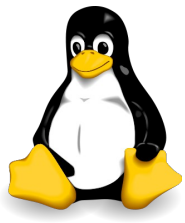
Apostila elaborada por Fernando Henrique e Paula da Luz - presidente@ifsc.usp.br
Mestrando em Física Computacional pelo Instituto de Física de São Carlos

Tutorial do Linux



Um pouco de História

O Kernel do Linux foi inicialmente desenvolvido pelo estudante finlandês Linus Torvalds numa tentativa de conseguir o seu próprio sistema operativo semelhante ao Unix (Unix-like) que corresse em processadores Intel 80386.

Linus obteve uma cópia do Minix estudou a mesma não ficando satisfeito com sua arquitetura. O projeto foi lançado em 1991 numa famosa mensagem para um grupo de discussão da Usenet. Curiosamente, o nome Linux foi criado por Ari Lemmke, administrador do site ftp.funet.fi que deu esse nome ao diretório FTP onde o kernel do linux estava inicialmente disponível (Linus tinha batizado como "Freax", inicialmente). Desde o princípio, ele recebeu a ajuda de hackers do Minix, e hoje recebe contribuições de milhares de programadores de todo mundo.

Arquitetura

O Linux é um núcleo (*kernel*) monolítico. Isto é, as funções do núcleo (agendamento de processos, gerenciamento de memória, operações de entrada e saída, acesso ao sistema de arquivos) são executadas no espaço do núcleo. Uma característica do Linux é que algumas das funções (drivers de dispositivos, suporte à rede, sistemas de arquivo, por exemplo) podem ser compiladas e executadas como módulos (LKM - loadable kernel modules), que são bibliotecas compiladas separadamente da parte principal do núcleo e podem ser carregadas e descarregadas após o núcleo estar em execução.

Portabilidade

Embora Linus Torvalds não tenha tido como objetivo inicial tornar o Linux um sistema portável, ele evoluiu nessa direção. Linux é hoje, na verdade, um dos núcleos (*kernels*) de sistema operacional com mais portabilidade, correndo em sistemas desde o iPaq (um computador portátil) até o IBM S/390 (um denso e altamente custoso mainframe)

De qualquer modo, é importante notar que os esforços de Linus foram também dirigidos a um diferente tipo de portabilidade. Portabilidade, de acordo com Linus, era a habilidade de facilmente compilar aplicações de uma variedade de fontes no seu sistema; portanto o Linux originalmente tornou-se popular em parte devido ao esforço para que as fontes GPL ou outras favoritas de todos executassem em Linux.

Termos de Licenciamento

Inicialmente, Torvalds lançou o Linux sob uma licença que proibia qualquer uso comercial. Isso foi mudado de imediato para a Licença Pública Geral GNU. Essa licença permite a distribuição e mesmo a venda de versões possivelmente modificadas do Linux mas requer que todas as cópias sejam lançadas dentro da mesma licença e acompanhadas do código fonte.

Distribuições

Atualmente, um Sistema Operacional GNU/Linux completo (equivalente a "distribuição de GNU/Linux") é uma coleção de software livre (e alguns não-livres) criados por indivíduos, grupos e organizações de todo o mundo, tendo o Linux como seu núcleo. Companhias como a Red Hat, a SuSE, a Mandriva (união da Mandrake com a Conectiva), bem como projetos de comunidades como o Debian ou o Gentoo, compilam o software e fornecem um sistema completo, pronto para instalação e uso.

Dentre as maiores, distribuídas em CDs, podem-se citar: Slackware, Debian, Suse, Ubuntu e Conectiva. O que faz a diferença é como estão organizadas e pré-configuradas as aplicações. A distribuição Conectiva Linux, por exemplo, tinha as suas aplicações traduzidas em português, o que facilitou que usuários que falam a Lingua Portuguesa tenham aderido melhor a esta distribuição. Hoje esta

distribuição foi incorporada à Mandrake, o que resultou na Mandriva. Para o português, existe também a distribuição brasileira Kurumin, construída sobre Knoppix e Debian.

