```

FG

Retorna qualquer processo parado ou em segundo plano para o primeiro.

```
$ fg 1
gimp
_
```

EXCLUSÃO

KILL

Elimina um processo existente no sistema através do *PID*.

Sintaxe:

```
$ kill [PARÂMETROS] [PROCESSO]
```

Exemplo: Caso desejarmos derrubar um determinado processo...

```
$ ps -p 1084
  PID TTY          TIME CMD
 1084 ttyS4      00:00:00 pppd
$ _
```

... basta utilizar o comando *kill* e fornecer o número do processo do programa ou atividade que desejamos encerrar...

```
$ kill 1084
```

KILLALL

Elimina um processo existente no sistema através do nome.

Sintaxe:

```
$ killall [PARÂMETROS] [PROCESSO]
```



Exemplo: de forma similar ao comando *kill*, caso desejarmos derrubar um determinado processo...

```
$ ps -p 1084
  PID TTY          TIME CMD
 1084 ttyS4      00:00:00 pppd
$ _
```

... basta utilizar o comando *killall* e fornecer o nome do processo do programa ou atividade que desejamos encerrar...

```
$ killall pppd
```

DESLIGAMENTO DO SISTEMA

Para que possamos encerrar todos os processos em execução, desligar e/ou reiniciar o sistema, poderemos nos valer do comando *halt* e *shutdown*.

HALT

Desliga o sistema.

```
# halt
```

Neste caso, os módulos de gerenciamento de energia (*acpi* e/ou *apm*) deverão estar previamente carregados.

SHUTDOWN

Reinicializa o sistema após finalizadas as tarefas em execução.

Sintaxe:

```
# shutdown [OPÇÕES] [HORÁRIO]
```

Onde:

- *-c*: cancela uma programação;
- *-h*: desliga o sistema;
- *-r*: reinicializa o sistema;
- *[HORÁRIO]*: definição de tempo em minutos.

Para se ter uma noção exata de sua utilidade...

```
# shutdown -h 30
```

Desliga o sistema após 30 minutos de sua execução;

```
# shutdown -r now
```

Desliga o sistema imediatamente (*now* = agora);

```
# shutdown -c
```

Cancela a programação de desligamento que foi agendada.

B

CONCLUSÃO

Na grande maioria das circunstâncias - em especial para os usuários menos habituados - sequer nos importaremos com os processos em execução e suas atividades relacionadas. Face a isto, apenas dispomos por incluir os comandos e parâmetros mais essenciais desta atividade. Mas, se ainda assim quiserem obter mais conforto, experimentem utilizar os utilitários disponíveis dos principais ambientes gráficos existentes. &-D

B

VIII. O SISTEMA DE INICIALIZAÇÃO

INTRODUÇÃO

Diferente do *MS-DOS* e *Windows*, os sistemas *GNU/Linux* lidam de forma diferente com o processo de inicialização do computador. Enquanto que no *Windows*, apenas vemos uma tela de apresentação indicado o carregamento do sistema, no *GNU/Linux* são mostrados uma série de processos.⁸

Neste capítulo, iremos conhecer como funciona o sistema de inicialização dos sistemas *GNU/Linux*, bem como suas características e particularidades, tendo um enfoque especial ao método de inicialização do *Slackware*.

OS MÉTODOS DE INICIALIZAÇÃO

Os métodos de inicialização padrão dos sistemas *Unix*, clones e variantes são respectivamente o *System V* e o estilo *BSD*.

SYSTEM V

O *System V (AT&T)* é o método mais utilizado pelas distribuições atuais. Consiste em utilizar dezenas de *scripts* para cada serviço à ser inicializado, todos armazenados em um diretório específico de acordo o nível de execução utilizado. Além disso, operar em vários modos existentes, todos numerados de 0 a 6.

ESTILO BSD

O estilo *BSD (Berkeley Software Distribution)* é atualmente adotado pelas distribuições *Slackware*, *Debian* e *SuSe*. Diferente do outro sistema de inicialização, o estilo *BSD* utiliza apenas alguns *scripts* são carregados durante o processo de inicialização, estes considerados mais rápidos e eficientes, além de uma maior simplicidade quando de sua manutenção. Além disso, opera em apenas 2 modos: o *single-user* ("S" - para manutenção) e o *multi-user* ("M" - uso para produção).

Os *scripts* de inicialização do *Slackware* obedece à este estilo, que por sua vez é simples e extremamente rápido, enquanto que as demais distribuições se baseiam na utilização do método de inicialização *System V*. Apesar disto, os níveis de execução são conforme o método *System V*.

8 Devido a existência de diferentes distribuições, estes aspectos poderão variar: as distribuições amigáveis tendem a apresentar uma tela gráfica no modo de espera, ao passo que as distribuições direcionadas para o público especializado tendem a mostrar a inicialização dos processos, conforme comentado neste parágrafo.

OS SCRIPTS DE INICIALIZAÇÃO

Os *scripts* de inicialização são arquivos de lote que armazenam as definições necessárias para a habilitação dos serviços necessários para o sistema, conforme especificado pelo seu nível de execução - o *runlevel*.

Estes são o *scripts* de inicialização que são os responsáveis pela inicialização tanto da máquina em si quanto dos níveis de execução que são definidos pela sua configuração e até mesmo no ato de seu desligamento:

- *rc.S*: inicialização (*start*) do *Slackware*;
- *rc.K*: responsável pelo nível de execução *1*, para a manutenção do sistema - *single user*;
- *rc.M*: utilizado nos demais níveis de execução, ou seja, para modo multi-usuário (*2, 3 e 5*);
- *rc.4*: necessário especialmente para carregar em modo gráfico (através dos gerenciadores *KDM, GDM e XDM*);
- *rc.0*: atalho simbólico para *rc.6*;
- *rc.6*: reinicialização da máquina (liga) e encerramento (desliga).

Quando o sistema inicializa, o *kernel* é carregado para a memória do sistema, após a inicialização dos dispositivos, este roda o *init*, o primeiro processo de execução do sistema - *PID 1*. Após ser carregado, o *kernel* executa o *runlevel* especificado pelo arquivo de configuração */etc/inittab*. Ainda neste mesmo arquivo de definições, especifica-se os *scripts* necessários para a execução do *runlevel* desejado.

Todos estes *scripts* são executados pelo *Bash* e podem ser editados manualmente, porém recomenda-se a realização prévia de cópias de segurança, para nos resguardarmos de algum imprevisto qualquer.

OS DEMAIS SCRIPTS...

Conforme dito anteriormente, todos estes *scripts* - além dos *scripts* de inicialização - são mantidos em */etc/rc.d/*. Para cada um serviço ou categoria de serviços, haverá um *script* específico para a sua habilitação.

