

Obtendo ajuda com o help

Bem-vindo!

Continuando nossos estudos para a prova do *Linux Essential* do *Linux Professional Institut* veremos agora como buscar ajuda na linha de comando. Na verdade essa seção acaba falando bastante sobre ajuda, ou seja, sobre o *help*. Entretanto, também abordaremos o *Man* e o *Info*. Veremos como o *Man* funciona por trás e passaremos por algumas das ferramentas que ele utiliza, como o *manpath* e etc. Veremos, ainda, mais informações a cerca do *info*. Estudaremos como funciona o *usr/share/doc* que armazena documentação relativa ao que está instalado e que lida com documentação em geral, as quais temos interesse de colocar em nosso sistema operacional. Iremos falar também sobre o *locate*, que não está, necessariamente, relacionado com "ajuda", mas que permite que localizemos arquivos em geral em nosso computador. Por fim, veremos também uma parte básica do *locate* e do *update db*.

Help

Vamos começar pegando o ajuda, se queremos fazer isso digitaremos e utilizaremos a palavra "help". Já passamos por esse comando antes, ele apareceu em outros momentos. Então, se digitarmos `help` e dermos um "Enter" teremos o seguinte:

```
compgen [-abdefghjklsuv] [-o op> return [n]
complete [-abdefghjklsuv] [-pr]> select NAME [in WORDS ... ;] >
compopt [-o|+o option] [-DE] [> set [-abefghkmnpstuvxBCHP] [-o >
continue [n] shift [n]
coproc [NAME] command [redirec> shopt [-pqsu] [-o] [optname .>
declare [-aAfFgIlNrtux] [-p] [> source filename [arguments]
dirs [-clpv] [+N] [-N] suspend [-f]
disown [-h] [-ar] [jobspec ...> test [expr]
echo [-neE] [arg ...] time [-p] pipeline
enable [-a] [-dnps] [-f file na> times
eval [arg ...] trap [-lp] [[arg] signal_spec>
exec [-cl] [-a name] [command > true
exit [n] type [-afptP] name [name ...]>
export [-fn] [name[=value] ...> typeset [-aAfFgIlrtux] [-p] n>
false ulimit [-SHabdefilmnpqrstuvx>
fc [-e ename] [-lnr] [first] [> umask [-p] [-S] [mode]
fg [job_spec] unalias [-a] name [name ...]
for NAME [in WORDS ... ] ; do > unset [-f] [-v] [-n] [name ..>
for (( exp1; exp2; exp3 )); do> until COMMANDS; do COMMANDS; >
function name { COMMANDS ; } o> variables - Names and meaning>
getopts optstring name [arg] wait [-n] [id ...]
hash [-lr] [-p pathname] [-dt]> while COMMANDS; do COMMANDS; >
help [-dms] [pattern ...] { COMMANDS ; }
```

As informações nos fornecem as diversas maneiras de como podemos utilizar o `help`. Por padrão digitamos o `help` e mais um nome: "help nome". Podemos ver que, justamente, essa observação está escrita na tela que aparece:

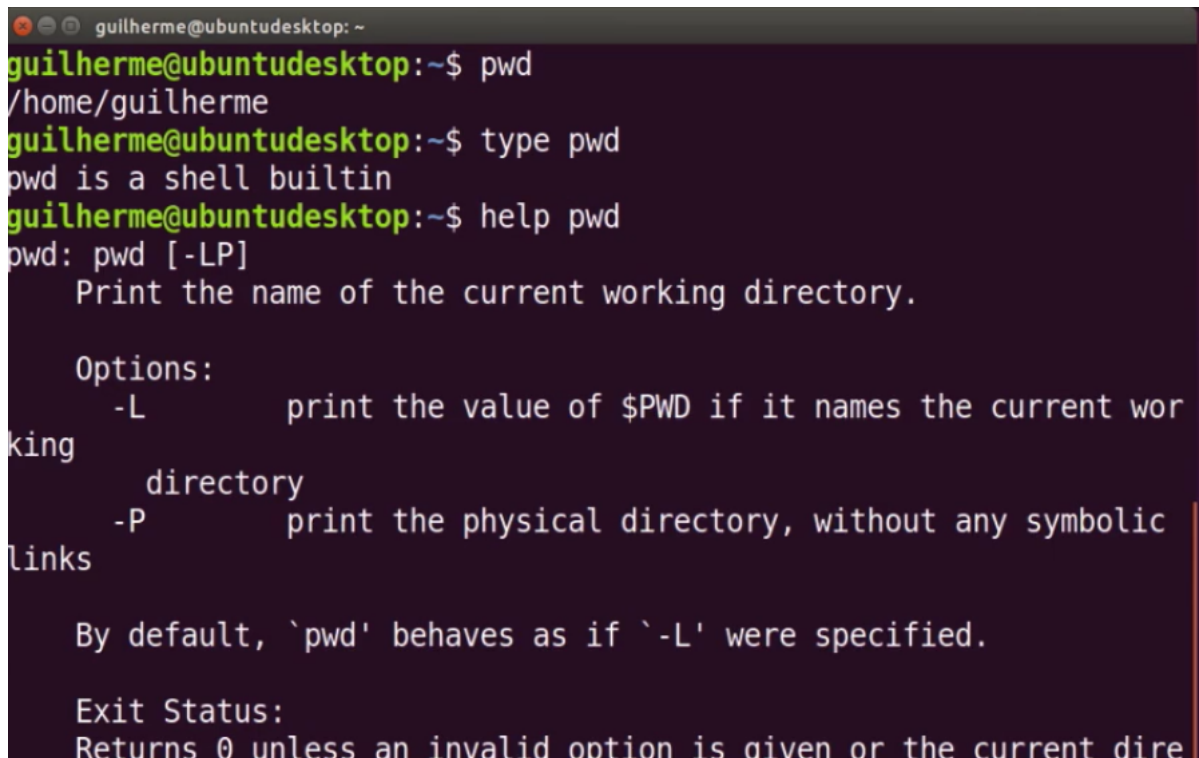
"Type `help name` to find out more about the function `name`."

