Linux: Utilização

Este documento apresenta uma visão geral sobre o uso de sistemas Linux. Como Linux procura atender padrões do mundo Unix, a maior parte das funções descritas aqui também se estende ou se assemelha ao que ocorre em qualquer sistema Unix.
Isto não é um guia completo (nada substitui o manual). Entretanto, a ideia é que sirva como um roteiro de estudo. Mãos ~~à obra~~ ao teclado!

1. Conectando-se ao sistema: *login*

O acesso a um sistema **Linux** depende da identificação de um nome de usuário e da senha correspondente, num procedimento de ***login***. Enquanto o usuário ***root*** tem **direito total sobre todos os recursos do sistema**, os demais usuários têm acesso para **leitura** de grande parte dos arquivos, podem **executar** a maioria das aplicações, e têm direito de escrita somente sobre seus respectivos diretórios de trabalho (***/home/fulano***) e área temporária (***/tmp***).

Caso o ambiente gráfico tenha sido instalado, acompanhado dos gerenciadores do *display* (xdm, gdm ou kdm), a opção de ***login*** através de uma interface gráfica é selecionada automaticamente. Independentemente de o modo gráfico estar configurado, é possível executar-se o procedimento de ***login*** em modo texto.

1.1 Login em modo texto

No Linux, a combinação de teclas [<ctrl>]<**alt**><**F*n***> (n={1,2,.., 6}) dá acesso a 6 ***terminais virtuais*** onde pode-se executar o procedimento de ***login*** numa interface de texto com acesso direto a um ***shell***.

Um *shell* é um programa **interpretador de comandos**, que tem a finalidade de controlar a ativação e manipulação de programas determinados pelo usuário. Comandos podem ser passados para o *shell* de maneira **interativa** ou através de **programas** (*scripts*).

- Diversas versões de *shell* podem estar instaladas num sistema, sendo listadas no arquivo ***/etc/shells***, sendo os mais comuns: **sh**, **bash**, **csh**, **ksh**, **tcsh**, **zsh** e **dash**. As diferenças entre esses programas podem ser notadas tanto no tratamento dos comandos interativos quanto na "linguagem" usada para escrever os *scripts*. Comandos úteis para a linguagem de *scripts* incluem, por exemplo análise de expressões, iterações, decisões e desvios.

Bash é talvez o ***shell*** mais utilizado atualmente. Informações sobre sua configuração e seus comandos internos podem ser obtidas digitando-se ***help*** diretamente no *prompt*.

OBS 1: há diferenças no uso de letras **maiúsculas** e **minúsculas**. Comandos, em geral, são todos definidos com letras minúsculas. Variáveis de ambiente comumente são maiúsculas.

OBS 2: nas documentações, é comum usar ***$ cmd*** para indicar a execução do comando *cmd* no *shell*. *$* indica o *prompt* de um usuário comum. Comandos que devem ser executados com privilégio de root normalmente são indicados com ***# cmd***.

OBS 3: *$ cmd1 ; cmd2 <enter>* indica dois comandos na mesma linha (pode haver mais). ";" indica separação de comandos na mesma linha.

OBS 4: $ .... \ <*enter*> indica para o *shell* que, embora o comando está se estendendo para uma nova linha, ele não foi concluído ainda. "\" é usado no final da linha nesses casos.

OBS 5: ~ representa a área (HOME) de um usuário. Por exemplo, ***~fulano*** refere-se ao HOME do usuário fulano. ***~/progs*** indica o diretório *progs*, localizado dentro no HOME do usuário atual. Ex: *$ ls ~fulano/pub* // lista o conteúdo do diretório pub, localizado na área (HOME) do usuário *fulano*.

Variáveis de ambiente são utilizadas nos *shells* para armazenar informações úteis sobre o usuário e sobre suas preferências de comandos e outras configurações. Ex: HOME, SHELL, USER, PATH, etc. A sintaxe para suas configurações depende do *shell* utilizado. No ***bash***, é feita como segue:

bash: // exemplo de ajuste do PATH. $ indica o *prompt* do *shell*

```
$ PATH=$PATH:$HOME/bin      // define a variável para o shell
$ export PATH               // exporta variável, tornando-a visível para
                            // programas executados a partir do shell
```

ou

```
$ export PATH=$PATH:$HOME/bin      // definição e exportação
```

Uma lista completa das variáveis de ambiente definidas numa sessão de *shell* pode ser exibida com o comando ***set***. O comando *echo*, que exibe texto no terminal, também pode ser usado para verificar o conteúdo de variáveis de ambiente.

set, ***unset***, ***setenv***, ***unsetenv***, ***export***: funções do *shell* para ler e escrever variáveis de ambiente

echo: exibe uma linha de texto.
Ex: *$ echo teste*; *echo $USER*

Alguns comandos do terminal permitem o ajuste de aspectos da **sessão** do *shell* corrente, incluindo a configuração da interface do terminal de entrada e saída de dados:

stty: exibe e ajusta configurações do terminal. Ex: *$ stty –a*; *stty erase ^H*
clear: limpa a tela do terminal. Ex: *$ clear*

Diversos comandos e recursos do *shell* oferecem facilidades para a ativação de comandos:

history: manipula a lista interna de comandos realizados no *shell*. Algumas variáveis de ambiente definem o seu comportamento. Enquanto a variável de ambiente HISTSIZE

define o tamanho do histórico guardado, HISTFILE e HISTFILESIZE definem o nome e o tamanho do arquivo onde as informações são salvas entre sessões.