```
# cd /etc/rc.d/
# ls
init.d/          rc.dnsmasq      rc.modules@     rc.sshd
rc.0@           rc.font*        rc.modules-2.6.21.5*  rc.syslog*
rc.4*           rc.gpm*         rc.modules-2.6.21.5-smp*  rc.sysvinit*
rc.6*           rc.hald*        rc.mysqlqd      rc.udev*
rc.K*           rc.hplip        rc.nfsd*        rc.wireless*
rc.M*           rc.httpd        rc.ntpd         rc.wireless.conf
rc.S*           rc.inet1*       rc.pcmcia      rc.ypp*
rc.acpid*       rc.inet1.conf   rc.rpc          rc0.d/
rc.alsa*        rc.inet2*       rc.samba       rc1.d/
```



rc.atalk	rc.inetd*	rc.saslauthd	rc2.d/
rc.bind	rc.ip_forward	rc.scanluns*	rc3.d/
rc.bluetooth	rc.keymap*	rc.sendmail	rc4.d/
rc.bluetooth.conf	rc.local*	rc.serial	rc5.d/
rc.cups	rc.messagebus*	rc.snmpd	rc6.d/
# _			

Subdividiremos por categoria e descreveremos breves comentários sobre os mesmos, para que possamos ter uma melhor compreensão.

SISTEMA & APLICAÇÕES

Todos os *scripts* aqui listados são utilizados para habilitar os recursos do sistema para o suporte a nível de *software* em geral:

- *rc.font*, *rc.fuse*, *rc.hald*, *rc.messagebus*, *rc.mysql* e *rc.syslog*.

Em destaque, os *scripts* *rc.hald* (servidor *HAL*) e *rc.messagebus* (servidor *IPC D-Bus*), pois é graças a eles que o *kernel* consegue realizar a detecção de *hardware* corretamente e a intercomunicação entre aplicações. Já o *rc.syslog* é de extremamente útil, pois registra todos os eventos do sistema em arquivos de *LOGs*. Consultando estes arquivos, poderemos verificar a ocorrência de falhas e invasões, além de obter outras informações úteis.

SUPORTE AO HARDWARE

Nesta categoria, estão listados os *scripts* definidos para prover ao sistema, o suporte geral aos recursos de *hardware* presentes:

- *rc.acpid*, *rc.alsa*, *rc.bluetooth*, *rc.gpm*, *rc.keymap*, *rc.pcmcia*, *rc.scanluns*, *rc.serial*, *rc.uddev* e *rc.wireless*.

O script *rc.alsa* é o responsável pelo carregamento dos módulos e suporte aos utilitários relacionados a arquitetura *ALSA*.; Já os *scripts* *rc.gpm* e *rc.keymap* definem as configurações gerais dos dispositivos de entrada e saída - *mouse* e teclado - no modo texto. Por último, o *rc.uddev* habilita os arquivos de dispositivos do sistema, conhecidos como *devices*: são através destes arquivos que o *kernel* realiza os processos interação com os recursos de *hardware* do sistema (computador).

SISTEMA DE IMPRESSÃO

Atualmente, o *CUPS* é o servidor de impressão mais popular para os sistemas *Unix*, com destaque para o *GNU/Linux* e os sistemas **BSDs*.

- *rc.cups* e *rc.hplip*.

O *script* responsável pela inicialização do servidor de impressão é o *rc.cups*; porém, é através do *script* *rc.hplip* é garantido o suporte para as impressoras fabricadas pela *HP* (*HP Linux Imaging and Printing*).



REDES & INTERNET

Pelo fato dos sistemas *GNU/Linux* serem excelentes opções para o uso em sistemas de redes, uma série de serviços relacionados é inicializado através de *scripts*....

- *rc.atalk*, *rc.bind*, *rc.dnsmasq*, *rc.httpd*, *rc.inetd* (*rc.inet1* e *rc.inet2*), *rc.ip_forward*, *rc.nfsd*, *rc.ntp*, *rc.rpc*, *rc.samba*, *rc.saslauthd*, *rc.sendmail*, *rc.snmpd*, *rc.sshd* e *rc.yip*.

E ainda que venhamos a utilizar o *Slackware* em um *desktop*, alguns destes *scripts* são muito importantes para garantir a conectividade com a *Internet*, como o *rc.inetd*.

CONFIGURAÇÕES LOCAIS

O *script rc.local* foi designado para o carregamento das configurações particulares da máquina local, sendo o último a ser executado.

```
#!/bin/sh
#
# /etc/rc.d/rc.local: Local system initialization script.
#
# Put any local setup commands in here:
...
```

Observem que este *script* não possui nenhuma definição de comandos e opções para a inicialização de serviços e aplicações, ficando totalmente à cargo do administrador definir quais destes é que deverão ser carregados.

O banco de dados *MySQL* (quando instalado manualmente) e os utilitários como o *HDParm* e o *SMARTD* são bons exemplos de programas os quais necessitam ter comandos definidos para sua habilitação.

CARREGAMENTO DE MÓDULOS

Para o carregamento de módulos, existe um *script* definido unicamente para esta função: o *rc.modules*. Este *script* possui aproximadamente 700 linhas, em que seu conteúdo, por sua vez são subdivididos em diversas seções para facilitar a organização dos comandos e instruções lá descritos.

```
#!/bin/sh
# rc.modules 11.0 Tue Jul 25 14:38:32 CDT 2006 pp (rb), pjv
#
# This file loads extra drivers into the Linux kernel.
#
# The modules will be looked for under /lib/modules/<kernel version number>
# On systems using KMOD and hotplug or udev this file should remain mostly
# commented out. Nearly all hardware device modules will be loaded
# automatically on such systems. This file should only be used when hotplug
# or udev are not loading a module that you require, or if you are not using
```



```

# hotplug or udev (which is going to become increasingly impossible...), or
# if you want to force a particular module to be loaded where alternatives
# exist.
#
# Many Linux kernel modules will accept extra options. The Linux kernel
# source is the best place to look for extra documentation for the various
# modules. This can be found under /usr/src/linux/Documentation if you've
# the installed the kernel sources.
#
# NOTE: This may not be a complete list of modules. If you don't see what
# you're looking for, look around in /lib/modules/2.x.x/ for an appropriate
# module. Also, if any problems arise loading or using these modules, try
# compiling and installing a custom kernel that contains the support instead.
# That always works. ;^)
...

```

Cabeçalho do script /etc/rc.d/rc.modules.

Uma observação interessante está no fato da inclusão de instruções para a habilitação de módulos especiais do sistema, como o suporte a uma placa de *fax-modem* (especialmente *softmodem*) ou qualquer outro periférico não suportado oficialmente pela distribuição. Alguns instruem em colocar estas definições em *rc.modules*; outros em *rc.local*. Independente do arquivo utilizado, inclua as instruções sempre no final do arquivo, para que possamos localizá-la de forma padronizada e com maiores facilidades.

COMPATIBILIDADE

A grande maioria das distribuições *GNU/Linux* optam por utilizar o método de inicialização *System V*, que por sua vez as aplicações disponíveis para o sistema estão condicionadas a utilizar as definições gerais deste método.

```

#!/bin/sh
#
# rc.sysvinit This file provides basic compatibility with SystemV style
# startup scripts. The SystemV style init system places
# start/stop scripts for each runlevel into directories such as
# /etc/rc.d/rc3.d/ (for runlevel 3) instead of starting them
# from /etc/rc.d/rc.M. This makes for a lot more init scripts,
# and a more complicated execution path to follow through if
# something goes wrong. For this reason, Slackware has always
# used the traditional BSD style init script layout.
#
# However, many binary packages exist that install SystemV
# init scripts. With rc.sysvinit in place, most well-written
# startup scripts will work. This is primarily intended to
# support commercial software, though, and probably shouldn't
# be considered bug free.
#
# Written by Patrick Volkerding <volkerdi@slackware.com>, 1999
# from an example by Miquel van Smoorenburg
<miquels@cistron.nl>
...

```



O *Slackware*, por não utilizar este método de inicialização, e por necessitar obter compatibilidade com tais programas, definiu o *script rc.sysvinit*.

A SEQUÊNCIA DE SCRIPTS NA INICIALIZAÇÃO

O primeiro *script* a ser executado é o *rc.S*, que roda apenas uma única vez durante a inicialização do sistema. Após o *rc.S*, vem o *rc.M*, responsável pela utilização do sistema no modo multi-usuário, desde que o nível de execução esteja pré-configurado em */etc/inittab* para isto.

O *script rc.K* somente é executado quando houver a necessidade de manutenção do sistema. Por entrar no modo monousuário, todos recursos multi-usuários são desabilitados, como também qualquer serviço (processos) para que possamos intervir no sistema.

OS NÍVEIS DE EXECUÇÃO

Runlevels - níveis de execução - são estágios de execução específicos que visam habilitar e/ou desabilitar um conjunto de serviços específicos para cada necessidade da máquina em uso. Por exemplo, em uma necessidade de uso comum, de manutenção, para propósitos específicos do sistema, enfim, existirá um nível de execução apropriado para cada um. Os níveis de execução poderão ser diferentes de acordo com a distribuição utilizada.

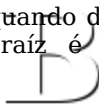
Segue abaixo os níveis de execução do *Slackware*:

- 0: atalho simbólico para *rc.6*;
- 1: modo *single user*, necessário para a manutenção do sistema;
- 2: sem uso (personalizável);
- 3: modo multi-usuário, autenticação em modo texto (*console*);
- 4: modo multi-usuário, autenticação em modo gráfico (*X11*);
- 5: sem uso (personalizável);
- 6: reinicialização e desligamento do sistema.

Em outras distribuições, as definições dos níveis de execução podem variar. Por exemplo, as *red-likes* utilizam o nível 5, que é destinado ao modo multi-usuário com autenticação gráfica, ao invés do nível 4 usado pelo *Slackware*.

NÍVEL 1 - MANUTENÇÃO DO SISTEMA

O nível 1 somente é executado quando da necessidade de manutenção do sistema. Somente a partição raiz é montada para as intervenções



necessárias, sendo este o principal motivo pelo qual não podemos definir uma partição especial para o diretório padrão do superusuário (*root*).

NÍVEL 3 E 4 - MODO MULTI-USUÁRIO

Os níveis 3 e 4 são definidos para a utilização normal do sistema. É com eles que iniciamos o sistema para realizar as atividades do nosso dia-a-dia - ou finais de semana, dependendo da nossa disponibilidade... &-D

A principal diferença está no carregamento da interface gráfica e seus respectivos gerenciadores de autenticação. O nível 3 inicializa o sistema em modo texto, onde o usuário digita seu apelido de autenticação e senha de acesso, para que acionem logo em seguida o ambiente gráfico, digitando para isto...

