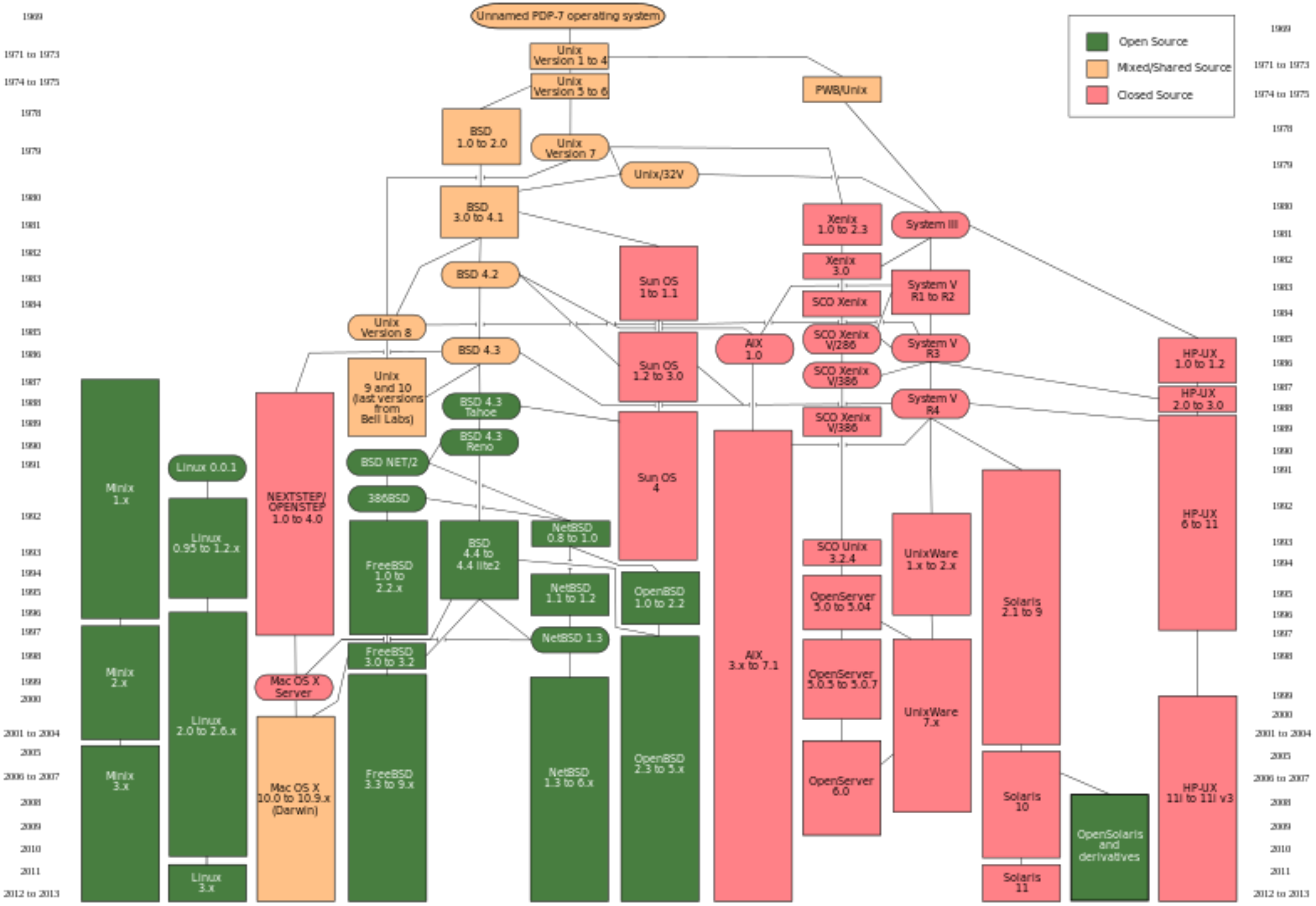
Introduction to the Command-line

Gayle Philip, VLSCI

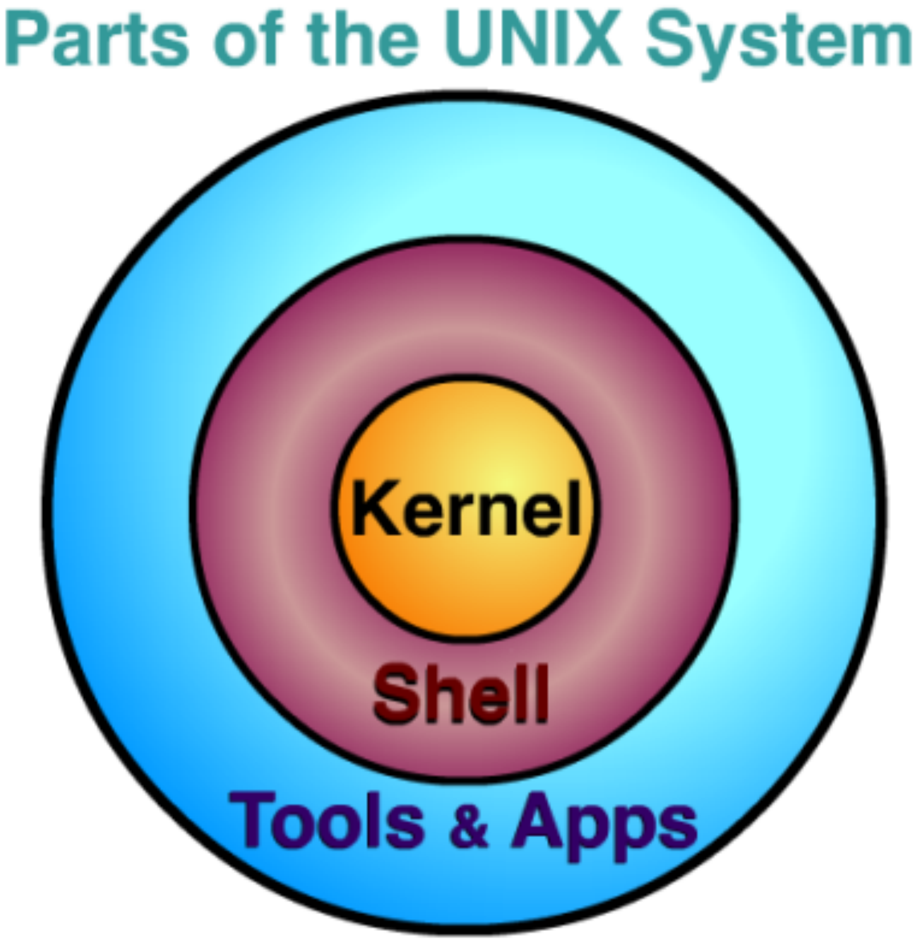
Curtin University, Perth.
12th - 14th July 2016.

Unix

- The UNIX operating system is a set of programs that act as a link between the computer and the user.
- First developed in the 1960s, and has been under constant development ever since.
- Today Unix is ubiquitous; it is found everywhere from smartphones to supercomputers.


🟩 Open Source
🟧 Mixed/Shared Source
🟥 Closed Source
1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2004
2005
2006 to 2007
2008
2009
2010
2011
2012 to 2013
Unnamed PDP-7 operating system
Unix Version 1 to 4
Unix Version 5 to 6
PWB/Unix
BSD 1.0 to 2.0
Unix Version 7
Unix/32V
BSD 3.0 to 4.1
Xenix 1.0 to 2.3
System III
Xenix 3.0
BSD 4.2
Sun OS 1 to 1.1
System V R1 to R2
SCO Xenix
Unix Version 8
AIX 1.0
SCO Xenix V/286
System V R3
BSD 4.3
Sun OS 1.2 to 3.0
HP-UX 1.0 to 1.2
SCO Xenix V/386
Unix 9 and 10 (last versions from Bell Labs)
System V R4
HP-UX 2.0 to 3.0
BSD 4.3 Tahoe
SCO Xenix V/386
Minix 1.x
NEXTSTEP/ OPENSTEP 1.0 to 4.0
BSD 4.3 Reno
Sun OS 4
Linux 0.0.1
BSD NET/2
386BSD
HP-UX 6 to 11
NetBSD 0.8 to 1.0
Linux 0.95 to 1.2.x
FreeBSD 1.0 to 2.2.x
BSD 4.4 to 4.4 lite2
SCO Unix 3.2.4
UnixWare 1.x to 2.x
Solaris 2.1 to 9
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
OpenServer 5.0 to 5.04
NetBSD 1.3
AIX 3.x to 7.1
Minix 2.x
FreeBSD 3.0 to 3.2
Mac OS X Server
OpenServer 5.0.5 to 5.0.7
UnixWare 7.x
Linux 2.0 to 2.6.x
HP-UX 11i to 11i v3
Mac OS X 10.0 to 10.9.x (Darwin)
FreeBSD 3.3 to 9.x
NetBSD 1.3 to 6.x
OpenBSD 2.3 to 5.x
OpenServer 6.0
Solaris 10
Minix 3.x
OpenSolaris and derivatives
Linux 3.x
Solaris 11

Unix Components



Unix Architecture

- Users communicate with the kernel through a program known as the shell.
- The shell is a command line interpreter that processes user requests.
- It translates (interprets) commands entered by the user and converts them into a language that is understood by the kernel.

