Josep Carreras LEUKAEMIA Research Institute

Introduction to Linux and the Shell:

Useful tools and commands for bioinformatic analyses

Angelika Merkel (Head of Bioinformatics Unit IJC)
28/11/2022

Workshop overview

1. Introduction to Linux
2. Practical session I:

   Linux & the Shell (basics and commands)
3. Practial session II:

   Shell scripts with vim (text editor)

What is Linux?

= free-open source computer software environment (operating system)

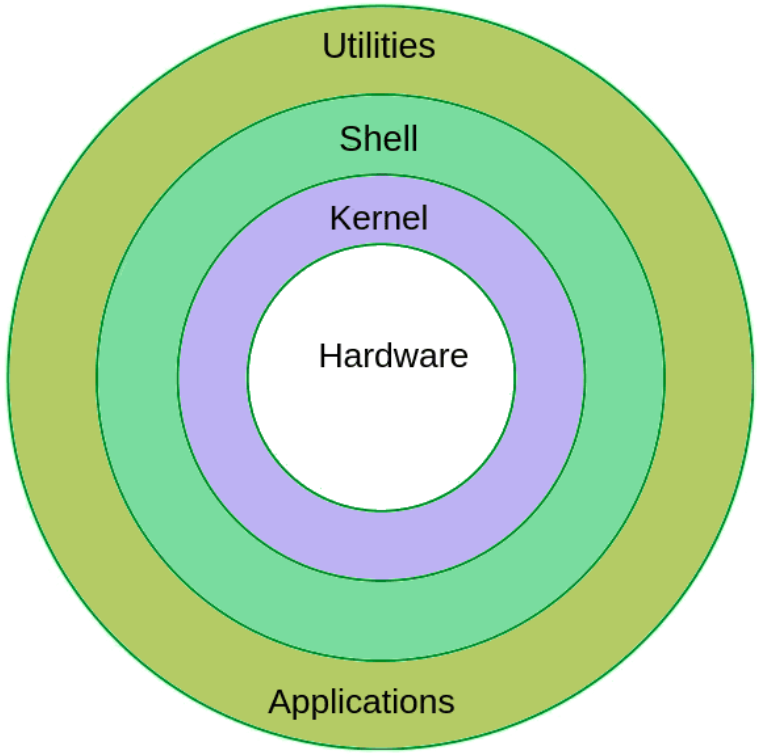
Free Software license:

- Freedom to run the program for any purpose
- Freedom to study and change the program; access to underlying source code
- Freedom to share copies to help your neighbor
- Freedom to distribute copies of modified versions for others

More than 300 Linux distributions!

https://upload.wikimedia.org/wikipedia/commons/6/6f/Linux_distros_tree.png

The Linux system



https://www.instructables.com/Linux-Presentation-in-PDF/

Utilities/ applications	programs
X	graphical system that provides windows, menus, icons, mouse support (KDE, GNOME)
Shell	user interface for typing commands, executing them and displaying the results (bsh, bash, zsh)
Kernel	low-level operating system for handling files, disks, networking, etc.

The shell

Shell = a command line interface: CLI (as supposed to graphical user interface: GUI)

Bash = Bourne Again SHell (enhanced version of the original Unix shell program, bsh written by Steve Bourne)

Terminal = a program called *terminal emulator,* opens a window and lets you interact with the shell (konsole, xterm, gnome-terminal, etc)

Let's get started..

Connect to the course server:

1. Open web browser
   `https://vpn.carrerasresearch.org/ > Login >> Linux_course`
2. Open terminal
   `ssh username@intercept`

First steps

1. The prompt
   ```
   [username@host]$
   ```
2. A typical command
   ```
   program [options] [arguments]
   ls –a mydirectory  # list all files in mydirectory
   ```
3. Getting help
   ```
   ls --help      # help option
   man ls         # manual pages for 'ls'
   ```
4. Command history:
   use ↑↓
   ```
   history
   ```
5. Autocomplete for files and commands

Finding your way around

1. Where am I?
```
pwd  # list the current working directory
```
2. What is a path?
/directory/directory/directory
3. The root directory '/'
4. Your home '~'
/home/username/
5. Go to a directory `'cd'`
```
cd somedirectory          # go to some directory
cd . ;                    # go to current directory
cd ..;                    # go one level up
cd -                      # go to previous directory
cd ~                      # go home
```

*E1

Basic files and directory operations

```
ls                                  # list files
mkdir mydirectory                   # create a directory
cp file newfile                     # copy file
cp -r mydirectory newdirectory      # copy directory (recursively)
mv filename newfilename             # move (rename) file or directory
rm file                             # remove (delete) file
rm -r directory                     # remove (delete) directory (recursively)
ln -s file -n alternativename       # create a soft link (alternative name)
```

Wildcards '*'

= shorthand for specifying files with similar names

```
ls *.bed  # list all filenames ending with '.bed'
```

