1 - O que é um comando Linux?

No MS-DOS os comandos não poderiam ser criados pelos usuários, ou sejam, eram limitados e geralmente estáticos.

No mundo Unix e por extensão, no Linux, o conceito é diferente. Um comando é qualquer arquivo executável.

1.1 Su

É usado geralmente para alternar entre diferentes usuários dentro de um terminal virtual.

Exemplo de comando: $ su user2 (será solicitada a senha do user2).

Quando acabarmos de trabalhar basta usar o comando **exit** para voltar ao usuário anterior.

Se você está logado como usuário e der o comando **su** sem nome de usuário, será solicitada a senha do Root e, quando ela for fornecida, será trocada para trabalhar como usuário-root.

Se você está logado como Root e der o comando **su** <o nome de algum usuário>, não será solicitado nenhum pedido de senha. Isso é interessante para o administrador, pois ele pode precisar se tornar diferentes usuários para depurar problemas, mas não necessariamente conhecer as senhas de outros usuários.

1.2 pwd, Cd

Esses comandos fornecem as ferramentas básicas de que você precisa para trabalhar com diretórios e arquivos.

O comando **Pwd** informa em qual diretório está atualmente.

O comando **Cd** muda seu diretório atual para qualquer diretório acessível no sistema.

Sintaxe: cd <nome_diretorio> ou
cd <nome_diretorio>/<nome_subdiretorio> ou
cd ../<nome_diretorio>

1.3 ls

O comando **ls** é usado para ver o conteúdo do diretório corrente.

Entre as opções mais úteis temos:

- a – Inclui, na listagem, todos os arquivos contidos no diretório, mesmo as referências do diretório onde estamos posicionados e do diretório “pai”, ou seja, o superior àquele onde estamos posicionados – que são representados por “. “ (diretório atual) e “..” (diretório pai).
- F – Anexa aos nomes dos arquivos um caractere, indicando seu tipo: diretório (/), programas executáveis (*), links simbólicos (@), para FIFOs (|), para sockets (=) e nada para arquivos comuns.
- l – Uso de formato longo, detalhando os dados referentes a (siga os números no exemplo abaixo): (1) permissões, (2) quantidade de

sub-diretórios ou se for 1 se trata de um arquivo, (3) nome do usuário que criou o arquivo e (4) do grupo a que este usuário pertence, (5) tamanho, (6) data da última alteração e (7) nome completo do arquivo. Veja o exemplo com os números indicando estas informações:

```
drwxr-xr-x    2     root    root    1024      Dec 23 15:22          bin
drwxr-xr-x    2     root    root    1024      Dec 31 05:48          boot
drwxr-xr-x    2     root    root    1024      Dec  6 15:51 cdrom
drwxr-xr-x    3     root    root    8192      Mar 11 10:17          dev
drwxrwxr-x    2     root    root    1024      Feb 27 13:52          dosa
|-----(1)----| (2)    (3)     (4)     (5)     |-------(6)------|     (7)
```

- R – Listagem recursiva. Irá também acessar os arquivos que estão colocados internamente nos subdiretórios, a partir do ponto em que estamos.
- u – Usa a data do último acesso ao arquivo para a classificação da saída.
- X – Usa a extensão do nome de arquivo para a ordenação.
- L – Mostra entradas apontadas pelos links simbólicos.
- n - Mostra UIDs e GIDs numéricos em vez dos nomes
- S – Ordenar pelo tamanho do arquivo

1.4 mkdir

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.
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.

Opções:

- **m** (mode) - Esta opção permite aos usuários especificar o modo a ser usado para os novos diretórios.
- **p** - Com esta opção, mkdir cria o nome do diretório através da criação de todos os diretórios-pai não existentes primeiro.

Exemplo: mkdir -p diretório 1/diretório 2/diretório 3
cria a estrutura de subdiretórios "diretório 1/diretório 2/diretório 3".

1.5 More e Less

O comando **more** permite que o usuário se movam uma linha ou uma tela para frente por vez, em um longo corpo de texto, assim como pesquisar esse texto. Pressiona a barra de espaço faz pular para frente uma página, enquanto pressionar Enter moverá para frente uma linha por vez.

