Guia de Referências do Linux

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A

- **adduser** - cria um novo usuário.
- **alias** - permite nomear um conjunto de comandos, a ser executado pelo sistema, por um único nome.
- **apropos** - informa quais os comandos do Linux possuem, em sua definição, uma determinada expressão.
- **ar** - inclui/atualiza/extrai/deleta arquivos de/em um repositório de arquivos.
- **at** - agenda tarefas a serem executadas pelo sistema.
- **atq** - lista as tarefas agendadas com o comando **at**.
- **atrm** - remove tarefas agendadas.
- **awk** - linguagem de processamento e procura de padrões.

B

- **bg** - faz um programa, que está executando em primeiro plano, passar a executar em segundo plano.

C

- **cal** - exibe um calendário simplificado.
- **cat** - concatena e/ou exibe um ou mais arquivos.
- **cd** - altera o diretório atual de trabalho do usuário.
- **chfn** - altera as informações apresentadas pelo utilitário **finger**.
- **chgrp** - altera o grupo de arquivos/diretórios.
- **chmod** - altera as permissões de acesso de arquivos/diretórios.
- **chown** - altera o dono e o grupo de arquivos/diretórios.
- **chsh** - altera o **shell** padrão do usuário.
- **clear** - limpa a tela do terminal.
- **comandos do linux** - explica o formato dos comandos do Linux.
- **compress** - compacta/descompacta arquivos.
- **consoles virtuais** - interface do Linux com os usuários.
- **cotas de disco** - define a quantidade de espaço em disco que cada usuário pode usar.
- **cp** - copia arquivos e diretórios.
- **crontab** - agenda tarefas para serem executadas periodicamente.
- **curingas** - *, ? e [ ]
- **cut** - seleciona colunas de um arquivo txt ou da entrada padrão.
- **cvs** - aplicativo para gerenciamento de código-fonte.

D

- **date** - exibe ou modifica a data e a hora do sistema.
- **dd** - converte e copia um arquivo.
- **depmod** - produz arquivo contendo as dependências de módulo.
- **df** - mostra o espaço livre das partições.

- **diff** - compara dois arquivos, em formato texto, linha a linha.
- **DISPLAY** - variável de ambiente que define onde a saída de uma aplicação gráfica deve aparecer.
- **dmesg** - mostra as mensagens da última inicialização do sistema.
- **du** - informações sobre o uso do disco.
- **dvips** - converte arquivo **.dvi** em **.ps** (PostScript).

E

- **echo** - mostra o conteúdo dos diretórios.
- **edquota** - editor do sistema de cotas.
- **emacs** - editor de textos distribuído pela *Free Software Foundation*.
- **env** - executa um programa/comando em um ambiente modificado.
- **exit** - encerra execução do **shell** atualmente em uso pelo usuário.
- **export** - comando do **shell** que permite configurar e compartilhar variáveis de ambiente entre diversos programas e bibliotecas acessados a partir do mesmo terminal.

F

- **fdformat** - formatação de baixo nível em disquete.
- **fdisk** - aplicativo utilizado para particionar discos.
- **fg** - faz um programa, que está sendo executado em segundo plano (**background**), passar a ser executado em primeiro plano.
- **file** - determina o tipo do arquivo.
- **find** - pesquisa arquivos em uma hierarquia de diretórios.
- **finger** - exibe informações sobre um usuário.
- **fortune** - aplicativo que exibe uma citação aleatória.
- **free** - exibe a quantidade de memória livre/usada no sistema.
- **fsck** - verifica e repara um sistema de arquivos.
- **fuser** - identifica os processos que estão usando um determinado arquivo e/ou diretório.

G

- **gcc** - compilador de C para Linux.
- **gdb** - aplicativo para depuração de programas C, C++ e Modula-2.
- **getty** - configura o modo de funcionamento, velocidade e comportamento da linha.
- **ghostview** - aplicativo para visualização de arquivos **.ps** (PostScript) e **PDF**.
- **GID** - número de identificação de grupo para o **kernel** do Linux.
- **grep** - procura padrões em um arquivo.
- **groupadd** - cria um novo grupo.
- **groupdel** - deleta um grupo.
- **groupmod** - modifica um grupo.
- **groups** - lista os grupos aos quais um usuário pertence.
- **grpconv** - usa o sistema **gshadow** para proteger as senhas dos grupos.
- **grpunconv** - elimina o uso do sistema **gshadow** de proteção de senhas.
- **gunzip** - descompacta arquivos.
- **gv** - mesmo que **ghostview**.
- **gzip** - compacta/descompacta arquivos.

H

- **head** - exibe o início de um arquivo texto.
- **help** - exibe informações sobre um comando.
- **hostname** - mostra ou muda o nome do computador na rede.
- **hwclock** - exibe ou modifica a data e a hora do relógio da máquina.

I

- **id** - identifica os **UIDs** e **GIDs** efetivos e reais.

- **ifconfig** - configura uma interface de rede.
- **init** - processo de controle da inicialização do sistema.
- **inode** - identificador de diretório/arquivo em um sistema de arquivos.
- **insmod** - carrega módulos do **kernel** na memória do sistema.
- **ipcrm** - remove recurso **ipc** (inter-process communication).
- **ipcs** - fornece informações sobre recursos **ipc** (inter-process communication).
- **ispell** - ferramenta para correção ortográfica.

J

- **jobs** - mostra os processos executando em segundo plano (**background**).

K

- **kernel** - núcleo do Linux.
- **kill** - envia um determinado sinal a um processo em execução no sistema.
- **killall** - envia um determinado sinal a um conjunto de processos que usam o mesmo nome.
- **kudzu** - ferramenta que detecta e configura dispositivos de hardware.

L

- **last** - exibe todas as informações referentes a entrada (login) e saída (logout) de usuários do sistema.
- **lastlog** - exibe informações referentes ao último login de cada usuário.
- **latex** - gera arquivo **.dvi** a partir da definição de macros **TEX**.
- **ldconfig** - atualiza os links para as bibliotecas compartilhadas.
- **less** - permite fazer a paginação de arquivos ou da entrada padrão.
- **lesskey** - personaliza as teclas usadas no paginador **less**.
- **lilo** - carregador de inicialização do **Linux**.
- **linux_logo** - aplicativo que exibe os logotipos do **Linux**.
- **linuxconf** - ferramenta de administração do **Linux**.
- **ln** - cria ligações entre arquivos.
- **locate** - lista os arquivos cujos nomes coincidem com o padrão informado.
- **login** - processo responsável por permitir ou não o acesso de usuários ao sistema.
- **lpc** - aplicativo para controle de impressão de arquivos.
- **lpq** - examina o conteúdo da fila de impressão.
- **lpr** - imprime arquivos.
- **lprm** - remove arquivos da fila de impressão.
- **ls** - lista o conteúdo de um diretório.
- **lsmod** - lista os módulos do **kernel** que estão carregados na memória.

M

- **make** - utilitário para verificação e compilação de grupo de arquivos.
- **MAKEDEV** - script usado para criar/alterar/deletar dispositos em **/dev**.
- **makewhatis** - cria a base de dados makewhatis.
- **man** - formata e apresenta páginas do manual on-line.
- **mc** - aplicativo para manipulação de arquivos e diretórios.
- **mesg** - habilita/desabilita o recebimento de mensagens de outros usuários.
- **mingetty** - **getty** mínimo para consoles virtuais.
- **mkbootdisk** - cria um disco de inicialização (emergência) do sistema.
- **mkdir** - cria diretórios.
- **mke2fs** - formata uma partição de disco usando o sistema de arquivos **ext2**.
- **mkfs** - constrói um sistema de arquivos Linux.
- **mkswap** - configura uma área de troca (**swap**) do **Linux**.
- **modprobe** - carrega módulos e verifica as dependências dos módulos.
- **módulos** - peças de código de objeto que podem ser carregados em um **kernel** em operação.
- **more** - permite fazer a paginação de arquivos ou da entrada padrão.

- **mount** - monta sistemas de arquivos.
- **mouseconfig** - aplicativo para configurar o mouse.
- **mpage** - permite imprimir várias páginas em uma única folha de papel.
- **mt** - controla unidades de fita.
- **mtools** - utilitários para acessar discos DOS no UNIX.
- **mv** - move (ou altera o nome de) arquivos.

N

- **newgrp** - muda, temporariamente, o grupo (**GID**) do usuário.
- **nice** - configura a prioridade de execução de um comando ou de um programa.
- **nl** - exibe o conteúdo de um arquivo enumerando as linhas.
- **nohup** - executa um comando imune a interrupções de conexão.

O

P

- **passwd** - altera a senha de um usuário.
- **permissão de acesso** - autorização para ler/gravar/executar um arquivo/diretório do sistema.
- **pico** - editor de texto baseado no sistema de mensagens **pine**.
- **PID** - número de identificação (**ID**) de um processo.
- **pr** - formata arquivos de texto para impressão.
- **printenv** - mostra as variáveis de ambiente utilizadas pelo sistema.
- **processo** - programa em execução.
- **ps** - exibe informações sobre os processos que estão executando na máquina.
- **pwconv** - usa o sistema **shadow** para proteger as senhas dos usuários.
- **pwd** - exibe o nome do diretório atual.
- **pwunconv** - elimina o uso do sistema **shadow** de proteção de senhas.

Q

- **quota** - fornece informações sobre o sistema de cotas.
- **quotaoff** - desabilita o sistema de cotas.
- **quotaon** - habilita o sistema de cotas.

R

- **RCS** - aplicativo para gerenciamento de código-fonte.
- **reboot** - reinicializa o computador.
- **redirecionadores de E/S** - **>**, **>>**, **<** e **|**.
- **renice** - altera a prioridade de um processo que está executando.
- **rev** - inverte as linhas de um arquivo.
- **rlogin** - inicia uma sessão de terminal remoto.
- **rm** - remove arquivos/diretórios.
- **rmdir** - remove diretórios vazios.
- **rmmod** - descarrega módulos do **kernel** da memória do sistema.
- **root** - administrador do sistema ou superusuário.
- **rpm** - gerenciador de pacotes da distribuição **Red Hat**.

S

- **separador de comandos** - **;** e **;**
- **set** - exibe a lista das variáveis de ambiente.
- **setserial** - exibe ou modifica as definições sobre porta serial.
- **setterm** - configura os atributos do terminal no modo texto.
- **shell** - interpretador de comandos do **Linux**.
- **shutdown** - encerra/reinicializa o sistema.

- **sistemas de arquivos** - como os arquivos e os diretórios são organizados no **Linux**.
- **sort** - ordena as linhas de arquivos textos.
- **source** - atualiza arquivos do sistema.
- **split** - divide um arquivo em arquivos menores.
- **strfile** - cria um arquivo de acesso aleatório para armazenamento de strings.
- **strings** - extrai strings de arquivos binários.
- **stty** - modifica e/ou mostra as configurações de linhas de terminal.
- **su** - executa um **shell** com substituição de usuário e grupo.
- **sudo** - executa um comando usando os privilégios de um outro usuário.
- **swap** - partição do disco usada como memória auxiliar (área de troca).
- **swapoff** - desabilita dispositivos e arquivos para paginação e troca.
- **swapon** - habilita dispositivos e arquivos para paginação e troca.
- **sync** - grava os dados da memória nas unidades de disco.

T

- **tac**- concatena e exibe um ou mais arquivos na ordem inversa.
- **tail** - exibe as últimas linhas de um arquivo texto.
- **talk** - permite conversa em tempo real entre dois usuários.
- **tar** - armazena e extrai arquivos de um arquivo **tar**.
- **teclas especiais** - conjunto de teclas com características especiais no Linux.
- **tee** - ler da entrada padrão e grava na saída padrão.
- **telinit** - processo de controle da inicialização do sistema.
- **telnet** - permite acessar uma máquina remotamente.
- **tempo de época** - data e hora que o Linux considera como o início dos tempos.
- **tex**- sistema de processamento de textos para formatação de documentos.
- **time** - mede o tempo necessário para executar um comando/aplicativo.
- **top** - utilitário que lista, em tempo real, os processos que estão usando a cpu.
- **touch** - muda a data e a hora do último acesso/modificação de um arquivo.
- **tr** - apaga ou altera caracteres.
- **tree** - apresenta, em formato de árvore, o conteúdo de um diretório.
- **type** - exibe o tipo de um arquivo.

U

- **UID** - número de identificação do usuário para o **kernel** do Linux.
- **umask** - define as permissões que não estão disponíveis aos usuários do sistema.
- **umount** - desmonta sistemas de arquivos.
- **uname** - exibe informações sobre o sistema.
- **uniq** - remove as linhas duplicadas de um arquivo ordenado.
- **unset** - apaga uma variável de ambiente.
- **updatedb** - atualiza banco de dados de nome de arquivos.
- **uptime** - diz há quanto tempo o sistema está funcionando.
- **useradd** - cria um novo usuário.
- **userdel** - remove usuário e seus respectivos arquivos do sistema.
- **usermod** - modifica uma conta de usuário do sistema.
- **users** - mostra os usuários que estão atualmente conectados ao sistema.

V

- **variáveis de ambiente** - conjunto de variáveis usadas para definir o ambiente de trabalho dos usuários do sistema.
- **vigr** - edita o arquivo **/etc/group**.
- **vim** - editor de textos em formato ASCII.
- **vipw** - edita o arquivo **/etc/passwd**.
- **visudo** - edita o arquivo **/etc/sudoers**.

W

- **w** - informa quais os usuários que estão conectados e o que eles estão executando.
- **wc** - conta linhas, palavras e caracteres de arquivos.
- **whatis** - mostra um resumo rápido sobre um ou mais comandos.
- **whereis** - lista as localizações de programas binários, fontes e documentações.
- **who** - informa quais os usuários que estão conectados.
- **whoami** - fornece a identificação efetiva do usuário.

X

- **X ou X Window** - é a interface gráfica (GUI) padrão do Linux.
- **xhost** - define quais máquina podem acessar o servidor X.
- **xman** - aplicativo gráfico de exibição da documentação do **Linux** para o sistema **X Window**.

Y

Z

- **zgrep** - procura padrões em um arquivo compactado.
- **zip** - compacta arquivos.

adduser ou useradd

adduser [opções] usuário

São algumas das opções deste comando

-d diretório : define o diretório **home** do usuário.

-e mm/dd/yy : data de expiração da conta do usuário.

-g grupo : especifica o **GID** do grupo padrão do usuário.

-G grupo1[,grupo2, ...] : especifica o GID dos outros grupos aos quais o usuário pertence.

-s shell : especifica o **shell** padrão do usuário.

-u uid : especifica o **UID** do usuário.

Comentários sobre as opções do comando

Para criar o usuário **aluno**,basta digitar

adduser aluno

Neste caso, o diretório home do usuário **aluno** é **/home/aluno**. Para definir **/home/meudir** como o diretório **home** do usuário **aluno**, devemos escrever

adduser -d /home/meudir aluno

O **UID** do novo usuário corresponde ao menor número, maior que 500, que ainda não está alocado. O **root**, entretanto, pode definir o **UID** de um determinado usuário usando a opção **-u**. Por exemplo,

adduser -u 600 aluno

aloca o **UID** 600 para o usuário **aluno**. Este número deve ser único a menos que a opção **-o** também seja definida. O comando

adduser -u 600 -o aluno

aloca o **UID** 600 para o usuário **aluno**, mesmo que este **UID** já esteja alocado a outro usuário.

É possível também definir os grupos aos quais o usuário pertence. Por exemplo,

adduser -g 600 -G 500,68 aluno

define que o grupo padrão do usuário **aluno** tem **GID** 600, além disso o usuário também pertence aos grupos com **GID** 500 e 68. Quando um novo usuário é criado e o grupo padrão do novo usuário não é fornecido, o sistema automaticamente cria um novo grupo para este usuário com **GID** igual ao número de **UID**. Caso o **GID** já esteja alocado a outro grupo, o sistema escolhe o menor valor, maior que o valor de **UID**, que esteja disponível.

Descrição

O comando **adduser** cria uma entrada para o usuário no arquivo **/etc/passwd**. Esta entrada tem o nome, a senha (caso seja usado o sistema **shadow** para criptografar as senhas, apenas um**x** ou um ***** é exibido neste campo, a senha criptografada é armazenada no arquivo **/etc/shadow**), **UID** (user identification), **GID** (group identification), informações sobre o usuário (inicialmente está vazio), o diretório **home** do usuário e o **shell** padrão do usuário. Por exemplo, a linha abaixo mostra a entrada criada em **/etc/passwd** para o usuário **aluno** que tem **UID** e **GID** iguais a 501 e usa o **shell bash** como padrão.

aluno:x:501:501::/home/aluno:/bin/bash

Caso seja criado um novo grupo durante o processo de criação de uma conta de usuário, o comando **adduser** cria uma entrada para o grupo no arquivo **/etc/group**. Esta entrada tem o nome do grupo, a senha criptografada do grupo, o **GID** do grupo e a lista dos usuários que são membros do grupo, apesar de este não ser o grupo padrão destes usuários. Por exemplo, a linha abaixo mostra a entrada criada em **/etc/group** para o grupo **aluno** que tem **GID** igual a 501 e onde o usuário **maria** é membro.

aluno:x:501:maria

É importante observar que o comando **adduser** também cria, automaticamente, o **diretório home** do usuário e coloca, neste diretório, os arquivos que o **shell** precisa para inicializar a conta do usuário. Os arquivos que são colocados automaticamente pelo sistema na conta do usuário podem ser vistos no diretório **/etc/skel**.

Observações

A configuração padrão usada pelo comando **adduser** é definida em **/etc/default/useradd** e em **/etc/login.defs**.

alias

alias [nome alternativo='comandos']

Comentários sobre o comando

Por exemplo, o comando

alias fd='mount /dev/fd0 /mnt/floppy; cd /mnt/floppy && ls'

define que toda vez que o usuário entrar com o comando **fd**, três ações devem ser executadas pelo sistema:

1. **mount /dev/fd0 /mnt/floppy** - é montada a unidade de disquete em /mnt/floppy;
2. **cd /mnt/floppy** - o diretório atual do usuário passa a ser /mnt/floppy;
3. **ls** - o sistema exibe o conteúdo do diretório /mnt/floppy.

Note que, neste exemplo, foram usados dois diferentes separadores de comandos: **ponto-e-vírgula** e **&&**. Comandos separados por **";"** são executados em sequência. Comandos separados por **&&** são executados de forma condicional, ou seja, o comando após o separador só é executado se o comando anterior tiver sido executado com sucesso.

Para obter mais informações sobre os sepadores de comandos, consulte o manual on-line. Por exemplo, se você usa o **shell bash**, digite

man bash

Observações

É importante observar que a definição feita com o comando **alias** não é permanente. Isto significa, que na próxima vez que você acessar o sistema, ela não mais existirá. Para criar uma definição permanente, deve-se incluir o comando **alias** em um dos arquivos carregados pelo **shell** durante o processo de inicialização. Por exemplo, o **shell bash** ler dois arquivos com as definições: **.bashrc** e **/etc/bashrc**. O primeiro arquivo possui as definições feitas pelo próprio usuário, enquanto o segundo arquivo possui as definições do sistema e que podem ser usadas por todos os usuários.

Para usar uma definição, imediatamente após a sua inclusão em **.bashrc**, sem a necessidade de sair do sistema e entrar novamente, basta digitar

source .bashrc

Para finalizar, convém notar que o comando **alias**, sem parâmetros, exibe a lista das definições que estão disponíveis ao usuário.

apropos

apropos expressão

Comentários sobre o comando

Por exemplo, o comando

apropos debugger

pode fornecer a seguinte saída no seu sistema

debugfs (8) - ext2 file system debugger
efence (3) - Electric Fence Malloc Debugger
gdb (1) - The GNU Debugger

que corresponde aos comandos que possuem a palavra **debugger** (depuração) na sua definição. O número, ao lado do comando, especifica o nível de execução do comando no sistema (vejaman para obter mais informações sobre a classificação dos comandos em níveis de execução).

Na realidade, o comando **apropos** pesquisa a base de dados **whatis** criada pelo **root** com o comando **makewhatis**.

ar

ar [[-] opções] repositório arquivos

São algumas opções deste comando

d : deleta arquivo(s) do repositório.

r : copia arquivo para o repositório e, se necesssário,i substitui a versão que já existe no repositório.

t : lista os arquivos existentes no repositório.

tv : lista informações mais detalhadas sobre os arquivos do repositório.

x : extrai arquivos do repositório.

Comentários sobres as opções do comando

Por exemplo,

ar r teste *.html

cria o repositório **teste** e copia todos os arquivos com extensão **html** do diretório atual para o repositório. Para ver a lista dos arquivos deste repositório, basta digitar

ar t teste

Não é obrigatório, com comando **ar**, o uso do sinal "-" antes das opções. Portanto, também podemos digitar o comando acima da seguinte forma

ar -t teste

at

at [opções] time

onde **time** corresponde ao horário em que a tarefa deverá ser executada.

time

São aceitas horas no formato HHMM ou no formato HH:MM. Outras opções válidas para horas são **midnight** (meia-noite), **noon** (meio-dia), **teatime** (hora do chá, ou seja 4:00 PM) e **now**(agora).

Junto com a hora pode-se também especificar o dia da tarefa no formato MMDDAA, MM/DD/AA ou MM.DD.AA. Além disso, é possível também definir datas como **today** (hoje) e **tomorrow**(amanhã).

Outra forma de definir o horário de execução de uma tarefa é especificar uma hora mais um contador de tempo. Por exemplo, **8:00 + 3 days** marca a tarefa para ser executada daqui a 3 dias às 8:00 horas da manhã. Pode-se usar como contador de tempo os termos **minutes** (minutos), **hours** (horas), **days** (dias) e **weeks** (semanas).

São algumas das opções deste comando

-c tarefa : exibe o conteúdo da tarefa especificada.

-d : é um alias para o comando **atrm**.

-f arquivo : a tarefa a ser executa está descrita no arquivo especificado.

-l : é um alias para o comando **atq**.

-m : envia um e-mail para o usuário quando a tarefa for concluída.

Comentários sobre as opções do comando

Para executar algumas tarefas às 20:00 de amanhã,basta digitar

at 20:00 tomorrow

Neste caso, é aberto um editor de linhas para que o usuário entre com os comandos. Pode-se digitar um comando por linha e dar ENTER após cada comando ou pode-se digitar vários comandos por linha, separando-os por ponto-e-vírgula. Para encerrar o editor, deve-se digitar **CTRL+D**.

É também possível criar um arquivo com todos os comandos a serem executados e pedir que o comando **at** o execute na hora desejada. Para o exemplo acima, se os comandos fossem colocados dentro do arquivo **teste**, teríamos apenas que digitar

at -f teste 20:00 tomorrow

Para ver as tarefas agendadas, digite

at -l

O comando acima informa o número da tarefa agendada, a data e a hora programada para a execução. Suponha que a seguite saída seja fornecida.

5 2001-09-26 20:00 a

Isto significa que o usuário possui apenas uma tarefa agendada. Esta tarefa será executada em 26/09/01 às 20:00. Note que o número de identificação desta tarefa é 5. Para ver o conteúdo desta tarefa, basta digitar

at -c 5

E para remover esta tarefa do sistema é só entrar com o seguinte comando

at -d 5

Observações

O root pode usar o comando **at** sem restrições. Para os outros usuários a permissão para usar este comando é determinada pelos arquivos **/etc/at.allow** e **/etc/at.deny**.

As tarefas agendadas ficam armazenadas em **/var/spool/at**.

O **daemon** responsável pela execução das tarefas agendadas pelo comando **at** é o **atd**. O script do **atd** fica armazenado em **/etc/rc.d/init.d**.

A tarefa agendada com o comando **at** é executada apenas uma vez. Para agendar tarefas que devem ser executadas periodicamente, use o comando **crontab**.

atq

atq

Comentário

O comando **atq** informa o número da tarefa agendada, a data e a hora de execução da tarefa.

Observações

O comando **atq** é um alias para o comando **at -l**.

As tarefas agendadas ficam armazenadas em **/var/spool/at**.

Comandos relacionados

- **at** : agenda tarefas.
- **atrm** : remove tarefas agendadas.

atrm

at tarefa

onde **tarefa** corresponde ao número de identificação da tarefa agendada e que pode ser obtido com o comando **atq** ou com o comando **at -l**.

Comentários sobre o comando

Suponha que a seguite saída seja fornecida com o comando **atq** ou com o comando **at -l**.

5 2001-09-26 20:00 a

Isto significa que o usuário possui apenas uma tarefa agendada. Esta tarefa será executada em 26/09/01 às 20:00. Note que o número de identificação desta tarefa é 5. Para remover esta tarefa, basta digitar,

atrm 5

Observações

As tarefas são agendadas com o comando **at** e ficam armazenadas em **/var/spool/at**.

awk

awk [-F<c>] padrões arquivo

ou
awk [-F<c>] -farq_padrões arquivo
onde

- **-F** : opção do comando **awk** para definir separador de campos.
- **c** : é o caractere especificado como separador de campos.
- **padrões** : correspondem ao conjunto de padrões a serem usados para processar o arquivo de saída.
- **-f** : opção do comando **awk** para especificar o nome do arquivo que possui o conjunto de padrões.
- **arq_padrões** : é o arquivo com o conjunto de padrões a serem usados para processar o arquivo de entrada.
- **arquivo** : é o arquivo de entrada que será processado pelo awk.

Comentários sobre as opções do comando

Suponha que queremos obter a lista dos usuários do sistema e o **shell** usado por cada um dos usuários. Estas duas informações fazem parte do arquivo **/etc/passwd** onde existe uma entrada para cada usuário do sistema com as seguintes informações: nome, senha, UID, GID, informações do usuário, diretório home e o **shell** padrão. Por exemplo, a linha abaixo mostra a entrada para o usuário **aluno** que usa o **shell bash** como padrão.

aluno:x:501:501::/home/aluno:/bin/bash

Note que os campos são separados pelo símbolo de dois pontos (":"). Podemos então, digitar o seguinte comando para obter os nomes e os shells dos usuários:

```
awk -F:  '{print $1 " => " $7}'  /etc/passwd
```

A opção **-F** define o caractere dois pontos como separador dos campos. O comando **print** define a exibição de três argumentos: o primeiro campo de cada linha ($1), o string " => " e o sétimo campo de cada linha ($7). Portanto, a saída do comando acima para a entrada do usuário **aluno** é

aluno => /bin/bash

Também podemos executar o comando abaixo para obter o mesmo resultado.

```
awk -F: -f lista_usu /etc/passwd
```

onde **lista_usu** é um arquivo texto com o seguinte conteúdo

```
{print $1 " => " $7}
```

Observações

A **awk** é uma linguagem muito rica pois permite, por exemplo, o uso de

- comandos de decisão (if else),
- laços (for, while, do while),
- expresões aritméticas (+, -, *, /, %, ++, --, =, +=, -=, >, >=, <, <=, !=, etc),
- funções (exp, log, sqrt, split, substr, getline, etc),
- variáveis, constantes e vetores.

Para obter maiores informações, consulte o manual on-line (digite **man awk**).

bg

bg [num]

onde **num** corresponde ao número do processo a ser colocado em segundo plano (**background**). Caso o número do processo não seja fornecido, o sistema assume que o processo a ser colocado em **background** é o processo atual.

Use o comando **jobs -l** para obter a lista dos processos e seus respectivos números.

Observações

A grande vantagem em se colocar um processo em **background** é deixar o terminal livre para a entrada de outros comandos, ou seja, o processo está sendo executado, mas ele não está alocado a nenhum terminal específico.

Podemos também inicializar um programa/comando em segundo plano usando o caractere **&**. Por exemplo,

xcalc &

faz com que a calculadora seja inicializada em segundo plano e portanto, o terminal em que o comando foi digitado continua livre para receber novos comandos.

cal

cal [mês ano]

onde mês tem valor de 1 a 12 e ano tem valor maior que ou igual a 1.

Comentários sobre as opções do comando

Por exemplo,

cal 7 2001

exibe o calendário correspondente a julho de 2001. Se você digitar apenas

cal 7

são exibidos os meses de 1 a 12 do ano 7 da era cristã.

É importante notar que, se nenhum parâmetro for especificado junto com o comando, o calendário do mês atual é exibido.

cat

cat [opções] arquivos

Comentários sobre o comando

Podemos usar o comando **cat** para exibir os arquivos na tela. Por exemplo,

cat teste1 teste2

mostra na tela o conteúdo dos arquivos **teste1** e **teste2**.

Para parar a rolagem da tela e permitir a navegação com o resultado do comando **cat**, use os comandos **more** ou **less** junto com o **pipe** ("|").

cat teste1 teste2 | more

ou

cat teste1 teste2 | less

Para concatenar vários arquivos e colocar o resultado em um outro arquivo deve-se usar o **redirecionador de saída >**. Por exemplo,

cat arq1 arq2 arq3 > arq_final

gera o arquivo **arq_final** que corresponde a concatenação dos arquivos **arq1**, **arq2** e **arq3**. Se já existe um arquivo chamado **arq_final**, este arquivo é destruído e criado novamente.

Para inserir o arquivo **arq4** no final do arquivo **arq_final** digite

cat arq4 >> arq_final

Também é possível criar um arquivo usando o comando **cat** junto com o redirecionador de saída **>**. Para ter um exemplo, basta digitar as três linhas abaixo (tecle **ENTER** ao final das duas primeiras linhas e **CTRL+D** ao final da terceira linha).

cat > teste
aqui você digita o arquivo e
depois tecla CTRL+D.

Para numerar as linhas do arquivo basta usar o parâmetro **n**. Por exemplo,

cat -n teste1.txt

exibe o número e o conteúdo de cada linha do arquivo **teste1.txt**.

cd

cd [diretório]

Comentários sobre o comando

Este comando permite ao usuário mudar o diretório de trabalho. A mudança de diretório pode ser feita de forma sequencial (de diretório pai para diretório filho ou vice-versa) ou pode ser feita de forma aleatória (de um diretório qualquer para outro diretório qualquer).

São exemplos de uso deste comando:

- para ir para o seu diretório **home**, digite

  cd

  ou

  cd ~/

- para ir para o diretório *raiz* do Linux, digite

  cd /

- para ir para um diretório filho do diretório atual, digite

  cd diretório

- para ir para o diretório pai do diretório atual, digite

  cd ..

- para ir para um diretório do mesmo nível (diretório irmão) do diretório atual, digite

  cd ../diretório

- para voltar ao último diretório visitado antes do diretório atual, digite

  cd -

- você pode também fornecer o caminho completo do diretório para onde quer ir. Por exemplo,

  cd /usr/lib

  Se a variável de ambiente **CDPATH** foi definida (veja **shell** para mais detalhes), o sistema faz a busca de acordo com o conteúdo desta variável. Por exemplo,

  CDPATH=.:~:/etc

  faz com que o sistema procure pelo diretório especificado na seguinte ordem:

  1. a partir do diretório atual (representado na definição pelo ponto);
  2. a partir do diretório raiz do usuário (representado na definição pelo símbolo **~**);
  3. a partir do diretório **/etc**.

  Note que as três opções definidas na variável estão separadas por dois pontos (":").

chfn

chfn [opções] [usuário]

São algumas das opções deste comando

-f nome : altera o nome do usuário; se o nome for composto por mais de uma palavra, escreva o nome entre aspas.

-p num : altera o número do telefone do trabalho.

-h num : altera o número do telefone residencial.

-o nome : altera o local de trabalho; escreva o nome entre aspas se nome for composto por mais de uma palavra.

Comentários sobre as opções do comando

Ao digitar o comando **chfn**, sem opções, o sistema pergunta por modificação em cada um dos campos. Caso não queira alterar um determinado campo, basta digitar **Enter.**

Observações

Apenas o **root** pode alterar informações de outros usuários. Portanto, não é necessário fornecer o nome do usuário se você deseja apenas alterar as informações da sua própria conta.

Os dados informados com este comando são armazenados no arquivo **/etc/passwd** na entrada correspondente ao usuário. Para ver as informações existentes, por exemplo, do usuário **aluno**, digite

finger aluno

chgrp

chgrp [opções] grupo arquivo...

São algumas das opções deste comando

-R : altera, recursivamente, o grupo de um diretório e de todos os arquivos e diretórios que estão abaixo do diretório em questão.

-c : informa quais arquivos/diretórios estão tendo o nome do grupo alterado.

Comentários sobre as opções do comando

Suponha, por exemplo, a existência de um diretório de nome **teste**. Queremos que este diretório e todo o seu conteúdo passe a pertencer ao grupo **desenvolvimento**. Podemos, então, digitar o comando

chgrp -Rc desenvolvimento teste

para alterar o grupo do diretório **teste** e de todos os arquivos e diretórios que estão hierarquicamente abaixo do diretório **teste**. Como a opção **-c** é usada, será mostrada a lista dos arquivos e diretórios que tiveram o nome do grupo alterado.

chmod

chmod [ugoa][+-=][rwxugost] arquivo...

Cada arquivo/diretório do sistema está alocado a um usuário (dono) e a um grupo. Isto significa que um arquivo está associado a um **UID** e a um **GID**. O **UID** e o **GID** são inicialmente herdados do usuário que cria o arquivo. Entretanto, é possível modificar o dono e o grupo de um arquivo que já existe.

O dono de um arquivo (ou de um diretório) pode definir quem tem acesso ao arquivo e qual tipo de acesso é permitido (leitura, gravação e/ou execução). Isto é chamado de **permissão de acesso**. O **root** é o único usuário do sistema que tem acesso a todos os arquivos e diretórios de todos os usuários.

São opções deste comando

A combinação das letras **ugoa** no comando **chmod** define quais os usuários estão tendo as suas permissões de acesso alteradas:

- **u** = o dono do arquivo;
- **g** = os usuários que são membros do mesmo grupo do arquivo;
- **o** = os usuários que não membros do grupo do arquivo;
- **a** = qualquer usuário do sistema.

Caso não seja especificada a classe dos usuários para os quais se está alterando as permissões, o sistema usa a opção **a** (todos os usuários).

Deve-se usar, no comando **chmod**, um operador para especificar o tipo de modificação que se está fazendo nas permissões:

- **o operador +** provoca a adição das permissões informadas às permissões já existentes;
- **o operador -** provoca a remoção de permissões especificadas;
- **o operador =** provoca a redefinição das permissões (semelhante a zerar as permissões e defini-las novamente).

A combinação das letras **rwxst** no comando **chmod** especifica as permissões de acesso:

- **r** = leitura.
- **w** = gravação.
- **x** = execução (para arquivos) ou autorização de acesso (para diretórios).
- **u** = usar as mesmas permissões já atribuídas ao dono do arquivo.
- **g** = usar as mesmas permissões já atribuídas ao grupo.
- **o** = usar as mesmas permissões já atribuídas a outros.
- **s** = permissão especial de execução de um arquivo ou de acesso a um diretório.
    - Caso esta permissão seja dada ao dono do arquivo (diretório), a pessoa (com autorização) que executar o arquivo (acessar o diretório), o fará com as permissões de dono do arquivo (diretório). Este tipo de permissão é conhecido como **SUID**. Por exemplo, se o **root** possui um programa **SUID**, esse programa executará com privilégios de **root**, mesmo que tenha sido inicializado por um usuário comum.
    - Caso esta permissão seja dada ao grupo do arquivo (diretório), a pessoa (com autorização) que executar o arquivo (acessar o diretório), o fará com se fosse membro do grupo a qual pertence o arquivo (diretório). Este tipo de permissão é conhecido como **SGID**.
- **t** = permissão especial de execução de um arquivo ou de acesso a um diretório para o resto dos usuários do sistema (não é o dono e não pertence ao mesmo grupo do arquivo/diretório). Este tipo de permissão é conhecida como **sticky bit**.
    - Caso esta permissão seja dada a um diretório, o usuário pode criar,alterar e apagar apenas os seus próprios arquivos que estão neste diretório. Por exemplo, o diretório**/usr/temp**, usado para armazenar arquivos temporários dos usuários do sistema, possui esta permissão.

- Caso esta permissão seja dada a um arquivo, o arquivo pode ser compartilhado entre os vários usuários do sistema. Veja **/usr/src/linux/Documentation/mandatory.txt** para mais detalhes.

Comentários sobre as opções do comando

Por exemplo, o comando

chmod ug+rw teste.txt

define que o arquivo **teste.txt** pode ser lido (**r**) e alterado (**w**) pelo dono (**u**) e pelos usuários que são membros do mesmo grupo (**g**) do arquivo **teste.txt**.

Modo Octal

É possível também utilizar o **modo octal** para alterar as permissões de acesso a um arquivo. Abaixo mostramos a lista das permissões de acesso no modo octal.

