Comandos

Diversos

Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro
Prof. Edwar Saliba Júnior
Julho / 2017

Comando `clear`

- Limpa a tela e posiciona o cursor no canto superior esquerdo do vídeo.

```
clear
```

Comando `date`

- Permite ver/modificar a Data e Hora do Sistema. Você precisa estar como usuário root para modificar a data e hora. .

```
date MesDiaHoraMinuto[AnoSegundos]
```

- Onde:

  `MesDiaHoraMinuto[AnoSegundos]`

  São respectivamente os números do mês, dia, hora e minutos sem espaços. Opcionalmente você pode especificar o Ano (com 2 ou 4 dígitos) e os Segundos.

  `+[FORMATO]`

  Define o formato da listagem que será usada pelo comando date. Os seguintes formatos são os mais usados:

  ```
  %d - Dia do Mês (00-31).
  %m - Mês do Ano (00-12).
  %y - Ano (dois dígitos).
  %Y - Ano (quatro dígitos).
  %H - Hora (00-24).
  %I - Hora (00-12).
  %M - Minuto (00-59).
  %j - Dia do ano (1-366).
  %p - AM/PM (útil se utilizado com %d).
  %r - Formato de 12 horas completo (hh:mm:ss AM/PM).
  %T - Formato de 24 horas completo (hh:mm:ss).
  %w - Dia da semana (0-6).
  ```

Comando `date`

- Outros formatos podem ser obtidos através da página de manual do date.
- Para maiores detalhes, veja a página de manual do comando date.
- Para ver a data atual digite: date
- Se quiser mudar a Data para 25/12 e a hora para 08:15 digite: `date 12250815`
- Para mostrar somente a data no formato dia/mês/ano: `date +%d/%m/%Y`

Comando `df`

- Mostra o espaço livre/ocupado de cada partição.

```
df [opções]
```

- Onde, no lugar de `opções` usa-se:

`-a`
Inclui sistemas de arquivos com 0 blocos.

`-h, --human-readable`
Mostra o espaço livre/ocupado em MB, KB, GB ao invés de blocos.

`-H`
Idêntico a -h mas usa 1000 ao invés de 1024 como unidade de cálculo.

`-k`
Lista em Kbytes.

`-l`
Somente lista sistema de arquivos locais.

`-m`
Lista em Mbytes (equivalente a --block-size=1048576).

- Exemplos: `df, df -h`

Comando `ln`

- Cria links para arquivos e diretórios no sistema. O link é um mecanismo que faz referência a outro arquivo ou diretório em outra localização. O link em sistemas GNU/Linux faz referências reais ao arquivo/diretório podendo ser feita cópia do link (será copiado o arquivo alvo), entrar no diretório (caso o link faça referência a um diretório), etc.

```
ln [opções] [origem] [link]
```

- Onde:
  - `origem`
    Diretório ou arquivo de onde será feito o link.
  - `link`
    Nome do link que será criado.

- opções:
  - `-s`
    Cria um link simbólico. Usado para criar ligações com o arquivo/diretório de destino.
  - `-v`
    Mostra o nome de cada arquivo antes de fazer o link.
  - `-d`
    Cria um hard link para diretórios. Somente o root pode usar esta opção.

Comando `ln`

- Existem 2 tipos de links: simbólicos e *hardlinks*.
- 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*). Use a opção `-s` para criar *links* simbólicos.
- O *hardlink* faz referência ao mesmo inodo do arquivo original, desta forma ele será perfeitamente idêntico, inclusive nas permissões de acesso, ao arquivo original.
- Ao contrário dos *links* simbólicos, não é possível fazer um *hardlink* para um diretório ou fazer referência a arquivos que estejam em partições diferentes.

Comando `ln`

- 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.

- Exemplos:

  `ln -s /dev/ttyS1 /dev/modem` - Cria o link `/dev/modem` para o arquivo `/dev/ttyS1`.

  `ln -s /tmp ~/tmp` - Cria um link `~/tmp` para o diretório `/tmp`.

