Guia Foca GNU/Linux
Capítulo 2 - Explicações Básicas

Este capítulo traz explicações sobre os principais componentes existentes no computador e do sistema operacional.

2.1 Hardware e Software

Hardware - Significa parte física do computador (disquete, pen-drive, impressoras, monitores, placa mãe, placa de fax, discos rígidos, etc).
Software - São os programas usados no computador (sistema operacional, processador de textos, planilha, banco de dados, scripts, comandos, etc).

2.2 Arquivos

É onde gravamos nossos dados. Um arquivo pode conter um texto feito por nós, uma música, programa, planilha, etc.

Cada arquivo deve ser identificado por um `nome`, assim ele pode ser encontrado facilmente quando desejar usa-lo. Se estiver fazendo um trabalho de história, nada melhor que salva-lo com o nome `historia`. Um arquivo pode ser binário ou texto (para detalhes veja Arquivo texto e binário, Seção 2.2.3).
O `GNU/Linux` é *Case Sensitive* ou seja, ele diferencia letras *maiúsculas* e *minúsculas* nos arquivos. O arquivo `historia` é completamente diferente de`Historia`. Esta regra também é válido para os *comandos* e *diretórios*. Prefira, sempre que possível, usar letras minúsculas para identificar seus arquivos, pois quase todos os comandos do sistema estão em *minúsculas*.
Um arquivo oculto no `GNU/Linux` é identificado por um "." no inicio do nome (por exemplo, `.bashrc`). Arquivos ocultos não aparecem em listagens normais de diretórios, deve ser usado o comando `ls -a` para também listar arquivos ocultos.

2.2.1 Extensão de arquivos

A extensão serve para identificar o tipo do arquivo. A extensão são as letras após um "." no nome de um arquivo, explicando melhor:

- `relatório.txt` - O `.txt` indica que o conteúdo é um arquivo texto.
- `script.sh` - Arquivo de Script (interpretado por `/bin/sh`).
- `system.log` - Registro de algum programa no sistema.
- `arquivo.gz` - Arquivo compactado pelo utilitário `gzip`.

- `index.html` - Página de Internet (formato Hypertexto).
  A extensão de um arquivo também ajuda a saber o que precisamos fazer para abri-lo. Por exemplo, o arquivo `relatório.txt` é um texto simples e podemos ver seu conteúdo através do comando cat, Seção 9.1, já o arquivo `index.php` contém uma página de Internet e precisaremos de um navegador para poder visualiza-lo (como o `lynx`, `Firefox` ou o `Konqueror`).
  A extensão (na maioria dos casos) não é requerida pelo sistema operacional `GNU/Linux`, mas é conveniente o seu uso para determinarmos facilmente o tipo de arquivo e que programa precisaremos usar para abri-lo.

2.2.2 Tamanho de arquivos

A unidade de medida padrão nos computadores é o `bit`. A um conjunto de 8 bits nós chamamos de `byte`. Cada arquivo/diretório possui um tamanho, que indica o espaço que ele ocupa no disco e isto é medido em `bytes`. O byte representa uma letra. Assim, se você criar um arquivo vazio e escrever o nome`Linux` e salvar o arquivo, este terá o tamanho de 5 bytes. Espaços em branco e novas linhas também ocupam bytes.

Além do byte existem as medidas Kbytes, Mbytes, Gbytes. Os prefixos K (quilo), M (mega), G (giga), T (tera) etc. vêem da matemática. O "K" significa multiplicar por 10^3, o "M" por 10^6, e assim por diante. Esta letras servem para facilitar a leitura em arquivos de grande tamanho. Um arquivo de 1K é a mesma coisa de um arquivo de 1024 bytes. Uma forma que pode inicialmente lhe ajudar a lembrar: K vem de Kilo que é igual a 1000 - 1Kilo é igual a 1000 gramas certo?.