Para pesquisar para frente o arquivo inteiro, pressione a tecla de barra (/), seguida da palavra ou frase que você deseja pesquisar e, em seguida, pressione Enter. Você pode repetir o processo pressionando a tecla **n**, após a primeira busca, evitando a necessidade de digitar a mesma palavra ou frase repetidamente.

O comando **Less** é uma versão amplamente aprimorada do comando more. Além das funções básicas descritas anteriormente, a seguir estão algumas das outras ações que podem ser realizadas em um corpo de texto:

- Pular diretamente para uma linha – Coloque o número da linha seguido da letra **g.**
- Pular diretamente para o início ou final do arquivo - Se digitar **g** você pula para a primeira linha do texto. Com o **G** sozinho você pula para a última linha do texto.
- Retroceder em um arquivo – A seta para cima faz mover para cima uma linha de texto por vez e a seta para baixo faz mover uma linha de texto para baixo.
- Pesquisar retroativamente em um arquivo – Uma barra normal (/) seguida de uma palavra ou frase, pesquisará para frente do texto, e um ponto de interrogação

1.6 Find

O comando **Find** pode ser usado para pesquisar arquivos pelo nome, data de criação ou modificação, proprietário, tamanho do arquivo e até o tipo do arquivo.

A sua estrutura básica é:

$ find [diretório inicial] [parâmetros] [ações]

Diretório inicial especifica onde a pesquisa vai iniciar.

Os parâmetros representam o lugar em que você especifica os critérios de busca.

A seção referente às ações indica a ação que será executada nos arquivos encontrados. Geralmente, você desejará usar a ação **–print**, o que indica que o nome e o caminho completos do arquivo devem ser apresentados.

Também é possível pesquisar nomes de arquivo parciais. Por exemplo, se você sabe que o arquivo que está procurando começa com *fo,* então pode usar a expressão fo* para indicar todos os arquivos que começam com *fo* e terminam com qualquer combinação. Quando você usa o caractere *, é importante colocar apóstrofos em torno da expressão inteira.

Ex.: $ find / -name ‘fo*’ –print

1.7 Grep

Comando **Grep** é usado para verificar o conteúdo de um ou mais arquivos na tentativa de encontrar a ocorrência de um padrão de texto específico dentro dos arquivos.

Em geral, o padrão para o comando é:

$ grep <opções> [texto para pesquisa] [arquivos]

Se você quiser procurar uma frase, como “rio de janeiro”, precisará colocar o texto padrão entre aspas:

$ grep “rio de janeiro” *

Opções:

- **l** – Mostra os nomes dos arquivos que contêm o texto de busca.
- **c** – Informa o número de linhas num arquivo que atende à pesquisa feita.
- **i** – Não diferencia letras maiúsculas de minúsculas para o termo que está sendo pesquisado.

Considere a situação em que você deseja uma listagem de todos os arquivos do diretório corrente, com a data da modificação de 12 de maio. Você poderia encontrar essa informação usando pipe com ls –l, através de um comando grep:

$ ls –l | grep “may 12”

1.8 Tar

O programa **tar** era usado originalmente para criar backups de sistema em fitas

A criação de um arquivo **tar** é fácil:

$ tar cvf tar-nome-arquivo lista-arquivo

Esse comando criará um novo arquivo, especificado pelo nome de arquivo *tar-nome-arquivo* (geralmente tem extensão .tar), e depois armazenará todos os arquivos da lista nesse arquivo.

Cada uma dessas opções é usada para controlar diferentes aspectos do comportamento do comando tar. O **c** indica que estamos criando um arquivo, **v** indica que o comando deve ser executado no modo verbose (o que significa que cada nome de arquivo será apresentado, à medida que é copiado no arquivamento) e **f** significa que estamos gerando um arquivo (em oposição a uma unidade de fita).

comando *tar* copia todos os arquivos e subdiretórios de um diretório e um arquivo, caso o diretório faça parte da lista de arquivos. Assim, se temos um diretório chamado *vnc* e queremos que todo o conteúdo desse diretório seja copiado em um novo arquivo, chamado *vnc.tar*, podemos usar

$ tar cvf vnc.tar vnc

e obter o seguinte resultado:

```
$ tar -cvf vnc.tar vnc
vnc/
vnc/license.txt
vnc/readme
vnc/classes/
vnc/classes/rfprrre.class
```

