**ls**

O comando *ls* exibe arquivos ou o conteúdo de um ou vários diretórios.   
  
Sintaxe:   
  
**$ ls [opções] [arquivo]**   
  
Opções:

* -a: Exibe arquivos ocultos;
* -A: Não exibe os diretórios . e ..;
* --author: Mostra o autor (criador) de cada arquivo;
* -b: Exibe caracteres de escape octais no lugar dos caracteres que não podem ser vistos, como o espaço em branco;
* --block-size=[tamanho]: Exibe o tamanho dos arquivos em múltiplos do número de bytes especificado nesse parâmetro;
* -B: Não exibe arquivos de backup (terminados com ~);
* -c: Lista os arquivos por ordem da data da última modificação;
* -C: Exibe a listagem em colunas;
* --color=[quando]: Controla quando as cores devem ser usadas para distinguir os tipos de arquivos. Os valores aceitos são:
  + never: Não usa cores pra nenhum tipo de arquivo;
  + always: Usar cores para todo tipo de arquivo;
  + auto: Seleciona quais arquivos serão exibidos em cores.
* -d: Exibe o diretório especificado, e não o seu conteúdo;
* -f: Ativa os parâmetros -a e -U e desabilita os parâmetros -l, -s e -t;
* -F: Acrescenta um caracter gráfico ao final de cada arquivo para identificar o seu tipo;
* -G: Não exibe informações dos grupos a que os arquivos pertencem;
* -h: Exibe os tamanhos dos arquivos em uma forma legível (2K, 21M, 1G);
* --si: Semelhante ao -h, mas usa múltiplos de 1000 bytes ao invés de 1024;
* -H: Exibe os arquivos para os quais os links simbólicos apontam, ao invés de listar só o link;
* -i: Exibe o número de índice (I-node) dos arquivos;
* -I: Não exibe entradas que contiverem o padrão informado;
* -k: Equivalente a --block-size=1k;
* -l: Listagem detalhada, com diversas informações sobre os arquivos;
* -L: Quando listar links simbólicos, lista o local para onde o link aponta, e não o link propriamente dito;
* -m: Lista os arquivos em linhas, separando cada item com uma vírgula;
* -n: O mesmo que o parâmetro -l, mas mostra as UID's e GID's ao invés dos nomes dos grupos;
* -o: O mesmo que o parâmetro -l, mas não exibe as informações sobre o grupo;
* -p: Adiciona um caracter para identificar o tipo do arquivo. O mesmo que -F, mas não utiliza o caracter \* (asterisco);
* -Q: Exibe os nomes das entradas entre " (aspas duplas);
* -r: Organiza a lista na ordem inversa;
* -R: Lista recursivamente o conteúdo dos diretórios e subdiretórios do diretório atual;
* -s: Exibe o tamanho de cada arquivo, em múltiplos de blocos (especificados com o parâmetro --block-size=[tamanho]);
* -S: Organiza a lista de acordo com o tamanho do arquivo;
* -t: Lista pela data de modificação;
* -u: Organiza a listagem pela data do último acesso;
* -U: Não organiza a listagem, exibindo os arquivos na seqüência em que estão gravadas no diretório;
* -w: Ajusta o tamanho da tela para o número de colunas especificado;
* -X: Organiza a listagem em ordem alfabética;
* -1: Lista apenas um arquivo por linha;

Em [arquivo], devemos informar quais arquivos (arquivos, diretórios, dispositivos, links, etc.) devem ser listados. Se não for informado nada, será listado o conteúdo do diretório atual (.).   
  
Pode-se também utilizar curingas para filtrar os arquivos que serão listados. Por exemplo, podemos usar ls \*.sxw para listar somente os arquivos terminados em .sxw.

**cd**

O comando *cd*, sigla de change directory (selecionar diretório), serve para acessar um determinado diretório.   
  