```
$ startx
```

Já o nível 4 inicializa o sistema em modo gráfico, disponibilizando um gerenciador de autenticação gráfico pré-definido que disponibiliza diversos recursos gráficos para um melhor conforto e facilidades aos usuários.

Apesar do nível 3 ser a opção menos confortável na realização da autenticação e seleção/inicialização da interface gráfica, é em muitas circunstâncias a mais prática nas circunstâncias em que ocorrem problemas que impedem a inicialização do servidor gráfico (*X.org*). Já o nível 4 é indicado especialmente para usuários leigos, os quais não possuem conhecimentos técnicos para a manipulação do sistema em modo texto, como a seleção e inicialização do ambiente gráfico (na utilização dos gerenciadores *KDM* e *GDM*), desligamento do sistema, etc. Praticamente todas as distribuições *friendly-user* o habilitam por padrão.

Somente o nível 4 possui um *script* de inicialização, o *rc.4*, que por sua vez define quais os gerenciadores de autenticação deverão ser rodados, onde deveremos comentar e/ou descomentar as linhas correspondentes aos gerenciadores que desejamos utilizar (ou não), ou ainda modificar a ordem de execução dos mesmos, para dar prioridade ao gerenciador desejado.

Estas instruções e muitas outras se encontram com maiores detalhes na *6a. Parte: Ambientes Gráficos -> Operações e ajustes afins*.

NÍVEL 0 E 6 - REINICIALIZAÇÃO DO SISTEMA

A partir do momento em que reinicializamos ou desligamos o computador, automaticamente o sistema utiliza o nível de execução 6 para que todos os serviços e processos sejam finalizados antes do sistema ser reinicializado. Ao utilizarmos o comando *shutdown*, estaremos acionando este nível.



DEMAIS NÍVEIS DE EXECUÇÃO (2 E 5)

Os níveis 2 e 5 não são utilizados pelo sistema, enquanto que existem outros níveis não especificados, de 7 a 9, que são utilizados especificamente para o desenvolvimento de níveis de execução customizados de acordo com as necessidades da máquina local.

OPERAÇÕES E AJUSTES AFINS

ATIVAR / DESATIVAR SCRIPTS DE INICIALIZAÇÃO

MÉTODO MANUAL

O método manual de ativar e/ou desativar os serviços na inicialização é feito através da mudança das permissões de execução (*flag x*) de seus respectivos *scripts*. Dentro do diretório */etc/rc.d*, basta executar...

```
# chmod a+x [SCRIPT]
```

... para ativarmos, ou...

```
# chmod a-x [SCRIPT]
```

... para desativarmos.

Poderemos também definir as demais permissões juntas da seguinte forma:

```
# chmod [PERMISSÃO] [SCRIPT]
```

Onde *[PERMISSÃO]* deve ser substituído pelas *flags* necessárias. Por exemplo, para ativarmos...

```
# chmod u+x rc.yo
```

... ou para desativarmos...

```
# chmod u-x rc.yo
```

Para obterem maiores informações sobre as permissões de acesso, consultem nesta parte, o capítulo *Manipulação de arquivos e diretórios*.

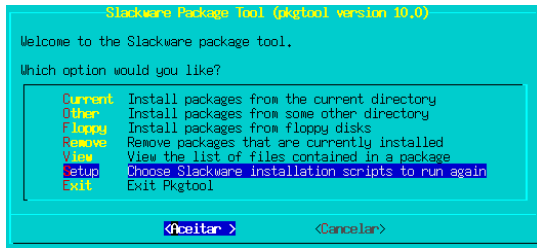
MÉTODO AUTOMATIZADO

O *Slackware* possui um atalho nos menus do *Pkgtool* para ativar e/ou desativar os *scripts* de inicialização. Basta carregarmos o utilitário...

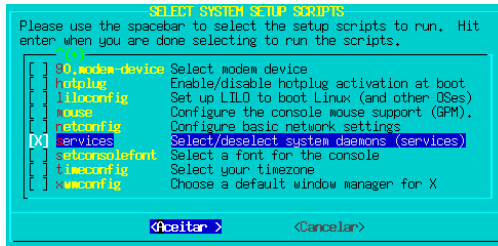
```
# pkgtool
```

... e acionar a opção *Setup* disponível na tela principal:

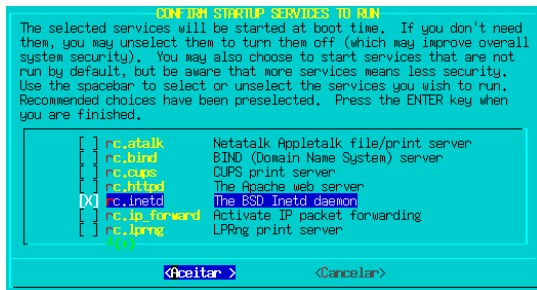




Em seguida, marcar a opção *services* e teclar <ENTER> em seguida...



Nesta 3a. tela, deveremos apenas definir quais os *scripts* que deverão ser ativados (marcando-os com a barra de espaço) e/ou desativados.



O ARQUIVO /ETC/INITTAB

O arquivo de configuração */etc/inittab* define todos os parâmetros e definições gerais do método de inicialização sistema.

```
#
# inittab          This file describes how the INIT process should set up
#                  the system in a certain run-level.
#
# Version:        @(#)inittab          2.04    17/05/93      MvS
#                  2.10    02/10/95      PV
#                  3.00    02/06/1999    PV
#                  4.00    04/10/2002    PV
#
# Author:         Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
```



```
# Modified by: Patrick J. Volkerding, <volkerdi@slackware.com>
#

# These are the default runlevels in Slackware:
# 0 = halt
# 1 = single user mode
# 2 = unused (but configured the same as runlevel 3)
# 3 = multiuser mode (default Slackware runlevel)
# 4 = X11 with KDM/GDM/XDM (session managers)
# 5 = unused (but configured the same as runlevel 3)
# 6 = reboot

# Default runlevel. (Do not set to 0 or 6)
id:3:initdefault:

# System initialization (runs when system boots).
si:S:sysinit:/etc/rc.d/rc.S

# Script to run when going single user (runlevel 1).
su:1S:wait:/etc/rc.d/rc.K

# Script to run when going multi user.
rc:2345:wait:/etc/rc.d/rc.M

# What to do at the "Three Finger Salute".
ca::ctrlaltdel:/sbin/shutdown -t5 -r now

# Runlevel 0 halts the system.
l0:0:wait:/etc/rc.d/rc.0

# Runlevel 6 reboots the system.
l6:6:wait:/etc/rc.d/rc.6

# What to do when power fails.
pf::powerfail:/sbin/genpowerfail start

# If power is back, cancel the running shutdown.
pg::powerokwait:/sbin/genpowerfail stop

# These are the standard console login gettys in multiuser mode:
c1:1235:respawn:/sbin/agetty 38400 tty1 linux
c2:1235:respawn:/sbin/agetty 38400 tty2 linux
c3:1235:respawn:/sbin/agetty 38400 tty3 linux
c4:1235:respawn:/sbin/agetty 38400 tty4 linux
c5:1235:respawn:/sbin/agetty 38400 tty5 linux
c6:12345:respawn:/sbin/agetty 38400 tty6 linux

# Local serial lines:
#s1:12345:respawn:/sbin/agetty -L ttyS0 9600 vt100
#s2:12345:respawn:/sbin/agetty -L ttyS1 9600 vt100

# Dialup lines:
#d1:12345:respawn:/sbin/agetty -mt60 38400,19200,9600,2400,1200 ttyS0 vt100
#d2:12345:respawn:/sbin/agetty -mt60 38400,19200,9600,2400,1200 ttyS1 vt100
```



```
# Runlevel 4 used to be for an X window only system, until we discovered
# that it throws init into a loop that keeps your load avg at least 1 all
# the time. Thus, there is now one getty opened on tty6. Hopefully no one
# will notice. ;^)
# It might not be bad to have one text console anyway, in case something
# happens to X.
x1:4:wait:/etc/rc.d/rc.4