Distribuições atuais de Linux:

Arch Linux	*Fedora*	*Kubuntu*	*Slackware Linux*
Caixa Mágica	*Gentoo Linux*	*Lycoris*	*Sorcerer GNU/Linux*
Conectiva	*GoboLinux*	*Mandriva*	*SuSE*
Cytrun Linux	*Kake Linux*	*Muriqui Linux*	*TechLinux*
Debian	*Kalango Linux*	*Red Hat Linux*	*tsl linux*
Debian-BR-CDD	*Knoppix*	*RXART Linux*	*Ubuntu Linux*
Famelix	*Kurumin Linux*	*Skolelinux*	*White Box*

O que é uma shell?

No mundo da computação, uma shell é um programa que interpreta comandos do usuario para que o sistema operacional possa entender e executar o que lhe é pedido.

A shell é uma interface em linha de comando, baseada em texto. O usuário pode digitar comandos para executar funções ou programas, abrir e navegar diretórios, e ver processos que estão ocorrendo no momento. Sendo a shell a unica camada para o sistema operacional, você pode fazer operações que não são possiveis usando usando uma interface grafica do usuario (do ingles GUI - graphical user interface). Alguns exemplos inclui mover arquivos dentro das pastas de sistema e deletar arquivos que são tipicamentes bloqueados. Para executar isso, você precisa saber as a sintaxe correta dos comandos e permitir o seu acesso como administrador do sistema.

Duas shells mais comumente utilizadas são a Bourne Again Shell (bash) e a Tenex C shell (tcsh).

Vale ressaltar que na linha de comandos de uma shell, podemos utilizar diversos comandos um após o outro, ou até mesmo combiná-los numa mesma linha. Se colocarmos diversas linhas de comandos em um arquivo texto simples, teremos em mãos um *Shell Script*, ou um script em shell, já que Script é uma descrição geral de qualquer programa escrito em linguagem interpretada, ou seja, não compilada. O script shell vai ser um assunto abordado mais adiante.

Lista de Comandos

Comandos para manipulação de arquivos

A partir do momento que vamos começar a mexer com arquivos, precisamos saber alguns comandos básicos para a sua manipulação.

- cd - Navegando entre diretórios
- ls - Listar arquivos
- mkdir - Cria um diretório
- rmdir - Remove um diretório vazio
- cp - Cópia de arquivos e diretórios
- mv - Move arquivos e diretórios
- rm - Deleta arquivos e diretórios
- ln - Linkando arquivos

- cat - Exibe o conteúdo de um arquivo ou direciona-o para outro
- less - Exibe o conteúdo de um arquivo grande, permitindo a navegação no mesmo
- pipe - Concatena comandos
- grep – Filtra e busca resultados

cd - Navegando entre diretórios

cd [nome_do_diretório]

Este comando acima mudará o diretório atual de onde o usuário está. Há também algumas abreviações de diretórios no Linux para a facilitação, estes são:

Abreviação	**Significado**
. (ponto)	*Diretório atual*
.. (dois pontos)	*Diretório superior*
~ (til)	*Diretório HOME do usuário*
/ (barra)	*Diretório Raiz*
- (hífen)	*Diretório anterior*

Por exemplo, se eu quero ir para o meu diretório home, faço o seguinte:

```
$ pwd
/usr/games
$ cd ~
$ pwd
/home/pres
```

Ou seja, eu estava no diretório `/usr/games`, e com um simples cd para o diretório `~`, fui para o meu diretório home (`/home/pres`). Quando você deseja saber o caminho completo do diretório em que você está, utilize o comando pwd. Se você deseja ir para um diretório que está na raiz diretamente, você usa o `/` antes, exemplo:

```
$ pwd
/usr/local/RealPlayer7/Codecs
$ cd /etc/rc.d
$ pwd
/etc/rc.d
$ cd -
$ pwd
/usr/local/RealPlayer7/Codecs
```

Eu estava no diretório `/usr/local/RealPlayer7/Codecs` e quis ir para o diretório `etc/rc.d` que está na raiz. Note depois que eu usei o hífen e fui de volta para o último diretório em que eu estava.

