Practical Bioinformatics

Basic Linux

Part 2

Stefan Wyder
stefan.wyder@uzh.ch
URPP Evolution
www.evolution.uzh.ch

Universität Zürich[UZH]

URPP

What we learned in Part 1

- Command -Option(s) Parameter(s)
  ls -rlh ~/data

- Working with files / directories
  ls, cd, mkdir, cp, mv, rm

- Directory structure
  everything is under the root: /

- Working with text files
  head, less, grep, cut, sort, tr, wc, uniq

- Tools can be connected by "|"

- The Mac OS X Shell differs from the typical Linux shell

What we will do today

- Connecting with Unix/Linux servers

- Automatizing tasks (scripting)
- Installing and running software
- Permissions

- File/Dir Compression&Extraction

- (Search Patterns, Regular Expressions)

Server commands

Command	Task
ssh -X *user@hostname*	Connect to server
scp <what> <towhere>	Transfer file from/to server
sftp *user@hostname*	Transfer file from/to server (interactive)

Connect to remote computer

```
$ ssh username@mnf-44.uzh.ch
```

```
RSA key fingerprint is 71:ed:af:1f:d6:0a:43:05:8d:11:34:68:
2c:2d:79:01.
Are you sure you want to continue connecting (yes/no)?
```

Type "yes" and press ENTER.
Then you will be asked for your password.

```
$ bash          # we will use the bash shell
$ whoami        # what is my username on this host
$ uname -a      # show basic info of the host OS
$ df -h         # displays free disk space
```

File/Dir compression

	Command	Meaning
1 file	**gzip** *filename*	compress file with gzip (adds .gz extension)
	gunzip *filename.gz*	decompress file with gzip (removes .gz extension)
Many files/ Directories	**tar xfz** *filename.tar.gz*	extract/decompress files from tar.gz archive
	tar zcvf *archive.tar.gz folder_to_compress*	creates archive.tar.gz
	unzip *filename.zip*	unzip archive

zmore, zless, zgrep, ...

Installing &Running programs

Bash (shell) scripts

- All commands you enter on the command line can be put in a text file which will be executed by bash line by line

- Scripts are simple text files + shebang line
  (all commands on a separate line)

```
#!/bin/bash
echo "Hello World"
echo "This is my first script"
```

Running programs

- Software are executable files (permissions!)

- Run a bash script

```
chmod +x script1.sh                    # make file executable
./script1.sh  or   bash path/script.sh     # run
```

- Its the same for any scripting language (python/perl,...)

```
chmod +x script.py
./script.py     or python path/script.py
```

- Run a binary

```
chmod +x bowtie
./bowtie
```

Installing Software (binaries)

- Packages
  Using a package manager - takes also care of dependencies
  - Linux:
    Ubuntu: via .deb files (e.g. aptitude or apt-get)
    Fedora/SUSE: via .rpm files
  - Mac OS X: homebrew (my favourite), MacPorts, fink

- Compiling from source
  Typically open-source software is written in C/C++ -> GCC compiler
  - Linux: install gcc using the package manager (apt-get search gcc, then apt-get install *gcc-XXXX*)
  - Mac Os X: install gcc using homebrew (brew search gcc, ...) or via XCode

```
./configure
make
make install       # optional
make clean         # optional
```

$PATH

- Bash only looks at certain directories for commands/software/programs

```
echo $PATH
/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin:/opt/X11/bin:/usr/texbin
```

- $PATH is an environment variable - used to configure your system

```
env
TERM_PROGRAM=Apple_Terminal
SHELL=/bin/bash
TERM=xterm-256color
CLICOLOR=1
TMPDIR=/var/folders/n3/l__1kfy131j6sl8t9_sptlh40000gn/T/
Apple_PubSub_Socket_Render=/tmp/launch-YtFYZC/Render
TERM_PROGRAM_VERSION=309
TERM_SESSION_ID=E48E5A7D-1DB0-4D37-AEFD-4711C32BEC3C
USER=swyder
COMMAND_MODE=unix2003
SSH_AUTH_SOCK=/tmp/launch-psPvtD/Listeners
Apple_Ubiquity_Message=/tmp/launch-NgiUKs/Apple_Ubiquity_Message
__CF_USER_TEXT_ENCODING=0x1F5:0:3
LSCOLORS=dxfxcxdxbxegedabagacad
PATH=/usr/bin:/bin:/usr/sbin:/sbin:/usr/local/bin:/opt/X11/bin:/usr/texbin
PWD=/Users/swyder/TALKS/URPP_Teaching
LANG=de_CH.UTF-8
HOME=/Users/swyder
SHLVL=1
LOGNAME=swyder
DISPLAY=/tmp/launch-P4TPOi/org.macosforge.xquartz:0
SECURITYSESSIONID=186a4
_=/usr/bin/env
```