The UNIX shell

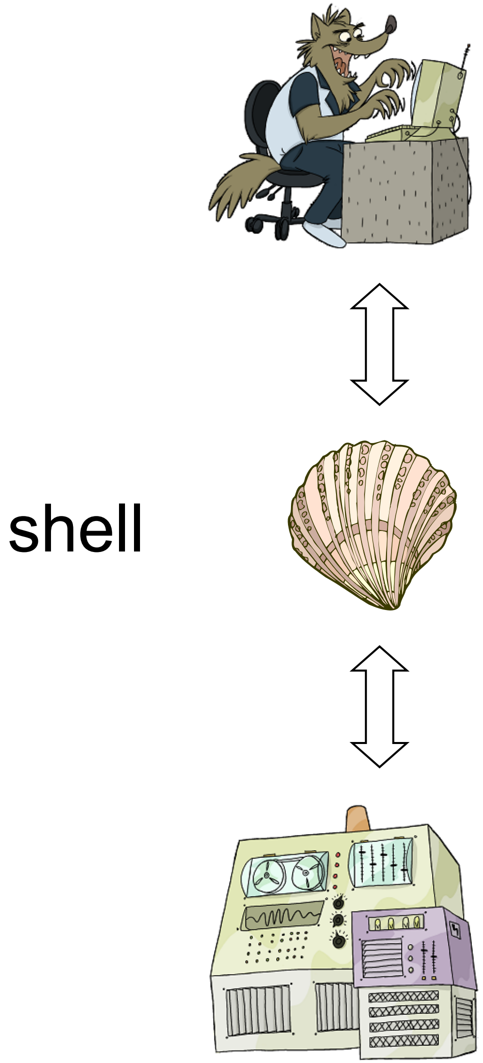
- There are many different “flavours” of the Unix shell but the most popular is bash (the Bourne again shell).
- Commands in UNIX are terse and often cryptic.
- Use it because:
  - many tools only have command-line interfaces
  - allows you to combine tools in powerful new ways

- User logs in

- User types command
- Computer executes command and prints output

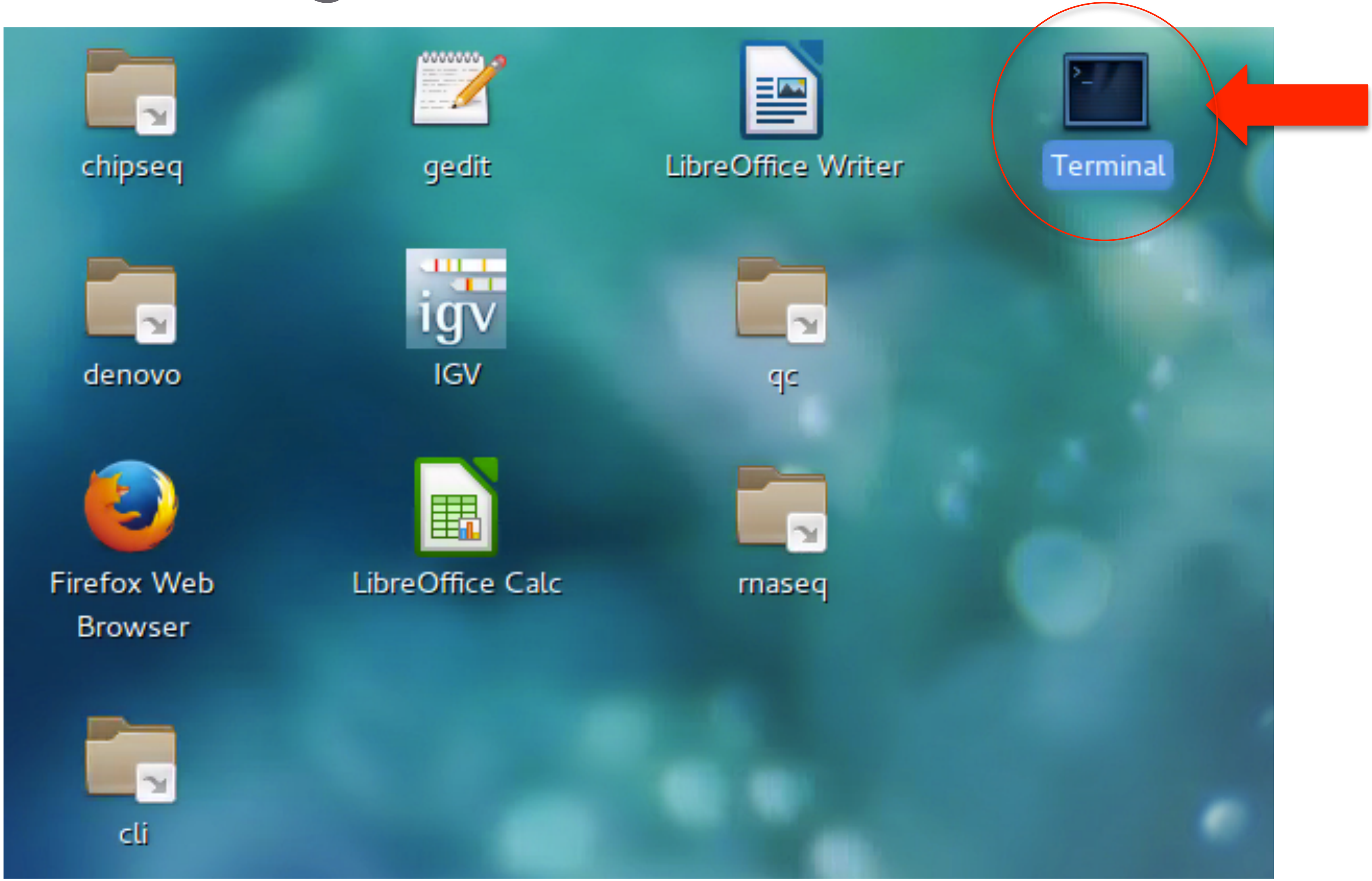
- User types another command
- Computer executes command and prints output

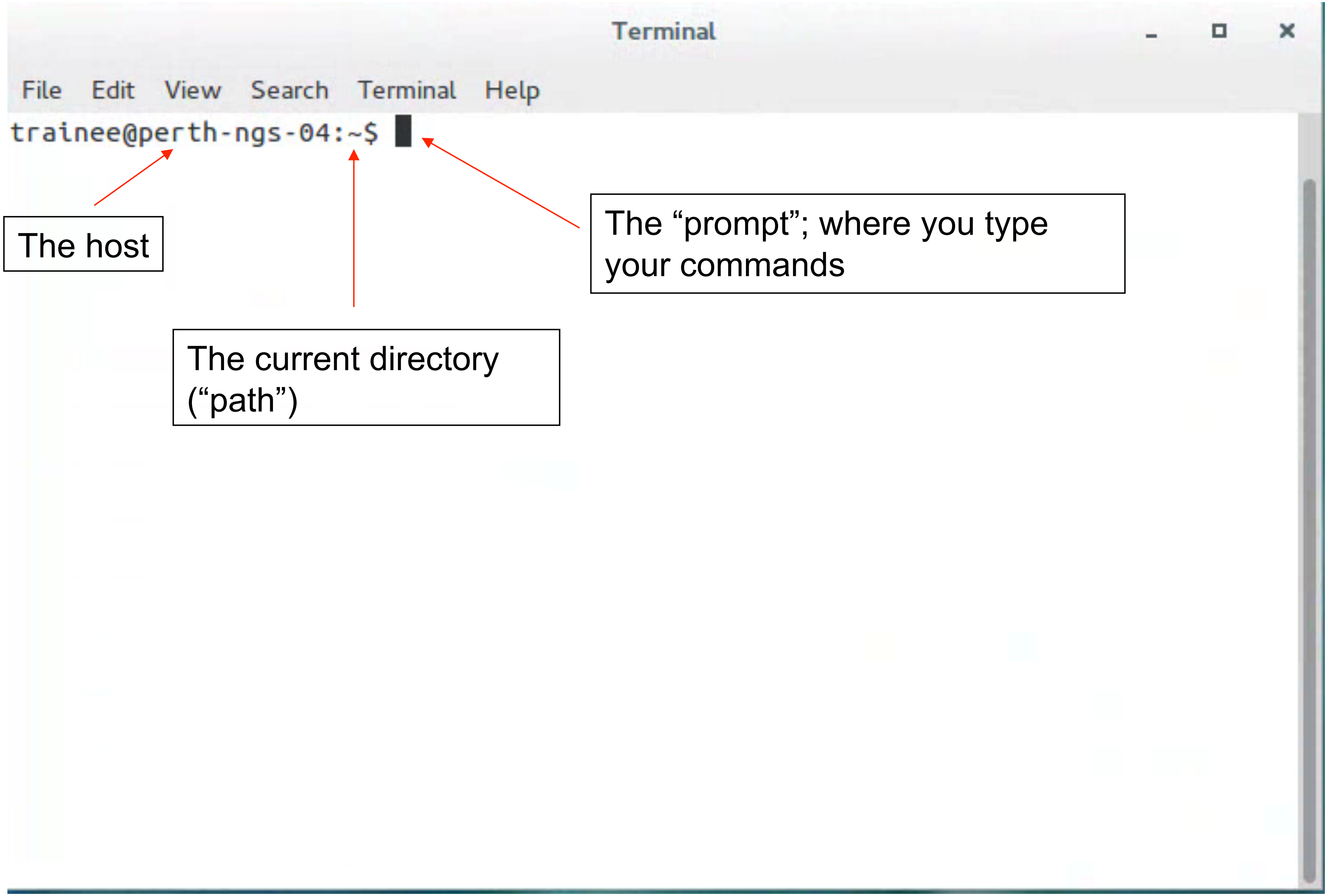
- User logs off



shell

Accessing the command-line





```
trainee@perth-ngs-04:~$
```

The host

The current directory (“path”)

The “prompt”; where you type your commands

Unix Commands

- In general, a Unix command has the format:

  - ‘Command’ ‘flags’ ‘arguments’
  - $ wc -l filename

- Change to your home directory
  $ cd

- To access the documentation/manual for any Unix command type ‘man command’
  $ man pwd