Comando du

- Mostra o espaço ocupado por arquivos e subdiretórios do diretório atual.

```
du [opções]
```

- Onde `opções` pode ser substituída por:

`-a, --all`
Mostra o espaço ocupado por todos os arquivos.

`-b, --bytes`
Mostra o espaço ocupado em bytes.

`-c, --total`
Faz uma totalização de todo espaço listado.

`-D`
Não conta links simbólicos.

`-h, --human`
Mostra o espaço ocupado em formato legível por humanos (Kb, Mb) ao invés de usar blocos.

`-H`
Como o anterior mas usa 1000 e não 1024 como unidade de cálculo.

`-k`
Mostra o espaço ocupado em Kbytes.

`-m`
Mostra o espaço ocupado em Mbytes.

`-S, --separate-dirs`
Não calcula o espaço ocupado por subdiretórios.

- Exemplo: `du -h, du -hc.`

Comando `find`

- Procura por arquivos/diretórios no disco. O comando `find` pode procurar arquivos através de sua data de modificação, tamanho, etc através do uso de opções. O comando `find`, ao contrário de outros programas, usa opções longas através de um "–".

```
find [diretório] [opções/expressão]
```

- Onde:

  `diretório`

  Inicia a procura neste diretório, percorrendo seu subdiretórios.

  `opções/expressão`

  `–name [expressão]`

  Procura pelo nome `[expressão]` nos nomes de arquivos e diretórios processados.

  `–depth`

  Processa os subdiretórios primeiro antes de processar os arquivos do diretório principal.

Comando `find`

- Onde:

  `-maxdepth [num]`

  Faz a procura até `[num]` subdiretórios dentro do diretório que está sendo pesquisado.

  `-mindepth [num]`

  Não faz nenhuma procura em diretórios menores que `[num]` níveis.

  `-mount, -xdev`

  Não faz a pesquisa em sistemas de arquivos diferentes daquele de onde o comando `find` foi executado.

  `-size [num]`

  Procura por arquivos que tiverem o tamanho `[num]`. `[num]` pode ser antecedido de "+" ou "-" para especificar um arquivo maior ou menor que `[num]`. A opção `-size` pode ser seguida de:

  `b` - Especifica o tamanho em blocos de 512 bytes. É o padrão caso `[num]` não seja acompanhado de nenhuma letra;

  `c` - Especifica o tamanho em bytes;

  `k` - Especifica o tamanho em Kbytes.

Comando `find`

- Onde:
  - `-type [tipo]`
    - Procura por arquivos do `[tipo]` especificado. Os seguintes tipos são aceitos:
      - `b` - bloco
      - `c` - caractere
      - `d` - diretório
      - `p` - *pipe*
      - `f` - arquivo regular
      - `l` - *link* simbólico
      - `s` - sockete

Comando `find`

- A maior parte dos argumentos numéricos podem ser precedidos por "+" ou "−". Para detalhes sobre outras opções e argumentos, consulte a página de manual.

- Exemplo:

  `find / -name grep` - Procura no diretório raiz e subdiretórios um arquivo/diretório chamado “grep”;

- `find / -name grep -maxdepth 3` - Procura no diretório raiz e subdiretórios até o 3o. nível, um arquivo/diretório chamado “grep”.

- `find . -size +1000k` - Procura no diretório atual e subdiretórios um arquivo com tamanho maior que 1000 kbytes (1Mbyte).

Comando `free`

- Mostra detalhes sobre a utilização da memória RAM do sistema.

```
free [opções]
```

- Onde `opções` pode ser substituída por:

`-b`
Mostra o resultado em bytes.

`-k`
Mostra o resultado em Kbytes.

`-m`
Mostra o resultado em Mbytes.

`-o`
Oculta a linha de *buffers*.

`-t`
Mostra uma linha contendo o total.

`-s [num]`
Mostra a utilização da memória a cada `[num]` segundos.

- O comando `free` é uma interface ao arquivo `/proc/meminfo`.

Comando `grep`

- Procura por um texto dentro de arquivo(s) ou no dispositivo de entrada padrão.

```
grep [expressão] [arquivo] [opções]
```

- Onde:

  `expressão`
  palavra ou frase que será procurada no texto. Se tiver mais de 2 palavras você deve identificá-lass com aspas "" caso contrário o `grep` assumirá que a segunda palavra é o arquivo!