Valor Octal	Valor Binário rwx	Significado
0	000	nenhuma permissão de acesso
1	001	permissão de execução (x)
2	010	permissão de gravação (w)
3	011	permissão de gravação e execução(wx)
4	100	permissão de leitura (r)
5	101	permissão de leitura e execução (rx)
6	110	permissão de leitura e gravação (rw)
7	111	permissão de leitura, gravação e execução (rwx)

Um exemplo do comando **chmod** usando a tabela acima é

chmod 764 teste.txt

Neste exemplo temos permissão de leitura, gravação e execução (7) para o dono do arquivo **teste.txt**, temos permissão para leitura e gravação (6) para os membros do grupo do arquivo e permissão de apenas leitura (4) para os outros usuários do sistema.

Os números octais são interpretados da direita para a esquerda, portanto o comando

chmod 64 teste.txt

define permissão de leitura (4) para os outros usuários do sistema e permissão de leitura e gravação (6) para os usuários do mesmo grupo do arquivo **teste.txt**.

Podemos também usar o modo octal para definir as permissões especiais.

Valor Octal	Valor Binário ugo	Significado
0	000	nenhuma permissão especial
1	001	sticky bit
2	010	SGID
3	011	SGID e sticky bit

4	100	SUID
5	101	SUID e sticky bit
6	110	SUID e SGID
7	111	SUID, SGID e sticky bit

No comando **chmod**, as permissões especiais, no modo octal, são definidas antes das permissões do dono, do grupo e do resto dos usuários. Por exemplo,

chmod 4760 teste.txt

define que o arquivo **teste.txt** é um arquivo **SUID** (4) com permissões 7, 6 e 0 para o dono, para os membros do grupo e para o resto dos usuários do sistema, respectivamente.

chown

chown dono[.grupo] arquivo...

São algumas das opções deste comando

-R : altera, recursivamente, o grupo de um diretório e de todos os arquivos e diretórios que estão abaixo do diretório em questão.

-c : informa quais arquivos/diretórios estão tendo o nome do grupo alterado.

Comentários sobre as opções do comando

Suponha, por exemplo, a existência de um diretório de nome **teste**. Queremos que este diretório e todo o seu conteúdo passe a pertencer ao usuário **aluno** e ao grupo **informatica**. Podemos, então, digitar o comando

chown -Rc aluno.informatica teste

para alterar o dono e o grupo do diretório **teste** e de todos os arquivos e diretórios que estão hierarquicamente abaixo do diretório **teste**. Como o argumento **-c** é usado, será mostrada a lista dos arquivos e diretórios que foram alterados.

É importante observar que este comando pode ser usado para alterar apenas o dono do arquivo, ou seja, você não é obrigado a especificar o nome do grupo no comando.

chsh

chsh [-s shell]

onde **shell** é o caminho completo do novo **shell**. Caso um **shell** não seja informado na linha de comando, o sistema solicitará ao usuário o nome do **shell**.

Comentários sobre as opções do comando

Por exemplo,

chsh -s /bin/ash

faz com que o **shell ash** passe a ser o **shell** padrão do usuário que digitou o comando.

Observações

A informação sobre qual **shell** é usado por cada usuário é armazenada no arquivo **/etc/passwd**. A linha abaixo mostra um exemplo de uma linha do **/etc/passwd** que associa o usuário **aluno** ao**shell bash**.

aluno:x:501:501::/home/aluno:/bin/bash

Portanto, se o comando **chsh** é executado com sucesso,o arquivo **/etc/passwd** é alterado pelo sistema. Apenas o **root** pode editar o arquivo **/etc/passwd** e modificá-lo diretamente (sem usar o comando **chsh**).

Para verificar quais os **shells** que estão disponíveis no sistema, digite

chsh -l

O comando acima, na realidade, apenas lista o conteúdo do arquivo **/etc/shells**.

comandos do linux

Comentários sobre os comandos

A maioria dos comandos do Linux possui o seguinte formato:

comando opções argumentos

onde **opções** usualmente começam com o caractere "-" seguido de uma ou mais letras. Por exemplo, para listar os arquivos e os diretórios filhos do diretório atual, basta digitar

ls

Já o comando

ls -al

exibe informações (tamanho, dono, grupo, etc) sobre os arquivos e os diretórios (**opção l**) e inclui na listagem os arquivos escondidos (**opção a**). **Importante:** os arquivos escondidos no Linux possuem nome que começam com um ponto (".").

As opções podem possuir um ou mais argumentos. Além disso, é possível existir argumentos que não pertençam a uma opção. Observe, por exemplo, o comando abaixo.

awk -F: -farq.txt /etc/passwd

Neste caso, temos o comando **awk** com duas opções (**F** e **f**) e três argumentos (**:**, **arq.txt** e **/etc/passwd**), sendo que o último argumento não pertence a nenhuma opção.

Para obter informações **on-line** sobre algum comando do Linux, digite **man** seguido do nome do comando. Por exemplo, para obter uma explicação sobre o comando **ls** digite

man ls

É também possível obter a lista dos parâmetros disponíveis usando o parâmetro/comando **help**.

Entrada e saída padrão dos comandos do Linux

- **stdin** (standard input) - entrada padrão para os comandos do Linux. Normalmente, a entrada padrão é o teclado, mas é possível usar o **redirecionador de saída <** para alterar a entrada padrão. Por exemplo, o comando

  tr a-z A-Z < teste

  usa o arquivo **teste** como entrada para o comando **tr**.

- **stdout** (standard output) - saída padrão dos comandos do Linux. Normalmente, a saída padrão é a tela da sessão onde o comando foi digitado, mas é possível usar o **redirecionador de saída > ou 1>** para alterar a saída padrão. Por exemplo,

  ls > teste

  ou

  ls 1> teste

  cria o arquivo **teste** com o conteúdo do diretório atual.

- **stderr** (standard error) - saída padrão para os erros encontrados durante a execução de um comando do Linux. Normalmente, a saída padrão é a tela da sessão onde o comando foi digitado, mas é possível usar o **redirecionador de saída 2>** para alterar a saída padrão. Por exemplo, suponha que o diretório **/meu_dir** não exista. Portanto, o comando

  ls /meu_dir 2> teste

  cria o arquivo **teste** com a mensagem de erro relacionada a execução do comando acima.

Observações

Neste **Guia de Referências** foi adotada a seguinte norma na apresentação dos comandos:

1. opções/argumentos **não obrigatórios** (os usuários podem ou não digitá-los) estão entre colchetes.
2. argumentos **obrigatórios** (devem ser fornecidos pelos usuários) não são delimitados por nenhum símbolo.

Por exemplo, o comando

cat [-n] arquivo

possui uma opção (**-n**) e um argumento (**arquivo**), sendo que apenas o segundo deve ser obrigatoriamente informado pelo usuário. Por exemplo, suponha que você queira exibir o conteúdo do arquivo **teste.txt**, então você pode digitar

cat -n teste.txt

ou
cat teste.txt

onde o primeiro comando numera as linhas do arquivo e o segundo comando apenas exibe o arquivo.

compress

Comentários sobre o comando

Para compactar um arquivo, digite

compress arquivo

O arquivo original é substituiído por um arquivo compactado com a extensão **.Z**.

Para descompactar, digite

uncompress arquivo

consoles virtuais

Modo texto e modo gráfico

A partir de uma mesma máquina podemos acessar o Linux usando diferentes usuários, ao mesmo tempo e de forma independente. Para isto, basta logar cada usuário em uma determinada console virtual. O Linux suporta 6 consoles no modo texto e 6 consoles no modo gráfico. As consoles do modo texto são numeradas de 1 a 6, enquanto as consoles do modo gráfico são numeradas de 7 a 12.

Estando no modo texto, para mudar de console, basta pressionar a tecla **ALT** junto com uma tecla de função. Por exemplo, **ALT+F2** mostra a console virtual 2. No modo gráfico é preciso também pressinar a tecla **CTRL** junto com as duas outras teclas. Assim, podemos logar um usuário diferente (ou um mesmo usuário) em mais de uma console.

A **console virtual 1** é, por padrão, utilizada para mostrar as mensagens de inicialização do sistema. Caso o modo de inicialização do sistema seja texto, o prompt de identificação dos usuários do sistema é apresentado em seguida na console 2. Caso o sistema seja inicializado no modo gráfico, a tela de identificação é apresentada na console 7.

É interessante observar que a console 1 mantém a exibição das últimas mensagens geradas pelo sistema durante o processo de inicialização. Entretanto, a console 1 está livre para ser utilizada, sendo necessário apenas que o usuário tecle **ENTER.**

Inicialização de várias interfaces gráficas

O sistema suporta a existência de várias interfaces gráficas rodando ao mesmo tempo, onde cada interface é associada a uma das consoles virtuais (de 7 a 12). É possível inicializar mais de uma interface gráfica a partir de uma mesma console virtual. Por exemplo, podemos digitar, a partir de uma console em modo texto,

kde -- :3 &

e em seguida

kde -- :1 &

Neste caso, duas interfaces gráficas, 3 e 1, são inicializadas pelo usuário. Se estas são as duas primeiras interfaces gráficas do sistema, então elas podem ser acessadas através das teclas**ALT+F7** e **ALT+F8**, respectivamente.

Observações

O modo de inicialização do sistema (texto ou gráfico, uma ou várias consoles) é definido no arquivo **/etc/inittab**.

cotas de disco

Definição

O administrador do sistema pode definir a quantidade de espaço em disco que cada usuário do sistema pode utilizar. Para isto, o Linux disponibiliza o sistema de cotas de disco. Este aplicativo permite definir uma cota padrão para todos os usuários do sistema e também cotas específicas para determinados usuários.

Instalação e customização

A maneira mais fácil de instalar e customizar este aplicativo é através do **linuxconf**. Os seguintes passos devem ser seguidos para usar o sistema de cotas:

1. deve-se habilitar o sistema de cotas no Linux

   linuxconf -> sistemas de arquivos -> acessar dispositivos locais -> editar partição -> ativar quota por usuário e/ou ativar quota por grupo

   Neste passo, o arquivo **/etc/fstab** é alterado para que a partição do Linux aceite e monte o sistema de cotas (veja abaixo um exemplo de uma partição Linux, definida no **fstab**, que suporta o aplicativo de cotas).

   /dev/hda2 / ext2 exec,dev,suid,rw,usrquota 1 1

2. deve-se definir uma cota padrão para os usuários e/ou grupos

   linuxconf -> sistemas de arquivos -> define quota padrão

3. caso deseje-se especificar uma cota diferente para algum usuário, deve-se editar as propridades da conta do usuário e alterar as informações desejadas

   linuxconf -> contas de usuários -> editar a conta a ser alterada

Palavras-chave

As seguintes palavras são usadas na definição do sistema de cotas:

- **Limite soft** - quantidade de espaço em disco que o usuário tem direito. Esta quantidade pode ser excedida por um determinado tempo (**período extra**), não podendo exceder a quantidade definida no **limite hard**.
- **Limite hard** - quantidade máxima de espaço em disco que o usuário pode usar. Este limite não pode ser excedido.
- **Período extra** - período de tempo no qual o usuário pode usar uma quantidade de espaço em disco superior ao **limite soft**, mas inferior ou igual ao **limite hard**. O padrão é sete dias.

Comandos relacionados

- **edquota** - editor de cotas dos usuários.
- **quota** - fornece informações sobre o sistema de cotas.
- **quotaoff** - desabilita o sistema de cotas.
- **quotaon** - habilita o sistema de cotas.

Observações

As definições do sistema de cotas são armazenadas no arquivo **/etc/quota.conf**.

cp

cp [opções] origem destino

onde

- **origem** corresponde ao arquivo a ser copiado;
- **destino** corresponde ao arquivo a ser criado pelo comando **cp**.

São algumas das opções deste comando

-b : gera cópia de segurança se o arquivo de destino já existir.

-f : substitui arquivos existentes sem pedir confirmação.

-i : pede permissão antes de substituir arquivos existentes.

-r : copia arquivos e subdiretórios (recursivo).

-v : lista os arquivos copiados.

Comentários sobre as opções do comando

Por exemplo, para copiar o arquivo **teste.txt** para **teste_bak.txt**, basta digitar

cp teste.txt teste_bak.txt

É possível especificar mais de um arquivo no comando **cp** usando os **curingas** *****, **?** e **[ ]**. O primeiro substitui um grupo qualquer de caracteres (qualquer número de carateres, inclusive zero, e para qualquer valor de caractere), o segundo substitui apenas um caractere (qualquer caractere), e o terceiro substitui um único caractere dentro de uma faixa de valores.

Exemplos:

cp teste*.txt /tmp/.

cp teste?.txt /tmp/.

cp teste[1-3].txt /tmp/.

O primeiro comando acima, copia todos os arquivos do diretório atual que começam por **teste** e têm extensão **txt** para o diretório **/tmp**. O segundo comando copia todos os arquivos que começam por **teste**, têm um caracter qualquer na sexta posição e extensão **txt** para o diretório **/tmp**. O último comando copia os arquivos **teste1.txt**, **teste2.txt** e **teste3.txt** (se existirem) para o diretório **/tmp**.

crontab

crontab [opções]

São algumas das opções deste comando

-e : edita o arquivo de **crontab**.

-l : lista o arquivo **crontab** na saída padrão.

-r : remove o arquivo **crontab** do usuário.

-u usuário : especifica o nome do usuário cujo arquivo será manipulado.

Comentários sobre as opções do comando

Por exemplo, o comando

crontab -u aluno -e

edita o arquivo **crontab** do usuário **aluno** (é preciso ter autorização para acessar o arquivo **crontab** de outro usuário).

Qualquer saída gerada pela execução dos arquivos **crontab** será enviada, via e-mail, para o dono do arquivo **crontab**.

Formato dos comandos

Cada linha do arquivo **crontab** possui: cinco campos de hora e data (veja a tabela abaixo); o nome de um usuário se o arquivo **crontab** for o do sistema; e o comando a ser executado.

Campo	Valores permitidos
minuto	0-59
hora	0-23
dia do mês	0-31
mês	0-12
dia da semana	0-7

Para os campos de data e hora pode-se usar um intervalo de números (**1-4**, ou seja, os números 1, 2, 3 e 4) ou uma lista de números (**1,4,5-7**, ou seja, os números 1, 4, 5, 6 e 7). Além disso, também é possível definir o valor de intervalo entre os números com o símbolo **/** (**0-23/2** corresponde à 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20 e 22).

Um asterisco nos campos de data e hora é usado para representar o intervalo completo (do primeiro ao último valor possível). Caso o asterisco seja seguido pelo símbolo **/** e por um número, então este número representa o intervalo entre os números do intervalo completo (***/2** no campo hora significa a cada duas horas).

O valor 0 e 7 no campo dia da semana corresponde ao dia de domingo. O valor 1 corresponde a segunda-feira. O valor 2 corresponde a terça-feira, e assim por diante.

O **cron** é o servidor que executa os comandos agendados. O **cron** ler os arquivos **crontab** armazenados no diretório **/var/spool/cron** a cada minuto para verificar as tarefas agendadas nestes arquivos.

Os arquivos **crontab** dos usuários são nomeados com o mesmo nome de login do usuário. Por exemplo, o arquivo **/var/spool/cron/aluno** é o arquivo **crontab** do usuário **aluno**. Abaixo temos um arquivo **crontab** de um usuário onde está agendado o comando **ls** para ser executado todos os domingos (0) no horário das 12:15.

```
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (- installed on Sun Jul 8 12:12:05 2001)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (- installed on Sun Jul 8 11:58:42 2001)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
15 12 * * 0 ls
```

Observações

As tarefas agendadas do sistema são armazenadas em **/etc/crontab** e executadas pelo daemon **crond** que está armazenado em **/etc/rc.d/init.d**.

curingas

Tipos de Curingas

Curingas são recursos usados para especificar um conjunto de arquivos e/ou diretórios de uma única vez. Permitindo assim, manipular vários arquivos/diretórios com um único comando.

Existem três tipos de curingas no Linux:

a. ***** que substitui um conjunto (zero ou mais de zero) de caracteres;
b. **?** que substitui um único caractere;
c. **[ ]**que substitui um conjunto de caracteres especificados dentro dos colchetes.

Exemplos

Por exemplo, suponha um diretório com os seguintes arquivos: **teste1.txt**, **teste2.txt**, **teste3.txt**, **teste4.txt**, **teste5.txt** e **teste10.txt**. O comando

```
rm teste*.txt
```

ou

```
rm teste*
```

remove todos os arquivos do diretório, enquanto o comando

```
rm teste?.txt
```

remove apenas os cinco primeiros arquivos (**teste1.txt** a **test5.txt**) e o comando

```
rm teste[2-4].txt
```

remove os arquivos **teste2.txt**, **teste3.txt** e **teste4.txt**. Se usarmos a vírgula no lugar do traço no exemplo acima, isto é,

```
rm teste[2,4].txt
```

apenas os arquivos **teste2.txt** e **teste4.txt** são apagados. O caractere **^** no início da expressão significa exclusão, ou seja, qualquer outro caractere menos os caracteres especificados. Portanto, o comando

```
rm teste[^2,4].txt
```

apaga os arquivos **teste1.txt**, **teste3.txt** e **teste5.txt**. Note que este comando não apaga o arquivo **teste10.txt** pois o curinga substitui apenas um caractere. Para substituir mais de um caractere podemos usar um ou mais curingas em um mesmo comando. Por exemplo,

```
rm teste[^2,4]*.txt
```

remove também o arquivo **teste10.txt**, além dos arquivos **teste1.txt**, **teste3.txt** e **teste5.txt**.

cut

cut -cX-Y arquivo

O comando acima seleciona da **X**-ésima a **Y**-ésima coluna do arquivo e exibe o resultado na saída padrão (normalmente a tela).

Comentários sobre o comando

Por exemplo, o comando

cut -c10-30 teste.txt

exibe, na tela, da décima a trigésima coluna do arquivo **teste.txt**, enquanto o comando

cut -c10 teste.txt

exibe apenas a décima coluna do arquivo **teste.txt**.

CVS

CVS significa **Concurrent Versions System** e é construído sobre a estrutura do **RCS** (**Revision Control System**). Entretanto, ao contrário do **RCS**, permite o controle de arquivos residentes em diretórios e máquinas distintas.

date

date [opções]

São algumas das opções deste comando

%A : dia da semana (domingo, ..., sábado).

%B : nome do mês (janeiro, ..., dezembro).

%H : hora do dia (0 a 23).

%M : minuto (0 a 59).

%S : segundos (0 a 61).

%T : hora no formato hh:mm:ss.

%Y : ano.

%a : dia da semana abreviado (dom, ..., sab).

%b : nome do mês abreviado (jan, ..., dez).

%c : dia da semana, data e hora.

%d : dia do mês.

%j : dia ano (1 a 366).

%m : mês (1 a 12).

%s : número de segundos desde das zero horas de 01/01/1970.

%w : dia da semana, onde 0 = domingo, 1 = segunda, ..., 6 = sábado.

%x : representação da data local.

%y : os dois últimos dígitos do ano.

MMDDhhmm[[CC]YY] : altera o mês (MM), o dia (DD), a hora (hh), os minutos (mm), os dois primeiros dígitos do ano (CC) e os dois útimos dígitos do ano (YY), onde os dois últimos argumentos (CC e YY) são opcionais.

Comentários sobre as opções do comando

Por exemplo, para imprimir o dia da semana, basta digitar

date '+%A'

É também possível usar o comando **date** para calcular datas. Para isto, deve-se usar a opção **-d** ou **--date**. Suponha que queremos saber a data de 2 meses e 15 dias a partir da data de hoje, então podemos digitar

date -d '2 months 15 days'

E para saber a data de 2 meses e 15 dias atrás, basta digitar

date -d '2 months 15 days ago'

Também podemos obter informações sobre uma determinada data com o comando **date**. Por exemplo, para obter o dia da semana em que cai o natal de 2000, podemos digitar

date -d '25 dec 2000' '+%A'

Observações

Ao digitar o comando **date**, sem parâmetros, o sistema informa o dia da semana, a data, a hora e o fuso horário utilizado.

Para que as alterações feitas na data e na hora do sistema sejam permanentes, pode-se usar o comando **hwclock** da seguinte forma

hwclock -w

dd

dd [opções]

São algumas das opções deste comando

if=*arquivo* : define o arquivo de entrada.

of=*arquivo* : define o arquivo de saída.

ibs=*bytes* : define o número de bytes a serem lidos por vez (o padrão é 512).

obs=*bytes* : define o número de bytes a serem gravados por vez (o padrão é 512).

bs=*bytes* : define o número de bytes a serem lidos e gravados por vez; este parâmetro anula os parâmetros **ibs** e **obs**.

count=*bytes* : define o número de blocos a serem copiados; cada bloco tem o número de bytes definidos pelo parâmetro **ibs**.

conv=*tipo* : converte o arquivo de acordo com o tipo especificado. Por exemplo,

- **ascii** converte um arquivo EBCDIC para ASCII;
- **ebcdic** converte um arquivo ASCII para EBCDIC;
- **lcase** converte as letras maiúsculas do arquivo em letras minúsculas;
- **ucase** converte as letras minúsculas do arquivo em letras maiúsculas.

Comentários sobre as opções do comando

Por exemplo, o comando

dd if=teste of=teste2 conv=ucase

cria o arquivo **teste2** a partir do arquivo **teste**, substituindo as letras minúsculas por letras maiúsculas.

Podemos alterar o comando acima para

dd if=teste of=teste2 count=1 conv=ucase

Neste caso, apenas o primeiro bloco do arquivo **teste** é copiado para **teste2**. Como o tamanho do bloco não é especificado, o sistema assume blocos de 512 bytes. Portanto, apenas os 512 primeiros bytes de **teste** são copiados para **teste2**.

Disquete de instalação

Para criar o disquete de instalação do **Conectiva Linux**, coloque o CD 1 e digite

dd if=/mnt/cdrom/images/boot.img of=/dev/fd0 bs=1440k

Para executar o comando acima é preciso ter montado o disquete e o CDROM (ver **mount**) e ter **permissão** de gravação no arquivo **/dev/fd0**.

depmod

depmod -a

Comentários sobre o comando

Deve-se executar o comando **depmod** para produzir um novo arquivo contendo as dependências dos módulos do **kernel** a serem instalados. Após executá-lo, pode-se usar o comando**modprobe** para instalar os módulos e as suas dependências.

df

df [opções] partição

São algumas das opções deste comando

-a : inclui também na listagem os sistemas de arquivos com zero blocos.

-k : lista o tamanho dos blocos em kbytes.

-m : lista o tamanho dos blocos em Mbytes.

-t tipo : especifica o tipo dos sistemas de arquivos a serem listados.

-x tipo : especifica o tipo dos sistemas de arquivos que não deve ser listado.

Comentários sobre as opções do comando

Por exemplo, o comando

df /dev/hda1

exibe as informações sobre o espaço livre da partição **/dev/hda1**.

Observações

Caso não seja fornecido o nome da partição, o sistema mostra as informações de todas as partições.

diff

diff arquivo1 arquivo2

São algumas das opções deste comando

-a : trata os arquivos como texto e os compara linha a linha, mesmo que não sejam arquivos de texto.

-i : ignora as diferenças entre letras maiúsculas e letras minúsculas.

Comentários sobre as opções do comando

Por exemplo, o comando

diff teste1 teste2

compara os arquivos **teste1** e **teste2**.

DISPLAY

Definição

No sistema **X**, é possível definir onde uma aplicação gráfica deve aparecer (em que máquina). Para isto basta alterar o conteúdo da **variável de ambiente** **DISPLAY**.

Verificar o valor da variável DISPLAY

Um **display** no X consiste, basicamente, de um teclado, um mouse e uma tela de vídeo. O **display** padrão de uma máquina corresponde a

máquina.domínio:0.0

ou simplesmente a

:0.0

onde **máquina** é o nome do computador onde o servidor X está sendo executado e **domínio** é o local da rede onde fica o computador. A sequência numérica indica o número do **display** (é usualmente zero, mas pode variar já que é possível existir múltiplos **displays** conectados em um computador) e o número da tela de vídeo (um **display** pode possuir múltiplas telas).

Para ver qual o **display** que você está usando no momento, digite

echo $DISPLAY

Alterar o valor da variável DISPLAY

Ao alterar o valor da variável **DISPLAY**, estaremos portanto alterando o local onde o **display** de uma máquina será mostrado. Suponha que você está usando a máquina **ipanema** do domínio**praias**, mas está conectado remotamente a máquina **copacabana**. Para executar, por exemplo, o aplicativo **ghostview** em **copacabana** mas ver o **display** em **ipanema**, basta digitar

export DISPLAY="ipanema.praias:0.0"

e depois entrar com o comando

ghostview

É também possível substituir os dois comandos acima pelo comando abaixo.

ghostview -display ipanema.praias:0.0

Observações

O programa **xhost** pode ser usado para definir quais máquinas podem ter acesso ao servidor X.

dmesg

dmesg [opções]

São algumas das opções deste comando

-n num : mostra apenas mensagens de inicialização do nível **num**.

-c : apaga as mensagens após exibi-las (apenas o **root** pode usar este parâmetro).

Comentários sobre as opções do comando

Por exemplo, o comando

dmesg -n 1

exibe as mensagens de nível 1 (*panic messages*) ocorridas durante a inicialização do sistema.

Observações

As mensagens apresentadas pelo comando **dmesg** são armazenadas em **/var/log/dmesg**.

du

du [opções] [arquivo ...]

São opções deste comando

-a : mostra o espaço ocupado por todos os arquivos (de forma recursiva).

-b : mostra o espaço ocupado em bytes.

-c : mostra o total do espaço ocupado.

-k : mostra o espaço ocupado em Kbytes (é o padrão).

-s : lista apenas o total para cada argumento.

Comentários sobre as opções do comando

Por exemplo, para ver o tamanho do arquivo **teste.txt** em bytes, digite

du -b teste.txt

Para obter o tamanho, em Kbytes, de todos os arquivos do diretório corrente com extensão **txt** e o total de espaço ocupado por estes arquivos, basta digitar

du -c *.txt

Observações

O uso do comando **du**, sem qualquer opção e sem nome de arquivo ou diretório, fornece a quantidade de espaço ocupada por cada subdiretório que se encontra hierarquicamente abaixo do diretório atual e a totalização dos espaços ocupados por esses subdiretórios.

dvips

dvips [options] arquivo

São algumas das opções deste comando

-c num : gera **num** cópias de cada página.

-C num : gera **num** cópias do arquivo.

-n num : gera no máximo **n** páginas.

-o arquivo : define o nome do arquivo de saída.

-p n : a primeira página do arquivo de saída será a página **n** do arquivo de entrada.

-pp lista : lista de páginas a serem geradas, podemos citar explicitamente as páginas (2,5,7,etc) ou especificar um conjunto de páginas (5-8).

Comentários sobre as opções do comando

Para criar o arquivo **teste.ps** a partir do arquivo **teste.dvi**, basta digitar

dvips -o teste.ps teste

Para definir que apenas 10 páginas são geradas a partir da vigésima página do arquivo **teste.dvi**, digite

dvips -o teste.ps -p 20 -n 10 teste

echo

echo diretório

Comentários o comando

Este comando exibe todos os nomes de arquivos do diretório em ordem alfabética, mas não formata a listagem em colunas. Por exemplo,

echo /*

lista o conteúdo do diretório **raiz**.

Este comando é também utilizado na programação **shell** para escrever mensagens na saída padrão. Por exemplo,

echo 'erro de execução'

mostra a mensagem definida entre aspas simples no terminal do usuário.

edquota

edquota [opções] nome

São algumas das opções deste comando

-u : edita a cota de um usuário. É o padrão.

-g : edita a cota de um grupo.

Comentários sobre as opções do comando

Por exemplo, o comando

edquota -u aluno

edita os valores no sistema de cotas para o usuário **aluno**. Abaixo, é mostrado um possível exemplo do arquivo editado.

Quotas for user aluno:
/dev/hda3: blocks in use: 9004, limits (soft = 35000, hard = 40000)
    inodes in use: 1249, limits (soft = 0, hard = 0)

Observações

O editor padrão do **edquota** é o **vim**.

Veja **cotas de disco** para mais detalhes.

emacs

Comentários sobre o emacs

Para editar ou criar um arquivo com este editor, basta digitar

emacs **arquivo**

Pode-se também usar as opções da ferramenta do X11 para definir a aparência do editor **emacs**. Por exemplo,

emacs -geometry 80x25+10+20 -fg white -bg black **arquivo**

define uma janela nas coordenadas x=10 e y=20 com 25 linhas sendo que cada linha suporta até 80 caracteres (este é o tamanho padrão).

Para acessar o **emacs** sem os recursos do ambiente gráfico (sem X11), podemos digitar

emacs-nox **arquivo**

É importante observar que o comando acima pode ser executado inclusive em um terminal X11, mas mesmo assim, as opções gráficas do ambiente não serão utilizadas.

Comandos do emacs

ALT+< : ir para o início do arquivo.

ALT+> : ir para o fim do arquivo.

ALT+D : remover a linha atual.

ALT+V : voltar uma tela.

CTRL+_ : desfazer a última operação.

CTRL+A : ir para o início da linha atual.

CTRL+B : faz o cursor retornar um caracter.

CTRL+D : apagar caractere atual.

CTRL+E : ir para o fim da linha atual.

CTRL+F : faz o cursor avançar um caracter.

CTRL+H : exibir ajuda.

CTRL+H,T : iniciar o tutorial.

CTRL+K : remover a linha atual.

CTRL+N : faz o cursor descer uma linha.

CTRL+P : faz o cursor subir uma linha.

CTRL+V : avançar uma tela.

CTRL+X+1 : excluir janela de ajuda.

CTRL+X+2 : incluir janela de ajuda.

CTRL+X,U : desfazer o último comando.

CTRL+X, CTRL+F : abrir um arquivo.

CTRL+X, CTRL+C : sair do emacs.

CTRL+X, CTRL+B : lista os arquivos abertos no emacs.

CTRL+X, CTRL+S : salvar o arquivo atual.

CTRL+X, CTRL+W : salvar o arquivo atual como.

ESC+B : faz o cursor voltar uma palavra.

ESC+F : faz o cursor avançar uma palavra.

env

env [opções] [comando]

São algumas das opções deste comando

-u variável : remove a variável de ambiente especificada durante a execução do comando.

-i : ignora todas as variáveis de ambiente.

Comentários sobre as opções do comando

Suponha a existência da seguinte definição no sistema

alias ls='ls --color=tty'

O uso do comando **alias**, neste exemplo, significa que, sempre que o usuário digitar

ls

e o sistema executará

ls --color=tty

Portanto, a listagem dos arquivos de um diretório sempre aparecerá colorida. Para fazer com que o sistema ignore as modificações feita pelo comando **alias** e qualquer variável de ambiente, basta digitar

env -i ls

Observações

O comando **env**, sem qualquer argumento, lista as **variáveis de ambiente**.

exit

Comentários sobre o comando

O usuário pode executar outro **shell**, além do **shell** padrão. Para isto, basta digitar na linha de comando o nome do novo **shell**. Para retornar ao **shell** anterior, deve-se digitar **exit**.

Caso o usuário não tenha chamado outro **shell**,

- no modo gráfico, o comando **exit** fecha a janela na qual o usuário digitou o comando;
- no modo texto, o comando **exit** encerra a sessão do usuário (volta para a tela inicial onde é feita a identificação dos usuários).

Observações

A informação sobre qual **shell** é usado por cada usuário é armazenada no arquivo **/etc/passwd**.

A lista dos **shells** disponíveis no sistema está no arquivo **/etc/shells**.

export

export variável=valor

ou

variável=valor; export

Comentários sobre o comando

Por exemplo, o comando

export TESTE=2

cria a **variável de ambiente** **TESTE** e atribui o valor 2 a esta variável.

Observações

É importante observar que a variável criada só existe para a sessão (terminal) onde o comando foi digitado.

Para apagar uma variável de ambiente, use o comando **unset**.

fdformat

fdformat [-n] dispositivo

onde

- **-n** : desabilita a verificaçã após a formatação.
- **dispositivo** : a identificação do disquete (por exemplo, /dev/fd0d360, /dev/fd0h1200, /dev/fd0D360, etc). Os dispositivos /dev/fd0 e /dev/fd1 só funcionarão quando o formato padrão estiver sendo usado ou quando o formato for auto detectado.

Comentários sobre as opções do comando

Por exemplo,

fdformat /dev/fd0

formata o disquete padrão. Use o comando **mkfs** em seguida para configurar o disquete para o sistema de arquivos **ext2**.

fdisk

fdisk [opções] [disco]

São algumas das opções deste comando

-l : lista as partições existentes no disco atual (caso o usuário não especifique o disco).

-u : fornece o tamanho das partições em número de setores ao invés de número de cilindros.

/mbr : apaga apenas o MBR.

Opções do comando fdisk

Para particionar o disco deve-se digitar apenas **fdisk** e informar as modificações a serem introduzidas. São algumas das opções disponíveis:

d : deleta uma partição.

l : lista os tipos de partições existentes.

m : lista as opções do fdisk.

n : adiciona uma nova partição.

p : lista as partições existentes.

q : sai do fdisk sem efetivar as alterações (caso tenha feito alguma).

t : muda o tipo da partição.

w : grava as alterações e encerra o fdisk (CUIDADO!).

fg

fg [num]

ou

fg [%ordem]

onde **num** corresponde ao número do processo e **ordem** corresponde a ordem de entrada do processo em **background**. Caso o comando seja digitado sem nenhum argumento, o sistema assume que o último processo a entrar em **background** (comando **bg** ou **&**), é o primeiro a sair do **background**.

Observações

Use o comando **jobs -l** para obter a lista dos processos e seus respectivos números.

file

file arquivo

Comentários sobre o comando

Suponha que o arquivo **teste** seja um arquivo de texto. O comando

file teste

então fornecerá a seguinte resposta

teste: International language text

find

find [caminho] [expressão]

onde:

- **caminho** fornece o nome do diretório por onde a pesquisa deve ser iniciada.
- **expressão** é feita de **opções** (que afetam toda a operação e sempre retorna verdadeiro), **testes** (que retorna um valor falso ou verdadeiro), e **ações** (que tem efeitos colaterais e retornam um valor falso ou verdadeiro), todos separados por operadores. O valor **-e** é assumido onde um operador é omitido. Se a expressão não contém ações diferentes de **-prune**, **-print** é executado para todos os arquivos para os quais a expressão é válida.

Opções

-daystart : medem o tempo (para **-amin**, **-atime**, **-cmin**, **-ctime**, **-mmin**, e **-mtime**) do início do dia ou de até 24 horas atrás.

-depth : processa o conteúdo de cada diretório antes do diretório em si.

-follow : resolve as ligações simbólicas. Tem ligação com a opção **-noleaf**.

-help, **--help** : imprime um resumo do uso das linhas de comando de **find** e finaliza.

-maxdepth n : desce no máximo em **n** níveis (um inteiro não negativo) de diretórios sob os argumentos da linha de comando. A opção **-maxdepth 0** significa a aplicação dos testes e ações somente nos argumentos da linha de comandos.

-mindepth n : não aplica qualquer teste ou ação a níveis menores que **n** níveis (um inteiro não negativo). A opção **-mindepth 1** significa testar todos os arquivos exceto os argumentos da linha de comandos.

-mount : não descer em diretórios de outros sistemas de arquivos. É um nome alternativo para a opção **-xdev**.

-noleaf : não otimizar ao assumir que um diretório contém 2 entradas a menos que as ligações simbólicas para ele definidas. Esta otimização não é aplicável quando a pesquisa de sistemas de arquivos se dá em sistemas que não seguem uma convenção Unix. Cada diretório em um sistema de arquivos tem no mínimo 2 ligações diretas: seu nome e sua entrada **`.'**. Adicionalmente, seus subdiretórios (caso existam) tem cada um uma ligação de entrada `..' para aquele diretório. Quando **find** está examinando um diretório, após considerar 2 entradas a menos do que os contadores de ligações apontem, ele sabe que o restante das entradas no diretório não são outros diretórios (e sim arquivos folha na árvore de diretórios). Se somente os nomes de arquivos necessitam ser examinados, não há necessidade de examinar os diretórios; obtendo-se assim um incremento na velocidade de pesquisa.

-version, **--version** : lista o número da versão do comando **find** e finaliza.

-xdev : não desce diretórios em outros sistema de arquivos.

Testes

Argumentos numéricos podem ser:

- **+n** : valores maiores que n.
- **-n** : valores menores que n.
- **n** : valor exatamente igual a n.

-amin n : arquivo foi acessado há **n** minutos.

-anewer arq : arquivo foi acessado mais recentemente do que **arq** tenha sido modificado. O teste **-anewer** é afetado pela opção **-follow** somente se **-follow** vir antes que **-anewer** na linha de comandos.

-atime num : procura por arquivos acessados há **num** dias.

-cmin n : o status do arquivo foi alterado em até **n** minutos atrás.

