Introduction to GNU-Linux

Andrea Melloncelli

December 14, 2015

Outline

The Linux terminal - How does it work?

- What a `shell` is: the shell is a program that takes commands from the keyboard and gives them to the operating system to perform.
- A terminal (or terminal emulator) is a program that opens a window and lets you interact with the shell. There are a bunch of different terminal emulators you can use. Most Linux distributions supply several, such as: gnome-terminal, konsole, xterm, rxvt, kvt, nxterm, and eterm.

Why the shell is so powerful?

1. You often deal with plain text files, because of the language of the shell is text.
2. There are plenty of powerful tool to manipulate them, without open a text editor.
3. You can stream your data from a command to another one, and then write the output on a file.
4. You can write functions and scripts, to automatize your life and perform complex frequent operation easily.

Navigation

- Arrow keys and history: up and down arrows scroll command history.
- `pwd` : working directory
- `ls` : list directory
- `cd` : change directory
- `..` : parent directory
- full path :

  `/usr/local/lib`

- relative path :

  `local/lib`

Get materials

```
sudo apt-get install git
```

or

```
sudo yum install git
```

and then

```
git clone https://github.com/quantide/howto-linux.git
```

Read files

New commands

- ls (list files and directories)
- cat (print the file to the screen)
- less (view text files)
- file (classify a file's contents)

Manipulating files

- cp - copy files and directories
- mv - move or rename files and directories
- rm - remove files and directories
- mkdir - create directories
- wildcards and glob pattern `http://en.wikipedia.org/wiki/Glob_%28programming%29`

Let's see some examples:

```
daco@blind:~/class$ ls *.txt
1.txt  3.txt  5.txt    a.txt  c.txt  e.txt  man_less.tx
2.txt  4.txt  aaa.txt  b.txt  d.txt  f.txt

daco@blind:~/class$ ls ?.txt
1.txt  2.txt  3.txt  4.txt  5.txt  a.txt  b.txt  c.txt
```

Let's see some examples:

```
daco@blind:~/class$ ls [[:alpha:]].txt
a.txt  b.txt  c.txt  d.txt  e.txt  f.txt

daco@blind:~/class$ ls [[:digit:]].txt
1.txt  2.txt  3.txt  4.txt  5.txt

daco@blind:~/class$ ls [abc].txt
a.txt  b.txt  c.txt

daco@blind:~/class$ ls [!abc].txt
1.txt  2.txt  3.txt  4.txt  5.txt  d.txt  e.txt  f.txt
```

Examples

```
daco@blind:~/class$ ls [ace].txt
a.txt  c.txt  e.txt

daco@blind:~/class$ ls [!ace].txt
1.txt  2.txt  3.txt  4.txt  5.txt  b.txt  d.txt  f.txt
```

Help and command type

- type - Display information about command type
- which - Locate a command
- help - Display reference page for shell builtin
- man - Display an on-line command reference

Examples

```
daco@blind:~/class$ type echo
echo is a shell builtin

daco@blind:~/class$ type type
type is a shell builtin

daco@blind:~/class$ type cp
cp is hashed (/bin/cp)
```

Examples

```
daco@blind:~/class$ type ls
ls is aliased to ‘ls --color=auto’

daco@blind:~/class$ which ls
/bin/ls

daco@blind:~/class$ ls /bin/ls -l
-rwxr-xr-x 1 root root 108708 Jan 14 04:50 /bin/ls

daco@blind:~/class$ help echo

daco@blind:~/class$ man ls
```

List

```
>  : output to a file
>> : append to a file
|  : pipe to another command
<  : take file as input
```

Standard Output

```
$ ls > lsout.log
$ cat lsout.log
$ cat 1.txt > lsout.log
$ cat lsout.log
$ cat 1.txt
$ ls >> lsout.log
$ cat lsout.log
$ ls > lsout.log
$ cat lsout.log
```

Standard Input

```
$ ls -tr > lsout.log
$ cat lsout.log
$ sort < lsout.log
$ sort lsout.log
$ sort < lsout.log
$ cat < lsout.log
$ sort    < lsout.log    > sorted_lsout.log
$ cat sorted_lsout.log
```

Pipelines

```
ls | less
```

regular-expression

http://en.wikipedia.org/wiki/Regular_expression

sed (Stream EDitor) - references

- http://sed.sourceforge.net
  - http://sed.sourceforge.net/sed1line.txt
- http://en.wikibooks.org/wiki/Sed
- http://www.grymoire.com/Unix/Sed.html

sed (Stream EDitor) - syntax

```
sed [-n] 'from,to commands'      # general form
sed -n '1,10 p'                  # lines from 1 to 10
sed -n '/from/,/to/ commands'    #   between strings
sed -n 's/abc/ABC/ p'            # substitute as string
```

awk

- http://en.wikibooks.org/wiki/An_Awk_Primer/Awk_Command-Line_Examples

awk - example

```
BEGIN{
    # initialization
}
{
    # do your job
  printf "  %s:{",($1)
  for(i=2;i<=NF;i++)
  {
    printf "'%s':%d,",names[i],($i)
  }
  print "},"
}
END{
    # end up with
}
```