# End of /etc/inittab
```

Dentre as intervenções mais realizadas, está na mudança do nível de execução. Por padrão, o *Slackware* utiliza o nível 3, mas em virtude do conforto proporcionado pela inicialização em modo gráfico, é recomendável utilizar o nível 4, o qual o sistema acessa um gerenciador de autenticação simples, intuitivo e fácil de usar. Consultem a *6a. Parte: Ambientes Gráficos -> Operações e ajustes afins*, para obterem maiores informações.

CONCLUSÃO

Conforme já dito diversas vezes, uma característica interessante do método de inicialização *BSD* está na velocidade e facilidade de customização. O fato de dispor apenas de alguns *scripts* com todas as seções pré-organizadas e comentadas não só facilitam as intervenções necessárias, como também agilizam na procura de parâmetros específicos. No método *System V*, teríamos o trabalho de identificar qual o *script* contém a configuração o qual desejamos editar, ao passo que no estilo *BSD* apenas bastaria localizar o *script* o qual contém a categoria do perfil de configuração e navegar nas seções comentadas. Poderemos também utilizar as ferramentas de busca dos editores de textos disponíveis para realizar a localização desejada. &;-D



IX. O GERENCIADOR DE INICIALIZAÇÃO

INTRODUÇÃO

Quando se utiliza um computador com mais de um sistema operacional, será necessário definir por qual deles será utilizado. Em se tratando de sistemas instalados em diferentes discos rígidos, bastaria inverter no *setup* da *BIOS* a ordem de inicialização. Mas mesmo assim seria um desconforto muito grande, pois toda vez que for necessário inicializar o outro sistema, teríamos que editar as configurações do *setup* para inverter a ordem de inicialização. E para aqueles sistemas instalados em um mesmo disco rígido, porém mantidos em diferentes partições, como fazer?

Para a solução destes problemas, existem os gerenciadores de inicialização, que são programas desenvolvidos para o gerenciamento de diferentes sistemas operacionais instalados em um mesmo equipamento. E neste capítulo, iremos conhecer o *LILO*, o gerenciador padrão do *Slackware*.

O LILO

✓ <<http://lilo.go.dyndns.org/>>.

O *LILLO* - *LI*nux *LO*ader - é um dos gerenciadores mais antigos existentes em sistemas *GNU/Linux*. Ele é o principal responsável pelo carregamento dos sistemas operacionais instalados, onde fornece ao usuário um simples menu de opções, bastando apenas ao usuário selecionar o sistema desejado. Ele fica residente no setor *MBR* do disco rígido.



O *LILO* é adicionado ao sistema por padrão durante a instalação do *Slackware*, onde será necessário apenas definir o perfil de configuração e o local onde deverá ser gravado. Para obterem maiores informações, consultem a *3a. Parte: A Instalação -> Instalação do Slackware*.

/ETC/LILO.CONF

Todos os parâmetros de configuração do *LILO*, são mantidos no arquivo */etc/lilo.conf*, o qual poderá ser alterado de acordo com as necessidades do usuário. Neste arquivo existem duas seções distintas para a configuração: a seção global e a seção de partições. Segue abaixo o conteúdo destas seções e as respectivas utilidades para cada parâmetro.

SEÇÃO GLOBAL

A seção global é responsável pelo funcionamento geral do *LILO*. É nela onde estarão todos os parâmetros pertinentes.

```
# LILO configuration file
# generated by 'liloconfig'
#
# Start LILO global section
boot = /dev/hda
message = /boot/boot_message.txt
prompt
timeout = 1200
# Override dangerous defaults that rewrite the partition table:
change-rules
  reset
```

Constam os seguintes parâmetros passíveis de edição:

- *boot = /dev/[DISCO_RÍGIDO]*

Informa qual unidade do sistema deverá inicializar.

- *Message = /boot/[MENSAGEM_TEXTO]*

Exibe um conjunto de instruções e informações básicas sobre os sistemas disponíveis, conforme a exibição do arquivo-texto indicado.

```
Welcome to the LILO Boot Loader!
```

```
Please enter the name of the partition you would like to boot
at the prompt below. The choices are:
```

```
DOS      - DOS or Windows (FAT/FAT32 partition)
Linux    - Linux (Linux native partition)
```

Este é o conteúdo do arquivo *boot_message.txt*, situado em */boot*. E se tratando de um arquivo-texto, é perfeitamente possível customizá-lo, mantendo apenas as instruções básicas sobre os sistemas disponíveis ou personalizando-o, de acordo com suas preferências.



Após esta operação, atualizem as informações na *MBR*, digitando...

```
# lilo
Added DOS *
Added Linux
```

Na próxima inicialização, o novo conteúdo será exibido.

- *default*

Especifica o sistema a ser carregado por padrão (por ordem de disponibilidade). Lógico que esta opção deverá existir quando houver mais de um sistema operacional.

- *prompt*

Indica a inicialização da linha de comando do *LILLO*. Caso esta opção não esteja habilitada e esta linha de comando não esteja presente, bastará utilizar <CTRL> + <ALT> ou <CTRL> + <SHIFT> para ativá-la.

- *timeout = [TEMPO_DE_ESPERA]*

Tempo o qual o gerenciador permanece aguardando uma resposta via teclado para a seleção dos sistemas. Caso não sejam selecionados, o gerenciador automaticamente carrega o sistema padrão da máquina.

- *Framebuffer*

Define a forma de exibição das fontes no modo texto durante a instalação do *Slackware* através do *framebuffer*, um recurso especial em que o *kernel* acessa diretamente a memória de vídeo e conseqüentemente utilizando seus recursos, como a resolução gráfica. Mas como fazer para habilitar ou desabilitar este recurso, além de alterar as resoluções presentes?

```
# Normal VGA console
vga = normal
# VESA framebuffer console @ 1024x768x64k
# vga=791
# VESA framebuffer console @ 1024x768x32k
# vga=790
# VESA framebuffer console @ 1024x768x256
# vga=773
# VESA framebuffer console @ 800x600x64k
# vga=788
# VESA framebuffer console @ 800x600x32k
# vga=787
# VESA framebuffer console @ 800x600x256
# vga=771
# VESA framebuffer console @ 640x480x64k
# vga=785
# VESA framebuffer console @ 640x480x32k
# vga=784
# VESA framebuffer console @ 640x480x256
# vga=769
# End LILLO global section
```

Observe que a opção padrão utilizada pelo gerenciador de inicialização é

B

vga = normal. Caso queira habilitar o *framebuffer*, bastará apenas selecionar a opção de resolução e profundidade de cor e desmarcar o comentário ao lado. Na próxima inicialização, este recurso estará em vigor.

SEÇÃO DE PARTIÇÕES

A seção de partições contém descritas as opções para de inicialização dos sistemas operacionais já instalados no micro.

Em uma unidade de armazenamento, onde existem duas partições para o *Windows* e *GNU/Linux*, a estrutura da seção de partições seria a seguinte:

```
# DOS bootable partition config begins
other = /dev/hda1
  label = DOS
  table = /dev/hda1
# DOS bootable partition config ends
# Linux bootable partition config begins
image = /boot/vmlinuz
  root = /dev/hda6
  label = Linux
  read-only
# Linux bootable partition config ends
```

Observe que, por sua vez, esta seção subdivide-se em mais duas distintas: uma especial para a inicialização de outros sistemas (*DOS*), e outra para a inicialização de sistemas *GNU/Linux* (*Linux*).

Ademais, segue as definições de cada parâmetro desta seção:

- *image = /[IMAGEM_COMPACTADA]*

Inicializa a imagem compactada do *kernel* utilizado. Este geralmente se encontra armazenado no diretório */boot*, porém você poderá utilizar outros locais conforme desejar. Por exemplo, sempre mantenho a seguinte opção:

```
image = /usr/src/linux/arch/i386/boot/bzImage
  root = /dev/hda6
  label = Experimental
  read-only
# Linux bootable partition config ends
```

É muito útil para experimentar novos *kernels*, sem ter que ficar remexendo no sistema toda vez para realizamos uma nova compilação.

- *root = /dev/[PARTIÇÃO_LINUX]*

Informa onde se encontra a partição de sistemas *GNU/Linux* instalada. Atentem-se para que diferentes sistemas (*kernels*) possam estar localizadas em uma mesma partição, como é recomendável ser feito no caso da compilação de um *kernel* experimental.

- *label = [NOME_DO_SISTEMA]*

Exibe o nome do sistema instalado. Ao utilizarmos diferentes



distribuições, deveremos atribuir sua nomenclaturas no *label* correspondente.

- *read-only*

Pelo fato da necessidade de checagem do sistema de arquivos antes de inicializado, esta partição deverá ser montada antes como somente leitura para depois ser montada normalmente.

- *other = /dev/[PARTIÇÃO_COM_OUTROS_SISTEMAS]*

Indica a partição onde se encontram outros sistemas operacionais para serem inicializados.

- *table = /dev/[DISCO_RÍGIDO]*

Informa em qual disco rígido se encontra outro sistema operacional instalado, se porventura existir.

- *alias = [CARACTER]*

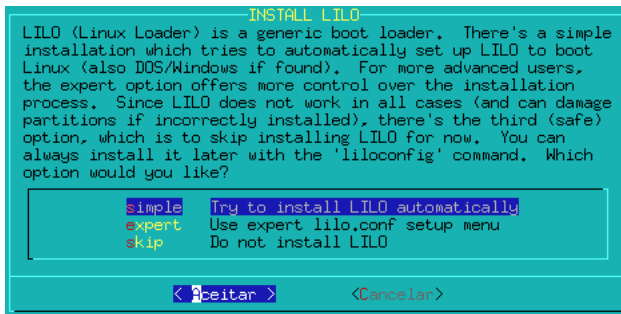
Utilizado em conjunto com a opção *single-key* da seção global, atribui um único caracter para que possamos selecionar o sistema operacional desejado (esta opção deverá estar dentro das definições dos sistemas listados, da mesma forma que *label*).

O LILOCONFIG

O *liloconfig* é um utilitário especialmente desenvolvido para definir as configurações básicas deste gerenciador de inicialização da máquina. Para executá-lo, basta digitar...