O `help` nos ajuda referente as funções de coisas definidas em cima do *shell*, isto é nos *builtin* do *shell*. Então, se quisermos podemos pegar um *builtin* do *shell*, por exemplo, o `pwd`. Para conferirmos que o `pwd` é, justamente, um *builtin*

do *shell*, podemos digitar `type pwd` e ele nos contestará confirmando que é:

```
> type pwd
pwd is a shell builtin
```

Como podemos observar, ele é um *builtin* do *shell*, então, podemos digitar `help pwd` que nos será mostradp um pouco da sua ajuda para nós:



```
guilherme@ubuntudesktop: ~
guilherme@ubuntudesktop:~$ pwd
/home/guilherme
guilherme@ubuntudesktop:~$ type pwd
pwd is a shell builtin
guilherme@ubuntudesktop:~$ help pwd
pwd: pwd [-LP]
    Print the name of the current working directory.

    Options:
      -L          print the value of $PWD if it names the current wor
king
                  directory
      -P          print the physical directory, without any symbolic
links

    By default, `pwd' behaves as if `-L' were specified.

    Exit Status:
    Returns 0 unless an invalid option is given or the current dire
```

E se pegarmos o `type cd`? Ele irá nos informar que é um *shell builtin* e podemos, então, digitar `help cd` e teremos também informações a cerca do `cd`. Da mesma maneira, se pegarmos o `ls` e digitarmos `type ls` ele nos informará que é um *aliased*:

```
> type ls
ls is aliased to `ls --color=auto`
```

Vamos buscar saber quem ele é, realmente, digitando `type -a ls` e teremos todas as suas variações:

```
>type -a ls
ls is aliased to `ls --color=auto`
ls is /bin/ls
```

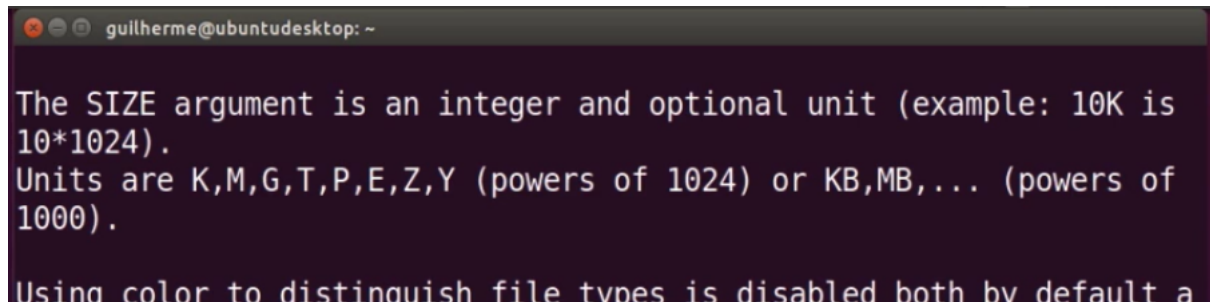
Veremos que esse comando está em `/bin/ls`. Será que ele possui um *help*? Digitando `help ls` veremos que não:

```
> help ls
bash: help: no help topics match `ls`. try `help help` or `man -k ls` or `info` ls.
```

O `ls` é um comando externo, ele é um programa externo, por isso, o `help` não tem essa informação para nós. O `help` só traz informações sobre os *builtins* do *shell*.

Agora, se tivermos outras situações em que queremos saber a ajuda de outros comandos?