Pathname Expansion

```
ls *.txt
```

Tilde Expansion

```
ls ~
ls ~username
```

Arithmetic Expansion

```
echo $(( 2 + 3 ))
```

Brace Expansion

```
echo {a,b,c}
echo {a..b}
echo {a{1..3},b{2..4}}
```

Parameter expansion

```
echo $HOME
```

Command Substitution

```
ls -l $(which cp)
```

Quoting

- double quoting

```
echo "My home folder is : $HOME"
```

- single quoting

```
echo 'The home folder variable is $HOME'
```

Permissions

New commands

- chmod - modify file access rights
- su - temporarily become the superuser
- sudo - temporarily become the superuser
- chown - change file ownership
- chgrp - change a file's group ownership

Jobs

Commands

- ps - list the processes running on the system
- kill - send a signal to one or more processes (usually to "kill" a process)
- jobs - an alternate way of listing your own processes
- bg - put a process in the background
- fg - put a process in the foreground

cut and paste: columns file manipulation

- cut
- paste

Bash scripts - Create a script

A script is a sequence of operations written in a file. You can execute that file instead of all operations one by one.

Bash scripts - Use a text editor

- gedit
- emacs
- vim

Bash scripts - Vim

Basics

1. normal mode: (default, enter it with 'Esc') every letter is a command.
2. insert mode: (Enter it with 'i') now you can write into the file like other text editor.
3. command mode: (Enter it with ':' from normal mode)

Other quick information at
`http://bullium.com/support/vim.html`

Bash scripts - Insert user comments

Everythin preceded by a '#' is ignored by bash.

```
# some comments
```

Make a file executable

Adding the permission of execution:

```
chmod a+x ./file.sh
```

Execute the script:

```
./file.sh
```

PATH

```
export PATH=$PATH:directory
```

Configuration file

`.bashrc`

Here scripts

```
cat - <<EOF
Hello world!
EOF
```

New commands

- if
- test
- exit
- read
- case
- for
- while

if

```
if [ -f .bash_profile ]; then
    echo "Loading your .bash_profile"
else
    echo "You have no .bash_profile!"
fi
```

Exit status

- How to get it

  `echo $?`

- success

  0

- failure

  1

  2

  ...

  255

Read input from stdin

```
read message
```

case

```
case $character in
    1 ) echo "You entered one."
;;
    2 ) echo "You entered two."
;;
    3 ) echo "You entered three."
;;
    * ) echo "You did not enter a number between 1 and
esac
```

for - 1

```
for i in word1 word2 word3; do
    echo $i
done
```

for - 2

```
for counter in {1..10..2}
do
    echo $counter
done
```

for - 3

```
for i in "$@"; do
    echo $i
done
```

while

```
number=0
while [ "$number" -lt 10 ]; do
    echo "Number = $number"
    number=$((number + 1))
done

while read LINE
do
  echo $LINE
done < ‘tail -f /var/log/messages‘
```

ssh client (the terminal)

```
ssh username@hostname [command]
```

ssh configuration files

You will find the configuration files in

`/etc/ssh/`
`~/.ssh/`

ssh server (config file and keys)

`/etc/ssh/sshd_config`

ssh authentication

1. password
2. asymmetric keys
3. asymmetric keys (with the private key-encrypted)

a-symmetric key - key generation

- generate a key pair if needed

  `ssh-keygen`

a-symmetric key - authorization

1. alternative 1

```
scp id_rsa.pub \
     daco@192.168.0.10:~/.ssh/pustufatruntu.pub
ssh remote-host
cat ~/.ssh/pustufatruntu.pub \
      >> ~/.ssh/authorized_keys
rm ~/.ssh/pustufatruntu.pub
```

2. alternative 2

```
cat ~/.ssh/id_rsa.pub| \
      ssh daco@blind 'cat - >> ~/.ssh/authorized_ke
```

3. alternative 3

```
ssh-copy-id -i ~/.ssh/id_rsa.pub blind
```

a-symmetric key - key management

How to know the fingerprint of a key

```
ssh-keygen -lf ~/.ssh/id_rsa.pub
```

or

```
ssh-add -l
```

a-symmetric key - ssh-agent

To launch it:

```
eval $(ssh-agent) > /dev/null
```

To terminate it:

```
eval $(ssh-agent -k) > /dev/null
```

Example ssh into spark cluster

```
ssh user1@52.31.253.222
```

password: spark154

Other terminal tools

- jobs and resource information: top, ps, nice
- system information: uname, df, du
- command to print: lpr, lpq, lprm
- filesystem management: mount, fdisk, cfdisk, parted
- concatenation of processes: & , fg , bg, &&, ||, xmessage
- file compression: tar, zip, unzip, rar
- file localization: mlocate, find
- email : mutt
- watch
- tree

References

- Bash Beginner Guide: `http://www.tldp.org/LDP/Bash-Beginners-Guide/html/Bash-Beginners-Guide.html`
- Bash Reference Manual: `http://www.gnu.org/software/bash/manual/bash.html`
- Advanced Bash-Scripting Guide: `http://www.tldp.org/LDP/abs/html/index.html`
- Reference card: `http://www.tldp.org/LDP/abs/html/refcards.html`