

Aula 103.1 – Execução de Comandos Sequenciais, history, man

O que veremos nesta aula:

Sequência de Comandos	2
O separador “;” (Ponto e virgula)	2
O comando && (E comercial)	2
O comando (Pipe)	3
O comando !! (Exclamacao)	3
Repetição de comandos	4
O comando history	4
Como limpar o arquivo de Histórico	6
Pesquisando comandos digitados	7
O Auto Completar (com TAB)	8
Comandos de ajuda	9
O comando man	9
O comando info	11
O comando whatis e apropos	11

Dando Continuidade à aula anterior, vamos ver mais algumas propriedades do bash e alguns outros comandos.

Sequência de Comandos

Habitualmente no Linux alguns usuários geralmente digitam comando por comando, porém é possível fazer com que o sistema execute os comandos sequencialmente, e há várias formas de fazer isso, vejamos:

O separador “;” (Ponto e virgula)

A primeira forma que é explicada na aula é o separador “;”, através deles podemos executar um comando atrás do outro, ele tem uma particularidade que é, independente se o comando está certo ou errado ele é executado.

```
$ clear ; date ; ls
```

Sendo assim, ele executa um comando por vez não importando se o comando anterior, no caso clear esteja certo ou não, ele executará em os demais comandos.

O comando && (E comercial)

Diferente do “;” o && acusa erro e para a execução se algum dos comandos estiverem errados.

Por exemplo: Se o primeiro comando for executado com sucesso ele parte para o segundo comando, caso contrário ele para a execução acusando o erro.

```
$ ls tmp/teste && echo Linux
```

Note que **primeiro** executamos o ls apontando o arquivo /teste dentro do diretório /tmp e em seguida ele deve executar o comando echo, caso não exista o arquivo dentro do diretório /tmp ele acusa erro e para a execução.

```
kali@kali:~$ ls /tmp/teste && echo Linux
ls: cannot access '/tmp/teste': No such file or directory
kali@kali:~$
```

IMPORTANTE: O comando && entende que:

Faça isso **E** isso= **faça ls && echo**

Quando o arquivo existe ele apenas executa normalmente.

```
kali@kali:~$ touch /tmp/teste
kali@kali:~$ ls /tmp/teste && echo Linux
/tmp/teste
Linux
kali@kali:~$
```

O comando || (Pipe)

O separador pipe faz o inverso do &&, ele executa o segundo comando caso o primeiro comando falhe. Vejam:

```
kali@kali:~$ ls /tmp/curso_lpi || echo Linux
ls: cannot access '/tmp/curso_lpi': No such file or directory
Linux
kali@kali:~$
```

Se o primeiro comando der certo ele ignora a regra do pipe e apenas executa o primeiro comando existente.

IMPORTANTE: O comando || entende:

Faça isso **OU** isso= **faça ls || echo**

```
kali@kali:~$ ls /tmp/curso_lpi || echo Linux
/tmp/curso_lpi
kali@kali:~$
```

O comando !! (Exclamacao)

O comando !! repete o último comando executado no bash.

```
$ !!
```

```
kali@kali:~$ date
Sun 29 Mar 2020 03:21:52 PM EDT
kali@kali:~$ !!
date
Sun 29 Mar 2020 03:21:56 PM EDT
kali@kali:~$
```

Repetição de comandos

O comando history

O comando history lista os todos os últimos comandos digitados no bash, cada usuário tem o seu arquivo de histórico.

```
kali@kali:~$ history
 1 firefox
 2 echo $BASH
 3 sudo su
 4 pwd
 5 LS
 6 ls
 7 cd Downloads/
 8 ls -l
 9 mv original.tgz ..
10 cd ..
11 ls
12 tar zxvf original.tgz
13 ls
14 cd Exercicios/
15 ls
16 cd ..
17 ls
18 mkdir LPIC-1
19 cd LPIC-1/
20 LS
21 ls
22 cd ../Downloads/
23 ls
24 mv original.pdf ../LPIC-1/
25 cd ../LPIC-1/
26 ls
27 exit
28 sudo su
29 clear
30 clear ; date ; ls
31 ls /tmp/teste && echo Linux
32 touch /tmp/teste
33 ls /tmp/teste && echo Linux
34 rm /tmp/teste
35 ls /tmp/curso_lpi || echo Linux
36 touch /tmp/curso_lpi
37 ls /tmp/curso_lpi || echo Linux
38 clear ; date ; ls
39 ls /tmp/teste && echo Linux
40 ls /tmp/curso_lpi || echo Linux
41 clear
42 history
43 clear
44 date
45 history
kali@kali:~$
```

Como vemos na imagem acima, o comando lista os últimos comando digitados, e na antes de cada comando há uma número.