Sintaxe:   
  
**$ cd [diretório]**   
  
Lembrando a lista dos diretórios do sistema:

* ..: Diretório acima do atual. Se você estiver no diretório /home/aluno/ e quiser acessar o diretório /home/, digite cd ... Se quiser acessar o diretório /home/davidson/, digite cd ../davidson/;
* ~: Diretório pessoal do usuário atual, ou seja, /home/[usuário]/;
* -: Diretório anterior. Se você estava no diretório /etc/ e mudou para o diretório /home/, digite cd - para voltar ao diretório /etc/.

Se for usado sem parâmetro, ou seja, apenas cd, você será redirecionado para o diretório pessoal do usuário atual.

**mkdir**

O comando *mkdir*, abreviatura de make directory (criar diretório), é usado para criar um novo diretório.   
  
Sintaxe:   
  
**$ mkdir [opções] [novo diretório]**   
  
Opções:

* -m: Especifica as permissões que do novo diretório terá;
* -p: Cria todos os diretórios e subdiretórios necessários;
* -v: Exibe uma mensagem para cada diretório criado.

Em [novo diretório] devemos colocar os diretórios que queremos criar. Não é necessário digitar o caminho completo, caso queiramos criar um diretório dentro do diretório atual. Podemos criar vários diretórios com um único comando, bastando separá-los com espaços. Se quiser que o nome do novo diretório tenha espaços em branco, escreva-o entre aspas duplas, dessa forma:   
  
**$ mkdir /home/davidson/"diretório com espaços em branco"**

**rmdir**

Esse comando é utilizado para apagar um diretório vazio.   
  
Sintaxe:   
  
**$ rmdir [opções] [diretório]**   
  
Opções:

* -p: Remove os diretórios-pai do diretório selecionado, se possível. Assim, o comando rmdir -p a/b/c/ vai apagar o diretório a/b/c/, depois o diretório a/b/ e por fim o diretório a/, se possível;
* -v: Mostra os detalhes da remoção dos diretórios.

Podem ser especificados mais de um diretório por vez. O rmdir só remove diretório vazios.

**pwd**

O *pwd*, sigla de print working directory (exibir diretório de trabalho), exibe o diretório atual. É equivalente a echo $PWD.   
  
Uso:   
  
**$ pwd**

**cat**

O comando *cat* concatena arquivos e imprime na saída padrão (exibe na tela). Em arquivos, usamos o cat para listar seu conteúdo na tela. Com o uso de direcionadores, podemos usá-lo para unir diferentes arquivos em um só, dentre outra funções.   
  
Sintaxe:   
  
**$ cat [opções] [arquivo]**   
  
Opções:

* -b: Numera as linhas, com exceção das linhas em branco;
* -E: Mostra um "$"? (cifrão) para indicar fim de linha;
* -n: Numera todas as linhas, incluindo as em branco;
* -s: Não mostra mais do que uma linha em branco. Se houver duas ou mais linhas em branco consecutivas, elas são truncadas e apenas uma é mostrada;
* -T: Substitui tabulações pelos caracteres "^I"?;
* -v: Substitui os caracteres não imprimíveis por símbolos, exceto tabulações e final de linha.

Exemplos de uso:   
  
**$ cat [arquivo1 arquivo2 arquivo3 ... arquivoN] > [arquivo]**   
  
Isso pode ser usado em qualquer tipo de arquivo, inclusive arquivos binários. É prática comum utilizar isso para juntar arquivos de vídeo grandes, como filmes, que muitas vezes são divididos em várias partes.   
  
Veja no exemplo abaixo, como unir as 2 partes do filme Matrix em um único arquivo:   
  
$ cat the-matrix\_part-1.mpeg the-matrix\_part-2.mpeg > the-matrix.mpeg   
  
Vale lembrar que tal procedimento não é corretamente executado com arquivos AVI.

**tac**

O *tac* faz o mesmo que o cat, mas exibe o arquivo pela ordem inversa, ou seja, começando pela última linha e terminando com a primeira.   
  
Uso:   
  
**$ tac [arquivo]**

**touch**

O comando *touch* é usado atualizar as informações sobre as datas de último acesso e última modificação de um arquivo.   
  