Você notará que a primeira linha indica a criação do diretório *vnc* no arquivo e depois a cópia dos arquivos deste diretório do arquivo *.tar.*

Para ver o conteúdo de um arquivo *.tar* existente, substituímos a opção **c** por **t.**

```
$ tar -tvf vnc.tar
```

Para extrair o conteúdo de um arquivo *.tar* no diretório corrente, substituímos o **c** ou **t** por **x:**

```
$ tar -xvf vnc.tar
```

1.9 Gzip

Embora o comando *tar* seja útil para o armazenamento de arquivos, ele não realiza qualquer compactação nos exemplos anteriores. No Linux, a compactação é obtida com o comando **gzip.**

Ao contrário dos arquivos ZIP do Windows, que compacta muitos arquivos em um único arquivo compactado, o comando gzip compacta apenas arquivos individuais, sem compactá-lo em um arquivo.

Por exemplo, se temos um arquivo particularmente grande, chamado *test.pdf* , que não usaremos por algum tempo e queremos compactá-lo para economizar espaço em disco, usamos o comando gzip:

```
$ gzip test.pdf
```

Isso compactará o arquivo e incluirá a extensão . gz no final do nome de arquivo, mudando o nome para test.pdf.gz.

Para fazer a compressão máxima usamos a extensão –9.

```
$ gzip –9 test.pdf
```

Você pode listar o conteúdo do arquivo compactado usando a extensão –l.

```
$ gzip –l test.pdf
```

Para descompactar um arquivo .gz, retornando o arquivo ao seu estado descompactado original, com o nome *teste.pdf.*

```
$ gzip -d teste.pdf.gz
```

Um comando alternativo, gunzip, elimina a necessidade de usar a opção –d:

$ gunzip test.pdf.gz

1.9.1 Combinando gzip e tar

As versões recentes de *tar* fornecem um método para acessar diretamente e criar arquivos tar compactados com gzip.

Apenas incluindo uma opção **z** em qualquer um dos comandos *tar* discutidos anteriormente, podemos criar um arquivo compactado sem a necessidade de um segundo comando.

$ tar -czvf vnc.tar.gz vnc (inclui todos os arquivos do diretório e subdiretórios de vnc compactando-os automaticamente)

$ tar -tzvf vnc.tar.gz vnc (apresenta o conteúdo de nosso arquivo text.tar.gz compactado)

$ tar -xzvf vnc.tar.gz vnc (extrai o conteúdo do arquivo).

1.10 cp

Para copiar um arquivo (ThisFile) do diretório corrente em um segundo arquivo (a ser chamado ThisFile-Acopy)

$ cp ThisFile ThisFile-Acopy

Se quisermos copiar ThisFile em /tmp, mas fornecer um nome diferente para o novo arquivo, podemos usar

$ cp ThisFile /tmp/NewFileName

Se você desse o comando “cp ThisFile NewFile” o conteúdo de NewFile seria sobrescrito por uma cópia de ThisFile e seria perdido para sempre.

Para evitar essa dificuldade, você pode usar o flag **–i** do comando cp, que obriga o sistema a confirmar quando qualquer arquivo for sobrescrito por uma cópia.

Você pode criar um **alias** para o comando cp executando o comando

$ alias cp=’cp –i’

Podemos configurar nosso shell Bash usando o arquivo oculto **.bashrc** para garantir que, sempre que nos conectarmos, esse alias esteja definido. Para isso devemos editar esse arquivo com qualquer editor de texto (Ex.: mcedit) e incluir o alias dentro dele.

Para criar um alias para todos os usuários vá ao diretório /etc/rc.d e crie um arquivo qualquer definindo todos os alias que desejar. Ao dar boot na

máquina todos os arquivos que estão abaixo desse diretório são executados, incluindo seu arquivo de alias que acaba de criar.

Podemos passar vários argumentos para o comando e o último deles será tratado como o destino e todos os arquivos precedentes serão copiados no destino.

```
$ cp   FileOne   FileTwo   FileThree   /tmp
```

Ao copiar vários arquivos desse modo, é importante lembrar-se de que o último argumento deve ser um diretório.

