**grep**

O *grep* é utilizado para encontrar padrões em arquivos de texto, ou, em outras palavras, procura num arquivo todas as linhas que possuem a palavra ou expressão informada e as exibe na tela.   
  
Sintaxe:   
  
**$ grep [expressão] [arquivo]**   
  
Assim, se você quiser varrer o arquivo /etc/fstab procurando pelas linhas que contém o texto "/dev/fd0", [digite](http://www.vivaolinux.com.br/artigo/Fundamentos-do-sistema-Linux-comandos-do-Linux/?pagina=2):   
  
**$ grep /dev/fd0 /etc/fstab**  
/dev/fd0        /media/floppy0  auto    rw,user,noauto  0       0   
  
Se o padrão informado não for encontrado no arquivo especificado, não é exibida nenhuma mensagem.

**cut**

O *cut* é utilizado para selecionar colunas de texto em um arquivo. Suponhamos que você tenha um arquivo da seguinte forma:   
  
A1 B1 C1  
A2 B2 C2  
A3 B3 C3   
  
E precise, por algum motivo, que somente a terceira coluna seja exibida na tela, dessa forma:   
  
C1  
C2  
C3   
  
É exatamente isso que o cut fará para nós.   
  
Sintaxe:   
  
**$ cut [opções] [arquivo]**   
  
Opções:

* -c: Especifica quais caracteres serão exibidos;
* -d: Especifica qual o caracter será usado como delimitador de campo;
* -f: Lista dos campos que serão exibidos;
* -s: Não exibe as linhas que não contém os delimitadores especificados em -d;
* --output-delimiter=[string]: Utiliza a string (seqüência de caracteres) especificada como o delimitador dos campos na saída.

Exemplos de uso:   
  
Considerando o arquivo lista.txt com o conteúdo mostrado acima, se quisermos exibir somente o quinto caracter de cada linha, usamos:   
  
**$ cut -c 5 lista.txt**  
1  
2  
3   
  
Se quisermos exibir os primeiros 5 caracteres, utilizamos -5 ao invés de 5:   
  
**$ cut -c -5 lista.txt**  
A1 B1  
A2 B2  
A3 B3   
  
Se quisermos exibir do quinto caracter em diante, usamos 5-, assim:   
  
**$ cut -c 5- lista.txt**  
1 C1  
2 C3  
3 C3   
  
Por fim, para exibir do segundo até o sexto caracter:   
  
$ cut -c 2-6 lista.txt  
1 B1  
2 B2  
3 B3   
  
Repare agora que o delimitador de cada coluna é o espaço em branco. Assim, se quisermos exibir apenas o primeiro campo, usamos:   
  
**$ cut -f1 -d' ' lista.txt**  
A1  
A2  
A3   
  
O parâmetro -f1 informa que queremos que seja exibido o primeiro campo, enquanto -d' ' (dois acentos agudos separados por espaço) informa que o separador de campos é o espaço em branco. O parâmetro -d não precisa ser utilizado no caso de o arquivo utilizar TAB (tabulação) como separador de campos.   
  
Seguindo esse raciocínio, se quisermos exibir os campos 1 e 3, usamos:   
  
**$ cut -f1,3 -d' ' lista.txt**  
A1 C1  
A2 C3  
A3 C3   
  
Com o uso do direcionador | (pipe) podemos processar o mesmo arquivo várias vezes com o cut, de modo a podermos fazer um processamento poderoso do arquivo de texto. Além disso, ele aceita a saída de outros programas como entrada, através do mesmo direcionador |.

**head**

Usamos esse comando para exibir as linhas iniciais de um arquivo.   
  
Sintaxe:   
  
**$ head [opções] [arquivo]**   
  
Opções:

* -c [-][n]: Sem o - (hífen), mostra os primeiros [n] bytes do arquivo. Com o -, mostra todos os bytes com exceção dos [n] últimos;
* -n [-][n]: Sem o -, mostra as primeiras [n] linhas do arquivo. Com o -, mostra todas as linhas com exceção das [n] últimas;
* -v: Imprime um cabeçalho com o nome do arquivo.

Exemplos de uso:   
  
Para ver as primeiras 5 linhas do arquivo file, faça:   
  
**$ head -n 5 file**  
1  
2  
3  
4  
5   
  
Se quiser exibir todas as linhas menos as 3 últimas:   
  
**$ head -n -3 file**  
1  
2  
3  
4  
5  
6  
7

**more**

Usamos esse comando para realizar a paginação de arquivos de texto cujo conteúdo não cabe na tela.   
  
Sintaxe:   
  
**$ more [opções] [arquivo]**   
  
Opções:

* -d: Exibe as mensagens [Press space to continue, 'q' to quit] (pressione espaço para continuar, 'q' para sair). Ao se pressionar espaço, a [tela](http://www.vivaolinux.com.br/artigo/Fundamentos-do-sistema-Linux-comandos-do-Linux/?pagina=2) rola uma página inteira. Se for pressionada alguma tecla inválida, é exibida a mensagem [Press 'h' for instructions.] (pressione 'h' para instruções.);
* -l: Evita que ocorram pausas toda vez que seja encontrado o caracter "^L" (alimentação de formulário) no texto;
* -s: Se houver múltiplas linhas em branco num arquivo, trunca elas em apenas uma;
* +/[padrão]: Começa a mostrar o texto a partir da primeira linha que contém o padrão informado;
* +[número]: Especifica qual linha deve ser a primeira a ser mostrada;
* -[número]: Especifica o tamanho da tela, em linhas.