-cnewer arq : o status do arquivo foi alterado mais recentemente do que **arq** foi modificado. O teste **-cnewer** é afetado pela opção **-follow** somente se **-follow** vir antes que **-cnewer** na linha de comando.

-ctime n : o status do arquivo foi mudado nos últimos **n** dias.

-empty : o arquivo está vazio e é ou um arquivo regular ou um diretório.

-false : sempre falso.

-fstype tp : arquivo está em um sistema de arquivos do tipo **tp**. Pode-se usar a opção **-printf** com diretivas **%F** para ver os tipos dos sistemas de arquivos.

-gid n : o **GID** (Group Identification) do arquivo é **n**.

-group gname : arquivo pertence ao grupo **gname** (pode-se também usar aqui o **GID** do grupo).

-ilname pattern : semelhante ao teste **-lname**, mas neste caso o teste é sensível a maiúsculas e minúsculas.

-iname pattern : semelhante ao teste **-name**, mas neste caso o teste não é sensível a maiúsculas e minúsculas.

-inum n : procura por arquivo com **inode** **n**.

-ipath pattern : semelhante ao teste **-path**, mas neste caso o teste não é sensível a maiúsculas e minúsculas.

-iregex pattern : semelhante ao teste **-regex**, mas neste caso o teste não é sensível a maiúsculas e minúsculas.

-links n : arquivo tem **n** ligações.

-lname padrão : arquivo é uma ligação simbólica cujos conteúdos coincidem com o padrão de ambiente **shell** **padrão**. Os metacaracteres não tratam `/' ou `.' de forma especial.

-mmin n : os dados dos arquivos foram modificados há **n** minutos.

-mtime n : os dados foram modificados em até **n** dias atrás.

-name arquivo : procura pelo arquivo especificado. O nome do arquivo (o caminho à frente do nome do arquivo não é considerado) deve coincidir com os padrões informados. Os metacaracteres (`*', `?', e `[]') não combinam com um `.' no início no nome do arquivo. Para ignorar um diretório e os arquivos nele contidos, deve-se usar a ação **-prune**.

-newer arq : o arquivo foi modificado mais recentemente que **arq**. O teste **-newer** é afetado pela opção **-follow** somente se **-follow** vem antes de **-newer** na linha de comando.

-nouser : nenhum usuário corresponde ao ID numérico do usuário dono do arquivo.

-nogroup : nenhum grupo corresponde ao ID numérico do grupo dono do arquivo.

-path padrão : nome do arquivo combina com os padrões especificados em **padrão**. Os metacaracteres não tratam `/' ou `.' de forma especial. Use a ação **-prune** para ignorar o diretório especificado por **padrão** e todos os seus subdiretórios.

-perm valor : arquivos que possuem os bits de permissão exatamente iguais a **valor** (octal ou simbólico).

-perm -valor : arquivos que possuem todos os bits de permissão definidos em **valor** (octal ou simbólico).

-perm +valor : arquivos que possuem qualquer bit de permissão definido em **valor** (octal ou simbólico).

-regex padrão : arquivos que combinem com a expressão regular em **padrão**. Isto é uma combinação com todo o caminho, e não somente uma pesquisa.

-size n[bckw] : procura por arquivos que tem **n** unidades de espaço. Ao usar + (-) antes de **n** procura-se por um arquivo maior (menor) que **n**. As unidades são

- **b** - blocos de 512 bytes (padrão);
- **c** - bytes;
- **k** - kilobytes;
- **w** - palavras de 2 bytes.

-true : sempre verdadeiro.

-type tipo : procura arquivos de um determinado tipo. São tipos possíveis

- **b** - blocos especiais (buffer)
- **c** - caracteres especiais
- **d** - diretórios
- **p** - conector definido (FIFO)
- **f** - arquivo regular
- **l** - ligação simbólica
- **s** - socket

-uid n : o **UID** do dono do arquivo é igual a **n**.

-used n : arquivo foi acessado **n** dias após a última alteração de seu status.

-user nome : arquivo pertence ao usuário especificado em **nome** (pode-se também usar aqui o **UID** do dono).

-xtype tp : semelhante a opção **-type** a menos que o arquivo seja uma ligação simbólica. Neste caso, se a opção **-follow** não foi informada, será verdadeiro se **tp** for igual a **l**. Ou seja, para ligações simbólicas **-xtype** checa o tipo de arquivo, ação que **-type** não executa.

Ações

-exec comando : executar **comando**; verdadeiro se o status de saída zero for retornado. Todos os argumentos para **find** serão considerados como argumentos do comando até que um argumento consistido por `;' seja encontrado. Os caracteres `{}' são substituídos pelo nome do arquivo que está sendo processado aonde que ele ocorra nos argumentos do comando, e não somente em argumentos onde esteja sozinho, como em algumas versões do **find**. Ambas as construções podem necessitar de um caractere de saída (como um `\') ou citados para protegê-los da expansão pelo ambiente de trabalho. O comando é executado no diretório inicial.

-fls arq : verdadeiro; semelhante a ação **-ls** mas grava o resultado em **arq** de forma similar a ação **-fprint**.

-fprint arq : verdadeiro; lista o nome completo em um arquivo **arq**. Caso **arq** não exista quando **find** está sendo executado, ele será criado; caso ele exista, será recriado sem conteúdo. Os arquivos de

nome **/dev/stdout** e **/dev/stderr** são tratados de forma especial; eles são a referência à saída padrão e a saída de erros padrão, respectivamente.

-fprint0 arq : verdadeiro; similar a ação **-print0** mas grava em **arq** como a ação **-fprint**.

-fprintf arq formato : verdadeiro; similar a ação **-printf** mas grava em **arq** como a ação **-fprint**.

-ok command : similar a **-exec** porém apresenta uma pergunta ao usuário na entrada padrão, e se a resposta não começar com `y' ou `Y', não executa o comando e retorna falso.

-print : verdadeiro; imprime o nome completo do arquivo na saída padrão, seguido de nova linha.

-print0 : verdadeiro; lista o nome completo do arquivo na saída padrão, seguido de um caractere nulo. Isso permite que um arquivo cujo nome contenha o caractere de nova linha seja corretamente interpretado pelos programas que processem a saída do comando **find**.

-printf formato : verdadeiro; lista **formato** na saída padrão, interpretando `\' como caractere de fuga e `%' como diretivas. Largura de campos e precisão podem ser especificadas de forma similar à função **printf()** da linguagem C. Diferentemente de **-print**, **-printf** não adiciona um caractere de nova linha ao final da cadeia de caracteres. Um caractere `\' seguido por outro é tratado como um caractere comum, sendo que ambos serão impressos. Um caractere `%' seguido por qualquer outro caractere será descartado (mas o outro caractere será impresso). O caractere de fuga e as diretivas são:

- **\a** - campainha de alarme.
- **\b** - retorno.
- **\c** - pára a impressão deste formato imediatamente e descarrega a saída.
- **\f** - alimentação de formulário.
- **\n** - nova linha.
- **\r** - avanço e retorno de linha.
- **\t** - tabulação horizontal.
- **\v** - tabulação vertical.
- **** - um literal de barra reversa (`\').
- **%%** - um sinal de percentual.
- **%a** - data e hora do último acesso ao arquivo no formato retornado pela função C **ctime()**.
- **%Ak** - data e hora do último acesso em um formato especificado em **k**, o qual é, ou `@' ou uma diretiva para a função **strftime()** da linguagem C. Os valores possíveis em **k** estão listados abaixo; alguns deles podem não estar disponíveis em todos os sistemas, devido às diferenças da função **strftime** entre eles.
    - @ - segundos desde 1 de Janeiro de 1970, 00:00 GMT.
    - campos de tempo:
        - **H** - hora (00..23)
        - **I** - hora (01..12)
        - **k** - hora (0..23)
        - **l** - hora (1..12)
        - **M** - minutos (00..59)
        - **p** - AM ou PM
        - **r** - hora, 12-horas (hh:mm:ss [AP]M)
        - **S** - segundos (00..61)
        - **T** - hora, 24-horas (hh:mm:ss)
        - **X** - representação local de horário (H:M:S)
        - **Z** - zona de horário (por exemplo, EDT), ou nada se nenhuma zona de horário for determinada.
    - Campos da data:
        - **a** - abreviatura local dos dias da semana (Dom..Sab)
        - **A** - nome completo do dia da semana, de tamanho variável (Domingo..Sábado)
        - **b** - nomes dos meses abreviados (Jan..Dez)
        - **B** - nomes dos meses completos com tamanho variável (Janeiro..Dezembro)
        - **c** - data e horário locais (Sáb Nov 04 12:02:33 EST 1998)
        - **d** - dia do mês (01..31)
        - **D** - data (mm/dd/aa)
        - **h** - igual a **b**

        - **j** - dia do ano (001..366)
        - **m** - mês (01..12)
        - **U** - número da semana no ano com domingo como o primeiro dia da semana (00..53)
        - **w** - dia da semana (0..6)
        - **W** - número da semana no ano com segunda-feira como o primeiro dia da semana (00..53)
        - **x** - representação local da data (mm/dd/yy)
        - **y** - últimos dois dígito do ano (00..99)
        - **Y** - ano (1970...)
    - **%b** - tamanho do arquivo em blocos de 512 bytes arredondado para cima.
    - **%c** - data da última mudança de status do arquivo no formato retornado pela função **ctime()** da linguagem C.
    - **%Ck** - data da última mudança de status do arquivos no formato especificado por **k**, o qual é o mesmo que para **%A**.
    - **%d** - profundidade dos arquivos na árvore de diretórios; zero significa que o arquivo é um argumento da linha de comando.
    - **%f** - nome do arquivo com exclusão de qualquer nome de diretório (somente o último elemento).
    - **%F** - tipo do sistema de arquivos em que o arquivo se encontra; este valor pode ser usado por **-fstype**.
    - **%g** - nome do grupo de arquivos, ou a identificação numérica do grupo caso este não tenha nome.
    - **%G** - identificação numérica do grupo do arquivo.
    - **%h** - caminho ou diretório(s) onde o arquivo está localizado (tudo exceto o último elemento).
    - **%H** - argumento da linha de comando na qual o arquivo foi encontrado.
    - **%i** - número do **inode** do arquivo (em decimais).
    - **%k** - tamanho do arquivo em blocos de 1K (arredondado para cima).
    - **%l** - objeto da ligação simbólica (vazio se o arquivo não for uma ligação simbólica).
    - **%m** - bits das permissões do arquivo (em octal).
    - **%n** - número de ligações diretas para o arquivo.
    - **%p** - nome do arquivo.
    - **%P** - nome do arquivo com o nome dos argumentos da linha de comando sob a qual ele foi removido.
    - **%s** - tamanho do arquivo em bytes.
    - **%t** - data da última modificação do arquivo no formato retornado pela função **ctime()** da linguagem C.
    - **%Tk** - data da última modificação do arquivo no formato especificado em **k**, o qual é o mesmo para **%A**.
    - **%u** - nome do dono do arquivo ou a sua identificação numérica se o dono não tiver nome.
    - **%U** - identificação numérica do dono do arquivo.
- **-prune** : caso **-depth** não seja informado, será verdadeiro; não descendo a partir do diretório atual. Caso **-depth** seja informado, será falso e não terá efeito.
- **-ls** : verdadeiro; lista o arquivo atual no formato **ls -dils** na saída padrão. O contador de blocos utiliza unidades de 1 Kb, a menos que a variável de ambiente **POSIXLY_CORRECT** esteja configurada, neste caso a unidade será de 512 bytes.

Operadores

Listados em ordem de precedência decrescente:

- **(expr)** - força a precedência.
- **! expr** - verdadeiro se **expr** for falsa.
- **-not expr** - o mesmo que **! expr**.
- **expr1 expr2** - são implícitas; **expr2** não será avaliada se **expr1** for falsa.
- **expr1 -a expr2** - o mesmo que **expr1 expr2**.
- **expr1 -and expr2** - o mesmo que **expr1 expr2**.
- **expr1 -o expr2** - Ou; **expr2** não é avaliada se **expr1** for verdadeira.
- **expr1 -or expr2** - o mesmo que **expr1 -o expr2**.
- **expr1 , expr2** - lista; ambas **expr1** e **expr2** sempre são avaliadas. O valor de **expr1** é descartado; o valor da lista é o valor de **expr2**.

Exemplos

- O comando

  find -name teste.tex

  procura o arquivo **teste.tex** no diretório atual e em todos subdiretórios abaixo do diretório atual.

- O comando

  find ~/ -maxdepth 3 -name teste.tex

  procura o arquivo **teste.tex** a partir do diretório **home** (~/) descendo no máximo 3 níveis de diretórios.

- O comando

  find . -path './testes' -prune -o -print

  lista todos os arquivos encontrados a partir do diretório corrente exceto os arquivos que estão no diretório **testes** ou em algum dos seus subdiretórios.

- O comando

  find . -size +1000k

  lista os arquivos com mais de 1000k de tamanho a partir do diretório atual (que está representado pelo '.').

- Para localizar os arquivos SUID (possuem **permissões** de proprietário), digite

  find / -perm +4000

- Use a opção **atime** para localizar programas novos ou acessados com mais freqüência. Por exemplo, o comando

  find /etc -atime -1

  lista os arquivos do diretório **/etc** que foram acessados há um dia ou menos, enquanto o comando

  find ~/ -atime +10

  lista os arquivos, a partir do diretório **home** do usuário, que foram acessados pela última vez há 10 dias ou mais.

- O comando

  find / -type l -print |perl -nle '-e || print';

  procura por todos os links perdidos e os imprime.

- O comando

  find . -user aluno1 -exec chown aluno2 {}

  altera o dono (veja comando **chown**) de todos os arquivos e subdiretórios a partir do diretório corrente de **aluno1** para **aluno2**.

Observações

O uso do comando **find**, sem argumentos, faz com seja exibida a lista de todos os arquivos que ficam hierarquicamente abaixo do diretório corrente.

finger

finger [usuário ou **usuário@domain]**

Observações

Se não for fornecido o nome do usuário, o sistema mostra a lista de todos os usuários conectados no momento.

Os dados informados com este comando são armazenados no arquivo **/etc/passwd** na entrada correspondente ao usuário.

fortune

fortune [opções]

As mensagens armazenadas pelo aplicativo **fortune** são divididas em várias categorias, onde cada categoria se subdivide em mensagens potencialmente ofensivas e mensagens não ofensivas.

São algumas das opções deste comando

-a : seleciona todos os arquivos de citações (ofensivas ou não).

-f : exibe a lista dos arquivos que podem ser usados pelo **fortune**.

-l : seleciona apenas mensagens longas.

-o : seleciona apenas os arquivos com mensagens ofensivas.

-s : seleciona apenas mensagens curtas.

Comentários sobre as opções do comando

Por exemplo, pode-se usar o aplicativo **fortune** para apresentar frases aleatórias sempre que o usuário logar no sistema. Para isto, basta incluir no final do arquivo **/etc/profile** os seguintes comandos

```
if [ -x /usr/games/fortune ]; then
        echo
        /usr/games/fortune literature
        echo
fi
```

No exemplo acima, usa-se o arquivo **/usr/share/games/fortune/literature.dat** como fonte das frases.

É também possível definir vários arquivos para o **fortune**, bem como a porcentagem de uso de cada arquivo. O comando

```
/usr/games/fortune 10% literature 90% work
```

indica que 10% das mensagens mostradas serão do arquivo **/usr/share/games/fortune/literature.dat** e 90% serão do arquivo **/usr/share/games/fortune/work.dat**.

Observações

Quando **fortune** é usado sem parâmetros, uma frase é selecionada aleatoriamente de qualquer um dos arquivos armazenados em **/usr/share/games/fortune**.

Você pode criar o seu próprio arquivo de citações com o comando **strfile**.

free

free [opções]

O comando **free** mostra a quantidade de memória livre e utilizada, a área de **swap** no sistema, a memória compartilhada e os buffers utilizados pelo **kernel**.

São opções deste comando

-k : as informações são dadas em Kbytes (é o padrão).

-m : as informações são dadas em Mbytes.

-o : oculta a linha com as informações sobre os buffers utilizados pelo **kernel**.

-t : mostra uma linha contendo a quantidade total de memória do sistema, a quantidade de memória livre e a quantidade de memória em uso.

-s num : mostra a quantidade de memória livre usada a cada **num** segundos.

Comentários sobre as opções do comando

Por exemplo, o comando

free -s 5

mostra, a cada 5 segundos, a quantidade de memória livre e ocupada do sistema.

fsck

fsck [opções] sistema

onde **sistema** pode ser o nome do dispositivo ou o ponto de montagem para o sistema de arquivos. Se mais de um sistema de arquivo for fornecido, o **fsck** tentará verificá-los em paralelo.

São algumas das opções deste aplicativo

-a : repara automaticamente o sistema de arquivos com defeito.

-A : analisa o arquivo **/etc/fstab** e tenta verificar todos os arquivos listados de uma vez.

-N : não executa, apenas mostra o que seria feito.

-r : pergunta se o sistema de arquivos deve ser reparado quando detecta alguma falha.

-V : produz uma saída detalhada.

Observações

O **fsck** é usado para verificar e, opcionalmente, reparar um sistema de arquivos do Linux. O código de erro retornado é a soma das seguintes condições:

0 - nenhum erro.

1 - erros do sistema de arquivos corrigidos.

2 - o sistema deve ser reiniciado.

4 - erros do sistema de arquivos não corrigidos.

8 - erro operacional.

16 - erro de uso ou de sintaxe.

128 - erro de biblioteca compartilhada.

Na realidade, o **fsck** é simplesmente um intermediário para os vários verificadores de sistemas de arquivos disponíveis no Linux (por exemplo, **fsck.ext2** para sistemas de arquivo do tipo**ext2**). O executável específico é primeiro procurado em **/sbin**, depois em **/etc/fs** e finalmente nos diretórios listados na **variável de ambiente** **PATH**.

fuser

fuser [options] nome...

onde **nome** pode ser um arquivo (executável ou não) ou um diretório.

São algumas das opções deste comando

-i : pede confirmação antes de matar um processo (usado junto com a opção **-k**).

-k : mata os processos que estão acessando o arquivo/diretório especificado.

-u : identifica o usuário de cada processo.

Comentários sobre as opções do comando

Por exemplo, para exibir os processos (com a identificação dos respectivos usuários) que estão usando o **shell bash**, o editor de texto **vim** e o diretório **/home/aluno1**, digite

fuser -u /bin/bash /bin/vim /home/aluno1

Uma possível saída para este comando é:

/bin/bash: 1113e(root) 1125e(aluno1) 1154e(aluno2)
/bin/vim: 1350e(aluno1)
/home/aluno1: 1125c(aluno1) 1350c(aluno1)

onde temos três processos usando o arquivo **/bin/bash** (1113, 1125 e 1154), apenas um processo usando o arquivo **/bin/vim** (1350) e dois processos usando o diretório **/home/aluno1** (1125 e 1350). Ao lado do número de um processo (PID) temos uma letra e a identificação do usuário a quem pertence o processo. A letra que aparece logo após o PID representa o tipo de acesso, onde podemos ter, por exemplo,

c : diretório atual (a partir do qual o processo foi inicializado).

e : arquivo sendo executado pelo processo.

r : diretório raiz do sistema (ponto de inicialização do processo).

Note que, no exemplo acima, o processo 1350 aparece nas duas últimas linhas exibidas pelo comando **fuser**. Isto significa que o aplicativo **vim** foi chamado a partir do diretório**/home/aluno1**.

Observações

A lista completa dos processos em execução no sistema pode ser vista no diretório **/proc**.

gcc

gcc [opções] arquivo

São algumas das opções do compilador gcc

-c : compila sem linkeditar.

-D macro : define nome de macro a ser usada dentro do programa (podemos também definir uma macro dentro do programa C através da inclusão da linha "#define macro"). O uso de macros permite selecionar quais partes do código C devem ser compiladas. Por exemplo,

```
#ifdef teste_macro
conjunto de instruções 1
#else
conjunto de instruções 2
#endif
```

O compilador usará o conjunto de instruções 1 se a macro **teste_macro** tiver sido definida, senão usará o conjunto de instruções 2.

-fPIC : gera código compilado independente de posição (permitindo que o código seja compartilhado por vários programas).

-g : gera informações para depuração (veja **gdb**) do programa.

-I diretório : adiciona diretório a lista dos diretórios pesquisados na busca por arquivos definidos por um comando **include**.

-l XXX : define biblioteca a ser incluida durante a linkedição. Ao nome especificado pelo usuário, é acrescentado "lib" como prefixo e ".a" e ".so" como sufixo, ou seja, o nome procurado neste exemplo é libXXX.a (biblioteca estática) e libXXX.so (biblioteca compartilhada).

-L diretório : adiciona diretório a lista dos diretórios pesquisados na busca por arquivos definidos como bibliotecas compartilhadas.

-o arquivo : nome do arquivo de saída executável (o nome padrão é **a.out**).

-shared : produz um objeto compartilhado que pode então ser linkeditado com outros objetos para formar um executável.

-w : omite todas as mensagens de advertência da compilação.

-Wall : exibe todas as mensagens de advertência da compilação.

Comentários sobre as opções do comando

Por exemplo, o comando

```
gcc -o teste teste.c
```

compila o arquivo **teste.c** e cria, caso não haja erro de compilação, o arquivo **teste**.

Observações

É possível que os pacotes referentes a compilação de programas C não tenham sido instalados junto com o sistema. Na distribuição **Conectiva 5.0**, por exemplo, use o comando **rpm** para fazer a instalação dos seguintes pacotes do CD 1:

binutils-2.9.5.0.24-2cl.i386.rpm
egcs-1.1.2-14cl.i386.rpm
glibc-2.1.2-11cl.i386.rpm
kernel-headers-2.2.14-14cl.i386.rpm
glibc-devel-2.1.2-11cl.i386.rpm

gdb

gdb progama [core]

O uso do parâmetro **core** na linha de comando acima suponhe a existência do arquivo **core** gerado durante a execução do programa devido a ocorrência de erros (use a opção **-g** na compilação de programas C e C++ para gerar este arquivo quando um erro for detectado).

São alguns dos comandos disponíveis dentro do gdb

break função : pára (temporariamente) a execução do programa quando encontra a função especificada.

c : continua a execução após a parada definida pelo comando **break**.

print variável : mostra o conteúdo da variável quando o programa foi encerrado.

run parâmetros : executa o programa (com a lista de parâmetros especificada se esta for definida).

quit : encerra o **gdb**.

getty

Definição

O **getty** é o segundo dos três programas (**init**, **getty** e **login**) usados pelo sistema para permitir o acesso dos usuários. Ele é iniciado pelo processo **init** para:

1. Abrir as linhas tty e configurar o seu modo de funcionamento.
2. Imprimir a identificação de acesso ao sistema e receber o nome do usuário.
3. Iniciar o processo **login** para o usuário.

Na realidade, um **getty** é um console do linux, ou seja, um ambiente para configurar e gerenciar os terminais do linux. Um console, gerado pelo **getty**, pode ser **virtual** ou real (se comunica por porta serial).

Caracteres especiais

O **getty** reconhece algumas sequências especiais na mensagem de entrada e na mensagem de acesso ao sistema (veja **/etc/issue** e **/etc/issue.net**). Por exemplo,

- **** : contra-barra (\).
- **\b** : retorno.
- **\n** : nova linha.
- **\r** : nova linha e posicionamento no seu início.
- **\s** : espaço simples.

Observações

As configurações de velocidade e os parâmetros usados pelo **getty** estão em **/etc/gettydefs**.

Normalmente, as distribuições executam um **getty** de funcionalidade reduzida chamado **mingetty**. Este gerenciador de terminais trabalha apenas com terminais virtuais, mas em compensação, requer para funcionar uma quantidade de memória bem menor que o **getty**.

ghostview ou gv

gv arquivo [options]

São algumas das opções deste comando

-geometry largura x altura : define a largura e a altura da janela do **gv**.

-center : páginas devem ser centralizadas automaticamente.

-page n : inicializa o **gv** exibindo a **n**-ésima página do arquivo.

Comentários sobre as opções do comando

Por exemplo,

gv teste.ps -geometry 700x500 -page 3

exibe a terceira página do arquivo **teste.ps** em uma janela com 700 pixels de largura e 500 pixels de altura.

Observações

É possível que o pacote referente a este aplicativo não tenha sido instalado junto com o sistema. O pacote necessário é **gv-xxx.i386.rpm**, onde **xxx** correspode à versão do pacote.

GID

O termo **GID** significa **Group IDentification**, ou seja, identificação de grupo.

No Linux, os arquivos (e diretórios) são organizados em grupos. O **GID** de um arquivo é inicialmente herdado do usuário que cria o arquivo. Entretanto, é possível modificar o grupo de um arquivo usando o comando **chgrp**.

Um grupo pode ter zero, um ou vários arquivos. Um arquivo pertence a um único grupo,mas os usuários do sistema podem ser membros de um ou mais grupos.

Valores de GID

O valor de **GID** é não negativo, sendo que **valores entre zero e 499** são tipicamente reservados para as contas do sistema e **valores maiores que 499** são alocados para os grupos de usuários do sistema.

O **GID zero** concede (a um usuário ou a um arquivo) acesso irrestrito ao sistema. Embora isso não seja o mesmo que ser o **root**, deve-se usá-lo com cuidado.

Outros comandos

O comando **groups** permite listar os grupos aos quais faz parte um usuário, enquanto o comando **id** identifica o **GID** de usuários.

Além disso, o **root** (adminstrador do sistema) pode usar os comandos **groupadd**, **groupmod** e **groupdel** para, respectivamente, criar, modificar e deletar grupos.

Observações

A lista dos grupos existentes no sistema é armazenada em **/etc/group**.

grep

grep [opções] expressão arquivo

onde

- **expressão** é a palavra ou frase a ser procurada no texto.
- **arquivo** é o nome do arquivo onde será feita a busca.

São algumas das opções deste comando

-c : imprime somente a contagem das linhas com **expressão**.

-i : ignora a diferença entre letras maiúsculas e letras minúsculas.

-l : exibe o nome do arquivo ao invés da saída normal do comando **grep**.

-n : mostra o número de cada linha em **arquivo** com **expressão**.

-v : mostra todas as linhas de **arquivo**, exceto as linhas com **expressão**.

Comentários sobre as opções do comando

Quando a expressão for composta por mais de uma palavra use aspa simples, por exemplo,

grep 'teste de ' projeto.txt

Para ver a lista dos serviços habilitados no **inetd.conf** (a linha não é iniciada com #), digite

grep -v ^# /etc/inetd.conf

Exemplos

Abaixo comentamos o uso do comando **grep** para fazer busca de padrões no arquivo **projeto.txt**.

- O ponto (.) em um padrão representa qualquer caractere.

  grep 'projeto.lin' projeto.txt

- Uma string entre colchetes compara qualquer caractere definido na string.

  grep '[Pp]rojeto' projeto.txt

- Pode-se usar o hífen para indicar uma série de caracteres.

  grep 'Projeto[a-x]' projeto.txt

- Pode-se usar ^ para indicar os caracteres que não estão na lista.

  grep 'Projeto[^a-m]' projeto.txt

- Uma expressão precedida por ^ encontra as linhas iniciadas pela expressão.

  grep '^Projeto' projeto.txt

- Uma expressão seguida por $ encontra as linhas terminada pela expressão.

grep 'Projeto$' projeto.txt

- A barra invertida desativa qualquer significado especial que um caractere possa ter.

  grep 'projeto\.lin' projeto.txt

groupadd

groupadd [opções] grupo

São algumas das opções deste comando

-g gid : fornece a identificação numérica do grupo (**GID**) sendo criado. O valor padrão é o menor valor, maior que 500, que ainda não está alocado a outro grupo. Valores entre 0 e 499 são tipicamente reservados para as contas do sistema.

-o : cria o grupo mesmo que já exista outro grupo com o **GID** especificado (usado em conjunto com a opção **-g**).

-f : força a criação do grupo. Caso o **GID** especificado já exista, o sistema escolhe outro valor (usado em conjunto com a opção **-g**).

-r : cria uma conta do sistema. Por padrão, o valor escolhido é o menor valor disponível entre 0 e 499, a menos que a opção **-g** seja usada.

Comentários sobre as opções do comando

Por exemplo,

groupadd -r teste

cria um grupo chamado **teste**.

Observações

A lista dos grupos existentes no sistema é armazenada em **/etc/group**.

groupdel

groupdel grupo

Comentários sobre as opções do comando

Por exemplo,

groupdel teste

apaga o grupo chamado **teste**.

Observações

A lista dos grupos existentes no sistema é armazenada em **/etc/group**.

ATENÇÃO: deve-se remover os usuários do grupo, antes de apagar o grupo, pois o Linux não faz nenhum tipo de verificação.

groupmod

groupmod [opções] grupo

São algumas das opções deste comando

-g gid : altera a identificação numérica do grupo (**GID**).

-n nome : altera o nome do grupo.

-o : altera o **GID** do grupo mesmo que já exista outro grupo com o **GID** especificado (usado em conjunto com a opção **-g**).

Comentários sobre as opções do comando

Por exemplo,

groupmod -n teste2 teste

altera o nome do grupo **teste** para **teste2**.

Observações

A lista dos grupos existentes no sistema é armazenada em **/etc/group**.

ATENÇÃO: arquivos/diretórios com o **GID** antigo deve ser alterado manualmente. O Linux não faz qualquer tipo de verificação.

groups

groups usuário

Comentários sobre o comando

Por exemplo, o comando

groups aluno

informa os grupos ao quais o usuário **aluno** pertence.

Observações

A lista dos grupos existentes no sistema é armazenada em **/etc/group**.

O uso do comando **groups**, sem parâmetros, faz com que o sistema informe os grupos dos quais o usuário é membro.

grpconv

grpconv

Descrição

Este comando migra as senhas criptografadas do **/etc/group** para o **/etc/gshadow**. Neste caso, a posição da senha em **/etc/group** é preenchida com um **"x"**.

Observação

Para eliminar o arquivo **/etc/gshadow**, use o comando **grpunconv**.

grpunconv

grpunconv

Descrição

Este comando migra as senhas criptografadas do **/etc/gshadow** para o **/etc/group**.

Observação

Para gerar novamente o arquivo **/etc/gshadow**, use o comando **grpconv**.

gunzip

gunzip arquivo

Comentários sobre o comando

Por exemplo, o comando

gunzip teste1.txt

descompacta o arquivo **teste1.txt.gz**.

ghostview ou gv

gv arquivo [options]

São algumas das opções deste comando

-geometry largura x altura : define a largura e a altura da janela do **gv**.

-center : páginas devem ser centralizadas automaticamente.

-page n : inicializa o **gv** exibindo a **n**-ésima página do arquivo.

Comentários sobre as opções do comando

Por exemplo,

gv teste.ps -geometry 700x500 -page 3

exibe a terceira página do arquivo **teste.ps** em uma janela com 700 pixels de largura e 500 pixels de altura.

Observações

É possível que o pacote referente a este aplicativo não tenha sido instalado junto com o sistema. O pacote necessário é **gv-xxx.i386.rpm**, onde **xxx** correspode à versão do pacote.

gzip

gzip [opções] arquivo

O arquivo (ou conjunto de arquivos) é substituído por um arquivo compactado com a extensão **.gz**. Entretanto, são mantidos o dono, as permissões e as datas de modificação do arquivo.

São algumas das opções deste comando

-c : grava o resultado na saída padrão e mantém o arquivo original inalterado.

-d : descompacta (igual ao comando **gunzip**).

-l : lista informações sobre os arquivos compactados/descompactados.

-r : compacta/descompacta recursivamente (navega a estrutura de diretórios recursivamente).

-t : verifica a integridade do arquivo compactado.

Comentários sobre as opções do comando

Por exemplo, para compactar o arquivo **teste1.txt**, basta digitar

gzip teste1.txt

Neste caso, o arquivo **teste1.txt.gz** substitui o arquivo original.

É também possível concatenar vários arquivos juntos. Os comandos

gzip -c teste1.txt > teste.gz
gzip -c teste2.txt >> teste.gz

criam o arquivo **teste.gz** que contém os arquivos **teste1.txt** e **teste2.txt**. Para descompactar o arquivo, basta digitar

gunzip -c teste

O arquivo **teste** é a concatenação dos arquivos **teste1.txt** e **teste2.gz**. Portanto, podemos também criar o arquivo **teste** com o comando

cat teste1.txt teste2.txt > teste

head

head [opções] arquivo

São algumas das opções deste comando

-c [num[bkm]] : mostra os **num** primeiros bytes, kbytes ou Mbytes do arquivo (o padrão é bytes).

-n [num] : mostra as **num** primeiras linhas do arquivo (o padrão é 10).

Exemplos

head -c 200 teste.txt

head -n 20 teste.txt

help

comando --help

ou

help comando

Exemplo

Por exemplo, para sabermos mais sobre o comando **ls** podemos digitar

ls --help

Será então exibida a lista dos parâmetros que podem ser usados com este comando. Entretanto, o parâmetro **--help** não funciona com os comandos internos do **shell** como, por exemplo, o comando **cd**. Neste caso, você pode digitar

help cd

para obter mais informações sobre os parâmetros do comando **cd** (a lista dos comandos internos do **shell** pode ser obtida com o comando **help**).

Observações

Use o comando **more** ou o comando **less** para paginar as informações obtidas com o **help**. Por exemplo,

ls --help | more

hostname

hostname [opções] [computador]

São algumas das opções deste comando

-a : mostra o **alias** da máquina (se for usado).

-i : mostra o(s) endereço(s) IP(s) da máquina.

-s : mostra o nome curto da máquina (até o primeiro '.').

Observações

Se o comando **hostname** for usado sem parâmetros, o nome da máquina é exibido.

Se o comando **hostname** for seguido por um nome, é alterado o nome da máquina.

As informações sobre os **hosts** da rede são armazenadas em **/etc/hosts**.

hwclock

hwclock [opções]

São algumas das opções deste comando

-r ou **--show** : exibe a data e a hora do relógio da máquina.

-w ou **--systohc** : copia a hora do sistema para o relógio da máquina.

-s ou **--hctosys** : copia a hora da máquina para a hora do sistema.

--set --date=novadata : modifica a data e a hora do relógio da máquina.

Comentários sobre as opções do comando

Por exemplo, o comando

hwclock --set --date="9/22/01 18:55:10"

define a nova data como 22 de setembro de 2001 e a nova hora como 18:55:10.

Observações

Na realidade, o comando **hwclock** é uma versão melhorada do antigo comando **clock**. Normalmente, o comando **clock** é apenas um link para o comando **hwclock**.

id

id [opções] [usuário]

São algumas das opções deste comando

-g : lista o número de identificação (**GID**) do grupo primário do usuário.

-G : lista os números de identificação (GID) dos grupos segundários do usuário.

-n : mostra o nome do usuário e/ou o nome do grupo ao invés do número de identificação (deve vir junto com outra opção).

-u : lista apenas o número de identificação (**UID**) do usuário.

Comentários sobre o comando

Por exemplo, o comando

id aluno

informa o **UID** e o **GID** do usuário **aluno**, enquanto o comando

id -nG aluno

informa os nomes dos grupos dos quais o usuário **aluno** é membro.

Observações

O uso do comando **id**, sem parâmetros, faz com que o sistema informe o **UID** do usuário e o **GID** dos grupos dos quais o usuário é membro.

ifconfig

ifconfig [interface]
ifconfig interface [aftype] opções | endereços ...

Descrição

O comando **ifconfig** é usado para configurar (e posteriormente manter) as interfaces de rede. É usado durante o **boot** para configurar a maioria das interfaces de rede para um estado usável. Depois disto, é normalmente necessário somente durante depurações ou quando for necessária uma configuração fina do sistema.

Se nenhum argumento for informado, **ifconfig** somente mostra o estado das interfaces correntemente definidas. Se um argumento **interface** for informado, ele mostra somente o estado da interface informada. De outra forma ele assume que os parâmetros devem ser configurados.

Se o primeiro argumento após o nome da interface for reconhecido como um nome de uma família de endereçamento suportada, esta família de endereçamento é usada na decodificação e apresentação de todos os endereços de protocolos. Atualmente as famílias de endereçamento suportadas incluem **inet** (TCP/IP, default), **inet6** (IPv6) **ax25** (AMPR Packet Radio), **ddp**(Appletalk Phase 2), **ipx** (Novell IPX) and **netrom** (AMPR Packet radio).

São algumas das opções deste comando

interface : o nome da interface de rede. Usualmente é um nome como **eth0**, **sl3** ou algo parecido (um nome de driver de dispositivo seguido por um número).

up : esta flag causa a ativação da interface. É especificada implicitamente se a interface receber um novo endereço (veja abaixo).

down : esta flag desativa o driver desta interface, é útil quando alguma coisa começar a ter problemas.