```
# liloconfig
```

... onde será apresentada uma tela com as instruções de como proceder passo-a-passo. Para alternarem entre as opções disponíveis no menu, utilizem a tecla <SETA_ACIMA> e <SETA_ABAIXO> para selecionar as opções do menu; <TAB> para posicionar o curso nas opções *Aceitar* ou *Cancelar*, onde <SETA_ESQ.> e <SETA_DIR.> alterna entre eles; e <ENTER> para definir as opções desejadas.



Tela de configuração do LILLO.



O *LILO* também é automaticamente instalado e definido durante a instalação do *Slackware*. Ao ser executado, deveremos definir qual o tipo de instalação. Seleccionem *simple* para uma instalação básica.

Estas e outras informações estão disponíveis na *3a. Parte: A instalação*. Consultem-na, para obterem maiores detalhes.

OPERAÇÕES MAIS FREQUENTES

Existe algumas operações ocasionalmente necessárias para a boa manutenção da inicialização do sistema, que dentre as quais, segue:

SELECIONAR O SISTEMA GNU/LINUX COMO PADRÃO

Após a inicialização do *LILO*, o sistema aguardará o usuário seleccionar qual será o sistema operacional a ser carregado. Por padrão, o *Windows* é carregado quando não é feita a escolha pelo usuário e o tempo de espera é consumido. Mas podemos alterar a opção padrão para que ela seja um sistema *GNU/Linux*. Para isto, acrescentem o parâmetro...

```
# LILO configuration file
# generated by 'liloconfig'
#
# Start LILO global section
append="hdd=ide-scsi"
boot = /dev/hda
message = /boot/boot_message.txt
prompt
default=[LABEL]
timeout = 1200
```

..., onde *[LABEL]* será o nome (rótulo) dado ao sistema descrito na seção de partições. O modo padrão é *Linux*. Em seguida, rodem o *lilo* novamente. Será exibida a seguinte mensagem.

```
# lilo
Added Windows
Added Linux *
# _
```

Significa que a operação foi realizada com sucesso.

MUDAR A RESOLUÇÃO DO FRAMEBUFFER

Podemos definir a resolução de vídeo manualmente, editando diretamente o arquivo */etc/lilo.conf* na seção global. Vamos supor que tivéssemos optado por utilizar a resolução de *1024x768x32k*:

```
# VESA framebuffer console @ 1024x768x32k
# vga=790
```

Ou então *800x600x32k*:



```
# VESA framebuffer console @ 800x600x32k
# vga=787
```

Basta apenas remarcar (ou apagar) o padrão atual (*vga = normal*) e desmarcar a opção desejada (*vga = [DESEJADO]*). Lembre-se que a placa de vídeo em uso deverá suportar este recurso, e o *monitor*, a resolução desejada. Após isto, salvem o arquivo e digitem na linha de comando...

```
# lilo
```

... para gravar um nova definição do *LILLO* no setor *MBR* do disco rígido com as alterações realizadas.

ADICIONAR MAIS UMA ENTRADA NO LILLO

Se fosse instalado mais um sistema operacional além do *Windows* e o *GNU/Linux* padrão, e quisessem incluí-lo no gerenciador, bastaria definir as informações necessárias dentro desta seção.

Vamos supor que resolvemos instalar uma outra versão de sistema *GNU/Linux* em um outro disco rígido, de um micro qualquer que estava encostado no canto. As instruções necessárias seria estas:

```
image = /[IMAGEM DO KERNEL]
root = /dev/[DISCO RÍGIDO]
label = [NOME DO SISTEMA]
read-only
```

Ou se fosse um outro sistema não *GNU/Linux*:

```
other = /dev/hdb1
label = [OUTRO SISTEMA]
table = /dev/hdb
```

Ao executar o *LILLO* para atualizar as alterações na *MBR*, será exibido o novo nome (*label*) do sistema operacional disponibilizado:

```
# lilo
Added DOS *
Added Linux
Added [NOVO SISTEMA]
# _
```

ADICIONAR UMA SENHA EXTRA

Sabendo-se da possibilidade de ter acesso ao sistema com a simples utilização do comando *linux single* na linha de comando do *LILLO*, a única forma de proteger-se deste inconveniente é atribuir uma senha de acesso. Para isto, insiram a seguinte linha no arquivo */etc/lilo.conf*:

```
# Start LILLO global section
boot = /dev/hda
message = /boot/boot_message.txt
```

```
password = [SENHA]
```

```
prompt
```

```
timeout = 1200
```

```
# Override dangerous defaults that rewrite the partition table:
```

Onde no lugar de [SENHA] deverá ser digitada a senha desejada. Lembre-se de que estes parâmetros deverão estar situados antes da linha *prompt*.

INICIALIZAR O SISTEMA EM MODO DE MANUTENÇÃO

Poderemos utilizar alguns parâmetros especiais na linha de comando do próprio *LILO* para inicializá-lo em modo de segurança. Assim, poderemos realizar certas tarefas para a sua manutenção.

```
boot: _
```

Ao inicializar o sistema, aguarde o carregamento da linha de comando do *LILLO*. Digitem o nome dado para a seleção do sistema (no caso padrão, *Linux*) e digite em seguida, *single* + <ENTER>:

```
boot: Linux single
```

Assim, o *kernel* será carregado em modo *single-user (init 1)*, montando tão somente a partição raiz e habilitando a conta de superusuário (*root*), propiciando um ambiente perfeito para a sua manutenção. Tal como acontece quando inicializamos o *Windows* em *Modo de Segurança*...

CRIAR UM DISCO DE RECUPERAÇÃO COM O LILO.CONF

Outra necessidade eventual está na utilização de um disco de recuperação que suporte o *LILLO*, caso este não possa selecionar o sistema a ser inicializado. Para isto, digitem na linha de comando...

```
# lilo -b /dev/[DEVICE]
```

Segue um exemplo prático, utilizando uma unidade de memória eletrônica:

```
# lilo -b /dev/sda1
```

PROBLEMAS MAIS FREQUENTES

Ao INVÉS DO SISTEMA INICIALIZAR, É EXIBIDO...

Ao invés do sistema inicializar, é exibido a seguinte cadeia de caracteres:

```
LI, LI-
```

```
... ou...
```

```
01 01 01
```

... indefinidamente, até o travamento. Isto ocorre devido a:

- Não-gravação e/ou não-atualização do *LILLO* na *MBR*;
- Substituição da posição das unidades de disco rígido após a

B

instalação - neste caso os *devices* dos discos rígidos não corresponderão com as definições gravadas inicialmente.

- Antivírus da *BIOS* ativado - Não somente nos sistemas *GNU/Linux*, como no próprio *Windows* a manutenção de um programa antivírus da *BIOS* acarreta diversos problemas de compatibilidade, em especial durante a instalação de programas. Desativem-no.

Estas são as principais causas do travamento na inicialização do *LILO*. Poderão existir quaisquer outras, mas as soluções poderão ser específicas, de acordo com a máquina utilizada. Para estas e outras circunstâncias, recomendamos elaborarem durante ou logo após a instalação do *Slackware* um disquete de inicialização.

REMOÇÃO DAS DEFINIÇÕES DO LILO NO SETOR MBR

Basicamente existem duas formas de remover as definições do *LILO* com a utilização dos interpretadores *MS-DOS*...

```
C:\> FDISK /MBR
```

... e *GNU BASH*...

