4LINUX

YOUR INTELLIGENCE IN LINUX

WWW.4LINUX.COM.BR

Administração Linux à Distância – LPIC-1

Ano V

Comandos Básicos GNU/Linux

LINUX ACADEMICS
www.linuxacademics.com.br

HACKERTEEN
www.hackerteen.com.br

Índice

l

m

n

o

p

r

s

t

u

w

Bibliografia

Comandos Básicos GNU/Linux

Apostila versão 1.0

"Na teoria, não há diferença entre teoria e prática. Mas, na prática, há."
- Jan L.A van de Snepscheut

Os comandos GNU/Linux possuem algumas características particulares. Eles devem ser digitados em letras minúsculas, ou seja, são **CASE-SENSITIVE**.

No mundo *NIX(Linux,Unix), o conceito de comandos é diferente do padrão MS-DOS. Um comando é qualquer arquivo executável, podendo ou não ser criado por você.

É no **shell** que os comandos são executados. Ele é o responsável pela **interação** entre o ***usuário*** e o ***sistema operacional***, pois ele é que interpreta os comandos e os traduz para uma linguagem simples e inteligível para kernel.

pwd

Indica em qual diretório me encontro

Exemplo:

```
$ pwd
/home/leo
```

cd

Esse comando mudará a localização do usuário na árvore de diretórios. Para diminuir a digitação, o GNU/Linux aceita algumas abreviações, chamadas de rotas relativas:

$ cd <caminho>

Irá para o diretório que você especificicar, exemplo:

$ cd /etc

$ cd ..

Irá para o diretório pai do diretório em que você está presente, exemplo:

```
$ pwd
/home/leo
$ cd ..
$ pwd
/home
```

Perceba que usei o comando pwd para verificar em que diretório eu me encontrava, depois usei o comando cd .. para ir ao diretório pai.

*Diretório pai nada mais é que o diretório que dá origem a outro.

$ cd ../../

Volta dois diretórios, exemplo:

```
$ pwd
/home/leo
$ cd ../../
$ pwd
/
```

$ cd ~

Irá para o diretório home (pessoal) do usuário que o estiver executando.

Exemplo:

```
$ pwd
/
$ cd ~
$ pwd
/home/leo
```

Perceba que usei o comando pwd para me situar, depois usei o comando cd ~ para voltar ao meu diretório pessoal.

$ cd

Se for executado somente este comando sozinho, você irá para ao diretório home (equivalente ao ~)

$ cd /

Irá para o diretório raiz do sistema.

ls

O comando ls é usado para listagem de arquivos e diretórios. Quando usado sem qualquer parâmetro o diretório corrente será listado.

$ ls -l

Lista os arquivos ou diretório de uma forma bem detalhada (quem criou, data de criação, tamanho, dono e grupo a qual eles pertencem)

Exemplo:

```
$ ls -l
drwxr-xr-x  15 leo  4linux     4096 2006-06-30 22:50 4linux/
```

Significado das colunas:

d significa que o arquivo é um diretório
rwxr-xr-x são as permissões do arquivo
15 é a quantidade de subdiretórios que existem no diretório 4linux/
4096 é tamanho do arquivo
2006-06-30 é a data da última alteração
22:50 é o horário da última alteração
4linux/ é o nome do arquivo, que no caso é um diretório

$ ls -lh

Acrescentando o h, podemos ver o tamanho do arquivo mais próximo da unidade de medida correta.

$ ls -a

Esse comando mostra os arquivos que estão ocultos.
Um arquivo para se tornar oculto deve ter um ponto antes do nome, exemplo: .4linux e só será visualizado com o parâmetro "-a"

$ ls -i

Esse comando além de listar o diretório, nos dá o inode do arquivo.
Inode é a verdadeira identidade de um arquivo no GNU/Linux, é uma sequência de números como um RG.

Esse comando é útil pois nos revela se um arquivo é um link simbólico ou hard link.

$ ls --color

Esse comando lista os arquivos em forma de cores com a finalidade de diferenciar os tipos de arquivos.

Cores:

Azul: diretórios
Azul ciano: atalhos (links simbólicos)
Vermelho: arquivos compactados
Magenta: arquivos de imagens (jpg,png,gif)
Verde: arquivo executável (não precisamente binário, mas que tenha permissão de execução)
Amarela: dipositivos do sistema (devices)
Cinza: arquivos de textos ou desconhecidos

$ ls -R

Faz uma listagem recursiva do diretório, ou seja, mostra o conteúdo dos seus subdiretórios.