$ ***!!*** : repete o último comando digitado
$ ***!num*** : repete o comando número num do histórico
$ ***!cmd*** : repete o último comando iniciado com a palavra ***cmd***.
$ ***^old^new*** : executa o último comando, substituindo a string *old* por *new*

alias: exibe ou ajusta *apelidos* para comandos. ***unalias*** os remove. Ex: *alias rm='rm –i'*
// indica para o *shell* que, ao digitar ***rm*** deve-se executar o comando ***rm -i***

Arquivos específicos presentes na área do usuário (*~/*)e no diretório de configurações do sistema (***/etc***) permitem a configuração das **sessões** de utilização de um *shell*. Há dois tipos de sessão de *shell*, a sessão de ***login***, iniciada a partir do procedimento de *login* texto ou via acesso remoto, e a sessão **interativa**, iniciada numa janela da interface gráfica, por exemplo, ou a partir de outra sessão de *shell*.
Para o programa *bash*, mais utilizado, os seguintes arquivos de configuração são relevantes.

Para um *shell* de *login*:

/etc/profile // configurações definidas pelo root, válidas para todos os usuários
~/.bash_profile // configurações específicas do usuário
~/.bash_login // configurações específicas do usuário
~/.profile // configurações específicas do usuário

Quando esse *shell* de *login* termina, são executados os comandos em *~/.bash_logout*

Para um *shell* interativo:

~/.bashrc // configurações definidas pelo usuário.

Comandos úteis nos arquivos de configuração incluem: ***set, alias***, ***stty***, etc.

Uma sessão de *login* em modo texto é encerrada com o comando *logout*. A combinação de teclas <**ctrl**> <**D**>, normalmente definida como fim de arquivo (EOF) via ***stty***, também pode comumente ser usada para encerrar uma sessão de *login*. *Exit* termina uma sequência de comandos num *shell* ou encerra uma sessão de *login*.

logout: comando interno do *shell* para encerrar uma sessão num computador
exit: função interna do *shell* para encerrar um nível na sequência de operações sendo executadas ou um *shell* interativo. No *shell* de *login* tem o mesmo efeito de *logout*

1.2 Login via Interface gráfica

Efetuando-se o *login* numa interface gráfica, o usuário passa a interagir com o sistema Linux através de um ambiente de janelas. Entre as opções de configuração do ambiente gráfico, que existem às dezenas, as mais comuns são ***GNOME*** e ***KDE***.

Tipicamente, os gerenciadores do ambiente gráfico (*window managers*) utilizam o conceito de janelas associadas às aplicações, direcionando suas operações de entrada e saída de dados. Janelas são dimensionáveis e podem ser sobrepostas, maximizadas e minimizadas. Mecanismos existem para selecionar a janela ativa, que recebe os eventos de teclado e mouse, tipicamente em função da posição do mouse ou do *click* de algum de seus botões. Janelas podem ter bordas variáveis, botões de controle (para minimização, maximização e encerramento da aplicação associada, além de regiões de controle de redimensionamento, tipicamente nos cantos das bordas. Dentro de uma janela é possível ter áreas de apresentação para desenhos e textos, áreas em que o usuário pode fornecer dados, além de botões. De maneira geral, cabe ao desenvolvedor de uma aplicação associada a uma janela determinar as ações associadas aos botões e eventos pré-definidos de controle de janela ou associados aos elementos que a janela contém.

Além de diferirem no formato e nas funcionalidades associadas às janelas, gerentes de interface gráfica possuem elementos gráficos comumente presentes na tela, como barras de controle de aplicações e ícones associados a recursos de armazenamento e dispositivos.

As funcionalidades associadas ao pressionamento dos botões do mouse na área de trabalho também podem variar, tipicamente servindo para oferece acesso a menus para manipulação de janelas ou aplicações.

Normalmente, todo ambiente gráfico costuma ter ao menos uma aplicação associada ao acesso e gerenciamento ao sistema de arquivos (*file manager*), editores de texto, navegadores Web e janelas de terminal associadas ao *shell* definido para a conta do usuário. Uma infinidade de utilitários com interfaces gráficas existem para sistemas Linux e costumam estar disponíveis nas variadas distribuições.

Programas gerenciadores de pacotes (grupos de programas) permitem a adição, remoção e atualização de pacotes, e costumam ser fornecidos pelas distribuições e ambientes gráficos.

Uma particularidade no acesso a sistemas de arquivos de dispositivos removíveis é a necessidade de "desmontar" o sistema de arquivo destes dispositivos antes que eles possam ser removidos com segurança. Isto ocorre pois o sistema operacional costuma manter *buffers* dos arquivos abertos em memória, de forma que nem todos os acessos aos arquivos geram a gravação imediata dos dados nos dispositivos. A montagem do sistema de arquivos em dispositivos é tipicamente feita de forma automática, mas a desmontagem deve ser feita manualmente, forçando a atualização dos dados no dispositivo.

Para encerrar uma sessão gráfica é preciso executar um procedimento de *logout*. Normalmente, há opções em alguma barra de menus que possibilita esse procedimento, além das opções de desligar e reiniciar o computador.

Também por motivos de consistência do sistema de arquivos, é importante selecionar um procedimento de desligamento via software antes de desligar o computador.

2. Obtendo ajuda

A melhor fonte de informações em sistemas Linux/Unix é o manual on-line existente em todos os sistemas! Informações sobre o sistema, comandos utilitários, chamadas de sistema, funções das APIs instaladas, arquivos de configuração e muito mais, são comumente instaladas num sistema Unix na forma de páginas de manual. Digitando-se comandos diretamente em um interpretador de comandos (*shell*), é possível acessar as suas funcionalidades.

man: comando de acesso às páginas do manual *on-line* do sistema.

Seções: (1) Comandos do usuário, (2) Chamadas do sistema, (3) Funções da biblioteca C, (4) Dispositivos e interfaces de rede, (5) Formatos de arquivos, (6) Jogos e demonstrativos, (7) Ambiente de trabalho, tabelas e macros *troff*, (8) Manutenção do sistema.

