COMANDOS BÁSICOS DO SISTEMA Linux

Obs: testar todos os comandos observando suas ações e anotar os resultados obtidos.

Comandos em Linux possuem algumas características particulares. Eles podem ser controlados por opções e devem ser digitados em letras minúsculas.

Atenção: em todos os comandos listados **NÃO DIGITAR O $**.

1) Comando **pwd** : Exibe o caminho do diretório corrente.

Sintaxe: pwd

2) Comando **cd** : Muda o diretório de trabalho corrente.

Sintaxe: cd <diretório>

onde (diretório) é o nome do diretório para o qual você deseja mudar. O símbolo "." refere-se ao diretório corrente e o símbolo ".." refere-se ao "diretório-pai". Para mover para um "diretório-pai", ou seja, um diretório acima do que você está, use o comando :

```
$ cd ..
```

Atenção: Note o espaço entre "cd" e ".."

Você também pode usar nomes-de-caminho (*pathnames*) como argumento para o comando "cd". Por exemplo, no comando

```
cd /diretorio1/diretorio2
```

você será posicionado diretamente em "diretório2". O uso de "cd" sem nenhum argumento fará com que você retorne para o seu "*home-directory*" .

3) Comando **ls** : Exibe informações sobre arquivos nomeados e diretórios, é usado para visualizar o conteúdo de um diretório.

Sintaxe: ls [opções] <diretório>

Quando executado sem qualquer parâmetro, mostra o conteúdo do diretório corrente. Assim, a linha de comando:

```
$ ls
```

mostra o conteúdo do diretório corrente naquele momento.

Como na maioria dos comandos LINUX, "ls" pode ser controlado por opções que começam com um hífen (-). Tenha sempre o cuidado de deixar um espaço antes do hífen. Uma opção bastante útil é "-a" (que vem do inglês *all*, tudo), e irá mostrar detalhes que você nunca imaginou sobre o seu diretório. Por exemplo,

```
$ ls -a
```

mostra todo o conteúdo do diretório (incluindo arquivos ocultos), que será exibido da seguinte forma:

```
.                .bacshrc          .fvwmrc
..               .emacs            .xinitrc
.bash_history    .exrc
```

Aqui, o ponto simples (.) refere-se ao diretório corrente, e o ponto duplo (..) refere-se ao diretório imediatamente acima dele. Mas o que são estes outros arquivos que se iniciam com um ponto? Eles são chamados arquivos escondidos (ocultos). A colocação do ponto na frente de seus nomes os impede de serem mostrados durante um comando "ls" normal.

Outra opção bastante usada é "-l" (que vem do inglês *long*). Ela mosta informação extra sobre os arquivos. Assim, o comando

```
$ ls -l
```

mostra, além do conteúdo do diretório, todas os detalhes sobre cada arquivo pertencente a ele. Por exemplo, suponha que você tenha executado este comando e na tela apareceu algo assim:

```
colunas: 1      2      3      4          5      6    7    8      9
-rw-r--r--      1     xyz   users      2321    Mar   15 1994 Fontmap
-rw-r--r--      1     xyz   users      14567   Feb    3 1995 file003
drwxr-xr-x      2     xyz   users      1024    Apr   23 1995 Programs
drwxr-xr-x      3     xyz   users      1024    Apr   30 1995 bitmaps
```

lendo da esquerda para direita, na **coluna 1** o primeiro caractere indica se o arquivo é um diretório (d) ou um arquivo comum (-). Em seguida temos as permissões de acesso ao arquivo, sendo as três primeiras referentes ao proprietário, as seguintes ao grupo e, por último, aos demais usuários. A **segunda coluna** mostra o número de *links* que o arquivo possui. A **terceira coluna** mostra o proprietário do referido arquivo, neste caso, o usuário cujo *user name* é "xyz". Na **coluna 4** é mostrado o grupo ao qual pertence o proprietário do arquivo (no exemplo temos o grupo "users").

Na **quinta coluna** temos o tamanho do arquivo em bytes. Por fim, na **sexta e sétima colunas**, temos a data da última modificação feita no arquivo e o nome do arquivo, respectivamente.

Vale lembrar que várias opções podem ser usadas de forma composta. Por exemplo, podemos executar o comando

```
$ ls -la
```

e este mostrará todos os detalhes que as opções "-l" e "-a" dispõem.

4) Comando **clear** : Limpa a tela.

5) Comando **mkdir** : Comando usado para a criação de novos diretórios.

Sintaxe : mkdir <diretório 1> <diretório 2> … <diretório n>

onde <diretório 1> até <diretório n> são os diretórios a serem criados. No exemplo

```
$ mkdir novo
```

é criado um diretório chamado "novo" dentro do diretório corrente.