$ ls -1 (número um)

Faz a listagem em apenas uma coluna.

$ ls -X

Usa a extensão do arquivo para ordenação.

$ dir

O mesmo que ls

$ vdir

O mesmo que ls -l

*Diretório corrente é o diretório que você entrou por último.

$ tree
Lista diretórios em formato de árvore.

mkdir

A função desse comando é criar diretórios.

$ mkdir <diretório>

$ mkdir -p <diretório>/<diretório>/<diretório>/<diretório>

Cria cadeia de parentesco.

Exemplo:

$ mkdir -p avo/pai/filho/neto

O diretório pai de neto é filho.
O diretório pai de filho é pai.
O diretório pai de pai é avo.

Essa opção **-p** também serve para esconder saída de erro, por exemplo:

Se eu tiver um diretório que já existe, mas eu não me lembro dele e vou criar o mesmo diretório, a mensagem de erro será suprimida.

Veja:

```
$ pwd
/home/leo/gnu
$ ls
linux/
$ mkdir linux
mkdir: não é possível criar o diretório `linux': Arquivo existe
$ mkdir -p linux
Não aparece mensagem de erro
$ ls
linux/
$ cd linux/
$ ls
linus
```

O interessante de se fazer isso é que não perdemos os arquivos que estavam nesse diretório antes de tentarmos criar ele novamente. Note que antes de eu criar novamente o diretório "linux" ele já existia e tinha um arquivo chamado "linus". Com o comando -p "recriei" o diretório, mas não perdi seu conteúdo antigo. Na verdade, com a opção -p o mkdir não faz nada se um diretório existir, ou seja, mantém tudo como está.

touch

Cria um arquivo vazio e altera data de modificação ou criação de um arquivo qualquer.

$ touch 4linux.txt

$ touch -t YYYYMMDDhhmm 4linux.txt

A opção **-t** altera a data do arquivo, observe que é na ordem descrescente, primeiro ano, depois mês, dia, hora e minuto.

Exemplo:

```
$ touch 4linux.txt
$ ls -l
-r8w-r--r--  1 leo users 0 2006-07-12 21:00 4linux.txt
$ touch -t 200607132100 4linux.txt
$ ls -l
-rw-r--r--  1 leo users 0 2006-07-13 21:00 4linux.txt
```

Mudei apenas o dia do mês.

clear

Limpa a tela.

$ clear

Trabalhando com entrada e saída de dados

Esta parte é extremamente importante, pois se trabalha bastante com isso.

Por default(padrão), a **entrada do Shell é o *teclado***, a **saída a *tela***, e o**s erros a tela também.**

Entrada de dados é representada por ***stdin;***
Saída de dados é representada por ***stdout;***
Saída de erros é representada por ***stderr;***

Mas isso pode ser mudado com o uso de caracteres de redirecionamento, veja abaixo:

Para mudar saída padrão:

> Redireciona a saída em um arquivo apagando o conteúdo anterior(se existir);
>> Redireciona a saída no final de um arquivo, preservando-o;
2> Faz o mesmo que o **>** mas acrescenta os erros da saída;
2>> Faz o mesmo que o **>>** mas acrescenta os erros da saída;

Para mudar entrada padrão:

< Indica para o Shell que a entrada não será o teclado;
<< Serve para indicar o escopo de um programa (rótulo);

Comandos auxiliares:

| (pipe, pronuncia-se paipe) Serve para canalizar saída de dado para outro comando;
tee Serve para canalizar saída de dado para um arquivo;
& Substitui o 2>>

Atenção: Para seguir os exemplos abaixo, abra um shell e crie um diretório chamado "shell" sem seu home e nele acrescente os arquivos script1,script2,script3.

Exemplo 1 (>,>>,2>>):

Observação: O diretório **papel/** inexiste.

$ ls shell/ papel/ > log_ls.txt

O comando **ls** listará os diretórios **shell/** e **papel/** e redirecioná a saída para o arquivo log_ls.txt.

Durante a execução do comando, será exibida a seguinte mensagem:

"/usr/bin/ls: papel/: Arquivo ou diretório não encontrado" (Saída de erro), como usamos o ">" ao invés de "2>" (para erro), o que ficou no arquivo foi só a saída certa.