Páginas do manual são normalmente armazenadas como arquivos texto formatados para exibição com comandos *roff*, depois de descompactados. O arquivo ***/etc/man.config*** (ou ***/usr/lib/man.config***) contém informações sobre os diretórios que contém páginas de manual, o que também pode ser especificado pela variável de ambiente MANPATH.

Ex: *man man*, *man 2 sleep*, *man -a passwd*, *man -k ...*

Daqui para frente, é bom ter ao menos dois *shells* ativos: um para pesquisar o comando com *man*, e outro para experimentá-lo!

Whatis e *apropos* são outros comandos de ajuda, que exibem parte das informações comumente mostradas pelo comando *man*:

whatis: apresenta a descrição de comandos, tipicamente uma parte inicial da página de manual completa. Ex: *whatis ls*

makewhatis: cria a base de dados para o comando *whatis*. É normalmente executado periodicamente, de forma automática, no sistema.

apropos: mostra seções e páginas do manual que contém referências a um comando, pesquisando a base de dados do comando *whatis*

Whereis e *which* ajudam na localização de comandos:

whereis: fornece a localização de um comando executável, tipicamente buscando-o nos diretórios que costumam conter arquivos executáveis na hierarquia do sistema de arquivos Unix. Ex: *whereis man*

which: apresenta o nome (caminho) completo de um comando, caso ele estema definido em algum caminho de busca do *shell* corrente (PATH). Ex: *which ls*

Informações específicas sobre o ***bash*** podem ser obtidas com o comando ***help***, digitado diretamente no *prompt* do ***shell***. Particularmente, vale a pena examinar a relação de

comandos internos (BUILTIN), o tratamento de sinais, a configuração do terminal, entre outros aspectos.

3. Acesso e identificação dos usuários

Uma vez conectado ao sistema, comandos podem ser utilizados pelo usuário para iniciar ou encerrar uma sessão e para identificação deste usuário e de outros usuários conectados ao sistema.

login: permite que um usuário abra uma sessão num computador

id: informa o número de identificação associado à conta do usuário
whoami: mostra o *username* efetivo do usuário corrente
groups: informa os grupos a que pertence o usuário corrente
who: mostra quem está usando o sistema
w: mostra informações sobre os usuários conectados ao sistema
users: mostra uma lista compacta dos usuários conectados ao sistema
last: mostra informações sobre o *login* e o *logout* dos usuários e terminais de acesso utilizados. Ex: *last*, *last root*
finger: programa para pesquisa de informações sobre usuários. Ex: *finger fulano*, *finger fulano@host.dom* (depende de um *servidor finger* estar ativo em *host.dom*)

Arquivos de informação sobre o usuário (consultados pelo comando finger):
~/.plan:
~/.project:

Uma característica importe em sistemas Unix é a possibilidade de alterar-se a identidade de um usuário numa sessão de *shell*. Ambientes gráficos costumam ter recursos equivalentes para permitir que um usuário comum realize certas atividades administrativas que requerem privilégio de super-usuário (*root*). Deste modo, o login a sistemas Unix é comumente feito com o mínimo de privilégio, usando privilégio de usuário comum, e, somente quando um privilégio especial é necessário uma senha adicional é fornecida. Isso favorece a segurança do sistema, normalmente impedindo a atuação de *malwares* ou ações indevidas por usuários inexperientes ou desatentos.

Num *shell*, *su* e *sudo* permitem a alteração dos privilégios de usuário:

su: substitui a identidade do usuário corrente. Sem parâmetros, refere-se ao usuário *root*. Ex: *su* *su – fulano* *su –c comando*
sudo: permite executar um comando como outro usuário. O arquivo */etc/sudoers* contém uma relação dos usuários autorizados a executar comandos específicos. Diferentemente do comando *su*, *sudo* solicita a senha do próprio usuário, ao invés da senha do usuário alvo (*root*, neste caso). Ex: *sudo apt-get update*

passwd: altera a senha de um usuário
yppasswd: altera a senha em um ambiente de rede usando o **NIS** (*Yellow Pages*).
smbpasswd: altera a senha no ambiente do samba ou em um domínio ***Windows***.

4. Informações do sistema

Além das informações sobre usuários, diversos utilitários fornecem informações sobre o sistema operacional, sobre sua configuração e sobre a configuração de dispositivos.

uname: exibe informações sobre o sistema. Ex: *uname –a*
uptime: informa há quanto tempo o sistema está sendo executado
hostname: exibe ou ajusta o nome do computador
domainname: informa ou ajusta o nome de domínio DNS do sistema
date: informa ou ajusta a data e o horário do sistema
dmesg: exibe ou controla o buffer circular de mensagens do *kernel*, tipicamente as mensagens dos programas durante o *boot* e aquelas geradas por controladores de dispositivos e *drivers*. Ex: *dmesg | more*

5. Organização do sistema de arquivos Unix

O sistema de arquivos em ambientes Unix é baseado numa única estrutura hierárquica (árvore) de diretórios. Não existem unidades lógicas tal como em sistemas Windows (C:, D:, por exemplo). Assim, há uma única **raiz** do sistema de arquivos, representada pelo símbolo "/" (barra invertida). Durante o *boot* do sistema, o sistema de arquivos de alguma partição em alguma unidade de armazenamento é **montado** nessa raiz, tornando-se acessível a partir das referências feitas a /xxx. Outros sistemas de arquivos em outras partições em dispositivos locais e remotos, fixos ou removíveis, também precisam ser montados em algum **subdiretório** da árvore do sistema de arquivos montado na raiz. Por exemplo, uma partição separada para conter as áreas de trabalho dos usuários pode ser montada no diretório **/home**. Sistemas de arquivos em mídias removíveis tipicamente são montados automaticamente no subdiretório /media/xxx.