ls - Listar arquivos

ls [opções] [arquivo/diretório]

Este comando lista os arquivos, nada mais que isso. Se você executar apenas o ls sozinho, ele vai mostrar todos os arquivos existentes no diretório atual. Há também as opções extras:

Parâmetro	**Significado**
-l	*Lista os arquivos em formato detalhado.*
-a	*Lista os arquivos ocultos (que começam com um .)*
-h	*Exibe o tamanho num formato legível (combine com -l)*
-R	*Lista também os subdiretórios encontrados*
-t	*Lista em ordem cronológica*
-s	*Lista em blocos de 1024-bytes, mostrando a esquerda*

Exemplo de uma listagem detalhada:

```
$ ls -l
total 9916
drwxrwxr-x   5 hugo hugo    1302 Aug 16 10:15 CursoC_UFMG
-rw-r--r--   1 hugo hugo  122631 Jul 12 08:20 Database.pdf
-rw-r--r--   1 hugo hugo 2172065 Jul 12 08:20 MySQL.pdf
-rw-r--r--   1 hugo hugo 2023315 Jul 12 08:20 PHP3.pdf
```

Podemos também usar no ls os wildcards, ou seja, caracteres que substituem outros. Exemplo: eu quero listar todos os arquivos que têm a extensão .txt, faço o seguinte:

```
$ ls *.txt
debian-install.txt manualito.txt named.txt    plip.txt
seguranca.txt      ipfw.txt      mouse.txt    placa_de_video.txt
rede.txt           sis.txt
```

O wildcard é o "*", que representa "tudo".txt. Existem outros wildcards, exemplo disso é o ponto de interrogação (?), que substitui apenas 1 caractere, exemplo:

```
$ ls manual?.txt
manual1.txt  manual2.txt  manual3.txt  manualx.txt  manualP.txt
```

Existe outro wildcard, que envolve os colchetes. Por exemplo:

```
$ ls manual[3-7].txt
manual3.txt  manual4.txt  manual6.txt  manual7.txt
```

Lista todos os arquivos que tiverem como manual?.txt, onde o ? pode ser substituído por 3, 4, 5, 6 e 7.

mkdir - Cria um diretório

mkdir <nome_do_diretório>

Cria um diretório. Exemplo:

```
$ mkdir ~/paginas
```

Este comando criará o diretório paginas no seu diretório `home`.

rmdir - Remove um diretório vazio

rmdir <nome_do_diretorio>

Apaga um diretório que esteja vazio. Exemplo:

```
$ rmdir /tmp/lixo
```

Isto apagará o diretório `/tmp/lixo` apenas se ele estiver vazio. Para apagar um diretório com seu conteúdo, refira-se ao comando rm.

cp - Cópia de arquivos e diretórios

cp [opções] <arquivo_origem> <arquivo_destino>

O comando cp copia arquivos e diretórios. Como opções dele, podemos ver:

Parâmetro	**Significado**
-i	*Modo interativo*
-v	*Mostra o que está sendo copiado*
-R	*Copia recursivamente (diretórios e subdiretórios)*

Exemplos:

Quero copiar `brasil.txt` para `livro.txt`, com a opção de modo interativo.

```
$ cp -i brasil.txt livro.txt
cp: sobrescrever `livro.txt'?
```

Como o arquivo `livro.txt` já existia, ele pergunta se quer sobrescrever, responda y(sim) ou n(não).