Se quisermos copiar um diretório inteiro e todos os seus subdiretórios, podemos usar o flag **–R** do comando cp.

```
$ cp   -R   SomeDir   /tmp
```

Esse comando copia a totalidade do subdiretório SomeDir para o diretório /tmp criando o diretório /tmp/Somedir.

1.10.1 Cópia avançada

Quando você copia um arquivo, o arquivo resultante normalmente pertence a quem copiou, e não a quem criou o arquivo.

Analogamente, quando um arquivo é criado em um diretório, ele possui um conjunto de permissões padrão atribuídas a ele. Ao copiar um arquivo, a cópia terá as permissões definidas de acordo com o padrão do diretório de destino, em vez de manter as permissões do arquivo original. Para mantermos os atributos originais usamos o flag **–p**

```
$ cp   -p   /tmp/TheFile   .
```

Normalmente, quando você copia um vínculo simbólico, o arquivo resultante é uma cópia do arquivo apontado pelo vínculo para o mesmo arquivo.

```
Lrwxrwxrwx   1   user2   users   2 Aps 5 13:10   TheFile - > OtherFile
```

Então a execução do comando cp

```
$cp   /tmp/TheFile   ~/NewFile
```

(o ~ significa que você irá fazer a cópia debaixo do diretório home do usuário atual)

resultaria em um arquivo que seria uma cópia de OtherFile.

Mas, e se quiséssemos copiar o vínculo, em vez do próprio arquivo? Para isso o comando cp tem um flag para tratar dessa situação: o flag **–d**, que indica a não-eliminação da referência ao vínculo simbólico. Poderíamos simplesmente usar o comando

```
$ cp   –d   /tmp/TheFile   ~/NewFile
```

Dito isso, é hora de reunir tudo. *E se quisermos usar o comando cp para criar uma cópia de backup útil de um diretório existente e todos os seus subdiretórios?*

$ cp -pdR TheDirectory /backups (cria uma cópia exata de TheDirectory no diretório /backups/TheDirectory)

Porém o comando cp fornece um modo simplificado para obter isso: o flag **–a.**

$ cp -a TheDirectory /backups

1.10.2 Evitando erros

- Você pode usar o flag **–b** para criar uma cópia de backup de qualquer arquivo que vá ser sobrescrito. Por padrão, o backup será o nome de arquivo original com um til (~) depois dele.
- É possível alterar o modo como o comando cp atribui nomes aos arquivos de backup, usando dois flags diferentes: **–S** e **–V.** O flag –s permite que você mude o caractere de til usado em nomes de backup para outra coisa.

  $ cp –b –S_ FileOne FileTwo

- O flag **–V** proporciona ainda mais flexibilidade, permitindo que o usuário especifique um dos três tipos de esquemas de atribuição de nomes de backup:

  - **t** ou **numbered** : cria backups numerados em seqüência. Se um arquivo de backup numerado já existir, então o novo arquivo de backup será numerado seqüencialmente, após o arquivo de backup existente; os nomes de arquivo resultantes são como os seguintes: FileName.~Number~ (Ex.:FileName.~2~)

    $ cp -b -V t FileOne FileTwo

  - **Nil** ou **existing:** se um arquivo de backup numerado já existe, então cria um arquivo de backup numerado; caso contrário, cria um arquivo de backup simples normal.
  - **Never** ou **simples:** cria um arquivo de backup simples usando o til padrão ou um caractere alternativo, indicado pelo flag –s.

1.11 rm

Este comando é utilizado para apagar arquivos. É importante lembrar que quando os arquivos são apagados, no sistema Unix, é impossível recuperá-los.

```
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.

Quando você apaga um arquivo com o comando "rm", você está apagando somente um link (ligação ou entrada) para um arquivo. Um arquivo somente será apagado verdadeiramente do sistema quando ele não possuir mais nenhuma ligação para ele, isto é, nenhum link referenciando-o. Geralmente, arquivos possuem somente um link, portanto o uso do comando "rm" irá apagar o(s) arquivo(s). No entanto, se um arquivo possuir muitos links, o uso de "rm" irá apagar somente uma ligação; neste caso, para apagar o arquivo, é necessário que você apague todos os links para este arquivo.

Você pode verificar o número de links que um arquivo possui utilizando o comando ls, com a opção "-l".

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** - Opção recursiva para remover um diretório e todo o seu conteúdo, incluindo quaisquer subdiretórios e seus arquivos.

 CUIDADO : diretórios e seus conteúdos removidos com o comando "rm -r" não podem ser recuperados.

1.11.1 Excluindo diretórios inteiros

Você pode remover o diretório inteiro usando o flag **–r**

```
$ rm -r tempInstall
```

Quando você está certo de que deseja excluir um diretório inteiro, vai querer usar o flag **–f** do comando rm.

