Linux for Beginners

3CPG Workshop

Robert Bukowski
Computational Biology Service Unit

http://cbsu.tc.cornell.edu/lab/doc/linux_workshop.pdf

Topics

- Reserving time on lab workstations
- Logging in to a workstation
- Terminal window and tricks
- Directory structure
- Working with files
- Transferring files to/from workstations
- Running applications
- Basics of shell scripting
- Trying it

Reserving time

- From any computer on the network, go to http://cbsu.tc.cornell.edu.
- Click on **CBSU/3CPG BioHPC Lab** tab -> **Reservations**
  - Also check out other links under this tab, like **Software, User's Guide, Forum**
- Log in with your user name (typically same as NetID) and the lab password
  - Initial lab password is sent to you in a "welcome e-mail" upon account creation
- Make a reservation
  - Note that only cbsuwrkst2, cbsuwrkst3, and cbsuwrkst4 are Linux machines, cbsuwrkst1 is running Windows.
  - If your reservation ends while you are still logged in, you may continue working. However, you will be disconnected once another user with valid reservation attempts to log in. To prevent this, go to the reservation page and extend the reservation, if possible.
  - Cancel reservations you no longer need (click on "X" near your reservation)

Logging in

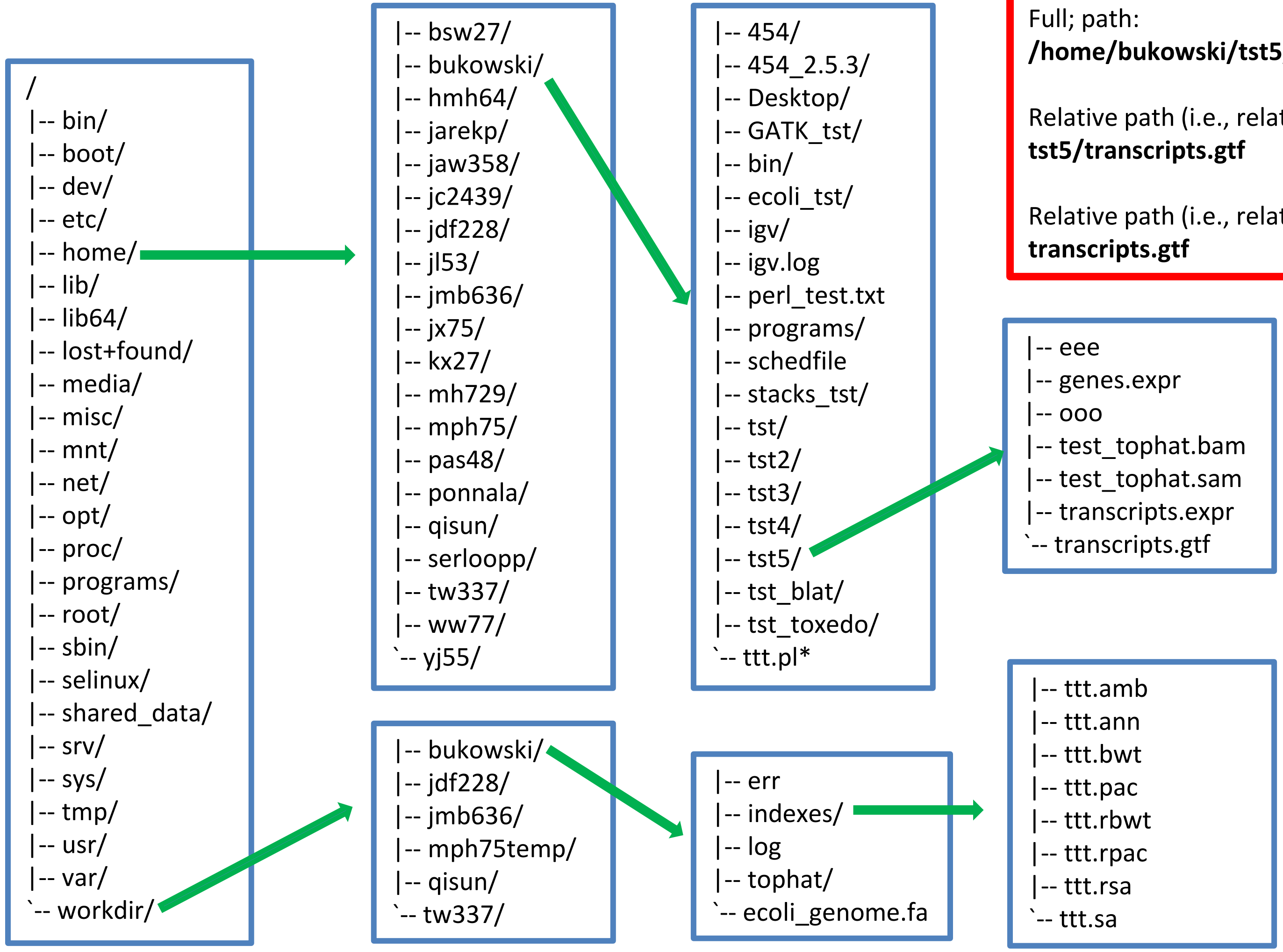
- From the console in 625 Rhodes
  - Turn on the monitor, then log in using the login screen
  - Right-click anywhere on the desktop, then select “Open terminal” – a terminal window will open
  - You can open additional terminals window the same way
- Remotely from a PC or Mac
  - Remote connection is through **ssh** (secure shell) – need a client
  - Install and configure remote access software. For details, consult http://cbsu.tc.cornell.edu/lab/doc/Remote_access.pdf
  - On Windows, use **PuTTy** to open a terminal window using ssh protocol; start **Xming** if you intend to run any graphical applications on a workstation (iAssembler, Firefox, IGV,…)
  - On Mac, Launch the Mac’s terminal window. Type **ssh cbsuwrkstX.tc.cornell.edu** (replace the “X” with the workstation that you just reserved). Enter the lab user name and password when prompted.
  - You may open several terminal windows, if needed.