Agora eu quero copiar o diretório `/home/ftp` com tudo dentro (até seus subdiretórios) para `/home/ftp2`, faço o seguinte:

```
$ cp -R /home/ftp /home/ftp2
```

mv - Move arquivos e diretórios

mv [opções] <arquivo_origem> <arquivo_destino>

Este comando simplesmente move algum arquivo para outro lugar. Ele também é usado para renomear um arquivo. Vale os parâmetros listados no comando `cp`. Por exemplo, se eu quero renomear o `industria.txt` para `fabrica.txt`, eu faço o seguinte:

```
$ mv industria.txt fabrica.txt
```

Se eu quiser mover o `industria.txt` para `/home/usuario` com o mesmo nome, faço:

```
$ mv industria.txt /home/usuario
```

rm - Deleta arquivos e diretórios

rm [opções] <arquivo>

Novamente, as opções listadas no comando `cp`, são válidas aqui, principalmente a `-i`, pois não queremo apagar um arquivo sem querer, né :-)

Este comando apaga definitivamente o arquivo ou diretório. Exemplo:

```
$ rm arquivo.bin
```

Para apagar um diretório com todo seu conteúdo, usa-se a opção `-r`, assim:

```
$ rm -r /tmp/lixo
```

ln - Linkando arquivos

ln -s <arquivo_origem> <link simbólico>

Este comando é usado para gerar links simbólicos, ou seja, que se comportam como um arquivo ou diretório, mas são apenas redirecionadores que mandam seu comando para outro arquivo ou diretório, por exemplo:

```
$ ln -s /manual /home/linux-manual
```

Este comando criará o link /home/linux-manual, se você der um `ls -l` você verá que o diretório `/home/linux-manual` está apontando para /manual. Se você ir para o `/home/linux-manual`, você na verdade estará no `/manual`, mas como é um link, não há diferença.

cat - Exibe o conteúdo de um arquivo ou direciona-o para outro

cat <arquivo>

Este comando existe para mostrar o conteúdo de um arquivo, ou para fazer a cópia deste arquivo, ou uma junção. Vejamos um exemplo, se eu quiser mostrar o conteúdo de `/home/usuario/contato`, eu digito:

```
$ cat /home/pres/contato
```

Aparecerá o conteúdo do arquivo contato:

Presidente – Fernando Luz

presidente_arroba_ifsc_ponto_usp_ponto_br

Este comando pode também servir de direcionador para outro arquivo. Indicadores são usados para isso:

Indicador ">" - faz uma cópia, exemplo:

```
$ cat contato1 > contato2
```

Indicador ">>" - Acrescenta um arquivo ao outro, exemplo:

```
$ cat contato1 >> contato2
```

less - Exibe o conteúdo de um arquivo grande, permitindo a navegação no mesmo

less <arquivo>

Um dos problemas do cat é que quando visualizamos o conteúdo de um arquivo grande, ele é jogado todo na tela, e acabamos vendo somente as linhas finais do texto. Para resolver este problema, criaram um comando chamado more, com as funções parecidas com a do cat, mas ele não permitia retrocesso na leitura do arquivo. Então Mark Nudelman iniciou seu trabalho no less em 1983 quando teve a necessidade de utilizar uma ferramenta como o more porém com rolagem para trás. Ela também deveria suportar a leitura de arquivos de log grandes demais na época para ferramentas como vi.

O nome less é o contrário de more, que vem de uma brincadeira com o nome do seu precessor.

```
$ less /home/pres/disertacao.tex
```

Aparecerá o conteúdo do arquivo contato:

Presidente – Fernando Luz

Ta na hora de começar hein!!! X-(

Os procedimentos executados com o cat, também são validos com o less. Pressione h enquanto ele é executado para ver as funcionalidades que o less permite (como busca dentro do texto).

pipe – Concatena comandos

<comando> | <comando>

O pipe é um das maneiras que o Linux pode utilizar para comunicação entre processos. De uma maneira simples podemos dizer que o pipe nada mais é do que o encadeamento de processos. Ao primeiro olhar o pipe pode até não chamar atenção dos principiantes, mas trata-se de um ferramenta muito poderosa. Esse encadeamento de processo pode ser ativado pelo usuário através do comando "|". Vamos demonstrar no exemplo abaixo o uso dessa ferramenta:

```
$ cp --help | less
```

O resultado do comando `cp -help`, é jogado para o comando less, que permite uma melhor forma de ler o resultado.

grep – Filtra e busca resultados

grep <arquivo>

Uma utilidade do comando grep é a de buscar por ocorrências de expressões dentro de 1 ou mais arquivos. Imagine que precisamos pesquisar a expressão "`servidor`" dentro de todos os arquivos do diretório `/etc`.

```
$ grep -s kernel /etc/*
/etc/filesystems:#Uncomment the following line if your modular kernel has vfat
/etc/make.conf.example:#setting for all linux kernel+glibc based systems.
/etc/modules.conf:#Crypto modules (see http://www.kerneli.org/)
/etc/modules.conf:#0xFFFFFFFF let the kernel module autodetect the correct value
```

O resultado aparece da seguinte forma.

nome do arquivo: linha que contém a expressão.