Outra forma que podemos executar novamente o comando é da seguinte forma:

```
$ !44
```

Sendo assim ele repetirá o comando 44, da lista acima.

```
44 date
45 history
46 ls
47 clear
48 history
49 clear
50 history
kali@kali:~$ !44
date
Sun 29 Mar 2020 03:29:53 PM EDT
kali@kali:~$
```

Uma outra forma ainda, é o “!” seguido da string, por exemplo:

```
$ !uname
```

Dessa forma ele buscará o comando no history e o executará da forma que foi executado pela última vez.

```
kali@kali:~$ !uname
uname
Linux
kali@kali:~$
```

Outro exemplo é

```
$ !ls
```

```
kali@kali:~$ !ls
ls /tmp/curso_lpi
/tmp/curso_lpi
kali@kali:~$
```

Como limpar o arquivo de Histórico

Para limparmos o arquivo de histórico do nosso usuário executamos o comando abaixo:

```
$ history -c
```

```
kali@kali:~$ history -c
kali@kali:~$ history
  1 history
kali@kali:~$
```

Podemos verificar onde estão armazenados os comandos digitados usando o comando set e filtrando pela variável de ambiente HISTFILE.

```
kali@kali:~$ set | grep HISTFILE
HISTFILE=/home/kali/.bash_history
HISTFILESIZE=2000
kali@kali:~$
```

Os comandos digitados ficam armazenados no arquivo **.bash_history**, este arquivo é encontrado dentro da pasta do usuário em /home/lpi1 como no exemplo:

```
kali@kali:~$ cat .bash_history
firefox
echo $BASH
sudo su
pwd
ls
ls
cd Downloads/
ls -l
mv original.tgz ..
cd ..
ls
tar zxvf original.tgz
ls
cd Exercicios/
ls
cd ..
ls
mkdir LPIC-1
cd LPIC-1/
ls
ls
cd ../Downloads/
ls
mv original.pdf ../LPIC-1/
cd ../LPIC-1/
ls
exit
kali@kali:~$
```

Mas, não havíamos zerado o arquivo history?

Quando fazemos o **login**, o que está no `.bash_history` é carregado em memória, e o comando `history` funciona com o que está na memória, quando fazemos o **logout** é feito um `append` do que há de novo na memória para o arquivo `.bash_history`.

Ao usarmos o `history -c`, os comandos do `history` em memória é limpo, e quando você faz o `logout`, é feito um `append` de nada no `.bash_history`, ou seja, ele não é alterado. Para limpar definitivamente há 2 opções, executar os comandos abaixo, forçando que o `history` em memória, em branco, seja refletido no `.bash_history`:

```
$ history -c && history -w
```

Ou simplesmente limpar manualmente o arquivo:

```
$ history -c
```

```
$ cat /dev/null > ~/.bash_history
```

```
kali@kali:~$ cat /dev/null > ~/.bash_history
kali@kali:~$ cat .bash_history
kali@kali:~$
```

Pesquisando comandos digitados

No `bash` ainda temos a possibilidade de buscar comandos que já digitamos anteriormente, para abrir a caixa de pesquisa pressionamos o **ctrl+R**. Este procura os comandos dentro do seu histórico de comandos.

Conforme digitamos o comando desejado ele já mostra os comandos iniciados com aquela string. Encontrando o comando, basta teclar o `enter`.

```
kali@kali:~$ cat .bash_history
kali@kali:~$
(reverse-i-search)`cat': cat .bash_history
```

O Auto Completar (com TAB)

Essa função se dá quando digitamos um comando, seja ele para arquivo ou diretórios, ele completa o comando quando pressionamos a tecla tab. O auto completar tem suas particularidades.

Exemplo para verificar todos os comandos que iniciam com “ls”:

\$ ls + TAB

```
kali@kali:~$ ls
lsattr  lscpu  lslogins  lsof  lstopo-no-graphics
lsblk  lsinitramfs  lsmem  lspci  lsusb
lsb_release  lsipc  lsmod  lspgpot
lslocks  lsns  lstopo
kali@kali:~$ ls
```

Exemplo para verificar todos os arquivos que comecam com “Script” no diretorio Exercicios:

\$./S + TAB -> Completa para ./Script_

\$./Script_ + TAB

```
kali@kali:~/Exercicios$ ./Script_
Script_Exemplo.sh  Script_Variavel.sh
kali@kali:~/Exercicios$ ./Script_
```


Comandos de ajuda

Um dos recursos de extrema importancia do Shell sao os comandos para obter ajuda. Entre eles temos o man, info, whatis e o apropos. Veremos mais sobre eles abaixo.