Assim, quando quiser ler um texto muito extenso sem precisar abrir um editor de textos para isso, use o more. O texto será exibido até ocupar a tela inteira, e então aparecerá um prompt escrito "--More--(xx%)". Presssione Enter para rolar o texto linha por linha. Se quiser cancelar a exibição e voltar para o prompt de comando, pressione "q".   
  
É possível usar o more para exibir vários arquivos seqüencialmente. Basta informar todos os arquivos separados por espaço.

**less**

O less tem a mesma utilidade do more, com a vantagem de poder rolar o texto exibido para cima e para baixo através do uso dos direcionais, além de contar com um localizador de texto. Para digitar o padrão que você deseja procurar precedido de / (barra).   
  
Sintaxe:   
  
**$ less [arquivo]**

**sort**

Usamos esse comando para classificar as linhas de um arquivo de texto.   
  
Sintaxe:   
  
**$ sort [opções] [arquivo]**   
  
Opções:

* -b: Ignora linhas em branco;
* -d: Descarta quaisquer linhas iniciadas com caracteres que não sejam espaço em branco ou alfanuméricos;
* -f: Ignora a diferença entre caracteres maiúsculos e minúsculos;
* -r: Exibe a lista na ordem inversa;
* -n: Organiza os números na ordem aritmética. Sem essa opção, a seqüência de caracteres 100, 10, 50 seria exibida na ordem 10, 100, 50. Com a opção -n, eles são exibidos na ordem 10, 50, 100;
* -c: Verifica se o arquivo já está organizado. Se não estiver, retorna a mensagem disorder on [arquivo] (desordem em [arquivo]);
* -o [arquivo]: Grava a saída do comando sort no arquivo especificado;
* -m [arquivo1] [arquivo2]: Combina o conteúdo dos dois arquivos gerando um único arquivo. Esse comando só funciona se ambos os arquivos já estiverem ordenados;
* -i: ignora os caracteres fora da faixa octal ASCII 040-0176;
* -t [caracter]: Usa o caracter especificado ao invés de espaço em branco como delimitador durante a organização das linhas;
* +[número 1] +[número 2]: Especifica qual campo (coluna) será usado como referência na organização. Os campos começam a ser contados de 0, e o separador de campos padrão é o espaço. Para selecionar outro delimitador, use o parâmetro -t. Os campos serão organizados de [número 1] até [número 2]. Se [número 2] não for especificado, os campos serão organizados de [número 1] até o final da linha;
* -k [número 1] [número 2]: Idêntico ao parâmetro anterior, mas os campos começam a ser contados de 1.

Exemplos de uso:   
  
Suponha que você tenha um arquivo file.list com o seguinte contéudo:   
  
**$ cat file.list**  
5 f j 33  
2 a c 1  
1 t 4 f  
6 b c \_  
10 i r 3   
  
Ao usarmos o comando sort, as linhas serão organizadas de acordo com a primeira coluna:   
  
**$ sort file.list**  
10 i r 3  
1 t 4 f  
2 a c 1  
5 f j 33  
6 b c \_   
  
Veja, entretanto, que temos um problema aqui: o 10 foi posto antes do 1, seguindo a ordem alfabética e não aritmética. Para corrigir isso, acrescentamos o parâmetro -n:   
  
**$ sort -n file.list**  
1 t 4 f  
2 a c 1  
5 f j 33  
6 b c \_  
10 i r 3   
  
Agora, se quisermos organizar as linhas tomando como referência a segunda coluna ao invés da primeira, fazemos o seguinte:   
  
**$ sort +1 file.list**  
2 a c 1  
6 b c \_  
5 f j 33  
10 i r 3  
1 t 4 f   
  
Podemos obter o mesmo resultado de outra forma:   
  
**$ sort -k 2 file.list**  
2 a c 1  
6 b c \_  
5 f j 33  
10 i r 3  
1 t 4 f   
  
Por fim, para gravar o resultado no arquivo fileorder.list, basta acrescetar o parâmetro -o fileorder.list.

**tail**

Esse comando é utilizando para mostrar as últimas linhas de um arquivo de texto.   
  
Sintaxe:   
  
**$ tail [opções] [arquivo]**   
  
Opções:

* -c [n]: Exibe apenas os últimos [n] bytes do arquivo;
* -n [n]: Exibe as últimas [n] linhas do arquivo;
* -f: Fica monitorando o arquivo, e exibe todas as novas linhas adicionadas a ele, em tempo real. Isso é muito utilizado para monitorar arquivos de log.

**wc**

Conta o número de linhas, palavras, caracteres e bytes nos arquivos.   
  
Sintaxe:   
  
**$ wc [opção] [arquivo]**   
  
Opções:

* -c: Exibe apenas o número de bytes;
* -m: Exibe o número de caracteres;
* -l: Exibe o número de linhas;
* -L: Exibe o comprimento, em caracteres, da maior linha do arquivo;
* -w: Exibe o número de palavras encontradas.

Se for usado sem argumentos, o wc exibe o número de linhas, palavras e caracteres do arquivo, respectivamente:   
  
**$ wc file.list**  
        20 42 file.list