[-]arp : habilita ou desabilita o uso do protocolo ARP para esta interface. Se o sinal de menos (-) estiver presente a opção é desligada.

[-]trailers : habilita ou desabilita o uso de trailer em frames Ethernet. Não é utilizada na implementação atual do pacote **net-tools**.

[-]allmulti : habilita ou desabilita o modo promiscuous da interface. Isto significa que todos os frames passarão pela camada de rede do kernel, permitindo monitoração da rede.

metric N : este parâmetro configura a métrica da interface. Não é usado atualmente, mas será implementado no futuro.

mtu N : este parâmetro configura a **Unidade Máxima de Transferência** (MTU) de uma interface. Para Ethernet é um número entre 1000-2000 (o padrão é 1500). Para SLIP, use algo entre 200 e 4096. Note que a implementação atual não manipula fragmentação IP ainda, então é melhor configurar a MTU com um tamanho adequado!

dstaddr addr : configura o endereço IP do "outro lado" no caso de um link Ponto-a-Ponto, como PPP. Esta palavra-chave tornou-se obsoleta e deve ser usada a nova palavra-chave**pointopoint**.

netmask addr : configura a máscara de rede IP para esta interface. Este valor assume o padrão usual das classes A, B ou C (deduzindo-o a partir do endereço IP da interface), mas pode ser configurado para qualquer valor para o uso de sub-redes.

irq addr : configura a linha de interrupção (IRQ) usada por este dispositivo. Muitos dispositivos não suportam configuração dinâmica de IRQ.

[-]broadcast [endereço] : se o argumento **endereço** for informado, configura o endereço de protocolo broadcast para esta interface. De outra forma ele somente configura a flag**IFF_BROADCAST** da interface. Se a palavra-chave for precedida por um sinal de menos (-), então a flag é removida.

[-]pointopoint [endereço] : esta palavra-chave habilita o modo ponto-a-ponto da interface, significando que ela é um link direto entre duas máquinas sem ninguém ouvindo. Se o argumento**endereço** for informado, configura o endereço de protocolo do outro lado do link, exatamente como a palavra-chave obsoleta **dstaddr** faz. De outra forma, ela somente configura a flag**IFF_POINTOPOINT** da interface. Se a palavra-chave for precedida por um sinal de menos (-), então a flag é removida.

hw : configura o endereço de hardware para esta interface, se o driver do dispositivo suportar esta operação. A palavra-chave deve ser seguida pelo nome da classe do hardware e o equivalente em ASCII do endereço de hardware. As classes de hardware atualmente suportadas incluem **ether** (Ethernet), **ax25** (AMPR AX.25), **ARCnet** e **netrom** (AMPR NET/ROM).

multicast : inicializa a flag de multicast para a interface. Normalmente, isto não será necessário já que os drivers ajustam as flags corretas por si só.

endereço : o nome ou endereço IP da máquina (um nome de máquina será traduzido para um endereço IP) da interface. Este parâmetro é necessário, apesar da sintaxe atualmente não requisitá-lo.

init

Nível de execução do sistema

O **init** é o pai de todos os processos e é o último passo executado no processo de inicialização do sistema. O **init** procura pelo arquivo **/etc/inittab** que descreve os processos a serem inicializados para o funcionamento do sistema.

O **init** distingue vários níveis de execução (**runlevels**), onde cada nível possui o seu próprio conjunto de processos a serem iniciados. São níveis de execução válidos para o Linux: 0-6, A, B e C. Os níveis de execução 0, 1 e 6 são reservados. O **nível 0** é usado para parar o sistema, o **nível 1** é usado para inicializar o sistema em modo monousuário e o **nível 6** é usado para reinicializar o sistema.

Quando o processo **init** ler o arquivo **/etc/inittab**, ele procura pela entrada **initdefault** que define o nível de execução inicial do sistema. Caso esta entrada não exista ou o arquivo **/etc/inittab**não seja encontrado, será solicitado ao usuário que ele informe o nível de execução do sistema.

Exemplo

Por exemplo, na distribuição **Conectiva Linux 5.0** temos a seguinte definição no arquivo **/etc/inittab**:

```
# Default runlevel. The runlevels used by RHS are:
# 0 - halt (Do NOT set initdefault to this)
# 1 - Single user mode
# 2 - Multiuser, without NFS (The same as 3, if you do not have networking)
# 3 - Full multiuser mode
# 4 - unused
# 5 - X11
# 6 - reboot (Do NOT set initdefault to this)
id:5:initdefault:
```

Podemos a partir do exemplo acima deduzir:

- O **nível 0** pára o sistema.
- O **nível l** inicia o sistema em modo **monousuário**.
- O **nível 2** é o modo multiusuário sem compartilhamento de arquivos.
- O **nível 3** é o modo multiusuário com o número completo de serviços.
- O **nível 4** não é definido. Portanto, é possível desenvolver um sistema de inicialização próprio e implementá-lo através deste nível.
- O **nível 5** inicializa o sistema como um terminal dedicado **X Windows**.
- O **nível 6** reinicializa o sistema.
- O nível de execução inicial do sistema (**initdefault**), neste exemplo, é o nível 5.

Note que as linhas que começam com o símbolo # são linhas de comentário. Apenas a última linha apresentada no exemplo acima é verificada pelo processo **init**.

Todas as configurações referentes aos modos de execução estão no diretório **/etc/rc.d**.

Observações

É possível parar o sistema usando o comando

init 0

ou reinicializar o sistema com o comando

init 6

Para alterar o nível de execução do sistema use o comando **telinit**.

inode

Cada diretório e arquivo do Linux é identificado para o **kernel** como um número de nó i (**inode**).

Um **inode** é, na realidade, uma estrutura de dados que possui informações sobre um determinado arquivo ou diretório como, por exemplo, dono, grupo, tipo e **permissões de acesso**.

O **inode** é exclusivo somente para o dispositivo (partição) dentro do qual ele está contido. Portanto, para identificar unicamente um arquivo, o **kernel** deve ter o número de dispositivo e o **inode**do arquivo.

Um arquivo possui um único **inode**, não importa por quantos nomes este arquivo é identificado no sistema. Logo, é o conjunto de **inodes** que indica o número de arquivos/diretórios que o sistema possui.

insmod

insmod [opções] módulo

O comando **insmod** tenta **linkar** um módulo ao **kernel** que está executando.

São algumas das opções deste comando

-f : força o carregamento do módulo, mesmo que a versão do módulo e do **kernel** não sejam compatíveis.

-p : testa se o módulo pode ser carregado com sucesso.

Observações

Para acrescentar um módulo do **kernel** automaticamente a cada inicialização do sistema, inclua o comando correspondente no arquivo **/etc/rc.d/rc.local** ou no arquivo **/etc/rc.d/rc.sysinit**.

Uma limitação do comando **insmod** é que ele não entende as dependências de módulos. Ao pedir que um determinado módulo seja carregado, este comando não carrega automaticamente os módulos necessários a execução do módulo especificado. O comando **modprobe**, ao contrário do comando **insmod**, carrega os módulos e verifica as dependências.

ipcrm

ipcrm [sem | shm | msg] id

onde **id** identifica o recurso a ser removido.

São algumas das opções deste comando

sem : o recurso a ser removido é um semáforo.

shm : o recurso a ser removido é um segmento compartilhado de memória.

msg : o recurso a ser removido é uma mensagem.

ipcs

ipcs [opções]

São algumas das opções deste comando

-m : informações sobre memória compartilhada.

-q : informações sobre filas de mensagens.

-s : informações sobre cadeias de semáforos.

-a : informações sobre todos os recursos (é o padrão).

Observações

O uso do comando **ipcs**, sem parâmetros, faz com que o sistema forneça informações sobre todos os recursos.

ispell

ispell [opções] arquivo...

São algumas das opções deste comando

-t : o arquivo de entrada está no formato TEX ou LATEX.

-b : cria um arquivo de backup com extensão **.bak**.

-x : não cria arquivo de backup.

-d : especifica o idioma da verificação ortográfica (por exemplo, **english** para inglês britânico, **american** para inglês americano, **francais** para francês, etc).

Comentários sobre as opções do comando

O Linux fornece dicionários para correção ortográfica de vários idiomas. Para ver qual o dicionário padrão utilizado pelo aplicativo **ispell**, basta digitar

printenv DICTIONARY

Para alterar o idioma padrão do aplicativo, é preciso alterar a **variável de ambiente** **DICTIONARY**. Por exemplo, suponha que queremos que a língua inglesa seja o idioma padrão utilizado pelo**ispell**, devemos então digitar

DICTIONARY=english

Note que a alteração feita acima na variável **DICTIONARY** só é válida enquanto o usuário estiver logado ao sistema. Para alterar em caráter permanente o idioma padrão do sistema, deve-se incluir o comando acima no arquivo de **profile** do usuário (na distribuição **Conectiva Linux**, este arquivo é o **.bash_profile** que fica no diretório **raiz** do usuário).

Observações

Os arquivos de dicionários disponíveis no seu sistema e utilizados pelo **ispell** são armazenados em **/usr/lib/ispell/**.

jobs

jobs

São algumas das opções deste comando

-l : exibe todas as informações sobre os processos do usuário.

-n : exibe apenas os números dos processos do usuário.

Comentários sobre as opções do comando

Por exemplo, o comando

jobs -l

fornece o número, o estado e o comando que inicializou cada processo associado ao usuário.

Observações

Os processos são colocados em segundo plano (**background**) através do símbolo **&** ou do comando **bg**. Isto significa que o processo continua executando normalmente, apenas não estar mais associado a um terminal.

kernel

Definição

O Linux, na realidade, é apenas o nome do **kernel** do sistema operacional. Uma distribuição Linux (por exemplo, **Conectiva Linux**, **Red Hat**, etc) é composta por uma coleção de aplicativos mais o **kernel Linux**.

O **kernel** do Linux é distribuído sob os termos da **GPL** (General Public License). Isto significa que o sistema é distribuído sem cobrança de taxas. Além disso, você pode alterar o código para atender as suas necessidades e distribui-lo livremente.

Uma distribuição Linux pode ser gratuita (por exemplo, **Red Hat**, **Debian**, etc) ou pode ser paga (por exemplo, **Susie**).

Versão do Kernel

O número da versão do **kernel** é composto de três partes: o número principal, o número secundário e o nível de revisão/manutenção. O primeiro número dificilmente muda. O segundo número muda quando mudanças substanciais são feitas no **kernel**, onde números pares (0,2,4,...) indicam versões estáveis e números ímpares (1,3,5,...) indicam versões de teste. A terceira parte (composta por dois números) muda freqüentemente.

Por exemplo, suponha a versão **2.2.5-15** do **kernel**. Neste caso, temos uma versão estável (2) onde o número da revisão é 5 e o nível de manutenção é 15.

A versão usada no seu sistema é indicada pelo arquivo **vmlinux**.

kill

kill [opções] [pid]

onde **pid** é o número de identificação do **processo** e pode ser obtido com o comando **ps**. Caso nenhum sinal seja especificado com o comando **kill**, o sinal **SIGTERM** (finaliza processo) é enviado.

São algumas das opções deste comando

-l - lista os sinais que podem ser enviados a um processo junto com o comando **kill**.

-s sinal - especifica o sinal a ser enviado, onde **sinal** pode estar no formato texto ou no formato de número.

-sinal - especifica o sinal a ser enviado, onde **sinal** é um número.

Comentários sobre as opções do comando

Suponha que o resultado do comando **ps** seja como mostrado abaixo.

```
PID   TTY  TIME     CMD
841   pts/0 00:00:00 bash
1314  pts/0 00:00:00 teste
```

Neste exemplo temos dois processos sendo executados: **bash** e **teste**. Para finalizar o processo **teste**, basta digitar

kill 1314

O comando acima corresponde a enviar o sinal **SIGTERM** (ou 15) ao processo.

Caso o processo não seja encerrado, você pode forçar o término do processo com o seguinte comando

kill -9 1314

O comando acima corresponde a enviar o sinal **SIGKILL** ao processo. Esta opção informa ao sistema que o comando **kill** não pode ser ignorado, ele deve ser imediatamente processado. Neste caso, o sistema não se preocupa em salvar dados ou apagar arquivos temporários criados durante a execução do processo.

Outros sinais que podem ser enviados com o comando **kill**:

- **SIGHUP (1)** : reinicializa o processo (o processo ler novamente os seus arquivos de configuração).
- **SIGSTP (20)** : suspende a execução de um processo.

Lista dos nomes e números dos sinais

Use o comando **kill -l** para obter a lista dos sinais no Linux.

1) SIGHUP	2) SIGINT	3) SIGQUIT	4) SIGILL
5) SIGTRAP	6) SIGABRT	7) SIGBUS	8) SIGFPE
9) SIGKILL	10) SIGUSR1	11) SIGSEGV	12) SIGUSR2
13) SIGPIPE	14) SIGALRM	15) SIGTERM	17) SIGCHLD
18) SIGCONT	19) SIGSTOP	20) SIGTSTP	21) SIGTTIN

22) SIGTTOU	23) SIGURG	24) SIGXCPU	25) SIGXFSZ
26) SIGVTALRM	27) SIGPROF	28) SIGWINCH	29) SIGIO
30) SIGPWR	31) SIGSYS	32) SIGRTMIN	33) SIGRTMIN+1
34) SIGRTMIN+2	35) SIGRTMIN+3	36) SIGRTMIN+4	37) SIGRTMIN+5
38) SIGRTMIN+6	39) SIGRTMIN+7	40) SIGRTMIN+8	41) SIGRTMIN+9
42) SIGRTMIN+10	43) SIGRTMIN+11	44) SIGRTMIN+12	45) SIGRTMIN+13
46) SIGRTMIN+14	47) SIGRTMIN+15	48) SIGRTMAX-15	49) SIGRTMAX-14
50) SIGRTMAX-13	51) SIGRTMAX-12	52) SIGRTMAX-11	53) SIGRTMAX-10
54) SIGRTMAX-9	55) SIGRTMAX-8	56) SIGRTMAX-7	57) SIGRTMAX-6
58) SIGRTMAX-5	59) SIGRTMAX-4	60) SIGRTMAX-3	61) SIGRTMAX-2
62) SIGRTMAX-1	63) SIGRTMAX		

Observações

Suponha que o uso do comando **jobs -l** forneça a seguinte saída

```
[1]     936 Parado          vi teste1
[2]     937 Parado          vi teste2
[3]-    938 Parado          vi teste3
[4]+    974 Parado          man ls
```

Neste exemplo, temos 4 processos parados. O primeiro número de cada linha indica a ordem de entrada do processo em **background**, enquanto o segundo indica o **PID** do processo. Para matar o segundo processo listado acima, podemos digitar

kill %2

ou

kill 937

killall

killall [opções] [nome]

onde **nome** é o nome de identificação do processos.

São algumas das opções deste comando

-i - pede confirmação antes de matar o processo.

-l - lista os sinais que podem ser enviados a um processo junto com o comando **killall**.

-v - informa se o sinal foi enviado com sucesso.

Comentários sobre as opções do comando

Por exemplo, o comando

killall -9 /usr/local/netscape/mozilla-bin

mata todos os processos relacioanados ao Netscape.

kudzu

kudzu [opções]

Se o **kudzu** detectar alguma alteração na configuração do sistema, ele tentará acionar automaticamente os módulos necessários para o funcionamentos dos dispositivos em questão.

São algumas das opções deste comando

-q : configura apenas os dispositivos que não necessitam de argumentos inseridos pelo usuário.

-s : não altera os dispositivos instalados.

-t num : encerra o aplicativo após **num** segundos se nenhum diálogo for iniciado com o **kudzu**.

Observações

A lista dos dispositivos de hardware instalados está em **/etc/sysconfig/hwconf**.

O arquivo de configuração dos **módulos** do **kernel** é **/etc/conf.modules**.

last

last [opções]

São algumas das opções deste comando

- **-a** - exibe o nome da máquina onde foi efetuado o **login**.
- **-d** - exibe o número de IP da máquina onde foi efetuado o **login**.
- **reboot** - exibe um registro de todas as reinicializações desde a criação do arquivo.

Comentários sobre as opções do comando

Por exemplo, o comando

last -d

informa os endereços IP das conexões não locais.

Observações

Caso nenhum argumento seja passado, o comando **last** exibe todas as informações armazenadas no arquivo **/var/log/wtmp** de todos os usuários do sistema.

Se o arquivo **wtmp** não existir, as informações não são gravadas.

lastlog

lastlog [opções]

São algumas das opções deste comando

-u usuário : exibe informações referentes apenas ao usuário especificado.

-t dias : exibe informações referentes aos últimos **t** dias. Esta opção anula a opção **-u**.

Comentários sobre as opções do comando

Por exemplo, o comando

lastlog -u aluno

informa o dia e a hora que o usuário **aluno** logou no sistema pela útima vez. Enquanto o comando

lastlog -t 3

exibe a lista dos usuários que logaram no sistema nos útimos 3 dias e informa o dia e a hora do útimo acesso de cada um desses usuários.

Observações

Caso nenhum argumento seja passado, o comando **lastlog** exibe todas as informações armazenadas no arquivo **/var/log/lastlog** de todos os usuários do sistema.

latex

latex arquivo

onde **arquivo** corresponde a um texto ASCII com macros TEX (veja **tex** para mais informações). O arquivo gerado com este comando tem extensão **.dvi** e pode ser formatado para POSTSCRIPT com o aplicativo **dvips**.

Observações

Em algumas distribuições do Linux, o **latex** não é instalado automaticamente. Para poder usar esta ferramenta, você deve instalar os seguintes pacotes:

dialog-0.6-17cl.i386.rp
tetex-1.0.7-2cl.i386.rpm
tetex-afm-1.0.7-2cl.i386.rpm
tetex-doc-1.0.7-2cl.i386.rpm
tetex-dvilj-1.0.7-2cl.i386.rpm
tetex-dvips-1.0.7-2cl.i386.rpm
tetex-latex-1.0.7-2cl.i386.rpm
tetex-xdvi-1.0.7-2cl.i386.rpm

ldconfig

Para atualizar o arquivo de cache **/etc/ld.so.cache** a partir do arquivo **/etc/ld.so.config**, basta digitar

ldconfig

less

less [opções] arquivo

São algumas das opções deste comando

+ : se a opção começa com este sinal, então a opção é usada como um comando inicial para o aplicativo **less**.

-? ou **--help**: lista um sumário das opções do aplicativo **less**.

+num : mostra arquivo a partir da linha **num**.

-N : numera as linhas do arquivo.

-n : não numera as linhas do arquivo (é o padrão).

F : continua indefinidamente tentando ler caracteres ao final do arquivo, assumindo que o arquivo está crescendo. O comportamento é semelhante ao comando **tail -f**.

Teclas

Você pode utilizar as seguintes teclas com o **less**:

- para sair do aplicativo digite **q** (de **quit** em inglês que significa sair);
- use as teclas **Page-Down**, **Ctrl+F** ou **Space** para avançar nas páginas;
- use as teclas **Page-Up** ou **Ctrl+B** para voltar as páginas;
- use **Enter** para avançar apenas uma linha por vez;
- digite **h** para ver a lista das teclas disponíveis para navegação no aplicativo.

Também é possível personalizar as teclas a serem usadas no aplicativo **less** usando o comando **lesskey**.

Comentários sobre o aplicativo

Para redirecionar a saída de um comando para o **less** use o **pipe**. Por exemplo, o comando

ls | less

faz com que a listagem dos arquivos do diretório corrente seja paginada pelo aplicativo **less**.

Para examinar arquivos que estão sendo constantemente atualizados use a opção **F**. Por exemplo,

less +F /var/log/messages

mostra as últimas linhas do arquivo de mensagens do sistema que é modificado pelo **syslog** (veja arquivo **/etc/syslog.conf** para mais detalhes).

Observação

Na realidade, o aplicativo **less** é uma versão melhorada do aplicativo **more**.

lesskey

lesskey [-o arq_saída] [arq_entrada]

onde

- **arq_entrada** é um arquivo texto com a descrição das teclas;
- **arq_saída** é o nome do arquivo binário gerado pelo **lesskey** e que será utilizado pelo comando **less**.

Se o arquivo de saída já existir, ele será apagado e criado novamente.

Arquivo padrão

O nome padrão do arquivo de saída do comando **lesskey** é armazenado na **variável de ambiente** **LESSKEY**. Para ver o nome deste arquivo, basta digitar

printenv LESSKEY

Definição de teclas

Podemos, por exemplo, criar um arquivo chamado **teste** que possui o seguinte conteúdo

^l forw-screen
^s back-screen
j quit

O arquivo acima define a tecla **CTRL+l** para avançar uma página, a tecla **CTRL+s** para voltar uma página e a tecla **j** para encerrar o aplicativo. É importante observar que as teclas padrão do aplicativo **less** que não são modificadas pelo comando **lesskey** podem ainda ser usadas dentro do paginador.

As teclas definidas no arquivo **teste** só poderão ser utilizadas no **less** após a execução do comando

lesskey teste

Observações

Entretanto, nem sempre o usuário tem permissão para executar o comando **lesskey**. Suponha, por exemplo, que a saída padrão do comando **lesskey** é o arquivo **/etc/lesskey** e que somente o **root** (administrador do sistema) tem autorização para escrever neste arquivo. Você pode então digitar

lesskey -o .less teste

O comando acima cria o arquivo binário **.less** com a definição das novas teclas. Para torná-lo arquivo padrão do comando **less** no seu ambiente de trabalho, basta alterar o conteúdo da variável **LESSKEY** com o seguinte comando

LESSKEY=.less

O comando acima assume que o arquivo **.less** foi criado no diretório raiz do usuário.

lilo

Definição

O nome LILO corresponde a **Linux Loader**, ou seja, o carregador do Linux. Isto significa que o LILO gerencia o processo de inicialização da máquina, permitindo que o usuário escolha qual o sistema operacional que será executado (se houver mais de um sistema operacional residente na máquina).

Pode-se instalar o **LILO** no **MBR** (Master Boot Record) ou no primeiro setor da partição de inicialização.

O **MBR** é o único setor de inicialização da unidade de disco carregado pela ROM BIOS. O MBR contém um pequeno programa de carregamento e uma tabela de partição.

Se o padrão DOS MBR for usado, ele carrega o setor de inicialização daquela partição ativa e então passa o controle para o setor de inicialização. Assim, tanto o MBR quanto o setor de boot da partição ativa estão envolvidos no processo de inicialização.

O LILO pode ser usado no lugar do DOS MBR. Assim, a inicialização não fica limitada à partição ativa do disco de boot, pois o LILO possui muito mais capacidade do que o DOS MBR.

Inicialização do sistema

São passos do processo de inicialização do sistema:

- o programa ROM BIOS carrega o setor de inicialização a partir do dispositivo de boot;
- o setor de inicialização contém ou carrega o carregador Linux (**LILO**);
- o **LILO** carrega o **Linux Kernel**;
- o **kernel** dá início ao processo **init** que carrega todos os serviços Linux.

Observações

O **LILO** pode gerenciar mais de sessenta imagens diferentes de inicialização e pode estar instalado em um disquete, em uma partição de disco rígido ou como arquivo de inicialização mestre.

O **LILO** é configurado pelo arquivo **/etc/lilo.conf**.

A instalação do **LILO** é feita através do programa **/sbin/lilo**. Este programa instala e atualiza o carregador de inicialização **LILO**. Portanto, após alguma modificação no arquivo **lilo.conf**, deve-se digitar **lilo** na linha de comando para atualizar o carregador.

Se você já instalou o sistema e não possui disquete de boot (ou perdeu o disquete), basta usar o seguinte comando para criar um disco de boot

/sbin/lilo -b /dev/fd0

linux_logo

linux_logo [opções]

São algumas das opções deste comando

-ascii : logotipo monocromático.

-banner : novo logotipo do linux. É o padrão.

-classic : logotipo original do Linux.

-f : limpa a tela antes de mostrar o logotipo.

-g : mostra apenas as informações do sistema.

-l : mostra apenas o logotipo.

Comentários sobre as opções do comando

Por exemplo, o comando

linux_logo -f -ascii

limpa a tela e exibe o logotipo monocromático do Linux.

linuxconf

Para executar o Linuxconf

- Usando a interface texto:

  linuxconf --text

- Usando a interface gráfica (a partir de um terminal X):

  linuxconf

- Usando a interface web:

  http://127.0.0.1:98/

ln

ln [opções] origem destino

onde

- **origem** é o arquivo a partir do qual será feito o link;
- **destino** é o nome do link que será criado.

São algumas das opções deste comando

-d : cria uma ligação direta (**hard link**), é o padrão;

-s : cria uma ligação simbólica (**soft link**);

-v : mostra o nome de cada arquivo antes de criar o link.

Comentários sobre as opções do comando

Há dois conceitos de **ligação** em Unix, normalmente chamados ligação direta (**hard link**) e ligação simbólica (**soft link**).

Uma ligação direta é somente um nome para um arquivo (e um arquivo pode ter diversos nomes). O arquivo será removido do disco quando o último nome for removido. Não há algo como um nome original: todos os nomes tem o mesmo status.

Uma ligação simbólica ou **dymlink**, é um pequeno arquivo especial que contém um caminho. Ligações simbólicas podem apontar para arquivos em diferentes sistemas de arquivo e não necessitam apontar para arquivos que realmente existem.

Em uma **ligação direta** os arquivos possuem o mesmo **inode**. Enquanto que em uma **ligação simbólica** os arquivos apontam para a mesma área de dados, mas possuem diferentes **inodes**.

Exemplos

Para criar, no diretório atual, um **link** simbólico chamado **teste** para o arquivo **testes/teste1.txt**, basta digitar

ln -s testes/teste1.txt teste

Podemos também criar **links** para vários arquivos de uma só vez, mas neste caso devemos especificar o diretório destino ao invés do nome do **link**. Por exemplo, suponha que queremos criar**links** no diretório atual para todos os arquivos do diretório **testes** que começam por **scrip**, basta digitar

ln -sv testes/script* .

Note que, no comando acima, a opção **-v** lista os **links** sendo criados e que o ponto ('.') representa o diretório atual.

Observações

Não é possível criar vínculos diretos para diretórios e nem para sistemas de arquivos.

locate

locate expressão

Definição

O comando **locate** utiliza um banco de dados de nomes de arquivos, chamado de **locatedb**, para efetuar a pesquisa. Esta base de dados é criada/atualizada pelo administrador do sistema (**root**) através do comando **updatedb** e é armazenada em **/var/lib** (na distribuição **Conectiva Linux**, o banco de dados corresponde ao arquivo **/var/lib/slocate/slocate.db** e também pode ser gerado através do comando **/etc/cron.daily/slocate.cron**).

Exemplo

Por exemplo, o comando

locate *cs*log*

pode apresentar a seguinte saída

/etc/csh.login
/etc/csh.logout

login

login [opções]

Definição

O comando **login** é usado para se obter acesso ao sistema. Além disso, ele pode ser usado para alterar de um para outro usuário a qualquer tempo.

O **login** é o terceiro dos três programas (**init**, **getty** e **login**) usados pelo sistema para permitir o acesso dos usuários.

São algumas das opções deste comando

nome : muda para o usuário especificado por **nome**.

-p : usado por **getty** para dizer ao **login** não destruir o ambiente.

-h nome da máquina : usado por outros servidores para passar o nome para o servidor remoto. Somente o superusuário pode usar esta opção

-f nome : usado para evitar uma segunda autenticação de acesso. Isto não funciona para o superusuário.

Observações

Caso nenhum argumento seja passado, o **login** solicita o nome do usuário.

Caso o usuário não seja o **root** e se o **/etc/nologin** existir, o conteúdo deste arquivo será listado na tela e o login será terminado.

Caso o usuário seja o **root**, então o acesso deve ocorrer em um terminal listado no **/etc/securetty**. Falhas serão listadas através do **syslog**.

lpc

lpc [comando]

A sigla **lpc** significa **line-printer control**.

Comandos

abort impressora : interrompe a impressão atual e desativa a impressora especificada.

clean impressora : remove todos os arquivos da fila da impressora especificada.

disable impressora : desabilita a fila de impressão.

enable impressora : habilita a fila de impressão.

restart impressora : reinicializa o daemon **lpd**.

start impressora : ativa a impressão da impressora especificada.

status impressora : verifica o status da impressora especificada.

stop impressora : pára a impressão após o término do serviço atual.

topq impressora num : insere o job de número **num** no topo da fila de impressão.

quit : sai do **lpc** (caso o usuário tenha digitado apenas **lpc**).

Caso a impressora não seja especificada nos comandos acima, a ação será executada para todas as impressoras do sistema.

Observações

O uso do comando **lpc**, sem especificar nenhum comando de controle, faz com que o sistema entre no ambiente do **lpc**. Para retornar ao prompt do sistema, basta digitar **quit**.

lpq

lpq [-P impressora]

onde **impressora** é o nome da impressora no sistema.

A sigla **lpq** significa **line-printer queue**.

Observações

Se o nome da impressora não for fornecido, será usado o nome da impressora padrão.

lpr

lpr [-P impressora] arquivo

onde

- **impressora** é o nome de uma impressora no sistema;
- **arquivo** é o nome do arquivo a ser impresso.

Observações

Se o nome da impressora não for fornecido, será usada a impressora padrão do sistema.

lprm

lprm [opções]

São algumas das opções deste comando

-a : remove arquivos de todas as filas de impressão as quais o usuário tem acesso.

-Pimpressora : especifica o nome da impressora. Caso não seja fornecido o nome da impressora, a impressora padrão é usada.

all : se este parâmetro for fornecido pelo **root**, todos os arquivos da fila de impressão são apagados; se o parâmetro for digitado por um usuário comum, apenas os arquivos do usuário são apagados.

num : remove o arquivo identificado pelo número **num** (use o comando **lpq** para obter o número de identificação de um arquivo na fila de impressão).

usuário : apaga todos os arquivos do usuário especificado (apenas o **root** pode fornecer este parâmetro).

Exemplos

Por exemplo, o comando

lprm -a all

remove todos os arquivos de todas as filas de impressão. Enquanto o comando

lprm -Pteste 30 12

remove os arquivos de números 30 e 12 da impressora **teste**.

ls

ls [opções] diretório...

São algumas das opções deste comando

-a : lista todos os arquivos do diretório (inclusive os arquivos ocultos).

-i : lista o número do índice (**inode**) de cada arquivo. Se dois arquivos em um mesmo sistema de arquivos possuem o mesmo **inode**, então ele estão vinculados a um mesmo arquivo físico (ligação direta).

-l : lista permissões, número de entidades (se for diretório, mostra a quantidade de subdiretórios existentes dentro dele; se for arquivo, mostra o número de referências que apontam para o arquivo), dono, grupo, tamanho do arquivo, data e hora da última atualização e o nome do arquivo.

-m : listagem de arquivos e diretórios separados por vírgulas.

-o : listagem longa sem os donos dos arquivos.

-r : listagem em ordem reversa.

-F : adiciona um "/" no final dos nomes de diretórios, um "*" no final dos nomes de arquivos executáveis, um "@" no final dos nomes de links, etc.

-R : lista diretórios e subdiretórios recursivamente.

-S : classifica os arquivos por tamanho.

--color : usa as cores definadas no arquivo **/etc/DIR_COLORS**.

Exemplos

Para listar todos os arquivos (inclusive os ocultos) de um diretório no formato longo, basta digitar

ls -al diretório

O comando

ls /home/aluno

lista todos os arquivos do subdiretório **aluno** que se encontra no diretório **/home**, enquanto o comando

ls /etc /usr

lista todos os arquivos e subdiretórios dos diretórios **/etc** e **/usr**.

lsmod

Exemplos

Abaixo mostramos um exemplo de uma possível saída deste comando. Cada linha do exemplo mostra as seguintes informações: nome, tamanho, contador de uso e lista dos módulos que referenciam o módulo em questão (dependem do módulo). A palavra **autoclean** indica que o módulo é removido automaticamente da memória quando não está sendo utilizado.

Module	Size	Used	by
parport_pc	5620	0	(autoclean)
parport	7124	0	(autoclean) [parport_pc]
nfsd	149048	8	(autoclean)
lockd	30888	1	(autoclean) [nfsd]
sunrpc	52548	1	(autoclean) [nfsd lockd]
aic7xxx	105584	5	

Observações

Na realidade, o comando **lsmod** apenas exibe o conteúdo do arquivo **/proc/modules**.

make

make [opções] alvo...

O utilitário **make** é utilizado para verificar quais arquivos precisam ser recompilados (foram alterados desde a última compilação) e para emitir a ordem de compilação (e **linkedição**).

São algumas das opções deste comando

-f arquivo : usa o arquivo especificado como **makefile**.

-e : variáveis de ambiente possuem precedência sobre variáveis do arquivo **makefile**.

Comentários sobre as opções do comando

Ao iniciar o comando **make** sem a especificação de um arquivo (**opção -f**), o utilitário faz a seguinte seqüência de busca no diretório atual :

1. GNUmakefile
2. makefile
3. Makefile

O utilitário executa apenas o primeiro arquivo encontrado. Caso não encontre nenhum desses arquivos no diretório atual e exista um subdiretório **RCS**, é então feita a pesquisa por um arquivo**RCS** com um dos três nomes descritos acima (na mesma ordem de seqüência). Caso um arquivo **RCS** seja encontrado, o **RCS** é chamado para criar uma cópia do arquivo no diretório atual e então este arquivo é executado pelo utilitário.

O comando **make** usa o conceito de alvo para definir os comandos a serem executados. Entenda por alvo um conjunto de instruções. Um alvo possui a seguinte estrutura dentro de um arquivo**makefile**

nome_alvo : [dependências]
comandos

onde temos a identificação do alvo, os pré-requisitos necessários para executar o alvo e os comandos associados ao alvo em questão. Um nome de alvo sempre inicia no começo da linha e é seguido por um sinal de dois-pontos ("**:**"). Um alvo pode ter como dependência uma lista de outros alvos ou uma lista de arquivos. Quando a dependência é uma lista de alvos, então cada alvo da lista deve ser executado antes do alvo especificado. Quando a dependência é uma lista de arquivos, então cada um dos arquivos da lista deve existir para o alvo poder ser executado.

Outra importante observação em relação a estrutura mostrada acima diz respeito a posição onde são escritas as linhas de ação (comandos). Em um arquivo **make**, os comandos devem iniciar com um caractere de tabulação. Portanto, deve-se pressionar a tecla **Tab**, antes de digitar os comandos do alvo.

Na linha do comando **make** pode-se definir um ou mais alvos. Caso nenhum alvo seja definido, o utilitário executa o primeiro alvo especificado dentro do arquivo.

Exemplo

Vejamos um exemplo.

```
1 Objetivo = Estudo do makefile
2 alvo1: alvo3
3 @echo $(Objetivo)
4 alvo2:
5 @echo este eh o alvo2 do $(Objetivo)
6 alvo3:
7 @echo alvo3 do makefile
```

Inicialmente é definida no **makefile** acima uma macro chamada **Objetivo**. Em seguida, são definidos três alvos: **alvo1**, **alvo2** e **alvo3**. Note que apenas o **alvo1** tem uma dependência: **alvo3**. Isto significa que antes de executar **alvo1**, o utilitário deverá executar **alvo3**. Além disso, observe que cada alvo usa o comando **echo** que exibe strings e conteúdo de macros (o símbolo @antes do comando **echo** apenas evita que o comando também seja exibido).

No primeiro alvo temos o símbolo **$** seguido pelo nome da macro entre parênteses. O símbolo **$** informa ao comando **@echo** para substituir o nome da macro pelo seu conteúdo (os parênteses não são exigidos se o nome da macro tem apenas uma letra). No segundo alvo temos uma string seguida pela macro **Objetivo**. E no terceiro alvo temos apenas uma string. Abaixo mostramos possíveis linhas de comando e suas saídas para este **makefile**.

1) make
alvo3 do makefile
Estudo do makefile

2) make alvo1
alvo3 do makefile
Estudo do makefile

3) make alvo2
este eh o alvo2 do Estudo do makefile

4) make alvo3
alvo3 do makefile

5) make alvo1 alvo2 alvo3
alvo3 do makefile
Estudo do makefile
este eh o alvo2 do Estudo do makefile
make: `alvo3' is up to date.