Sintaxe:   
  
**$ touch [opções] [arquivo]**   
  
Se o arquivo não existir, ele é criado, por padrão. Isso faz o touch ser muito utilizado para criar arquivos vazios, através do comando touch [arquivo].   
  
Opções:

* -a: Modifica apenas a data do último acesso;
* -c: Não cria arquivos, caso eles não existam;
* -m: Modifica apenas a data de modificação;
* -t: A data e hora a ser utilizada para o último acesso ou última modificação. O formato utilizado é MMDDhhmm (mês, dia, hora e minuto);

**cp**

O *cp*, abreviação de copy (copiar), é utilizado para copiar arquivos e diretórios de um local para outro, com o mesmo nome ou com nome diferente.   
  
Sintaxe:   
  
**$ cp [opções] [origem] [destino]**

* -b: Cria um arquivo dos arquivos de destino se eles estiverem para ser sobrescritos;
* -P: Quando tratar de links simbólicos, copia o link, e não o local para onde o link aponta;
* -f: Operação forçada. Se um dos arquivos de destino não puder ser aberto, apaga-o e repete a operação;
* -i: Pede confirmação antes de sobrescrever um arquivo;
* -L: Quando tratar de links simbólicos, copia o local para o onde o link aponta, e não o link;
* -p: Preserva as propriedades do arquivo (permissões, dono e datas);
* --preserve=[propriedade]: Escolhe quais propriedades preservar, separadas por vírgula. Podem ser:
  + mode: Preserva as permissões;
  + ownership: Preserva a informação de dono do arquivo;
  + timestamp: Preserva as datas de acesso e modificação.
* --no-preserve=[propriedade]: Escolhe quais propriedades não devem ser preservadas. As opções são as mesmas que do parâmetro --preserve;
* -R ou -r: Modo recursivo, copia todos os arquivos e subdiretórios do diretório especificado. Esse parâmetro deve ser usado para copiar diretórios inteiros;
* --target-directory=[diretório]: Especifica para qual diretório devem ser copiados os arquivos/diretórios especificados;
* -u: Copia apenas os arquivos novos. Se um arquivo que estiver sendo copiado já existir no diretório de destino, sua cópia será ignorada;
* -v: Mostra os detalhes da cópia dos arquivos.

Exemplos de uso:   
  
Para copiar o arquivo file.gz para o diretório /tmp/:   
  
**$ cp file.gz /tmp**   
  
Para para fazer uma cópia do arquivo file.gz com o nome file-copia.gz:   
  
**$ cp file.gz file-copia.gz**   
  
Para copiar os arquivos file1, file2 e file3 para o diretório /home/davidson/doc/:   
  
**$ cp file1 file2 file3 /home/davidson/doc**   
  
Para copiar o diretório img/ para o diretório /tmp/upload/:   
  
**$ cp -r img /tmp/upload**   
  
Para copiar os arquivos file1, file2 e file3 e o diretório img/ para o diretório /tmp/upload/:   
  
**$ cp -r file1 file2 fil3 img /tmp/upload**

**mv**

Utilizamos o *mv* mover ou renomear arquivos.   
  
Sintaxe:   
  
**$ mv [opções] [destino]**   
  
Opções:

* -b: Cria um backup dos arquivos de destino, se eles forem sobrescritos;
* -f: Força as operações, sem fazer perguntas caso seja necessário sobrescrever arquivos e outros;
* -i: Modo [interativo](http://www.vivaolinux.com.br/artigo/Fundamentos-do-sistema-Linux-comandos-do-Linux/), pede confirmação para sobrescrever arquivos;
* --target-directory=[diretório]: especifica o diretório de destino para os arquivos;
* -u: Só move os arquivos novos. Se o arquivo que está sendo movido já estiver presente no diretório de destino, ele é ignorado;
* -v: Mostra os detalhes do processo de movimentação.