O comando man

O comando **man** e o principal comando de ajuda. Ele mostra o manual de ajuda do comando, basicamente todos os comandos tem o seu manual de referênciã, e eles são acessados pelo man exemplo:

```
$ man ls
```



The screenshot shows a terminal window with the title 'kali@kali: ~'. The terminal displays the man page for the 'ls' command. The page is titled 'LS(1)' and 'User Commands'. It includes sections for NAME, SYNOPSIS, and DESCRIPTION. The DESCRIPTION section explains that 'ls' lists information about files and directories, sorted alphabetically by default. It also lists various options and their functions, such as '-a' for listing all files, '-l' for long format, and '-C' for columnar output. A yellow text overlay on the right side of the terminal reads: 'Pressione "q" para sair e /String para pesquisar'. At the bottom of the terminal, a status bar indicates 'Manual page ls(1) line 1 (press h for help or q to quit)'.

```
kali@kali: ~
File Actions Edit View Help
LS(1) User Commands LS(1)
NAME
  ls - list directory contents
SYNOPSIS
  ls [OPTION]... [FILE]...
DESCRIPTION
  List information about the FILES (the current directory by default). Sort entries alpha-
  betically if none of -cftuvSUX nor --sort is specified.

  Mandatory arguments to long options are mandatory for short options too.

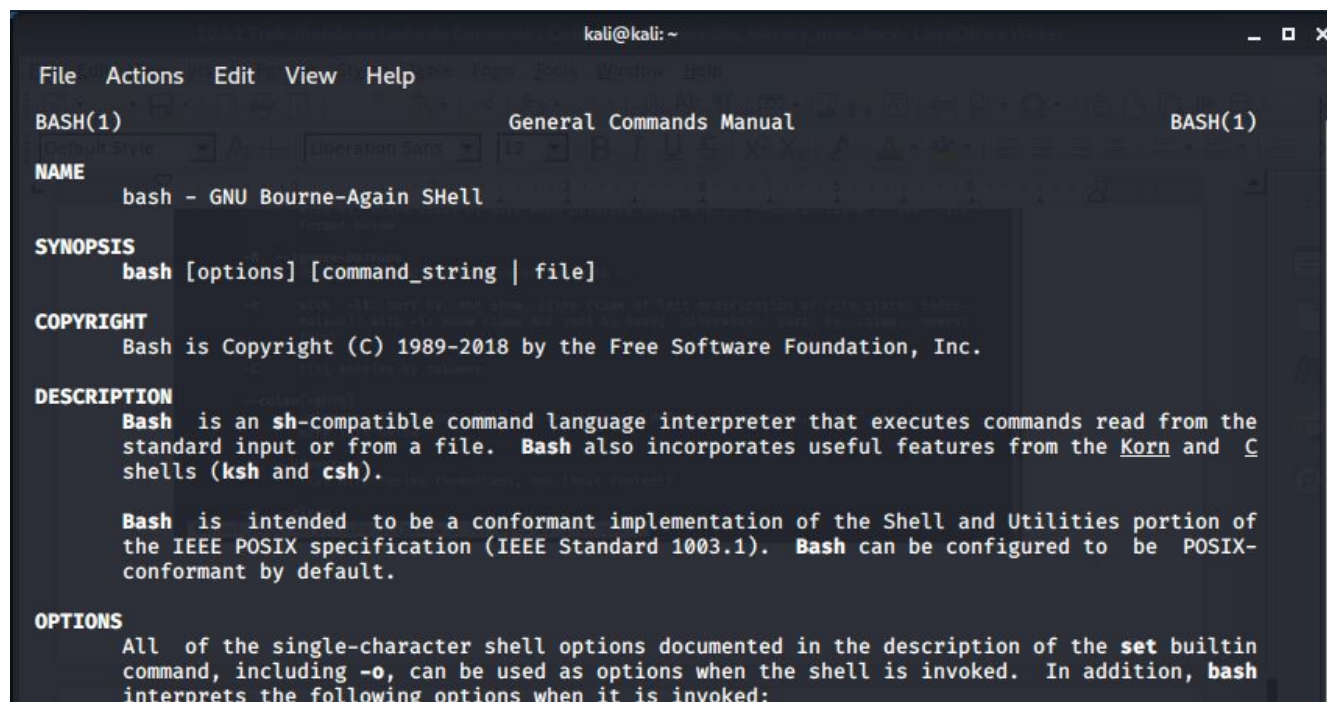
  -a, --all
      do not ignore entries starting with .
  -A, --almost-all
      do not list implied . and ..
  --author
      with -l, print the author of each file
  -b, --escape
      print C-style escapes for nongraphic characters
  --block-size=SIZE
      with -l, scale sizes by SIZE when printing them; e.g., '--block-size=M'; see SIZE
      format below
  -B, --ignore-backups
      do not list implied entries ending with ~
  -c
      with -lt: sort by, and show, ctime (time of last modification of file status infor-
      mation); with -l: show ctime and sort by name; otherwise: sort by ctime, newest
      first
  -C
      list entries by columns
  --color[=WHEN]
      colorize the output; WHEN can be 'always' (default if omitted), 'auto', or 'never';
      more info below
  -d, --directory
      list directories themselves, not their contents
  -D, --dired
      ...

Manual page ls(1) line 1 (press h for help or q to quit)
```

Após o comando ser executado é aberto o manual de referência completo do comando ls.