Para ver o conteúdo do log_ls.txt (a saída de erro não apareceu nele), faça:

$ cat log_ls.txt

Para acrescentar a saída com erros, mude o comando para:

$ ls papel/ 2>> log_ls.txt

Exemplo 2 (pipe e tee):

$ ls shell/ | sort | tee listagem.txt

Este comando lista o conteúdo do diretório "shell" canalizando sua saída para o comando "sort", que ordena os arquivos por ordem alfabética, canalizando sua saída para o comando "tee" que canaliza toda a saída para o arquivo "listagem.txt".

cat

Esse comando serve para visualizar e concatenar arquivos.

```
$ cat 4linux.txt
Your Intelligence in Linux

$ cat HackerTeen.txt
Profissionalizando jovens para o bem

$ cat 4linux.txt HackerTeen.txt > Ensino.txt
$ cat Ensino.txt
Your Intelligence in Linux
Profissionalizando jovens para o bem
```

O que fizemos aqui foi jogar o conteúdo dos arquivos 4linux.txt e HackertTeen.txt para o arquivo Ensino.txt

tac

É o inverso de cat.

Exibe na tela do fim para o começo.

$ tac <arquivo>

mv

Esse comando tem duas funções, a primeira é mover arquivos, a segunda é modificar o nome de um arquivo.

```
$ ls paises
argentina/  brasil/
$ cd argentina/
$ ls
buenosaires/  saopaulo/ <- Opa, essa cidade é no brasil
$ mv saopaulo/ /paises/brasil/
$ ls /paises/brasil/
saopaulo
```

Agora, vamos corrigir essa palavra, pois o nome dessa cidade tem acento:

```
$ mv saopaulo sãopaulo
$ ls /paises/brasil
sãopaulo
```

rm

Sua função é remover um arquivo ou diretório.

$ rm -rf <diretório>

Remove o diretório recursivamente de modo forçado.

$ rm -i <arquivo>

Essa opção pede confirmação ao usuário na hora de excluir o arquivo.

$ rm -v <arquivo>

Ativa o modo verbose, ele fala tudo o que vai sendo removido.

rmdir

Esse comando remove diretórios vazios.

$ rmdir <diretório>

$ rmdir -p <diretório-pai><diretório-desejado>

Remove o diretório desejado e seu diretório pai.

Obs: Só será possível remover o diretório-pai se você estive fora do mesmo, ou seja, você deve estar em outro diretório que não seja o que você vai remover.

file

Esse comando diz que tipo é o arquivo.

$ file arquivo

Exemplo:

```
$ file imagem
imagem: JPEG image data, JFIF standard 1.01
```

echo

Escreve uma mensagem na tela.

$ echo "Mensagem que eu quero jogar na tela"

Exemplo:

$ echo "Seja bem vindo" > /etc/issue

head

Mostra as 10 primeiras linhas de um arquivo (cabeçalho).

$ head <arquivo>

$ head -n X <arquivo>

Mostra o número X de linhas iniciais de um arquivo.

Exemplo:

$ head -n 3 <arquivo>

nl

Exibe o arquivo com linhas numeradas.

$ nl <arquivo>

wc

É um contador de palavras, letras, caracteres.

$ wc -l <arquivo>

Conta linhas.

$ wc -w <arquivo>

Conta palavras.

$ wc -c <arquivo>

Conta os bytes do arquivo.

tail

Mostra a quantidade de linhas determinadas de forma decrescente.

$ tail <arquivo>

$ tail -n 3 <arquivo>

Mostrará as 3 últimas linhas do arquivo.

$ tail -F <arquivo_de_log>

Excelente para monitoração de logs em tempo real.

Exemplo:

$ tail -F /var/log/messages

sort

Sua função é ordenar o arquivo na forma crescente.

$ sort <arquivo>

$ sort -r <arquivo>

Faz o inverso, ou seja, na forma descrescente.

df

Exibe a quantidade de disco vazio.

$ df <arquivo/diretório/partição>

$ df -h <arquivo/diretório/partição>

Aproxima para a unidade de medida mais próxima, mais legível para o ser humano.

$ df -k <arquivo/diretório/partição>

Mostra em kilobytes.

