

Linux Shell

Jan Pribošek

7.4.2017

I. Basic commands

MAN

Manual pages - built in documentation

```
1  # usage
2  man [command]
3
4  man man
5  man ls
```

LS

List directory contents

```
1  # usage
2  ls -[options]
3
4  ls
5  ls -a      # display hidden files
6  ls -la     # display hidden files in long format
7  ls -la -r  # display all in reverse order
8  ls -la -h  # display file sizes in human readable format
```

- hidden files: dotfiles [.filename]
- file extensions: no file extensions, everything in Linux is file
- naming files: use [file_name], [fileName], [file-name] instead of spaces

PWD

Print working directory

```
1  pwd
```

SHEBANG LINE

Shebang line is used to tell the system the name of the interpreter that should be used to execute the script

```
1  #!/bin/bash
2  #!/usr/bin/python3
```

CD

Change directory

```
1  # usage
2  cd [path]
3
4  cd          # change to home folder
5  cd -        # change to previous working directory
6  cd ..       # change to parent folder
```

```
1  # Tips
2
3  /var/log    # absolute path - begins with root directory
4  var/log     # relative path - begins with current working directory
5
6  .           # working directory
7  ..          # working directory parent directory
8
9  (TAB)        # autocomplete
10 (TAB,TAB)    # show all possibilities
11 (Select,middle mouse button) # copy & paste
```

CAT

Concatenate files and print on standard input

```
1  # usage
2  cat [file1] [file2] [file3] ...
3
4  # print from bottom to top
5  tac [file1] [file2] [file3] ...
```

TAIL

Display last n lines

```
1  # usage
2  tail -n 5 [file]    # display last 5 lines of file
```

HEAD

Display first n lines

```
1  # usage
2  head -n 3 [file]    # display first 3 lines of file
```

II. Manipulating files and directories

TOUCH

Create new empty file

```
1 touch [filename]
```

MKDIR

Create new empty folder

```
1 mkdir [directory]
2 mkdir [dir2] [dir2] [dir3]
```

CP

Copy files

```
1 # usage
2 cp [/path/to/input] [/path/to/output]
3
4 cp [file1] [file2] [dir]
5 cp -r [dir1] [dir2]      # copy recursively (when copying directories)
6 cp -u [input] [output]  # copy only files that do not exist or are newer
```

MV

Move or rename files

```
1 # usage
2 mv [path/to/input] [path/to/output_folder]
3
4 # rename file
5 mv ~/Documents/file123 ~/Documents/file
```

RM

Remove files or folders

THIS IS DANGEROUS COMMAND, BE CAREFUL!

```
1  # Be careful when using these
2  rm [file]          # remove file
3  rm -r [folder]     # remove folder recursively
4  rm -rf [folder]    # remove folder recursively, and do not prompt
5
6  rm -r -i [folder]  # prompt before every removal
7  rm -r -I [folder] # prompt once
```

III. Intermediate commands

WILDCARDS

Select filenames based on patterns

```
1  *           # Any characters
2  ?           # Any single character
3  [:alnum:]   # Any alphanumeric character
4  [:alpha:]   # Any alphabetic character
5  [:digit:]   # Any digit
6  [:lower:]   # Any lowercase letter
7
8             # There are more but we are ending here
```

```
1  # Examples:
2  a*          # show everything that starts with a
3  *.txt       # show everything that ends with .txt
4  a*.txt      # Any file beginning with 'a' and ending with .txt
5  ab??def     # File: ab + "any 2 characters" + def
6  [[:digit:]]* # Any file starting with digit
7  [![:lower:]]* # Any file not starting with lowercase
```

GREP

Print lines matching a pattern (filter)

```
1  grep [text]    # display lines containing [text]
2  grep -v [text] # display lines not containing [text]
3  grep -E '[exp1]|[exp2]|[exp3]' # display lines containing exp1 or exp2 or exp3
```

PIPE

Piping standard output of left command to standard input of the right command

```
1  # pipe from left to right command
2  cmd1 | cmd2 | cmd3 | cmd 4
3
4  cat [file] | grep [text]  # read file and print lines matching a pattern
5
6  # helpers
7  uniq      # removes any duplicates from the list
8  uniq -d   # show only duplicates
9  sort      # sorting output of commands
10
11 [cmd] | sort | uniq  # show sorted unique lines of cmd output
```

REDIRECT (>, >>)

Used to redirect standard output to files. All errors are sent to standard error.

```
1  [cmd] [operator] [filename]
2
3  ls -l /usr/bin > output.txt  # text file is always overwritten
4  ls -l /usr/bin >> output.txt # output is appended to text file
5  ls -l /bin/usr 2> error.txt  # 2> is used to redirect error stream
6  ls -l /usr/bin &> output.txt # redirect both outputs to text file
7  [cmd] 2> /dev/null          # throw away unwanted output
```