Uma observação é que, quando o comando é interno, ou seja, faz parte do bash ele não possui o man, sendo assim temos que consultar o manual do bash:

```
$ man bash
```

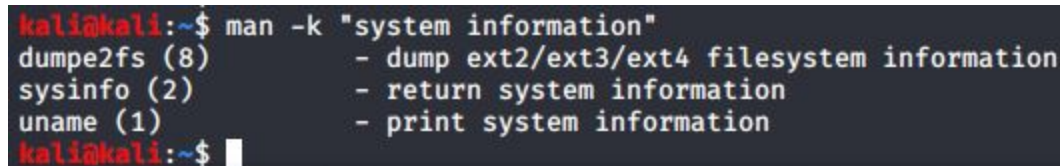
A screenshot of a terminal window showing the man page for the 'bash' command. The window title is 'kali@kali: ~'. The terminal has a menu bar with 'File', 'Actions', 'Edit', 'View', and 'Help'. The man page content is displayed in a light blue font on a dark background. It includes sections for NAME, SYNOPSIS, COPYRIGHT, DESCRIPTION, and OPTIONS. The DESCRIPTION section states that Bash is an sh-compatible command language interpreter. The OPTIONS section mentions that all single-character shell options documented in the set builtin command can be used as options when the shell is invoked.

```
kali@kali: ~
File Actions Edit View Help
BASH(1) General Commands Manual BASH(1)
NAME
  bash - GNU Bourne-Again SHell
SYNOPSIS
  bash [options] [command_string | file]
COPYRIGHT
  Bash is Copyright (C) 1989-2018 by the Free Software Foundation, Inc.
DESCRIPTION
  Bash is an sh-compatible command language interpreter that executes commands read from the
  standard input or from a file. Bash also incorporates useful features from the Korn and C
  shells (ksh and csh).
  Bash is intended to be a conformant implementation of the Shell and Utilities portion of
  the IEEE POSIX specification (IEEE Standard 1003.1). Bash can be configured to be POSIX-
  conformant by default.
OPTIONS
  All of the single-character shell options documented in the description of the set builtin
  command, including -o, can be used as options when the shell is invoked. In addition, bash
  interprets the following options when it is invoked:
```

Outra forma de usar o man é com o parâmetro **-k**:

```
$ man -k "system information"
```

Dessa forma o comando traz qualquer referência que contenha o conteúdo "system information" e o comando do conteúdo informado. Veja:

A screenshot of a terminal window showing the output of the command 'man -k "system information"'. The terminal shows three results: 'dumpe2fs (8)' for dumping ext2/ext3/ext4 filesystem information, 'sysinfo (2)' for returning system information, and 'uname (1)' for printing system information. The prompt is 'kali@kali:~\$'.

```
kali@kali:~$ man -k "system information"
dumpe2fs (8)      - dump ext2/ext3/ext4 filesystem information
sysinfo (2)      - return system information
uname (1)        - print system information
kali@kali:~$
```

Novamente:

```
$ man -k "updatedb"
```

```
kali@kali:~$ man -k "updatedb"
updatedb.conf (5)      - a configuration file for updatedb(8)
updatedb (8)          - update a database for mlocate
updatedb.mlocate (8)  - update a database for mlocate
kali@kali:~$
```

O comando info

Um pouco diferente do **man** é o comando **info**, ele basicamente é um man de forma reduzida. Tanto o **man** quanto o **info** são comandos para buscar ajuda sobre determinados comandos:

```
$ info ls
```

O comando whatis e apropos

O comando **whatis** é baseado no comando. Ele te informa a descrição do comando que desejar:

```
$ whatis tcpdump
```

```
kali@kali:~$ whatis tcpdump
tcpdump (8)          - dump traffic on a network
kali@kali:~$
```

Já o comando **apropos** faz a busca baseado na descrição e traz o comando referente a ela, assim como o **man -k**:

```
$ apropos "attack"
```

```
kali@kali:~$ apropos "attack"
airbase-ng (8)      - multi-purpose tool aimed at attacking clients as opposed to the Access Poin...
ettercap (8)        - multipurpose sniffer/content filter for man in the middle attacks
kali@kali:~$
```