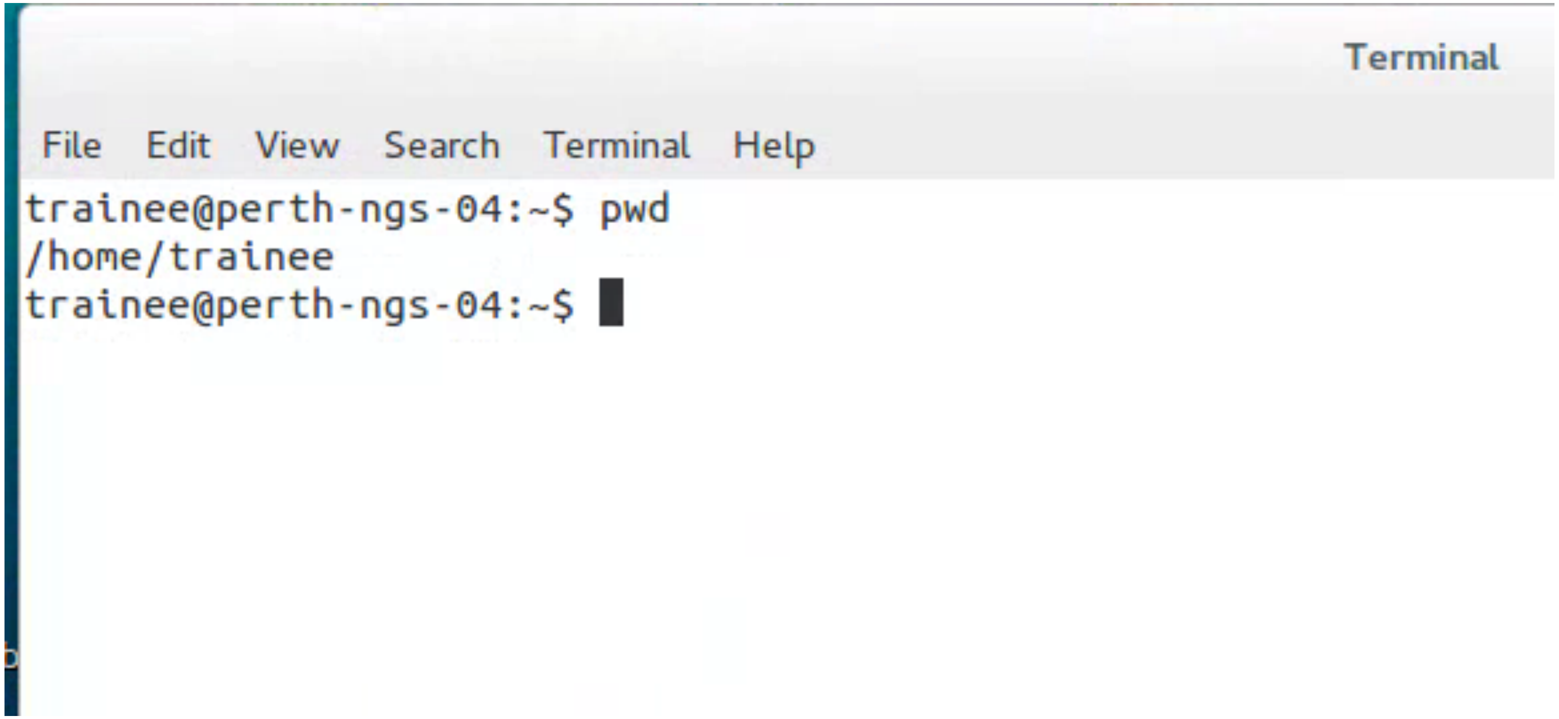
The first command

- Type the command pwd (**p**rint **w**orking **d**irectory), then press the Enter/Return key to send the command to the shell.

- More specifically, when we type pwd into the shell:
  - finds a program called pwd,
  - runs that program,
  - displays that program's output, then
  - displays a new prompt to tell us that it's ready for more commands.

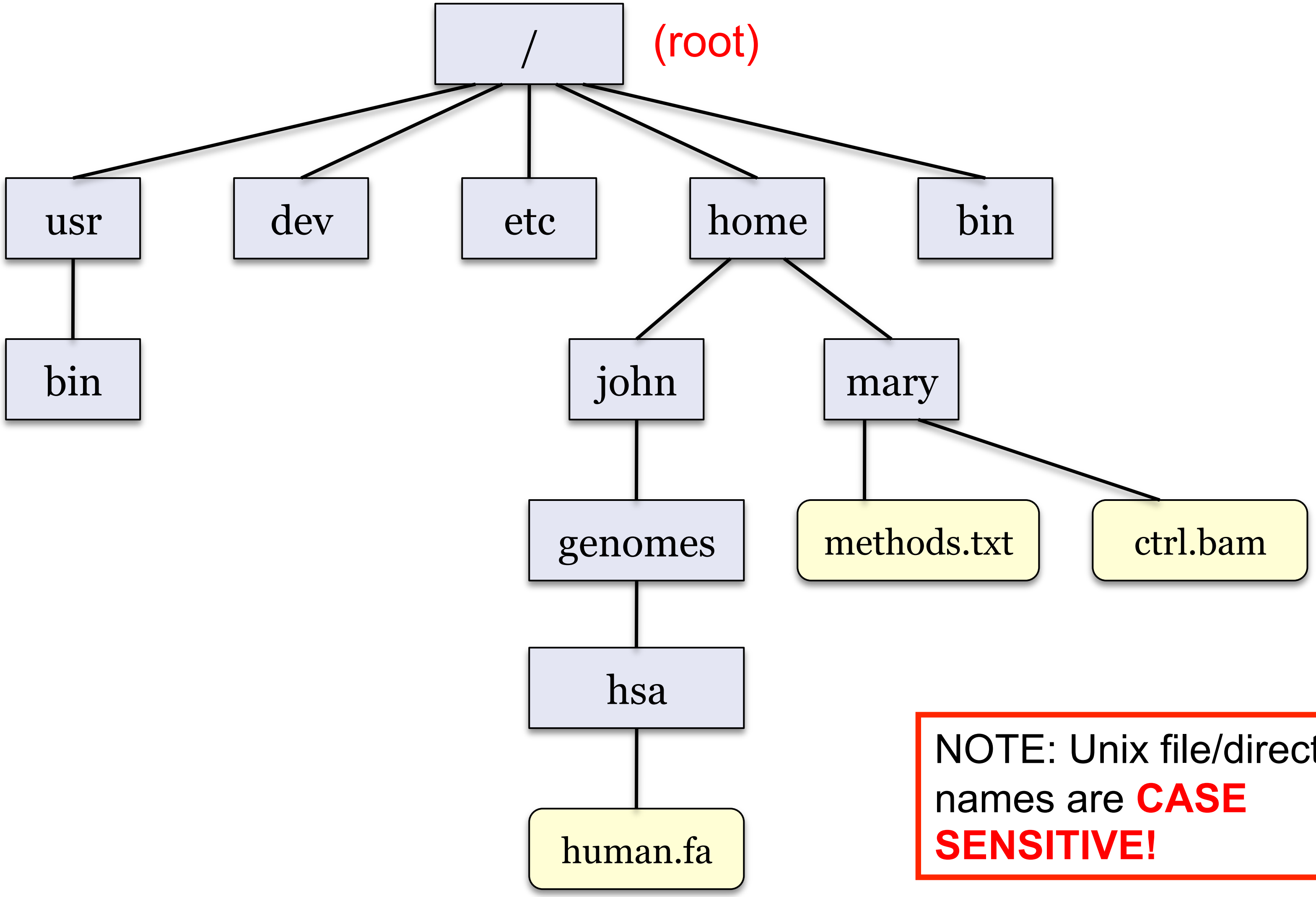
Where are we: **pwd**

- To find your current path use pwd - **p**rint **w**orking **d**irectory



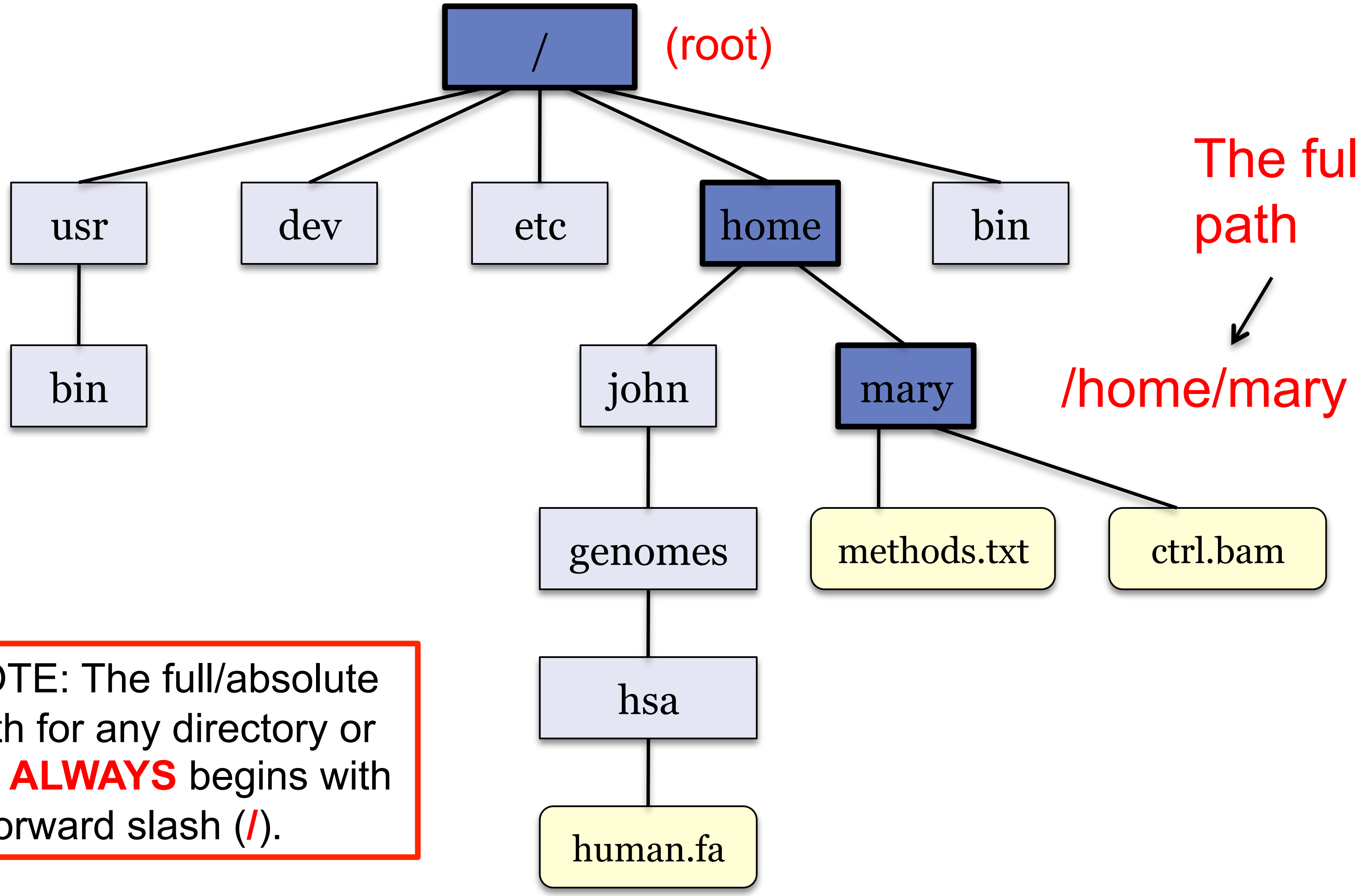
```
trainee@perth-ngs-04:~$ pwd
/home/trainee
trainee@perth-ngs-04:~$ 
```