Exemplos de uso:   
  
Para mover o arquivo file1 para o diretório /home/davidson/doc/:   
  
**$ mv file1 /home/davidson/doc**   
  
Para mover o diretório /home/davidson/doc/ para /tmp/upload/:   
  
**$ mv /home/davidson/doc /tmp/upload**   
  
Para renomear o arquivo package.tar.gz para pacote.tar.gz:   
  
**$ mv package.tar.gz pacote.tar.gz**   
  
Para mover o arquivo file1 e o diretório img/ para o diretório /tmp/upload/:   
  
**$ mv file1 img /tmp/upload**

**rm**

O *rm* é utilizado para excluir arquivos.   
  
Sintaxe:   
  
**$ rm [opções] [arquivo]**   
  
Opções:

* -f: Modo forçado, não pede confirmação para realizar as operações;
* -i: Pede confirmação antes de remover qualquer arquivo;
* -R, -r: Exclui recursivamente todo o conteúdo do diretório e o próprio diretório. Quando quiser excluir um diretório que não está vazio, utilize esse parâmetro;
* -v: Mostra os detalhes das exclusões.

**ln**

Esse é o comando utilizado para criar links, simbólicos ou absolutos.   
  
Sintaxe:   
  
**$ ln [opções] [alvo] [nome do link]**   
  
Opções:

* -b: Se houver um arquivo com o mesmo nome do link que está sendo criado no diretório de destino, cria um backup do arquivo existente;
* -d: Permite ao administrador do sistema (root) criar um hardlink (link absoluto) para um diretório;
* -f: Força a criação dos links;
* -n: Trata um link simbólico pra um diretório como se fosse um arquivo normal;
* -i: Pergunta antes de remover arquivos existentes;
* -s: Cria um link simbólico;
* --target-directory=[diretório]: Especifica em qual diretório o link deve ser criado;
* -v: Exibe o nome de cada link antes de criá-lo.

Exemplos de uso:   
  
Se você quiser criar um link simbólico para o arquivo /home/davidson/doc/ no diretório atual, com o mesmo nome do diretório real (no caso, doc):   
  
**$ ln -s /home/davidson/doc**   
  
Se você quiser fazer a mesma coisa, mas preferir que o link criado tenha o nome "documentos":   
  
**$ ln -s /home/davidson/doc documentos**   
  
Se você quiser criar um link absoluto (hardlink), oculte o parâmetro -s:   
  
**$ ln /home/davidson/doc**

**cmp**

Esse comando é utilizado para comparar dois arquivos e mostrar a primeira diferença entre eles. Use para [certificar](http://www.vivaolinux.com.br/artigo/Fundamentos-do-sistema-Linux-comandos-do-Linux/)-se de que dois arquivos possuem ou não o mesmo conteúdo.   
  
Sintaxe:   
  
**$ cmp [opções] [arquivo1] [arquivo2]**   
  
Opções:

* -b: Imprime os bytes que são diferentes entre si;
* -i [n]: Não considera os primeiros [n] bytes de cada arquivo;
* -l: Mostra os número dos bytes e os valores diferentes;
* -s: Não mostra nenhum detalhe, apenas sai com status 1 se alguma diferença for encontrada.

Exemplos de uso:   
  
Vamos comparar os arquivos file1 e file2:   
  
**$ cmp file1 file2**  
file1 file2 differ: byte 10, line 2

**diff**

Esse comando compara dois arquivos de texto e mostra as diferenças entre eles.   
  
Sintaxe:   
  
**$ diff [opções] [arquivo1] [arquivo2]**   
  
Opções:

* -i: Ignora as diferenças de letras maiúsculas/minúsculas;
* -E: Ignora as diferenças de tabulação;
* -b: Ignora diferenças na quantidade de espaço em branco;
* -w: Ignora qualquer espaço em branco;
* -B: Ignora linhas em branco a mais ou a menos;
* -a: Compara os arquivos como arquivos de texto, ainda que não sejam;
* -u [n]: Mostra [n] linhas do conteúdo final do arquivo1, adicionadas as diferenças do arquivo2;
* -q: Mostra apenas se o conteúdo dos arquivos são ou não diferentes;
* -y: Mostra os arquivos em duas colunas, indicando as diferenças;
* -t: Expande as tabulações, convertendo-as em espaços, na saída;
* -r: Compara recursivamente todo o conteúdo de um diretório;
* -S [arquivo]: Quando comparar diretórios, inicia a comparação pelo arquivo especificado.

