

Lecture 02

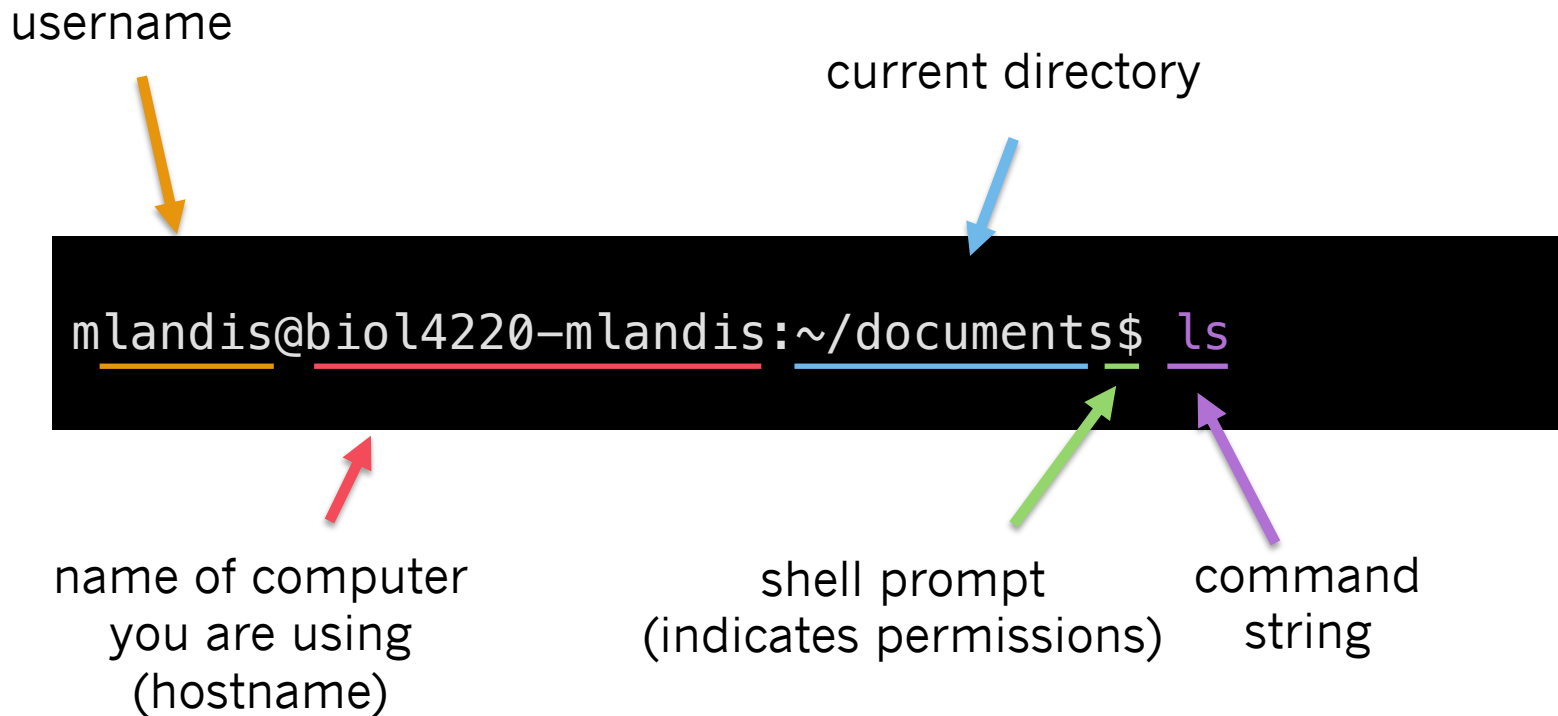
Unix commands & filesystem



Course: Practical Bioinformatics (BIOL 4220)
Instructor: Michael Landis
Email: michael.landis@wustl.edu