Terminal window

- User communicates with the machine via **commands** typed in the terminal window
  - Typically, each command is typed in one line and “entered” by hitting the “Enter” key on the keyboard.
  - Commands may request information (e.g., list user’s files), launch a simple task (e.g., rename a file), or start an application (e.g., Firefox web browser, BWA aligner, IGV viewer, …)
- Helpful tricks to avoid excessive command typing
  - Use copy/paste. Any text “mouse-selected” while holding the left mouse button is copied to clipboard. It may then be pasted, e.g., into a command, by clicking the right mouse button.
  - Use Up/Down Arrow keys – this will cycle through recently executed commands.
  - Use the TAB key – this will often present you with a list of choices after typing a part of a command – no need to remember everything.

Directory structure

(example)



```
/
|-- bin/
|-- boot/
|-- dev/
|-- etc/
|-- home/
|-- lib/
|-- lib64/
|-- lost+found/
|-- media/
|-- misc/
|-- mnt/
|-- net/
|-- opt/
|-- proc/
|-- programs/
|-- root/
|-- sbin/
|-- selinux/
|-- shared_data/
|-- srv/
|-- sys/
|-- tmp/
|-- usr/
|-- var/
`-- workdir/
```

```
|-- bsw27/
|-- bukowski/
|-- hmh64/
|-- jarekp/
|-- jaw358/
|-- jc2439/
|-- jdf228/
|-- jl53/
|-- jmb636/
|-- jx75/
|-- kx27/
|-- mh729/
|-- mph75/
|-- pas48/
|-- ponnala/
|-- qisun/
|-- serloopp/
|-- tw337/
|-- ww77/
`-- yj55/
```

```
|-- 454/
|-- 454_2.5.3/
|-- Desktop/
|-- GATK_tst/
|-- bin/
|-- ecoli_tst/
|-- igv/
|-- igv.log
|-- perl_test.txt
|-- programs/
|-- schedfile
|-- stacks_tst/
|-- tst/
|-- tst2/
|-- tst3/
|-- tst4/
|-- tst5/
|-- tst_blat/
|-- tst_toxedo/
`-- ttt.pl*
```

```
|-- eee
|-- genes.expr
|-- ooo
|-- test_tophat.bam
|-- test_tophat.sam
|-- transcripts.expr
`-- transcripts.gtf
```

```
|-- bukowski/
|-- jdf228/
|-- jmb636/
|-- mph75temp/
|-- qisun/
`-- tw337/
```

```
|-- err
|-- indexes/
|-- log
|-- tophat/
`-- ecoli_genome.fa
```

```
|-- ttt.amb
|-- ttt.ann
|-- ttt.bwt
|-- ttt.pac
|-- ttt.rbwt
|-- ttt.rpac
|-- ttt.rsa
`-- ttt.sa
```

Referring to files:

Full; path:
/home/bukowski/tst5/transcripts.gtf

Relative path (i.e., relative to /home/bukowski)
tst5/transcripts.gtf

Relative path (i.e., relative to /home/bukowski/tst5)
transcripts.gtf

Working with Directories

Right after logging in or opening a terminal window, you are in your **home directory** (e.g., /home/bukowski).