```
$ rm -rf tempInstall
```

1.12 rmdir

é utilizado para apaga diretórios vazios.

```
Sintaxe: rmdir (diretório 1) (diretório 2) ... (diretório n)
```

onde (diretório 1) até (diretório n) são os diretórios a serem apagados. O comando "rmdir" se recusa a apagar um diretório inexistente, exibindo a mensagem:

```
rmdir : (nome-do-diretório) : No such file or directory
```

Quando usar "rmdir", lembre-se que o seu diretório de trabalho corrente não pode estar contido no(s) diretório(s) a ser(em) apagado(s). Se você tentar remover seu próprio diretório corrente, será exibida a seguinte mensagem:

```
rmdir : . : Operation not permited
```

Se o diretório o qual você deseja remover não estiver vazio, utilize o comando "cd" para acessar os arquivos dentro do diretório, e então remova estes arquivos utilizando o comando "rm".
Opções:
-p Permite aos usuários remover o diretório e seu diretório pai, o qual se torna vazio. Uma mensagem será exibida na saída padrão informando se o caminho ("path") inteiro foi removido ou se parte do caminho persiste por algum motivo.
 CUIDADO : diretórios removidos com o comando "rmdir" não podem ser recuperados!

1.13 mv

Vamos começar considerando a operação básica de movimentação:

$ mv FileOne /tmp

É possível mover o arquivo para o diretório /tmp e mudar o seu nome usando o seguinte comando:

$ mv FileOne /tmp/NewFileName

Usando esse conceito, você pode renomear um arquivo. Basta mover um arquivo de seu nome existente para um novo nome no mesmo diretório:

$ mv FileOne NewFileName

Ao copiar arquivos, é possível mover vários deles de uma vez, pois o comando mv pode aceitar mais de dois argumentos e o último argumento servirá como diretório de destino da movimentação.

$ mv *.bak *.tmp *.old /tmp

É possível mover diretórios inteiros com o comando mv, sem usar nenhum flag especial. Se houvesse um subdiretório chamado TheDir no diretório atual e quiséssemos movê-lo de modo que ele se tornasse um subdiretório sob /tmp, usaríamos o comando mv exatamente como fizemos para arquivos:

$ mv /Thedir /tmp
Opções:

- **b** - Fará uma cópia de segurança de arquivos que serão sobrepostos pela movimentação, caso já existam arquivos com aqueles nomes no volume de destino.
- **u** - Atualiza apenas os arquivos que tiverem data de atualização anterior ao que está sendo movido sobre outro, já existente. Assim sendo, apenas os mais novos irão substituir as versões mais antigas.

1.14 cat

Oficialmente usado para concatenar arquivos. Também usado para exibir 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 seqüência e exibe-o na saída padrão. Deste modo , a linha de comando:

cat < arquivo >

exibirá o arquivo em seu terminal; e a linha de comando :

cat < arquivo1 > < arquivo2 > > < arquivo3 >

concatenará "arquivo1" e "arquivo2", e escreverá o resultado no “arquivo 3” . O símbolo ">", usado para redirecionar a saída para um arquivo, tem caráter destrutivo; em outras palavras, o comando acima escreverá por cima do conteúdo de < arquivo3 >. Se, ao invés disto, você redirecionar com o símbolo ">>", a saída será adicionada a <arquivo3 >, ao invés de escrever por cima de seu conteúdo.

1.15 chgrp

Modifica o grupo de um arquivo ou diretório.

Sintaxe: chgrp [-f] [-h] [-R] gid nome-do-arquivo

"chgrp" modifica o identificador de grupo ("group ID" , gid) dos arquivos passados como argumentos. "gid" pode ser um número decimal especificando o group id, ou um nome de grupo encontrado no arquivo "/etc/group". Você

deve ser o proprietário do arquivo, ou o superusuário, para que possa utilizar este comando.

Opções:

-f Esta opção não reporta erros

-h Se o arquivo for um link simbólico, esta opção modifica o grupo do link simbólico. Sem esta opção, o grupo do arquivo referenciado pelo link simbólico é modificado.

- R Esta opção é recursiva. "chgrp" percorre o diretório e os subdiretórios, modificando o GID à medida em que prossegue.

1.16 chmod

Modifica as permissões de um arquivo ou diretório. Você deve ser o proprietário de um arquivo ou diretório, ou ter acesso ao root, para modificar as suas permissões.

Sintaxe : chmod permissões nome_do_arquivo

onde :

permissões - indica as permissões a serem modificadas;
nome - indica o nome do arquivo ou diretório cujas permissões serão afetadas.

As permissões podem ser especificadas de várias maneiras. Aqui está uma das formas mais simples de realizarmos esta operação :

1- Use uma ou mais letras indicando os usuários envolvidos: . u (para o usuário) . g (para o grupo) . o (para "outros") . a (para todas as categorias acima)

2- Indique se as permissões serão adicionadas (+) ou removidas (-).

3- Use uma ou mais letras indicando as permissões envolvidas : . r (para "read") (ler) . w (para "write") (escrever) . x (para "execute") (executar)

Exemplo : No exemplo a seguir, a permissão de escrita ("write") é adicionada ao diretório "dir1" para usuários pertencentes ao mesmo grupo. (Portanto, o argumento "permissões" é g+w e o argumento "nome" é dir1).