ARCHIVING

Archiving and compressing

```
1  # c - create, x - extract, v - verbose, f - file, - standard IO
2  tar cvf archive.tar file1 file2 # create archive
3  tar xvf archive.tar              # unpack tar
4  gzip [file]                      # compress
5  gunzip [file]                   # uncompress file.gz
6  # one liner
7  tar z[xvf/cvf] file.tar.gz      # zxvf -> extract; zcvf -> create tar.gz
8
9  zip [name.zip] file1 file2       # zip file1 file2 into name.zip
10 unzip [name.zip]                 # unzip name.zip
11 zip -r [name.zip] dir1 dir2      # zip directories
```

CHMOD

Change permissions of file or folder

A) Octal representation (base 8)

Number	Meaning
0	---
1	--X
2	-W-
3	-WX
4	r--
5	r-X
6	rW-
7	rWX

```
1 chmod [owner group other] [file]
2
3 # Examples
4 chmod 777 build_script.sh
5 chmod 755 build_script.sh
6 chmod 000 build_script.sh
```

B) Symbolic representation

User	Meaning	Operators
u	owner	+ Add
g	group	- Remove
o	other	= Only specified permission should be applied
a	all	

```
1 chmod [user][operator] [file]
2
3 # Examples:
4 chmod a+x build_script.sh # Added execution rights to all
5 chmod g-w build_script.sh # Remove write permissions from group
6 chmod u-x build_script.sh # Remove execute permissions from owner
```

Use what works for you

SCP

Moving files over network

```
1  # local files to remote
2  scp [file1] [file2] [username]@[server_ip]:[/path/to/remote/folder]
3  # remote files to local
4  scp [username]@[server_ip]:[filename] [/local/folder]
5  # copying local directory to remote server
6  scp -r [directory] [username]@[server_ip]:[/path/to/remote/folder]
7
8  # if you are transferring large numbers of files often, take a look at rsync
```

IV. Advanced commands

AWK

Awk is a programming language. Nowadays it's mostly used for picking a single field out of an input stream. For complicated string manipulations use Python or Perl.

```
1  # usage:
2  [cmd] | awk [awk_cmd]  # or
3  awk [awk_cmd] [file]
4
5  awk '{print $1}'          # print only the first field of each line from file
6  awk '/[expr]/ {print $1}' # print first field of lines containing [expr]
7  awk '/^[expr]/ {print $1}' # match beginning of the field with expr
8  ps aux | awk 'length($0) > 40' # displays lines from ps aux longer than 40 chars
9  awk -F: '{print $1}' /etc/passwd | sort # display sorted list of login user names
10
11 -F: # change field separator to : (default is space)
```

SED

Takes an input stream, change it according to the expression and prints results to standard input

```
1  sed 's/[exp]/[text]'      # substitute expression with text
2  sed '/[exp]/d'            # deletes line that matches regular expression exp
3
4  # Examples
5  sed 's/:/%/g'              # substitute [:] with [%] globally
6  sed 2,4d                   # delete 2nd to 4th line
```


V. Shell scripts

COMMAND SUBSTITUTION

Use bash commands in bash scripts

```
1 $(command)
2
3 # Example:
4 echo $(uname -a) # prints system information
5 echo $((1+5))    # note the double braces around arithmetic expressions
6
7 echo `uname -a`  # alternative notation
```

BRACE EXPANSIONS

Execute the same command

```
1 echo folder_{1, 2, 3}
2 echo folder_{1..5}
3 echo folder_{A..Z}
```

QUOTES

A) Double quotes (")

All special characters lose their meaning, exceptions: \$, \, '

```
1 echo "$(hostname) computer uptime: $(uptime)"
```

B) Single quotes (')

Suppress all expansions

```
1 echo '$(hostname) computer uptime: $(uptime)'
```

SHELL SCRIPT

Shell scripts are meant to be small, aimed at manipulating files and executing simple commands. When complicated string manipulation is needed use high level scripting language such as Python or Perl.

VI. Tips & Tricks

1. Subshell

```
1 (cd [path] && [cmd]) # execute this and jump to current working directory
```

2. Bang Bang

```
1 !! # execute previous command
2 sudo !! # execute previous command with sudo privileges
```

3. Quick copy

Share files to another computer on localhost

```
1 # start basic web server on port 8000 from current working directory
2 python -m SimpleHTTPServer # python2
3 python3 -m http.server # python3
```