6) make alvo3 alvo2 alvo1
alvo3 do makefile
este eh o alvo2 do Estudo do makefile
Estudo do makefile

7) Objetivo=teste make -e (ou make -e Objetivo=teste)
alvo3 do makefile
teste

Podemos fazer algumas observações em relação ao exemplo acima:

1. Os comandos **make** e **make alvo1** são idênticos pois quando o alvo não é especificado na linha de comando, o utilitário sempre executa o primeiro alvo especificado no arquivo**makefile**.
2. O quinto exemplo acima mostra uma mensagem de advertência do **make**. Este exemplo solicita a execução de **alvo1**, **alvo2** e então **alvo3**. Só que **alvo3** é pré-requisito de **alvo1** e portanto, o utilitário executa **alvo3**, **alvo1** e **alvo2**, quando chega para executar o **alvo3**, ele verifica que este já foi executado e exibe a mensagem de advertência.
3. O sexto exemplo executa inicialmente **alvo3** e **alvo2**. Quando vai executar **alvo1**, verifica que este requer a execução de **alvo3** antes de executar **alvo1**. Como **alvo3** já foi executado,**alvo1** é executado imediatamente.
4. O último exemplo define uma variável de ambiente também chamada **Objetivo**; o uso da opção **-e** com o comando **make**, informa que o conteúdo desta variável de ambiente deve sobrepor o conteúdo da macro de mesmo nome definida dentro do arquivo **makefile**.

Observações

Embora não exista um padrão formal para nomes de alvos, normalmente, são adotados os seguintes nomes de alvos nos arquivos **makefiles**:

- **all** : nome do primeiro alvo e portanto, o nome padrão.
- **clean** : usado para apagar arquivos-objeto (***.o**) e outros arquivos temporários.
- **clobber** : usado para apagar arquivos executáveis.
- **install** : usado para instalar programas.
- **distclean** : usado para apagar todos os arquivos que não fazem parte da distribuição original.

No exemplo mostrado acima, usamos uma macro de nome **Objetivo**. Esta macro foi definida pelo usuário dentro do **makefile**. Porém, podemos também fazer uso de macros embutidas, ou seja, de macros internamente mantidas pelo **make** e que portanto, podem ter o seu conteúdo alterado durante a execução e não precisam ser definidas pelo usuário. São alguns exemplos de macros embutidas:

- **$*** : nomes dos arquivos dependentes sem sufixo (extensão).
- **$@** : nome do arquivo alvo com sufixo (extensão).
- **$<** : nomes dos arquivos dependentes com sufixo (extensão).

O comando **make** baseia-se bastante no uso de sufixos de arquivo. São alguns sufixos de arquivos reconhecidos pelo **make**:

- **.c** : arquivo-fonte em linguagem C
- **.cc** : arquivo-fonte em linguagem C++
- **.h** : arquivo de cabeçalho em linguagem C
- **.o** : arquivo-objeto compilado
- **.s** : arquivo-fonte assembler
- **.sh** : arquivo shell

A partir dos sufixos de arquivo podemos definir regras de inferências nos arquivos **makefiles**. Estas regras definem como arquivos são gerados a partir de outros arquivos. Podemos, por exemplo, escrever

.c.o : gcc -c -o $@ $<

A regra de inferência acima define como um arquivo com sufixo **.o** é gerado a partir de um arquivo **.c**. Note que usamos duas macros embutidas: a primeira macro (**$@**) fornece o nome do arquivo final com sufixo **.o** e a segunda (**$<**) fornece o nome do arquivo original com sufixo **.c**. Para a regra de inferência mostrada acima poderíamos, por exemplo, criar o seguinte alvo

teste.o : teste.c funcoes.h

Neste caso, o comando **make teste.o** faz com que o utilitário **make** execute o seguinte comando

gcc -c -o teste.o teste.c funcoes.h

onde a opção **-c** define apenas compilação e a opção **-o** define o nome do arquivo de saída. Os arquivos **teste.c** e **funcoes.h** são os arquivos de entrada. Uma observação final em relação a regras de inferência é que não se deve definir dependências (pré-requisitos) para regras de inferência. Em algumas plataformas UNIX, a regra com pré-requisitos é interpretada como alvo.

Mais exemplos

Vejamos agora um exemplo completo de um **makefile**. O exemplo mostrado abaixo gera o executável **solv** a partir dos arquivos **solvers.o**, **option_1.o**, **option_2.o**, **option_3.o** e **misc.o**.

```
1 CC=gcc
2 INCLUDEDIR=.
3 FLAGS=-g -I${INCLUDEDIR}
4 .c.o:
5 ${CC} ${FLAGS} -c -o $@ $<
6 OBJS = solvers.o option_1.o option_2.o option_3.o misc.o
7 solv: objects
8 ${CC} -o $@ ${OBJS} -g -lm
```

```
9 objects: ${OBJS}
10 solvers.o: solvers.c define.h misc.c
11 option_1.o: option_1.c define.h misc.c
12 option_2.o: option_2.c define.h misc.c
13 option_3.o: option_3.c define.h misc.c
14 misc.o: misc.c define.h
15 clean:
16 rm -fr *.o
```

Inicialmente, são definidas 3 macros: **CC**, **INCLUDEDIR** e **FLAGS**. Na linha 4 é definida uma regra de inferência para gerar os arquivos com sufixo **.o** a partir dos arquivos com sufixo **.c**. Na linha 6 é definida a macro **OBJS**. Nas linhas de 7 a 16 são definidos 8 alvos: **solv**, **objects**, **solvers.o**, **option_1.o**, **option_2.o**, **option_3.o**, **misc.o** e **clean**. O primeiro alvo (**solv**) é o alvo padrão, ou seja, o alvo que será executado caso o usuário digite **make**. O alvo **solv** tem como pré-requisito o alvo **objects** e como linha de ação o comando **gcc**. O alvo **objects** não tem linhas de ação, apenas pré-requisitos: os alvos definidos nas linhas de 10 a 14. Note que estes alvos lista apenas as dependências, portanto será utilizada a regra de inferência definida nas linhas 4 e 5 na execução de cada alvo. Por exemplo, na geração do arquivo **solvers.o** o seguinte comando é executado

```
gcc -g -I. -c -o solvers.o solvers.c define.h misc.c
```

onde **-g**, **-I** e **-c** correspodem, respectivamente, as opções de depuração, inclusão de diretório no caminho de busca e apenas compilação. O alvo **objects** é usado para gerar os arquivos definidos na macro **OBJS**. Após a conclusão do alvo **objects**, o comando retorna ao alvo **solv** que executa a seguinte linha de ação

```
gcc -o solv solvers.o option_1.o option_2.o option_3.o misc.o
```

e finaliza a execução do **makefile**. Caso o usuário tenha digitado **make clean**, o alvo **clean** é executado. Este alvo apaga todos os arquivos com sufixo **.o** do diretório. Note que também é possível utilizar o **makefile** para gerar apenas os arquivos **.o**, pois existem alvos definidos com os nomes desses arquivos. Por exemplo, o comando

```
make option_1.o
```

gera o arquivo **option_1.o**.

Vejamos um segundo exemplo de **makefile**. O exemplo mostrado abaixo gera o executável **teste** a partir dos arquivos **teste.o**, **prog1.o** e **prog2.o**.

```
1 CC = gcc
2 OBJS = teste.o prog1.o prog2.o
3 .c.o:
4 $(CC) -c -o $@ $<
5 all: $(OBJS)
6 $(CC) $(OBJS) -o teste
7 clean:
8 rm -f *.o core
9 clobber: clean
10 rm -f teste
```

Inicialmente, são definidas 2 macros: **CC** e **OBJS**. Nas linha 3 e 4 é definida uma regra de inferência para gerar os arquivos com sufixo **.o** a partir dos arquivos com sufixo **.c**. Nas linhas de 5 a 10 são definidos 3 alvos: **all**, **clean** e **clobber**. O primeiro alvo (**all**) é o alvo executado caso o usuário digite apenas **make**. Este alvo tem como pré-requisito a lista dos arquivos com extensão**.o**. Para gerar estes arquivos o **make** fará uso da regra de inferência definida nas linhas 3 e 4. Por exemplo, na geração do arquivo **teste.o** o seguinte comando é executado

```
gcc -c -o teste.o teste.c
```

Após a geração dos arquivos **teste.o**, **prog1.o** e **prog2.o**, o **make** gera o executável **teste** com o seguinte comando (linha 6)

gcc teste.o prog1.o prog2.o -o teste

O alvo **clean** (linhas 7 e 8) é utilizado para apagar os arquivos com extensão **.o** e o arquivo **core** (gerado pelo sistema em caso de erro de execução). O alvo **clobber** (linhas 9 e 10) é utilizado também para apagar estes arquivos além do arquivo executável **teste** (note que o alvo **clobber** tem o alvo **clean** como pré-requisito).

MAKEDEV

MAKEDEV [opções] dispositivo

São algumas das opções deste comando

-n : não executa ações, apenas lista as ações que seriam executadas.

-d : remove o dispositivo.

-V : informa a versão do aplicativo **MAKEDEV**.

Observações

O script **MAKEDEV** se encontra no próprio diretório **/dev**. Para executá-lo, portanto, deve-se digitar **./MAKEDEV** dentro do diretório **/dev** (use o comando **cd** para mudar de diretório), caso este diretório não tenha sido definido na variável de ambiente **PATH**.

Por exemplo,

./MAKEDEV ttyS4

cria o dispositivo serial **ttyS4** e o seu correspondente dispositivo de discagem **cua4**.

makewhatis

Este comando cria a base de dados **whatis** com informações sobre os comandos do sistema. Apenas o **root** pode executar o comando **makewhatis**.

Observações

A base de dados **whatis** é acessada pelos comandos

- **apropos**
- **man -k**
- **whatis**

man

man comando

Por exemplo,

man ls

apresenta a página do manual com as informações sobre o comando **ls**.

Navegação nas páginas do manual on-line

- para sair do manual on-line digite **q** (de **quit** em inglês que significa sair).
- use as teclas **Page-Down**, **Ctrl+F** ou **Space** para avançar nas páginas do manual.
- use as teclas **Page-Up** ou **Ctrl+B** para voltar páginas no manual.
- use **Enter** para avançar apenas uma linha por vez.
- digite **h** para ver a lista das teclas disponíveis para navegar no manual on-line;

Observações

Para saber quais os comandos que estão relacionados a um determinado assunto, use o parâmetro **-k**. Por exemplo, se você quiser ver quais os comandos do Linux estão relacionados a memória, basta digitar

man -k memory

Convém aqui fazermos três observações sobre o comando acima. Primeiro, o comando **man** só funcionará com este parâmetro depois que o banco de dados **whatis** for criado. O administrador do sistema (**root**) pode criar este banco de dados através do comando **makewhatis**. Segundo, o comando **man** com parâmetro **-k** é equivalente ao comando **apropos**. Terceiro, ao lado de cada comando listado existe um número que indica o nível de execução do comando. Abaixo temos a lista padrão das seções do manual on-line onde cada seção corresponde a um nível de execução.

(1) comandos de usuário
(2) chamadas do sistema
(3) chamadas de bibliotecas
(4) dispositivos
(5) formatos de arquivo
(6) jogos
(7) diversos
(8) comandos de administração do sistema
(9) chamadas internas do kernel
(l) comandos SQL
(n) comandos Tcl/Tk

A lista dos diretórios que possuem as páginas do manual e a ordem em que as páginas devem ser recuperadas são definidas no arquivo **/etc/man.config**. Portanto, se comandos de níveis diferentes possuem o mesmo nome, apenas a primeira página encontrada será mostrada. Para obter a página do comando de um determinado nível, digite

man n comando

onde **n** corresponde ao nível de execução do **comando**. Por exemplo, digite

man swapon

e depois

man 2 swapon

A primeira página no exemplo acima tem nível **8** e explica o comando **swapon** usado para controlar a área de troca, enquanto a segunda página explica a chamada do sistema **swapon.**

Comentários sobre o comando

Para copiar o conteúdo do **man** para um arquivo, basta digitar

man comando > arquivo

Por exemplo,

man ls > teste

Para obter mais informações sobre o comando **man** digite

man man

e para ler mais sobre a formatação das páginas do manual on-line, digite

man 7 man

mc

mc [opções]

Definição

O **mc** é um browser de diretórios e gerenciador de arquivos de sistemas operacionais da família Unix.

Para executar o aplicativo, basta digitar

mc

São algumas das opções deste comando

-a : desabilita o uso de caracteres gráficos.

-c : usa cores.

-d : desabilita o mouse.

São exemplos de teclas que você pode usar no aplicativo

F1 : ajuda.

F2 : exibe menu do usuário.

F3 : visualiza o conteúdo do arquivo.

F4 : edita arquivo.

F5 : copia arquivos.

F6 : move arquivos.

F7 : cria diretório.

F8 : remove arquivos.

F9 : move cursor para o menu superior do **mc**.

F10 : encerra **mc**.

Observações

O nome **mc** é acrônimo para **Midnight Commander**.

mesg

mesg [y/n]

onde

- **y** : habilita o recebimento de mensagens.
- **n** : desabilita o recebimento de mensagens.

Observações

O uso do comando **mesg**, sem parâmetros, informa qual a situação do usuário.

mingetty

mingetty [--noclear] [--long-hostname] tty

São opções deste comando

--noclear : não limpa a tela antes de solicitar o nome de acesso do usuário (o padrão é limpar a tela).

--long-hostname : por padrão o nome da máquina é somente apresentada até o primeiro ponto. Com a utilização desta opção, o nome completo será mostrado.

Caracteres especiais

O **mingetty** reconhece as seguintes sequências de caracteres que podem ser utilizadas no arquivo **/etc/issue** e no arquivo **/etc/issue.net**:

- **\d** : exibe a data atual.
- **\l** : exibe o nome do terminal atual.
- **\m** : exibe a arquitetura da máquina.
- **\n** : exibe o nome de máquina.
- **\o** : exibe o nome do domínio.
- **\r** : exibe a versão do sistema operacional.
- **\t** : exibe a hora local.
- **\s** : exibe o nome do sistema operacional.
- **\u** : exibe o número de usuários que esteja utilizando o sistema no momento.
- **\v** : exibe a data de compilação do sistema operacional.

Exemplo

Suponha que o conteúdo do arquivo **/etc/issue** seja o seguinte:

Data: \d
Terminal: \l
Sistema Operacional: Linux \r

Então, o sistema apresentaria, no modo texto, um login semelhante ao mostrado abaixo:

Data: Tue Set 11 2001
Terminal: tty1
Sistema Operacional: Linux 2.2.17-14cl
localhost login:

Observações

Na realidade, o **mingetty** é um **getty** de funcionalidade reduzida.

mkbootdisk

mkbootdisk [opções] kernel

onde

- **kernel** é a versão do **kernel** a ser incluída no disco.

São algumas das opções deste comando

--device dispositivo : nome do dispositivo onde a imagem do sistema será criada (o padrão é **/dev/fd0**).

Comentários sobre as opções do comando

Por exemplo, para criar um disco de emergência com a versão 2.2.17-14cl do kernel em um disquete, basta digitar

mkbootdisk --device /dev/fd0 2.2.17-14cl

Note que no exemplo acima, não é necessário informar o nome do dispositivo pois é usado o dispositivo padrão.

Observações

Para ver qual a versão do kernel que você está usando, você pode:

- usar o comando **uname -r**;
- verificar o arquivo **/etc/lilo.conf**.

mkdir

mkdir diretório

Comentários sobre o comando

Por exemplo, o comando

mkdir testes

cria o diretório **testes** dentro do diretório atual.

Para criar uma árvore de diretórios, use a opção **-p** com o comando **mkdir**. Por exemplo, o comando

mkdir -p testes/linux

cria o diretório **testes** e dentro deste diretório o subdiretório **linux**.

Observações

É importante observar que para criar um novo diretório é preciso ter permissão de gravação no diretório pai.

Por padrão, as **permissões** associadas a um diretório são **777** (leitura, gravação e acesso para todos os usuários do sistema) menos as permissões definidas no **umask**.

mke2fs

mke2fs [opções] dispositivo

onde **dispositivo** é o arquivo especial correspondente ao dispositivo.

São algumas das opções deste comando

-b num : especifica num bytes por bloco.

-c : verifica e formata o disco usando um método mais rápido.

-m num : reserva num% de espaço no disco para o superusuário.

Comentários sobre as opções do comando

Por exemplo, o comando

mke2fs /dev/hda1

formata o dispositivo **/dev/hda1** para suportar um sistema de arquivos do tipo **ext2**.

Observações

É possível usar o comando **mke2fs** para formatar e configurar disquetes para o sistema de arquivos **ext2**. Neste caso, temos uma formatação de alto nível.

mke2fs /dev/fd0

Use o comando **fdformat** para uma formatação de baixo nível e depois use o comando **mkfs** para configurar o disquete para o sistema de arquivos **ext2**.

mkfs

mkfs [opções] dispositivo [blocos]

onde

- **dispositivo** : é o arquivo especial correspondente ao dispositivo.
- **blocos** : é quantidade de blocos a ser utilizada pelo sistema de arquivos.

São algumas das opções deste comando

-t tipo : especifica o tipo de sistema de arquivos a ser criado. O padrão é **ext2**.

-c : checa o dispositivo a procura de blocos defeituosos durante a criação do sistema de arquivos.

Comentários sobre as opções do comando

Por exemplo, o comando

mkfs -t ext2 /dev/fd0 1440

configura o disquete para o sistema de arquivos **ext2**.

Observações

Use o comando **fdformat** para uma formatação de baixo nível, antes de usar o comando **mkfs**.

mkswap

mkswap [-c] dispositivo [bloco]

Opções do comando

- a opção **-c** pede que se verifique a existência de blocos ruins antes de criar o sistema de arquivos.
- **dispositivo** tem normalmente o seguinte formato

  /dev/hda[1-8]
  /dev/hdb[1-8]
  /dev/sda[1-8]
  /dev/sdb[1-8]

- **bloco** corresponde ao número de blocos que a área de **swap** deve possuir sendo que cada bloco tem 1 Kb. O valor mínimo (MINCOUNT) e o valor máximo (MAXCOUNT) possíveis para este parâmetro é calculado da seguinte forma:

  MINCOUNT = 10 * tamanho_página / 1024

  MAXCOUNT = (tamanho_página - 10) * 8 * tamanho_página / 1024

  Caso este parâmetro não seja fornecido, ele é determinado automaticamente pelo **mkswap**.

Comentários sobre as opções do comando

Antes de criarmos uma área de **swap**, precisamos verificar o tamanho de página da máquina. Podemos obter esta informção no arquivo **/proc/cpuinfo**. Por exemplo, suponha que temos uma página de 4 KB. Portanto, temos

MINCOUNT = 10 * 4096 / 1024 = 40

MAXCOUNT = (4096 - 10) * 8 * 4096 / 1024 = 130752

Podemos então escolher para o número de blocos um valor entre 40 e 130752. Suponha que o valor escolhido para **blocos** é 8000 (o que corresponde a 8 MB) e que o nome do arquivo de troca é **teste**. Para configurar o arquivo de troca devemos então executar a seguinte sequência de comandos:

```
dd if=/dev/zero of=teste bs=1024 count=8000
mkswap teste 8000
sync
swapon teste
```

O comando **dd** cria o arquivo de **swap**. O comando **mkswap** configura o arquivo para **swap**. O comando **sync** atualiza o arquivo criado. O comando **swapon** habilita o arquivo criado para ser área de **swap**.

modprobe

modprobe módulo

Comentários sobre o comando

O **modprobe** lê o arquivo de dependências de módulos gerado pelo comando **depmod**. Por isso, deve-se executar o comando

depmod -a

para produzir um novo arquivo contendo as dependências de módulo. Após executá-lo, pode-se usar o comando **modprobe** para instalar qualquer módulo e ter os outros módulos dos quais ele depende automaticamente instalados.

Observações

Para acrescentar um módulo automaticamente a cada inicialização do sistema, inclua o comando correspondente no arquivo **/etc/rc.d/rc.local** ou no arquivo **/etc/rc.sysinit**.

módulos

Definição

Os módulos permitem adicionar **drivers** de dispositivos ao sistema em operação em tempo real. Isto significa que o sistema pode inicializar um **kernel** qualquer do Linux e então adicionar os**drivers** necessários para o uso do hardware que faz parte do sistema. O hardware é imediatamente disponibilizado sem a necessidade de reinicialização do sistema.

Os módulos de código carregáveis são armazenados em **/lib/modules/XXX/misc**, onde **XXX** é a versão do **kernel** do sistema (por exemplo, 2.2.12-5cl).

Comandos relacionados

Para verificar quais os **módulos** que estão atualmente carregados na memória, use o comando **lsmod** ou verifique o conteúdo do arquivo **/proc/modules**.

Para verificar as dependências dos **módulos** use o comando **depmod**.

Para fazer a instalação de **módulos**, em tempo de execução, use o comando **insmod** ou o comando **modprobe**.

Para remover **módulos**, em tempo de execução, use o comando **rmmod**.

Observações

O arquivo de configuração dos **módulos** é **/etc/conf.modules**.

more

more [opções] arquivo

São algumas das opções deste comando

-d : mostra o menu de navegação do aplicativo.

+num : mostra o arquivo a partir da linha **num**.

Navegação

- para sair do aplicativo digite **q** (de **quit** em inglês que significa sair).
- use as teclas **Page-Down**, **Ctrl+F** ou **Space** para avançar
- use as teclas **Page-Up** ou **Ctrl+B** para voltar.
- use **Enter** para avançar apenas uma linha por vez.

Comentários sobre o comando

Para redirecionar a saída de um comando para o **more**, digite

comando | more

Por exemplo,

ls | more

mount

mount [opções] dispositivo ponto_de_montagem

Em um sistema Linux, os arquivos estão organizados em uma grande árvore, onde o diretório raiz (pai de todos os outros diretórios) é representado como **/**. Estes arquivos podem estar distribuídos por diversos dispositivos como disquetes, cd-rooms, HDs, etc.

O comando **mount** é usado para incluir o **sistema de arquivos**, de um dispositivo qualquer, à grande árvore de arquivos.

São algumas das opções deste comando

-a : monta todos os sistemas de arquivos especificados no arquivo **/etc/fstab**.

-r : monta a partição somente para leitura.

-t tipo : especifica o tipo de sistema de arquivo que será montado. São exemplos de tipos de sistemas:

- **ext2** - para partições Linux.
- **vfat** - para partições Windows (permite nomes de arquivos com até 32 caracteres).
- **msdos** - para partições DOS.
- **iso9660** - para unidades de CD-ROM.
- **umsdos** - para partição DOS com alguns recursos de Linux.

-v : lista o sistema de arquivo de cada dispositivo montado.

-w : monta a partição para leitura/gravação (é o padrão).

Comentários sobre as opções do comando

O exemplo abaixo monta uma partição DOS localizada em **/dev/hda1** no diretório **/mnt/dos**

mount -r -t msdos /dev/hda1 /mnt/dos

Se o dispositvo já foi descrito no **/etc/fstab**, mas ainda não foi montado, então basta especificar o nome do dispositivo ou o ponto de montagem ao digitar o comando **mount**. Por exemplo, suponha que o sistema tenha a seguinte linha no **/etc/fstab**

/dev/fd0 /mnt/floppy vfat exec,dev,suid,rw,noauto 0 0

O comando acima define que o disquete (**/dev/fd0**) deve ser montado abaixo do diretório **/mnt/floppy**, sendo que o sistema de arquivos usado é **vfat** (Windows/DOS). A opção **noauto** define que o dispositivo não deve ser montado automaticamente durante a inicialização do sistema. Para montar o disquete você deve digitar

mount /mnt/floppy/

ou

mount /dev/fd0

Observações

O arquivo **/etc/mtab** possui a lista de todos os sistemas de arquivos atualmente montados no Linux.

O comando **mount**, sem parâmetros, faz o sistema listar o conteúdo do arquivo **/etc/mtab**.

Veja o comando **umount** para saber como desmontar um sistema de arquivos.

mouseconfig

Comentários sobre o comando

No Linux o link simbólico **/dev/mouse** aponta para o dispositivo utilizado pelo systema para configurar o mouse. Para ver qual a configuração atual no seu sistema, basta digitar o comando

ls -l /dev/mouse

Por exemplo,

lrwxrwxrwx 1 root root   5 Jul 17 18:20 /dev/mouse -> ttyS0

informa a existência de mouse serial na primeira porta do equipamento. Para fazer com que o sistema detecte e configure automaticamente o mouse instalado, basta digitar

mouseconfig

Para evitar que o sistema configure o mouse use as opções **--noprobe** e **--expert** com o comando **mouseconfig**, ou seja,

mouseconfig --noprobe --expert

É possível também definir o tipo do mouse ao chamar o aplicativo **mouseconfig**. Suponha, por exemplo, que você queira criar um mouse **logitech**. Você pode então usar o comando da seguinte forma

mouseconfig logitech

Observações

Para ver a lista dos mouses suportados pelo sistema, digite

mouseconfig --help

mpage

mpage [opções] arquivo

São algumas das opções deste comando

-1 : imprime uma página por papel.

-2 : imprime duas páginas por papel.

-4 : imprime quatro páginas por papel.

-8 : imprime oito páginas por papel.

-b : define o tipo do papel (A4, legal, letter, etc).

-c : junta vários arquivos em uma única impressão.

-H : cria cabeçalho para cada página.

-l : usa formato **landscape** (o padrão é **portrait**).

Comentários sobre as opções do comando

Por exemplo, para criar o arquivo **teste.ps** com quatro páginas por folha e formato A4 a partir do arquivo **teste.txt**, basta digitar

mpage -4 -b a4 teste.txt > teste.ps

Observações

Este comando ler um arquivo **ASCII** ou **PostScript** e cria um arquivo **PostScript** para impressão.

mt

mt [opções] [comando]

São algumas das opções deste comando

-f fita : especifica o dispositivo de fita.

-h : exibe lista de comandos disponíveis.

Observações

São alguns comandos que podem ser usados em conjunto com o **mt**:

- **rewind** : rebobinar a fita.
- **erase** : apagar o conteúdo da fita.

mtools

Definição

O **mtools** é uma coleção de ferramentas que permite, a partir do Linux, manipular arquivos em sistemas MS-DOS. Abaixo discutimos algumas destas ferramentas.

mcd

Este aplicativo muda o diretório corrente de trabalho em um sistema MS-DOS.

Por exemplo, o comando

mcd a:/testes

faz com que o sistema passe a trabalhar com o diretório **testes** do disquete MS-DOS. Isto significa que qualquer comando seguinte do **mtools** usará o diretório **testes** como padrão.

mcopy

Este aplicativo copia arquivos ou diretórios entre os sistemas Linux e MS-DOS.

mcopy [opções] arquivo_origem arquivo_destino

São algumas das opções disponíveis:

- **/** : copia também os subdiretórios e os seus conteúdos.
- **-m** : preserva a data de modificação do arquivo.
- **-n** : não solicita confirmação ao regravar arquivos do sistema Linux.
- **-o** : não solicita confirmação ao regravar arquivos do sistema MS-DOS.
- **-p** : preserva os atributos dos arquivos copiados.
- **-v** : ecoa na tela os nomes dos arquivos copiados.

Por exemplo, o comando

mcopy -o * a:/.

copia todos os arquivos do diretório corrente para o disquete que possui formato MS-DOS. A opção **-o** indica que não há necessidade de pedir confirmação em caso de regravação de algum arquivo que já existe no disquete.

mdel

Este aplicativo apaga arquivos MS-DOS.

mdel [-v] arquivo1 [arquivo2, ...]

A opção **-v** faz com que o sistema exiba os nomes dos arquivos apagados.

Por exemplo, o comando

mdel a:/teste.txt

apaga o arquivo **teste.txt** no disquete que possui formato MS-DOS.

mdir

Este aplicativo exibe o conteúdo de diretórios de arquivos MS-DOS.

mdir [opções] diretório

São algumas das opções disponíveis:

- **/** : exibe também os subdiretórios e os seus conteúdos.
- **-a** : exibe também os arquivos ocultos.
- **-f** : não exibe espaço livre do sistema MS-DOS.
- **-w** : exibe apenas os nomes dos arquivos (sem tamanhos e sem datas de criação).

Por exemplo, o comando

mdir a:

exibe o conteúdo de um disquete que possui formato MS-DOS.

mformat

Este aplicativo cria, no disquete, um sistema de arquivos no formato MS-DOS. Esta é uma formatação rápida pois apenas cria o setor de inicialização, a FAT e o diretório raiz.

mformat [opções] unidade:

São algumas das opções disponíveis:

- **-h cabeças** : o número de cabeças (lados).
- **-s setores** : o número de setores por trilhas.
- **-t cilindros** : o número de cilindros.
- **-M tamanho** : o tamanho de cada setor.

Por exemplo, o comando

mformat a:

formata o disquete do drive **a:**.

mmd

Este aplicativo cria um subdiretório MS-DOS.

mmd diretório

Por exemplo, o comando

mmd testes

cria o diretório **testes** no disquete MS-DOS.

mrd

Este aplicativo remove um subdiretório MS-DOS.

mrd diretório

Por exemplo, o comando

mrd testes

apaga o diretório **testes** no disquete MS-DOS.

mren

Este aplicativo renomeia arquivos e subdiretórios MS-DOS.

mren arquivo_origem arquivo_destino

Por exemplo, o comando

mren teste.txt classe.txt

renomeia o arquivo **teste.txt** para **classe.txt**.

mv

mv [opções] origem destino

onde

- **origem** corresponde ao arquivo a ser movido.
- **destino** corresponde ao nome final do arquivo.

São algumas das opções deste comando

-b : gera cópia de segurança se o arquivo de destino já existir.

-f : move o arquivo sem pedir confirmação (mesmo que já exista um arquivo no destino com o mesmo nome).

-i : move o arquivo, mas pede confirmação caso já exista um arquivo no destino com o mesmo nome.

-v : exibe os nomes dos arquivos antes de movê-los.

Comentários sobre as opções do comando

Por exemplo, para renomear o arquivo **teste.txt** para **teste2.txt**, basta digitar

mv teste.txt teste2.txt

Pode-se usar os caracteres **?**, ***** e **[ ]** como **curingas**. O primeiro substitui apenas um único caractere, o segundo substitui um número qualquer de caracteres, enquanto o terceiro substitui um único caractere dentro de um certo limite de valor. Por exemplo, suponha que o diretório atual tenha os seguintes arquivos: **teste1.txt**, **teste2.txt**, **teste3.txt** e **teste10.txt**. O comando

mv teste*.txt /tmp/. ou mv teste* /tmp/.

move os quatro arquivos para o diretório **/tmp**, enquanto o comando

mv teste?.txt /tmp/.

move apenas os arquivos **teste1.txt**, **teste2.txt** e **teste3.txt** para o diretório **/tmp**. Para mover apenas os dois primeiros arquivos, basta digitar

mv teste[1-2].txt /tmp/.

newgrp

newgrp [grupo]

onde **grupo** corresponde ao **nome** do grupo desejado (não pode ser o **GID** do grupo).

Observações

Para retornar ao grupo anterior, basta digitar **exit**.

Se nenhum nome de grupo é fornecido com o comando, assume-se que o novo grupo corresponde ao grupo padrão do usuário (para saber qual é o grupo padrão de um usuário veja o arquivo**/etc/passwd**).

nice

nice [opções] [comando...]

A prioridade de execução de um **processo** pode variar de **-20** (maior prioridade) a **19** (menor prioridade). Por padrão, a prioridade dos processos é zero.

São algumas das opções deste comando

-n valor ou **-valor** : adiciona o valor especificado à prioridade padrão de execução no sistema.

Comentários sobre o comando

Por exemplo, o comando

nice -5 find / -name gcc

define que o comando **find** deve ser executado com prioridade 5, enquanto

nice --5 find / -name gcc

define uma prioridade -5 para o mesmo comando.

Observações

Quando o valor de ajuste da prioridade não é definido junto com o comando **nice**, o sistema assume o valor de ajuste igual a 10.

Apenas o administrador do sistema (**root**) pode definir prioridades negativas.

Se nenhum argumento é fornecido com o comando **nice**, o sistema exibe a prioridade padrão atualmente em uso.

Para alterar a prioridade de um processo que está em execução, use o comando **renice**.

Para ver a prioridade de execução dos processos, use o aplicativo **top**.

nl

nl [opções] [arquivo]

São algumas das opções deste comando

-i num : a numeração das linhas é feita com incremento de **num**.

-v num : a numeração das linhas começa com **num**.

Comentários sobre as opções do comando

Por exemplo, o comando

nl -i 10 -v 100 teste

exibe o conteúdo do arquivo **teste** numerando as linhas a partir do valor 100 com incremento de 10.

Observações

Se o nome do arquivo não é fornecido com o comando, a entrada padrão é lida.

O comando **nl arquivo** corresponde ao comando **cat -n arquivo**.

nohup

nohup comando

Descrição

O **nohup** ignora os sinais de interrupção de conexão durante a execução do comando especificado. Assim, é possível o comando continuar a executar mesmo depois que o usuário se desconectar do sistema.

Se a saída padrão é uma **tty**, esta saída e o erro padrão são redirecionados para o arquivo **nohup.out** (primeira opção) ou para o arquivo **$HOME/nohup.out** (segunda opção). Caso nenhum destes dois arquivos possam ser criados (ou alterados se já existem), o comando não é executado.

O **nohup** não coloca o comando que ele executa em **background**. Isto deve ser feito explicitamente pelo usuário.

Comentários sobre o comando

Por exemplo,

nohup find / -name gcc &

executa o comando **find** em background (**&**) e sem permitir interrupções de conexão (**nohup**).

passwd

passwd [-u] [usuário]

Opções do comando

A opção **-u** é usada para indicar que a atualização só é efetuada após a data de expiração da senha atual.

Observações

O usuário pode alterar a própria senha digitando apenas **passwd**.

O superusuário (**root**) pode alterar a senha de outro usuário digitando o comando **passwd** junto com o nome do usuário.

permissão de acesso

Cada arquivo/diretório do sistema está associado a um usuário (dono) e a um grupo. O dono de um arquivo (ou de um diretório) pode definir quem tem acesso ao arquivo e qual tipo de acesso é permitido (leitura, gravação e/ou execução). Isto é chamado de **permissão de acesso**.

Níveis de acesso

Para cada arquivo e diretório no Linux são definidos três níveis de acesso:

a. **dono** - é a pessoa que criou o arquivo ou o diretório. Somente o dono e/ou o administrador do sistema (**root**) pode alterar as permissões de acesso.
b. **grupo** - conjunto de usuários que são membros do mesmo grupo a qual pertence o arquivo.
c. **outros** - o resto dos usuários do sistema.

Tipos de permissão

Para cada nível de acesso temos três tipos de permissão:

a. **leitura (r)** - permite ler o conteúdo de um arquivo/diretório.
b. **escrita (w)** - permite alterar um arquivo/diretório.
c. **execução (x)** - permite executar um arquivo ou acessar um diretório.

Exemplo

Para ver as permissões dos arquivos e subdiretórios do diretório corrente, basta digitar o comando **ls -l**. Abaixo mostramos um exemplo.

Permissões	Ligações diretas	dono	grupo	tamanho	Última Atualização	Arq/Dir
drwxrwxr-x	4	aluno	basico	1024	May 10 22:07	.
drwx------	8	aluno	basico	1024	May 10 21:50	..
-rw-rw----	1	aluno	basico	34585	May 10 22:07	linux.txt
-rw-rw-r--	1	aluno	basico	15258	Apr 23 12:26	perl.txt
drwxrwxr-x	2	aluno	basico	1024	Apr 28 10:56	pesquisa_alunos
drwxrwxr-x	2	aluno	basico	1024	Apr 24 11:38	testes_perl
-rwxrwsr-x	1	aluno	basico	14785	Nov 15 10:50	teste

A primeira coluna do exemplo acima mostra as permissões de acesso dos subdiretórios e arquivos, onde

- O primeiro caractere diz qual é o tipo do objeto:
  - **-** para arquivo comum;
  - **b** para dispositivos de bloco (oferecem grandes quantidades de dados de cada vez).
  - **c** para dispositivo de caracteres (oferecem dados de um caractere de cada vez);
  - **d** para diretório;
  - **l** para link simbólico;
  - **p** para FIFO ou Named Pipe;
  - **s** para socket mapeado em arquivo;
- Os três caracteres seguintes mostram as permissões de acesso do dono do arquivo.

- O quinto, o sexto e o sétimo caracteres dizem quais as permissões de acesso para os usuários que são membros do grupo ao qual pertence o arquivo/diretório.
- Os três últimos caracteres especificam as permissões para os outros usuários do sistema.

Por exemplo, o arquivo **linux.txt** acima tem permissão de leitura e escrita apenas para o dono e para os membros do grupo do arquivo.

Permissões especiais

Existem três permissões especiais de arquivo:

- **SGID** - com esta permissão, o usuário executa o arquivo ou acessa o diretório como se fosse membro do grupo ao qual pertence o arquivo/diretório.
- **SUID** - com esta permissão, o usuário executa o arquivo ou acessa o diretório como se fosse o dono.
- **sticky bit** - o usuário pode criar/alterar/deletar (apenas) os seus próprios arquivos no diretório que possui esta permissão, mesmo que não seja o dono do diretório e nem membro do grupo ao qual o diretório pertence. Quando o **sticky bit** é usado em um arquivo, significa que este arquivo é compartilhado por vários usuários.