```
$ ls -l dir1
drwxr-xr-x 3 dir1 1024 Feb 10 11:15 dir1
$ chmod g+w dir1
$ ls -l dir1
drwxrwxr-x 3 dir1 1024 Feb 10 11:17 dir1
```

Como você pôde verificar, o hífen (-) no conjunto de caracteres para grupo foi modificado para "w" como resultado deste comando.

Quando você cria um novo arquivo ou diretório, o sistema associa permissões automaticamente. Geralmente, a configuração "default" (assumida) para os novos arquivos é:

- r w - r - - r - -

e para novos diretórios é:

d r w x r - x r - x

1.17 chown

Modifica o proprietário de um arquivo ou diretório.

Sintaxe: chown [-fhR] (proprietário) (nome-do-arquivo)

O argumento "proprietário" especifica o novo proprietário do arquivo. Este argumento deve ser ou um número decimal especificando o userid do usuário ou um "login name" encontrado no arquivo "/etc/passwd".

Somente o proprietário do arquivo (ou o superusuário) pode modificar o proprietário deste arquivo.

Opções:

- f Esta opção não reporta erros.

- h Se o arquivo for um link simbólico, esta opção modifica o proprietário do link simbólico. Sem esta opção, o proprietário do arquivo referenciado pelo link simbólico é modificado.

- r Esta opção é recursiva. "chown" percorre o diretório e os subdiretórios, modificando as propriedades à medida em que prossegue.

1.18 du

Exibe o espaço ocupado de um diretório e de todos os seus subdiretórios, em blocos de 512 bytes; isto é, unidades de 512 bytes ou caracteres.

"du" mostra a utilização do disco em cada subdiretório.

1.19 date

Exibe a data configurada no sistema.

O comando "date", a nível de usuário, exibe na tela a data configurada no sistema. Ele pode ser usado com opções que mostram a data local ou data universal GMT - Greenwich Mean Time. A configuração dos dados deste comando só podem se realizadas pelo super-usuário.

Para exibir a data local, basta executar "date". Caso queira a data GMT utilize a opção "-u".