  `arquivo`
  Arquivo onde será feita a procura.

- opções

  `-A [número]`
  Mostra o `[número]` de linhas após a linha encontrada pelo `grep`.

  `-B [número]`
  Mostra o `[número]` de linhas antes da linha encontrada pelo `grep`.

  `-f [arquivo]`
  Especifica que o texto que será localizado, esta no arquivo `[arquivo]`.

Comando `grep`

- opções

  `-h, --no-filename`
  Não mostra os nomes dos arquivos durante a procura.

  `-i, --ignore-case`
  Ignora diferença entre maiúsculas e minúsculas no texto procurado e arquivo.

  `-n, --line-number`
  - Mostra o nome de cada linha encontrada pelo `grep`.

  `-E`
  Ativa o uso de expressões regulares.

  `-U, --binary`
  Trata o arquivo que será procurado como binário.

- Se não for especificado o nome de um arquivo ou se for usado um hífen "`-`", `grep` procurará a *string* no dispositivo de entrada padrão. O `grep` faz sua pesquisa em arquivos texto. Use o comando `zgrep` para pesquisar diretamente em arquivos compactados com `gzip`, os comandos e opções são as mesmas.

- Exemplos:

```
grep "capítulo" texto.txt
ps ax|grep inetd
grep "capítulo" texto.txt -A 2 -B 2
```

Comando `head`

- Mostra as linhas iniciais de um arquivo texto.

```
head [opções]
```

- Onde:
  - `-c [numero]`
    Mostra o `[numero]` de bytes do início do arquivo.
  - `-n [numero]`
    - Mostra o [numero] de linhas do inicio do arquivo. Caso não for especificado, o `head` mostra as 10 primeiras linhas.

- Exemplos:

```
head teste.txt
head -n 20 teste.txt
```

Comando `nl`

- Mostra o número de linhas junto com o conteúdo de um arquivo.

```
nl [opções] [arquivo]
```

- Onde opções pode ser substituída por:

`-f [opc]`
Faz a filtragem de saída de acordo com `[opc]`:

`a`
Numera todas as linhas.

`t`
Não numera linhas vazias.

`n`
Numera linhas vazias.

`texto`
Numera somente linhas que contém o [texto].

`-v [num]`
Número inicial (o padrão é 1).

`-i [num]`
Número de linhas adicionadas a cada linha do arquivo (o padrão é 1).

- Exemplos:

```
nl /etc/passwd
nl -i 2 /etc/passwd
```

Comando `more`

- Permite fazer a paginação de arquivos ou da entrada padrão. O comando `more` pode ser usado como comando para leitura de arquivos que ocupem mais de uma tela. Quando toda a tela é ocupada, o `more` efetua uma pausa e permite que você pressione `Enter` ou `espaço` para continuar avançando no arquivo sendo visualizado. Para sair do more pressione `q`.

```
more [arquivo]
```

- Onde: `arquivo` é o arquivo que será paginado.

- Para visualizar diretamente arquivos texto compactados pelo `gzip` ou `.gz` use o comando `zmore`.

- Exemplos:

```
more /etc/passwd
cat /etc/passwd|more
```

Comando `less`

- Permite fazer a paginação de arquivos ou da entrada padrão. O comando `less` pode ser usado como comando para leitura de arquivos que ocupem mais de uma tela. Quando toda a tela é ocupada, o `less` efetua uma pausa (semelhante ao `more`) e permite que você pressione `Seta para Cima` e `Seta para Baixo` ou `PgUP`/`PgDown` para fazer o rolamento da página. Para sair do `less` pressione `q`.

```
less [arquivo]
```

- Onde `arquivo` é o arquivo que será paginado.

- Para visualizar diretamente arquivos texto compactados pelo utilitário `gzip` (arquivos `.gz`), use o comando `zless`.