Exemplos de uso:   
  
Vamos considerar os arquivos file1 e file2, com o seguinte conteúdo:   
  
**$ cat file1**  
5 f j 33  
diferença  
2 a c 1  
1 t 4 f  
6 b c \_  
10 i r 3   
  
**$ cat file2**  
5 f j 33  
2 a c 1  
1 t 4 f  
6 b c \_  
10 i r 3  
outra diferença   
  
Aplicando o *diff* nos dois arquivos, temos o seguinte retorno:   
  
**$ diff file1 file2**  
2d1  
< diferença  
6a6  
> outra diferença   
  
O diff exibe informações sobre o que é necessário fazer para que o conteúdo de file1 seja igual ao de file2. Nesse caso, 2d1 quer dizer que a diferença foi encontrada na linha 2, e que é necessário apagar (d = delete) a palavra diferença do arquivo file1. O 6a6 nos diz que a diferença foi encontrada na linha 6, e que é necessário adicionar (a = add) a linha outra diferença no arquivo file1.   
  
O parâmetro -y exibe esses parâmetros de forma mais clara:   
  
**$ diff -y file1 file2**  
5 f j 33                                                        5 f j 33  
diferença                                                     <  
2 a c 1                                                         2 a c 1  
1 t 4 f                                                         1 t 4 f  
6 b c \_                                                         6 b c \_  
10 i r 3                                                        10 i r 3  
                                                              > outradiferença   
  
Veja que os sinais de maior e menor (>, <) sempre apontam para a linha que é diferente. Se a diferença estiver no primeiro arquivo, listado à esquerda, ela deve ser apagada. Se a diferença estiver no segundo arquivo, listado à direita, ela deve ser adicionada.   
  
O diff, quando usado em conjunto com o utilitário patch, fornece uma grande funcionalidade para atualizações. Um grande exemplo é o código-fonte do kernel [Linux](http://www.vivaolinux.com.br/linux/). Ao invés de gravar a nova versão do kernel inteira, é possível gravar a penas as diferenças entre eles, algo como:   
  
**$ diff [kernel-antigo] [kernel-novo] > diferenças.diff**   
  
E depois utilizar o utilitário patch para gravar as diferenças no kernel-antigo, fazendo-o ficar com o mesmo conteúdo de kernel-novo. A grande vantagem é que não é necessário o usuário baixar todo o kernel, que é muito grande, mas apenas o arquivo com as diferenças, bem pequeno.

**patch**

Utilizamos esse comando para atualizar as diferenças geradas através do comando diff. Suponhamos os arquivos file1 e file2, que são diferentes. Podemos criar as diferenças entre os dois arquivos com o comando diff:   
  
**$ diff file1 file2 > file.diff**   
  
Esse comando gera um arquivo file.diff com as diferenças entre os arquivos file1 e file2. Podemos agora usar o comando patch para aplicar as diferenças no arquivo file1, fazendo seu conteúdo ficar igual ao de file2:   
  
**$ patch file1 file.diff**   
  
Sintaxe:   
  
**$ patch [opções] [arquivo] [arquivo de patch]** (para arquivos)  
**$ patch [opções] < [arquivo de patch]** (para diretórios)   
  
Opções:

* -p [n]: Nível do diretório onde será aplicado o patch. Se [n] for 0, o patch será aplicado no diretório atual. Se for 1, será aplicado no diretório acima (..), se 2, 2 diretórios acima (../..) e assim por diante;
* -b: Cria cópias dos arquivos originais antes de aplicar o patch;
* -binary: Lê e grava usando o modo binário;
* -d [diretório]: Muda para o diretório especificado antes de aplicar o patch;
* -E: Remove arquivos vazios após a aplicação do patch;
* -n: Interpreta o arquivo de patch como um .diff normal;
* -N: Não desfaz patches já aplicados;
* -s: Modo silencioso, não exibe mensagens de erro;
* -u: Interpreta o patch em formato unificado. Use isso se o arquivo de patch foi gerado com diff -u.