```
# /sbin/lilo -U
```

Geralmente a remoção do *LILO* na *MBR* se faz quando há necessidade da desinstalação dos sistemas *GNU/Linux* ou utilização de discos rígidos os quais possuem ou tiveram um sistema *GNU/Linux* instalado.

RECUPERAR A SENHA DO SUPERUSUÁRIO

Em algumas circunstâncias poderão ocorrer a perda (esquecimento, erro de digitação) da senha do superusuário para a administração do sistema. Caso isto ocorra, simplesmente digitem na linha de comando do *LILO*...

```
LILO: linux single
```

O sistema será executado em modo monousuário, o qual com a utilização do comando *passwd*, poderemos alterar a senha original para um termo conhecido. Lembrem-se que este recurso somente funcionará caso não tenhamos definido a opção *password* na configuração do *LILO*. Neste caso deveremos ter em mãos a senha de acesso do *LILO* para que possamos iniciar o sistema em modo monousuário.

ATUALIZANDO AS ALTERAÇÕES DESEJADAS

Para atualizar qualquer modificação realizada no arquivo */etc/lilo.conf*, basta apenas executarmos o *LILO* e verificar a saída de vídeo:

```
# lilo
```

```
Added Windows
```

```
Added Linux *
```



_

Se não houver nenhuma mensagem de erro, significa que o *LILO* atualizou as configurações desejadas corretamente, onde na próxima inicialização elas estarão habilitadas. Caso contrário, reeditem novamente o arquivo de configuração e corrijam as alterações feitas, se necessário.

CONCLUSÃO

Apesar de simples e com uma aparência "*arcaica*" (como muitos dizem por aí), o *LILO* disponibiliza todas as funcionalidades necessárias para o gerenciamento de múltiplos sistemas operacionais do computador. Pode não ser tão flexível ou possuir maiores vantagens que a utilização de outros bons gerenciadores, porém devido as suas características únicas, seu uso é simples e sua manutenção, muito fácil! &;-D

B

X. GERENCIAMENTO DE PROGRAMAS

INTRODUÇÃO

Como qualquer outra distribuição *GNU/Linux*, o *Slackware* também possui seu sistema de gerenciamento de pacotes nativo, além de diversos outros utilitários que auxiliam a administração dos programas utilizados.

Neste capítulo, iremos conhecer as principais ferramentas, métodos e instruções para as necessidades mais rotineiras, além de informações gerais sobre os demais processos pertinentes.

A NOMENCLATURA DOS PACOTES

A nomenclatura dos arquivos empacotados obedecem a um formato especificado, tendo separadas suas definições apenas por hífen e ponto:

```
[NOME]-[VERSÃO]-[REVISÃO]-[PLATAFORMA].[FORMATO]
```

Utilizaremos como exemplo o pacote fictício *darkstar-1.0-1.i386.rpm*:

```
Nome:      darkstar
Versão:    1.0
Revisão:   1
Plataforma: i386
Formato:   Red Hat Package
```

Observem que a extensão de um arquivo é a identificação universal, herdada dos sistemas operacionais da *Microsoft* (*MS-DOS* e *Windows*), Independente do arquivo ser um pacote ou conter qualquer outro conteúdo. No caso acima, o formato *Red Hat Package* adota a extensão *.rpm*; já o formato utilizado pela distribuição *Debian* e baseados utilizam a extensão *.deb*, o *Slackware* por sua vez adota o padrão *.tgz*.

Em muitos casos, dada a compilação específica de um determinado aplicativo para uma distribuição baseada na *Red Hat*, são adicionados 2 caracteres para informar a distribuição do qual este pacote é compatível.

- *cl* -> *Conectiva Linux*;
- *mk* -> *Mandrake Linux*;

Para exemplo:

- *quake-1.1-6cl-i386.rpm*;
- *qt-2.2mk-i586.rpm*;

Existem outras nomenclaturas que poderão existir. Vejam alguns exemplos:

```
darkstar-0.6beta-i686.rpm
```

Trata-se de um pacote que se encontra na versão *0.6*, porém em fase de



testes (*beta*), compilado para a arquitetura *i686* e empacotado no formato *RPM*. Para máquinas de produção (que necessitam de programas estáveis), geralmente não é recomendável a instalação destes pacotes.

```
darkstar-1.0-noarch.tgz
```

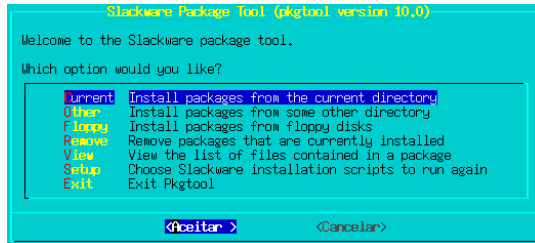
Pacote que se encontra na versão *1.0*, sem correção e que é compatível com qualquer arquitetura existentes (*noarch*), empacotado no formato nativo do *Slackware*. Somente pacotes que contêm o código-fonte ou arquivos textos das aplicações é que pertencem a esta categoria de arquitetura universal. O pacote de internacionalização do *KDE* e os fontes do *kernel* são exemplos.

FERRAMENTAS & MÉTODOS...

SLACKWARE PACKAGE TOOLS

✓ <<http://www.slackware.org/>>.

O gerenciador de pacotes padrão do *Slackware* é o *Slackware Package Tools*, um conjunto de ferramentas além do menu interativo que visa facilitar ao máximo o gerenciamento de pacotes.



Pkgtools.

Para ter acesso ao gerenciador, digitem na linha de comando...

```
# pkgtools
```

... onde será apresentada a tela principal da ferramenta.

Além do aplicativo, existem diversos utilitários de linha de comando que complementam o gerenciador, os quais são: *installpkg*, *upgradepkg*, *removepkg*, *makepkg* e *explodepkg*. Por este capítulo tratar de apenas instruções básicas para o uso em *desktops*, apenas descreveremos os três primeiros, já que atendem satisfatoriamente suas necessidades.

Para realizar a instalação de um pacote:

```
# installpkg [PACOTE]-[VERSÃO]-[ARQUITETURA]-[REVISÃO].tgz
```

Para realizar a atualização de um pacote:

```
# upgradepkg [PACOTE]-[VERSÃO]-[ARQUITETURA]-[REVISÃO].tgz
```



Para realizar a remoção de um pacote:

```
# removepkg [PACOTE]
```

Dependendo da forma com que o pacote foi compilado (especialmente por terceriso) talvez seja necessário informar como parâmetro, toda a nomenclatura do pacote, mas sem a extensão *.tgz*, para que o *removepkg* possa realizar a remoção.

RED HAT PACKAGE MANAGEMENT

✓ <<http://www.rpm.org/>>.

O *RPM* foi o primeiro gerenciador de pacotes desenvolvido para a instalação de programas pré-compilados.

```
RPM(8) Red Hat Linux RPM(8)
```

NAME

rpm - RPM Package Manager

SYNOPSIS

QUERYING AND VERIFYING PACKAGES:

rpm {-q|--query} [select-options] [query-options]

rpm {-V|--verify} [select-options] [verify-options]

rpm --import PUBKEY ...

rpm {-K|--checksig} [--nosignature] [--nodigest]
PACKAGE_FILE ...

INSTALLING, UPGRADING, AND REMOVING PACKAGES:

rpm {-i|--install} [install-options] PACKAGE_FILE ...

rpm {-U|--upgrade} [install-options] PACKAGE_FILE ...

rpm {-F|--freshen} [install-options] PACKAGE_FILE ...

rpm {-e|--erase} [--allmatches] [--nodeps] [--noscripts]

lines 1-39

Manual Eletrônico do RPM.

Criado pela *Red Hat*, é o gestor padrão da maioria das distribuições, além de ser um dos requerimentos estabelecidos pelo padrão *LSB*. Por isto o *Slackware* também suporta a instalação de pacotes no padrão *RPM*.

As sintaxes básicas necessárias para o uso do *RPM* são:

- Para instalar um pacote...

```
# rpm -ivh [PACOTE]-[VERSÃO]-[ARQUITETURA]-[REVISÃO].rpm --nodeps
```

- Para atualizar um pacote...

```
# rpm -Uvh [PACOTE]-[VERSÃO]-[ARQUITETURA]-[REVISÃO].rpm
```

- Para verificar um pacote...

B

```
# rpm -Vf [PACOTE]
```

- Para remover um pacote...