- Exemplos:

```
less /etc/passwd
cat /etc/passwd|less
```

Comando `sort`

- Organiza as linhas de um arquivo texto ou da entrada padrão.

```
sort [opções] [arquivo]
```

- Onde:

`arquivo`
É o nome do arquivo que será organizado. Caso não for especificado, será usado o dispositivo de entrada padrão (normalmente o teclado ou um "|").

`opções`
`-b`
Ignora linhas em branco.
`-d`
Somente usa letras, dígitos e espaços durante a organização.
`-f`
Ignora a diferença entre maiúsculas e minúsculas.
`-r`
Inverte o resultado da comparação.

Comando `sort`

`opções`

`-n`

Caso estiver organizando um campo que contém números, os números serão organizados na ordem aritmética. Por exemplo, se você tiver um arquivo com os números

100
10
50

Usando a opção `-n`, o arquivo será organizado desta maneira:

10
50
100

Caso esta opção não for usada com o `sort`, ele organizará como uma listagem alfabética (que começam de a até z e do 0 até 9)

10
100
50

`-c`

Verifica se o arquivo já esta organizado. Caso não estiver, retorna a mensagem "disorder on arquivo".

`-o arquivo`

Grava a saída do comando `sort` no arquivo.

Comando `sort`

- Exemplos de uso do comando `sort`:

  `sort texto.txt` - Organiza o arquivo `texto.txt` em ordem crescente;

  `sort texto.txt -r` - Organiza o conteúdo do arquivo `texto.txt` em ordem decrescente;

  `cat texto.txt | sort` - Faz a mesma coisa que o primeiro exemplo, só que neste caso a saída do comando `cat` é redirecionado a entrada padrão do comando `sort`.

  `sort -f texto.txt` - Ignora diferenças entre letras maiúsculas e minúsculas durante a organização.

Comando `tail`

- Mostra as linhas finais de um arquivo texto.

```
tail [opções]
```

- Onde:

`-c [numero]`
Mostra o `[numero]` de bytes do final do arquivo.

`-n [numero]`
Mostra o `[numero]` de linhas do final do arquivo.

`-f`
Mostra continuamente linhas adicionadas no final do arquivo.

- Exemplos:

```
tail teste.txt
tail -n 20 teste.txt
```

Comando `time`

- Mede o tempo gasto para executar um processo (programa);

```
time [comando]
```

- Onde: `comando` é o comando/programa que deseja medir o tempo gasto para ser concluído;

- Exemplo:

```
time ls
time find / -name crontab
```

Comando `touch`

- Muda a data e hora que um arquivo foi criado. Também pode ser usado para criar arquivos vazios. Caso o `touch` seja usado com arquivos que não existam, por padrão ele criará estes arquivos.

```
touch [opções] [arquivos]
```

- Onde:

  `arquivos`

  São arquivos que terão sua data/hora modificados.

  `opções`

  `-t [AAAA]MMDDhhmm[.segundos]`

  Usa opcionalmente o Ano (`AAAA`), obrigatoriamente o Mês (`MM`), Dia (`DD`), Horas (`hh`), minutos (`mm`) e opcionalmente `segundos` para modificação do(s) arquivos ao invés da data e hora atual.

  `-a, --time=atime`

  Faz o `touch` mudar somente a data e hora do acesso ao arquivo.

Comando `touch`

- Continua

  `Opções`

  `-c, --no-create`

  Não cria arquivos vazios, caso os arquivos não existam.

  `-m, --time=mtime`

  Faz o `touch` mudar somente a data e hora da modificação.

  `-r [arquivo]`

  Usa as horas no `[arquivo]` como referência ao invés da hora atual.

- Exemplos:

`touch teste` - Cria o arquivo teste caso ele não existir;

`touch -t 10011230 teste` - Altera da data e hora do arquivo para 01/10 e 12:30;

`touch -t 120112301999.30 teste` - Altera da data, hora ano, e segundos do arquivo para 01/12/1999 e 12:30:30;

`touch -t 12011200 *` - Altera a data e hora do arquivo para 01/12 e 12:00.