Num *shell*, o comando ***ls*** é utilizado para listar os conteúdos dos diretórios. Usando o parâmetros –la ($ ls –la /), é possível ver informações adicionais sobre o sistema de arquivos raiz e seus subdiretórios.

Para uniformização de sistemas Linux, há uma hierarquia definida para os diretórios a partir da raiz. Chamada de *File System Hierarchy* (FHS – http://www.pathname.com/fhs), essa hierarquia contém os seguintes diretórios principais:

- / – diretório raiz, concentra toda a estrutura de diretórios
- **/etc** – arquivos de configuração
- **/bin** – utilitários de uso geral
- **/sbin** – utilitários de administração, alguns com uso restrito ao usuário ***root***.
- **/dev** – arquivos especiais que representam dispositivos, criados com o comando *mknod* (/dev/MAKEDEV).
- **/usr** – hierarquia secundária
- **/usr/lib** bibliotecas de programas (*linkad*as com programas de usuário: lib*.a e lib*.so)
- **/usr/include** arquivos de definições e protótipos de funções (*header files*: *.h)

- **/lib –** bibliotecas compartilhadas essenciais
- **/home** – área de trabalho dos usuários
- **/var** – diretório para área de *spool* de impressão, e-mails e arquivos de *log*
- **/boot** – arquivos para iniciação do sistema (*boot*) e configurações
- **/mnt** – diretório onde tipicamente são *montados* sistemas de arquivos temporátios
- **/media** – ponto de montagem para mídias removíveis (*cd/dvd, usb*)
- **/tmp** – armazenamento temporário
- **/proc** – sistema de arquivos em memória com informações sobre o sistema e seus processos (específico de sistemas Linux)
- **/opt** – aplicativos não fornecidos com o sistema
- **/srv** – dados de serviços providos por esse sistema

[FHS] The following commands, or symbolic links to commands, are required in `/bin`.

cat	*Utility to concatenate files to standard output*
chgrp	*Utility to change file group ownership*
chmod	*Utility to change file access permissions*
chown	*Utility to change file owner and group*
cp	*Utility to copy files and directories*
date	*Utility to print or set the system data and time*
dd	*Utility to convert and copy a file*
df	*Utility to report filesystem disk space usage*
dmesg	*Utility to print or control the kernel message buffer*
echo	*Utility to display a line of text*
false	*Utility to do nothing, unsuccessfully*
hostname	*Utility to show or set the system's host name*
kill	*Utility to send signals to processes*
ln	*Utility to make links between files*
login	*Utility to begin a session on the system*
ls	*Utility to list directory contents*
mkdir	*Utility to make directories*
mknod	*Utility to make block or character special files*
more	*Utility to page through text*
mount	*Utility to mount a filesystem*
mv	*Utility to move/rename files*
ps	*Utility to report process status*
pwd	*Utility to print name of current working directory*
rm	*Utility to remove files or directories*
rmdir	*Utility to remove empty directories*
sed	*The `sed' stream editor*
sh	*The Bourne command shell*
stty	*Utility to change and print terminal line settings*
su	*Utility to change user ID*
sync	*Utility to flush filesystem buffers*
true	*Utility to do nothing, successfully*
umount	*Utility to unmount file systems*
uname	*Utility to print system information*

6. Utilitários para manipulação de arquivos e diretórios

ls: mostra o conteúdo de diretórios. Ex: *ls –l /home*
pwd: informa o nome do diretório corrente. Ex: *pwd*
cd: muda o diretório de trabalho. Ex: *cd /etc*, *cd ..*, *cd ~/www*, *cd ../../local*
cp: copia arquivos. Ex: *cp /tmp/arq .*, *cp –r dir1 ../dir2*, *cp arq1 arq2*
mv: move ou renomeia arquivos ou diretórios. Ex: *mv /tmp/arq .*, *mv arq novo_nome*
mkdir: cria diretórios. Ex: *mkdir dir*, *mkdir www pub tmp*
rm: remove arquivos. Ex: *rm arq*, *rm –i arq*, *rm –f arq*, *rm –rf diret*
rmdir: remove diretórios. Ex: *rmdir diret*, *rmdir dir1 dir2 /tmp/dir3*
ln: cria um *link* para um arquivo ou diretório. Ex: *ln –s /bin/ls dir*, *ln –s /tmp tmp*
cat: lista o conteúdo de arquivos. Ex: *cat /etc/fstab*
more: filtro de exibição de dados. Ex: *more /etc/fstab*
less: filtro de exibição de dados. Ex: *less /etc/fstab*
cmp: compara 2 arquivos. Ex: *cmp arq1 arq1*
diff: exibe as diferenças entre 2 arquivos texto, linha por linha. Ex: *diff arq1 arq2*
find: percorre uma hierarquia de diretórios. Ex: *find . –name .doc –print*, *find / -name *.jpg –exec rm –f {} \;*
grep: imprime linhas que possuem um padrão especificado. Ex: *grep root /etc/passwd*
file: determina o tipo de um arquivo (texto, binário, script do shell, etc.). Ex: *file /bin/ls*, *file /etc/passwd*
tail: exibe a parte final de um arquivo. Ex: *tail /var/log/messages*, *tail –20 /var/log/messages*
head: exibe as primeiras linhas de um arquivo. Ex: *head /var/log/messages*, *head –20 /var/log/messages*
cut: seleciona partes de uma linha de texto. Ex: *cut –c 10-20 /etc/passwd*, *cut –d: –f 5 /etc/passwd*
wc: contador de palavras, linhas e bytes. Ex: *wc –l /etc/passwd*
sort: ordena linhas de arquivo texto. Ex: *sort arq*, *sort –n –r arq*
touch: altera as datas de acesso e modificação de arquivos. Ex: *touch *.h *.c*
lsof: lista arquivos abertos.