pwd
*(**p**rint **w**orking **d**irectory) – show the current directory*

cd
C**hange (current) **d**irectory; without additional arguments, this command will take you to your **home directory

cd ./
***C**hange (current) **d**irectory to the same one (i.e., do nothing). Note: **./** refers to the current directory.*

cd /workdir/bukowski/indexes
***C**hange (current) **d**irectory from wherever to /workdir/bukowski/indexes.*

cd indexes
***C**hange (current) **d**irectory to indexes (will work if the current directory contains "indexes")*

cd ../
***C**hange (current) **d**irectory one level back (closer to the root)*

cd ../../../
***C**hange (current) **d**irectory three levels back (closer to the root)*

mkdir /home/bukowski/my_new_dir
***M**ake a new **dir**ectory called "my_new_dir" in /home/bukowski*

mkdir my_new_dir
***M**ake a new **dir**ectory called "my_new_dir" in the current directory*

rm –Rf /home/bukowski/my_new_dir
***Rem**ove directory called "my_new_dir" in /home/bukowski with all its content (i.e. all files and subdirectories will be gone)*

rm –Rf my_new_dir
***Rem**ove directory called "my_new_dir" in current directory with all its content (i.e. all files and subdirectories will be gone)*

Listing content (files and subdirectories) of a directory

ls

(list)

ls –al

List all files and directories in current directory in long format

ls –al /home/bukowski/tst

List content of /home/bukowski/tst (which does not have to be the current directory)

ls –alt

Lists content of the current directory sorted according to modification time (use **ls –altr** *to sort I reverse)*

ls –alS

Lists content of the current directory sorted according to size (use **ls –alSr** *to sort in reverse)*

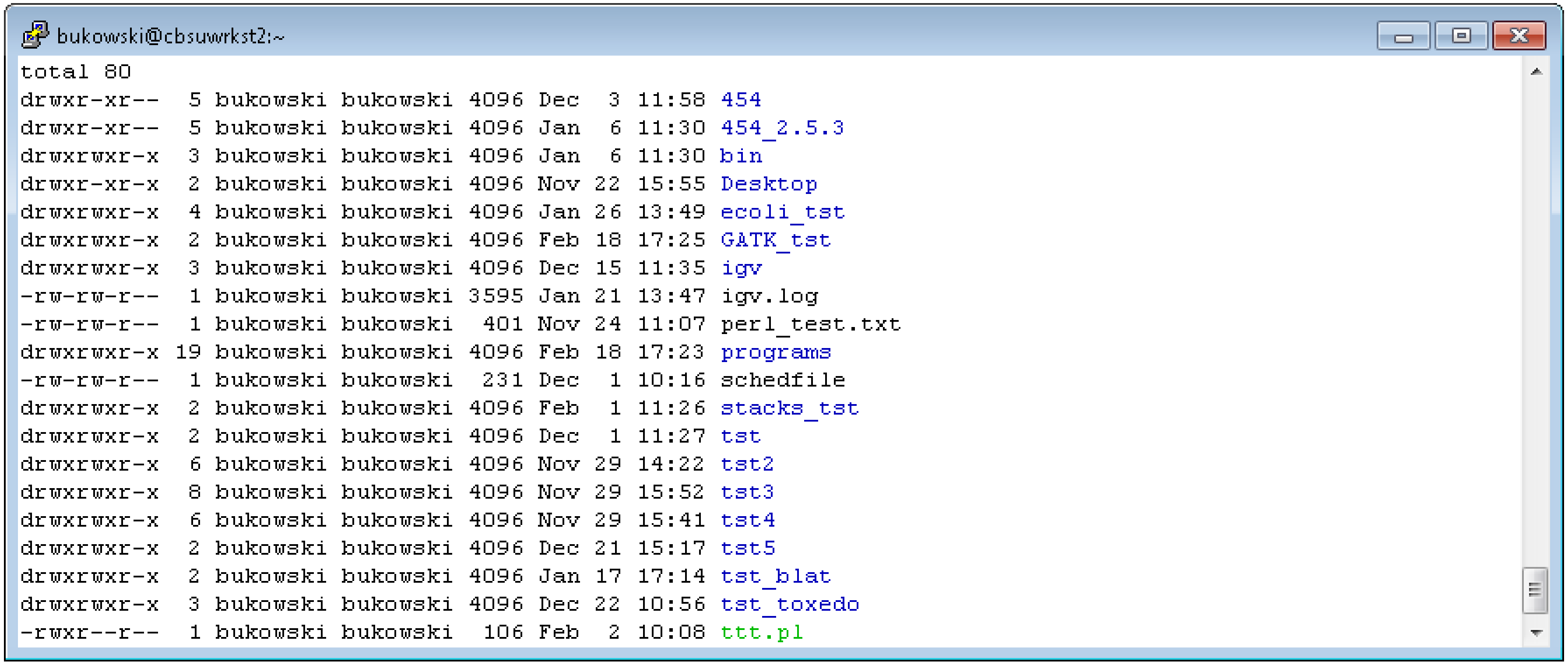
ls –al | more

Lists content of the current directory using pagination – useful if the file list is long (SPACE bar will take you to the next page)