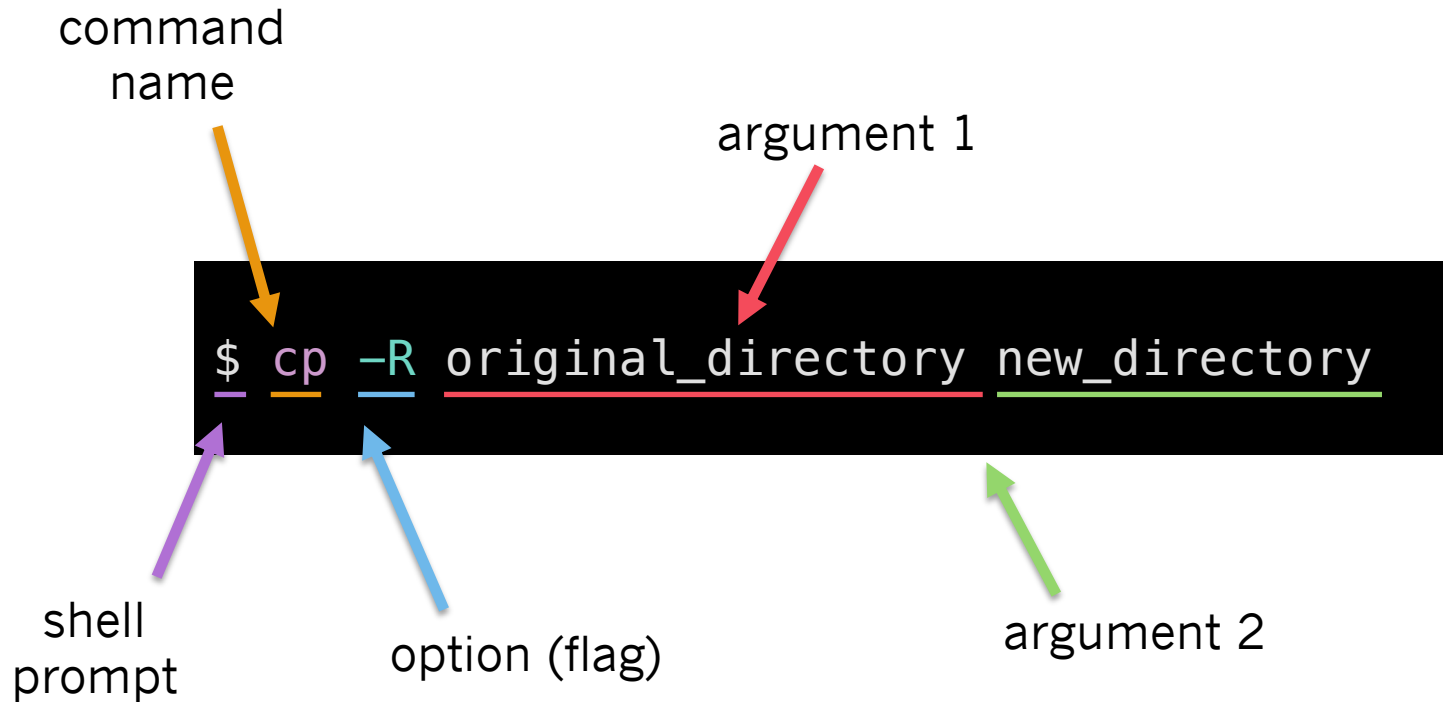


Command line



The **command line** accepts commands
provided by the user (you!)

Command string



A ***command*** is applied against an ***argument(s)*** and its behavior can be modified by ***option(s)***

Some commands require a specific number of arguments (exactly 0 or 1 or 2 or ...)

```
$ pwd  
$ cd documents  
$ mv file.txt backup.txt
```

Other commands are flexible

```
$ cat file1.txt file2.txt file3.txt  
$ rm tmp1.txt tmp2.txt tmp3.txt
```

Options enable optional(!) extra command features

```
$ ls -l -a -r -t  
$ ls -lart
```

Some options require arguments

```
ssh -p 22 128.252.89.47
```

A ***comment*** is a “non-command” that helps communicate intent to humans

```
$ # list _all_ files by reverse time sort  
$ # note: MJL forgot to use the -a flag and accidentally  
$ #      deleted all user profiles on 21-08-18  
$ ls -lart
```

“Documentation is a love letter that
you write to your future self.”

- *Damian Conway*

Commands are ***executed*** one at a time, and in the order that they are received

```
$ # first we'll create a folder  
$ mkdir data  
$ # then we'll enter the folder  
$ cd data
```

Multiple commands separated by ; can be executed in a single line of text

```
$ # first create then copy output.txt  
$ touch output.txt; cp output.txt copy.txt
```

echo, print text

```
$ # echo prints the argument(s)  
$ # to standard output (stdout)  
$ echo Hello, world!  
Hello, world!  
$ echo GATTACA  
GATTACA
```