$ df -m <arquivo/diretório/partição>

Mostra em megabytes.

du

Exibe a quantidade de disco usado.

$ du <arquivo/diretório/partição>

$ du -h <arquivo/diretório/partição>

Aproxima para a unidade de medida mais próxima, mais legível para o ser humano.

$ du -b <arquivo/diretório/partição>

Mostra em bytes.

$ du -k <arquivo/diretório/partição>

Mostra em kilobytes.

$ du -m <arquivo/diretório/partição>

Mostra em megabytes.

$ du -l <arquivo/diretório/partição>

Mostra a quantidade de links que arquivo/diretório/partição tem.

$ du -s <arquivo/diretório/partição>

Modo silencioso, ou seja, não mostra subdiretórios.

date

Exibe data e hora ou os altera.

Exemplo:

```
$ date
Qui Jul 13 10:05:04 BRT 2006
```

Alterando data e hora:

 # date MMDDhhmmYYYYY

MM mês
DD dia
hh hora
mm minuto
YYYY ano

Opções	***Função***
%a	Abrevia o nome do dia da semana
%A	Mostra o nome da semana por extenso
%b	Abrevia o nome do dia do mês
%B	Mostra o nome do mês por extenso
%c	Mostra o nome do dia e mês abreviados
%d	Mostra o mês em formato numérico
%D	Mostra a da no formato mmddyy
%y	Mostra os 2 últimos dígitos do ano
%Y	Mostra os 4 dígitos do ano
%x	Mostra data no formato DD-MM-YYYY

Exemplo:

```
# date +%x
14-07-2006
```

hwclock

Permite visualizar data/hora do relógio do hardware.

 # hwclock

clock

Permite visualizar data/hora do relógio do hardware.

 # clock

time

Exibe a quantidade de tempo que um comando qualquer leva para realizar sua tarefa.

$ time <comando>

Exemplo:

```
$ time ls
real    0m0.133s
user    0m0.000s
sys     0m0.008s
```

timeconfig

Define o fuso horário para o sistema.

 # timeconfig

setclock

Sua função é ajustar o relógio do hardware a partir do horário do sistema.

 # setclock

uptime

Exibe a quantidade de tempo desde a última reinicialização do sistema.

$ uptime

exit

Fecha uma sessão de login.

$ exit

su

Troca de usuários.

$ su (sem nada, vira root)

$ su <usuário>

reboot / init 6 / telinit 6

Reinicia a máquina.

 # reboot
 # init 6
 # telinit 6

shutdown

Desliga ou reinicia a máquina.

 # shutdown <opções> <hora> <mensagem>

 # shutdown -h <tempo> ou <horário> (em minutos)

Exemplos:

 # shutdown -h now

Desliga a máquina imediatamente.

 # shutdown 09:00 A manutenção do servidor será iniciada às 09:00

 # shutdown -h 30

Desliga a máquina após meia-hora.

 # shutdown -r now

Reinicia a máquina imediatamente.

 # shutdown –r 30

Reinicia a máquina após meia-hora.

 # shutdown -c <Mensagem de aviso>

Cancela o desligamento.

Dica:

Você pode permitir que usuários comuns desligem a máquina.

Para isso faça:

1° passo: Crie um arquivo chamado **shutdown.allow** em /etc/

vi /etc/shutdown.allow

2° passo: Coloque nesse arquivo o nome dos usuários que terão o poder de desligar a máquina

usuariox
usuarioy

3° passo: Editar o arquivo /etc/inittab, a linha que se refere ao shutdown deve ficar assim:

ca::ctrlaltdel:/sbin/shutdown **-a** -h now

A opção **-a** faz o /etc/inittab buscar informações no **/etc/shutdown.allow.**

Para o usuário normal desligar o sistema, basta esse digitar CTRL+ALT+DEL. Atenção, esse comando só vai funcionar se tiver um usuário do shutdown.allow ou root logado.

É comum você testar com um usuário normal que não tenha permissão para desligar a máquina em shutdown.allow e a máquina desligar assim mesmo. Isso acontece pois há algum usuário que tem essa permissão logado na máquina.

init 0 ou **telinit 0** ou **halt**

Desliga a máquina.

 # init 0
 # telinit 0
 # halt

uname

Mostra detalhes do sistema, como kernel, arquitetutura da máquina, nome do sistema etc.