Note que estas permissões se referem a execução de um arquivo ou a autorização de acesso a um diretório. Portanto, elas estão relacionadas a permissão **x**. Os seguintes valores são usados para identificar o uso de permissão especial em um arquivo/diretório:

Permissão	Nível	Significado
S	dono	SUID
s	dono	SUID + permissão de execução para o dono
S	grupo	SGID
s	grupo	SGID + permissão de execução para o grupo
T	outros	sticky bit
t	outros	sticky bit + permissão de execução para o resto dos usuários do sistema

No exemplo mostrado acima, temos a seguinte linha:

-rwxrwsr-x 1 aluno basico 14785 Nov 15 10:50 teste

Note que o arquivo **teste** possui permissão **s** para o grupo. Isto significa que este é um arquivo **SGID** e que pode ser executado pelos membros do grupo a qual pertence o arquivo (neste exemplo, o dono e o resto dos usuários do sistema também possuem permissão de execução).

Comandos relacionados

- **chgrp** - para mudar o grupo de um arquivo ou de um diretório.
- **chmod** - para mudar a permissão de acesso a um arquivo ou a um diretório.
- **chown** - para mudar o dono de um arquivo.

pico

pico [opções] arquivo

São algumas das opções deste aplicativo

-m : habilita o uso do mouse no editor de texto (só funciona se o **pico** for chamado a partir de uma janela do **X**).

+n : inicializa com o cursor na **n**-ésima linha.

-x : desabilita menu de ajuda na parte de baixo da janela do **pico**.

-w : desabilita a quebra automática de linha.

São alguns dos comandos do pico:

CTRL+A : ir para o início da linha atual.

CTRL+B : o cursor retorna um caractere.

CTRL+C : mostrar a posição atual do cursor.

CTRL+D : apagar caractere atual.

CTRL+E : ir para o fim da linha atual.

CTRL+F : o cursor avança um caractere.

CTRL+G : exibir menu de ajuda.

CTRL+J : justificar o parágrafo atual.

CTRL+K : remover a linha atual.

CTRL+N : o cursor desce uma linha.

CTRL+O : salvar arquivo como.

CTRL+P : o cursor sobe uma linha.

CTRL+R : inserir um arquivo.

CTRL+T : fazer verificação ortográfica.

CTRL+U : colar texto ou desfazer a justificação.

CTRL+V : avançar uma página.

CTRL+X : sair do pico.

CTRL+W : procurar texto.

PID

Definição

Cada processo sendo executado no Linux está associado a um número inteiro positivo que é conhecido como **PID** (Process IDentification).

Existem alguns **PID**s que são especiais. Por exemplo,

- o **PID 1** corresponde ao processo **init** que é o primeiro processo a ser inicializado pelo Linux;
- o **PID 2** corresponde ao daemon **kflushd** que esvazia o cache de disco;
- o **PID 3** corresponde ao processo **kswapd** que é utilizado para fazer troca (**swap**) de memória virtual para o arquivo de troca.

Os números de **PID** crescem até um valor alto e então reiniciam. Entretanto, dois processos nunca podem ter o mesmo número de **ID**. É responsabilidade do **kernel** do Linux garantir de que um novo processo obtenha seu próprio número único de **ID**.

Observações

O diretório **/proc** possui a lista dos processos que estão executando no sistema.

pr

pr [opções] arquivo

São algumas das opções deste comando

-d : saída com espaçamento duplo.

-h : define um cabeçalho para as páginas de impressão.

-t : não mostra nenhum tipo de cabeçalho.

printenv

printenv [variáveis]

Exemplo

O comando

printenv USER SHELL

mostra o nome do usuário e o **shell** utilizado pelo usuário.

Observações

O uso do comando **printenv** sozinho, sem o nome de qualquer variável de ambiente, faz com que os valores de todas as variáveis de ambiente sejam exibidos.

processo

Definição

A seguinte definição de **processo** foi apresentada por Tanenbaum em **"Sistemas Operacionais - Projeto e Implementação"**:

A idéia-chave aqui é que um processo é um tipo de atividade. Ele tem um programa, entrada, saída e um estado. Um único processador pode ser compartilhado entre vários processos, com algum algoritmo de agendamento sendo utilizado para determinar quando parar de trabalhar em um processo e servir a um diferente.

Ainda segundo Tanenbaum, são estados de um processo:

- **executando** - o processo está utilizando a CPU;
- **pronto** - o processo está temporariamente parado para permitir que outro processo execute;
- **bloqueado** - o processo é incapaz de executar até que um evento aconteça.

Um processo pode criar outros processos. Dizemos que um processo é pai do processo que ele criou. Além disso, um processo-filho pode também criar novos processos, formando assim uma árvore hierárquica de processos.

No Linux

O **init** é o primeiro processo inicializado no linux e é o pai de todos os outros processos. Se um processo termina e deixa processos-filho ainda executando, o processo **init** assume a paternidade destes processos.

Quando um usuário trabalha no modo monousuário, um único processo **shell** é inicializado. Qualquer comando digitado pelo usuário neste terminal irá gerar um ou mais processos. A árvore hierárquica dos processos tendo o **shell** como raiz é chamada de **sessão**.

Quando um usuário inicializa um gerenciador de janelas (ver **X Windows**), para cada terminal criado é alocado um processo **shell** responsável pelo terminal em questão. Cada terminal corresponde a uma sessão aberta pelo usuário sendo que as sessões são independentes entre si. Isto significa que uma alteração feita em um dos terminais não será reconhecida pelos outros terminais pois cada terminal possui o seu próprio conjunto de **variáveis de ambiente**. Por exemplo, ao entrar com o comando **su** em um dos terminais, apenas o terminal em questão passará a exibir o login do **superuser**.

Background e Foreground

No linux, um processo pode estar em **foreground** ou em **background**, ou seja, em primeiro plano ou em segundo plano. Por exemplo, ao digitar o comando

```
ls -R /etc > teste
```

o sistema criará o arquivo **teste** com o conteúdo de todos os diretórios e arquivos que se encontram abaixo do diretório **/etc**. Durante a execução do comando acima, nenhum outro comando poderá ser digitado pelo usuário no mesmo terminal. Isto significa que o comando está sendo executado em primeiro plano impedindo assim a inicialização de outras atividades no mesmo terminal.

Para o exemplo acima, é possível liberar o **shell** para outras atividades enquanto o arquivo **teste** é criado. Basta que você digite

```
ls -R /etc > teste &
```

O símbolo **&** indica que o comando deve ser executado em **background**, ou seja, em segundo plano.

Se um processo está executando em **foreground** e você deseja colocá-lo em **background**, você deve:

1. interromper a execução do processo com as teclas CTRL-Z;
2. digitar o comando **bg**.

Para trazer um processo do modo **background** para o modo **foreground**, você deve usar o comando **fg**.

Modelo Cliente-Servidor

O Linux implementa muitas das suas funções usando o modelo cliente-servidor. Isto significa que existem processos que são criados especificamente para executar determinadas tarefas. Estas tarefas especiais são oferecidas aos outros processos do sistema na forma de serviços.

O processo responsável pela execução de um determinado serviço no sistema é chamado de **servidor**, enquanto o processo que solicita o serviço ao sistema é chamado de **cliente**.

Normalmente, as aplicações servidoras (**daemons**) são executadas em **background**, enquanto as aplicações clientes são executadas em **foreground**.

A grande vantagem em implementar funções do sistema dessa forma é tornar o **kernel** do Linux menor, pois tudo que o kernel faz neste caso é gerenciar a comunicação entre clientes e servidores. Além disso, o administrador pode escolher os serviços que o sistema tornará disponíveis.

São exemplos de **daemons** no Linux:

- **atd (at daemon)** - servidor que executa serviços agendados pelo comando **at**.
- **crond (cron daemon)** - servidor que executa serviços agendados pelo comando **crontab**.
- **httpd (http daemon)** - servidor de páginas HTML e scripts CGI (é o servidor web Apache).
- **inetd (inet daemon)** - servidor responsável pelos serviços de rede.
- **lpd (printer daemon)** - servidor de impressão de arquivos.
- **syslogd (syslog daemon)** - servidor de logs do sistema (para mais detalhes veja **/etc/syslog.conf**).

Por padrão, podemos usar os seguintes argumentos com os **daemons**:

- **start** : inicializa o daemon.
- **stop** : pára o daemon.
- **start** : verifica a situação do daemon.

Portanto, o comando

/etc/rc.d/init.d/atd status

fornece informação sobre o servidor **at**.

Observações

O diretório **/proc** possui a lista dos processos que estão em execução no sistema. Cada processo é identificado neste diretório por um número. Por exemplo, o processo **init** é o processo 1.

Comandos relacionados

- **bg** - coloca um processo em **background**.
- **fg** - coloca um processo em **foreground**.
- **fuser** - identifica os processos que estão usando um determinado arquivo ou diretório.
- **ipcrm** - remove recurso **ipc** (inter-process communication).
- **ipcs** - fornece informações sobre recursos **ipc** (inter-process communication).
- **jobs** - fornece a lista dos processo executando em **background**.
- **kill** - envia um determinado sinal a um processo em execução.

- **killall** - envia um determinado sinal a um conjunto de processos que usam o mesmo nome.
- **nice** - define a prioridade de execução de um processo.
- **nohup** - executa um comando (processo) imune a interrupções de conexão.
- **ps** - exibe informações sobre os processos em execução.
- **renice** - altera a prioridade de um processo em execução.
- **top** - lista os processos que estão utilizando a CPU.

ps

ps [opções]

São algumas das opções deste comando

-a : mostra os processos de todos os usuários.

-e : mostra as variáveis de ambiente no momento da inicialização do processo.

-f : mostra a árvore de execução de comandos.

-x : mostra os processos que não foram iniciados no console.

-u : fornece o nome do usuário e a hora de início do processo.

Comentários sobre as opções do comando

Por exemplo, o comando

ps -aux

exibe as informações sobre todos os processos que estão sendo executados no sistema.

pwconv

pwconv

Descrição

Este comando migra as senhas criptografadas do **/etc/passwd** para o **/etc/shadow**. Neste caso, a posição da senha em **/etc/passwd** é preenchida com um **"x"**.

Observação

Para eliminar o arquivo **/etc/shadow**, use o comando **pwunconv**.

pwunconv

pwunconv

Descrição

Este comando migra as senhas criptografadas do **/etc/shadow** para o **/etc/passwd**.

Observação

Para gerar novamente o arquivo **/etc/shadow**, use o comando **pwconv**.

quota

quota [opções]

São algumas das opções deste comando

-u [usuário] : informações sobre a cota do usuário.

-g [grupo] : informações sobre a cota do grupo do qual o usuário é membro.

Comentários sobre as opções do comando

Por exemplo, o comando

quota -u aluno

exibe informações sobre a cota do usuário aluno.

Observações

É preciso ser root para obter informações sobre o sistema de cotas de outros usuários.

Veja **cotas de disco** para mais detalhes.

quotaoff

quotaoff [opções] [sistema de arquivos]

São algumas das opções deste comando

-a : desabilita as cotas de todos os sistemas de arquivos definidos no **/etc/fstab**.

-u : desabilita as cotas de usuários. É o padrão.

-g : desabilita as cotas dos grupos.

Comentários sobre as opções do comando

Por exemplo, o comando

quotaoff /dev/hda2

desabilita o sistema de cotas do **/dev/hda2**.

Observações

É preciso ser root para executar este comando.

Veja **cotas de disco** para mais detalhes.

quotaon

quotaon [opções] [sistema de arquivos]

São algumas das opções deste comando

-a : habilita as cotas de todos os sistemas de arquivos definidos no **/etc/fstab**.

-u : habilita as cotas de usuários. É o padrão.

-g : habilita as cotas dos grupos.

Comentários sobre as opções do comando

Por exemplo, o comando

quotaon /dev/hda2

habilita o sistema de cotas do **/dev/hda2**, conforme definido em **/etc/fstab**.

Observações

É preciso ser root para executar este comando.

Veja **cotas de disco** para mais detalhes.

RCS

RCS significa **Revision Control System** e é um sistema para controlar acesso a arquivos compartilhados por vários usuários.

Comandos do RCS

Para que um programa passe a ser gerenciado pelo **RCS**, é preciso que ele seja primeiro verificado. Para isto basta digitar

ci programa

onde **ci** significa **ckeck in RCS revisions**. Em resposta ao comando acima, o **RCS** solicita a descrição do programa especificado. Para encerrar a descrição, basta utilizar um ponto em uma linha isolada ou utilizar **Ctrl+D**. O **RCS** então apaga o arquivo original e cria um novo arquivo com mesmo nome e extensão "**,v**". Este arquivo é somente para leitura e é maior que o arquivo original porque o sistema **RCS** acrescenta algumas informações adicionais a esse arquivo tais como versão, descrição recentemente digitada, histórico de alteração, etc. Para que uma cópia do arquivo (apenas para leitura) seja deixado pelo **RCS** no diretório, basta usar a opção **-u**, ou seja,

ci -u programa

É também possível definir o número da versão que o **RCS** armazenará o arquivo. Por exemplo,

ci -r2.0 programa

define que o número da nova versão é 2.0. O **RCS** não aceita um número de versão menor que o número da última versão gravada.

Para obter uma cópia de trabalho de um arquivo gerenciado pelo **RCS**, deve-se digitar

co programa

onde **co** significa **ckeck out RCS revisions**. Esta cópia entretando é apenas para leitura. Para obter uma cópia do arquivo para alteração e bloqueiar o acesso a esse arquivo por parte dos outros usuários, basta digitar

co -l programa

Pode-se também recuperar uma determinada versão do programa gravado pelo **RCS**. Por exemplo,

co -r2.0 programa

recupera a versão 2.0 do programa especificado.

Para desbloquear um arquivo sem fazer nenhuma alteração, deve-se digitar

rcs -u arquivo

Para verificar quais as alterações existentes entre a última versão gravada do programa pelo **RCS** e a versão atual no seu diretório, digite

rcsdiff programa

Por exemplo, a seguinte saída

```
11d10
< ******
15a15,16
> #include "def2.h"
> #include "def3.h"
```

informa que a linha 11 do arquivo original foi removida e que duas novas linhas, 15 e 16, foram incluidas no arquivo. É também possível verificar as diferenças existentes entre duas versões de um arquivo **RCS**. Por exemplo,

```
rcsdiff -r2.0 -r1.0 teste.c
```

lista as diferenças existentes entre a versão 2.0 e a versão 1.0 do arquivo **teste.c** gerenciado pelo **RCS**.

Para obter o histórico das alterações de um arquivo **RCS**, basta digitar

```
rlog programa
```

Para apagar todos os arquivos **RCS**, em um diretório, que não estão bloqueados para alteração (apenas leitura), pode-se usar o seguinte comando:

```
rcsclean
```

Uma forma de organizar os arquivos gerenciados pelo **RCS** é criar um subdiretório (dentro do diretório dos arquivos **RCS**) com o nome **RCS** e mover todos os arquivos **RCS** para este novo subdiretório. Ao usar o comando **ci** ou **co**, o **RCS** passará a usar este subdiretório como padrão para armazenamento e recuperação de arquivos **RCS**.

Palavras-chave do RCS

É possível documentar o código-fonte usando as informações armazenadas pelo **RCS**. São exemplos de palavras-chave conhecidas pelo **RCS**:

$Author$: identificação do usuário da última revisão.

$Date$: data e hora da última revisão.

Id : nome do arquivo, versão, data e hora da atualização, autor e estado do programa no gerenciamento do **RCS**.

Log : mensagem de log gerada durante a verificação.

$Revision$: número da versão do arquivo.

$Source$: caminho completo do arquivo **RCS**.

Quando alguma destas palavras é usada no código-fonte, o **RCS** a substitui automaticamente durante a verificação do arquivo (comando **ci**). Portanto, quando estas palavras são usadas dentro de códigos de programas compiláveis, é importante que sejam definidas dentro de comentários. Por exemplo, podemos definir em um programa C:

```
/*
$Id$
$Log
*/
```

Mas também é possível utilizar estas informações dentro de um programa C. Para isto basta definir uma string. Por exemplo,

static const char rcsid[] = "Id";

Para verificar quais as palavras-chave são usadas em um arquivo **RCS**, podemos digitar

ident arquivo

Observações

O **RCS** gerencia os programas apenas localmente, ou seja, dentro de um determinado diretório.
O **CVS** (**Concurrent Versions System**) é uma ferramenta construída sobre a estrutura do**RCS** e que opera sobre uma rede.

Para ver a documentação online do **RCS**, digite

man rcsintro

reboot

reboot

Observações

Este comando corresponde aos comandos **init 6** e **shutdown -r now**.

redirecionadores de E/S

São exemplos de operadores de E/S

Operador	Significado
> **ou** 1>	redireciona para saída
>>	redireciona para fim de arquivo
<	redireciona para a entrada
<<	redireciona para a entrada e mantém a entrada aberta até que seja digitado algum caractere de EOF (fim de arquivo) como, por exemplo, CTRL+D
>& **ou** 2>	redireciona a saída de erros
|	redireciona a saída de um comando para a entrada de um outro comando
tee	redireciona o resultado para a saída padrão e para um arquivo, deve ser usado em conjunto com o "|"

Exemplos

a. O comando **cat** abaixo copia o conteúdo dos arquivos **teste1.txt** e **teste2.txt** para o arquivo **teste**. Caso o arquivo **teste** não exista, o arquivo é criado. Caso o arquivo **teste** exista, ele é sobreposto.

```
cat teste1.txt teste2.txt > teste
```

b. O comando **cat** abaixo adiciona o conteúdo do arquivo **teste3.txt** no final do arquivo **teste**. Caso o arquivo **teste** não exista, ele é criado.

```
cat teste3.txt >> teste
```

c. O comando **cat** abaixo recebe como entrada o arquivo **teste.txt** e portanto, mostra na tela o conteúdo deste arquivo.

```
cat < teste.txt
```

d. Suponha que o diretório **xxxxx** não existe. Portanto, o comando

```
ls xxxxxx
```

apresenta uma mensagem de erro. Para direcionar a saída de erro deste comando para o arquivo **resultado**, basta digitar

```
ls xxxxxx >& resultado
```

ou

```
ls xxxxxx 2> resultado
```

e. Abaixo temos três comandos aninhados: primeiro, o comando **cat** exibe o conteúdo do arquivo **teste.txt**; segundo, o resultado do comando **cat** é usado como entrada do comando **wc**que conta o número de linhas do arquivo; e terceiro, o número de linhas do arquivo **teste.txt** é gravado no arquivo **resultado.txt**.

```
cat teste.txt | wc -l > resultado.txt
```

f. Podemos também usar o comando **tee** junto com o caractere **|** para que o resultado de um comando seja redirecionado para a saída padrão e para um arquivo. Por exemplo, suponha que queremos que o arquivo **teste.txt** seja copiado para o arquivo **teste2.txt** e que seja também exibido na tela. Então, basta digitarmos

```
cat teste.txt | tee teste2.txt
```

renice

renice prioridade [opções]

onde **prioridade** corresponde ao valor da nova prioridade a ser usada pelo(s) **processo(s)**. A prioridade de execução de um processo pode variar de **-20** (maior prioridade) a **19** (menor prioridade). Por padrão, a prioridade dos processos é zero.

São algumas das opções deste comando

-g gid : altera a prioridade de todos os processos que possuem o **gid** especificado.

-u usuário : altera a prioridade de todos os processos do usuário especificado.

-p pid : altera a prioridade do processo que possui o **pid** especificado. É o padrão.

Comentários sobre o comando

Por exemplo, o comando

renice +5 -p 374 895 -u aluno

altera para 5 a prioridade dos processos com **pid** 374 e 895 e a prioridade de todos os processos do usuário **aluno**.

Observações

Apenas o administrador do sistema (**root**) pode alterar prioridades de processos que não lhe pertençam e os usuários comuns podem somente incrementar o valor da prioridade.

Para definir, durante a inicialização de um processo, uma prioridade de execução diferente da prioridade padrão do sistema (valor zero), use o comando **nice**.

Para ver a prioridade de execução dos processos, use o aplicativo **top**.

rev

rev [arquivo]

Este comando copia o arquivo especificado para a saída padrão invertendo os caracteres de cada uma das linhas.

Comentários sobre o comando

Suponha que o arquivo **teste** possui o seguinte conteúdo:

teste1
12345
teste2
abcdefgh
9876543210

O comando

rev teste

mostra a seguinte saída

1etset
54321
2etset
hgfedcba
0123456789

Observações

Caso o nome de um arquivo não seja definido junto com o comando **rev**, o sistema usará a entrada padrão para receber os dados.

rlogin

rlogin [-l usuário] servidor

Este comando permite trabalhar remotamante com outro servidor.

Observações

Caso o nome do usuário não seja fornecido no comando, ele é solicitado durante o estabelecimento da sessão.

rm

rm [opções] arquivo...

São algumas das opções deste comando

-f : apaga sem pedir confirmação.

-i : apaga após pedir confirmação.

-r : apaga arquivos e subdiretórios.

-v : lista arquivos deletados.

Comentários sobre as opções do comando

Por exemplo, o comando

rm teste.txt

apaga o arquivo **teste.txt**.

Observações

Pode-se usar os caracteres **?**, ***** e **[ ]** como **curingas**. O primeiro substitui apenas um único caractere, o segundo substitui um número qualquer de caracteres, enquanto o terceiro substitui um único caractere dentro de um certo limite de valor. Por exemplo, suponha que o diretório atual tenha os seguintes arquivos: **teste1.txt**, **teste2.txt**, **teste3.txt** e **teste10.txt**. O comando

rm teste*.txt ou rm teste*

apaga os quatro arquivos, enquanto o comando

rm teste?.txt

apaga apenas os arquivos **teste1.txt**, **teste2.txt** e **teste3.txt**. Para apagar apenas os dois primeiros arquivos, basta digitar

rm teste[1-2].txt

rmdir

rmdir diretório

Comentários sobre o comando

Por exemplo, o comando

rmdir testes

remove o diretório **testes** dentro do diretório atual.

Para remover uma árvore de diretórios, use a opção **-p** com o comando **rmdir**. Por exemplo, o comando

rmdir -p testes/linux

remove o subdiretório **linux** e em seguida remove o diretório **testes**, pai do primeiro diretório.

Observações

Este comando só remove diretórios vazios. Para remover diretórios com todos os seus arquivos e subdiretórios use o comando **rm -r**. Por exemplo,

rm -r testes

remove o diretório **testes** e todos os arquivos e diretórios que estão hieráquicamente abaixo deste diretório.

É preciso ter permissão de gravação para remover um diretório.

rmmod

rmmod [opções] módulo

Este comando remove **módulos** do **kernel** que está executando.

São algumas das opções deste comando

-a : remove todos os módulos não usados.

-s : redireciona a saída do comando para o **syslog**.

Observações

Use o comando **lsmod** para listar os módulos que estão adicionados ao kernel.

root

Definição

O **root** é o usuário com maior nível de autorização dentro do **Linux**. Por exemplo, o **root** pode criar novos usuários, apagar usuários, alterar a configuração do sistema, etc.

Observações

Um usuário comum pode executar tarefas e/ou comandos como **root** usando o comando **su**, se conhecer a senha do superusuário.

rpm

rpm [opções]

Definição

O **rpm** é um poderoso gerenciador de pacotes que pode ser usado para construir, instalar, consultar, verificar, atualizar e desinstalar pacotes de software. Um pacote consiste em um arquivo de arquivos e informações sobre o pacote, incluindo nome, versão e descrição.

Os pacotes RPM possuem o seguinte formato:

pacote-versão-release.plataforma.rpm

Por exemplo,

gtk+-1.2.8-1.i386.rpm

temos que

- nome do arquivo = gtk+;
- versão = 1.2.8;
- release = 1;
- plataforma = i386.

Modos de Operação

O **rpm** possui vários modos de operação. Por exemplo,

- **Modo de Instalação**

  rpm -i [opções] pacote

- **Modo de Consulta**

  rpm -q [opções]

- **Modo de Verificação**

  rpm -V|-y|--verify [opções]

- **Modo de Desinstalação**

  rpm -e pacote

São algumas das opções deste comando

-i pacote : instala pacote.

-e pacote : desinstala pacote.

-F pacote : atualiza pacote já instalado (compara as versões).

-vh : exibe o caractere # à medida em que o arquivo é desempacotado e verifica dependências.

-q aplicativo : exibe a versão do pacote que forneceu o aplicativo especificado.

-qa : exibe a versão de todos os pacotes instalados.

-qf arquivo : consulta o pacote que contém o arquivo (forneça o caminho completo do arquivo).

-qi pacote : exibe as informações sobre o pacote, incluindo nome, versão e descrição.

-ql pacote : exibe a lista dos arquivos do pacote.

-qlc pacote : exibe a lista apenas dos arquivos de configuração do pacote.

-qld pacote : exibe a lista apenas dos arquivos de documentação do pacote.

-qR pacote : exibe a lista dos pacotes dos quais o pacote especificado depende.

-qs pacote : exibe o estado de cada arquivo do pacote (normal, instalado ou substituído).

-U pacote : substituir pacote (desinstalar e instalar novamente).

-V pacote : verifica pacote.

-Va : verifica todos os pacote instalados.

-Vf arquivo : verifica o pacote que possui o arquivo especificado.

-Vp pacote : verifica o pacote instalado e o arquivo de pacote RPM.

Verificação

No modo de verificação (opção **-V**), o formato da saída é uma string de 8 caracteres, um possível caractere **c** (neste caso o arquivo é de configuração) e o nome do arquivo. Cada uma das 8 caracteres representa a comparação de um atributo do arquivo com o valor registrado no banco de dados RPM. O caractere "." (ponto) indica que o teste foi bem sucedido (são iguais). A tabela abaixo mostra os caracteres que denotam falha em algum teste.

Código	Significado
5	Soma MD5
S	Tamanho do arquivo
L	Vínculo simbólico (link)
T	Mtime
D	Dispositivo
U	Usuário
G	Grupo
M	Modo (inclui permissões e tipo de arquivo)

Por exemplo, o comando

rpm -V php4

verifica o pacote **php4**. Uma possível saída para este comando é:

S.5....T c /etc/php4/cgi/php.ini

Neste caso, existem três atributos diferentes no arquivo **php.ini** que faz parte do pacote testado. Note que este é um arquivo de configuração do **php4**.

Exemplo

Abaixo mostramos o comando que instala e verifica o pacote que contém o compilador **gcc**.

rpm -ivh gcc-2.95.2-7cl.i386.rpm

A instalação acima não será feita caso o pacote seja dependente de outro(s) pacote(s). Neste caso, o sistema especifica o nome do pacote que está faltando, e você deve instalar o pacote citado antes de tentar instalar o **gcc** novamente.

separador de comandos

Usando ; para separar comandos

É possível digitar vários comandos em uma única linha de comandos, basta separar os comandos com o símbolo ponto-e-vírgula ("**;**"). Por exemplo,

cp teste.txt teste2.txt; ls

faz com que o sistema, primeiro copie o arquivo **teste.txt** para **teste2.txt** e em seguida, liste os arquivos do diretório corrente.

Usando ;\ para separar comandos

É também possível digitar vários comandos em diversas linhas antes do Linux executá-los. Para isto, basta separá-los com ponto-e-vírgula e barra ("**;**"). Usando o mesmo o exemplo acima temos

cp teste.txt teste2.txt;\
ls

set

set

Comentários sobre o comando

Não há necessidade de especificar parâmetros.

Para ver apenas o valor de uma determinada variável de ambiente use o comando **printenv**.

setserial

setserial [opções] device [parâmetro1 [arg]] ....

onde **device** é um equipamento serial que normalmente possui o formato **/dev/cua[0-3]**.

São algumas das opções deste comando

-a : exibe todas as informações disponíveis sobre o equipamento especificado.

-b : imprime um resumo das informações sobre o equipamento especificado.

-g : exibe as informações sobre os equipamentos especificados (mais de um nome é fornecido).

São algumas dos parâmetros deste comando

baud_base n : define a velocidade da porta como **n**.

irq n : define a interrupção da porta como **n**.

port n : define a porta de I/O do equipamento como **n**.

uart tipo : define o tipo do equipamento, onde **tipo** pode ser none (desabilita a porta), 8250, 16450, 16550, 16550A, 16650, 16650V2, 16654, 16750, 16850, 16950 ou 16954.

Comentários sobre as opções do comando

Por exemplo, o comando

setserial -a /dev/cua0

mostra todas as informações relacionadas ao equipamento **cua0**, enquanto o comando

setserial /dev/cua0 irq 4

define a interrupção 4 para o mesmo equipamento.

Observações

Se nenhuma opção é definida com o comando **setserial**, o sistema fornece o tipo, o número e a velocidade da porta do equipamento especificado.

setterm

setterm [opções]

Descrição

O comando **setterm** envia para a saída padrão um string de caracteres para ativar uma habilidade específica do terminal. As opções não implementadas pelo terminal são ignoradas.

São algumas das opções do comando

-background [black|red|green|yellow|blue|magenta|cyan|white|default] : altera a cor de fundo do terminal para a cor especificada.

-blank [num] : configura a proteção de tela no **shell** onde **num** corresponde ao número de minutos de inatividade do computador para que a proteção de tela seja executada. O valor de **num**pode variar de zero a 60 sendo que o valor zero corresponde a desativação da tela de proteção.

-foreground [black|red|green|yellow|blue|magenta|cyan|white|default] : altera a cor da letra do terminal para a cor especificada.

-reset : coloca o terminal na condição de recém ligado.

-store : passa a assumir as opções definidas no comando como padrão.

Comentários sobre o comando

Por exemplo, o comando

setterm -store -background blue

altera a cor do fundo do terminal para azul. Esta definição é agora padrão para os todos os terminais do sistema.

shell

Definição

Os comandos digitados pelo usuário são interpretados pelo **shell**. Esses comandos podem ser comandos embutidos do **shell**, mas na maioria das vezes eles são programas externos.

- A lista dos comandos embutidos pode ser obtida com o comando **help**. Para ver estes comandos, digite

  help

- Os comandos não embutidos são programas que se iniciam invocando-se um arquivo executável em algum lugar no sistema de arquivos do **Linux** (o **shell** pesquisa em todos os diretórios listados na variável de ambiente **PATH**).

O **shell** analisa sintaticamente a linha de comando depois que ela é lida. A variável de ambiente **IFS** determina como isso é feito. Normalmente, **IFS** é configurada de tal forma que espaços em branco separam os diferentes argumentos de linha de comando.

Existem vários **shells** para **Linux**, onde cada **shell** tem seus próprios recursos, capacidades e limitações. Por exemplo, o **shell** padrão para a distribuição **Conectiva** é o **bash**.

Na realidade, o **shell** é apenas um arquivo executável armazenado em **/bin**. No modo gráfico, um **shell** é executado em cada terminal aberto.

Para ver qual é o seu **shell** padrão, basta digitar o comando

printenv SHELL

O comando acima exibe o conteúdo da variável ambiente **SHELL** que contém o caminho completo do **shell**. Outra forma de saber qual é o **shell** padrão, é verificar o último parâmetro definido para o usuário no arquivo **/etc/passwd**. Por exemplo,

aluno:x:501:501::/home/aluno:/bin/bash

mostra que o usuário **aluno** usa o **shell bash**.

Carregando o shell

Suponha que o **shell** padrão do sistema seja o **bash**. Então, quando o usuário acessa o sistema, o **bash** carrega os seguintes arquivos do diretório **home** do usuário (estes arquivos são criados automaticamente pelo comando **adduser**, veja o diretório **/etc/skel** para saber quais são os arquivos padrão no seu sistema):

a. **.bashrc** : contém funções e nomes alternativos definidos pelo usuário.
b. **.bash_history** : contém a lista dos últimos (o padrão é 1000) comandos digitados pelo usuário.
c. **.bash_logout** : contém os comandos executados pelo sistema no fechamento da sessão pelo usuário.
d. **.bash_profile** : contém as variáveis de ambiente do usuário.

e os seguintes arquivos do diretório **/etc**:

a. **bashrc** : contém funções e nomes alternativos (**aliases**) do sistema.
b. **profile** : contém as variáveis de ambiente do sistema.

Os arquivos no diretório **home** são exclusivos do usuário e portanto, uma alteração nestes arquivos afeta apenas o usuário em questão. Os arquivos do **/etc** são comuns a todos os usuários e portanto, uma alteração nestes arquivos afeta todos os usuários do sistema.

Os arquivos carregados pelo **shell** definem as variáveis de ambiente, que nada mais são que definições e valores que o **shell** e os outros programas do sistema reconhecem. Para ver quais as variáveis de ambiente no seu sistema você pode digitar **printenv**, **env** ou **set**. Por exemplo, são algumas das variáveis de ambiente do **bash**:

- **$** : número do processo do comando em execução.
- **CDPATH** : mostra o caminho de busca para o comando **cd** (por exemplo, **CDPATH=.:~:/etc** faz com que o sistema utilize a seguinte ordem de busca do diretório especificado: 1) a partir do diretório atual; 2) a partir do diretório raiz do usuário; 3) a partir do diretório **/etc**).
- **HISTFILE** : mostra o nome do arquivo que armazena as linhas de comando digitadas pelo usuário (no **shell bash** o arquivo padrão é o **.bash_history**).
- **HISTSIZE** : mostra o número de linhas de comando digitadas pelo usuário que são memorizadas pelo sistema.
- **HOME** : mostra o diretório **home** do usuário.
- **IFS** : separador de campos usado para definir como dividir as linhas em palavras para serem processadas separadamente. O valor padrão de IFS é **<space><tab><new-line>**.
- **LOGNAME** : mostra o nome de acesso do usuário.
- **MAIL** : mostra o diretório que contém as mensagens de correio eletrônicas recebidas pelo usuário.
- **OLDPWD** : mostra o diretório anterior de trabalho do usuário.
- **OSTYPE** : mostra o sistema operacional em uso.
- **PATH** : mostra caminho de busca dos comandos digitados pelo usuário.
- **PPID** : mostra o número de identificação do processo que inicializou o **shell** do usuário. Existe um diretório em **/proc** com este número e que contém informações sobre o processo em questão.
- **PS1** : mostra a definição do **prompt** da linha de comando.
- **PS2** : mostra a definição do **prompt** secundário da linha de comando.
- **PWD** : mostra o diretório atual de trabalho do usuário.
- **SHELL** : mostra o nome do **shell** atualmente em uso.
- **SHLVL** : mostra o número de **shells** atualmente em execução na conta do usuário.
- **TERM** : mostra o tipo de terminal em uso.
- **TZ** : define o fuso horário a ser usado pelo sistema.
- **UID** : mostra o número de identificação do usuário.
- **USER** : mostra o nome do usuário atual.

É possível criar novas variáveis, excluir variáveis existentes ou apenas alterar o conteúdo de uma variável de ambiente.

Para criar uma nova variável de ambiente, basta definir o nome e o valor da nova variável de ambiente e usar o comando **export** para permitir que a variável seja visualizada pelos aplicativos (por exemplo, um novo **shell** ou um novo terminal) inicializados no mesmo terminal (neste caso a variável existirá enquanto a sessão estiver aberta). Por exemplo,

TESTE=10; export TESTE

ou

export TESTE=10

cria a variável de ambiente **TESTE** com valor inicial 10. O comando **export** faz com que a nova variável seja conhecida por todos os processos a partir deste **shell**. Os nomes das variáveis de ambiente são, tradicionamente, definidas usando apenas letras maiúsculas. Entretanto, isto não é obrigatório. Você pode também usar letras minúsculas. Mas, **CUIDADO!** O Linux é **case sensitive**. Isto significa que o sistema diferencia letras maiúsculas de letras minúsculas. Portanto, o comando

teste=10; export teste

cria uma nova variável de ambiente chamada **teste** e que é diferente da variável **TESTE** criada anteriormente.

É importante observar que as variáveis de ambiente definidas a partir da linha de comando são temporárias. Para criar uma variável permanente, deve-se acrescentar a definição e o comando**export** no arquivo **.bash_profile** (no caso de **shell bash**).

Para excluir uma variável de ambiente, deve-se usar o comando **unset**. Por exemplo, o comando

unset teste

exclui a variável de ambiente **teste** criada no exemplo anterior.

Para alterar o valor de uma variável de ambiente, basta fornecer o nome da variável e o novo valor a ser atribuido a variável. Para a variável de ambiente **TESTE** definida anteriormente nesta seção, podemos digitar

TESTE=200

e a variável **TESTE** passa a ter valor 200. Neste caso, o conteúdo da variável é completamente alterado. Pode ser que ao invés de alterar o conteúdo, você queira apenas acrescentar mais alguma informação ao conteúdo armazenado em uma variável. Por exemplo, suponha que você queira acrescentar o diretório **/teste/bin** no caminho de busca, ou seja, no **PATH**. Devemos, então digitar

PATH=$PATH:/teste/bin

O símbolo **$** usado acima antes de **PATH** informa ao **shell** para usar o conteúdo da variável de ambiente PATH.