7. Utilitários para agrupamento, conversão e compressão de arquivos

tar: cria arquivos para *tapes* e adiciona ou remove arquivos. Ex: *tar –tvf arq.tar*, *tar –cvf dir.tar dir*, *tar –xvf dir.tar, tar cvzf diret.tgz diret, tar xvzf diret.tgz*
dd: copia arquivos, podendo realizar conversões de formato. Ex: *dd if=bootnet.img of=/dev/fd0*. Pode acessar dispositivos diretamente, sem passar pelo sistema de arquivos, mas requer privilégio de *root*.
cpio: copia arquivos de/para dispositivos de E/S
uuencode* / *uudecode: codifica um arquivo binário para uma representação que pode ser enviada por e-mail. Uudecode decodifica o arquivo.
compress* / *uncompress: comprime e descomprime dados. Ex: *compress log*, *uncompress log.Z*
gzip* / *gunzip: comprime ou expande arquivos. Ex: *gzip arq.ext*, *gunzip arq.ext.gz*

zip / **un*z*ip**: empacota e comprime / descomprime arquivos. Ex: *zip arq.ext*, *unzip arq.ext.zip*

bzip2 / ***bunzip2***: comprime / descomprime arquivos. Ex: *bzip2 arq.ext*, *bunzip2 arq.ext.bz2*

8. Gerenciamento de partições e sistemas de arquivo

Assim como em outros sistemas operacionais, Linux permite a manipulação e o acesso a diversas partições nos discos. O utilitário ***fdisk*** é utilizado para essas manipulações. O nome do dispositivo a ser manipulado é especificado como parâmetro (e.g. ***/dev/hda***, ***/dev/hdb***, ...). Um sistema de arquivos deve ser criado, formatando-o com o comando *mkfs*, em cada partição que se deseja utilizar (formatação).

fdisk: manipula tabelas de partições de discos rígidos. Ex. *fdisk /dev/hda*

du: exibe estatísticas da utilização do disco. Ex: *du –sk* .

df: exibe informações sobre o espaço livre no disco. Ex: *df –k*

quota: informa limites estabelecidos e ocupação do disco pelo usuário. Ex: *quota –v*

mkfs: constrói sistemas de arquivos em partições. Ex: *mkfs –t ext2 /dev/hda2*, *mkfs –t msdos /dev/fd0*

fsck: verifica e repara sistemas de arquivos. Ex: *fsck /dev/hda3*

mkswap (linux): cria uma área de swap no Linux. Ex: *mkswap /dev/hda5*

swapon / ***swapoff*** : ativa / desativa arquivos e dispositivos de memória virtual e swap

mount / ***umount***: monta / desmonta sistemas de arquivos. Ex: *mount /dev/sda2 /home*

showmount: exibe informações sobre os sistemas de arquivo exportados via NFS por um servidor

tune2fs: ajusta parâmetros de um sistema de arquivos **ext2**. Ex: *tune2fs –l /dev/hda3*, *tune2fs –j /dev/hda2*

9.Segurança e direitos de acesso

A segurança e os direitos de acesso a arquivos e diretórios em sistemas Unix é baseada na identificação do **proprietário** e de um **grupo** associados. ***Chown***, ***chmod*** e ***chgrp*** realizam os ajustes necessários, considerando direitos para leitura, escrita e execução (wrxwrxwrx).

Usando o comando ***ls –l*** podemos ver informações sobre os direitos de acesso a arquivos e diretórios. Toda entrada no sistema de arquivos têm um grupo de informações de controle, com 10 colunas, exibidas à esquerda pelo comando ls. A coluna mais à esquerda indica o tipo da entrada no sistema de arquivos. Seguem-se 9 colunas de permissões, divididas em 3 grupos de 3 permissões.

d_____	indica diretório
l_____	indica *link*
b_____	indica dispositivo de bloco
c_____	indica dispositivo de caracter
s_____	indica *socket*
p_____	indica *pipe*

_rwx_____ direitos do **proprietário** (*owner*) ao arquivo ou diretório
____rwx___ direitos do **grupo** ao arquivo ou diretório
_____rwx direitos dos **outros** usuários (não *owner* ou grupo) ao arq. ou diret.

Permissões de arquivos: **r** = leitura; **w** = escrita; **x** = execução
Permissões de diretórios: **r** = ls; **w** = criação, remoção, renomeação; **x** = permite *entrar* no diretório, tornando-o o diretório corrente com o comando ***cd***

- **chown** – altera proprietário de arquivos e diretórios
- **chgrp** – altera o grupo associado a arquivos e diretórios
- **chmod** – ajusta direitos de acesso

Ex: *chown fulano /home/fulano* // ajusta o proprietário do diretório fulano
chown –R fulano /home/fulano // ajusta recursivamente o proprietário ...
chgrp grupo diretório // ajusta o grupo associado ao diretório

chmod [u,g,o,a][+,-,=][r,w,x,X,s,t] arq
chmod +x arq, chmod o-w arq, chmod u+w,g+r,o-r arq

Considerando as informações de direitos de acesso como **bits em dígitos octais** (0-7), é possível ajustar-se diretamente os atributos dos arquivos e diretórios. Para tanto, 3 dígitos são utilizados, respectivamente para direitos do **proprietário**, do **grupo** e dos **demais** usuários. Os 3 bits de cada dígito correspondem, em ordem, aos direitos para **leitura**, **escrita** e **execução**.

chmod 755 arq -> rwxr_xr__ arq - 111 101 101
chmod 640 arq -> rw_r_____ arq - 110 100 000

O comando ***umask***, normalmente ajustado nos arquivos de configuração do *shell*, define os direitos de acesso que devem ser atribuídos aos novos arquivos criados. Ex: *umask 022* define que membros do grupo associado e demais usuários não terão direito de escrita (w) ao arquivo ou diretório.