$ uname -a

Mostra tudo.

$ uname -r

Mostra versão do kernel.

$ uname -m

Mostra a arquitetura da máquina.

$ uname -n

Mostra o hostname.

$ uname -p

Mostra o tipo do processador

$ uname -v

Mostra a data da versão do kernel.

$ uname -o

Mostra o nome do sistema operacional.

$ uname -s

Mostra o nome do kernel.

arch

Mostra a arquitetura da máquina.

$ arch

whoami

Informa que usuário sou.

$ whoami

w

Visualizar quem está logado no sistema, mostrando o que está sendo feito, o tempo de uso do processador etc.

$ w

who

Exibe quem está logado, em qual TTY, e a data e hora que logaram.

last

Fornece uma listagem dos últimos usuários que estiveram logados no sistema.

history

Mostra os últimos 1000 comandos executados em modo texto.

$ history

dmesg

Mostra as mensagens do kernel.

 # dmesg

Melhore a visualização:

 # dmesg | more

whereis

Localiza o binário, o código-fonte e o manual.

$ whereis <comando>

$ whereis -b <comando>

Localiza somente o binário.

$ whereis -m <comando>

Localiza somente os manuais.

$ whereis -s <comando>

Localiza somente o código-fonte.

Exemplo:

```
$ whereis ls
ls: /bin/ls /usr/bin/ls /usr/man/man1/ls.1.gz /usr/share/man/man1/ls.1.gz
```

locale

Exibe informações sobre a localidade que está configurada no sistema.

Exemplo:

```
$ locale
LANG=pt_BR
LC_CTYPE="pt_BR"
LC_NUMERIC="pt_BR"
LC_TIME="pt_BR"
LC_COLLATE="pt_BR"
LC_MONETARY="pt_BR"
LC_MESSAGES="pt_BR"
LC_PAPER="pt_BR"
LC_NAME="pt_BR"
LC_ADDRESS="pt_BR"
LC_TELEPHONE="pt_BR"
LC_MEASUREMENT="pt_BR"
LC_IDENTIFICATION="pt_BR"
LC_ALL=pt_BR
```

locate/slocate

Busca uma palavra numa base de dados que é sempre atualizada com o comando **updatedb**. É considerado mais fácil e rápido que o **find**.

$ locate <palavra>

Exemplo:

Se quero procurar no sistema um arquivo que tenha a palavra "copy", faço isso:

```
$ locate copy
/usr/lib/python2.4/copy.py
/mnt/windows/WINDOWS/system32/diskcopy.dll
/usr/bin/mcopy
```

updatedb

Atualiza base de dados do comando **locate/slocate.**

```
# updatedb
```

which

Informa o caminho de um arquivo executável.

```
$ which ls
/usr/bin/ls
```

find

Procura arquivos ou diretórios.

$ find <diretório> <expressão>

Exemplos:

$ find /etc **-name** lilo.conf
A busca vai ser feita no diretório raiz "/", buscando o arquivo "lilo.conf"

$ find /usr/local **-name** "*.html" **-type f -print**

Este exemplo busca em /usr/local todos os arquivos com extensão html e os mostra na tela.

$ find /usr/local **-name** "*.html" **-type f**
-exec chmod 644 **{} \;**

Este exemplo busca em /usr/local todos os arquivos com extensão html e os mostra na tela, além de mudar suas permissões.

grep

Realiza uma filtragem de uma saída de comando, ou de uma string em um arquivo texto

Opções:

-i => filtra uma string, independente do fato de ser maiúscula ou minúscula

-v => filtra tudo da saída ou arquivo, menos a string solicitada

Exemplos:

$ grep -i gnu /etc/issue => Serão filtradas todas as ocorrências da palavra gnu no arquivo (maiúscula ou minúscula)

$ ls /etc **| grep** **^**in => Filtra no diretório /etc tudo o que começar com a sílaba in (o **^** indica a partir de onde começar)

pgrep

Busca o PID de um programa em execução através do nome.

$ pgrep <programa>

Exemplo:

```
$ pgrep firefox
7079
```

more

Faz paginação, para paginar use <**ENTER**>.

$ more <arquivo texto>

less

Faz paginação, mas permite o uso das setas para movimentação.

$ less <arquivo texto>

free

Fornece dados sobre o usuário da memória.