Suponha agora que temos **USER=aluno** e que queremos **USER=aluno linux**. Então podemos definir

USER="$USER Linux"

As aspas acima são necessárias devido ao espaço em branco existente no novo conteúdo da variável **USER**. Note que novamente usamos **$** para indicar ao **shell** que se deve substituir o nome **USER** pelo conteúdo da variável ambiente **USER**.

Os exemplos mostrados acima são alterações temporárias no ambiente do usuário. Para tornar uma alteração efetiva, deve-se alterar o arquivo profile do usuário (**.bash_profile** no **shell bash**). Por exemplo, suponha que queremos personalizar o **prompt** das linhas de comando. Podemos então incluir no arquivo de profile

PS1="[\W]\$"

e

export PS1

onde o novo **prompt** apenas exibe o nome do diretório atual de trabalho do usuário (**\W**) e o símbolo **$** (**\$**).

Consulte o manual on line para conhecer um pouco mais sobre os comandos embutidos no seu **shell** (por exemplo, **man bash** para ler sobre o **script bash**).

Usando as variáveis de ambiente em um programa C

O terceiro argumento da função **main()** é a lista de variáveis de ambiente. Em um programa C, pode-se usar a função **putenv()** para criar/alterar variáveis de ambiente, a função **getenv()** para obter o valor das variáveis e a função **unsetenv()** para remover variáveis.

O exemplo mostrado a seguir exibe um **dump** das variáveis de ambiente e de algumas funções de C, antes e depois, que algumas modificações são introduzidas no ambiente do sistema.

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<unistd.h>
4 #include<string.h>
5 #include<errno.h>
6 void imprime_variavel(const char *desc, const void *addr)
7 {
8 printf("0x%08lX %-s\n", (long)addr, desc);
9 }
10 void dump(char * const envp[], const char *cabecalho)
11 {
12 int x; /* índice envp[x] */
13 /* Exibe um título */
14 printf("\n%s\n\n", cabecalho);
15
16 /* Exibe as variáveis de ambiente */
17 for (x = 0; envp[x] != 0; ++x)
18 {
19 imprime_variavel(envp[x], envp+x); /* imprime variável */
20 }
21 /* Exibe outras variáveis */
22 imprime_variavel("stderr",stderr);/* Dump do endereço de stderr */
23 imprime_variavel("stdout",stdout);/* Dump do endereço de stdout */
24 imprime_variavel("stdin", stdin); /* Dump do endereço de stdin */
25 imprime_variavel("dump", dump); /* Dump do endereço de main */
26 }
27 int main(int argc, char * const argv[], char * const envp[])
28 {
29 /* Faz dump das variáveis de ambiente */
30 dump(envp, "AMBIENTE INICIAL:");
31 /* inclui duas novas variáveis de ambiente */
32 if (putenv("TESTE1=100") == -1)
33 {
34 printf("%s: putenv()\n", strerror(errno));
35 exit(1);
36 }
37 if (putenv("TESTE2=teste de variaveis") == -1)
38 {
39 printf("%s: putenv()\n", strerror(errno));
40 exit(2);
41 }
42 /* altera valor da variável de ambiente LOGNAME */
43 if (putenv("LOGNAME=linux") == -1)
44 {
45 printf("%s: putenv()\n", strerror(errno));
46 exit(3);
47 }
48 /* Faz dump das variáveis de ambiente */
49 dump(envp, "AMBIENTE Modificado:");
50 /* Verifica o valor de LOGNAME, TESTE1 e TESTE2 */
51 printf("\ngetenv(LOGNAME) = %s;", getenv("LOGNAME"));
52 printf("\ngetenv(TESTE1) = %s;", getenv("TESTE1"));
53 printf("\ngetenv(TESTE2) = %s;\n", getenv("TESTE2"));
54 return 0;
55 }
```

Uma observação interessante em relação ao programa acima diz respeito ao **envp[]** antes e depois da alteração da variável **LOGNAME** e da inclusão das variáveis **TESTE1** e **TESTE2** : o array **envp[]** não se altera. Isto significa que as alterações introduzidas pelo programa no ambiente são feitas em outra área de

memória e monitoradas pelo sistema enquanto o programa está sendo executado. Não é possível alterar as variáveis de ambiente do **shell** pois as variáveis do **shell** são definidas na própria memória privada do **shell**.

Alteração do shell

É possível mudar de **shell** em tempo de execução. Para isto, basta digitar o nome do novo **shell** na linha de comandos. Para encerrar o novo **shell** e voltar para o **shell** anterior basta digitar**exit**. Para ver quais os **shells** que estão disponíveis no sistema, basta digitar

```
chsh -l
```

Na realidade, este comando exibe o conteúdo do arquivo **/etc/shells**. Para mudar o **shell** padrão, o usuário pode usar o comando **chsh** (**change shell**). Suponha, por exemplo, que queremos adotar como padrão o **shell ash**, então podemos digitar

```
chsh -s /bin/ash
```

O Linux apenas pedirá a senha do usuário logado para confirmar a alteração. Na próxima vez que o usuário logar no sistema, o **ash** será o seu **shell** padrão. O uso do comando **chsh**, sem parâmetros, fará com que, além da senha, o Linux também solicite o nome do novo **shell**. Neste caso, deve-se fornecer o caminho completo do novo **shell**.

É também importante observar que quando uma nova sessão é aberta com o comando **bash**, apenas os arquivos **/etc/bash** e **~/.bashrc** são lidos. Nestes arquivos são colocados os comandos que devem continuar existindo na nova sessão como, por exemplo, os **aliases**.

Programação em shell script

Como vimos acima, o **shell** funciona como mediador entre o usuário e o **kernel** do sistema. Quando o usuário digita um comando, o **shell** analisa o comando e solicita ao **kernel** a execução das ações correspondentes ao comando em questão.

Normalmente, o usuário (principalmente se ele é o administrador do sistema) executa um mesmo conjunto de tarefas diariamente. O ideal então é que o usuário possa automatizar as suas tarefas, ou seja, o usuário digita um único comando e o **shell** o interpreta como um conjunto de comandos a serem executados pelo sistema. Este tipo de automatização de tarefas é possível através da programação **shell**. Abaixo mostramos alguns exemplos de **scripts**.

- **EXEMPLO 1**

  Suponha que você queira bloqueiar o acesso de um usuário ao sistema e caso o usuário tente acessar o sistema mostrar uma mensagem de aviso. Para isto é necessário: criar um **shell script** que será usado na inicialização da conta do usuário e definir este **script** para ser chamado quando o usuário logar na conta. Por exemplo, queremos bloqueiar a conta do usuário**aluno**. Então, devemos

  a) criar o **script** abaixo (sem a numeração do lado esquerdo)

```
1 #!/bin/sh
2 echo '************************************************'
3 echo '* Sua conta foi encerrada. Procure o suporte. *'
4 echo '************************************************'
5 sleep 10s
6 exit
```

  - A linha 1 indica que o script deve ser executado pelo **shell sh**. O símbolo # significa início de um comentário (o resto da linha não é interpretada), mas quando **#!** aparecem na

primeira linha de um **script**,estamos informando ao sistema para procurar o programa definido a seguir (neste caso o **/bin/sh**) e transmitir o resto do arquivo a esse programa.

- O comando **echo** (linhas 2 a 4) imprime o texto entre aspa na tela.
- O comando **sleep 10s** (linha 5) dá uma pausa de 10 segundos antes de continuar a execução do **script**
- O comando **exit** (linha 6) finaliza o **script**.

Salve o script acima em **/bin** com o nome **nsh** e torne este arquivo executável (use o comando **chmod +x nsh**).

b) alterar o arquivo **/etc/passwd** para que o novo **script** seja chamado quando o usuário logar no sistema.

```
aluno:x:501:501::/home/aluno:/bin/nsh
```

Agora, quando o usuário **aluno** tentar logar no sistema, ele receberá a mensagem que a conta foi encerrada.

- **EXEMPLO 2**

Abaixo temos um segundo exemplo de **script** de **shell**. Este **script** mostra quais as **permissões de acesso** do usuário em relação a um determinado arquivo.

```
1 #!/bin/sh
2 echo -n 'Forneça o nome do arquivo a ser verificado: '
3 read
4 if [ $REPLY ] #usuário digitou o nome do arquivo?
5 then
6 arq=$REPLY
7 if [ -e $arq ]; then # o arquivo existe ?
8 echo 'o arquivo existe'
9 if [ -r $arq ]; then # o usuário pode ler o arquivo?
10 echo 'você pode ler o arquivo'
11 else
12 echo 'você não pode ler o arquivo'
13 fi
14 if [ -w $arq ]; then # o usuário pode alterar o arquivo?
15 echo 'você pode alterar o arquivo'
16 else
17 echo 'você não pode alterar o arquivo'
18 fi
19 if [ -x $arq ]; then # o usuário pode executar o arquivo?
20 echo 'você pode executar o arquivo'
21 else
22 echo 'você não pode executar o arquivo'
23 fi
24 if [ -d $arq ]; then # o arquivo é um diretório?
25 echo 'O arquivo é um diretório'
26 fi
27 else
28 echo 'o arquivo nâo existe'
29 fi
30 else
31 echo 'Você não forneceu o nome do arquivo'
32 fi
33 exit
```

Em relação ao **script** acima podemos comentar:

- A linha 1 especifica que o **script** deve ser executado pelo **shell sh**.

- A linha 2 exibe na tela a mensagem 'Forneça o nome do arquivo a ser verificado: '. O parâmetro **-n** do comando **echo** indica que o cursor não deve mudar de linha após a exibição da frase. Em relação ao comando **echo**, é também importante observar que podemos usar tanto aspas simples quanto aspas duplas, embora sejam interpretadas de forma diferente pelo **shell**. Por exemplo, suponha uma variável denominada **TESTE** e que tenha armazenado o valor **Maria**. A execução dos comandos

```
echo 'TESTE = $TESTE'
echo "TESTE = $TESTE"
```

  mostram, respectivamente, os seguintes resultados

```
TESTE = $TESTE
TESTE = Maria
```

  No primeiro caso, o uso de aspas simples informa ao **bash** para imprimir o conteúdo do **string**, sem nenhuma preocupação adicional. No segundo caso, o uso de aspas duplas faz com que o **sh** substitua o nome da variável **TESTE** pelo seu conteúdo. O símbolo **$** indica ao **sh** quem é variável dentro do string.

- O comando **read** da linha 3 recebe o nome do arquivo digitado pelo usuário e o armazena na variável padrão **REPLY**. Pode-se também usar uma variável qualquer para receber o nome do arquivo. Por exemplo, você pode alterar a linha 3 para **read arq**. Neste caso, o **sh** passa a executar duas ações quando interpreta a linha 3: primeiro, cria a variável **arq**, e segundo, armazena o nome do arquivo nesta variável.
- A linha 4 do **script** verifica se o usuário digitou algo (ele pode ter apenas teclado **ENTER**). O comando **if** possui a seguinte estrutura

```
if [ condição ]
then
    comandos
else
    comandos
fi
```

  Antes de discutirmos o **script**, algumas observações em relação ao comando **if** tornam-se necessárias. A condição (ou condições) a ser testada deve ser colocada entre colchetes (pode-se também usar explicitamente a palavra **test** no lugar dos colchetes, por exemplo, você pode substituir **if [$REPLY]** por **if test $REPLY**). Além disso, deve existir um espaço entre a condição e o colchete (abrindo e/ou fechando). O **sh** retorna 0 ou 1 como resultado do teste, dependendo se a condição é verdadeira (valor zero) ou falsa (valor 1). Caso a condição seja verdadeira, são executados os comandos definidos logo após o comando **then**. Caso a condição seja falsa, são executados os comandos logo após o comando **else** (você não é obrigado a definir um **else** para cada **if**). O comando **if** é fechado com um comando **fi**.

  Olhando novamente o segundo exemplo de **script** podemos notar a seguinte estrutura

```
4 if [ $REPLY ] #usuário digitou o nome do arquivo?
5 then
     executa linhas de 6 a 29
30 else
     executa linha 31
32 fi
```

  A linha 4 verifica se a variável **REPLY** tem algum valor armazenado. Caso o resultado do teste seja verdadeiro, são executados os comandos da linha 6 a linha 29; caso o resultado do teste seja falso, apenas a linha 32 é executada.

- A linha 6 define a variável **arq** e copia o conteúdo da variável **REPLY** para a nova variável. Podemos aqui fazer três observações: primeiro, não existe espaço em branco entre as variáveis e o símbolo de atribuição ("="); segundo, a criação da variável **arq** não é necessária, poderíamos continuar usando a variável **REPLY** no resto do **script**; e terceiro, a variável **arq** é uma variável local (não existe um comando **export** para esta variável), isto significa que esta variável existe somente durante a execução do **script**.
- A linha 7 mostra o comando **if** com o operador **-e**. O uso deste operador faz com que o **sh** verifique a existência do arquivo cujo nome foi informado pelo usuário. Podemos ver que o **script** tem a seguinte estrutura a partir deste teste

```
7 if [ -e $arq ]; then # o arquivo existe ?
        executa linhas de 8 a 26
27 else
        executa linha 28
29 fi
```

Note que temos dois comandos na linha 7: o comando **if** e o comando **then**. Quando mais de um comando são colocados em uma mesma linha, eles devem ser separados por um**ponto-e-vírgula**.

As linhas de 8 a 26 utilizam operadores junto com o comando **if** para verificar as **permissões** do arquivo. Abaixo mostramos alguns operadores que podem ser usados para testar arquivos (digite **man test** para obter mais informações).

-b : o arquivo existe e é um arquivo é especial de bloco ?
-c : o arquivo existe e é um arquivo é especial de caractere ?
-d : o arquivo existe e é um diretório ?
-e : o arquivo existe ?
-f : o arquivo existe e é um arquivo normal ?
-r : o arquivo existe e o usuário pode lê-lo ?
-s : o arquivo existe e tem tamanho maior que zero ?
-x : o arquivo existe e o usuário pode executá-lo ?
-w : o arquivo existe e o usuário pode alterá-lo ?
-G : o arquivo existe e pertence ao grupo do usuário ?
-L : o arquivo existe e é um link simbólico ?
-O : o arquivo existe e o usuário é dono do arquivo ?
arq1 nt arq2 : o arquivo arq1 é mais novo que o arquivo arq2 ?
arq1 ot arq2 : o arquivo arq1 é mais antigo que o arquivo arq2 ?

- **EXEMPLO 3**

O terceiro **script** é uma ferramenta de **backup** para arquivos com extensão **txt**. O usuário fornece o nome do diretório e todos os arquivos **.txt** deste diretório são copiados para o diretório **backup**. Se o diretório **backup** já existe, ele é inicialmente apagado e depois criado.

```
1 #!/bin/sh
2 if [ ! $1 ]
3 then
4 echo 'Você deve fornecer o nome do diretório'
5 exit 1
6 fi
7 if [ ! -d $1 ]
8 then
9 echo "$1 não é um diretório"
10 exit 1
11 fi
12 rm -fr backup
13 mkdir ~/backup
14 for i in $1/*.txt; do
15 echo $i
16 cp -f $i ~/backup/
```

```
17 done
18 exit
```

Podemos comentar em relação ao **script** acima:

- A linha 1 especifica que o **script** deve ser executado pelo **shell sh**.
- A linha 2 testa se o usuário forneceu algum argumento de linha de comando. A variável **$1** corresponde ao primeiro argumento, a variável **$2** corresponde ao segundo argumento, e assim por diante. A variável **$0** possui o nome do programa e a variável **$*** possui a lista das variáveis (**$0**, **$1**, **$2**, ...) . O símbolo **!** é o símbolo de negação (NÃO), portanto estamos perguntando na linha 2 se o agumento não foi fornecido pelo usuário. Caso seja verdade, o programa mostra a frase 'Você deve fornecer o nome do diretório' e encerra a execução de forma anormal (**exit 1**).
- A linha 7 verifica se o nome fornecido pelo usuário não é o nome de um diretório. Caso isto seja verdade, o programa é encerrado.
- A linha 12 apaga o diretório **~/backup** e todos os seus arquivos, caso este diretório exista.
- A linha 13 cria o diretório **~/backup**. Note que o diretório é filho do diretório principal (**raiz**) do usuário.
- Nas linhas de 14 a 17 temos a cópia dos arquivos. A linha 14 define que para cada arquivo **.txt** do diretório fornecido pelo usuário, deve-se executar os comandos das linhas 15 (exibe o nome do arquivo na tela) e 16 (copia o arquivo para diretório **~/backup**). As instruções dentro do laço **for** serão executadas tantas vezes quantas forem o número de arquivos com extensão **txt** no diretório fornecido pelo usuário. Portanto, a variável local armazena um nome do arquivo diferente a cada execução do laço.
- A linha 18 encerra o **script**.

- **EXEMPLO 4**

O quarto exemplo de script é o arquivo **.bash_profile** usado pelo sistema para definir o ambiente de trabalho do usuário durante o processo de inicialização.

```
1 # .bash_profile
2 if [ -f ~/.bashrc ]; then
3 . ~/.bashrc
4 fi
5 # User specific environment and startup programs
6 PS1="[\W]\$"
7 PATH=$PATH:$HOME/bin
8 BASH_ENV=$HOME/.bashrc
9 USERNAME=""
10 export USERNAME BASH_ENV PATH PS1
```

Podemos notar:

- A linha 1 é apenas uma linha de comentário. Não há necessidade de especificar o nome do interpretador pois o **script** é automaticamente executado pelo **shell** durante a inicialização da conta do usuário.
- A linha 2 verifica se o arquivo **~/.bashrc** existe e se ele é um arquivo normal. Caso seja verdade, a linha 3 é executada. O ponto (.) no início da linha 3 indica que o arquivo definido a seguir na mesma linha deve ser carregado e executado.
- As linhas de 5 a 10 definem variáveis de ambiente (**PS1**, **PATH**, **BASH_ENV** e **USERNAME**).

É importante observar que existe outra forma de executar os scripts. Ao invés de alterar as **permissões de acesso** do script com o comando **chmod**, pode-se simplesmente especificar o nome do **shell** na linha de comando. Por exemplo, suponha que queremos executar o arquivo **teste** usando o **script bash**. Então podemos digitar

```
bash teste
```

shutdown

shutdown [opções] [msg]

onde **msg** corresponde a mensagem enviada aos usuários logados.

São algumas das opções deste comando

now : executa o comando imediatamente.

-t seg : executa o comando após **seg** segundos.

-r : reinicializa o sistema após a parada.

-h : desliga o sistema.

-k : apenas manda mensagem para usuários sem parar o sistema.

Exemplos

shutdown -h now

shutdown -h -t 15 "O sistema será encerrado em 15 segundos"

shutdown -r now

shutdown -r -t 10 "o sistema será reinicializado após 10 min"

Observações

Outra forma de reinicializar o sistema é usar a combinação das teclas **CTRL+ALT+DEL**. Para restringir o uso desta forma de reinicialização, basta criar o arquivo **/etc/shutdown.allow**.

O comando **init** pode também ser usado no lugar do comando **shutdown**. Para parar o sistema podemos usar o comando

init 0

e para reinicializar o sistema podemos usar o comando

init 6

sistemas de arquivos

Partições

Podemos dividir um disco rígido em várias partes ou partições, onde cada partição é independente das outras, ou seja, cada partição pode ter o seu próprio sistema de arquivo. Isto significa que uma partição do disco não interfere nas outras partições. Podemos, por exemplo, instalar o **Linux** em uma partição e o **Windows** em outra partição.

Um disco pode ser dividido em até 4 partições. Uma partição pode ser primária ou estendida. Sendo que, no máximo, apenas uma partição pode ser do tipo estendida. Isto significa que você pode ter 4 partições primárias ou 3 partições primárias e uma partição estendida. É possível dividir uma partição estendida em até 64 partições menores chamadas de partições lógicas (a partição estendida não armazena dados e sim, outras partições lógicas). Não é possível, entretanto, dividir uma partição primária.

A tabela onde são armazenadas as informações sobre as partições fica no primeiro setor do disco e chama-se **MBR** (Master Boot Record). Esta tabela possui 4 entradas onde cada entrada descreve uma única partição.

Deve-se colocar o DOS em uma partição primária, pois este sistema operacional não consegue inicializar numa partição lógica, enquanto o Linux nã possui nenhuma restrição (veja **LILO** para mais detalhes). Por outro lado, o Linux requer na sua instalação a criação de pelo menos duas partições, uma para instalar o próprio Linux (partição Linux nativo) e a outra para servir de memória auxiliar para o Linux (partição de **swap** ou troca). A partição Linux nativo é conhecida por diretório **raiz** do Linux e é representada por ***/***.

O programa mais comumente usado para particionar discos é o **fdisk**. O problema com este aplicativo é que ele destrói os dados armazenados ao particionar o disco.

Sistemas de arquivos

A tabela abaixo mostra os tipos mais comuns de sistemas de arquivos que são reconhecidos pelo Linux.

Tipo	Descrição
ext	Sistema nativo Linux antigo
ext2	Atual sistema nativo Linux de arquivos
iso9660	Sistema de arquivos do CD-ROOM (read only)
msdos	Sistema de arquivos DOS
nfs	Sistema de arquivo remoto NFS
proc	Sistema de arquivos Linux Process Information
swap	Sistema de arquivos de troca (swap)
umsdos	Sistema de arquivos para suportar arquivos DOS e Linux coexistentes
vfat	Sistema de arquivos Windows (permite definição de nomes de arquivos com até 32 caracteres)

Para verificar quais os sistemas de arquivos que o seu Linux suporta, basta verificar o conteúdo do arquivo **/proc/filesystems**. O exemplo abaixo mostra que o sistema pode montar um CD-ROOM e um sistema de arquivos DOS ou Windows.

```
        ext2
nodev proc
        iso9660
nodev devpts
        msdos
```

O suporte para diferentes sistemas de arquivos pode ser obtido através de **módulos** de **kernel** carregáveis no diretório **/lib/modules/XXX/fs**, onde **XXX** é a versão atual do Linux.

Sistemas de arquivos do Linux

No Linux, um diretório (corresponde ao conceito de pasta do Windows) pode ter outros diretórios ou arquivos. Dizemos que um diretório é **filho** de outro diretório quando ele está logo abaixo do diretório em questão. O diretório que está um nível acima é chamado de diretório **pai**.

O diretório **raiz** do Linux (ou diretório **/**) é o diretório com maior hierarquia entre todos os diretórios do sistema. Isto significa que todos os diretórios do Linux ficam abaixo deste diretório. Por padrão (independente da distribuição usada), temos no Linux os seguintes diretórios abaixo do diretório **raiz** :

- **bin** - diretório com os arquivos binários de comandos disponíveis para todos os usuários.
- **boot** - diretório com os arquivos do **boot** de inicialização.
- **dev** - diretório com as definições dos dispositivos de entrada/saída.
- **etc** - diretório com os arquivos de configuração do sistema.
- **home** - diretório que armazena os diretórios dos usuários do sistema.
- **lib** - diretório com as bibliotecas e módulos (carregáveis) do sistema.
- **mnt** - diretório usado para montagem de partições.
- **opt** - diretório usado para instalar pacotes opcionais que não fazem parte da distribuição **Linux**.
- **proc** - diretório com informações sobre os processos do sistema.
- **root** - diretório **home** do administrador do sistema.
- **sbin** - diretório com os arquivos binários de comandos disponíveis para administrar o sistema.
- **tmp** - diretório com arquivos temporários.
- **usr** - diretório com arquivos binários, páginas de manual e outros arquivos imutáveis utilizados pelos usuários do sistema, mas que não são essenciais para o uso do sistema como, por exemplo, o sistema de janelas X, jogos, bibliotecas compartilhadas e programas de usuários e de administração.
- **var** - diretório com arquivos de dados variáveis (spool, logs, etc).

O diretório inicial de um usuário é conhecido como diretório **home**, diretório **raiz do usuário** ou diretório **principal do usuário**. Este diretório é o de maior hierarquia no conjunto de diretórios do usuário. O usuário tem total controle sobre o seu diretório **home** e sobre todos os arquivos e diretórios que estão abaixo dele. Isto significa que o usuário pode criar, deletar e modificar tudo que está abaixo do seu diretório principal. Além disso, o usuário pode definir as **permissões de acesso** (leitura, gravação e execução) dos outros usuários em relação aos seus arquivos.

Normalmente, o diretório **raiz** de um usuário está localizado em **/home** e possui o mesmo nome do **login** de acesso do usuário (o **root** pode mudar este padrão durante a criação da conta do usuário). Por exemplo, o diretório **raiz** do usuário **aluno**, por padrão, é **/home/aluno**.

Convém também observar que é possível colocar os subdiretórios do diretório raiz em partições separadas. O objetivo é facilitar a manutenção do sistema e aumentar a segurança dos dados. Portanto, a distribuição do diretório raiz em várias partições é uma escolha pessoal do administrador do sistema. Normalmente, é sugerido que os seguintes diretórios possuam uma partição própria: **/home**, **/opt**, **/tmp**, **/usr** e **/usr/local**.

sort

sort [opções] arquivo

São algumas das opções deste comando

-b : ignora espaços em branco.

-c : retorna uma mensagem de erro se o arquivo não estiver ordenado.

-f : ignora a diferença entre letras maiúsculas e letras minúsculas.

-r : organiza em ordem decrescente.

-n : os valores numéricos são organizados na ordem aritmética.

Comentários sobre as opções do comando

Por exemplo, suponha um arquivo chamado **teste.txt** com o seguinte conteúdo:

```
aaa
BBB
  53
0101
02
2
34
Aa
aA
```

A saída para o comando

sort teste.txt

é a seguinte

```
  53
0101
02
2
34
Aa
BBB
aA
aaa
```

Note que o comando trata o arquivo como um conjunto de caracteres onde a **ordem crescente** é: espaços, números, letra maiúsculas [A-Z] e letras minúsculas [a-z].

source

source arquivo

Comentários sobre o comando

Por exemplo, após modificar o seu arquivo de **profile** você pode usar o comando **source** para tornar as alterações válidas, sem precisar sair e entrar novamente no sistema. Para os usuários que usam o **shell** **bash**, este arquivo é o **.bash_profile** que fica no diretório inicial do usuário e portanto, o seguinte comando deve ser digitado

source .bash_profile

split

split [opções] [arquivo [prefixo]]

onde

- **arquivo** é o nome do arquivo a ser dividido. Se **arquivo** não é fornecido, o sistema usa a entrada padrão.
- **prefixo** corresponde ao prefixo usado na geração dos nomes dos arquivos menores.

São algumas das opções deste comando

-b num[bkm] : coloca **num** bytes, kbytes ou Mbytes em cada arquivo.

-l num: coloca **num** linhas em cada arquivo.

Comentários sobre as opções do comando

Os nomes dos arquivos de saída são formados do **prefixo** mais os sufixos **aa** (para o primeiro arquivo), **ab** (para o segundo arquivo), e assim por diante (até **zz**). Portanto, a saída é **prefixoaa**,**prefixoab**, etc. Caso não seja fornecido um valor para ser usado como **prefixo**, o sistema usa o caractere **x**.

Por exemplo, o comando

split -l 500 teste.txt novo

divide o arquivo **teste.txt** em arquivos menores de 500 linhas cada. Os arquivos criados são **novoaa**, **novoab**, **novoac**, etc.

Observações

Note que o último arquivo gerado pelo comando pode ter um tamanho menor que os demais arquivos, pois o último arquivo representa o resto da divisão do arquivo maior em arquivos menores.

strfile

strfile [opções] arquivo [arquivo de saída]

O comando **strfile** lê um arquivo que contém frases separadas pelo símbolo **%** e cria um arquivo para acesso aleatório dessas frases.

O arquivo de saída possui extensão **dat**.

São algumas das opções deste comando

-c char : altera o caractere delimitador de **%** para **char**.

-o : os strings são organizados em ordem alfabética.

-r : os strings são organizados aleatoriamente.

Comentários sobre as opções do comando

Suponha que o arquivo **teste** seja um conjunto de frases separadas pelo símbolo **%** como mostrado abaixo.

A banker is a fellow who lends you his umbrella when the sun is shining
and wants it back the minute it begins to rain.
        -- Mark Twain
%
A classic is something that everyone wants to have read
and nobody wants to read.
        -- Mark Twain, "The Disappearance of Literature"
% A horse! A horse! My kingdom for a horse!
        -- Wm. Shakespeare, "Henry VI"
%

O comando

strfile teste

cria então o arquivo **teste.dat** de acesso aleatório, ou seja, uma das frases acima será selecionada aleatoriamente a cada leitura do arquivo **teste.dat**.

Observações

Um exemplo do uso de um arquivo criado pelo comando **strfile** é o aplicativo **fortune**.

strings

strings [opções] arquivo

São algumas das opções deste comando

-a : verifica todo o arquivo.

-f : coloca o nome do arquivo de entrada no início de cada linha de saída.

-v : mostra a versão do aplicativo.

Comentários sobre as opções do comando

Por exemplo, o comando

strings /bin/bash

exibe os strings (imprimíveis) do arquivo binário **/bin/bash**.

stty

stty [opções]

São algumas das opções deste comando

-a : exibe todas as configurações atuais do terminal.

-g : exibe todas as configurações atuais do terminal em um formato que pode ser usado como argumento para outro comando **stty** para restaurar as configurações atuais.

sane : restaura a configuração padrão no terminal.

Comentários sobre as opções do comando

Caso o comando **stty** seja usado sem nenhum argumento, o sistema informa a velocidade de comunicação e os parâmetros de linha que são diferentes dos valores configurados por **stty sane**.

su

su [opções] [usuário]

Digitando apenas **su** temos acesso de superusuário (**root**) após o fornecimento da senha. Digite **exit** para retornar a seção anterior.

São algumas das opções deste comando

- **usuário -c comando** : não começa um novo **shell**, apenas executa o comando como outro usuário. Por exemplo, suponha que queremos ver o conteúdo do arquivo **/etc/shadow** que possui as senhas criptografadas dos usuários. Apenas o **root** possui permissão de leitura para este arquivo, portanto podemos digitar

  ```
  su root -c 'more /etc/shadow'
  ```

  O sistema então solicita a senha do usuário **root** antes de executar o comando. Na relidade, não há necessidade de especificar o usuário no comando acima pois quando o nome do usuário não é fornecido, o sistema assume como padrão o usuário **root**.

- **- usuário** : altera as **variáveis de ambiente** como **TER**, **HOME** e **SHELL**. Caso o nome do usuário não seja informado, o sistema assume que o usuário é o **root**. Logo, para trabalhar como **root**, basta digitar

  ```
  su -
  ```

- **-s shell** : executa um determinado **shell**. Por exemplo, o comando

  ```
  su -s /bin/sh
  ```

  apenas modifica o **shell** do usuário, mas o comando

  ```
  su - root -s /bin/sh
  ```

  altera o usuário para **root** utilizando o **shell sh** com este usuário.

sudo

sudo [opções] [comando]

São algumas das opções deste comando

-l : lista os comandos permitidos (e os comandos proibidos) para o usuário na máquina atual.

-u usuário : o **sudo** executa o comando com os privilégios do usuário especificado.

Comentários sobre as opções do comando

Por exemplo, o comando

sudo vipw

permitirá que o usuário execute o aplicativo **vipw** como se fosse o **root**.

Observações

Para executar um comando **sudo**, é preciso que exista a autorização no arquivo **/etc/sudoers**.

swap

Descrição

O **swap** é uma área de troca usada para aumentar a quantidade de memória RAM do sistema.

Até o **kernel** **2.1**, o tamanho máximo de uma partição de **swap** era de 128MB. A partir do **kernel 2.2** essa limitação passou a ser de 2GB.

O Linux pode ter mais de 8 áreas de troca ativas ao mesmo tempo. Entretanto, o total da área de **swap** não pode ultrapassar 16GB.

Comandos relacionados

- **mkswap** : formata uma área de **swap**.
- **swapoff** : desabilita dispositivos de **swap**.
- **swapon** : habilita dispositivos de **swap**.

swapoff

swapoff [opções]

São opções deste comando

dispositivo : desabilita o dispositivo especificado como área de troca.

-a : desabilita todos os dispositivos marcados com **sw** em **/etc/fstab**.

-h : apresenta mensagem de ajuda.

-V : lista a versão do programa.

Comentários sobre as opções do comando

Por exemplo,

swapoff /dev/hda3

desabilita a partição **/dev/hda3** como área de troca.

swapon

swapon [opções]

São algumas das opções deste comando

-a : disponibiliza todos os dispositivos marcados com "sw" em /etc/fstab.

-h : apresenta mensagem de ajuda.

-s : lista um resumo do uso da área de troca por dispositivo.

-V : lista a versão do programa.

Comentários sobre as opções do comando

Deve-se usar primeiro o comando **mkswap** para criar a área de **swap** e só então, usar o comando **swapon**. Por exemplo,

swapon /dev/hda3

habilita a partição **/dev/hda3** como área de troca.

sync

sync

Comentários sobre o comando

Quando você acessa um arquivo do disquete ou do disco rígido, o **Linux** faz uma cópia deste arquivo e coloca esta cópia na memória **cache**. É com esta cópia que o **Linux** trabalha, pois o acesso à memória é mais rápido que o acesso a uma unidade de disco (rígido ou flexível).

Portanto, quando você modifica o arquivo, o **Linux** modifica, na realidade, a cópia do arquivo armazenada na **cache**, e não o arquivo real. De tempos em tempos, o **Linux** grava os dados da memória nas unidades de disco.

O comando **sync** é utilizado para forçar a atualização destes dados. Por exemplo, se você gravou um arquivo em disquete, antes de removê-lo da unidade, é importante garantir que a cópia realmente seja feita. Para isto basta digitar o comando **sync** (você também pode especificar a opção **sync** durante a **montagem** da unidade de disquete).

tac

tac arquivo...

O comando **tac** imprime os arquivos na tela sendo que cada arquivo é exibido da última a primeira linha.

Observações

Note que o nome **tac** corresponde a **cat** lido de trás para frente. O comando **tac** funciona de forma similar ao comando **cat**, a principal diferença entre os dois comandos se encontra na ordem de leitura do conteúdo de cada arquivo.

tail

tail [opções] arquivo

São algumas das opções deste comando

-c num[bkm] : mostra os **num b**ytes, **k**bytes ou **m**bytes finais do arquivo (o padrão é bytes).

-f : continua indefinidamente tentando ler caracteres ao final do arquivo, assumindo que o arquivo está crescendo. O comportamento é semelhante ao comando **less +F**.

-n num : mostra as **num** últimas linhas do arquivo (o padrão é 10 linhas).

Comentários sobre as opções do comando

Para examinar arquivos que estão sendo constantemente atualizados use a opção **-f**. Por exemplo,

tail -f /var/log/messages

mostra as últimas linhas do arquivo de mensagens do sistema que é modificado pelo **syslog** (veja arquivo **/etc/syslog.conf** para mais detalhes).

talk

talk usuário@host

Observações

Talk é um programa de comunicação visual que permite que dois usuários compartilhem a mesma tela. O texto digitado por um usuário é visualizado pelo outro usuário (corresponde a um **chat**para duas pessoas).

tar

tar [opções] arquivo_tar expressão

São algumas das opções deste comando

-c : cria um novo arquivo **tar**.

-t : lista o conteúdo do arquivo **tar**.

-x : extrai o conteúdo do arquivo **tar**.

-v : mostra mensagens.

-f arquivo : define o nome do arquivo **tar**.

z : filtra os arquivos através de **gzip**.

Comentários sobre as opções do comando

Por exemplo, suponha que você queira copiar todos os arquivos **txt** de um diretório para um único arquivo chamado **colecao_txt**. Para isto, basta você digitar:

tar cvf colecao_txt *.txt

Para verificar o conteúdo do arquivo **colecao_txt**, digite

tar tvf colecao_txt

Para extrair todos arquivos de **colecao_txt**, digite

tar xvf colecao_txt

Para extrair apenas um determinado arquivo de **colecao_txt**, digite

tar xvf colecao_txt nome_arquivo

Observações

A extensão **.tar** não é obrigatória para os arquivos gerados com o comando **tar**, mas é aconselhável. O uso desta extensão facilitará a identificação do tipo do arquivo (você também pode usar o comando **file** para verificar o tipo do arquivo).

Os arquivos que possuem a extensão **.tar.gz** podem ser descompactados e extraídos com as opções **xzvf** do comando **tar**. Isto corresponde a usar o comando **gunzip** para descompactar o arquivo **tar** e depois usar o comando **tar xvf** para extrair os arquivos.

A palavra **tar** significa **Tape ARchiving**.

teclas especiais

TAB

A tecla **TAB** completa a digitação dos comandos e/ou aplicativos do sistema. Por exemplo, digite **makew** e tecle **TAB**. O Linux irá completar o comando para **makewhatis** (caso exista este comando no seu sistema). Se você digitar apenas **make** e pressionar duas vezes a tecla **TAB**, o Linux fornecerá a lista dos comandos e aplicativos que começam por **make**.

CTRL+ALT+(+ do teclado numérico) ou CTRL+ALT-(- do teclado numérico)

Se você configurou mais de uma resolução da tela dentro do **X-Window**, é possível alternar entre estas configurações usando as teclas **CTRL+ALT+** ou as teclas **CTRL+ALT-**.

CTRL+ALT+DEL

Ao pressionar as teclas **CTRL+ALT+DEL**, em conjunto, o sistema será reinicializado. Para evitar que qualquer usuário possar reinicializar o sistema, deve-se criar o arquivo**/etc/shutdown.allow** com o nome dos usuários que possuem autorização para *rebootar* a máquina. Para inibir o uso destas teclas deve-se comentar a seguinte linha no arquivo **/etc/inittab**

ca::ctrlaltdel:/sbin/shutdown -t3 -r now

ALT+Fn

As teclas **ALT+Fn** permite a navegação, no **modo texto**, entre as **consoles virtuais** da máquina, onde **Fn** é uma tecla de função. Por exemplo, **ALT+F1** mostra a console virtual 1 e **ALT+F3**mostra a console virtual 3.

CTRL+ALT+Fn

O conjunto de teclas **CTRL+ALT+Fn** permite a navegação, no **modo gráfico**, entre as **consoles virtuais** da máquina, onde **Fn** é uma tecla de função. Por exemplo, **CTRL+ALT+F1** mostra a console virtual 1 e **CTRL+ALT+F3** mostra a console virtual 3.

tee

tee [opções] [arquivo]

onde **arquivo** é o nome do arquivo de saída onde os dados de entrada serão gravados (além de serem também exibidos na saída padrão).

São algumas das opções deste comando

-a : adiciona a entrada padrão no final do **arquivo** especificado.

-i : ignora os sinais de interrupção.

Comentários sobre as opções do comando

O comando

tee teste.txt

faz com que toda entrada digitada pelo usuário seja ecoada na tela e gravada no arquivo **teste.txt**. Para acrescentar os novos dados de entrada no final do arquivo **teste.txt**, sem perder as informações que este arquivo já possui, digite

tee -a teste.txt

É possível usar o **PIPE** (representado pelo símbolo **|**) para redirecionar a saída de algum comando para o **tee**. Por exemplo,

cat teste.txt | tee teste2.txt

exibe o conteúdo do arquivo **teste.txt** na tela além de copiá-lo para o arquivo **teste2.txt** (para saber mais sobre o **PIPE**, veja **redirecionadores de E/S**).

Observações

Caso o nome de um arquivo não seja definido junto com o comando **tee**, o sistema exibirá os dados apenas na saída padrão.

telinit

telinit [-t seg] [0123456abcSsQqUu]

O **telinit** recebe um caractere como argumento e sinaliza ao **init** (processo com ID 1) para executar a ação apropriada.

São algumas das opções deste comando

-t seg : espera **seg** segundos para executar o comando. O padrão é 5 segundos.

0, **1**, **2**, **3**, **4**, **5** ou **6** : faz o **init** mudar para o nível de execução especificado.

a, **b** ou **c** : faz o **init** executar a entrada no **/etc/inittab** correspondente ao nível de execução especificado.

S ou **s** : faz o **init** entrar em modo monousuário.

Q ou **q** : faz o **init** reexaminar o arquivo **/etc/inittab**.

U ou **u** : faz o **init** se reexaminar, sem verificar o arquivo **/etc/inittab**. Um nível de execução (1-5,Ss) deve ser informado junto com esta opção.

Observações

O arquivo **/sbin/telinit** é um link simbólico do arquivo **/sbin/init**. Portanto, também pode-se usar o comando **init** diretamente.

telnet

telnet [opções] host

onde **host** é o nome DNS (ou endereço IP) com quem será feita a conexão.

São algumas das opções deste comando

-a : tenta login automático.

-l usuário : faz a conexão usando **usuário** como identificação.

Comentários sobre as opções do comando

Por exemplo, o comando

telnet -l aluno 10.1.2.3

solicita uma conexão como o usuário **aluno** na máquina **10.1.2.3**.

tempo de época

O tempo de época para o UNIX é **01/01/1970, 00:00:00, GMT**.

tex

Geração de documentos

O uso do **tex** para a geração de documentos pode ser dividido em quatro etapas:

1. O uso de um editor de texto qualquer (por exemplo, **emacs**, **pico**, **vim**, etc) para digitar o documento com extensão **.tex**. O arquivo criado possui macros que definem como o texto deverá ser formatado.
2. Geração do arquivo com extensão **.dvi** com o comando **latex**
3. Conversão do arquivo **.dvi** em um arquivo PostScript (extensão **.ps**) com o comando **dvips**
4. visualização do arquivo com extensão **.ps** com o comando **gv** (ou **ghostscript**).

Exemplo

Abaixo mostramos um pequeno exemplo de um arquivo com comandos **tex**. Digite este arquivo e salve-o com o nome de **teste.tex**.