Unix File System Tree Structure



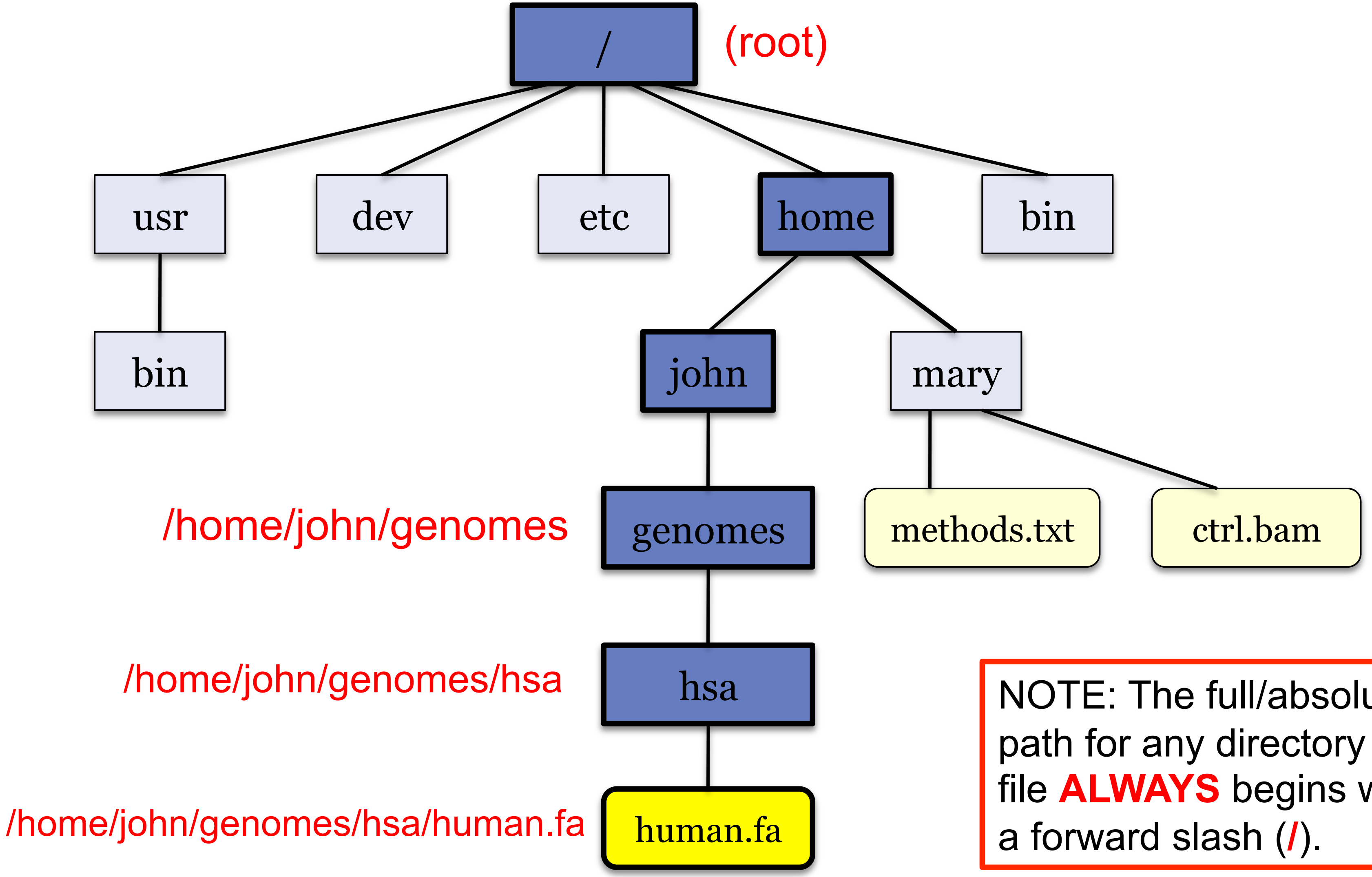
NOTE: Unix file/directory names are **CASE SENSITIVE!**

Full/Absolute File Paths



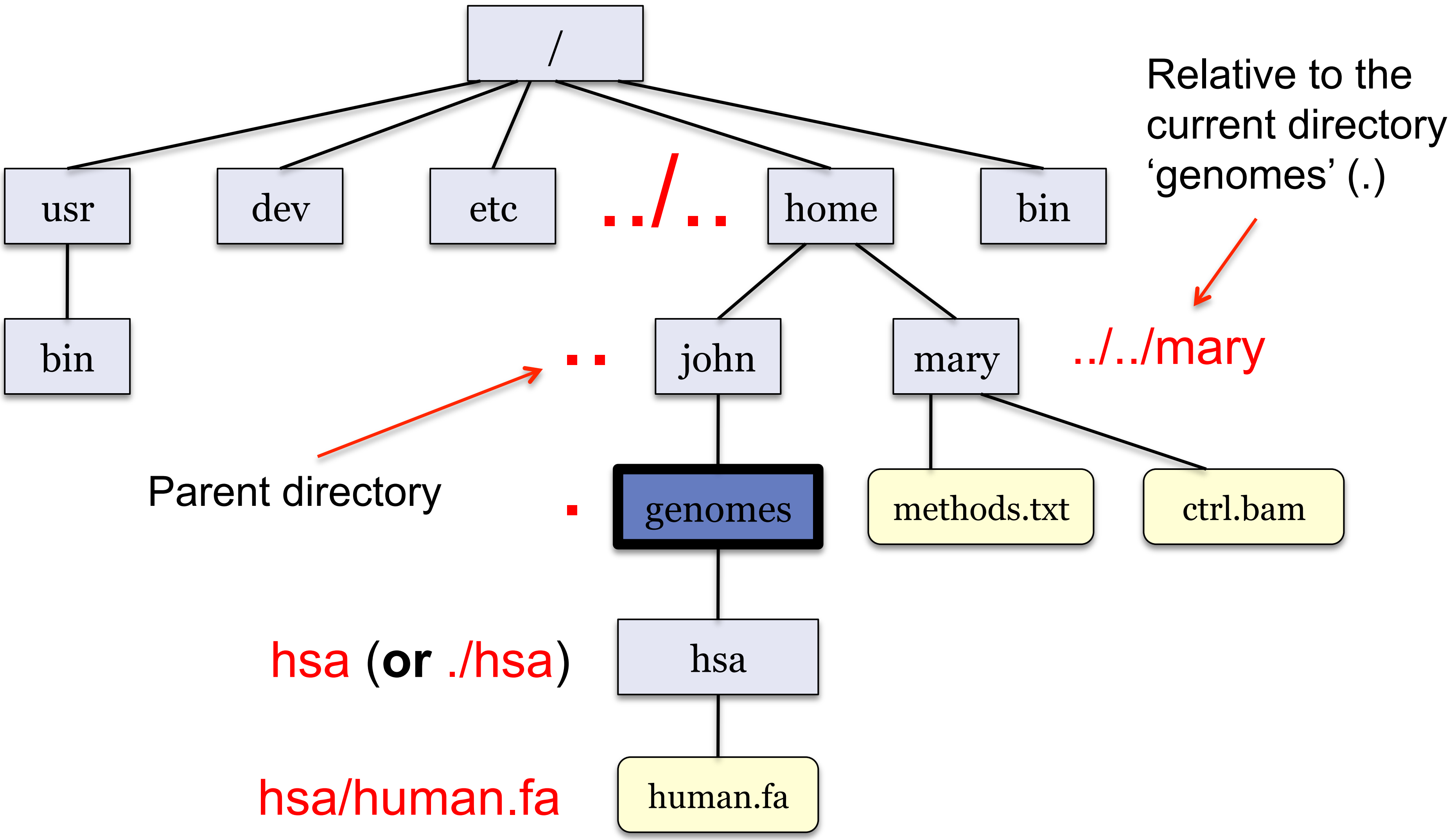
NOTE: The full/absolute path for any directory or file **ALWAYS** begins with a forward slash (/).

Full/Absolute File Paths



NOTE: The full/absolute path for any directory or file **ALWAYS** begins with a forward slash (**/**).