$ free

Exemplo:

```
$ free
             total       used       free     shared    buffers     cached
Mem:        244700     236752       7948          0       7996      83068
-/+ buffers/cache:     145688      99012
Swap:       506008     120484     385524
```

alias

Cria um atalho para um comando.

$ alias <comando_original>=”nome_desejado”

Exemplo:

$ alias listar=”ls”

cut

Sua função é extrair campos de arquivos.

$ cut -c<caracteres> <arquivo>

Exemplos:

```
$ cat agenda.txt
Carol Maria     333-3333
Katia Maria     444-4444
```

```
Paula Pereira   555-555

$ cat agenda.txt | cut -c-5
Carol
Katia
Paula

$ cat agenda.txt | cut -f1 -d " "
Carol
Katia
Paula

$ cat /etc/passwd
root:x:0:0::/root:/bin/bash

$ cat /etc/passwd | cut -f1 -d :
root
```

pr

Quebra arquivos dentro de múltiplas páginas de saída, tipicamente usado para impressão.

```
$ pr <arquivo>
```

Exemplo:

```
$ pr -l 50 arquivo.txt | more
```

Fixa um máximo de 50 linhas por página.

tr

Sua função é transcrever uma cadeia de caracteres.

```
$ tr <antes> <depois>
```

Exemplos:

```
$ cat teste.txt
4linux
$ cat teste.txt | tr [a-z] [A-Z]
4LINUX

$ cat teste.txt
4linux

$ cat teste.txt | tr 4 ' '
 linux
```

Observação: Como a saída padrão é a tela, o arquivo continuará como antes. Para mudar isso é só mudar a saída padrão:

```
$ cat teste.txt | tr 4 ' ' | tee teste.txt
```

wget

Realiza download via linha de comando.

```
$ wget <URL>
```

Exemplo:

```
$ wget -c ftp://ftp.linux.ncsu.edu/pub/fedora/linux/core/5/i386/iso/FC-5-i386-disc4.iso
```

A opção **-c** permite para o download com **CTRL**+**C** e depois continuar de onde parou.

lpq

Exibe a fila de impressão.

```
$ lpq
hp is ready and printing
Rank    Owner   Job     File(s)                         Total Size
active  leo     69      leo.txt                         10146816 bytes
```

lprm

Exclui um arquivo da fila de impressão.

$ lprm <Job>

Exemplo:

```
$ lprm 69
$ lpq
hp is ready
no entries
```

lpr

Imprime.

$ lpr <arquivo.txt>

od

Conversão de letras ou números.

$ od -x <arquivo com números>

Converte para hexadecimal.

$ od -o <arquivo com números>

Converte para octal.

reset

Usado especialmente para quando nosso terminal fica bagunçado.

Exemplo:

$ cat /bin/ls

Com esse comando, nosso terminal vai ficar completamente bagunçado e com letras estranhas, para limpar isso basta digitar:

$ reset

cmp

Compara dois arquivos.

$ cmp <arquivo1> <arquivo2>

fmt

Reformata parágrafos em arquivos de textos corrigindo margem.

$ fmt <arquivo.txt>

ln

Sua função é criar links. O link é um mecanismo que faz referência a outro arquivo ou diretório em outra localização. Fazendo uma analogia com Windows seria um atalho.

$ ln <opções> <origem> <link>

Existe dois tipos de links. São eles:

Link simbólico (symlinks)

O link simbólico cria um arquivo especial no disco (do tipo link) que tem como conteúdo o caminho para chegar até o arquivo alvo (isto pode ser verificado pelo tamanho do arquivo do link).

Criando link simbólico:

$ ln -s <origem> <link>

Exemplo:

 # ln -s /dev/ttyS1 /dev/modem

Cria o link /dev/modem para o arquivo /dev/ttyS1

Hard link

O hardlink faz referência ao mesmo inode do arquivo original, desta forma ele será perfeitamente idêntico, inclusive nas permissões de acesso, ao arquivo original. É utilizado para dar nomes diferentes a um mesmo arquivo.

Hard links não são tão comuns quanto links simbólicos, mas podem ser úteis quando se deseja ter a segurança de que se o arquivo original for apagado, o link continue sendo acessível.

*Um **inode** é um identificador (endereço físico) único que um arquivo recebe.