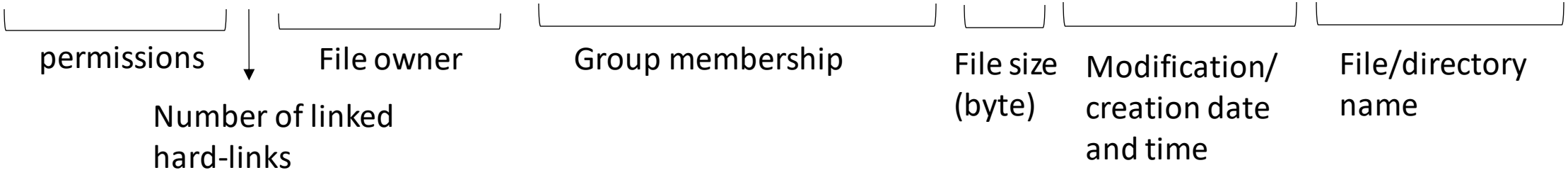
*E2

File attributes with '`ls`'

ls : list information about files, default current directory

```
-l                          # use a long listing format
-h, --human-readable        # with -l and -s, print sizes like 1K 234M 2G etc.
-t                          # sort by modification time, newest first
```

```
amerkel@INTERCEPT:/home$ ls -lth
total 44K
drwx------  6 amerkel       isilon_merkel_group    4,0K nov 28 11:04 amerkel
drwxrwxrwt  2 super          root                   4,0K nov 28 10:39 shared
drwx------  5 idevillasante isilon_merkel_group    4,0K nov 28 10:10 idevillasante
drwx------  5 jalcantara    isilon_admins          4,0K nov 28 09:41 jalcantara
drwxr-xr-x 19 super          super                  4,0K nov 28 09:40 super
```



File properties

`chown` # change ownership of a file or directory

`chmod` # change permissions of a file or directory

o,g,u,a = owner, group, users, all
r,w,x = read, write, execute

```
$ chmod a+rx myfile    # add read and execute permissions for all for file
```

`du` # measure disk usage of file and directories

-d*N* # *N* depth of directory hierarchy
-h # human readable size

```
$ du -d0 -h mydirectory
```

Compressed files and directories

```
tar  # pack or unpack directories
  tar tzf myfile.tar.gz            # list contents
  tar xzf myfile.tar.gz            # unpack
  tar czf myfile.tar.gz dirname    # pack directory dirname (gzipped)
```

```
gzip, gunzip      # compress or uncompress files in GNU zip format (.gz)
bzip2, bunzip2    # compress or uncompress files in Burrows-Wheeler format (.bz2)
zip, unzip        # compress or uncompress file in Windows zip format (.zip)
```

```
gzip myfile  # produces myfile.gz and the original file is deleted
zcat myfile  # uncompress to standard output
```

Standard input/output, standard error

Default input device = keyboard, Default output device = screen

- Standard input to the Shell can come from the keyboard or a file:

```
$  mycommand < infiles
```

- Likewise, any command that writes to standard output can have its output directed to a file as well:

```
$  mycommand > outfile  # create/overwrite outfile
$  mycommand >> outfile # append to outfile
```

- Standard error are system messages written standard output

```
$ ls lala
 ls: cannot access 'lala': No such file or directory
```

- Standard error can be directed to a file as well:

```
$  mycommand 2> errorfile
$  mycommand > outfile 2> errorfile # separate files for out and error
```

File viewing and info

```
cat                     # view files in their entirety, concatenate
  $  Cat file1 file2 > newfile   # concatenate two file into a new file
  $  Cat file1 >> newfile        # append a file to another


less & more             # view files by page

head or tail            # view the first or last part of a file

  $ head -20 file       # view the first 20 lines of a file


wc                      # count characters, words, lines in a file
  $ wc -l file          # count all lines in file
```

The pipe '|' operator

Using the shell, you can redirect standard of one command to be the standard input of another:

```
$ head myfile | wc -l
```

*E4

File text manipulations

```
Sort      # print lines of text file in specified order
            (default: alphabetical)
    -n          # sort numerically
    -r          # reverse order
    -k f        # sort by field f
    -c          # don't sort, only check if it is sorted

   $ sort -k 5n myfile    # sort by 5th field numerically


Uniq      # filter matching line from input
    -c          # count number of unique lines
    -u          # print only unique lines
    -d          # print only duplicated lines
```

File text manipulations

```
Cut     # extract columns of a text file
      -f 1,3     # extract columns 1 and 3
                   (ranges: 1-3 = 1 to 3; 5- = 5 to last; -16 = first to 16 )
      -d,        #  field seperator  (default: tab seperator)

Paste   # combine two text files side by side
     $  paste file1 file2
```

Combine cut and paste redirecting standard output:

```
$ cut –f 1 file1 | paste file2 -
```

*E5

File text manipulations: regular expression

```
Grep  # print all lines matching a regular expression
  -v   # print lines that do not match the regular expression
  -w   # match only complete words
  -c   # print a count of matching lines
  -A N # after each matching line, print the next N lines from this file
  -B N # before each matching line, print the next N lines from this file
  –E   # or egrep for extended regular expression
```

```
$ grep chr1 peaks.bed
```

For a tutorial on how to use regular expression and extended regular expression look here

*E6

More powerful text manipulations

Sed = pattern-matching engine that can perform manipulations on lines of text

```
$ sed 's/red/hat/g' myfile     # print file with all occurences of "red" substituted for "hat"
$ sed '1,10d' myfile           # print file with the first 10 lines removed
```

Awk = pattern matching language. It can match data by regular expression and perform actions on the data.