As entradas padrão em um diretório (por exemplo, os arquivos ".", para o próprio diretório, e ".." para o diretório pai) são criadas automaticamente. A criação de um diretório requer permissão de escrita no diretório pai.

O identificador de proprietário (*owner id*), e o identificador de grupo (*group id*) dos novos diretórios são configurados para os identificadores de proprietário e de grupo do usuário efetivo, respectivamente.

6) Comando **cat** : Oficialmente usado para concatenar arquivos, é também usado para exibir no terminal todo o conteúdo de um arquivo de uma só vez, sem pausa.

Sintaxe: cat < arquivo1 > < arquivo2 >... < arquivo n >,

onde <arquivo1> até <arquivo n> são os arquivos a serem mostrados. “cat” lê cada arquivo em sequencia e exibe-o na saída padrão. Deste modo, a linha de comando

```
$ cat arquivo1.txt
```

exibirá o conteúdo do “arquivo1.txt” no terminal. A linha de comando abaixo:

```
$ cat arquivo1.txt arquivo2.txt > arquivo3.txt
```

concatenará os arquivos “arquivo1.txt” e “arquivo2.txt” escrevendo o resultado em “arquivo3.txt”. O símbolo “>” é usado para redirecionar a saída do terminal para dentro de um arquivo (nesse caso o “arquivo3.txt”). Este símbolo tem caráter 'destrutivo', ou seja, se o arquivo “arquivo3.txt” já existir, o comando acima escreverá por cima do conteúdo dele.

Se ao invés de concatenar você quiser apender (adicionar) um conteúdo no final de um arquivo já existente, substitua na linha de comando o símbolo “>” por “>>”.

7) Comando **cp** : Copia arquivos para um outro arquivo ou diretório.

Sintaxe: cp <arquivo1> <arquivo2> ... <arquivo n> <destino>

onde <arquivo1> até <arquivo n> são os arquivos a serem copiados e <destino> é o arquivo ou o diretório para onde os arquivos serão copiados. O(s) arquivo(s) fonte(s) e o <destino> não podem ter o mesmo nome. Se o arquivo-destino não existe, “cp” criará um arquivo com o nome especificado. Se o arquivo-destino já existia antes e não for um diretório, o comando “cp” escreverá o novo conteúdo por cima do antigo.

```
$ cp -r dirtemp dirtemp1
```

Este comando copia todos os arquivos e subdiretórios dentro do diretório “dirtemp” para um novo diretório chamado “dirtemp1”. Esta é uma cópia recursiva, como designado pela opcão “-r”. Se você tentar copiar um diretório sem usar esta opcão, você verá uma mensagem de erro.

Na linha de comando

```
$ cp arquivo1.txt arquivo2.tx
```

o “arquivo1.txt” é o arquivo de entrada da operação de cópia, e o “arquivo2.txt” é a saída produzida. Os arquivos de origem e destino devem ter nomes distintos, pois se tiverem o mesmo nome, então será emitida uma mensagem de diagnóstico indicando que são idênticos, e o arquivo não será copiado sobre si mesmo. Se o “arquivo2.txt” já existia, seu conteúdo será substituído pelo conteúdo do “arquivo1.txt”.

No comando abaixo:

```
$ cp [-ipr] <arquivo 1> <arquivo n> <destino>
```

os parâmetros:

-i : Pede confirmação para cada arquivo a ser copiado.

-p : Mantém na cópia as datas de modificação e permissões do arquivo original.

-r : Copia recursivamente arquivos e diretórios. Neste caso <destino> deve se referir a um diretório.

Exemplo: `$ cp -ipr dirtemp dirtemp1`

8) Comando **rm** : Este comando é usado para apagar arquivos.

Sintaxe: rm <arquivo 1> <arquivo 2> ... <arquivo n>

onde <arquivo 1> até <arquivo n> são os arquivos a serem apagados.

Se um arquivo não possuir permissão de escrita e a saída-padrão for um terminal, todo o conjunto de permissões do arquivo será exibido, seguido por um ponto de interrogação. É um pedido de confirmação. Se a resposta começar com "y" ("yes" = sim), o arquivo será apagado, caso contrário ele será mantido no sistema.

*Opções:

-f : Remove todos os arquivos (mesmo se estiverem com proteção de escrita) em um diretório sem pedir confirmação do usuário.

-i : Esta opção pedirá uma confirmação do usuário antes de apagar o(s) arquivo(s) especificado(s).

-r : Opcão recursiva para remover um diretório e todo o seu conteúdo, incluindo quaisquer subdiretórios e seus arquivos.

Atenção: É importante lembrar que quando os arquivos são apagados, no sistema Linux, não é possível recuperá-los facilmente.

9) Comando **mv :** Move arquivos (ou diretório) para um outro arquivo ou diretório.

Este comando faz o equivalente a uma cópia seguida pela deleção do original. Pode ser usado para renomear arquivos.