Comando `uptime`

- Mostra o tempo de execução do sistema desde que o computador foi ligado.

```
uptime
```

Comando `dmesg`

- Mostra as mensagens de inicialização do kernel. São mostradas as mensagens da última inicialização do sistema.

```
dmesg | less
```

Comando `mesg`

- Permite ou não o recebimentos de requisições de talk de outros usuários.

```
mesg [y/n]
```

- Onde `y` permite que você receba "talks" de outros usuários.

- Digite `mesg` para saber se você pode ou não receber "talks" de outros usuários. Caso a resposta seja "`n`" você poderá enviar um talk para alguém mas o seu sistema se recusará em receber talks de outras pessoas.

- É interessante colocar o comando `mesg y` em seu arquivo de inicialização `.bash_profile` para permitir o recebimento de "talks" toda vez que entrar no sistema.

Comando `echo`

- Mostra mensagens. Este comando é útil na construção de *scripts* para mostrar mensagens na tela para o usuário acompanhar sua execução;

```
echo [mensagem]
```

- A opção `-n` pode ser usada para que não ocorra o salto de linha após a mensagem ser mostrada.

Comando `su`

- Permite ao usuário mudar sua identidade para outro usuário sem fazer o *logout*. Útil para executar um programa ou comando como `root` sem ter que abandonar a seção atual;

```
su [usuário] [-c comando]
```

- Onde: `usuário` é o nome do usuário que deseja usar para acessar o sistema. Se não digitado, é assumido o usuário `root`. Caso seja especificado `-c comando`, executa o comando sob o usuário especificado;

- Será pedida a senha do superusuário para autenticação. Digite `exit` quando desejar retornar a identificação de usuário anterior.

Comando `sync`

- Grava os dados do cache de disco na memória RAM para todos os discos rígidos e flexíveis do sistema. Cache é um mecanismo de aceleração que permite que um arquivo seja armazenado na memória ao invés de ser imediatamente gravado no disco. Quando o sistema estiver ocioso o arquivo é gravado para o disco. O GNU/Linux procura utilizar toda memória RAM disponível para o cache de programas acelerando seu desempenho de leitura/gravação.

```
sync
```

- O uso do `sync` é útil em disquetes quando gravamos um programa e precisamos que os dados sejam gravados imediatamente para retirar o disquete da unidade. Mas o método recomendado é especificar a opção `sync` durante a montagem da unidade de disquetes.

Comando `uname`

- Retorna o nome e versão do kernel atual.

```
uname
```

Comando `reboot`

- Reinicia o computador.

```
reboot
```

Comando `shutdown`

- Desliga/reinicia o computador imediatamente ou após determinado tempo (programável) de forma segura. Todos os usuários do sistema são avisados que o computador será desligado . Este comando somente pode ser executado pelo usuário `root` ou quando é usada a opção `-a` pelos usuários cadastrados no arquivo `/etc/shutdown.allow` que estejam logados no console virtual do sistema.

```
shutdown [opções] [hora] [mensagem]
```

`hora`

Momento que o computador será desligado. Você pode usar `HH:MM` para definir a hora e minuto, `MM` para definir minutos, `+SS` para definir após quantos segundos, ou `now` para imediatamente (equivalente a `+0`).

- O `shutdown` criará o arquivo `/etc/nologin` para não permitir que novos usuários façam login no sistema (com exceção do `root`). Este arquivo é removido caso a execução do `shutdown` seja cancelada (opção `-c`) ou após o sistema ser reiniciado.

`mensagem`

Mensagem que será mostrada a todos os usuários alertando sobre o reinicio/desligamento do sistema.

Comando `shutdown`

`opções`

`-h`
Inicia o processo para desligamento do computador.

`-r`
Reinicia o sistema

`-c`
Cancela a execução do `shutdown`. Você pode acrescentar uma mensagem avisando aos usuários sobre o fato.