```
$date
Wed Jan 8 12:05:57 EDT 1997
```

Aqui a data é exibida em 6 campos que representam o dia da semana abreviado, o mês do ano abreviado, o dia do mês, a hora disposta em horas/minutos/segundos, a zona horária e o ano.

Podemos acertar a hora, usando o comando na seguinte forma:

$date –s 09:30 (formato hora:minuto)

ou

$date –s 09/18 (formato mm/dd)

1.20 file

Exibe o tipo de um arquivo.

Alguns arquivos, tais como arquivos binários e executáveis, não podem ser visualizados na tela. O comando "file" pode ser útil se você não tem certeza sobre o tipo do arquivo. O uso do comando permitirá a visualização do tipo do arquivo.

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

1.21 init / telinit

É o pai dos processos. O seu papel principal é criar os processos a partir de programas armazenados no arquivo **/etc/inittab**. Este arquivo tem entradas que fazem com que o **init** inicie **getty**s em cada linha que os usuários podem usar para acessar o sistema. Ele controla ainda processos autônomos requeridos por qualquer sistema em particular.

1.21.1 Níveis de execução

É uma configuração de software do sistema que permite que um grupo selecionado de processos sejam inicializados. Os processos acionados por **init** para cada um dos níveis de execução são definidos no arquivo **/etc/inittab**. **Init** pode estar em um dos oito níveis de execução: **0-6** e **S** ou **s.**

1.21.2 Iniciando

Após o **init** ser iniciado com o último passo da sequencia de inicialização, ele procura pelo arquivo **/etc/inittab** e verifica se há alguma entrada para o tipo **initdefault.** A entrada **initdefault** define o nível de execução inicial do sistema. Caso não haja tal entrada, um nível de execução deve ser informado na console do sistema.

1.21.3 Telinit

/sbin/telinit é um link simbólico de **/sbin/init.** Ele recebe um argumento de um caracter e sinaliza ao **init** para executar a ação apropriada. Os seguintes argumentos servem como diretivas para **telinit:**

0,1,2,3,4,5 ou 6 - Dizem ao **init** para mudar o nível de execução.

a,b,c - Dizem ao **init** para processar somente aquelas entradas no arquivo **/etc/inittab** que tenham os níveis de execução a,b ou c.

Q ou q - Dizem ao **init** para reexaminar o arquivo **/etc/inittab.**

S ou s - Dizem ao **init** para entrar em modo monousuário.

1.22 runlevel

Encontra o nível de execução anterior e o atual do sistema.

runlevel [utmp]

O comando runlevel lê o arquivo utmp do sistema (normalmente /var/run/utmp) para localizar o registro do nível de execução, mostrando o nível de execução anterior e o atual na saída padrão, separado por um espaço simples. Se não existir um nível de execução anterior, a letra **N** será impressa em seu lugar.

1.23 apropos

Procura em uma base de dados pela expressão informada.
Este comando procura em uma base de dados de comandos do sistema por uma descrição curta mostrando o resultado na saída padrão. Sua atualização é feita pelo comando makewhatis (/usr/bin).

Exemplo:

```
[root@guarani /tmp]# apropos gif
giftopnm (1)      - convert a GIF file into a portable anymap
pppmtogif (1)     - convert a portable pixmap into a GIF file
Colour (3)        - Colour manipulation routines for use with GIFgraph
GIFgraph (3)      - Graph Plotting Module for Perl 5
```

1.24 type

Mostra a localização de um arquivo. type
Este comando mostra a localização de um arquivo, através do caminho do sistema.

```
[marisa@guarani log]$ type bash
bash is /bin/bash
```

1.25 diff

Compara dois arquivos em formato texto linha a linha.

O comando diff procura encontrar o menor conjunto de diferenças entre as linhas dos arquivos, listando as que devem ser mudadas no primeiro arquivo para torná-lo idêntico ao segundo.

Exemplo:

```
[marisa@guarani log]$ diff  linguagens  linguagens.old
2c2
< java  -  ainda vai ser boa um dia
---
> java  -  ainda vai ser uma boa linguagem um dia
```

1.26 zip

Programa de compactação de arquivos.

O zip é um comando de compressão e empacotamento de arquivos. Ele é análogo à combinação dos comandos tar e compress e é compatível com o pkzip e winzip das plataformas DOS/Windows.

Exemplo:

```
[marisa@guarani log]$ l previsao
-rw-rw-r--    1  marisa   marisa    3274   jul  27   11:37    previsao