Sintaxe: mv <arquivo 1> <arquivo 2> ... <arquivo n> <destino>

onde <arquivo 1> até <arquivo n> são os arquivos a serem movidos, e <destino> é o arquivo ou o diretório para onde os arquivos serão movidos.

a) Se <destino> não for um diretório, somente um arquivo deverá ser especificado como fonte. Se for um diretório, mais de um arquivo poderá ser especificado.

b) Se <destino> não existir, "mv" criará um arquivo com o nome especificado. Se <destino> existir e não for um diretório, seu conteúdo será apagado e o novo conteúdo será escrito no lugar do antigo. Se <destino> for um diretório, o(s) arquivo(s) será(ão) movido(s) para este diretório.

Atenção: Os arquivos "fonte" e "destino" não precisam compartilhar o mesmo diretório pai.

*Opções:

-i : Com esta opção, "mv" irá perguntar a você se é permitido escrever por cima do conteúdo de um arquivo destino existente. Uma resposta "y" (yes = sim) significa que a operação poderá ser executada. Qualquer outra resposta impedirá que "mv" escreva por cima do conteúdo de um arquivo já existente.

No exemplo

```
$ mv arquivo1.txt dirtemp1
```

o arquivo "arquivo1.txt" foi movido para o diretório "dirtemp1".

10) Comando **file :** Exibe o tipo de um arquivo. Alguns arquivos, tais como arquivos binários e executáveis, por vezes, podem não ser visualizados na tela. O comando "file" pode ser útil se você não tem certeza sobre o tipo do arquivo.

```
$ file                    copyfile $ copyfile: ascii text
```

11) Comando **more**: Exibe o conteúdo de um arquivo, uma tela cheia de cada vez, fazendo normalmente uma pausa após cada tela cheia (quando exibe "`--More--`" na parte de baixo da tela). Ao apertar a tecla "Enter", será exibida uma linha a mais. Ao apertar a tecla "espaço" outra tela cheia será exibida. O caracter "b" reexibe a tela anterior. O caracter "q" provoca a parada de execução do comando more.

Sintaxe: more <arquivo 1> <arquivo 2> ... <arquivo n>

onde <arquivo 1> até <arquivo n> são os arquivos a serem exibidos.

Pode-se procurar por uma palavra (ou uma cadeia de caracteres) em um arquivo. Para isso, pressione o caracter "/", digite a palavra (ou a cadeia de caracteres) a ser procurada e tecle "Enter".

12) Comando **man**: Exibe uma página do manual interno do Linux, para um dado comando ou um recurso.

Sintaxe : man <comando>

onde "comando" é o nome do comando ou recurso que se deseja obter a ajuda.

13) Exemplos 1. O comando abaixo lista os diretórios e arquivos do /.

$ ls /

14)O comando abaixo lista os diret´orios e arquivos do /etc.

$ ls /etc

15)Para listar o conteúdo do / e do /etc, de uma só vez, use:

$ ls / /etc

Liste o conteúdo do diretório /tmp.

16) Exemplo

O comando abaixo listar´a todos os arquivos e diret´orios contidos no barra, incluindo os ocultos.

$ ls -a /

17) COMANDOS

cd [diretório] Exemplos

Para entrar no diretório root, use

$ cd /

Para entrar no diretório /tmp, basta usar o seguinte commando

$ cd /tmp

Para subir um diret´orio acima, use:

$ cd ..

Para voltar ao diretório imediatamente anteriormente acessado, basta usar:

$ cd –

Entre no diretório home do seu usuário ("/home/seu-usuario-aqui").

Agora use o seguinte comando:

$ cd ../../

mkdir (make directory) Cria novos diret´orios (vazios).

Sintaxe básica: $ mkdir [caminho1/diret´orio1] [caminho2/diret´orio2] ...

Exemplos 1. Para criar os diret´orios “Pasta1” e “Pasta2” dentro do diret´orio /tmp, fazemos:

$ mkdir /tmp/Pasta1 /tmp/Pasta2

Naturalmente, se estivéssemos dentro do diretório /tmp, não seria necessário usar o caminho absoluto:

$ pwd
/tmp
$ mkdir Pasta1 Pasta2

rmdir (remove directory) Remove um ou mais diretórios vazios.
Sintaxe básica:
$ rmdir [caminho1/diret´orio1] [caminho2/diret´orio2] ...
Exemplos 1. Para remover os diretórios “Pasta1” e “Pasta2” criados como nos exemplos do comando mkdir, poderíamos usar:
$ rmdir /tmp/Pasta1 /tmp/Pasta2
Exerc´ıcios 1. Vá até seu diretório home e crie um diretório chamado “Teste”. Use o comando ls para ver que o diretório foi criado. Remova o diretório criado e use novamente o comando ls para ver que a pasta foi removida.