```
# rpm -e [PACOTE]
```

Lógico que haverá pequenas variações na definição das opções acima descritas de acordo com as circunstâncias. Neste caso, a consulta do manual eletrônico é mais que suficiente para sanar nossas dúvidas.

COMPILAÇÃO DO CÓDIGO-FONTE

Na necessidade de se instalar um pacote disponível somente em código-fonte, teremos que realizar manualmente a sua configuração, a compilação e por fim, a instalação. Apesar de não ser um processo tão fácil em comparação a instalação do pacote compilado, a maioria dos programas livres existentes utilizam os comandos básicos para esta finalidade.

Para que possamos compilar qualquer código-fonte, necessitaremos criar os *Makefiles*, que são arquivos que contém as instruções (parâmetros) necessárias para o sistema gerar os arquivos binários do programa desejado. Para isto, executem no diretório do pacote:

```
# ./configure --[OPÇÕES]
```

Lembrem-se de que em algumas circunstâncias - devido a natureza do pacote em questão - não será necessária a realização desta etapa, sendo apenas necessária as duas descritas logo à seguir.

Após realizarmos a configuração, deveremos compilar o pacote com...

```
# make
```

Baseado nas instruções geradas pelo *configure* e gravadas nos *Makefiles*, o comando *make* criará os arquivos binários necessários para compor o corpo do programa que desejamos instalar.

Terminada a compilação do programa, digitem na linha de comando...

```
# make install
```

... para concluir a instalação do programa. Poderemos também omitir a instrução *make* do processo de compilação e lançar diretamente o *make install*. Assim será feita a compilação e, logo em seguida, a instalação.

Para que seja desinstalado os binários de um arquivo gerados a partir do código-fonte, deveremos digitar...

```
# make uninstall
```

... lembrando que nem todos os programas possuem esta opção disponível.

OUTROS UTILITÁRIOS

Além de suportar os dois processos de instalação de pacotes, o *Slackware*



também possui utilitários que facilitam o gerenciamento destes, tais como o *rpm2tgz*, que converte os pacotes *RPM* para o formato nativo do *Slackware*, e o *script CheckInstall*, que possibilita empacotar os arquivos gerados pela compilação do código-fonte de aplicativos e utilitários.

CONCLUSÃO

Em virtude da necessidade de estarem disponíveis na parte *Conhecimentos Gerais* todas as instruções básicas para a utilização das informações nas partes posteriores deste livro, damos apenas uma pequena introdução básica ao que se refere a instalação de programas nos sistemas *GNU/Linux*. O principal objetivo está na necessidade dos usuários terem uma pequena base-técnica que os auxiliem no entendimento das instruções das demais partes do livro. Por isto, se quiserem se aprofundar mais, recomendamos que consultem a *5a. Parte - Gerenciamento de Programas*. &-D



XI. VARIÁVEIS DE SISTEMA

INTRODUÇÃO

Ao trabalhar com o *Windows*, em algumas situações necessitaremos lidar com algumas variáveis do sistema (como a *path*, por exemplo). Especialmente em relação a série *9X*, teremos que utilizar a linha de comando para realizar as intervenções necessárias.

Com estes mesmos conceitos, as variáveis dos sistemas *GNU/Linux* não só apresentam as mesmas funcionalidades conhecidas no sistema operacional habitual, como outras, das quais iremos conhecer neste capítulo.

AS VARIÁVEIS

PATH / ROOTPATH

Para aqueles que migraram do *MS-DOS* acreditamos que estão bem familiarizados com a variável *PATH*, muito utilizada para habilitar seus comandos em qualquer ponto (diretório) do sistema. *PATH* – pronuncia-se “páf” – é apenas o caminho de procura para arquivos executáveis. A variável armazena na memória do sistema caminhos para a localização de comandos e executáveis evocados em qualquer ponto do sistema para a execução.

Observem os caminhos */bin*, */sbin*, */usr/bin* e */usr/local/bin*: estas são as localizações dos comandos utilizados pelos usuários, comandos do administrador e evocados pelo próprio sistema, binários de aplicações do sistema e binários de aplicações instaladas que não fazem parte dos pacotes disponíveis na distribuição (*local*) respectivamente.

```
$ echo $PATH
/
usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/usr/games:/opt/www/htdig/bin:/usr/
lib/java/bin:/usr/lib/java/jre/bin:/opt/kde/bin:/usr/lib/qt-3.2.1/bin:/usr/sha
re/texmf/bin:
$ _
```

Isto quer dizer que, toda vez ao evocarmos um comando ou aplicação, o interpretador inicialmente irá buscar tais executáveis nos diretórios especificados pela variável *\$PATH*. Caso não seja encontrado, será exibido...

```
bash: [EXECUTÁVEL]: command not found
```

Se estas especificações não existissem, teríamos que especificar todo o caminho onde se encontra o executável:

```
$ /bin/free
```

```
$ /usr/bin/emacs
$ /usr/local/firefox/firefox
```

E para evocar uma aplicação instalada em um local exótico? Vejam o caso da *SDK Java*: para que possamos o executável, teríamos que digitar...

```
$ /usr/lib/java/bin/java [PARÂMETROS]
```

Para resolver este problema, bastaria apenas atualizar a variável *\$PATH* para conter o caminho do binário *java*, e assim digitar apenas...

```
$ java [PARÂMETROS]
```

Os sistemas *GNU/Linux* - neste caso, o *Slackware* - ao serem instalados possuem este caminho pré-configurados numa seção do arquivo */etc/profile*.

```
# Set the default system $PATH:
PATH="/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/usr/games"
```

Para que reconfiguremos toda vez que inicializar o sistema, termos que acrescentar o caminho aos já existentes. No caso do *Java*, seria...

```
# Set the default system $PATH:
PATH="/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/usr/games:/usr/lib/java/bin/"
```

Quanto a variável *ROOTPATH*, sem grandes mistérios, ela apenas define o caminho dos executáveis para o administrador do sistema.

HOME

Exibe o diretório atual (raíz) onde se encontra o usuário autenticado.

```
$ echo $HOME
/home/darkstar
$ _
```

OSTYPE

Exibe o sistema operacional (*kernel*) em uso.

```
$ echo $OSTYPE
linux-gnu
$ _
```

SHELL

Exibe qual o interpretador de comandos usado na seção.

```
$ echo $SHELL
/bin/bash
$ _
```



TERM

Já o *TERM* apenas indica o tipo de terminal utilizado no momento.

```
$ echo $TERM
xterm
$ _
```

USER

Exibe a conta de usuário autenticada naquele instante.

```
$ echo $USER
darkstar
$ _
```

COMANDOS RELACIONADOS

ECHO

Apenas exibe o valor (conteúdo) das variáveis em questão.

Sintaxe:

```
$ echo $[VARIÁVEL]
```

Exemplo:

```
$ echo $HOME
/home/darkstar
$ _
```

SET

O comando *set* é o responsável pela atualização das variáveis do sistema.

Sintaxe:

```
$ set [VARIÁVEL] [VALOR]"
```

Observem que o valor (conteúdo) da variável deverá se encontrar entre aspas duplas. Para habilitarmos um conteúdo para a variável *PATH*, deveremos utilizar...

```
$ set PATH /usr/local/tools"
```

Somente utilizando este comando, serão apagadas todas as definições anteriores. Para mantê-las, utilizamos a variável *\$PATH*. Com isto, poderemos acrescentar este valor utilizando o comando...

```
$ set PATH $PATH;/usr/local/tools;" [DEFINIÇÃO2];" [DEFINIÇÃO3]"
```

... sem apagar as definições anteriores. Lembrem-se que estas definições somente se encontrarão presentes na sessão atual do terminal.

Um detalhe interessante é que, quando o comando é digitado em um

B

terminal, são exibidos os valores correntes de todas as variáveis do sistema:

```
$ set
BASH=/bin/bash
BASH_VERSINFO=( [0]="2" [1]="05b" [2]="0" [3]="1" [4]="release" [5]="i486-
slackware-linux-gnu" )
BASH_VERSION='2.05b.0(1)-release'
COLORTERM=
COLUMNS=80
CPLUS_INCLUDE_PATH=/usr/lib/qt-3.2.1/include:/usr/lib/qt-3.2.1/include
DIRSTACK=(
DISPLAY=:0.0
...
$ _
```

EXPORT

Exporta valores de variáveis.

Sintaxe:

```
$ export [VARIÁVEL]=[VALOR]
$ export [VARIÁVEL]=[VALOR (COM ESPAÇOS)]"
$ export [VARIÁVEL]='[VALOR (COM ESPAÇOS)]'
```

Exemplo:

```
$ export JAVA_HOME=/usr/java/bin
```

Da mesma forma que é feito com o *set*, apagaremos as definições anteriores da variável. Se quisermos mantê-las, teremos que utilizar...