Podemos também utilizar o grep como um filtro de resultados. Utilizando o comando `pipe (|)`, podemos concatenar a saida de um comando com grep, como no exemplo abaixo que estamos interessados em saber quais documentos foram mexidos pela última vez no ano de 2007.

```
$ ls -ln | grep 2007-
drwx------ 2 1000 100    4096 2007-02-05 18:57 amsn_received
drwxr-xr-x 2 1000 100    4096 2007-03-01 20:06 Desktop
-rw-r--r-- 1 1000 100    5483 2007-02-01 13:57 DissertaÃ§Ã£o.tex
-rw-r--r-- 1 1000 100    1229 2007-02-05 18:07 Dissertação.log
-rw-r--r-- 1 1000 100       0 2007-02-05 18:07 Dissertação.tex
drwxr-xr-x 8 1000 100    4096 2007-01-30 10:45 Mestrado
-rw-r--r-- 1 1000 100    6149 2007-02-01 13:56 tese.tex
```

Comandos sobre processos do sistema

- ps - Listando processos
- kill - Matando um processo
- killall - Matando processos pelo nome

ps - Listando processos

ps [opções]

Quando um programa é executado no sistema, ele recebe um número de identificação, o chamado PID. Este comando lista esses processos executados, e apresenta o PID. Além do PID, ele também mostra o comando executado (CMD) e também o STAT (status atual do processo executado, veja nota abaixo), além de outros.

O status do processo é identificado por letras, aqui segue uma tabela com as definições de cada letra:

Letra	**Definição**
0	*Não existente*
S	*Descansando, fora de funcionamento (Sleeping)*
R	*Rodando (Running)*
I	*Intermediando (Intermediate)*
Z	*Terminando (Zumbi)*
T	*Parado (Stopped)*
W	*Esperando (Waiting)*

Agora um exemplo para este comando:

```
$ ps aux
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0  1120   52 ?        S    Dec25   0:05 init
root         2  0.0  0.0     0    0 ?        SW   Dec25   0:00 [kflushd]
root         3  0.0  0.0     0    0 ?        SW   Dec25   0:00 [kupdate]
root         4  0.0  0.0     0    0 ?        SW   Dec25   0:00 [kpiod]
root      1004  0.0  0.0 10820   48 ?        SN   Dec25   0:00 [mysqld]
root      1007  0.0  0.0  2852    0 ?        SW   Dec25   0:00 [smbd]
pres      1074  0.0  0.0  1736    0 tty1     SW   Dec25   0:00 [bash]
pres      1263  0.0  0.0  1632    0 tty1     SW   Dec25   0:00 [startx]
pres      1271  0.0  0.0  2304    0 tty1     SW   Dec25   0:00 [xinit]
pres      1275  0.0  2.4  4312 1360 tty1     S    Dec25   0:16 wmaker
pres      2461  0.0  0.0  1636    0 tty1     SW   07:09   0:00 [netscape]
pres      9618  0.9  4.9  5024 2688 pts/1    S    09:56   0:06 vim d03.html
```

Este parâmetro (`aux`) fez o ps listar todas as informações sobre todos os processos executados.

kill - Matando um processo

kill [-SINAL] <PID>

O comando kill é muito conhecido, ele serve para matar um processo que está rodando. Matar? Terminar este processo, finalizar natoralmente! Para matar um processo, temos de saber o PID dele (veja o comando ps), e então executar o kill neste PID. Vamos killar o Netscape:

```
$ ps aux | grep netscape
pres      2461  0.0  0.0  1636    0 tty1     SW   07:09   0:00 [netscape]
$ kill -9 2461
```