LOTS more options for ls – try **man ls** to see them all (may be intimidating).

Listing content of a directory

ls -al



```
total 80
drwxr-xr--  5 bukowski bukowski 4096 Dec  3 11:58 454
drwxr-xr--  5 bukowski bukowski 4096 Jan  6 11:30 454_2.5.3
drwxrwxr-x  3 bukowski bukowski 4096 Jan  6 11:30 bin
drwxr-xr-x  2 bukowski bukowski 4096 Nov 22 15:55 Desktop
drwxrwxr-x  4 bukowski bukowski 4096 Jan 26 13:49 ecoli_tst
drwxrwxr-x  2 bukowski bukowski 4096 Feb 18 17:25 GATK_tst
drwxrwxr-x  3 bukowski bukowski 4096 Dec 15 11:35 igv
-rw-rw-r--  1 bukowski bukowski 3595 Jan 21 13:47 igv.log
-rw-rw-r--  1 bukowski bukowski  401 Nov 24 11:07 perl_test.txt
drwxrwxr-x 19 bukowski bukowski 4096 Feb 18 17:23 programs
-rw-rw-r--  1 bukowski bukowski  231 Dec  1 10:16 schedfile
drwxrwxr-x  2 bukowski bukowski 4096 Feb  1 11:26 stacks_tst
drwxrwxr-x  2 bukowski bukowski 4096 Dec  1 11:27 tst
drwxrwxr-x  6 bukowski bukowski 4096 Nov 29 14:22 tst2
drwxrwxr-x  8 bukowski bukowski 4096 Nov 29 15:52 tst3
drwxrwxr-x  6 bukowski bukowski 4096 Nov 29 15:41 tst4
drwxrwxr-x  2 bukowski bukowski 4096 Dec 21 15:17 tst5
drwxrwxr-x  2 bukowski bukowski 4096 Jan 17 17:14 tst_blat
drwxrwxr-x  3 bukowski bukowski 4096 Dec 22 10:56 tst_toxedo
-rwxr--r--  1 bukowski bukowski  106 Feb  2 10:08 ttt.pl
```

File permissions ("d" means this is a directory)