To install an executable

1. You copy it into one of the folders in $PATH
2. You add its directory to $PATH
3. You create a symbolic link to it into a folder contained in $PATH

Details

2. export PATH=$PATH:*directory*

3. sudo ln -s executable *directory*

Permissions

```
$ ls -l
```

```
-rw-r--r--   1 swyder  staff      6677 30 Okt 22:02 At.gff
-rw-r--r--@  1 swyder  staff   3723486 28 Okt 17:30 CheatSheetguideUNIX.pdf
drwxr-xr-x  11 swyder  staff       374 11 Nov 16:13 FASTAS
-rw-r--r--   1 swyder  staff        21 30 Okt 22:04 indA.txt
```

Permissions

Owner

Group

File size
(bytes)

Last modification

File type
d : directory
- : regular file

Changing Permissions

r: read
w: write
x: execute

```
$ chmod [ugo][+-][rwx] file
```

u: user
g: group
o: other/world

Make a file executable (for you)

```
$ chmod +x file
chmod ug+rx my_file
```

Setting exact permission (only reading, for you)

```
$ chmod =r file
```

Removing permissions (for you)

```
$ chmod -wx file
```

Summary: executables

- Come in two flavours: Scripts / Binaries
- Execute permissions must be set:

  `chmod +x programname`

- Scripts mostly start with the shebang line, telling the shell which interpreter to use. E.g.

  #!/usr/bin/perl

- Executing of executables

  ```
  ./prg
  ./prg.sh   (bash prg.sh)
  python prg.py
  perl prg.pl
  ```

Regular Expressions (=regex)

- a way to describe *set of strings*
- many Linux tools use it (egrep, sed) as well most programming languages

Wildcard	Matches
*	zero or more characters
?	exactly one character
[abcde]	exactly one of the characters listed
[a-e]	exactly one character in the given range
[!abcde]	any character not listed
[!a-e]	any character that is not in the given range
{URPP,evolution}	exactly one entire word from the options given

Regex	chr	chr[1-5]	chr.	AAF12\.[1-3]	AT[1,5]G[:digit:]+\.[1,2]
	chr1	chr1	chr1	AAF12.1	AT5G08160.1
	chr2	chr2	chr2	AAF12.2	AT5G08160.2
	chr3	chr3	chr3	AAF12.3	AT5G10245.1
	chr4	chr4	chr4		AT1G14525.1
	chr5	chr5	chr5		
	chr6		chr6		

- Example: from TAIR9_mRNA.bed, filter out the mRNA structures from chr1 and only on the + strand.

- `$ egrep '^chr1.+\+' TAIR9_mRNA.bed  > out.txt`

^ matches the start of a string

. matches any char

Since + is a special character (standing for a repeat of one or more), we need to escape it.

\+ matches a '+' symbol as such

^chr1
Matches lines
With 'chr1' appearing
At the beginning

.+
matches
any string

Together in this order, the regex
Filters out lines of chr1 on + strand

```
chr1 2025600 2027271 AT1G06620.1 0 + 2025617 2027094 0 3541,322,429,
chr1 16269074 16270513 AT1G43171.10 + 1626998816270327 0 1
chr1 28251959 28253619 AT1G75280.10 + 2825202928253355 0 5
chr1 693479 696382 AT1G03010.10 + 693479 696188 0 592,67,1197,247
```

Sources & Links

Acknowledgements

- Some exercises from Gregor Roth / von Mering group (IMLS, UZH)

Material

- SIB course http://edu.isb-sib.ch/course/view.php?id=41
- O'reilly Books http://oreilly.com/linux/
- Video tutorials (~100 min) http://software-carpentry.org/v4/shell/index.html
- Cheatsheet http://www.embnet.org/sites/default/files/quickguides/guideUNIX.pdf

Permission code

Owner	Group	Other
r w x	r w x	r - x
4+2+1	4+2+1	4+0+1
7	7	5

r: read
w: write
x: execute

chmod 775 *file*