[marisa@guarani log]$ zip  previsao  previsao
adding:  previsao   (deflated   59%)
[marisa@guarani log]$ l previsao*
-rw-rw-r--    1  marisa   marisa    3274   jul  27   11:37 previsao
-rw-rw-r--    1  marisa   marisa    1497   jul  27   11:44 previsao.zip
```

1.27 sort

Ordena as linhas de arquivos texto.

O comando sort ordena as linhas de um arquivo texto. Existem diversas opções de ordenamento: ascendente, descendente, por campo do arquivo, etc.

Exemplo:

```
[marisa@guarani log]$ ls –l alunos
-rw-rw-r--    1  marisa   marisa    3274   jul  27   11:37 alunos

[marisa@guarani log]$ cat alunos
linus
alan
bill
eric

[marisa@guarani log]$ sort alunos
alan
bill
eric
linus

[marisa@guarani log]$ sort  –r  alunos
```

linus
eric
bill
alan

1.28 cut

Seleciona campos de uma tabela. cut

A entrada padrão é tratada como uma tabela. O comando seleciona colunas da tabela para serem removidas ou copiadas na saída padrão.

Exemplo:

```
[marisa@guarani log]$ cat   linguagens
C  -  o assembler do passado
Java – ainda vai ser boa um dia
Perl  -  existe mais de um jeito de fazer isso
Php  -  pré processador html

[marisa@guarani log]$ cut  -c1-5  linguagens
C  -  o
Java
Perl
Php
[marisa@guarani log]$ cut  -d '-'  -f  1   linguagens
C
Java
Perl
Php

[marisa@guarani log]$ cut  -d '-'  -f  2   linguagens
o assembler do passado
ainda vai ser boa um dia
existe mais de um jeito de fazer isso
pré processador html
```

1.29 tr

Converte ou remove caracteres. tr

Este comando copia da entrada padrão para a saída padrão substituindo ou removendo os caracteres selecionados. Qualquer caractere de entrada encontrado em expr1 é convertido para o caractere da posição correspondente em expr2.

Exemplo:

```
[marisa@guarani log]$ tr a-z A-Z < linguagens
C  -  O ASSEMBLER DO PASSADO
JAVA – AINDA VAI SER BOA UM DIA
PERL  -  EXISTE MAIS DE UM JEITO DE FAZER ISSO
PHP  -  PRÉ PROCESSADOR HTML
```

1.30 Outros comandos

comm: Compara dois arquivos para determinar quais linhas são comuns entre eles.

du: Relatório no uso do sistema de arquivos.

ed: Editor de texto.

ex: Editor de texto.

mail: Usado para receber ou enviar e-mail.

nroff: Usado para formatar textos.

tset: Escolher o tipo de terminal.

umask: Permite que o usuário especifique uma nova criação de camuflagem.

uniq: Compara dois arquivos. Procura e exibe em linhas o que e incomparável em um arquivo.

uucp: Execução UNIX-para-UNIX

wc: Exibe detalhes no tamanho do arquivo.

write: Usado para mandar mensagens para outro usuário.

1.31 Criando vínculos simbólicos

- Os vínculos simbólicos (que são apenas ponteiros para um arquivo real em outra posição) são usados normalmente por administradores de sistema e projetistas de aplicativos.
- Usa-se o comando **ln** com o flag **–s** para indicar um vínculo simbólico.

$ ln -s /bin/cp MyCopy (cria um vínculo chamado “MyCopy” para acessar virtualmente o diretório “/bin/cp”).

who

Mostra quem está atualmente conectado no computador. Este comando lista os nomes de usuários que estão conectados em seu computador, o terminal e data da conexão.

`who [`*opções*`]`

onde:

opções

-H, --heading

Mostra o cabeçalho das colunas.

-i, -u, --idle

Mostra o tempo que o usuário está parado em Horas:Minutos.

-m, i am

Mostra o nome do computador e usuário associado ao nome. É equivalente a digitar `who i am` ou `who am i`.

-q, --count

Mostra o total de usuários conectados aos terminais.

-T, -w, --mesg

Mostra se o usuário pode receber mensagens via `talk` (conversação).

- + O usuário recebe mensagens via talk
- - O usuário não recebe mensagens via talk.
- ? Não foi possível determinar o dispositivo de terminal onde o usuário está conectado.