Independentemente do **proprietário** associado a um arquivo executável, **processos** iniciados a partir deles preservam a **identidade** do **usuário** que os inicia, e não do dono do arquivo. Entretanto, para permitir que processos especiais sejam executados com direitos de acesso de usuários ou grupos específicos (tipicamente o *root*), é possível forçar a manutenção da identidade do usuário (***setuid***) ou grupo (***setgid***) do arquivo no processo criado a partir dele.

chmod u+s arq, *chmod g+s arq*, *chmod +s arq, chmod g-s arq, chmod –s arq*

Atribuindo-se o atributo *setuid* (chmod u+s), o processo gerado a partir do arquivo herdará as permissões associadas ao dono do arquivo. Isso é útil, por exemplo, para o programa *ping*, que requer privilégio de root para criar um *socket* do tipo *raw* para envio de pacotes ICMP. Para evitar problemas de segurança, contudo, ping, e programas que precisam se *setuid*, costumam usar os privilégios no início de suas

execuções, mas diminuir os privilégios a seguir, em tempo de execução, retornando às permissões do usuário original que iniciou o processo.

Para preservar os direitos de acesso a arquivos em diretórios compartilhados, cancelando a herança de permissões, é possível ainda ajustar um outro atributo, chamado ***sticky bit***. Atribuído ao diretório /tmp, e.g., faz com que, embora todos os usuários possam escrever nesse diretório, os direitos de cada arquivo sejam preservados.

ls –ld /tmp → *drwxrwxrwt 4 root root 4096 Aug 24 16:21 /tmp*

chmod +t diret
chmod –t diret

Normalmente, combina-se a atribuição de privilégios totais a um grupo associado a um diretório, de forma que todos possam modificá-lo, sem alterar conteúdos ali gerados por outros usuários.

chmod 777 diret; chmod +t diret; chgrp grupo diret

Usando a notação com números **octais**, o ajuste dos atributos **setuid**, **setgid** e **sticky bit** pode sem feito com um dígito a mais, anterior aos dos direitos de acesso do proprietário do arquivo ou diretório.

chmod 4754 prog -> ajusta o bit **setuid**
chmod 2755 prog -> ajusta o bit **setgid**
chmod 1777 /tmp -> ajusta o **sticky bit**

No ambiente gráfico, é possível ajustar direitos de acesso de maneira mais simplificada através de algum gerenciador de arquivos. Com o gerenciador ***nautilus***, do ***gnome***, basta selecionar-se uma pasta ou arquivo e clicar-se o botão da direita sobre ele. A opção ***propriedades*** permite o ajuste das ***permissões*** de acesso, proprietário e grupo associados.

10.Manipulação de processos

Processos no Linux podem ser iniciados diretamente através do *shell* utilizado. Alguns comandos internos do *shell* e utilitários permitem o controle de suas execuções. A execução de programas, criando processos a partir do *shell* pode ser feita colocando-os em primeiro ou segundo planos, *foreground* ou *background*.

prog <enter> // execução em **primeiro plano** (*foreground*). Shell espera programa terminar antes de voltar a mostrar o prompt para o usuário
prog ***&*** *<enter>* // execução em **segundo plano** (*background*), liberando o *shell*

Quando digitando um comando no *shell* o sinal "**;**" permite separar diversos comandos numa única linha. Ex: *prog1 ; prog2; ...*

Por outro lado, é possível que um comando ou conjunto de comandos expandam-se por mais de uma linha. Para tanto, usa-se o sinal "\" ao final de cada linha que não é a última do comando.

Ex: *programa parâmetros ... \ <enter>*
... mais parâmetros <enter>

Algumas combinações de teclas pressionadas no **terminal** geram o envio de **sinais** para o processo em *foreground*:

<ctrl> **C** // interrompe (termina) a execução de um processo em *foreground*
<ctrl> **Z** // suspende (pára) a execução de um processo em *foreground*

Os comandos internos do *shell jobs*, *bg* e *fg* permitem observar e mudar o plano de execução dos processos iniciados a partir da sessão corrente:

jobs: comando interno do *shell* que mostra os processos parados ou sendo executados em *background* que foram iniciados a partir do *shell* corrente.

bg: comando interno do *shell* que envia para execução em *background* um processo suspenso ou parado cuja ativação foi feita pelo *shell* corrente.

fg: comando interno do *shell* que envia um processo parado ou em *background* para execução em *foreground*

Os utilitários *ps*, *kill*, *killall* e *pkill* servem para identificar processos em execução e enviar sinais a eles. Todo processo em execução é identificado no sistema operacional através de um valor único, chamado ***pid*** (*process idendificator*). Esse identificador pode ser usado para o envio de **sinais** entre processos, como feito por um usuário a partir do *shell*.