E o processo do Netscape foi morto!

killall - Matando processos pelo nome

killall [-SINAL] <comando>

Faz a mesma coisa que o kill, só que a vantagem aqui é que você não precisa saber o PID do processo, e sim o nome. A desvantagem é que se tiver dois processos com o mesmo nome, os dois são finalizados. Seguindo o exemplo do comando kill:

```
$ ps aux | grep netscape
pres      2461  0.0  0.0  1636    0 tty1     SW   07:09   0:00 [netscape]
$ killall -9 netscape
```

Dicas Básicas

Para saber informações da máquina onde você esta trabalhando, basta ver o arquivo `/proc/cpuinfo`, e para verificar a memória do seu sistema, observe o arquivo `/proc/meminfo`.

```
$ cat /proc/cpuinfo | grep "model name"
model name      : AMD Athlon(TM) XP1700+
$ cat /proc/meminfo | grep MemTotal
MemTotal:       514716 kB
```

O comando grep foi concatenado apenas para não mostrar a saída inteira do arquivo.

As vezes, é interessante que alguns comandos tenham na sua chamada um valor diferente do padrão, imagine que toda vez que vamos deletar um arquivo, temos que colocar -i para ele sempre pedir a confirmação dos arquivos a serem deletados. Para isso podemos definir as chamadas padrão dos comandos da shell editando o arquivo `./bashrc`.

```
$ emacs .bashrc
```

E inclua a linha a seguir no arquivo.

```
alias rm="rm -i"
```

Outros tipos de comandos

Descompactar arquivos

Extensão .tar.gz	*tar zxpvf arquivo.tar.gz*
Extensão .tar	*tar xpvf arquivo.tar*
Extensão .zip	*unzip arquivo.zip*

Compactar arquivos

Empacotar um diretório em .tar	*tar cvf diretorio/*
Empacotar um diretório em .tar.gz	*tar zcvf diretorio/*
Empacotar um diretório em .zip	*zip -r* [*arquivo-destino*].zip [diretório-*origem*]

Espaço em disco

df -h	*Mostra o espaço em disco das partições montadas*
du -hs	*Mostra o espaço ocupado pelo diretório atual*

Informações do sistema

date	*Mostra a data e hora atual*
uptime	*Mostra quanto tempo seu sistema está rodando*
free	*Exibe a memória livre, a usada, e os buffers da memória RAM*
top	*Mostra os processos que mais gastam memória*

Programas (console)

vi	*Editor de texto*
emacs	*Editor de texto*
links	*Navegador Web*

Bibliografias consultadas

BOAVENTURA, Frederico Freire. **GNU/Linux.** Disponível em: <http://galahad.com.br/lnx/index.php>. Acesso em: 04 mar. 2007.

CISNEIROS, Hugo. **Página do Eitch.** Disponível em: <http://www.devin.com.br/eitch/>. Acesso em: 04 mar. 2007.

KLIMAS, Stan; KLIMAS, Peter; KLIMAS, Marrei. **LINUX NEWBIE ADMINISTRATOR GUIDE.** Disponível em: <http://www.onlinux.com.br/dicas/lnag/Linux_help.htm>. Acesso em: 03 mar. 2007.

MITRE, J. F.. **.bashrc.** Disponível em: <http://www.vivaolinux.com.br/conf/verConf.php?codigo=341>. Acesso em: 03 mar. 2007.

SHELL(2) Disponível em: <http://www.istf.com.br/vb/archive/index.php?t-5484.html>. Acesso em: 03 mar. 2007.

SILVA, Gleydson Mazioli da. **Guia Foca GNU/Linux Intermediário.** Disponível em: <http://www.htmlstaff.org/guiafoca/intermediario/>. Acesso em: 04 mar. 2007.

WIKIPÉDIA Disponível em: <http://pt.wikipedia.org/wiki/>. Acesso em: 15 fev. 2007.