Existem outras maneiras que costumam aparecer e que utilizamos bastante. Uma maneira é escrever `ls -- help`, essa opção nos traz a ajuda desse comando. Um programa como o `ls` tem suporte ao `--help`:

A terminal window with a dark background and light text. The title bar shows 'guilherme@ubuntu desktop: ~'. The text in the terminal is the help output for the 'ls' command, explaining the 'SIZE' argument and units, and mentioning that color is disabled by default.

```
guilherme@ubuntu desktop: ~  
The SIZE argument is an integer and optional unit (example: 10K is  
10*1024).  
Units are K,M,G,T,P,E,Z,Y (powers of 1024) or KB,MB,... (powers of  
1000).  
Using color to distinguish file types is disabled both by default a
```

É um *shell builtin*? Por exemplo, o `pwd`, será que ele possui um `--help`?

```
> pwd --help  
bash: pwd: --: invalid option  
pwd: usage: pwd [-LP]
```

Ele não possui! Ele fala que o `--help` não é uma opção válida, se olharmos o `pwd` possui um `-L` e um `-P` maiúsculo. Um outro comando como o `cd` `--help` também vai nos informar que não é uma opção válida:

```
> cd --help  
bash: cd: --: invalid option  
cd: usage: cd [-L][-P [-e]] [-@]] [dir]
```

Isso significa que todos os *shell builtin* não possuem `--help`?

Não necessariamente, pode ser que hoje não e amanhã eles passem a ter. O *shell* não tem a obrigação de não suportar o `--help`, só que ele pode suportar a ajuda. Por padrão ele utiliza o `help`. O `help` e o *builtin* podem ser usados juntos.

Acabamos de ver uma outra maneira de pegar `help` com comandos, utilizando o `--help`, assim, podemos usar `ls --help`, `zip --help` e ele trará o `help` desses comandos para nós. Existem outras maneiras de trazer o `help` desses comandos. Em geral `--help` é um comando muito grande, então, as pessoas normalmente digitam `-h`. Digitaremos `ls -h`:

```
> ls -h
arquivos.zip          falha                loja                 Public              zip
Desktop               ~falha              mostra_idade        sucesso
Documents             help                mostra_idade~       sucesso~
Downloads             log completo.txt    Music               Templates
examples.desktop      log completo.txt~   Pictures            Videos
```

Mas, repare que no `ls -h` não funcionou como ajuda. Qual outra opção que poderíamos utilizar? Poderíamos, por exemplo, usar o `-?`. Se digitarmos `ls -?` teremos o seguinte:

```
> ls -?
ls: invalid option -- `?'
Try `ls --help' for more information
```

Também não serviu como ajuda. As vezes, esses dois, são suportados como atalhos para `--help`. Mas por mais que o `--help` exista no `ls --help`. As vezes, algumas aplicações e programas vão ter uma outra opção como um atalho que é o `-h` ou o `-?`. Tudo isso vai depender de cada programa, pois, não é um padrão obrigatório. Por exemplo, o `pwd` que é um *shell builtin* não suporta o `--help`, `-?` e `-h`.

Um programa é obrigado a suportar um `--help`?

Vamos testar digitando o `ls --help` e vemos que ele nos traz a ajuda.

Ele está suportando, não é que é obrigatório que os programas reajam com o `--help`, mas é bem provável que sim. Então, será que o `ls` reage com algum atalho ou abreviação? Vamos digitar `ls -h`:

```
> ls -h
arquivos.zip          falha                loja                 Public              zip
Desktop               ~falha              mostra_idade        sucesso
Documents             help                mostra_idade~       sucesso~
Downloads             log completo.txt    Music               Templates
examples.desktop      log completo.txt~   Pictures            Videos
```

Ele não traz a ajuda, traz alguma outra coisa. Um outro atalho que aparece também é o `-?` e ele nos informa para usar o `ls --help`:

```
> ls -?
ls: invalid option -- `?'
Try `ls --help' for more information
```

Cada programa possui um padrão diferente nessa questão da ajuda. Vamos observar o `zip`. Se digitarmos `zip --help`, `zip -h` ou `zip -?` ele nos trará a ajuda, inclusive, se digitarmos `zip -qualquer coisa` ele também nos mostra a ajuda. O `zip` por padrão já traz a ajuda para nós e nos informa que se podemos digitar `-h2` para pedir mais ajuda. Então, podemos digitar também `zip -h2`:

```

guilherme@ubuntudesktop: ~
ive in
g to a
ps
ce)
-MM
files
hive
e files
ated)
dir, which allows using seekable temp file when writin
write once CD, such archives compatible with more unzi
(could require additional file copy if on another devi
input patterns must match at least one file and matched
must be readable or exit with OPEN error and abort arc
(without -MM, both are warnings only, and if unreadabl
are skipped OPEN error (18) returned after archive cre

```

Repare que cada programa tem um jeito distinto para pegar ajuda, o importante é saber que se for um *shell builtin*, poderemos utilizar o `help`. Se for um programa podemos primeiro tentar com `--help`, `-h` ou `-?`. Essas são as variações que costumam trazer para nós um pouco de ajuda sobre aquele comando. Nenhum programa externo do *shell* tem que suportar, necessariamente, qualquer um dos comandos. Todos são opções, alguns vão suportar e outros não, mas, já é alguma ajuda.

Isso é uma ajuda no terminal, uma ajuda que podemos ter no terminal que é, exatamente, o que a prova cobra para nós. A prova quer que saibamos como rodar comandos de ajuda na linha do terminal e podemos fazer isso utilizando algumas das opções que citamos anteriormente. Em geral, o programa reage a alguma delas, e de verdade mesmo, o `--help` é a mais utilizável, mas as outras são úteis também.