As limitações dos hard links são:

1) O arquivo e o hard link que aponta pra ele devem obrigatoriamente estar localizados no mesmo sistema de arquivos já que o hard link aponta para um endereço físico (inode) e não se pode garantir que estes endereços sejam únicos em tipos diferentes de sistemas de arquivos.

2) A outra limitação é que um hard link não pode apontar para um diretório

Criando hard link:

$ ln <origem> <link>

Observações:

- Se for usado o comando rm com um link, somente o link será removido.
- Se for usado o comando cp com um link, o arquivo original será copiado ao invés do link.

- Se for usado o comando mv com um link, a modificação será feita no link.
- Se for usado um comando de visualização (como o cat), o arquivo original será visualizado.

Ao contrário dos links simbólicos, não é possível fazer um hardlink para um diretório ou fazer referência a arquivos que estejam em partições diferentes.

Dica de como descobrir se um link é hard:

```
$ ls -di dvd /mnt/dvd
 480057 dvd@  1272591 /mnt/dvd/
```

A opção **-i** do **ls** mostra o inode, nesse caso é um link simbólico pois os inodes são diferentes.

Vamos criar um arquivo vazio e depois um hard link:

```
$ touch arquivo_original

$ ln arquivo_original arquivo_link
$ ls -i arquivo*
481340 arquivo_cópia  481340 arquivo_original
```

Veja que o inode é o mesmo.

```
$ ls -l arquivo*
-rw-r--r--  2 leo users  0 2006-07-16 19:20 arquivo_cópia
-rw-r--r--  2 leo users  0 2006-07-16 19:20 arquivo_original
```

Note que o primeiro caracter que se refere ao link "arquivo_cópia" não está o acusando como link.

Se fosse um link simbólico, veja:

```
$ ls -l arquivo*

lrwxrwxrwx 1 leo users 0 2006-07-16 19:20 arquivo_cópia ->
arquivo_original
```

O **l** indica um link simbólico.

startx

Sua função é iniciar o X-Windows (interface gráfica) do Linux;

```
$ startx
```

diff

Compara o conteúdo de dois arquivos e lista quaisquer diferenças.

Exemplo:

```
$ diff arquivo1 arquivo2 > diferenças
```

Compara o **arquivo1** e **arquivo2** e joga a saída (as diferenças) para o arquivo **diferenças.**

join

Sua função é unir registros de dois arquivos de texto.

$ join <opções> <arquivoX> <arquivoY>

Exemplo:

```
$ cat estado
001:Estado:São Paulo
002:Estado:Goiás
003:Estado:Rio de Janeiro

$ cat capital
001:Capital:São Paulo
002:Capital:Goiânia
003:Capital:Rio de Janeiro

$ join -t: -o 1.1 1.2 1.3 2.1 2.2 2.3 estado capital
001:Estado:São Paulo:001:Capital:São Paulo
002:Estado:Goiás:002:Capital:Goiânia
003:Estado:Rio de Janeiro:003:Capital:Rio de Janeiro
```

A opção **-t** indica o separador de campos, que no nosso caso é ":".

A opção **-o** monta o registro de acordo com a ordem **X.Y**

Onde **X** indica o primeiro arquivo, e **Y** o segundo indica o campo.

Uni os campos 1, 2 e 3 dos arquivos estado e capital.

man

Exibe manual.

$ man <comando>

Exemplo:

$ man ls

apropos

Exibe **todas** as páginas do manual de um certa string.

Exemplo:

```
$ apropos passwd
passwd               (1)  - change user password
passwd               (5)  - password file
```

* Não coloquei toda a saída, mas é um grande leque de opções.

Daí é só usar:

$ man 1 passwd (explica o comando)
$ man 5 passwd (explica sobre o arquivo /etc/passwd)

whatis

Exibe as seções do man para um comando.

Exemplo:

```
$ whatis passwd
passwd               (1)  - change user password
passwd               (5)  - password file
passwd [ssl_passwd]  (1)  - compute password hashes
```

makewhatis

Atualiza a base de dados do comando whatis.

```
# makewhatis
```

--help

Exibe as opções rápidas de um comando.

Exemplo:

```
$ ls --help
```

uniq

Sua função é trabalhar com registros duplicados.