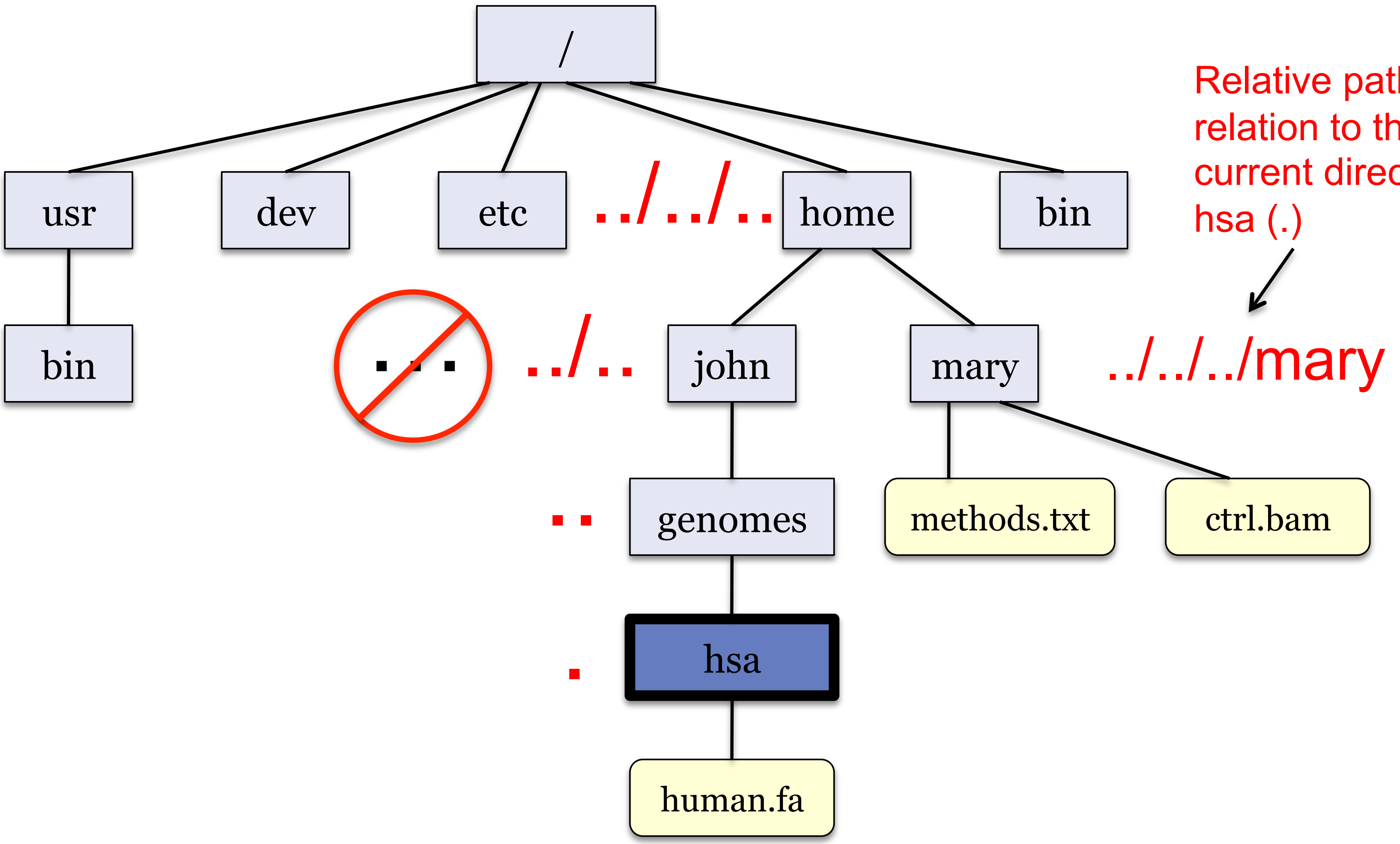
Relative File Paths



Relative to the current directory ‘genomes’ (.)

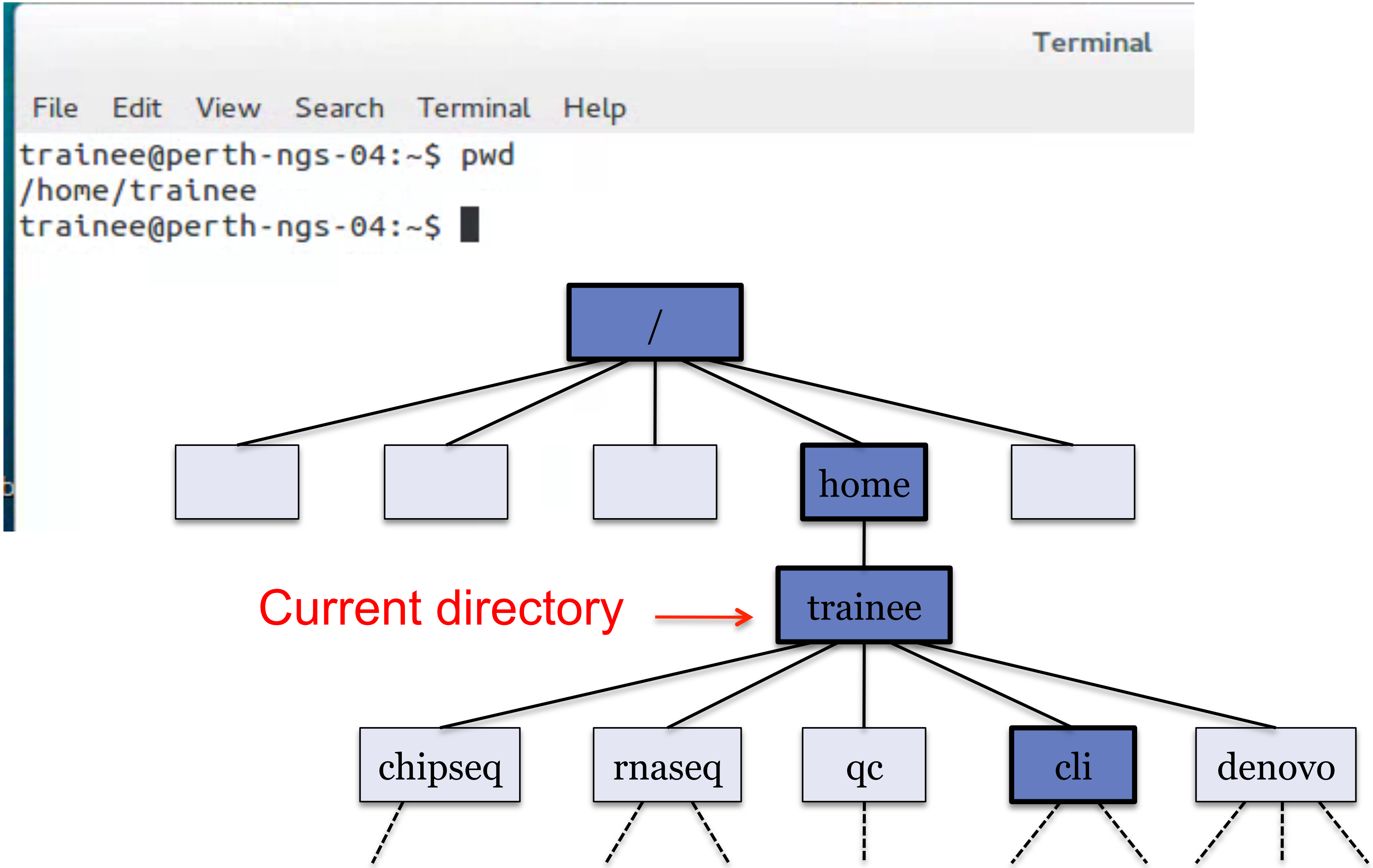
Parent directory

Relative File Paths



Relative path in relation to the current directory hsa (.)

Where am I?: pwd



```
trainee@perth-ngs-04:~$ pwd
/home/trainee
trainee@perth-ngs-04:~$ █
```

Current directory →

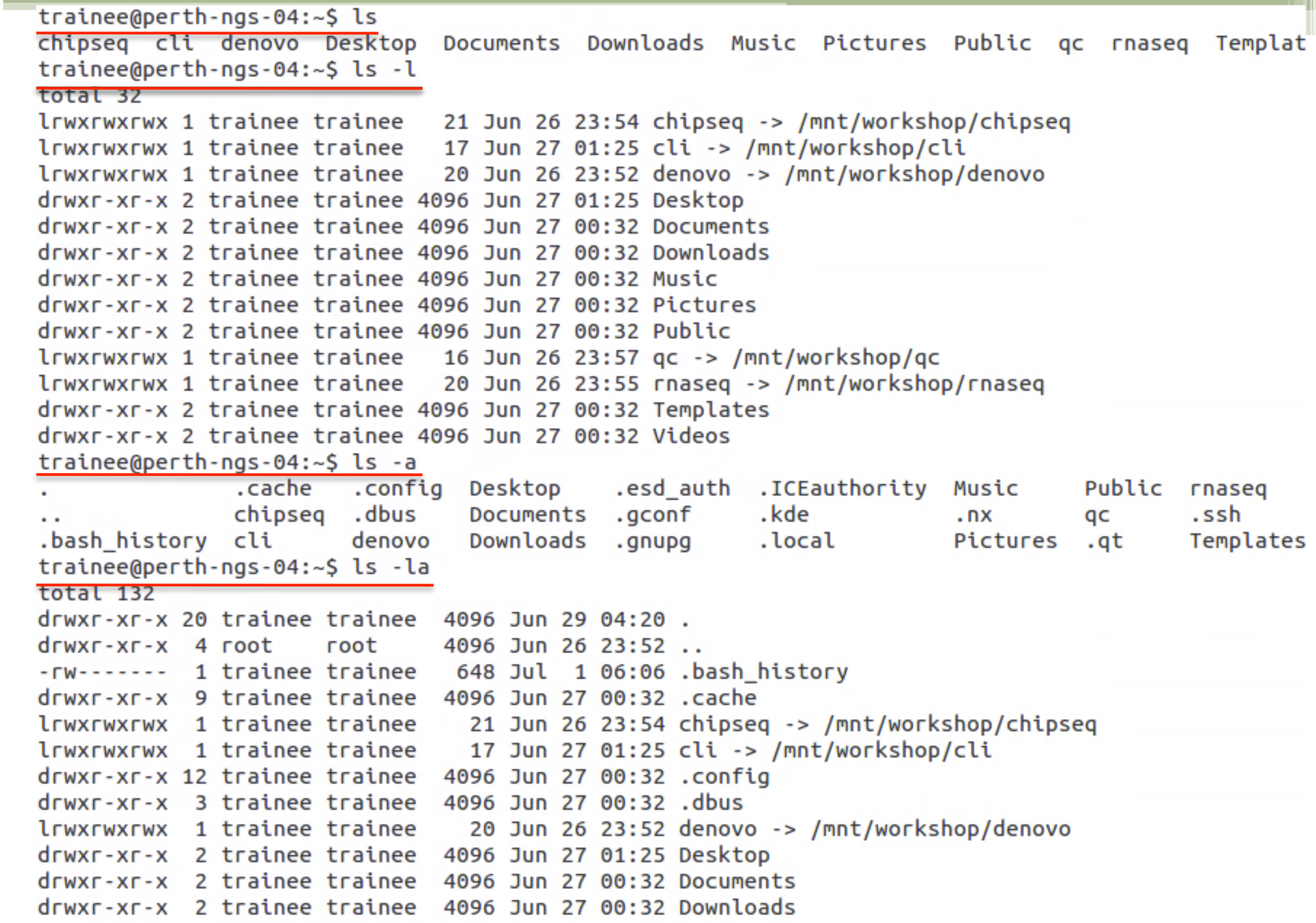
View Files: ls

- To view all the files in your current directory use “ls”

- Unix flags modify default behaviour. E.g.