```
export JAVA_HOME=$JAVA_HOME:/usr/java/bin
```

Devemos lembrar que após estas definições, a variável atualizada estará presente apenas durante esta seção do comando. Tal como o comando *set*, quando o *export* é digitado em um terminal, serão exibidos os valores correntes de todas as variáveis do sistema.

INTERNACIONALIZAÇÃO

As variáveis de internacionalização são extremamente importantes para que possamos configurar determinados aplicativos e ambientes gráficos para a nossa língua nativa.

A forma clássica de ajustar uma variável é:

```
# export [VARIÁVEL]=[VALOR]
```

As principais variáveis são:

- *LC_ALL*: geral (define todas as variáveis de uma só vez);
- *LANG*: define o idioma local;
- *LC_MESSAGES*: exibição de mensagens dos aplicativos;



- *LC_TYPE*: layout do teclado e comportamento de teclas especiais.

Basicamente os valores que deveremos atribuir são as especificações de língua. Para definir os valores do *Brasil*, deveremos utilizar a *flag pt_BR*.

Segue um exemplo básico de como definir, com apenas o uso de um único comando, todas as variáveis de internacionalização do sistema:

```
# export LC_ALL=pt_BR
```

ARQUIVOS DE CONFIGURAÇÃO

São inúmeros os arquivos que definem as variáveis do sistema, sejam globais, para cada aplicação ou um usuário específico. Aqui descreveremos apenas aqueles aplicáveis para as necessidades de um usuário *desktop*.

/ETC/PROFILE

O arquivo */etc/profile* contém as definições globais utilizadas pelo sistema em geral. Qualquer parâmetro aqui definidos irão refletir em quaisquer seções e contas de acesso autenticados ao sistema.

Cabeçalho...

```
# /etc/profile: This file contains system-wide defaults used by
# all Bourne (and related) shells.
```

Variáveis de ambiente...

```
# Set the values for some environment variables:
export MINICOM="-c on"
export MANPATH=/usr/local/man:/usr/man:/usr/X11R6/man
export HOSTNAME=`cat /etc/HOSTNAME`
export LESSOPEN="|lesspipe.sh %s"
export LESS="-M"
```

```
# If the user doesn't have a .inputrc, use the one in /etc.
if [ ! -r "$HOME/.inputrc" ]; then
    export INPUTRC=/etc/inputrc
fi
```

Caminhos dos executáveis...

```
# Set the default system $PATH:
PATH="/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/usr/games"
```

Caminhos dos executáveis (superusuário)...

```
# For root users, ensure that /usr/local/sbin, /usr/sbin, and /sbin are in
# the $PATH. Some means of connection don't add these by default (sshd comes
# to mind).
if [ "`id -u`" = "0" ]; then
    echo $PATH | grep /usr/local/sbin 1> /dev/null 2> /dev/null
    if [ ! $? = 0 ]; then
        PATH=/usr/local/sbin:/usr/sbin:/sbin:$PATH
```



```

fi
fi
Definições do terminal...
# I had problems using 'eval tset' instead of 'TERM=', but you might want to
# try it anyway. I think with the right /etc/termcap it would work great.
# eval `tset -sQ "$TERM"`
if [ "$TERM" = "" -o "$TERM" = "unknown" ]; then
    TERM=linux
fi

```

Definições do interpretador *ksh93*...

```

# Set ksh93 visual editing mode:
if [ "$SHELL" = "/bin/ksh" ]; then
    VISUAL=emacs
# VISUAL=gmacs
# VISUAL=vi
fi

```

Formato do sinal de prontidão...

```

# Set a default shell prompt:
#PS1='`hostname` : `pwd` # '
if [ "$SHELL" = "/bin/pdksh" ]; then
    PS1='! $ '
elif [ "$SHELL" = "/bin/ksh" ]; then
    PS1='! ${PWD/#$HOME/~}$ '
elif [ "$SHELL" = "/bin/zsh" ]; then
    PS1='%n@m:%m:%~%# '
elif [ "$SHELL" = "/bin/ash" ]; then
    PS1='$ '
else
    PS1='\u@\h:\w\$ '
fi
PS2='> '
export PATH DISPLAY LESS TERM PS1 PS2

```

Permissões de acesso padrão para a criação de arquivos...

```

# Default umask. A umask of 022 prevents new files from being created group
# and world writable.
umask 022

```

Definição de cores para o interpretador de comandos...

```

# Set up the LS_COLORS and LS_OPTIONS environment variables for color ls:
if [ "$SHELL" = "/bin/zsh" ]; then
    eval `dircolors -z`
elif [ "$SHELL" = "/bin/ash" ]; then
    eval `dircolors -s`
else
    eval `dircolors -b`
fi

```

Notificação de recebimento de correio...

```

# Notify user of incoming mail. This can be overridden in the user's
# local startup file (~/.bash.login or whatever, depending on the shell)
if [ -x /usr/bin/biff ]; then

```



```
biff y
fi
```

Definições adicionais dos scripts *profile*...

```
# Append any additional sh scripts found in /etc/profile.d/:
for file in /etc/profile.d/*.sh ; do
    if [ -x $file ]; then
        . $file
    fi
done
```

Definições de caminhos para usuários comuns...

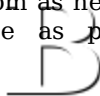
```
# For non-root users, add the current directory to the search path:
if [ ! "`id -u`" = "0" ]; then
    PATH="$PATH:."
fi
```

O DIRETÓRIO /ETC/PROFILE.D/

O diretório */etc/profile.d* possui definições de *scripts* adicionais para diversos ambientes gráficos e aplicações.

```
# ls -l /etc/profile.d/
total 88
-rwxr-xr-x 1 root root 164 2003-03-14 01:00 bsd-games-login-
fortune.csh*
-rwxr-xr-x 1 root root 141 2003-03-14 01:00 bsd-games-login-
fortune.sh*
-rwxr-xr-x 1 root root 32 2003-01-11 06:27 gtk+.csh*
-rwxr-xr-x 1 root root 43 2003-01-11 06:28 gtk+.sh*
-rwxr-xr-x 1 root root 102 2000-10-30 23:42 htdig.csh*
-rwxr-xr-x 1 root root 101 2000-10-30 23:43 htdig.sh*
-rwxr-xr-x 1 root root 146 2003-09-12 21:00 j2sdk.csh*
-rwxr-xr-x 1 root root 145 2003-09-12 21:00 j2sdk.sh*
-rwxr-xr-x 1 root root 176 2003-09-15 03:29 kde.csh*
-rwxr-xr-x 1 root root 85 2003-09-15 03:29 kde.sh*
-rwxr-xr-x 1 root root 227 2003-03-10 03:30 lang.csh*
-rwxr-xr-x 1 root root 225 2003-03-10 03:31 lang.sh*
-rwxr-xr-x 1 root root 51 2003-02-10 04:25 mc.csh*
-rwxr-xr-x 1 root root 45 2003-02-10 04:25 mc.sh*
-rwxr-xr-x 1 root root 31 2003-01-16 02:42 metacity.csh*
-rwxr-xr-x 1 root root 31 2003-01-16 02:41 metacity.sh*
-rwxr-xr-x 1 root root 443 2003-09-14 13:59 qt.csh*
-rwxr-xr-x 1 root root 396 2003-09-14 13:59 qt.sh*
-rwxr-xr-x 1 root root 50 2002-10-22 02:13 tllib.csh*
-rwxr-xr-x 1 root root 63 2002-10-22 02:13 tllib.sh*
-rwxr-xr-x 1 root root 134 2000-04-24 19:46 tetex.csh*
-rwxr-xr-x 1 root root 118 2000-04-24 19:46 tetex.sh*
# _
```

Cada arquivo possui definições de variáveis especiais para cada aplicação, de acordo com a nomenclatura dos mesmos. Com apenas uma visualização em seu conteúdo e, de acordo com as necessidades, deveremos realizar a edição direta destas para que as propriedades desejadas estejam



presentes nas próximas seções.

~/**.BASHRC**

Para aqueles usuários que preferem definir personalizações especiais (e por isto não podem alterar as definições globais */etc/profile*) para as suas necessidades, eles poderão especificá-las em um arquivo chamado *.bashrc*. No *Slackware*, este não existe, sendo necessário criá-lo manualmente.

```
~$ mcedit .bashrc
```

A partir daí é só acrescentar as definições desejadas.

```
#!/bin/bash  
[DEFINIÇÕES PERSONALIZADAS]
```

Lembrem-se que a definição *#!/bin/bash* é opcional.

CONCLUSÃO

Existem diversas variáveis e opções de ajustes através da linha de comando disponíveis nos sistemas *GNU/Linux*. Definir e exemplificá-las aqui, além de trabalhoso, seria desnecessário, visto que a grande maioria dos simples usuários sequer têm necessidade delas, quanto mais vontade de intervir em ajustes desta categoria. Por isto optamos simplesmente por colocar apenas as variáveis de maior relevância a categoria destes usuários. &;-D