```
\documentstyle[12pt]{article}
\begin{document}
\section{Introdu\c c\~ao}
\label{sec:introducao}
ATM ({\em Asynchronous Transfer Mode}) \'e considerado um ambiente
flex\'{\i}vel e eficiente para tr\'afego multim\'{\i}dia. Um
importante servi\c co solicitado \'e a especifica\c c\~ao do retardo
fim-a-fim para as c\'elulas de uma conex\~ao.
\end{document}
```

O primeiro comando do arquivo acima define o tipo de documento a ser formatado: é um artigo e a letra padrão do texto é de tamanho 12. O comando seguinte informa o início do documento. A terceira linha indica o início de uma seção do artigo com o título **Introdução**. O comando **\em** na quarta linha indica que a frase entre chaves terá formato itálico. A última linha do exemplo informa o fim do documento. Note que a acentuação no documento é feita sempre com uma barra invertida ("\") seguida do acento e da letra a ser acentuada. Além disso, a cedilha é definida pelos caracteres **\c c** (não esqueça o espaço entre os dois caracteres **c**).

Agora, gere o arquivo **teste.dvi** com o comando

```
latex teste.tex
```

Para converter este arquivo para o formato **PostScript** digite

```
dvips -o teste.ps teste
```

Para visualizar o texto, basta digitar

```
gv teste.ps
```

time

time comando/programa

Exemplo

O comando

time ls -al /etc

mostra o tempo gasto para executar o comando **ls -al** para o diretório **/etc**.

top

top [opções]

São algumas das opções deste comando

-c : mostra a linha de comando ao invés do nome do programa.

-d num : atualiza a tela após **num** segundos (o padrão é 5 segundos).

-s : execução deve ser feita em modo seguro.

São exemplos de teclas que você pode usar no aplicativo

espaço : atualiza imediatamente a tela.

h : ajuda.

k : encerra um processo (comando **kill**). Será solicitado o **PID** do processo e o sinal a ser enviado ao processo. O sinal 15 provoca o término normal do processo, enquanto o sinal 9 provoca o término forçado do processo. O padrão é o sinal 15 (**SIGTERM**).

M : ordena processos de acordo com o uso da memória.

n ou **#** : altera a quantidade de processos a serem apresentados na tela. É solicitada a entrada de um número. O valor padrão é zero que corresponde ao número de processos que a tela pode suportar.

P : ordena processos de acordo com o uso de CPU.

q : encerra o aplicativo.

r : altera a prioridade de um processo (comando **renice**). Será solicitado o **PID** do processo e o valor da nova prioridade a ser usada pelo processo (o valor padrão é 10). O usuário comum só pode informar um valor positivo maior que a prioridade atual. O **root** pode informar qualquer valor entre **-20** (maior prioridade) e **19** (menor prioridade).

s : altera o tempo entre as atualizações de tela. É solicitada a entrada do tempo de espera em segundos. O valor zero corresponde a atualização contínua. O padrão é 5 segundos.

T : ordena processos de acordo com o tempo de execução.

u : exibe os processos de um determinado usuário. É solicitada a entrada do nome (login) do usuário. O padrão é branco (todos os usuários).

Exemplo

O comando

top -d 10

inicializa o aplicativo **top** e atualiza as informações apresentadas a cada 10 segundos.

touch

touch [opções] arquivo

Comentários sobre as opções do comando

A opção **-t** altera a data e a hora do último acesso/modificação de um arquivo. Por exemplo, para alterar a data e hora do último acesso do arquivo **teste.txt** para 01/05 e 12:31, respectivamente, digite

touch -t 05011231 teste.txt

Note que o parâmetro do exemplo acima define mês, dia, horas e minutos. Podemos também usar o comando **touch** para modificar o ano.

touch -t 9805011231 teste.txt
touch -t 1005011231 teste.txt

O primeiro comando acima modifica o ano para 1998, enquanto o segundo comando modifica o ano para 2010.

Observações

Caso o arquivo não exista, o comando **touch** cria o arquivo vazio com a data especificada (ou com a data do dia). Por exemplo, podemos criar o arquivo vazio **teste.txt** com a data atual, apenas digitando

touch teste.txt

Para atualizar as datas de acesso de todos os arquivos do diretório corrente, digite
touch *

tr

tr [-d] expr1 [expr2]

onde

- **-d** : remove qualquer caractere informado em **expr1**.
- **expr1** : conjunto de caracteres que serão alterados/deletados pelo comando **tr**.
- **expr2** : conjunto de caracteres que substituirão os caracteres definidos em **expr1**.

Comentários sobre o comando

O comando

tr ajx MNP

altera os caracteres **a**, **j** e **x** para **M**, **N** e **P**, respectivamente. Na realidade, o comando acima abre um modo de edição onde o usuário pode digitar textos. O texto digitado pelo usuário é então alterado pelo comando **tr**. Para sair do modo de edição, use **CTRL+C**.

Para definir um conjunto de caracteres com o comando **tr**, use o hífen. Por exemplo,

tr a-z A-Z < teste

exibe, na saída padrão, o arquivo **teste** substituindo as letras minúsculas pelas letras maiúsculas (para mais detalhes sobre o símbolo **<**, veja **redirecionadores de E/S**).

Também é possível eliminar caracteres com o comando **tr**. Por exemplo,

tr -d a-cz < teste

elimina os caracteres **a**, **b**, **c** e **z** ao exibir o conteúdo do arquivo **teste**.

tree

tree [opções] diretório

Este comando tem a mesma função do comando **ls**. A diferença consiste na maneira como as informações são mostradas.

São algumas das opções deste comando

-d : lista somente os subdiretórios.

-a : lista todos os arquivos, inclusive os arquivos ocultos.

-f : exibe o caminho completo dos arquivos.

-p : exibe as permissões dos arquivos.

Observações

Caso o usuário não forneça o nome do diretório, assume-se o diretório atual.

É possível que o seu sistema não possua o comando **tree**. Isto significa que, para usá-lo, você deverá fazer a instalação manualmente. Na distribuição **Conectiva 5.0**, por exemplo, você deve montar o CD 1 e depois instalar o pacote correspondente com os seguintes comandos:

```
mount /mnt/cdrom
rpm -ivh /mnt/cdrom/conectiva/RPMS/tree-1.2-8cl.i386.rpm
umount /mnt/cdrom
```

Para maiores detalhes, veja os comandos **mount**, **rpm** e **umount**.

type

type arquivo

Comentários sobre o comando

Para ver qual o tipo do arquivo **teste**, basta digitar

type teste

Observações

O comando **type** é um comando interno do **shell**. Portanto, para ver a lista de parâmetros deste comando, digite

help type

Outra forma de identificar o tipo de um arquivo é usando o comando **file**.

UID

Definição

A sigla **UID** significa **User IDentification**.

O Linux utiliza o número de **UID** para monitorar os usuários e para verificar as permissões desses usuários.

Observações

O número de **UID** zero é especial sob o Linux pois é o número do usuário **root** ou **superusuário**. Este número tem acesso irrestrito ao sistema Linux.

O **root** (administrador do sistema) pode usar os comandos **useradd**, **usermod**, **userdel** e **id** para, respectivamente, criar usuários, modificar usuários, deletar usuários e identificar **UID** de usuários.

umask

Definição

Quando o usuário cria um arquivo (diretório), o sistema associa ao objeto criado um conjunto de **permissões de acesso**. Estas permissões indicam quem pode ler, alterar e/ou executar (acessar) o arquivo (diretório).

Por padrão,

- as permissões iniciais de um arquivo são **666** (leitura e gravação para todo e qualquer usuário do sistema);
- as permissões iniciais de um diretório são **777** (leitura, gravação e acesso para todo e qualquer usuário do sistema).

Quando um usuário cria um arquivo (ou diretório), o sistema associa a este arquivo (diretório) as permissões padrão menos o valor do **umask**.

Comentários sobre o comando

Para verificar a configuração atual de **umask**, basta digitar na linha de comando

umask

Para alterar o valor de **umask**, basta incluir o novo valor após o comando **umask**. Por exemplo, o comando

umask 002

define que, por padrão, novos arquivos terão permissão **664**, enquanto novos diretórios terão permissão **775**.

Suponha que o usuário **aluno** tenha definido o **umask** como 002. Portanto, quando este usuário cria o arquivo **teste**, temos as seguintes permissões

-rw-rw-r-- 1 aluno aluno 0 May 28 17:20 teste

que corresponde ao valor **664** (666 - 002).

Comandos relacionados

Para saber mais sobre a definição das permissões em modo octal, veja o comando **chmod**.

umount

umount [opções] dispositivo / ponto de montagem

São algumas das opções deste comando

-a : desmonta todos os sistemas de arquivos especificados em **/etc/mtab**.

-r : no caso da desmontagem falhar, tenta remontar apenas para leitura.

-v : exibe mensagens durante o processo de desmontagem da partição.

Comentários sobre as opções do comando

Por exemplo, para desmontar o sistema de arquivos de um disquete, montado no diretório **/mn/floppy**, basta digitar

umount /dev/fd0

ou

umount /mnt/floppy/

Observações

Veja o comando **mount** para saber mais sobre a montagem de **sistemas de arquivos**.

uname

uname [opções]

São algumas das opções deste comando

-a : exibe todas as informações sobre o sistema.

-m : exibe o tipo de máquina (hardware).

-n : exibe o nome de rede da máquina.

-r : exibe a versão do sistema operacional.

-s : exibe o nome do sistema operacional.

-v : exibe a data de compilação do sistema operacional.

Comentários sobre as opções do comando

O comando

uname -a

pode apresentar uma saída semelhante a

Linux localhost.domain 2.2.17-14cl #1 qui nov 2 00:24:54 EST 2000 i686 unknown

onde temos o nome do sistema operacional, o nome de rede da máquina, a versão e a data de compilação do sistema operacional e o hardware da máquina.

Observações

Caso nenhuma opção seja definida, apenas o nome do sistema operacional é apresentado (equivale a opção **-s**).

unique

unique [opções] [arquivo]

São algumas das opções deste comando

-c : exibe o número de ocorrências de cada linha do arquivo.

-d : exibe apenas as linhas com mais de uma ocorrência onde cada linha é mostrada apenas uma vez.

-D : exibe todas as linhas com mais de uma ocorrência. Se uma determinada linha possui duas ocorrências, ela é exibida duas vezes.

-i : ignora a diferença entre letras maiúsculas e letras minúsculas.

-u : exibe apenas as linhas que possuem uma única ocorrência.

Comentários sobre as opções do comando

Suponha que o arquivo **teste** possui o seguinte conteúdo:

aaaa
aaaa
AAAA
bbb
bbbbb
cccc
CCCC

O comando

uniq -cd teste

apresenta a seguinte saída

2 aaaa

Enquanto o comando

uniq -cdi teste

mostra

3 aaaa
2 cccc

No primeiro comando, não é feita a diferença entre letras minúsculas e letras maiúsculas e portanto, temos apenas duas linhas idênticas. No segundo comando, com o uso da opção **-i**, é ignorada a diferença entre caixa alta e caixa baixa e portanto, temos dois conjuntos de linhas, o primeiro com 3 ocorrências e o segundo com 2 ocorrências.

Para ver as linhas do arquivo **teste** que possuem uma única ocorrência e sem considerar a diferença entre letras minúsculas e letras maiúsculas, digite

uniq -ui teste

O sistema apresentará a seguinte resposta:

bbb
bbbbb

Observações

Caso o nome de um arquivo não seja definido junto com o comando **uniq**, o sistema usará a entrada padrão para receber os dados.

unset

unset variável

Comentários sobre o comando

Por exemplo, o comando

unset TESTE

apaga a **variável de ambiente** **TESTE**.

Observações

Para criar uma variável de ambiente, use o comando **export**.

updatedb

updatedb

Observações

O banco de dados de nomes de arquivos é chamado de **locatedb** e fica localizado em **/var/lib**.

Para pesquisar nomes de arquivos no banco de dados deve-se usar o comando **locate**.

uptime

Este comando mostra a hora atual do sistema, o número de usuários conectados e a carga média do sistema nos últimos 1, 5 e 15 minutos.

Observações

Este comando consulta o arquivo **/var/run/utmp** para verificar quem está atualmente usando o sistema e o diretório **/proc** para obter informações sobre os processos.

adduser ou useradd

adduser [opções] usuário

São algumas das opções deste comando

-d diretório : define o diretório **home** do usuário.

-e mm/dd/yy : data de expiração da conta do usuário.

-g grupo : especifica o **GID** do grupo padrão do usuário.

-G grupo1[,grupo2, ...] : especifica o GID dos outros grupos aos quais o usuário pertence.

-s shell : especifica o **shell** padrão do usuário.

-u uid : especifica o **UID** do usuário.

Comentários sobre as opções do comando

Para criar o usuário **aluno**,basta digitar

adduser aluno

Neste caso, o diretório home do usuário **aluno** é **/home/aluno**. Para definir **/home/meudir** como o diretório **home** do usuário **aluno**, devemos escrever

adduser -d /home/meudir aluno

O **UID** do novo usuário corresponde ao menor número, maior que 500, que ainda não está alocado. O **root**, entretanto, pode definir o **UID** de um determinado usuário usando a opção **-u**. Por exemplo,

adduser -u 600 aluno

aloca o **UID** 600 para o usuário **aluno**. Este número deve ser único a menos que a opção **-o** também seja definida. O comando

adduser -u 600 -o aluno

aloca o **UID** 600 para o usuário **aluno**, mesmo que este **UID** já esteja alocado a outro usuário.

É possível também definir os grupos aos quais o usuário pertence. Por exemplo,

adduser -g 600 -G 500,68 aluno

define que o grupo padrão do usuário **aluno** tem **GID** 600, além disso o usuário também pertence aos grupos com **GID** 500 e 68. Quando um novo usuário é criado e o grupo padrão do novo usuário não é fornecido, o sistema automaticamente cria um novo grupo para este usuário com **GID** igual ao número de **UID**. Caso o **GID** já esteja alocado a outro grupo, o sistema escolhe o menor valor, maior que o valor de **UID**, que esteja disponível.

Descrição

O comando **adduser** cria uma entrada para o usuário no arquivo **/etc/passwd**. Esta entrada tem o nome, a senha (caso seja usado o sistema **shadow** para criptografar as senhas, apenas um**x** ou um ***** é exibido neste campo, a senha criptografada é armazenada no arquivo **/etc/shadow**), **UID** (user identification), **GID** (group identification), informações sobre o usuário (inicialmente está vazio), o

diretório **home** do usuário e o **shell** padrão do usuário. Por exemplo, a linha abaixo mostra a entrada criada em **/etc/passwd** para o usuário **aluno** que tem **UID** e **GID** iguais a 501 e usa o **shell bash** como padrão.

aluno:x:501:501::/home/aluno:/bin/bash

Caso seja criado um novo grupo durante o processo de criação de uma conta de usuário, o comando **adduser** cria uma entrada para o grupo no arquivo **/etc/group**. Esta entrada tem o nome do grupo, a senha criptografada do grupo, o **GID** do grupo e a lista dos usuários que são membros do grupo, apesar de este não ser o grupo padrão destes usuários. Por exemplo, a linha abaixo mostra a entrada criada em **/etc/group** para o grupo **aluno** que tem **GID** igual a 501 e onde o usuário **maria** é membro.

aluno:x:501:maria

É importante observar que o comando **adduser** também cria, automaticamente, o **diretório home** do usuário e coloca, neste diretório, os arquivos que o **shell** precisa para inicializar a conta do usuário. Os arquivos que são colocados automaticamente pelo sistema na conta do usuário podem ser vistos no diretório **/etc/skel**.

Observações

A configuração padrão usada pelo comando **adduser** é definida em **/etc/default/useradd** e em **/etc/login.defs**.

userdel

userdel [-r] usuário

Observações

Este comando deleta todas as entradas relacionadas ao usuário especificado nos seguintes arquivos : **/etc/passwd**, **/etc/shadow** e **/etc/group**.

O uso da opção **-r** faz com que o sistema delete o diretório **home** do usuário.

usermod

usermod [opções] usuário

São algumas das opções deste comando

-d diretório [-m] : cria um novo diretório **home** para o usuário. A opção **-m** faz com que o diretório atual do usuário seja movido para o novo diretório.

-e mm/dd/yy : altera a data de expiração da conta do usuário.

-g grupo : altera o **GID** do grupo padrão do usuário para o valor especificado.

-G grupo1[,grupo2, ...] : define o GID dos outros grupos aos quais o usuário pertence.

-l nome : altera o nome de identificação do usuário (o usuário não pode estar logado).

-s shell : altera o **shell** do usuário.

-u uid : altera o número de **UID** do usuário.

Comentários sobre as opções do comando

Por exemplo, o comando

usermod -s ash aluno

faz com que o **shell** padrão do usuário **aluno** passe a ser o **ash**.

Observações

Este comando altera (se necessário) as entradas relacionadas ao usuário especificado nos seguintes arquivos: **/etc/passwd**, **/etc/shadow** e **/etc/group**.

users

users [options]

São opções deste comando

--help : exibe ajuda para este comando.

--version : exibe a versão do aplicativo users.

Comentários sobre as opções do comando

O comando

users

fornece a lista dos usuários atualmente logados no sistema.

variáveis de ambiente

Escopo das Variáveis

O usuário pode alterar o conteúdo das variáveis, remover variáveis existentes e/ou criar novas variáveis de ambiente. Entretanto, as alterações feitas por um usuário não afetarão os outros usuários do sistema. Além disso, estas alterações podem ser permanentes (continuam valendo no próximo acesso do usuário) ou podem ser temporárias.

As definições/alterações feitas no ambiente do usuário sobrepõem as definições feitas pelo sistema. Um exemplo disto está na definição do **prompt** da linha de comando. Existe uma variável de ambiente (**PS1**) que define o formato do **prompt**. O adminstrador (**root**) pode definir um **prompt** padrão para todos os usuários do sistema, mas cada usuário pode customizar esta variável localmente.

Observações

As variáveis de ambiente definidas pelo sistema, para todos os usuários, estão em **/etc/profile**.

As variáveis de ambiente específicas de um usuário são definidas em um arquivo no diretório raiz do usuário (caso o usuário use o **shell** **bash**, este arquivo é o **.bash_profile**).

Comandos relacionados

- **export** - cria variáveis de ambiente.
- **unset** - remove variáveis de ambiente.

vigr

Comentários sobre o aplicativo

O editor padrão de **vigr** é o **vim**. Para alterar o editor padrão deve-se alterar (ou definir, caso não exista) a **variável de ambiente** **EDITOR**. Por exemplo, o comando

export EDITOR=pico

configura o **pico** como editor padrão do **vigr**.

Observações

É importante notar que o comando **vigr** apenas edita o arquivo **/etc/group** e permite que o **root** altere o seu conteúdo. Nenhuma verificação de consistência é feita pelo aplicativo. Portanto, é da responsabilidade do **root** garantir que a alteração feita corresponde a realidade.

vim

vim [opções] [arquivo]

São algumas das opções deste aplicativo

-b : permite editar arquivo binário.

-h : exibe opções do aplicativo.

+n : inicializa o cursor na **n**-ésima linha.

-r : recupera a última versão do arquivo (o arquivo estava sendo editado quando o **vim** foi encerrado de forma não normal, por exemplo, devido a falta de energia elétrica).

-R : abre arquivo apenas para leitura.

Observações

O **vim** (VI Improvement) é uma melhoria do antigo editor de texto **vi**. Este por sua vez é uma melhoria do editor de texto orientado a linha chamado **ed**. Existe também uma versão do **vim** para ambiente **X** chamada **gvim**.

O **vim** possui três formas de trabalho: **modo de linha**, **modo de edição** e **modo de comandos**. A mudança de um modo para outro modo é feita através do uso da tecla **Esc**.

Após o arquivo ser aberto pelo **vim**, o modo de comandos é ativado. No **modo de comandos**, as teclas digitadas pelo usuário são interpretadas pelo **vim** como ações a serem executadas dentro do arquivo aberto. No **modo de edição**, as teclas digitadas pelo usuário são ecoadas na tela. Para entrar neste modo, pode-se digitar, por exemplo, "a" (adicionar), "i" (incluir),etc. No**modo de linha**, o usuário define ações a serem executadas no arquivo como um todo (por exemplo, salvar, substituir caracteres, sair do aplicativo, etc). Para entrar neste modo, deve-se digitar ":".

São alguns dos comandos do vim no modo de comandos

0 : mover o cursor para o início da linha em que o cursor está posicionado.

a : inserir texto após a posição atual do cursor.

A : inserir texto no final da linha atual.

dd : deletar linha atual.

[n]+dd : deletar **n** linhas a partir da linha atual.

G : ir para o fim do arquivo.

[n]+G : ir para a **n**-ésima linha do arquivo.

h : voltar um caractere.

H : ir para a primeira linha exibida na tela atual.

i : inserir texto a partir da posição atual do cursor.

I : inserir texto no início da linha atual.

j : descer uma linha.

J : juntar a linha atual com a linha seguinte.

[n]+J : juntar **n** linhas consecutivas a partir da linha atual.

k : subir uma linha.

l : avançar um caractere.

L : ir para a última linha exibida na tela atual.

n : procurar, a partir da posição atual do cursor, a próxima ocorrência do texto definido no último comando **/**.

N : procurar, a partir da posição atual do cursor e indo em direção ao início do arquivo, a próxima ocorrência do texto definido no último comando **/**.

o : inserir uma linha em branco após a linha atual.

O : inserir uma linha em branco acima da linha atual.

p : inserir linhas copiadas após a linha atual.

P : inserir linhas copiadas antes da linha atual.

r : substituir o caractere atual.

R : substituir um conjunto de caracteres.

s : deletar o caractere atual e inserir texto.

S : apagar linha e inserir novo texto na linha.

u : desfazer a última alteração feita no texto e ainda não desfeita.

U : desfazer a última alteração feita no texto.

x : apagar caractere onde o cursor está posicionado.

$: mover o cursor para o fim da linha em que o cursor está posicionado.

[n]+y : copiar **n** linhas a partir da linha atual.

yy : copiar a linha atual.

[n]+Y : copiar **n** linhas a partir da linha atual.

YY : copiar a linha atual.

CTRL+B : voltar uma página.

CTRL+F : avançar uma página.

F1 : exibir tela de ajuda.

[n]+ENTER : ir para **n** linhas abaixo da linha atual.

[n]+. : repetir o último comando que alterou o texto **n** vezes a partir da posição atual do cursor.

[n]+~+ENTER : inverter a caixa (**case**) dos **n** caracteres seguintes ao cursor.

/texto : procurar pela primeira ocorrência do texto especificado a partir da posição atual do cursor.

São alguns dos comandos do vim no modo de linha

:r arquivo : incluir arquivo a partir da linha atual do cursor.

:q+ENTER : sair da tela de ajuda.

:q! : sair do **vim** sem salvar as alterações.

:w arquivo : salvar arquivo com o nome especificado.

:wq : sair do **vim** salvando as alterações.

:X : criptografa o arquivo.

vipw

Comentários sobre o aplicativo

O editor padrão de **vipw** é o **vim**. Para alterar o editor padrão deve-se alterar (ou definir, caso não exista) a **variável ambiente** **EDITOR**. Por exemplo, o comando

export EDITOR=pico

configura o **pico** como editor padrão do **vipw**.

Observações

É importante notar que o comando **vipw** apenas edita o arquivo **/etc/passwd** e permite que o **root** altere o seu conteúdo. Nenhuma verificação de consistência é feita pelo aplicativo. Portanto, é da responsabilidade do **root** garantir que a alteração feita corresponde a realidade.

visudo

Comentários sobre o aplicativo

O editor padrão de **visudo** é o **vim**. Para alterar o editor padrão deve-se alterar (ou definir, caso não exista) a **variável ambiente** **EDITOR**. Por exemplo, o comando

export EDITOR=pico

configura o **pico** como editor padrão do **visudo**.

Observações

É importante notar que o comando **visudo** apenas edita o arquivo **/etc/sudoers** e permite que o **root** altere o seu conteúdo. Nenhuma verificação de consistência é feita pelo aplicativo.

w

w [opções]

São algumas das opções deste comando

-h : não mostra o cabeçalho das colunas.

-l : listagem longa (completa); é o padrão.

-s : listagem curta.

-u : não mostra o **UID** dos processos.

Comentários sobre as opções do comando

Por exemplo, o comando

w -s

exibe uma listagem curta dos usuários do sistema e o que estão fazendo.

wc

wc [opções] arquivo

São opções deste comando

-c : conta caracteres.

-l : conta linhas.

-L : mostra o comprimento da linha mais longa.

-w : conta palavras.

Observações

Caso nenhuma opção seja informada, o sistema informa o número de linhas, de palavras e de caracteres do arquivo especificado.

whatis

whatis comandos

Comentários sobre o comando

Este comando procura palavras-chaves em um banco de dados que possui pequenas descrições dos comandos do sistema (este banco de dados é criado pelo comando **makewhatis**). Por exemplo,

whatis man ls

pode ter uma saída semelhante a definida abaixo.

man (1) - format and display the on-line manual pages
man (7) - macros to format man pages
man.conf (5) - configuration data for man
ls (1) - list directory contents

whereis

whereis [opções] nome

São algumas das opções deste comando

-b : lista apenas arquivos binários.

-m : lista apenas arquivos de documentação.

-s : lista apenas arquivos fontes.

Comentários sobre as opções do comando

Por exemplo,

whereis shells

pode fornecer a seguinte saída

shells: /etc/shells /usr/man/man5/shells.5.gz

onde **/etc/shells** é um arquivo binário com os nomes dos **shells** que o usuário pode usar
e **/usr/man/man5/shells.5.gz** é a documentação sobre os **shells**.

who

who [opções]

São algumas das opções deste comando

-H : mostra o cabeçalho das colunas.

-i, -u : após um tempo de acesso, mostra o tempo que o usuário esteve inativo (hh:min). O valor '**.**' significa que o usuário esteve inativo no último minuto. O valor '**old**' significa que o usuário esteve inativo nas últimas 24 horas.

-m : mostra o nome do usuário.

-q : mostra os nomes e a quantidade total de usuários conectados.

Comentários sobre as opções do comando

Por exemplo, o comando

who -m

identifica o usuário que está logado no terminal e é idêntico ao comando **whoami**.

whoami

whoami

Observações

Este comando é idêntico ao comando **who -m**.

X Window

Definições

O pacote **XFree86**, também conhecido como **X11R6** (versão 11 release 6), fornece todos os arquivos para a instalação do sistema **X Window** tais como binários, bibliotecas e utilitários.

O aplicativo **Xconfigurator** é utilizado para configurar o servidor **XFree86** em várias distribuições Linux. Isto significa que dispositivos como placa de vídeo e monitor podem ser configurados com este aplicativo.

É importante notar que o **sistema X** gerencia uma única janela por vez. Para que um usuário possa ter acesso a várias janelas dentro de um mesmo **login**, é necessário utilizar um gerenciador de janelas. São exemplos de gerenciadores de janelas: **gnome**, **kdm**, **windowmaker**, **xdm**, etc.

Observações

Na realidade, o **Xconfigurator** apenas altera o arquivo **XF86Config** que fica armazenado em **/usr/X11R6/lib/X11** ou em **/etc/X11**.

No sistema X, é possível definir onde uma aplicação gráfica deve aparecer (em que máquina). Para isto basta alterar o conteúdo da variável **DISPLAY**.

Pode-se usar o programa **xhost** para definir quais usuários e máquinas podem ter acesso ao servidor X.

xhost

xhost [[+-]nome...]

Definição

O programa **xhost** é usado para adicionar ou deletar nomes de máquinas e de usuários à lista dos que possuem autorização para acessar o servidor **X**.

Por exemplo, é preciso ter acesso ao servidor X para poder redirecionar a saída de uma aplicação gráfica (veja variável de ambiente **DISPLAY** para mais detalhes).

São algumas das opções deste comando

+ : acesso é garantido a todos os usuários.

- : acesso é garantido apenas aos usuários que estão na lista.

+nome : o nome especificado (máquina ou usuário) é adicionado a lista garantido assim acesso ao servidor X.

-nome : o nome especificado (máquina ou usuário) é retirado da lista que permite acesso ao servidor X.

Se nenhum parâmetro é fornecido com o comando **xhost**, o sistema apenas informa se o controle de acesso ao servidor X está sendo feito ou não e exibe a lista dos usuários que estão conectados.

Observações

O arquivo **/etc/ssh/ssh_config** possui o parâmetro **ForwardX11** que define se as conexões X11 devem ser automaticamente redirecionadas. Por padrão temos

ForwardX11 no

Pode-se portanto alterar este parâmetro para

ForwardX11 yes

O uso da definição acima implica que não há mais necessidade de utilizar o comando **xhost** pois as conexões X11 são automaticamente redirecionadas.

Caso você altere o arquivo **/etc/ssh/ssh_config**, o daemon do servidor **sshd** deve ser reinicializado com o seguinte comando:

sshd restart

xhost

xhost [[+-]nome...]

Definição

O programa **xhost** é usado para adicionar ou deletar nomes de máquinas e de usuários à lista dos que possuem autorização para acessar o servidor **X**.

Por exemplo, é preciso ter acesso ao servidor X para poder redirecionar a saída de uma aplicação gráfica (veja variável de ambiente **DISPLAY** para mais detalhes).

São algumas das opções deste comando

+ : acesso é garantido a todos os usuários.

- : acesso é garantido apenas aos usuários que estão na lista.

+nome : o nome especificado (máquina ou usuário) é adicionado a lista garantido assim acesso ao servidor X.

-nome : o nome especificado (máquina ou usuário) é retirado da lista que permite acesso ao servidor X.

Se nenhum parâmetro é fornecido com o comando **xhost**, o sistema apenas informa se o controle de acesso ao servidor X está sendo feito ou não e exibe a lista dos usuários que estão conectados.

Observações

O arquivo **/etc/ssh/ssh_config** possui o parâmetro **ForwardX11** que define se as conexões X11 devem ser automaticamente redirecionadas. Por padrão temos

ForwardX11 no

Pode-se portanto alterar este parâmetro para

ForwardX11 yes

O uso da definição acima implica que não há mais necessidade de utilizar o comando **xhost** pois as conexões X11 são automaticamente redirecionadas.

Caso você altere o arquivo **/etc/ssh/ssh_config**, o daemon do servidor **sshd** deve ser reinicializado com o seguinte comando:

sshd restart

zgrep

zgrep expressão arquivo

onde

- **expressão** é a palavra ou frase a ser procurada no texto.
- **arquivo** é o nome do arquivo compactado onde será feita a busca.

Observações

Na realidade, este aplicativo é um script de **shell** que serve de interface para o uso do aplicativo **grep** em arquivos compactados. O aplicativo descompacta o arquivo e em seguida chama o**grep** passando todas as opções definidas com o comando pelo usuário.

zip

Comentários sobre o comando

Para compactar um arquivo, digite

zip arquivo_compactado arquivo_original

É adicionado ao nome do arquivo compactado a terminação **.zip**. Portanto, não é necessário definir a extensão do arquivo. Você pode inclusive usar o mesmo nome do arquivo original, pois o arquivo original continua existindo após a compactação.

Para descompactar, digite

unzip arquivo_compactado

Observações

O comando **zip** é análogo à combinação dos comandos **tar** e **compress**. Além disso, este comando é compatível com o **pkzip** e o **winzip** da plataforma DOS/Windows.

Referencia:

http://www.uniriotec.br/~morganna/guia/introd_guia.html