/s, list files

```
$ # ls lists all contents in the current
$ # working directory, printed to stdout
$ ls
labs  lectures  notes.txt

$ # ls accepts target directories as
$ # arguments, too
$ ls labs lectures
labs:
lab_01  lab_02
lectures:
lect_01.pdf  lect_02.pdf
```

cat, concatenate

```
$ # cat concatenates the contents of  
$ # one or more files together, and then  
$ # prints everything to stdout  
$ cat hello.txt  
Hello,  
$ cat world.txt  
world!  
$ cat hello.txt world.txt  
Hello, world!
```

mv, move

```
$ # mv moves files and folders within the  
$ # filesystem; one use is to rename files  
$ ls  
notes.txt  
$ mv notes.txt old_notes.txt
```

```
$ # mv can rename folders, too  
$ ls  
lectures  
$ mv lectures old_lectures  
$ ls  
old_lectures
```

cp, copy

```
$ # cp copies the first argument (file) to  
$ # the location targeted by the second  
$ # argument (file)  
$ ls  
notes.txt  
$ cp notes.txt old_notes.txt  
$ ls  
notes.txt  old_notes.txt
```

```
$ # cp cannot copy folders (unless -R option used)  
$ cp lectures old_lectures  
cp: lectures is a directory (not copied).
```

rm, remove

```
$ # rm removes a file from the filesystem;  
$ # removed files are not easily restored!  
$ ls  
notes.txt  
$ rm notes.txt  
$ ls  
  
$
```

nano, text editor

Easy to use, but simple features

```
GNU nano 7.2                                error.log *
#68 165.4      @ ./essentials.jl:926 [inlined]
#68 165.4      [15] invoke_in_world
#68 165.4      @ ./essentials.jl:923 [inlined]
#68 165.4      [16] _require_prelocked(uuidkey::Base.PkgId, env::String)
#68 165.4      @ Base ./loading.jl:1803
#68 165.4      [17] macro expansion
#68 165.4      @ ./loading.jl:1790 [inlined]
#68 165.4      [18] macro expansion
#68 165.4      @ ./lock.jl:267 [inlined]
#68 165.4      [19] __require(into::Module, mod::Symbol)
#68 165.4      @ Base ./loading.jl:1753
#68 165.4      [20] #invoke_in_world#3
#68 165.4      @ ./essentials.jl:926 [inlined]
#68 165.4      [21] invoke_in_world
#68 165.4      @ ./essentials.jl:923 [inlined]
#68 165.4      [22] require(into::Module, mod::Symbol)
#68 165.4      @ Base ./loading.jl:1746
#68 165.4      [23] include

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

<https://www.nano-editor.org/dist/latest/cheatsheet.html>

vim, text editor

Harder to use, but far more powerful

```
1 # read in accession list
2 ACCESSIONS=`cat $1 | cut -d"," -f2`
3 SEQ_DIR=$2
4
5 # enter sequence directory
6 mkdir -p $SEQ_DIR
7 cd $SEQ_DIR
8
9 # get each accession
10 for ACC in $ACCESSIONS;
11 do
12     # fetch the accession, separating each coding gene into its own fasta entry
13     efetch -db nuccore -format fasta_cds_na -id $ACC > $ACC.fasta
14 done
15
16 # loop through all fasta files in seq_dir
17 for FILE in `ls *.fasta`;
18 do
19     # get accession from filename
20     ACC=`echo $FILE | cut -d '.' -f 1`
21     # transform text, using `!` as special delimiter, to be cut later
22     TEXT=`cat $FILE | tr -d '\n' | sed 's/>/\n>/g' | rev | sed 's/]/!]/' | rev`
23     # loop through each entry

```

NORMAL example_get_seq.sh utf-8[unix] 1% 1/52 :1

Cheatsheet:

<https://vimsheet.com/>

Tutorials:

<https://openvim.com/>

<https://www.vimninja.com/>

Unix filesystem

The **filesystem** organizes files and folders into a hierarchical structure

- **files** contain data, e.g.
 - text, programs, music
- **folders** contain files and/or other folders

The filesystem also helps secure **user permissions** to read/write/execute filesystem objects (*more details later*)

Example filesystem

```
$ tree
.
├── home
│   ├── course_project.md
│   ├── course_schedule.md
│   ├── how_to_guide.md
│   ├── labs
│   │   ├── lab_01A.md
│   │   └── lab_01B.md
│   └── lectures
│       ├── lect_01A.pdf
│       └── lect_01B.pdf

```

3 directories, 7 files

home is the parent directory for *labs*;
labs contains two pdf files

Filesystem paths

A ***path*** is the address of a filesystem object

- the path lists all parent directories, from *deep* to *shallow*, to locate the object
- directories are separated by /
- file paths end with the ***filename***, e.g. *output.txt*
- folder paths do not end with a filename

file path