Sinais são eventos enviados pelo sistema operacional para processos. O envio pode ocorrer devido à ocorrência de algum evento durante a execução de um processo, como uma exceção, ou então pode ser enviado devido a uma solicitação explícita de outro processo. De maneira geral, sinais são usados para parar ou interromper a execução de processos.

ps: utilitário que informa o *status* de processos. Ex: *ps –ef*, *ps –aux*

kill: utilitário que envia um sinal (*signal*) para um processo especificado pelo seu pid. Ex: *kill –l* (lista sinais disponíveis), *kill 1234*, *kill –9 666*

killall: envia um sinal para o processo especificado pelo seu nome. Ex: *killall vi*

Além do envio de sinais através do comando *kill*, a configuração do terminal de acesso apresenta combinações de teclas para o envio direto de alguns sinais relevantes no controle de execução de processos em *foreground*. Essas teclas são definidas com o comando ***stty***, e incluem sinais para suspender, parar, interromper, terminar e continuar a execução do processo.

stty –a: exibe a configuração dos comandos especiais do console. Entre as configurações, podem ser destacados:

***intr CHAR*:** envia um sinal SIGINT (2). Ex: *stty intr ^C*
quit CHAR: envia um sinal SIGQUIT (3) ao processo em *foreground*.
***susp CHAR*:** envia um sinal SIGSTP (20), suspendendo a execução de um processo. Ex: *stty susp ^Z*

stop CHAR: para a saída de dados (*output*) do processo em *foreground*.
start CHAR: reinicia a saída de dados (*output*) depois de tê-la interrompido.

Processos têm prioridades, chamadas **valor de *nice***, que variam de -20 a 19. -20 é o valor mais prioritário e 19 o menos. O valor padrão para essa prioridade é 0, que é um valor intermediário. Um usuário pode diminuir a prioridade de um processo, aumentando seu valor de *nice*. Assim, pode-se variar o valor de nice de 0 (valor padrão) até 19. Apenas o usuário *root* pode atribuir valores de prioridade na faixa de -1 a -20, que corresponde aos mais prioritários.

De maneira geral, é comum diminuir a prioridade de processos longos, sem interatividade no terminal, de forma que eles não prejudiquem o tempo de resposta de processos interativos.

nice: executa um comando com baixa prioridade de escalonamento. Ex: *nice –n +10 prog*
renice: ajusta a prioridade de um processo já criado.Ex: *renice +1 pid*, *renice 20 pid*
ionice: ajusta a prioridade das operações de E/S do processo.
nohup: inicia a execução de um programa imune a *hangups* (não é encerrado com o fim do *shell* a partir do qual foi iniciado). Ex: *nohup prog &*
disown: remove um job especificado da tabela de jobs ativos do shell corrente. Opcionalmente, não o remove da tabela mas apenas faz com que ele não receba o sinal SIGHUP caso o shell o receba.

at e *cron* são utilitários para programar a execução de processos.

at: enfileira um *job* para execução posterior. É preciso que o servidor **atd** esteja sendo executado. Ex: *at –f prog 00:00 01.01.05*
atq: examina os *jobs* selecionados para execução posterior
atrm: remove os *jobs* selecionados para execução posterior
atrun: executa *jobs* selecionados para execução posterior
batch: executa comandos quando a carga (*load*) do sistema estiver abaixo de um valor especificado

crontab: programa usado para instalar, desinstalar ou listar as tabelas usadas pelo *daemon* **cron**. Cada usuário pode ter sua própria tabela. Requer que o servidor **crond** esteja em execução.

Útil na criação de scripts, o comando *sleep* suspende temporariamente a execução de um processo:

sleep: suspende a execução de um comando por pelo menos o tempo especificado em segundos. Ex: *sleep 10; echo fim!*

É possível contabilizar os recursos de processamento usado por processos:

time: contabiliza o tempo de execução de um processo. Ex: *time gcc ...*

top: mostra processos com maior ocupação da UCP e controla suas execuções.

O encerramento de processos é feito com o envio de um sinal apropriado. A partir do shell, há várias formas de fazê-lo:

a) em *foreground*: **<ctrl> C**
b) em *background*: **fg; <ctrl> C**
c) iniciado por outro *shell*:
ps –ef | grep [nome_prog ou "*login"*]**; kill –9 pid**
ps –U *login*; kill –9 pid
killall –9 nome_prog

11. Redirecionamento de entrada e saída de dados

A entrada e saída de dados em processos no ambiente Unix é feita através de 3 arquivos abertos automaticamente nas suas ativações: ***stdin* (0)**, ***stdout* (1)** e ***stderr* (2)**. De maneira geral, esses arquivos apontam naturalmente para o terminal ou para a janela associada ao *shell* de ativação do processo.

Ajustes especiais indicados na linha de comando de ativação de um processo, entretanto, permitem redirecionar esses dados para outros arquivos, ou mesmo para a comunicação direta entre processos, através de *pipes*.

> *ou* 1>: redirecionamento da saída de dados de um processo (*overwrite*). Ex: *ls > diret*
>> *ou* 1>>: redirecionamento da saída de dados (*append*). Ex: *ls >> diret*
2> : redirecionamento das mensagens de erro. Ex: *make >& arq_msg_erro*
2>> : redirecionamento das mensagens de erro (*append*). Ex: *prog 2>> msg_erro*
< : redirecionamento da entrada de dados de um processo. Ex: *prog1 < arq_dados*
&>: redirecionamento de *stdout* e *stderr* para o mesmo local. Ex: *prog &> saida*
| (***pipe***): cria um mecanismo de comunicação entre processos. Ex: *ls –la | more*

tee: copia os dados do arquivo padrão de entrada para o arquivo padrão de saída, fazendo opcionalmente uma cópia para outros arquivos de saída. Ex: *p1 | tee result*

12.Gerenciamento de memória e recursos de programação com memória compartilhada

free: (linux) mostra as quantidades de memória livre e ocupada
ipcs: informa o *status* de mecanismos de comunicação inter-processo. Ex: *ipcs*

ipcrm: remove uma fila de mensagem, semáforo ou memória compartilhada. Ex: *ipcrm {shm | msg | sem} id ...*

13. Desligando o sistema

Para oferecer um melhor desempenho nos sistemas de arquivos, caches são mantidos pelo sistema operacional em áreas de memória. Entretanto, uma desvantagem dessa técnica é que, se o sistema for desligado de maneira abrupta, é possível que arquivos abertos sejam corrompidos. Assim, um procedimento de desligamento do sistema deve ser realizado antes que o computador possa ser desligado.