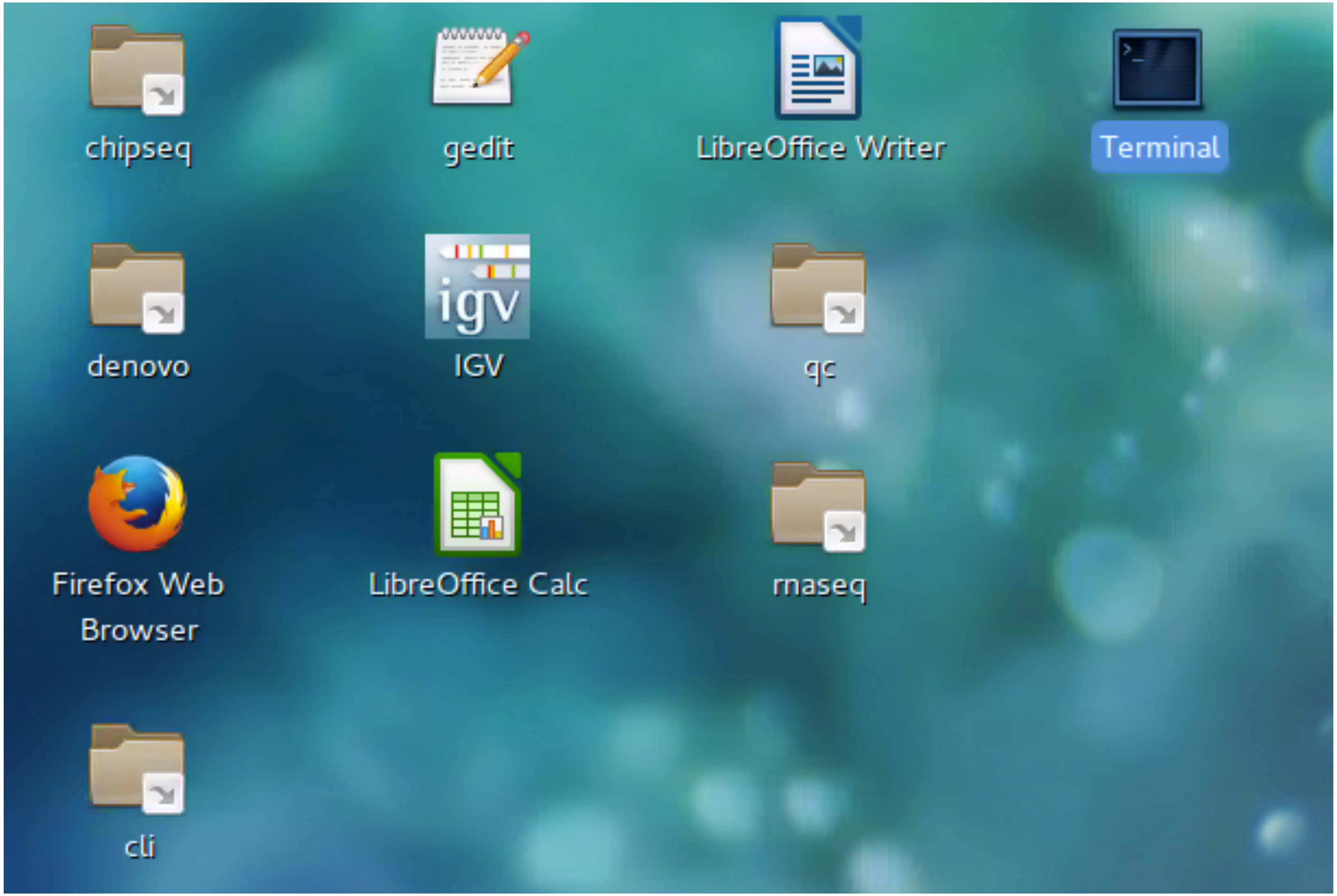
```
$ ls -l
$ ls -a
$ ls -la
```

  - -l flag gives you long list
  - -a stands for "show all”
  - -la combines both behaviours



```
trainee@perth-ngs-04:~$ ls
chipseq  cli  denovo  Desktop  Documents  Downloads  Music  Pictures  Public  qc  rnaseq  Templat
trainee@perth-ngs-04:~$ ls -l
total 32
lrwxrwxrwx 1 trainee trainee   21 Jun 26 23:54 chipseq -> /mnt/workshop/chipseq
lrwxrwxrwx 1 trainee trainee   17 Jun 27 01:25 cli -> /mnt/workshop/cli
lrwxrwxrwx 1 trainee trainee   20 Jun 26 23:52 denovo -> /mnt/workshop/denovo
drwxr-xr-x 2 trainee trainee 4096 Jun 27 01:25 Desktop
drwxr-xr-x 2 trainee trainee 4096 Jun 27 00:32 Documents
drwxr-xr-x 2 trainee trainee 4096 Jun 27 00:32 Downloads
drwxr-xr-x 2 trainee trainee 4096 Jun 27 00:32 Music
drwxr-xr-x 2 trainee trainee 4096 Jun 27 00:32 Pictures
drwxr-xr-x 2 trainee trainee 4096 Jun 27 00:32 Public
lrwxrwxrwx 1 trainee trainee   16 Jun 26 23:57 qc -> /mnt/workshop/qc
lrwxrwxrwx 1 trainee trainee   20 Jun 26 23:55 rnaseq -> /mnt/workshop/rnaseq
drwxr-xr-x 2 trainee trainee 4096 Jun 27 00:32 Templates
drwxr-xr-x 2 trainee trainee 4096 Jun 27 00:32 Videos
trainee@perth-ngs-04:~$ ls -a
.              .cache   .config  Desktop    .esd_auth  .ICEauthority  Music     Public  rnaseq
..             chipseq  .dbus    Documents  .gconf     .kde           .nx       qc      .ssh
.bash_history  cli      denovo   Downloads  .gnupg     .local         Pictures  .qt     Templates
trainee@perth-ngs-04:~$ ls -la
total 132
drwxr-xr-x 20 trainee trainee  4096 Jun 29 04:20 .
drwxr-xr-x  4 root    root     4096 Jun 26 23:52 ..
-rw-------  1 trainee trainee   648 Jul  1 06:06 .bash_history
drwxr-xr-x  9 trainee trainee  4096 Jun 27 00:32 .cache
lrwxrwxrwx  1 trainee trainee    21 Jun 26 23:54 chipseq -> /mnt/workshop/chipseq
lrwxrwxrwx  1 trainee trainee    17 Jun 27 01:25 cli -> /mnt/workshop/cli
drwxr-xr-x 12 trainee trainee  4096 Jun 27 00:32 .config
drwxr-xr-x  3 trainee trainee  4096 Jun 27 00:32 .dbus
lrwxrwxrwx  1 trainee trainee    20 Jun 26 23:52 denovo -> /mnt/workshop/denovo
drwxr-xr-x  2 trainee trainee  4096 Jun 27 01:25 Desktop
drwxr-xr-x  2 trainee trainee  4096 Jun 27 00:32 Documents
drwxr-xr-x  2 trainee trainee  4096 Jun 27 00:32 Downloads
```

View the Files Another Way



Changing Directories: cd

-
  ```
  $ pwd
  /home/trainee
  ```

- cd = "change directory”

- cd does NOT print anything, but if we run pwd after it, we can see where we are:

  ```
  $ cd cli        or        $ cd /home/trainee/cli
  $ pwd
  /home/trainee/cli
  ```

Trick: type ‘cd c’ and hit <TAB>; this is autocomplete

Creating and Deleting Things

- **Objectives**
  - Create a directory
  - Create files using an editor or by copying and renaming existing files.
  - Display the contents of a directory using the command line.
  - Delete specified files and/or directories.

Creating Things

- Create a new directory called ‘thesis’ using the command mkdir (the command has no output):

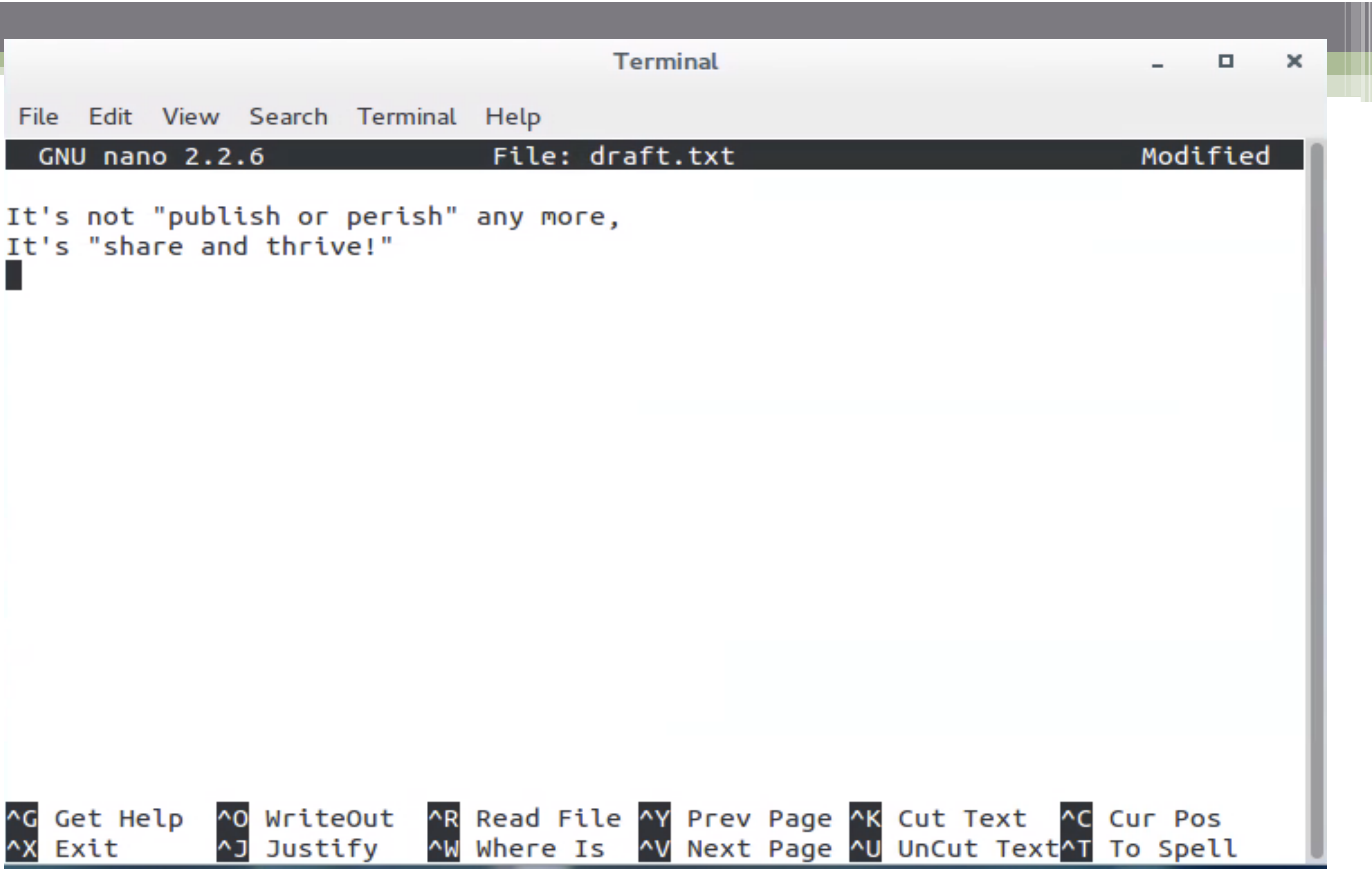
```
$ mkdir thesis
```

- However, there's nothing in it yet:

```
$ ls -F thesis
```

- Change the working directory to thesis using cd, then run a text editor called nano to create a file called draft.txt

```
$ cd thesis
$ nano draft.txt
```



```
  GNU nano 2.2.6              File: draft.txt                          Modified

It's not "publish or perish" any more,
It's "share and thrive!"

^G Get Help    ^O WriteOut    ^R Read File   ^Y Prev Page   ^K Cut Text    ^C Cur Pos
^X Exit        ^J Justify     ^W Where Is    ^V Next Page   ^U UnCut Text^T To Spell
```

Note: Use control-X to exit the editor and return to the shell.
Unix documentation often uses the shorthand “^A” to mean “control-A”.