```
/home/mlandis/Biol4220/lectures/lect_02.pdf
```

nested folders

filename

An ***absolute path*** specifies all nested folders, beginning with the deepest folder, the ***root directory***

A ***relative path*** specifies only those folders needed to locate a resource *relative* to your current location in the filesystem

If your current location is */home/mlandis/Biol4220*, these paths refer to the same file

```
/home/mlandis/Biol4220/lectures/lect_02.pdf
```

```
lectures/lect_02.pdf
```

Special directories

The **home directory** (~) is a shortcut for */home/username*; most of your work takes place in ~

The **current directory** (.) is a shortcut for the directory you occupy currently; can be useful to make relative paths explicit

The **parent directory** (..) is a shortcut for the directory that *contains* the current directory; useful for moving “up” one directory

These paths are equivalent
if you're located in */home*

```
# absolute
/home/mlandis/Biol4220/labs
# relative
mlandis/Biol4220/labs
# using home directory (~)
~/Biol4220/labs
# using current directory (.)
./mlandis/Biol4220/labs
# using parent directory (..)
/home/mlandis/Biol4220/lectures/../../labs
```

mkdir, make directory

```
# mkdir makes a new directory
# specified by the path argument
~/labs$ ls
# relative path
~/labs$ mkdir lab_01
~/labs$ ls
lab_01
# using .. in path
~/labs$ mkdir ../lectures
~/labs$ ls ..
labs  lectures
# absolute path
~/labs$ mkdir /home/mlandis/labs/lab_02
~/labs$ ls
lab_01  lab_02
```

cd, change directory

```
# cd changes into the directory
# specified by the path argument
~$ ls
labs
~$ ls labs
lab_01 lab_02
# relative path
~$ cd labs/lab_02
# using .. in path
~/labs/lab_02$ cd ../lab_01
# absolute path
~/labs/lab_01$ cd /home/mlandis
~$
```

rmdir, remove directory

```
# rmdir removes an empty directory
# specified by the path argument
~$ ls
labs lectures
# labs isn't empty!
~$ rmdir labs
rmdir: labs: Directory not empty
~$ ls labs
lab_01 lab_02
~$ ls labs/lab_01 labs/lab_02
lab_01:
lab_02:
# remove subdirectories
~$ rmdir labs/lab_01
~$ rmdir labs/lab_02
# now remove labs
~$ rmdir labs
```


cp -R, copy folders

```
# cp -R will copy a directory and
# recursively copy all internal files
# and directories
~$ ls lectures
lect_01.pdf lect_02.pdf
# cp cannot target directories by default
~$ cp lectures lectures_old
cp: lectures is a directory (not copied).
# add the -R flag
~$ cp -R lectures lectures_old
~$ ls
lectures  lectures_old
```

rm -rf, remove completely

```
# WARNING: this is a very dangerous command!
#
# rm -rf will remove a file or directory,
# along with all of its contents, without
# any warnings or user interactions
~$ ls
labs
~$ ls labs
labs_01 labs_02
~$ rmdir labs
rmdir: labs: Directory not empty
~$ rm labs
rm: labs is a directory
~$ rm -rf labs
~$ ls
~$
```

Suppose this lists all filesystem objects

- what directories are shown?
- what files are shown?
- which folder contains three files?
- which folder contains two directories?
- including the root directory, how many directories are in the absolute path for lect_02.pdf?

```
/home/mlandis/Biol4220  
/home/mlandis/Biol4220/notes.txt  
/home/mlandis/Biol4220/labs  
/home/mlandis/Biol4220/labs/lab_01.pdf  
/home/mlandis/Biol4220/labs/lab_02.pdf  
/home/mlandis/Biol4220/lectures  
/home/mlandis/Biol4220/lectures/lect_01.pdf  
/home/mlandis/Biol4220/lectures/lect_02.pdf  
/home/mlandis/Biol4220/lectures/lect_03_draft.pdf
```

How would you execute this series of commands?
(using only what we learned in this lecture)

1. change to the home directory
2. copy the *notes.txt* file into lectures
3. delete the lectures directory
4. copy the labs directory into your home directory

```
/home/mlandis/Biol4220
/home/mlandis/Biol4220/notes.txt
/home/mlandis/Biol4220/labs
/home/mlandis/Biol4220/labs/lab_01.pdf
/home/mlandis/Biol4220/labs/lab_02.pdf
/home/mlandis/Biol4220/lectures
/home/mlandis/Biol4220/lectures/lect_01.pdf
/home/mlandis/Biol4220/lectures/lect_02.pdf
/home/mlandis/Biol4220/lectures/lect_03_draft.pdf
```

Overview for Lab 02