Da mesma forma 1Mb (ou 1M) é igual a um arquivo de 1024K ou 1.048.576 bytes

1Gb (ou 1G) é igual a um arquivo de 1024Mb ou 1048576Kb ou 1.073.741.824 bytes (1 Gb é igual a 1.073.741.824 bytes, são muitos números!). Deu pra notar que é mais fácil escrever e entender como 1Gb do que 1.073.741.824 bytes :-)

A lista completa em ordem progressiva das unidades de medida é a seguinte:

```
Símbolo 10^ 2^ Nome

K 3 10 Quilo

M 6 20 Mega

G 9 30 Giga

T 12 40 Tera

P 15 50 Peta

E 18 60 Eta

Z 21 70 Zetta

Y 24 80 Yotta
```

2.2.3 Arquivo texto e binário

Quanto ao tipo, um arquivo pode ser de texto ou binário:

`texto`

> Seu conteúdo é compreendido pelas pessoas. Um arquivo texto pode ser uma carta, um script, um programa de computador escrito pelo programador, arquivo de configuração, etc.

`binário`

> Seu conteúdo somente pode ser entendido por computadores. Contém caracteres incompreensíveis para pessoas normais. Um arquivo binário é gerado através de um arquivo de programa (formato texto) através de um processo chamado de `compilação`. Compilação é basicamente a conversão de um programa em linguagem humana para a linguagem de máquina.

2.3 Diretório

Diretório é o local utilizado para armazenar conjuntos arquivos para melhor organização e localização. O diretório, como o arquivo, também é "*Case Sensitive*" (diretório `/teste` é completamente diferente do diretório `/Teste`).
Não podem existir dois arquivos com o `mesmo` nome em um diretório, ou um sub-diretório com um mesmo nome de um arquivo em um mesmo diretório.
Um diretório nos sistemas `Linux/UNIX` são especificados por uma "/" e não uma "\" como é feito no `DOS`. Para detalhes sobre como criar um diretório, veja o comando `mkdir` (mkdir, Seção 8.4).

2.3.1 Diretório Raíz

Este é o diretório principal do sistema. Dentro dele estão todos os diretórios do sistema. O diretório Raíz é representado por uma "/", assim se você digitar o comando `cd /` você estará acessando este diretório.
Nele estão localizados outros diretórios como o `/bin, /sbin, /usr, /usr/local, /mnt, /tmp, /var, /home,` etc. Estes são chamados de *sub-diretórios* pois estão dentro do diretório "/". A estrutura de *diretórios* e *sub-diretórios* pode ser identificada da seguinte maneira:

- /
- /bin
- /sbin
- /usr
- /usr/local
- /mnt

- /tmp
- /var
- /home

A estrutura de diretórios também é chamada de `Árvore de Diretórios` porque é parecida com uma *árvore* de cabeça para baixo. Cada diretório do sistema tem seus respectivos arquivos que são armazenados conforme regras definidas pela *FHS* (*FileSystem Hierarchy Standard - Hierarquia Padrão do Sistema de Arquivos*) versão 2.0, definindo que tipo de arquivo deve ser armazenado em cada diretório.

2.3.2 Diretório atual

É o diretório em que nos encontramos no momento. Você pode digitar `pwd` (veja pwd, Seção 8.3) para verificar qual é seu diretório atual.
O diretório atual também é identificado por um "." (ponto). O comando comando `ls .` pode ser usado para listar seus arquivos (é claro que isto é desnecessário porque se não digitar nenhum diretório, o comando `ls` listará o conteúdo do diretório atual).

2.3.3 Diretório home

Também chamado de diretório de utilizador. Em sistemas `GNU/Linux` cada utilizador (inclusive o root) possui seu próprio diretório onde poderá armazenar seus programas e arquivos pessoais.
Este diretório está localizado em `/home/[login]`, neste caso se o seu login for "joao" o seu diretório home será `/home/joao`. O diretório home também é identificado por um ~(til), você pode digitar tanto o comando `ls /home/joao` como `ls ~` para listar os arquivos de seu diretório home.
O diretório home do utilizador root (na maioria das distribuições `GNU/Linux`) está localizado em `/root`.
Dependendo de sua configuração e do número de utilizadores em seu sistema, o diretório de utilizador pode ter a seguinte forma:`/home/[1letra_do_nome]/[login]`, neste caso se o seu login for "joao" o seu diretório home será `/home/j/joao`.

2.3.4 Diretório Superior

O diretório superior (Upper Directory) é identificado por `..` (2 pontos).
Caso estiver no diretório `/usr/local` e quiser listar os arquivos do diretório `/usr` você pode digitar, `ls ..` Este recurso também pode ser usado para copiar, mover arquivos/diretórios, etc.

2.3.5 Diretório Anterior

O diretório anterior é identificado por "-". É útil para retornar ao último diretório usado.

Se estive no diretório `/usr/local` e digitar `cd /lib`, você pode retornar facilmente para o diretório `/usr/local` usando `cd -`.

2.3.6 Caminho na estrutura de diretórios

São os diretórios que teremos que percorrer até chegar no arquivo ou diretório que que procuramos. Se desejar ver o arquivo `/usr/doc/copyright/GPL`você tem duas opções:

- Mudar o diretório padrão para `/usr/doc/copyright` com o comando `cd /usr/doc/copyright` e usar o comando `cat GPL`
- Usar o comando `"cat"` especificando o caminho completo na estrutura de diretórios e o nome de arquivo: `cat /usr/doc/copyright/GPL`.

As duas soluções acima permitem que você veja o arquivo `GPL`. A diferença entre as duas é a seguinte:

- Na primeira, você muda o diretório padrão para `/usr/doc/copyright` (confira digitando `pwd`) e depois o comando `cat GPL`. Você pode ver os arquivos de `/usr/doc/copyright` com o comando `"ls"`. `/usr/doc/copyright` é o caminho de diretório que devemos percorrer para chegar até o arquivo `GPL`.
- Na segunda, é digitado o caminho completo para o `"cat"` localizar o arquivo `GPL`: `cat /usr/doc/copyright/GPL`. Neste caso, você continuará no diretório padrão (confira digitando `pwd`). Digitando `ls`, os arquivos do diretório atual serão listados.

O *caminho de diretórios* é necessário para dizer ao sistema operacional onde encontrar um arquivo na "árvore" de diretórios.

2.3.7 Exemplo de diretório

Um exemplo de diretório é o seu diretório de utilizador, todos seus arquivos essenciais devem ser colocadas neste diretório. Um diretório pode conter outro diretório, isto é útil quando temos muitos arquivos e queremos melhorar sua organização. Abaixo um exemplo de uma empresa que precisa controlar os arquivos de Pedidos que emite para as fábricas:

/pub/vendas - diretório principal de vendas /pub/vendas/mes01-99 - diretório contendo vendas do mês 01/1999 /pub/vendas/mes02-07 - diretório contendo vendas do mês 02/2007 /pub/vendas/mes01-08 - diretório contendo vendas do mês 03/2008

`mes01-99, mes02-07, mes01-08` são diretórios usados para armazenar os arquivos de pedidos do mês e ano correspondente. Isto é essencial para organização, pois se todos os pedidos fossem colocados diretamente no diretório vendas, seria muito difícil encontrar o arquivo do cliente "João" do mês 01/2007.
Você deve ter reparado que usei a palavra *sub-diretório* para mes01-99, mes02-07 e mes03-08, porque que eles estão dentro do diretório vendas. Da mesma forma, `vendas` é um sub-diretório de `pub`.

2.3.8 Estrutura básica de diretórios do Sistema Linux

O sistema `GNU/Linux` possui a seguinte estrutura básica de diretórios organizados segundo o FHS (Filesystem Hierarchy Standard):
`/bin`

Contém arquivos programas do sistema que são usados com freqüência pelos utilizadores.

`/boot`

Contém arquivos necessários para a inicialização do sistema.

`/cdrom`

Ponto de montagem da unidade de CD-ROM.

`/media`

Ponto de montagem de dispositivos diversos do sistema (rede, pen-drives, CD-ROM em distribuições mais novas).

`/dev`

Contém arquivos usados para acessar dispositivos (periféricos) existentes no computador.

`/etc`

Arquivos de configuração de seu computador local.

`/floppy`

Ponto de montagem de unidade de disquetes

`/home`

Diretórios contendo os arquivos dos utilizadores.

`/lib`

Bibliotecas compartilhadas pelos programas do sistema e módulos do kernel.

`/lost+found`

Local para a gravação de arquivos/diretórios recuperados pelo utilitário `fsck.ext2`. Cada partição possui seu próprio diretório `lost+found`.

`/mnt`

Ponto de montagem temporário.

`/proc`

Sistema de arquivos do kernel. Este diretório não existe em seu disco rígido, ele é colocado lá pelo kernel e usado por diversos programas que fazem sua leitura, verificam configurações do sistema ou modificar o funcionamento de dispositivos do sistema através da alteração em seus arquivos.

`/root`

Diretório do utilizador `root`.

`/sbin`

Diretório de programas usados pelo superutilizador (root) para administração e controle do funcionamento do sistema.

`/tmp`

Diretório para armazenamento de arquivos temporários criados por programas.

`/usr`

Contém maior parte de seus programas. Normalmente acessível somente como leitura.

`/var`

Contém maior parte dos arquivos que são gravados com freqüência pelos programas do sistema, e-mails, spool de impressora, cache, etc.

2.4 Nomeando Arquivos e Diretórios

No `GNU/Linux`, os arquivos e diretórios pode ter o tamanho de até *255* letras. Você pode identifica-lo com uma extensão (um conjunto de letras separadas do nome do arquivo por um ".").
Os programas executáveis do `GNU/Linux`, ao contrário dos programas de `DOS` e `Windows`, não são executados a partir de extensões `.exe, .com` ou `.bat`. O `GNU/Linux` (como todos os sistemas POSIX) usa a *permissão de execução* de arquivo para identificar se um arquivo pode ou não ser executado.
No exemplo anterior, nosso trabalho de história pode ser identificado mais facilmente caso fosse gravado com o nome `trabalho.text` ou `trabalho.txt`. Também é permitido gravar o arquivo com o nome `Trabalho de Historia.txt` mas não é recomendado gravar nomes de arquivos e diretórios com espaços. Porque será necessário colocar o nome do arquivo entre "aspas" para acessa-lo (por exemplo, `cat "Trabalho de Historia.txt"`). Ao invés de usar espaços, prefira *capitalizar* o arquivo (usar letras maiúsculas e minúsculas para identifica-lo): `TrabalhodeHistoria.txt`.

2.5 Comandos

Comandos são ordens que passamos ao sistema operacional para executar uma determinada tarefa.

Cada comando tem uma função específica, devemos saber a função de cada comando e escolher o mais adequado para fazer o que desejamos, por exemplo:

- `ls` - Mostra arquivos de diretórios
- `cd` - Para mudar de diretório

Este guia tem uma lista de vários comandos organizados por categoria com a explicação sobre o seu funcionamento e as opções aceitas (incluindo alguns exemplos).

É sempre usado um espaço depois do comando para separá-lo de uma opção ou parâmetro que será passado para o processamento. Um comando pode receber opções e parâmetros:

opções

As *opções* são usadas para controlar como o comando será executado, por exemplo, para fazer uma listagem mostrando o *dono, grupo, tamanho dos arquivos* você deve digitar `ls -l`.

Opções podem ser passadas ao comando através de um "-" ou "--":

-

Opção identificada por uma letra. Podem ser usadas mais de uma opção com um único hífen. O comando `ls -l -a` é a mesma coisa de `ls -la`

--

Opção identificada por um nome. Também chamado de opção extensa. O comando `ls --all` é equivalente a `ls -a`.

Pode ser usado tanto "-" como "--", mas há casos em que somente "-" ou "--" esta disponível.

parâmetros

Um parâmetro identifica o *caminho, origem, destino, entrada padrão* ou *saída padrão* que será passada ao comando.
Se você digitar: `ls /usr/share/doc/copyright`, `/usr/share/doc/copyright` será o parâmetro passado ao comando `ls`, neste caso queremos que ele liste os arquivos do diretório */usr/share/doc/copyright*.
É normal errar o nome de comandos, mas não se preocupe, quando isto acontecer o sistema mostrará a mensagem `command not found` (comando não encontrado) e voltará ao aviso de comando. As mensagens de erro não fazem nenhum mal ao seu sistema, somente dizem que algo deu errado para que você possa corrigir e entender o que aconteceu. No `GNU/Linux`, você tem a possibilidade de criar comandos personalizados usando outros comandos mais simples (isto será visto mais adiante). Os comandos se encaixam em duas categorias: *Comandos Internos* e *Comandos Externos*.

Por exemplo: `"ls -la /usr/share/doc"`, `ls` é o comando, `-la` é a opção passada ao comando, e `/usr/share/doc` é o diretório passado como parâmetro ao comando `ls`.

2.5.1 Comandos Internos

São comandos que estão localizados dentro do interpretador de comandos (normalmente o `Bash`) e não no disco. Eles são carregados na memória RAM do computador junto com o interpretador de comandos.
Quando executa um comando, o interpretador de comandos verifica primeiro se ele é um *Comando Interno* caso não seja é verificado se é um *Comando Externo*.
Exemplos de comandos internos são: `cd, exit, echo, bg, fg, source, help`

2.6 Comandos Externos

São comandos que estão localizados no disco. Os comandos são procurados no disco usando o `path` e executados assim que encontrados.
Para detalhes veja path, Seção 7.2.

2.7 Aviso de comando (Prompt)

Aviso de comando (ou Prompt), é a linha mostrada na tela para *digitação de comandos* que serão passados ao `interpretador de comandos` para sua execução.
A posição onde o comando será digitado é marcado um "traço" piscante na tela chamado de *cursor*. Tanto em shells texto como em gráficos é necessário o uso do cursor para sabermos onde iniciar a digitação de textos e nos orientarmos quanto a posição na tela.
O aviso de comando do utilizador `root` é identificado por uma "#" (tralha), e o aviso de comando de utilizadores é identificado pelo símbolo "$". Isto é padrão em sistemas `UNIX`.
Você pode retornar comandos já digitados pressionando as teclas `Seta para cima` / `Seta para baixo`.
A tela pode ser rolada para baixo ou para cima segurando a tecla `SHIFT` e pressionando `PGUP` ou `PGDOWN`. Isto é útil para ver textos que rolaram rapidamente para cima.

Abaixo algumas dicas sobre a edição da linha de comandos (não é necessário se preocupar em decora-los):

- Pressione a tecla `Back Space` ("**<--**") para apagar um caracter à esquerda do cursor.
- Pressione a tecla `Del` para apagar o caracter acima do cursor.
- Pressione `CTRL+A` para mover o cursor para o inicio da linha de comandos.
- Pressione `CTRL+E` para mover o cursor para o fim da linha de comandos.
- Pressione `CTRL+U` para apagar o que estiver à esquerda do cursor. O conteúdo apagado é copiado para uso com `CTRL+y`.
- Pressione `CTRL+K` para apagar o que estiver à direita do cursor. O conteúdo apagado é copiado para uso com `CTRL+y`.
- Pressione `CTRL+L` para limpar a tela e manter o texto que estiver sendo digitado na linha de comando (parecido com o comando `clear`).
- Pressione `CTRL+Y` para colocar o texto que foi apagado na posição atual do cursor.

2.8 Interpretador de comandos

Também conhecido como "shell". É o programa responsável em interpretar as instruções enviadas pelo utilizador e seus programas ao sistema operacional (o kernel). Ele que executa comandos lidos do dispositivo de entrada padrão (teclado) ou de um arquivo executável. É a principal ligação entre o utilizador, os programas e o kernel.
O `GNU/Linux` possui diversos tipos de interpretadores de comandos, entre eles posso destacar o `bash, ash, csh, tcsh, sh,` etc. Entre eles o mais usado é o `bash`. O interpretador de comandos do DOS, por exemplo, é o `command.com`.
Os comandos podem ser enviados de duas maneiras para o interpretador: `interativa` e `não-interativa`:
`Interativa`

Os comandos são digitados no aviso de comando e passados ao interpretador de comandos um a um. Neste modo, o computador depende do utilizador para executar uma tarefa, ou próximo comando.

`Não-interativa`

São usados arquivos de comandos criados pelo utilizador (scripts) para o computador executar os comandos na ordem encontrada no arquivo. Neste modo, o computador executa os comandos do arquivo um por um e dependendo do término do comando, o script pode checar qual será o próximo comando que será executado e dar continuidade ao processamento.

Este sistema é útil quando temos que digitar por várias vezes seguidas um mesmo comando ou para compilar algum programa complexo.

O shell `Bash` possui ainda outra característica interessante: A completação dos nomes. Isto é feito pressionando-se a tecla TAB. Por exemplo, se digitar "ls tes" e pressionar <tab>, o `Bash` localizará todos os arquivos que iniciam com "tes" e completará o restante do nome. Caso a completação de nomes encontre mais do que uma expressão que satisfaça a pesquisa, ou nenhuma, é emitido um beep. Se você apertar novamente a tecla TAB imediatamente depois do beep, o interpretador de comandos irá listar as diversas possibilidades que satisfazem a pesquisa, para que você possa escolher a que lhe interessa. A completação de nomes funciona sem problemas para comandos internos.
Exemplo: `ech` (pressione TAB). `ls /vm`(pressione TAB)

2.9 Terminal Virtual (console)

Terminal (ou console) é o teclado e tela conectados em seu computador. O `GNU/Linux` faz uso de sua característica *multi-usuária* usando os "terminais virtuais". Um terminal virtual é uma segunda seção de trabalho completamente independente de outras, que pode ser acessada no computador local ou remotamente via `telnet, rsh, rlogin,` etc.
No `GNU/Linux`, em modo texto, você pode acessar outros terminais virtuais segurando a tecla `ALT` e pressionando `F1 a F6`. Cada tecla de função corresponde a um número de terminal do 1 ao 6 (o sétimo é usado por padrão pelo ambiente gráfico X). O `GNU/Linux` possui mais de 63 terminais virtuais, mas apenas 6 estão disponíveis inicialmente por motivos de economia de memória RAM (cada terminal virtual ocupa aproximadamente 350 Kb de memória RAM, desative a quantidade que não estiver usando para liberar memória RAM para uso de outros programas!) .
Se estiver usando o modo gráfico, você deve segurar `CTRL`+ `ALT` enquanto pressiona uma tela de <F1> a <F6>. Para voltar ao modo gráfico, pressione`CTRL`+`ALT`+ <F7>.
Um exemplo prático: Se você estiver usando o sistema no Terminal 1 com o nome "joao" e desejar entrar como "root" para instalar algum programa, segure`ALT` enquanto pressiona <F2> para abrir o segundo terminal virtual e faça o login como "root". Será aberta uma nova seção para o utilizador "root" e você poderá retornar a hora que quiser para o primeiro terminal pressionando `ALT`+<F1>.

2.10 Login

Login é a entrada no sistema quando você digita seu *nome* e *senha*. Por enquanto vou manter o seu suspense sobre o que é o *logout*.

2.11 Logout

Logout é a saída do sistema. A saída do sistema é feita pelos comandos `logout`, `exit`, `CTRL+D`, ou quando o sistema é reiniciado ou desligado.

2.12 Curingas

Curingas (ou referência global) é um recurso usado para especificar um ou mais arquivos ou diretórios do sistema de uma só vez. Este é um recurso permite que você faça a filtragem do que será listado, copiado, apagado, etc. São usados 4 tipos de curingas no `GNU/Linux`:

- "*" - Faz referência a um nome completo/restante de um arquivo/diretório.
- "?" - Faz referência a uma letra naquela posição.
- `[padrão]` - Faz referência a uma faixa de caracteres de um arquivo/diretório. Padrão pode ser:
- `[a-z][0-9]` - Faz referência a caracteres de `a` até `z` seguido de um caracter de `0` até `9`.
- `[a,z][1,0]` - Faz a referência aos caracteres `a` e `z` seguido de um caracter `1` ou `0` naquela posição.
- `[a-z,1,0]` - Faz referência a intervalo de caracteres de `a` até `z` ou `1` ou `0` naquela posição.
  A procura de caracteres é "Case Sensitive" assim se você deseja que sejam localizados todos os caracteres alfabéticos você deve usar `[a-zA-Z]`.
  Caso a expressão seja precedida por um ^, faz referência a qualquer caracter exceto o da expressão. Por exemplo `[^abc]` faz referência a qualquer caracter exceto `a`, `b` e `c`.
- `{padrões}` - Expande e gera strings para pesquisa de padrões de um arquivo/diretório.
- `X{ab,01}` - Faz referência a seqüencia de caracteres `Xab` ou `X01`
- `X{a-z,10}` Faz referencia a seqüencia de caracteres `Xa-z` e `X10`.

O que diferencia este método de expansão dos demais é que a existência do arquivo/diretório é opcional para geração do resultado. Isto é útil para a criação de diretórios. Lembrando que os 4 tipos de curingas ("*", "?", "[]", "{}") podem ser usados juntos. Para entender melhor vamos a prática:

Vamos dizer que tenha 5 arquivo no diretório `/usr/teste`: `teste1.txt, teste2.txt, teste3.txt, teste4.new, teste5.new`.
Caso deseje listar **todos** os arquivos do diretório `/usr/teste` você pode usar o coringa "*" para especificar todos os arquivos do diretório:
`cd /usr/teste` e `ls *` ou `ls /usr/teste/*`.
Não tem muito sentido usar o comando `ls` com "*" porque todos os arquivos serão listados se o `ls` for usado sem nenhum Coringa.
Agora para listar todos os arquivos `teste1.txt, teste2.txt, teste3.txt` com excessão de `teste4.new`, `teste5.new`, podemos usar inicialmente 3 métodos:

- Usando o comando `ls *.txt` que pega todos os arquivos que começam com qualquer nome e terminam com `.txt`.
- Usando o comando `ls teste?.txt`, que pega todos os arquivos que começam com o nome `teste`, tenham qualquer caracter no lugar do coringa `?` e terminem com `.txt`. Com o exemplo acima `teste*.txt` também faria a mesma coisa, mas se também tivéssemos um arquivo chamado `teste10.txt` este também seria listado.
- Usando o comando `ls teste[1-3].txt`, que pega todos os arquivos que começam com o nome `teste`, tenham qualquer caracter entre o número 1-3 no lugar da 6a letra e terminem com `.txt`. Neste caso se obtém uma filtragem mais exata, pois o coringa *?* especifica qualquer caracter naquela posição e [] especifica números, letras ou intervalo que será usado.

Agora para listar somente `teste4.new` e `teste5.new` podemos usar os seguintes métodos:

- `ls *.new` que lista todos os arquivos que terminam com `.new`
- `ls teste?.new` que lista todos os arquivos que começam com `teste`, contenham qualquer caracter na posição do coringa *?* e terminem com `.new`.
- `ls teste[4,5].*` que lista todos os arquivos que começam com `teste` contenham números de 4 e 5 naquela posição e terminem com qualquer extensão.

Existem muitas outras formas de se fazer a mesma coisa, isto depende do gosto de cada um. O que pretendi fazer aqui foi mostrar como especificar mais de um arquivo de uma só vez. O uso de curingas será útil ao copiar arquivos, apagar, mover, renomear, e nas mais diversas partes do sistema. Alias esta é uma característica do `GNU/Linux`: permitir que a mesma coisa possa ser feita com liberdade de várias maneiras diferentes.