Exemplo:

```
$ cat texto.txt
Brasil
Brasil
Argentina
EUA

$ uniq texto.txt
Brasil
Argentina
EUA
```

paste

Faz o inverso do comando cut, ou seja, enquanto o cut separa pedaços, o paste os uni.

Exemplo:

```
$ cat /etc/passwd
root:x:0:0::/root:/bin/bash
leo:x:1000:100:,,,:/home/leo:/bin/bash

$ cat /etc/passwd | cut -f1 -d: > /tmp/logins
$ cat /etc/passwd | cut -f3 -d: > /tmp/uid
$ paste /tmp/logins /tmp/uid
root    0
leo     1000
```

rev

Inverte a sequência de letras em um arquivo.

Exemplo:

```
$ cat texto.txt
Brasil
Brasil
Argentina
EUA

$ rev texto.txt
lisarB
lisarB
anitnegrA
AUE
```

sed

O *SED* é um editor de textos **não interativo**.. Vem do inglês "Stream EDitor", ou seja, editor de fluxos (de texto).

A característica principal do *SED* é poder editar arquivos automaticamente.

Então sempre que você precisar fazer alterações sistemáticas em **vários** arquivos, o *SED* é uma solução eficaz.

Por exemplo, você tem um diretório cheio de relatórios de vendas, e descobriu que por um erro na geração, todas as datas saíram erradas, com o ano de **1999** onde era para ser **2000**. Num editor de textos normal, você tem que abrir os relatórios um por um e alterar o ano em todas as ocorrências.

Certo, isso não é tão complexo se o editor de textos possuir uma ferramenta de procura e troca, também chamado de substituição.

Mas então suponhamos que o erro da data não seja o ano, e sim o **formato**, tendo saído como **mm/dd/aaaa** quando deveria ser **dd/mm/aaaa**. Aqui não é uma substituição e sim uma troca de lugares, e uma ferramenta simples de procura e troca não poderá ajudar.

Esse é um caso típico onde o *SED* mostra seu poder: alterações complexas em vários arquivos.

Utilizando o *SED*, a solução para este problema (que veremos adiante) é até simples, bastando definir uma série de regras de procura e troca, e o programa se encarregará de executá-las e arrumar os relatórios.

O *SED* não é uma linguagem de programação completa, pois não possui variáveis, funções matemáticas, interação com o sistema operacional, entre outras limitações. Mas bem, ele é um **manipulador de texto** e não uma linguagem de uso geral.

Sintaxe:

$ sed expressão regular <arquivo>

Exemplo:

```
$ cat sistemas
ATENCAO, LEIA O TEXTO COM CUIDADO
APRENDA SED
Ele é usado muito em Linux
APROVEITE!!!
```

```
$ sed 's/Linux/Unix <- ACHEI!!!!/' sistemas
ATENCAO, LEIA O TEXTO COM CUIDADO
APRENDA SED
Ele é usado muito em Unix <- ACHEI!!!!
APROVEITE!!!
```

Substitui a palavra Linux por Unix <- ACHEI!!!!

info

Muitos programas GNU são documentados através de *arquivos info* ao invés de páginas de manual. Estes arquivos incluem informação detalhada sobre o programa, opções e exemplos de uso e estão disponíveis através do comando info

$ info <comando/programa>

Exemplo:

$ info ls

split

O comando split é usado para dividir um arquivo em pedaços menores, muito útil quando se tem dois disquetes e um arquivo de 2 Mb.

Exemplo:

split --lines=50 arquivo.txt

Isso irá gerar X arquivos com 50 linhas cada.

split --bytes=1048576 backup.tar.gz

Que irá gerar *X* arquivos com 1 Mb cada.

Onde:

1048576=1024*1024*1

Ou seja, 1Mb corresponde a 1048576 bytes.

type

Mostra o caminho completo do comando.

```
$ type shutdown
shutdown is /usr/bin/shutdown
```

expand

Converte tabs em espaços.

unexpand

Converte espaços em espaços.

Bibliografia

Guia Foca Linux: http://focalinux.cipsga.org.br/

Apostila da IBM LPI certification, autor Daniel Robbins.

Programação Shell Linux 3ª edição, autor Julio Cezar Neves.

http://aurelio.net/sed/sed-HOWTO/sed-HOWTO-2.html#ss2.1

http://www.debian.org/doc/index.pt.html

http://vivaolinux.com.br/dicas/verDica.php?codigo=17

Man Pages.