Displaying the contents of a file

- There’s no output from nano on the screen after it exits, but ls shows that we have created a file called draft.txt.

  ```
  $ ls
  draft.txt
  ```

- The commands cat, less and more can all be used to display/ view the contents of the file draft.txt. cat displays the entire files's contents, while more and less display the file’s contents one page at a time.

  ```
  $ cat draft.txt
  It's not "publish or perish" any more,
  It's "share and thrive!"
  ```

Moving and Copying

- Rename draft.txt to quotes.txt using the command mv.

```
$ mv draft.txt quotes.txt
$ ls
quotes.txt
```

- The “mv” command can be also be used to move files to other locations in the file system e.g. to the directory cli

```
$ pwd
/home/trainee/cli/thesis
$ mv quotes.txt ..        (or $ mv quotes.txt /home/trainee/cli)
$ ls
$ ls ..
1000gp.vcf  molecules  quotes.txt  thesis
```

- **C**hange **d**irectory to ‘cli’

```
$ cd ..  (or $ cd /home/trainee/cli)
```

Moving and Copying

- cp works very much like mv, except it copies a file instead of moving it:

```
$ cp quotes.txt quotations.txt
$ ls
1000gp.vcf  molecules  quotations.txt  quotes.txt  thesis
```

- The “cp” command can also be used to copy the file to other locations in the file system:

```
$ cp quotes.txt thesis  (or $ cp quotes.txt /home/trainee/cli/thesis)
$ ls thesis
quotes.txt
```

Warning: By default the cp and mv commands will overwrite a destination file if it already exists. This means you will lose the old contents of the destination file!

Removing files

- Let’s clean up and remove the file quotations.txt using the command rm

```
$ rm quotations.txt
$ ls
1000gp.vcf  molecules  quotes.txt  thesis
```

- If we try to remove the entire thesis directory using “rm thesis”, we get an error message:

```
$ rm thesis
rm: cannot remove `thesis/': Is a directory
```

NOTE: rm works on files, not directories.

Deleting directories

- rmdir is the command to remove a directory.

```
$ rmdir thesis
rmdir: failed to remove `thesis': Directory not empty
```

- To delete the directory ‘thesis’, it must be empty. The file ‘quotes.txt’ inside of it must be deleted first. This safety feature can save you a lot of grief!

```
$ rm thesis/quotes.txt
$ ls thesis
```

- The directory is now empty, so rmdir can delete it:

```
$ rmdir thesis
```

Deleting is Forever!

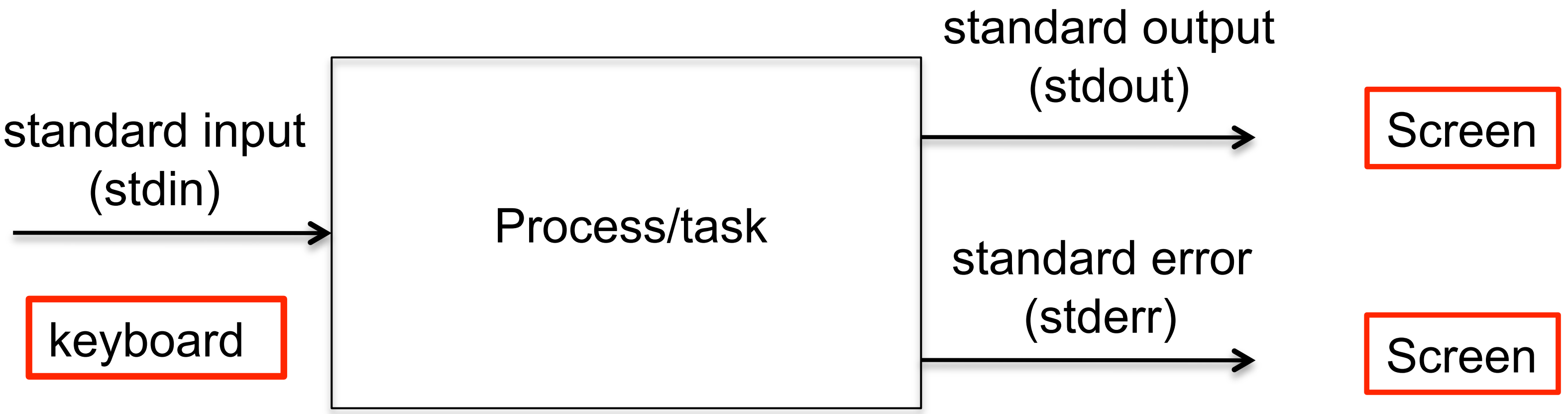
Unix does **NOT** have a trash bin.

Pipes and Filters

- **Objectives**

  - Redirect a command's output to a file.
  - Process a file instead of keyboard input using redirection.
  - Construct command pipelines with two or more stages

Processes

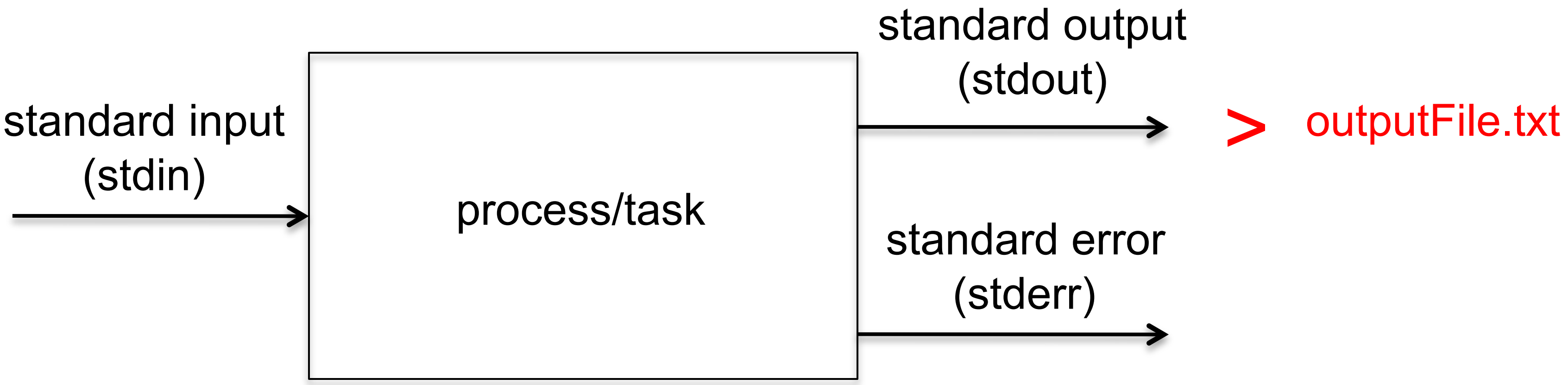


```
$ head -10 inputFile.txt
```

Execute task →

The first 10 lines of inputFile.txt will be outputted to the screen.

Redirecting standard output (> or >>)

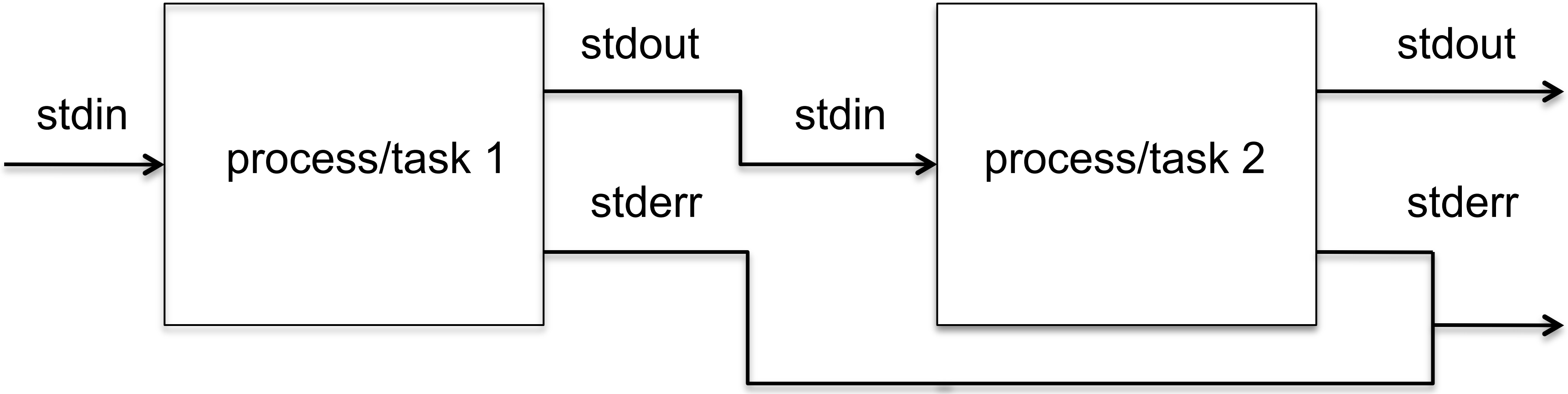


```
$ head -10 inputFile.txt > outputFile.txt
```

If the file outputFile.txt already exists, using '>' will overwrite the file.