sync: força a conclusão de operações de disco pendentes, esvaziando o *cache* do disco
shutdown: encerra a operação do sistema. Ex: *shutdown –h now*, *shutdown –h –t 5*, *shutdown –r now*
halt: encerra a operação do sistema
reboot: reinicia a operação do sistema operacional
<**ctrl**> <**alt**> <**del**>: reinicia o sistema (linux)
init / ***telinit***: controla a iniciação e o nível de execução do sistema (*runlevel*). Enquanto o nível **6** promove a **reiniciação** do sistema, o nível **0** faz o seu **encerramento**. Ex: *init 0*, *init 6*

14.Utilitários de rede

ifconfig: configura e exibe parâmetros de interfaces de rede. Ex: *ifconfig*
route: manipula e exibe a tabela de rotas. Ex: *route*
netstat: informa sobre as conexões de rede, tabelas de rotas, estatísticas sobre as interfaces e outras informações. Ex: *netstat –na*, *netstat –r*, *netstat –tcp*

arp: consulta e atualização da tabela de resolução de endereços. Ex: *arp -a*
ping: envia pacotes ICMP ECHO_REQUEST para *hosts* na rede, que enviam pacotes ICMP ECHO_RESPONSE caso estejam ativos
nslookup, ***host***, ***dig***: utilitários para consultas de nomes de domínio na Internet
traceroute: imprime o caminho dos pacotes entre o computador local e uma estação remota

ftp: programa de transferência de arquivos da ARPANET
telnet: programa para acesso de computadores remotos através do protocolo TELNET

ssh: programa de *shell* remoto com transmissão de dados criptografados
scp: programa de transferência de arquivos com transmissão criptografada

rlogin: programa de *login* remoto
rsh: programa de *shell* remoto. Permite executar comandos ou abrir uma sessão interativa
rcp: programa para cópia de arquivos entre estações

mail: programa para envio e recebimento de mensagens do correio eletrônico
pine: programa para email e news na Internet

15.Impressão

lpr: impressão *off line*. Usa o servidor de *spool* para impressão de arquivos quando o dispositivo estiver disponível. Ex: *lpr arquivo*
lpq: examina a fila de impressão. Ex: *lpq*
lprm: remove *jobs* da fila do *spool* de impressão. Ex: *lprm 123*

16.Editores de texto

Modo texto: ***vi***, ***emacs***, ***pico***
Ambiente gráfico: ***kedit, kwrite, xedit, ...***

17.Gerenciamento de pacotes

A instalação de programas num sistema Linux pode ocorrer de diversas formas. Tendo um compilador C instalado, é possível copiar os arquivos fontes de um programa desejado para um diretório local, configurá-lo, compilá-lo e copiar os arquivos gerados para os diretórios apropriados.

```
gunzip pacote.src.tar.gz
tar –xvf pacote.src.tar
cd pacote.src
./configure
make
make install
```

Outros pacotes necessários para uma aplicação devem ser copiados e configurados num procedimento semelhante. A instalação de novas versões desses pacotes envolve a retirada manual dos arquivos e a configuração da nova versão.

Para simplificar esse procedimento, foi criado o conceito de pacote de software no formato RPM (*RedHat Package Manager*), desenvolvido inicialmente pela empresa RedHat. Outros formatos foram equivalentes foram adotados posteriormente por outras distribuições, como .deb.

Pacotes nesses formatos podem ser facilmente instalados, verificados, atualizados e desinstalados. Para tanto, basta copiar o arquivo no formato apropriado, que já concentra num único arquivo todos os arquivos necessários para a sua configuração e instalação.

Outros pacotes necessários para uma aplicação devem ser previamente instalados, contudo.

Uso do rpm:

rpm –ivh pacote.rpm : instala pacote
rpm –Uvh pacote.rpm : instala pacote, removendo suas versões anteriores
rpm –e pacote : remove pacote
rpm – q pacote : verifica status de pacote

Ferramentas adicionais foram desenvolvidas posteriormente, possibilitando a busca de pacotes a partir de repositórios na Internet, tratando também de dependências entre pacotes e de suas instalações automáticas.

Ferramentas como ***yum*** e ***apt*** tratam dessas funções, bastando especificar em arquivos de configuração os repositórios apropriados para buscas de programas. Distribuições Linux já possuem configurações apropriadas na instalação do sistema.

Ex. apt:

apt-get update: atualiza a lista de pacotes que podem ser instalados ou atualizados a partir das fontes especificadas.
apt-get install pacote: instala o pacote, com suas respectivas dependências
apt-get upgrade: atualiza as versões dos pacotes instalados
apt-get dist-upgrade: atualiza uma distribuição
apt-get remove pacote: remove pacote
apt-cdrom add: adiciona informações sobre CDs como fonte para instalações
apt-get source pacote: copia código fonte do pacote, incluindo suas dependências.

Alguns ambientes gráficos possuem também ferramentas para gerenciamento de pacotes.

18. Comunicação entre usuários

write: escreve uma mensagem para outro usuário conectado ao sistema
wall: envia uma mensagem para o terminal de todos os usuários conectados
mesg: controla a escrita de mensagens no terminal. Ex: *mesg y*, *mesg n*

talk: programa para conversa com outros usuários – *chat*, Ex:. *talk fulano*, *talk fulano@host.dom* (requer que o programa ***talkd*** esteja habilitado na máquina remota)

19.Miscelânea

expr: interpreta argumentos como uma expressão. Ex: *expr 2 + 2*, *expr 2 * 3*, *expr 5 / 2*, *expr 5 % 2*
cal: exibe um calendário. Ex: *cal*, *cal 2003*
sed: stream editor.
awk: