04

# Utilizando o comando man para obter ajuda

## Man

Vamos falar sobre diversos programas e outras aplicações que podemos usar para pegar informações. Ao mesmo tempo que falaremos sobre o `--help`, vamos abordar também um meio de pegar informações e que não é necessariamente utilizando a linha de comando. Esse meio é o navegador.

Sempre podemos procurar `linux ls help` no *Google* e, com isso, podemos obter a ajuda necessária sobre algum comando. Por exemplo, após realizarmos uma busca no *Google* podemos acessar algum dos *links*, no caso, entraremos no [(http://linuxcommand.org/man_pages/ls1.html)http://linuxcommand.org/man_pages/ls1.html (http://linuxcommand.org/man_pages/ls1.html)](http://linuxcommand.org/man_pages/ls1.html).


Essa página de ajuda é razoavelmente interessante. Ela parece um manual e traz as informações necessárias sobre o comando `ls`, que é um comando de usuário e que lista o conteúdo de um diretório, as opções e os arquivos. Ele lista as informações sobre os arquivos que falamos e, por padrão, mostra o diretório atual. Como podemos observar, o uso dos colchetes nos indicam que isso é opcional:

```
ls [OPTION]... [FILE]
```

Esse comando organiza alfabeticamente as coisas, a não ser que passemos algumas das seguintes opções `-c`, `-f`, `-x`, `-t`, `-s`, `-S` ou `--source` que ele ordena de maneira distinta. Nessa página encontramos as diversas opções que possuímos, por exemplo, quem é o autor desse comando, no caso nos é informado que é Richard Stallmanse, se tivermos um *bug* no `ls` para onde deveremos enviar um e-mail, menciona o *copyright* e outros lugares onde podemos buscar informações.

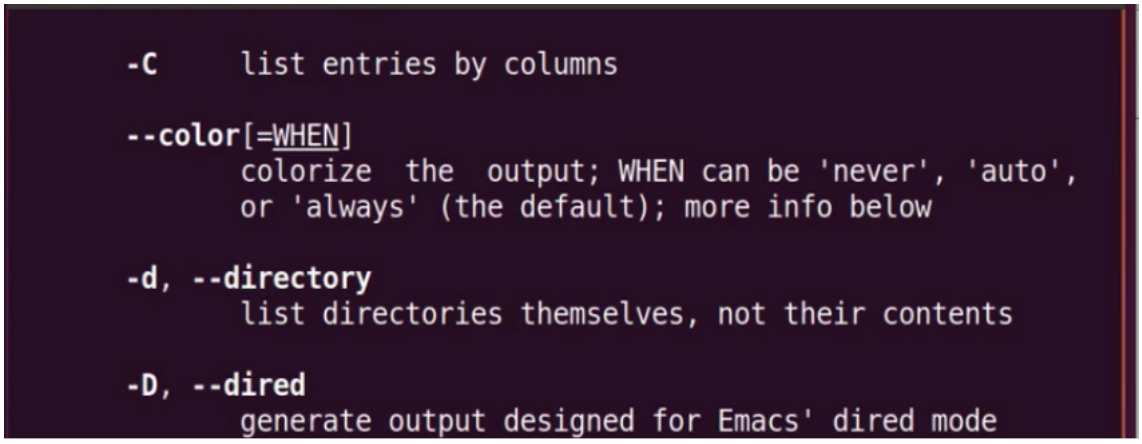
Será que a única maneira de chegar nessa documentação é através do navegador?

Vamos para o nosso terminal!

Dentro do nosso terminal vamos digitar `manual ls` ou `man ls` , onde esta segunda opção é uma versão abreviada. Digitando isso estamos pedindo o manual do comando `ls` e após escrever essas duas palavras e dando um "Enter" nisso teremos diversas informações, como `LS(1)` , User Commands* e etc.

Todas essa opções, na verdade, se parecem bastante com a página da Internet que acabamos acessando em nossa pesquisa no *google*. O *google* possui diversos sites que armazenam manuais que estão disponíveis no *Linux*.

Já aprendemos que para acessar esses materiais no *Linux* devemos digitar `man ls` e ele entra no visualizador do comando. Podemos usar a seta para cima e para baixo para navegar nesse manual e quando quisermos sair basta fazer o que está escrito em baixo digitando `h` para `help` e `q` de `quit` para sair.

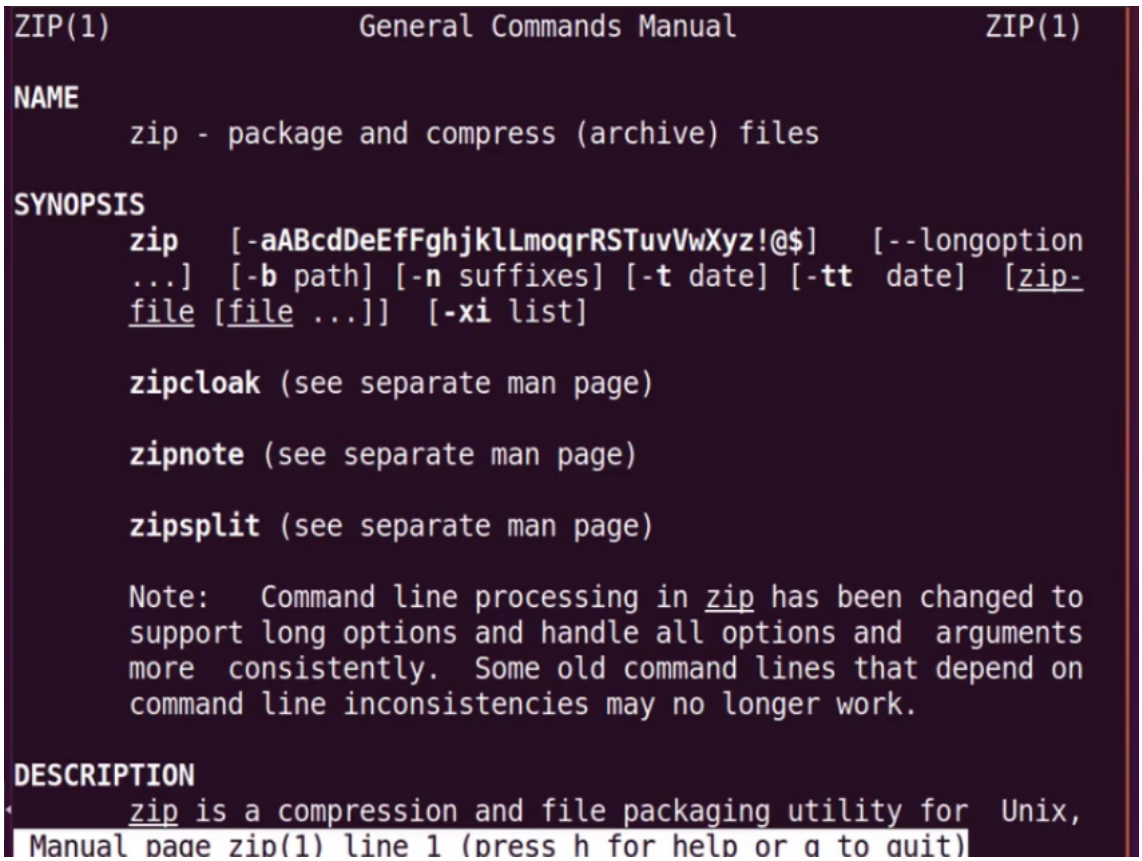



Retomando o `man ls` traz o manual do comando, mas ele também é encontrado no navegador e nos mais diversos sites. Alguns links trazem o manual traduzido parcialmente e outros não. O ideal é tentarmos usar em inglês uma vez que a prova cobra os comandos, justamente, nesse idioma. Então, temos que nos acostumar as palavras que aparecem aqui.

O `man` é extremamente importante para nós. Ele traz a página do manual de comando que o acompanha. No caso, digitando `man ls` ele traz a página do manual desse comando.

O que mais podemos pedir?

Vamos pegar um outro comando para testar, podemos utilizar o `man zip` e teremos o seguinte:




E se pegarmos um *builtin* do *shell*? Será que teremos também? Se digitarmos `man pwd` teremos um manual:


Mas, lembre-se que o `pwd` é um *builtin* do *shell* e se digitarmos `type pwd` é, justamente, o que ele nos informa: `pwd is a shell builtin .`

Mas, se dermos um `type -a pwd` veremos todas as suas variações e que ele também possui um script no `/bin/pwd` :

```
> type -a pwd
pwd is a shell builtin
pwd is /bin/pwd
```

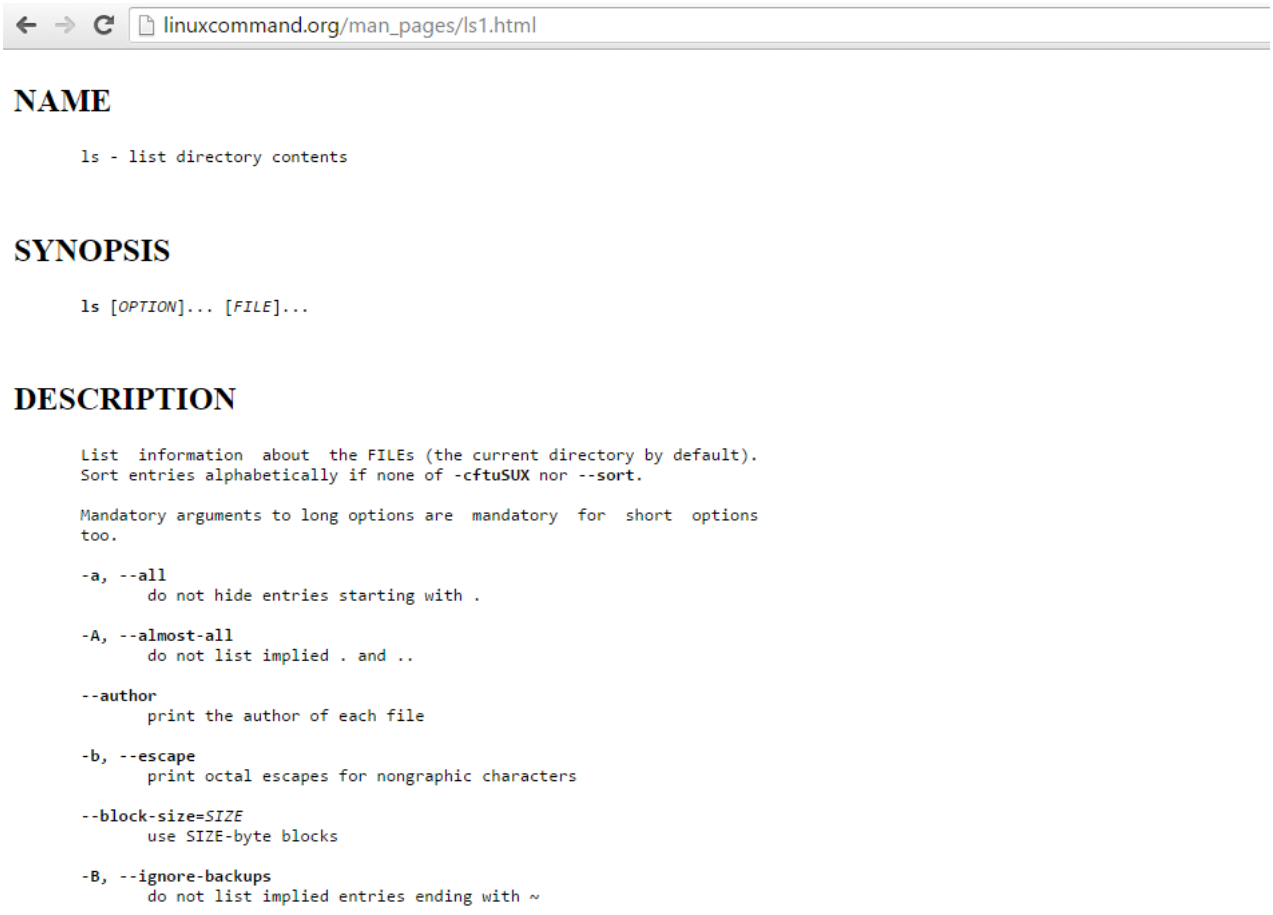
Quem era um *shell builtin*? O `help`. E, será que o `exit` também é? É só um *shell builtin*:

```
> type -a help
help is a shell builtin
> type -a exit
exit is a shell builtin
```

Será que por serem *shell builtin* eles possuem manual? Vamos testar!

```
> man help
No manual entry for help
```

O `help` não possui manual, mas se dermos um `man exit` vemos que ele possui um manual. Repare que nem todos os *shell builtin* possuem um manual. Alguns possuem e outros não, entretanto, a grande maioria do que vêm no *Linux* possui um manual. E esse manual é dividido em diversas sessões. Algumas das sessões são muito importantes, como, por exemplo, o *Name*, a *Synopsis* que dá uma noção de como evocamos esse comando, a *Description* que nos traz o que esse comando faz e quais os efeitos que ele vai causar em nosso programa e etc.




Teremos diversas opções que podem ser usadas com o comando:

SELinux options:

**--lcontext**
Display security context. Enable -l. Lines will probably be too wide for most displays.

**-Z, --context**
Display security context so it fits on most displays. Displays only mode, user, group, security context and file name.

**--scontext**
Display only security context and file name.

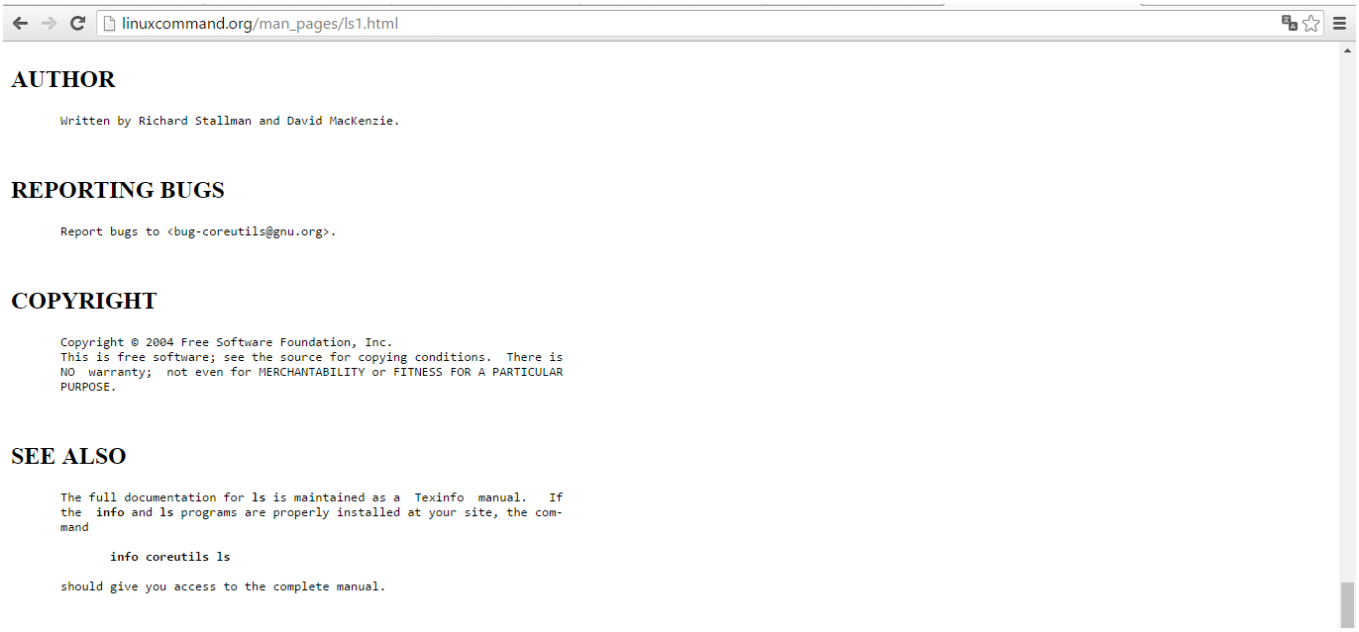
**--help** display this help and exit

**--version**
output version information and exit

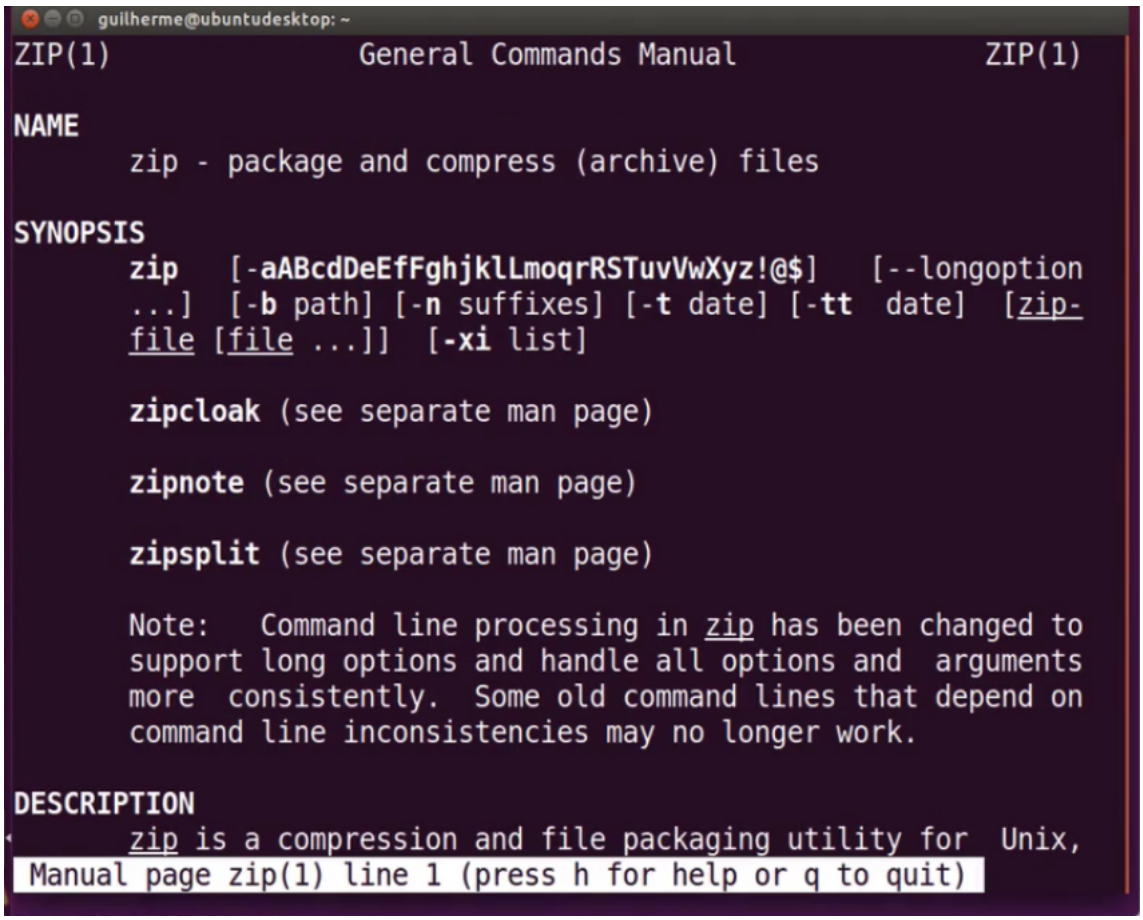
SIZE may be (or may be an integer optionally followed by) one of following: kB 1000, K 1024, MB 1000*1000, M 1024*1024, and so on for G, T, P, E, Z, Y.

By default, color is not used to distinguish types of files. That is equivalent to using **--color**=*none*. Using the **--color** option without the optional WHEN argument is equivalent to using **--color**=*always*. With **--color**=*auto*, color codes are output only if standard output is connected to a terminal (tty).

E, temos também, o argumento, o *file*, que está dentro da *descrição*, que é onde ele vai buscar. As vezes temos alguns exemplos, nesse caso temos direto o Autor, Reporte em caso de Bug, *Copyright* e Veja Mais, que contêm informações sobre outros comandos ligados.




Repare que se escrevermos `man ls` no Terminal teremos também informações de sinopse, descrição, opções, autor, como reportar os *bugs* e *Copyright*, isto é, os mesmos setores que o site nos informava:




E o `zip` ? Se dermos um `man zip` encontraremos, nome, sinopse, descrição, autor, uso como usar e quando usar, opções, exemplos e etc. O `zip` nos mostra exemplos, ele mostra como fazer o *match* de padrões, como vimos com o *globbing*, o `zip` possui uma espécie de *globbing* específico dele. Assim, podemos passar para ele se quisermos extrair ou compactar arquivos.

Essas questões estão explicadas no *Pattern Matching*. Repare que os padrões dessas seções não são totalmente valorizadas em todos os comandos e programas do *Linux*, esse padrão de seções nome, sinopse, exemplos, copyright e etc é apenas um dos padrões que existe. Esses são alguns padrões que aparecem, entretanto, o programa pode decidir ter outras seções em nosso manual.

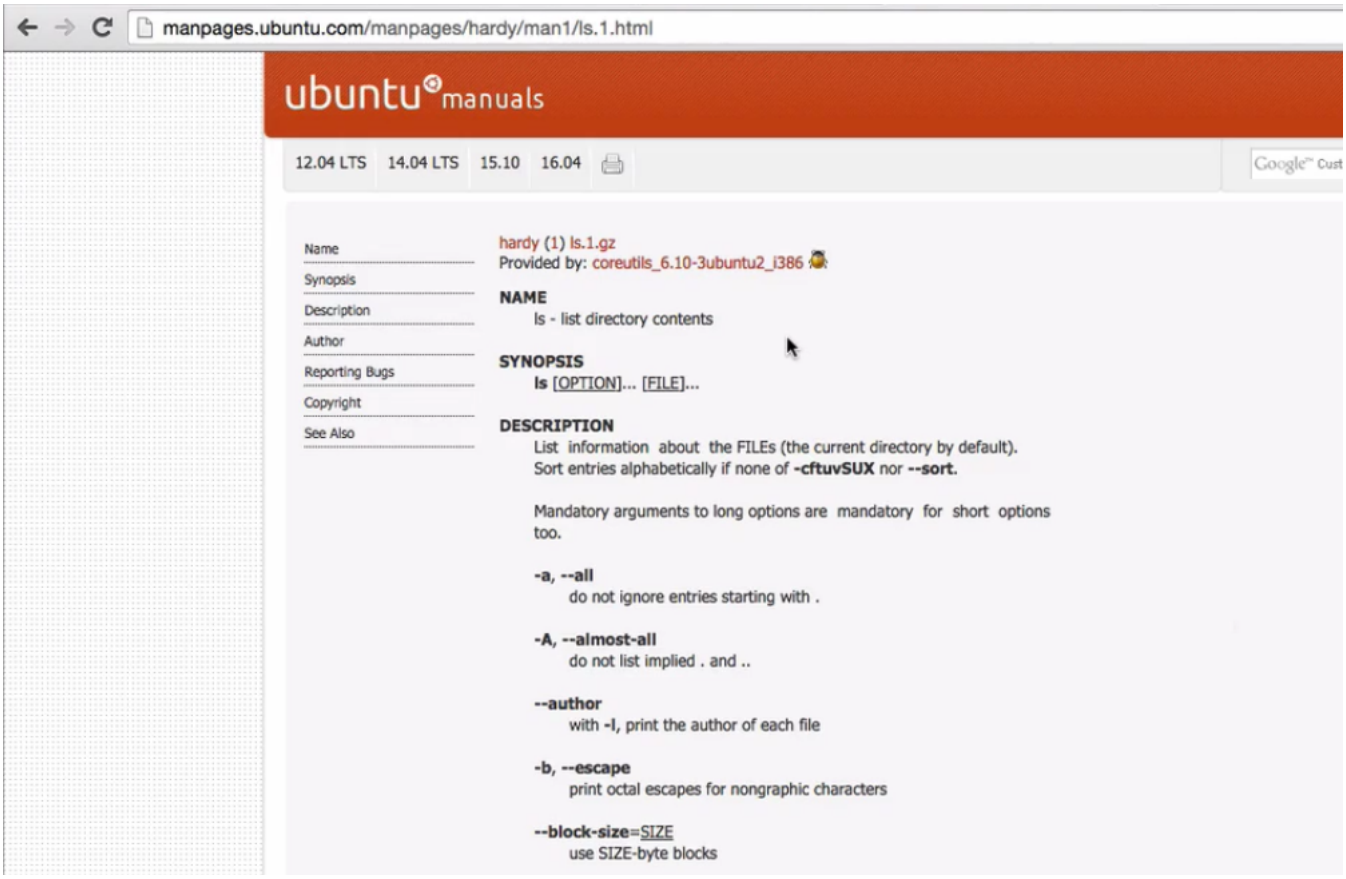
Então, o manual que consultamos na Internet é um *online manual* e essa documentação está tanto aqui no terminal, que é extremamente útil no dia a dia, quanto no navegador. Para consultar no terminal basta utilizar o `man` ou, se quisermos consultar online podemos fazer uma pesquisa, digitando também `man zip` no *Google*. Aparecerão diversos links contendo informações a respeito desse tema.

Vamos digitar no navegador *linux man zip* para ver se as informações se tornam mais específicas, vamos clicar no link http://linux.die.net/man/1/zip (http://linux.die.net/man/1/zip). Nessa página encontraremos a descrição, o uso e etc.

Mas, temos que ter atenção!

As vezes o `zip` do manual online é de uma versão diferente da que estamos usando. É um manual de uma versão de *Unix* que não é a do *Linux* que estamos utilizando. Assim, não necessariamente o comando manual que achamos na Internet é exatamente compatível com o que estamos utilizando. Existem pequenas diferenças e, assim, talvez elas não estejam descritas no manual que encontrarmos pela Internet e que possui uma versão distinta. Por esse motivo é importante que ao buscarmos isso no *google* se mencione o sistema operacional.

No nosso caso, existe uma preferência por escrever *Ubuntu man ls*. Com isso é possível que abra um link do próprio *Ubuntu*, o que é preferível, como o seguinte link http://manpages.ubuntu.com/manpages/wily/man1/ls.1.html (http://manpages.ubuntu.com/manpages/wily/man1/ls.1.html)




Assim, temos um pouco mais de certeza de que estamos pegando o programa na variação específica dele que queremos. É claro que se possuímos um sistema operacional instalado, podemos navegador, diretamente, no Terminal com o `man` e o nome do comando. E essa é uma sacada que podemos utilizar nesse contexto.

Várias distribuições tentam colocar para nós uma página de manual, eletrônica, para cada comando que existe. Mas, como vimos nem todos os comandos como o *bash* e o *help* nem possuem página de manual. A maior parte dos comandos que temos possuem um manual.

## Muito mais sobre o man

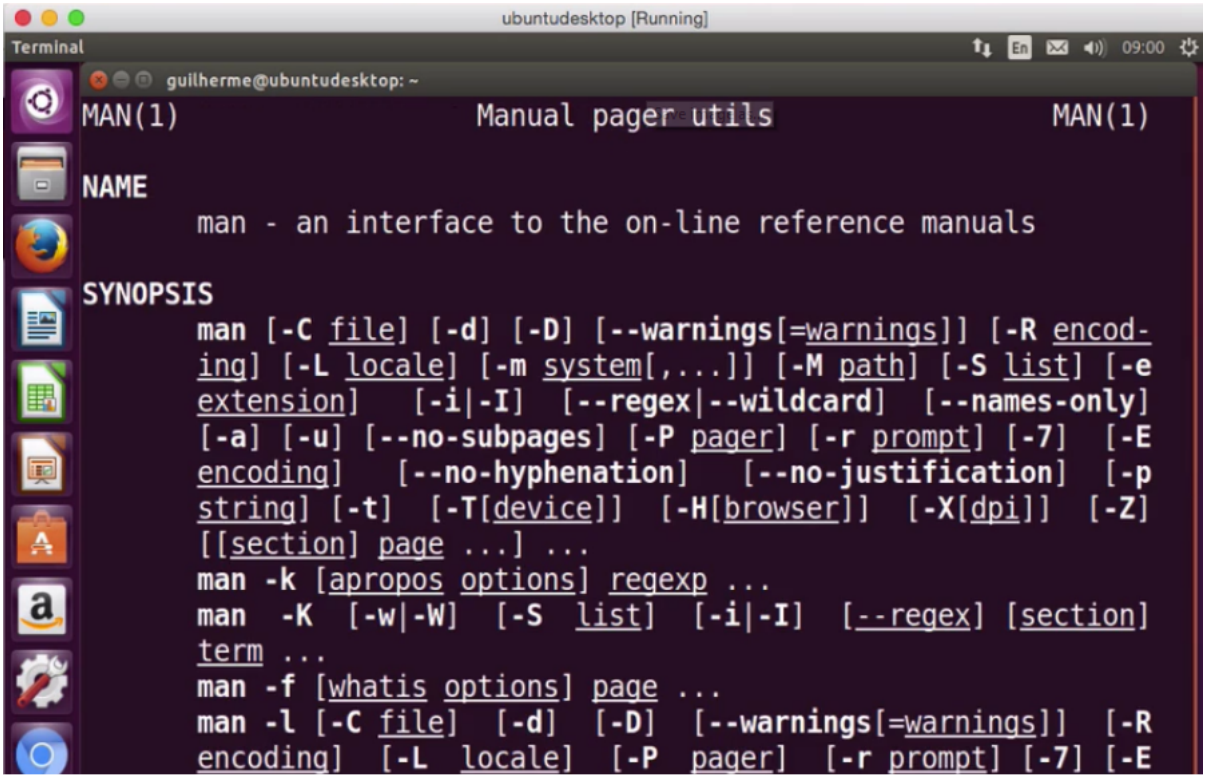
É legal saber que o *Linux* já vem com manuais eletrônicos para pegarmos ajuda deles, além do comando `help` temos o comando `man` . O `man` é oriundo de manual eletrônico, mas não no sentido de ele estar na internet, mas sim de que ele está digital. Assim, o `man` serve para pegarmos esses manuais. E onde estão eles? Lembra-se do comando `whereis` ? Ele indica onde estão localizadas as coisas. Então, se digitarmos `whereis ls` ele nos informa o seguinte:

```
> whereis ls
ls: /bin/ls /usr/share/man/man1/ls.1.gz
```

Ele nos informa que o `ls` possui arquivos no `/bin/ls` e no `usr/share/man/man1/ls.1.gz` . Esse segundo é o diretório onde ficam os manuais por padrão.

Mas, o que significa o `man/man1/ls.?.gz` ?

O `gz` significa que está `zipado` no formato de "gzip" e que, portanto, o manual está compactado dentro. O "ls" é o nome do comando. O número "1" significa que tipo de comando é esse. Estamos sempre usando a palavra "comando", mas podem ser utilizadas outras nomenclaturas. Para sabermos qual é a seção, qual o tópico relativo ao que estamos querendo executar vamos dar uma olhada no `man` do próprio `man`, para isso digitaremos `man man` e teremos o seguinte:



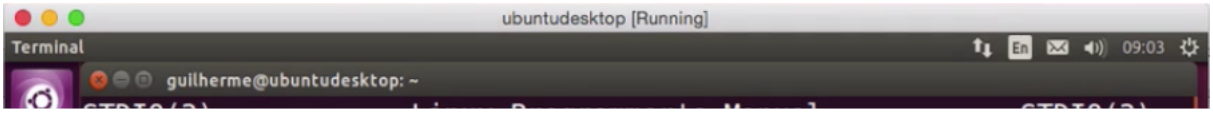

Aqui temos uma interface para os manuais de referência que estão digitalizados e nessa tela temos diversas explicações de várias coisas. Por exemplo, as seções, os tópicos, estão separados em nove áreas, a primeira é: (1) programas executáveis ou comandos do *shell*. Por isso, quando buscamos no *google* o `ls help` e caímos naquela primeira página temos a indicação de um número 1.


Lembra-se que ele falava `ls1` ? O número "1" indica a nossa sessão, isto é o "*Executable programs or shell commands*". Se fosse uma outra coisa, por exemplo, um jogo, ele estaria sendo indicado na sessão 6, de *Games*. Se pegássemos uma rotina de "Kernel" estaria na sessão 9 e assim por diante. Mas, atenção, isso não é um padrão! Essas são as seções que existem, na 1, temos os programas executáveis ou comandos do *shell*, como é o caso do `ls` que está nessa seção número 1.

E que outras seções existem?

Temos os sistemas de evocações, de bibliotecas, funções de sistemas, arquivos especiais, formatos de arquivos , jogos e etc. Perceba que temos diversas seções para o `man` .

Vamos pegar um outro exemplo que não seja o man do próprio `man` ? Vamos pegar o `man` de um arquivo. Por exemplo, o `man stdio` que serve para programar entrada e saída padrão de um programa, onde "std" significa abreviação da palavra "padrão" em inglês, *standart*, o "i" que é de *input* e o "o" que é de "output". Esse é o nome do arquivo. Então, se digitarmos `man stdio` e dermos um "Enter" aparecerá a seguinte tela:




Vamos dar um `man` no próprio `man` , novamente, para visualizarmos o que era a seção 3. Vemos que são as funções dentro das bibliotecas que acessamos. O `stdio` tem diversas funções para acessarmos, tanto de entrada quanto de saída. O `stdio` não é o arquivo em si, este é o `stdio.h` que incluímos em nossos programas em "c".

Repare que dependendo sobre o que estamos pedindo ajuda ele estará em uma seção diferente, podendo estar em uma seção de 1 até 9. Repare que ele está falando que a seção vai de 1 até 9 e dentro de uma página do `man` temos diversas seções, o nome, a sinopse, status de saída e etc. Aqui, o `man` possui um certo padrão, mas vimos que no `zip` ele pode criar alguns outros padrões. Não necessariamente ele precisa seguir as seções dentro de uma página. E cuidado com a palavra "seção", o manual do `man` usa essa palavra para dizer se ele está do 1 até o 9, isto é, usa a palavra para indicar o número do seu tópico, sua categoria. Ele também usa a palavra "seção" para delimitar o *name*, *sinopsis*, *description* e etc. Ele usa a palavra "seção" tanto para delimitar o número 1 que aparece ao lado do ls que tínha no site, o `LS(1)` quanto para delimitar o nome, a sinopses e etc.

Cuidado, o manual do `man` está usando a palavra seção para identificar as duas coisas!

Costumamos fazer uma divisão, chamamos, por exemplo, o "Nome" de seção e o número (1) de tópico. O tópico pertence a um capítulo, então o capítulo vai de 1 até 9. Podemos ter outros números, se o usuário quiser escolher, mas nesse caso não teremos nenhum padrão mesmo.

Repare que mesmo dentro desses tópicos, como se fossem partes de um livro, teremos diversos capítulos, mas que as vezes podemos encontrar uma palavra que acaba aparecendo em dois capítulos diferentes.

Vamos apresentar uma palavra que é tanto um comando quanto um arquivo de configuração. Essa palavra é extremamente famosa, ela se chama *crontab* e isso permite que configuremos programas ou comandos a serem executados com uma cerca frequência. É um programa que permite fazermos essas configurações. Ele é o *contrab 1*, mas ele também é o nome de um arquivo, no caso o `contrab 5` que possui essa configuração. Lembra-se de algo que mencionamos? Dependendo do que

linkamos estamos em uma documentação de sistema operacional ou em outra. Por isso, vamos pesquisar o que o `contrab` significa, para tanto, especificaremos o sistema operacional `ubuntu contrab man`.

***Atenção!*** Temos que atentar quando buscarmos algo no navegador, de preferência é preciso especificar o sistema operacional explícito.

Vimos que temos diversos capítulos que o próprio manual chamou de seções. Então, vamos imaginar que "ajuda" é um livro e que ele possui vários capítulos, ou *chapters*. Nesses diversos capítulos, cada um possui diversas páginas e cada página é um comando, que possui uma página de um manual. E cada uma dessas páginas possui diversas seções, por exemplo, nome, descrição, sinopse e etc. Então, cada uma das páginas do manual vai ter diversas seções, isto é, as seções que ele bem entender.

Então, o `man` é o comando usado para acessar o manual online e utilizamos ele para buscar um comando. Para acessar o manual do `zip` basta digitarmos `man zip`. E como estamos navegando no manual do `zip`? Utilizamos as setas de "para cima" e o "para baixo" para percorrer o texto. Além disso, podemos usar alguns outros comandos, por exemplo, o espaço que nos faz ir para a próxima página da nossa tela.

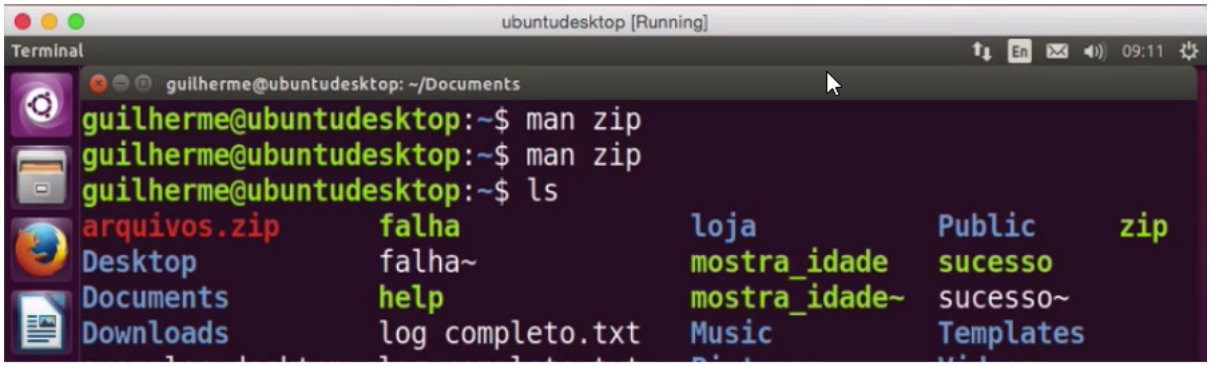
E se, quisermos encontrar, especificamente, o `zip -r` no manual?

Como o manual do `zip` é muito grande, podemos utilizar a `/` para nos facilitar a procura. Digitamos a `/` e mais o que queremos buscar, por exemplo, o `-r`, ficaremos com `/-r`. Ele mostrará os resultados e se não encontrarmos `-r` podemos digitar o "n" para passar para o próximo resultado até encontrar o que estamos buscando.

Bom, encontramdo o que queríamos, o `-r` ou o `-recurse-paths`, temos as suas definições, eles são responsáveis por uma busca que vai entrando nos diretórios.

Apenas retomando, entramos no manual, digitando `man zip` e usamos as setas para navegar na página, usamos o espaço para ir para a próxima página, a barra ( `/` ) seguida de um nome que queiramos buscar, por exemplo `/recursive` que a busca será realizada e para procurarmos novamente é preciso digitar `n` e para sair usamos o `q`.

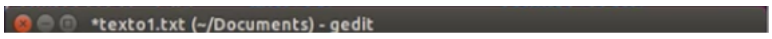
Por padrão esse visualizador de texto que usamos no `man` é um comando chamado `less` e ele possui, justamente, essa função, ser um visualizador de textos.Então, se temos um texto grande, por exemplo, os arquivos de log que estão no `cd Documents/`:




Podemos digitar o comando `gedit texto1.txt` para abrir o editor com esse arquivo e teremos o seguinte texto:

"o conteudo do primeiro arquivo"

Se dermos um "Enter" após essa primeira e única linha e escreveremos "Mais uma linha" podemos reparar que o editor nos indica lá embaixo a linha que estamos. Assim, vamos copiar esse "Mais uma linha" nas várias linhas seguintes. Observe:




Vamos colar diversas vezes isso com o propósito de termos um arquivo único e gigante. Vamos escrever na penúltima linha "linha 253" e na última linha "Mais conteúdo". Vamos fechar esse arquivo e vamos digitar no terminal `less texto1.txt` evocando o texto do arquivo que estávamos modificando. Teremos o seguinte:


É o mesmo visualizador, então, utilizamos as setas para navegar, espaço para ir para frente, `/` para buscarmos alguma coisa, então, por exemplo, vamos buscar aquela "Linha 253", digitando `/25` ele busca isso para nós e usamos o `q` para sair.

O visualizador do `man`, por padrão, é o visualizador chamado `less` que serve para visualizarmos um arquivo e que é bem útil no nosso dia a dia. Então, além disso, que outra maneira podemos navegar com o `man`?

Ainda podemos falar para o `man` que queremos buscar uma palavra que deve aparecer em diversos documentos, e como não sabemos, exatamente. O que queremos procurar é o tal do `ascii`. Então, digitando `man ascii` e dando um "Enter", teremos o seguinte:

Teremos diversas informações, mas não é bem isso, ainda, queríamos buscar o `ascii` em outros documentos. Vamos usar, então, o `man -k` que faz com que sejam buscadas as palavras-chaves, as *keywords*, daqui vêm o `k`.

Dando um "Enter" ele nos traz diversas páginas de comando e arquivos que trazem uma referência a palavra chave `ascii`. Então, por exemplo, o `isascii` é uma função que diz se um carácter é `ascii` ou não é. O `strtod` converte um `ascii` para uma `floating point`. Temos mais em cima o próprio `ascii`, o `man ascii` traz mais informações acerca do `ascii`.

Repare que com o `-k` buscamos a palavra chave e buscamos diversas referências para a palavra chave. É bastante utilizado no dia a dia. As vezes sabemos o programa com o qual queremos pegar ajuda, por exemplo, podemos utilizar o `man unzip` ou podemos também buscar o `man -k printf` e teremos diversos `printf` com informações sobre eles.

Resumindo o `man` é um programa fundamental no dia a dia de um usuário do *Linux*. Para obter informações a partir dele podemos digitar `man` + o nome de alguma coisa e se não sabemos exatamente o que estamos buscando e queremos procura

algo usando uma palavra chave podemos digitar `man -k` + a palavra que estamos buscando. E onde estão armazenadas essas documentações? Para descobrir isso digitamos `whereis` + o nome que queremos encontrar, por exemplo `whereis crontab` .

Vamos digitar isso e ver o que aparece:

```
> whereis crontab
crontab: /usr/bin/crontab /etc/crontab /usr/share/man/man1/crontab.1.gz /usr/share/man/man5/crontab
```

Olhando esse resultado vemos que o programa `crontab` está no "/usr/bin/crontab" e que o arquivo encontra-se no "/etc/crontab", o manual do capítulo 1 está no "/usr/share/man/man1/crontab.1.gz" e temos também o capítulo 5, no "/usr/share/man/man5/crontab.5.gz".

Os manuais são armazenados de forma compactada, quando acessamos o manual é que ele vai descompactar o arquivo e nos mostrar a informação ou qualquer variação disso. Não importa como o `man` faz isso internamente, o importante é que ele busca essa informação, encontra e traz essa página do manual para nós.

Se formos no `man` do próprio `man` , digitando `man man` acharemos informações sobre ele, por exemplo, de como ele funciona no dia a dia e de como podemos utilizá-lo . Se não estivermos com o terminal aberto, podemos buscar a partir do próprio navegador.

Como sugerimos buscar no navegador?

É mais seguro e específico a busca que inclui o nome do sistema operacional, por exemplo, o `man` e o nome do comando que queremos buscar. Com isso, acharemos uma página específica daquele sistema operacional e que pode trazer informações específicas sobre esse sistema que pode não coincidir com outro sistema. O ideal é executar no terminal, pois, com isso, estará sendo executada a versão própria que estamos utilizando. Pode ocorrer, no caso de usarmos um navegador, de versões mais novas e antigas de uma mesma coisa possuírem diferenças e isso pode estar, eventualmente, desatualizado ao buscarmos no navegador.

O `man` , nosso terminal online não significa, necessariamente, que é algo que está na rede, mas sim que é algo digital. Esse `man` usado no terminal nos traz informações sobre a versão específica, buscando no navegador temos, sempre, que conferir as informações o que acaba tornando o processo mais trabalhoso no dia a dia. Além de corrermos o risco de, por exemplo, digitarmos apenas `man contab` e de repente pegarmos informações de um sistema operacional que é diferente do nosso e os padrões e opções funcionarem de uma maneira diferente. O que pode nos trazer erros na execução do comando. A dica é usar o `man` do terminal.

## Man por trás dos panos

Agora que já vimos diversas funcionalidades do `man` , gostaríamos de mostrar como funciona por trás dos panos. Quando digitamos `whereis ls` ele nos contesta o seguinte:

```
> whereis ls
ls: /bin/ls /usr/share/man/man1/ls.1.gz
```

Ele nos mostra os dois locais onde o `ls` está armazenado.

Mas, como o `man` sabe que os manuais estão em um diretório específico? Como o `shell` fazia para saber que os programas a serem executados estavam nesse diretório ou em outros?

Usando a variável `$PATH`. Vamos ver o que acontece quando digitamos `echo $PATH`:

```
> echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
```

Assim, a variável `$PATH` indica para o *shell* onde ele deveria buscar os comandos e da mesma maneira esta variável `$PATH` ajuda a buscar comandos, existe também a variável `$MANPATH`. Essa variável diz para o `man`, onde ele deve procurar os manuais.

Se digitarmos `echo $MANPATH` teremos uma resposta vazia. Como ele acha essa variável vazia?

Se a variável é vazia ele procura um arquivo de configuração e este arquivo está em `cat /etc/manpath.config`, após digitar isso no Terminal vamos dar um "Enter" e analisar o que temos:

Temos essa janela que está cheia de configurações do `man`. Como essa tela está muito cheia de dados vamos dar um `clear` e vamos digitar `less /etc/manpath.config` para ficar mais legível.

Agora, conseguimos ler algo. É explicado, por exemplo, que o cerquilha é um comentário e que, para adicionarmos um `PATH` devemos escrever `MANDATORY_MANPATH` e etc. E temos outras diversas instruções. Com esses comandos temos indicações de onde procurar as coisas:

Não precisamos saber em detalhes como funciona o arquivo de configuração, apenas precisamos saber que ele existe. O `man`, por padrão, lê o arquivo para saber onde deve procurar os manuais e depois usa a variável. Como ela está vazia ele ignora ela e procura apenas no local onde deve fazer a busca. Mas, é estranho ter que ficar olhando a tela de configuração toda a vez, portanto, existe um programa de ajuda chamado `manpath` que nos mostra, de forma resumida, onde ele deve procurar os manuais:

```
> manpath
/usr/local/man:/usr/local/share/man:/usr/share/man
```

Quando escrevemos `manzip` ele procurará nesses diretórios, encontrará e mostrará para nós a informação.

Vamos usar a variável `$MANPATH` e vamos escolher uma variável qualquer, por exemplo, `/home/guilherme`. Teremos `MANPATH=/home/guilherme` e com isso estamos passando que ele deve buscar no `/home/guilherme`. Assim, se digitarmos `manpath` teremos o seguinte:

```
> manpath
/usr/local/man:/usr/local/share/man:/usr/share/man
```

Ou seja, sem grandes diferenças. E por que? Porque essa é uma variável que não é de ambiente. Então, vamos dar um `export MANPATH` e agora é uma variável de ambiente. Vamos escrever `manpath` e ver o que acontece:

```
> export MANPATH
> manpath
```

```
manpath: warning: $MANPATH set, ignoring /etc/manpath.config/home/guilherme
```

Então, temos que a variável já esta setada, vamos usar ela e ignoraremos o `config`. Repare que, atualmente, ao usar a variável começamos a correr um risco pois estamos ignorando o arquivo de configuração.

O que podemos tentar fazer agora?

Podemos tentar acessar o `man zip`. E em qual diretório será que ele vai estar? No `/home/guilherme`. E será que ele acha algo? Vamos digitar isso e ver o que acontece:

```
> man zip
No manual entry for zip
See `man 7 undocumented`for help when manual pages are not available
```

Não encontrou documentação e isso ocorre, pois ele está buscando apenas no `/home/guilherme`.

O que já vimos?

Vimos que podemos utilizar o comando `manpath` e que ele nos informa onde buscar as informações. Esse comando dá preferência pela variável `MANPATH` com letras maiúsculas que é uma variável de ambiente, senão, ele lê o arquivo de `/etc/manpath.config`, que é o arquivo de configuração, e isso não precisamos entender em detalhes. Então, temos esse arquivo e conseguimos encontrar o manual.

Vamos fechar a tela que estivemos usando até o momento e vamos buscar o manual em um local padrão. Fechamos e abrimos um novo Terminal. Agora, já que sabemos como ele busca coisas, por exemplo, usando o `whereis` e o nome de outro comando, vamos, como ilustração usar `whereis + zip`, isto é, `whereis zip`. Agora, já sabemos como ele encontra as coisas.

O que vamos fazer?

Vamo criar um diretório temporário e entrar nele e vamos copiar o arquivo e para fazer isso vamos usar o comando `cp` e o local do arquivo `/usr/share/man/man1/zip.1.gz`:

```
> mkdir temp
> cd temp
~/temp$ cp /usr/share/man/man1/zip.1.gz .
zip.1.gz
```

Agora, no diretório atual temos uma cópia do arquivo. E vamos descompactar o arquivo, se para compactar é usado o `gzip` vamos usar o `gunzip` para descompactar. Então, digitamos `gunzip zip.1.gz`.

```
~/temp$ gunzip zip.1.gz
~/temp$ ls
zip.1
```

E para visualizar esse arquivo? Como fazemos? Digitamos `cat zip.1` e teremos o manual:

Lembrando que podemos usar o `less zip.1` para ler. Vamos usar o `less` :

Esse arquivo está todo formatado, ele possui um formato para podermos gerar uma página do manual, isto é, ele é cheio de barras invertidas, pontos, comandos e etc. Ou seja, tem todo um formato para que se possa gerar uma página no manual.

Esse formato é bem definido e é convertido naquela `man` que podemos ver quando digitamos `man zip` . Aquele arquivo de texto é lido, interpretado e transformado nisso:

Existe um formato para isso e existe um interpretador que é o `groff` , que interpreta o arquivo fonte, que é esse arquivo no formato zip.1 que nós temos. Então, o `groff` transforma aquele formato de arquivo em nosso manual. Ele pode transformar o arquivo em diversos formatos, antigamente, era extremamente útil transformar, por exemplo, em `pdf` para efetuar a impressão do conteúdo. Atualmente, não queremos os arquivos impressos, é preferível ler no terminal ou exportar para o *HTML* para visualizar. Então, na prática, provavelmente, não utilizaremos o `groff` na linha de comando, pois, não iremos criar comandos novos. Na prática, a maior parte dos usuários do *Linux*, que fazem o *Linux essentials* e que já possuem uma, não precisam, necessariamente, criar arquivos de ajuda, portanto, não usam tanto o `groff` .

O que precisamos é conhecer como o `man` funciona e é por isso que estamos estudando esse comando. O `man` usa o `manpath` para saber onde estão os manuais e a variável `$MANPATH` para sobrescrever isso, o arquivo de configuração, que é o arquivo que porta a configuração, usa o `groff` , o `gunzip` para descompactar na memória e para converter no formato que vemos em nossa tela, bonitinho. Além disso o `man` usa um pouco mais, lembra-se quando utilizamos o `man -k ascii` ? Quando digitamos isso queríamos tudo que tivesse `ascii` , bom, não é necessariamente o `man` que faz essa busca, mas é um outro programa o `apropos` . O `apropos` é quem faz a procura, então, o resultado é mais ou menos o mesmo quando fazemos o `man -k` .

Vamos digitar `apropos` , teremos o seguinte:

É a mesma coisa que o `manpath` e o `groff` , isto é, o `man` é composto de diversos programas que utiliza para ler o arquivo de configuração e interpretar a variável de ambiente, descompactar e mostrar a informação na tela, para buscar o *k word* e etc. Para tudo isso ele utiliza pequenas outras extensões, programas, ou chamadas de bibliotecas que fazem o trabalho para ele.

Então, só de utilizar o `man` vimos que ele usa diversas outras coisas, o `apropos` para buscar por *k words* , o `groff` para converter o formato original do arquivo para a nossa tela, assim como também pode ser utilizado para converter outros formatos, o `less` que ele usa para navegar no arquivo, o `manpath` que é um programa e também uma variável (quando em maiúsculo) de ambiente que utilizamos para definir o `path` onde ele vai procurar as coisas, o `/etc/manpath.config` que possui as configurações do `man` em geral. Ou seja, as diversas configurações do `manpath` . Então, como podemos perceber, ele utiliza diversas coisas por trás.

E, ainda, podemos utilizar o `whatis ascii` , ou seja, "o que é o ascii?" e ele fala para nós quem ele é:

```
~/temp$ whatis ascii
ascii (7)       - ASCII character set encoded in octal, dec...
```

Da mesma maneira que usamos o `whatis` para o `ascii` poderíamos usar para o `zip` :

```
~/temp$ whatis ascii
zip (1)       - package and compress (archive) files
```

Ele busca pelo nome, mas ele traz só a primeira informação, só para sabermos "quem é" aquilo que estamos buscando. Por isso escrevemos apenas `whatis` e o nome daquilo que procuramos. Ele procura nos manuais e traz para nós apenas o título, o capítulo a que pertence, a informação inicial dele e é isso. E se buscarmos algo que não existe ele não trará informações a respeito pois não as encontra. Por exemplo, isso acontecerá se buscarmos o `whatis help`, ele nos contestará `help: nothing appropriate.` Pois, ele não acaha o manual. Além disso o `whatis` trabalha com o `man` em diversas partes.

Então, repare a quantidade enorme de programas que utilizamos junto com o `man` no dia a dia, se somos apenas um usuário final acabamos usando ele junto com outras coisas, mas para a prova é bom sabermos muito mais. Já vimos o `man`, as `man pages` e veremos mais adiante outras coisas.