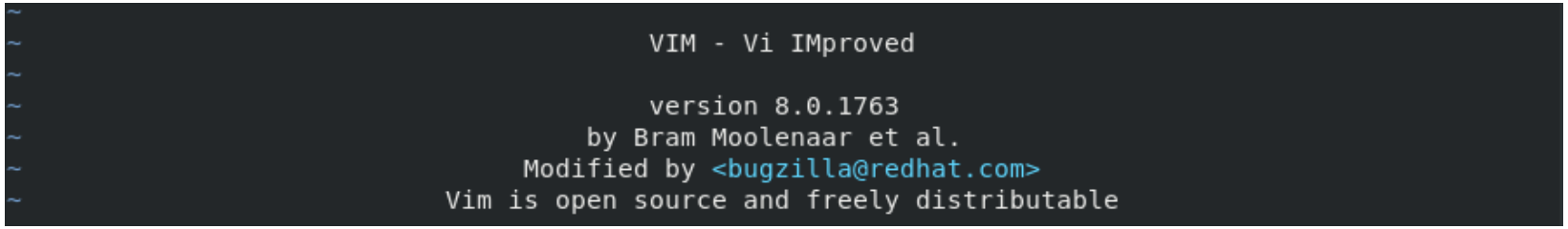
```
 $ awk '{print $1, $5}' myfile             # print first and fifth field
 $ awk '$1== "chr1" && $5 >100' myfile     # filter first and fifth
field by                                                strings/values
 $ awk '{$1 ~/chr1/}' myfile               # filter first field for matches'chr1'
```

Online tutorials Linux

https://ryanstutorials.net/linuxtutorial/

Programming with the shell

Vim command line editor



Vim shellscript.sh # opens a new file 'shellscript.sh' with vim

Basic vim:

[esc] # switch mode
i # enter edit mode

: # enter command mode

wq # when in command modes: save and exit vim
q! # when in command mode: exit without saving

For more on vim, check here

Shell script

1. create shellscript.sh

```
#! /bin/bash


### Author:

### Date:

### Description:
### My first shell script



# do something

echo "Hello world!"
```

2. execute

make the script executable

```
$ chmod a+x shellscript.sh
```

run the script

```
./shellscript.sh
```

Shell variables: $

Build-in variables

PWD = current working directory

PATH = path executables

Assign variables

```
MYVAR="Hello world!"
Echo $MYVAR          # return variable
printf $Myvar        # print variable formatted
MYVAR=`ls *bed`      # assign the output of a command to a variable
MYVAR=$( ls *bed )
```

Arguments

By default, any string after a script is passed to the script as a build-in variable

```
$ ./shellscript.sh arg1 arg2
Arg1 -> $1
Arg2 -> $2
```

Control structures: if-else

If-else

```
if [condition]
 then
   statement1
 else
   statement2
fi
```

example

```
#! /bin/bash

N=$1
M=$2
if [ $N -eq $M ]
 then
   echo "Hello world!"
 else
   echo "Good bye world!"
fi
```

Testing expressions (bash)

Statement	Description
test EXPRESSION	Test if an expression is true
[EXPRESSION]	Short hand for test, used on all POSIX shells
[[EXPRESSION]]	Short hand for test, available with newer shells like bash, ksh, zsh

Expression	Meaning
&&	Logical AND
||	Logical OR
-eq	Equality check
-ne	Inequality check
-lt	Less Than
-le	Less Than or Equal
-gt	Greater Than
-ge	Greater Than or Equal

Testing if a file exists

```
FILE=/etc/resolv.conf

if [ -e "$FILE" ]
 then
  echo "$FILE exists."
fi
```

Control structure: for-loop

for loop

```
for iteration
 do
  something
done
```

example

```
#! /bin/bash

for i in {1..10..2}
 do
   echo "Hello $i world!"
done
```

Control structure: while-loop

While loop

```
while [condition]
 do
  something
done
```

example

```
#! /bin/bash

COUNTER=0
while [ $COUNTER -le 5 ]
do
  echo "Welcome $COUNTER times"
  COUNTER=$(( $COUNTER + 1 ))
done
```

More on shell scripting

https://ryanstutorials.net/bash-scripting-tutorial/

Thank you!

IJC Bioinformatics