- O `shutdown` envia uma mensagem a todos os usuários do sistema alertando sobre o desligamento durante os 15 minutos restantes e assim permite que finalizem suas tarefas. Após isto, o `shutdown` muda o nível de execução através do comando `init` para `0` (desligamento), `1` (modo monousuário), `6` (reinicialização). É recomendado utilizar o símbolo "`&`" no final da linha de comando para que o `shutdown` seja executado em segundo plano;

- Quando restarem apenas 5 minutos para o reinicio/desligamento do sistema, o programa login será desativado, impedindo a entrada de novos usuários no sistema;

- O programa `shutdown` pode ser chamado pelo `init` através do pressionamento da combinação das teclas de reinicialização `CTRL+ALT+DEL` alterando-se o arquivo `/etc/inittab`. Isto permite que somente os usuários autorizados (ou o `root`) possam reinicializar o sistema.

Comando `shutdown`

Exemplos:

"`shutdown -h now`" - Desligar o computador imediatamente;

"`shutdown -r now`" - Reinicia o computador imediatamente;

"`shutdown 19:00 A manutenção do servidor será iniciada às 19:00`" - Faz o computador entrar em modo monousuário (init 1) às 19:00 enviando a mensagem A manutenção do servidor será iniciada às 19:00 a todos os usuários conectados ao sistema;

"`shutdown -r 15:00 O sistema será reiniciado às 15:00 horas`" - Faz o computador ser reiniciado (init 6) às 15:00 horas enviando a mensagem O sistema será reiniciado às 15:00 horas a todos os usuários conectados ao sistema;

`shutdown -r 20` - Faz o sistema ser reiniciado após 20 minutos;

`shutdown -c` - Cancela a execução do `shutdown`.

Comando `wc`

- Conta o número de palavras, bytes e linhas em um arquivo ou entrada padrão. Se as opções forem omitidas, o `wc` mostra a quantidade de linhas, palavras, e bytes.

```
wc [opções] [arquivo]
```

- Onde:

`arquivo`
: Arquivo que será verificado pelo comando `wc`.

`opções`
: `-c, --bytes`
  Mostra os bytes do arquivo.
: `-w, --words`
  Mostra a quantidade de palavras do arquivo.
: `-l, --lines`
  Mostra a quantidade de linhas do arquivo.

- A ordem da listagem dos parâmetros é única, e modificando a posição das opções não modifica a ordem que os parâmetros são listados.

Comando `wc`

- Exemplo:

`wc /etc/passwd` - Mostra a quantidade de linhas, palavras e letras (bytes) no arquivo `/etc/passwd`;

`wc -w /etc/passwd` - Mostra a quantidade de palavras;

`wc -l /etc/passwd` - Mostra a quantidade de linhas;

`wc -l -w /etc/passwd` - Mostra a quantidade de linhas e palavras no arquivo `/etc/passwd`.

Comando `seq`

- Imprime uma sequência de números começando em `[primeiro]` e terminando em `[último]`, utilizando `[incremento]` para avançar.

```
seq [opções] [primeiro] [incremento] [último]
```

- Onde:

  `primeiro`
  : Número inicial da sequência.

  `incremento`
  : Número utilizado para avançar na sequência.

  `último`
  : Número final da sequência.

Comando `seq`

```
opções
  -f, --format=[formato]
      Formato de saída dos números da seqüência. Utilize o estilo do printf para ponto flutuante (valor padrão: %g).
  -s, --separator=[string]
      Usa [string] para separar a seqüência de números (valor padrão: \n).
  -w, --equal-width
      Insere zeros na frente dos números mantendo a seqüência alinhada.
```

- Observações:
  - Se `[primeiro]` ou `[incremento]` forem omitidos, o valor padrão `1` será utilizado.
  - Os números recebidos são interpretados como números em ponto flutuante.
  - `[incremento]` deve ser positivo se `[primeiro]` for menor do que o último, e negativo caso contrário.
  - Quando utilizarmos a opção `--format`, o argumento deve ser exatamente: `%e`, `%f` ou `%g`.

- Exemplos:

```
seq 0 2 10
seq -w 0 10
seq -f%f 0 10
seq -s", " 0 10
```

Referências

- GUIA FOCA GNU/Linux. Iniciante. Disponível em: <http://www.guiafoca.org/cgs/guia/iniciante/ch-cmdv.html>. Acesso em: 27 jul. 2017.