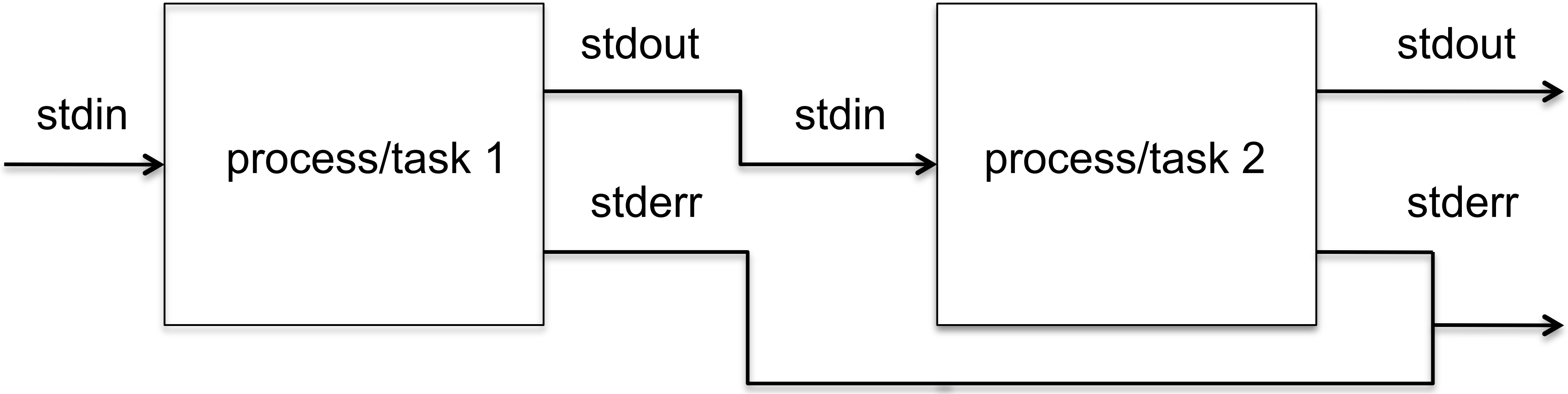
Using '>>' will append the output onto the end of the file.

Pipes - |



This connection of processes is achieved using a pipe (|).

Pipes - |



head -10 inputFile.txt | sort -n

i.e. Get the first 10 lines of inputFile.txt and numerically sort the output.

```
$ head -10 inputFile.txt | sort -n
$ ls molecules
cubane.pdb    ethane.pdb    methane.pdb
octane.pdb    pentane.pdb   propane.pdb

$ cd molecules

$ wc *.pdb
       20   156 1158 cubane.pdb
       12    84  622 ethane.pdb
        9    57  422 methane.pdb
       30   246 1828 octane.pdb
       21   165 1226 pentane.pdb
       15   111  825 propane.pdb
      107   819 6081 total
```

Wildcards

- * is a wildcard. It matches zero or more characters, so *.pdb matches anything ending in .pdb

```
cubane.pdb
ethane.pdb
methane.pdb
octane.pdb
pentane.pdb
propane.pdb
```

- ? is also a wildcard, but it only matches a single character. This means that p?.pdb matches pi.pdb or p5.pdb, but not propane.pdb.

- Show only the number of lines per file (wc -l)

```
$ wc -l *.pdb
      20  cubane.pdb
      12  ethane.pdb
       9  methane.pdb
      30  octane.pdb
      21  pentane.pdb
      15  propane.pdb
     107  Total
$ wc -l *.pdb > lengths.txt
$ ls lengths.txt
lengths.txt

$ cat lengths.txt
     20  cubane.pdb
     12  ethane.pdb
      9  methane.pdb
     30  octane.pdb
     21  pentane.pdb
     15  propane.pdb
    107  total
```

Sorting and piping

```
$ sort -n lengths.txt
  9  methane.pdb
 12  ethane.pdb
 15  propane.pdb
 20  cubane.pdb
 21  pentane.pdb
 30  octane.pdb
107  Total

$ sort -n lengths.txt > sorted_lengths.txt

$ head -1 sorted_lengths.txt
  9  methane.pdb

$ sort -n lengths.txt | head -1
  9  methane.pdb

$ wc -l *.pdb | sort -n | head -1
  9  methane.pdb
```

Finding a pattern with ‘grep’

- Use grep (*__g__lobally search a __r__egular __e__xpression* and *__p__rint*) to select lines from text files that match simple patterns.

```
$  grep "tane" lengths.txt
   30 octane.pdb
   21  pentane.pdb
```

- Option -n, which numbers the lines that match

```
$ grep –n "tane" lengths.txt
   4:     30 octane.pdb
   5:     21 pentane.pdb
```

- grep -i makes matching case-insensitive
- grep -v inverts the match e.g. grep -v "tane" lengths.txt will find everything EXCEPT octane.pdb and pentane.pdb
- Options can be combined, such as: -iv

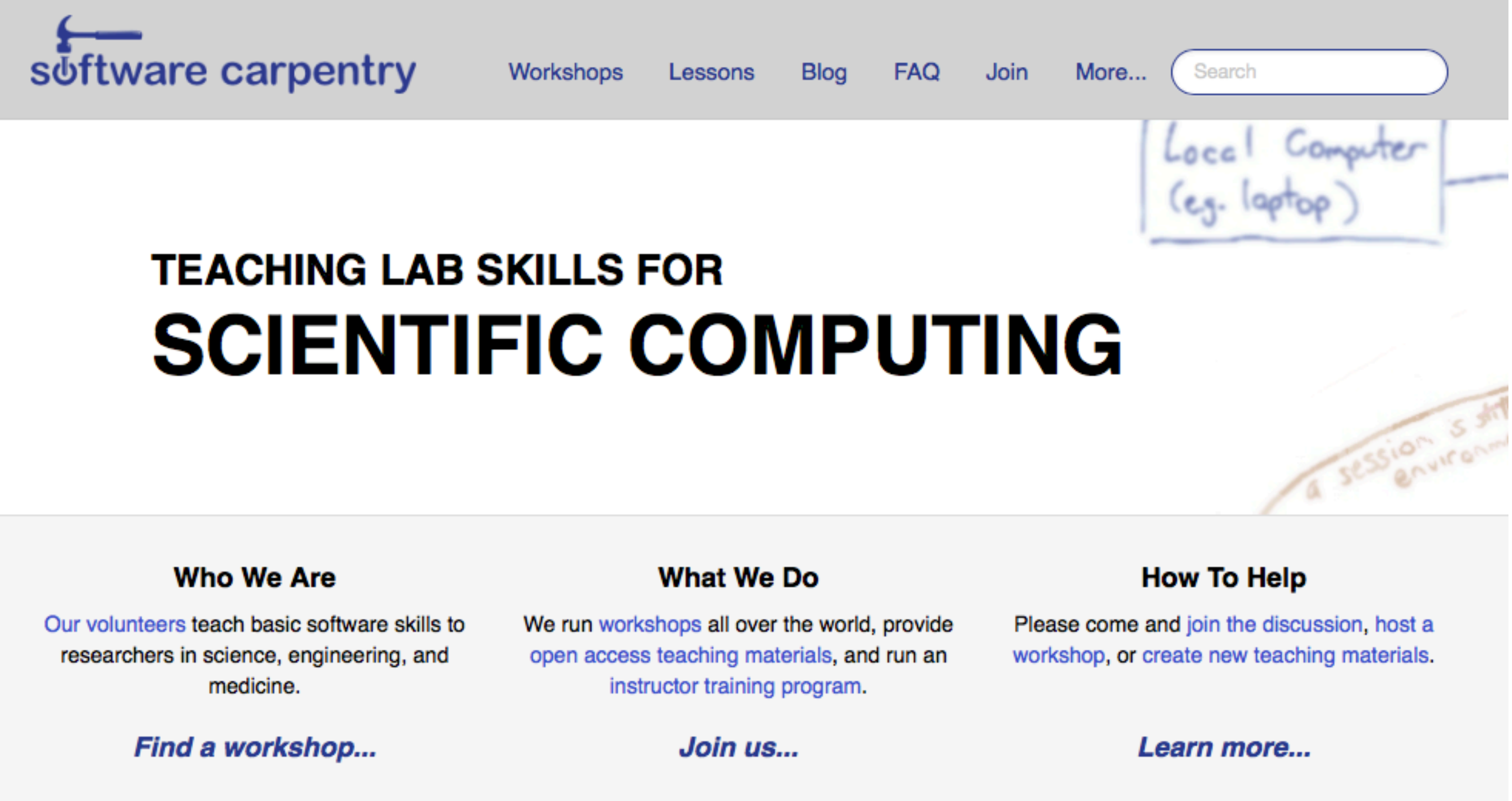
Some other useful tips

- <Tab> to autocomplete (useful to avoid errors)
- Re-run last command: **!!**
- List previous commands: history
- !(command number) to re-run a command e.g. !21
- Up and down arrows to scroll through previous commands -> edit command line
- Go directly to your home directory: cd
- Ctrl-C to terminate a command
- Clear terminal window: clear

- Yes, you can copy and paste

Useful Resource

- http://software-carpentry.org



TEACHING LAB SKILLS FOR

SCIENTIFIC COMPUTING

Who We Are

Our volunteers teach basic software skills to researchers in science, engineering, and medicine.

Find a workshop...

What We Do

We run workshops all over the world, provide open access teaching materials, and run an instructor training program.

Join us...

How To Help

Please come and join the discussion, host a workshop, or create new teaching materials.

Learn more...

Finding files

- “grep” finds lines in files, the “find” command finds files themselves
- `$ cd ..`
- `$ find . -type d`   Finding a directory

```
./
./molecules
```

- `$ find . -type f`   Finding a file

```
./molecules/cubane.pdb
./molecules/ethane.pdb
./molecules/lengths.txt
./molecules/methane.pdb
./molecules/octane.pdb
./molecules/pentane.pdb
./molecules/propane.pdb
```

Finding a file by name

```
$ find . -name *.pdb
  ./molecules/cubane.pdb
  ./molecules/ethane.pdb
  ./molecules/methane.pdb
  ./molecules/octane.pdb
  ./molecules/pentane.pdb
  ./molecules/propane.pdb

$ find . -name cubane.pdb
  ./molecules/cubane.pdb
```

Listing vs. Finding

`ls` and `find` can be made to do similar things given the right options, but under normal circumstances, `ls` lists everything it can, while `find` searches for things with certain properties and shows them.