Owner and group

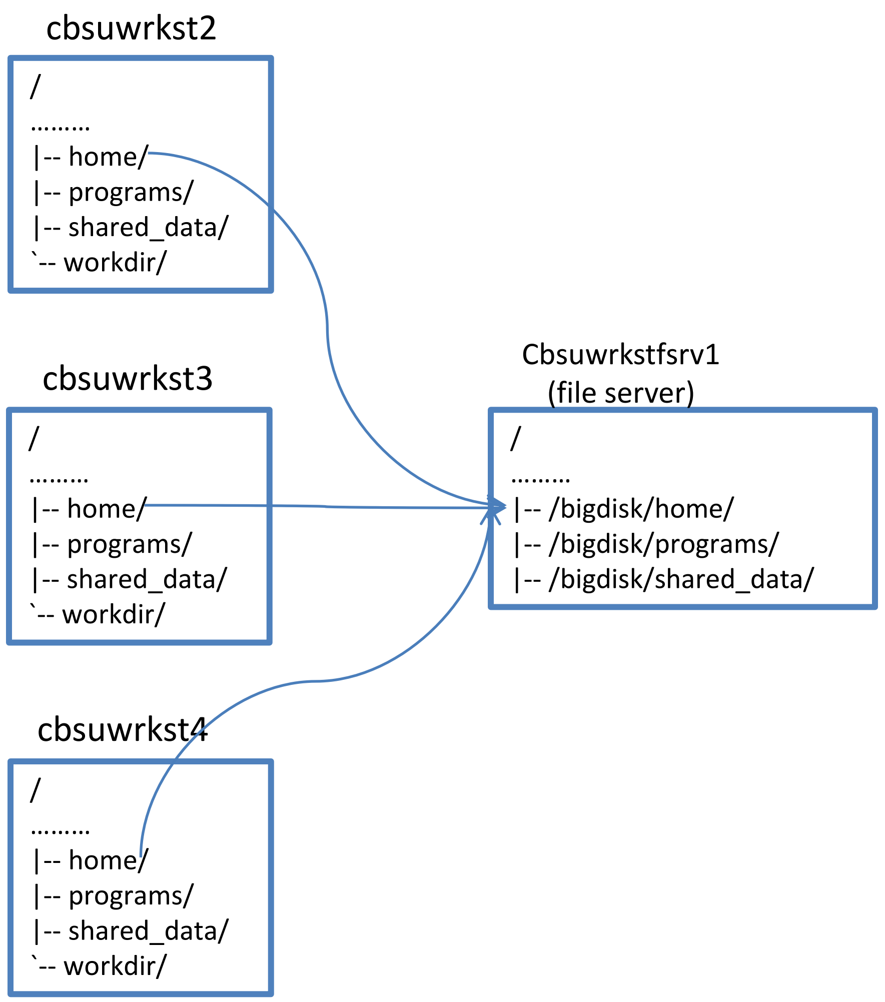
Last modification time

File name (directories in blue, executable files in green)

Size (in bytes)

Local vs. network directories

(3CPG LAB – specific)



<u>Network directories</u>

/home, /programs, /shared_data (with all subdirectories)

- Physically located on the file server
- Visible from all workstations
- Relatively SLOW access – DO NOT run any calculations there, avoid transferring large files there

<u>Local directories:</u>

/workdir (with all subdirectories), all other directories

- Physically attached to "its own" workstation
- Not visible from other workstations
- Fast access – all calculations should be run in **/workdir**

Working with files

File names

- Use only letters (upper- and lower-case), numbers from 0 to 9, a dot (.), and an underscore (_).
- Avoid other characters, as they may have special meaning to either Linux, or to the application you are trying to run.
- Use of special characters in file names is possible if absolutely necessary, but will lead to problems if done incorrectly.
- Extensions (like ".zip", ".gz", ".ps",…) are commonly used to denote the type of file, but are not necessary to "open" a file. While working in command line terminal you always explicitly specify a program which is supposed to work with this file.
- The dot (.) does not have any special meaning in Linux file names.

Working with files

There are many types of files. Here are the most important:

- Binary files (cannot be viewed using a text editor)
    - Executables (e.g., samtools, bwa, bowtie)
    - Data in binary format (e.g, BAM files, index files for BWA or Bowtie, formatted BLAST databases)
    - Compressed files (usually *.gz, *.zip, *.bz2,…, but extensions not necessary)
- Text files (can be viewed and modified using a text editor)
    - Text documents (e.g., README files)
    - Data in text format (e.g., FASTA, FASTQ, VCF, …)
    - Scripts:
        - Shell scripts (usually *.sh or *.csh)
        - Perl scripts (usually *.pl)
        - Python scripts (usually *.py)
        - …
- Symbolic links: pointers to other files or directories. In the example below, file /programs/bin/samtools/samtools is a symbolic link to /programs/samtools-0.1.11/samtools. Note the “l” character in the first column of output from “ls –al”.

```
$ cd /programs/bin/samtools
$ ls –al samtools
lrwxrwxrwx  1 root root   30 Nov 23 13:44 samtools -> ../../samtools-0.1.11/samtools
```

Working with files

Copying a file

cp <source_file> <destination_file>

Examples:

- **cp sample_data.fa /workdir/bukowski/sample.fa** *(copy file sample_data.fa from the current directory to /workdir/bukowski and give the copy a name sample.fa; destination directory must exist)*
- **cp /workdir/bukowski/my_script.sh .** *(copy file myscript.sh from /workdir/bukowski to the current directory under the same file name)*
- **cp /home/bukowski/*.fastq /workdir/bukowski** *(copy all files with file names ending with ".fastq" from /home/bukowski to /workdir/bukowski; destination directory must exist)*
- **cp –R /workdir/bukowski/tst5 /home/bukowski** *(if tst5 is a directory, it will be copied with all its files and subdirectories to directory /home/bukowski/tst5; if /home/bukowski/tst5 did not exist, it will be created).*
- Try **man cp** for all options to the **cp** command.

Working with files

Moving and renaming files

mv <source_file> <destination_file>

Examples:

- **mv my_file_one my_file_two** *(change the name of the file my_file_one in the current directory)*
- **mv my_file_one /workdir/bukowski** *(move the file my_file_one from the current directory to /workdir/bukowski without changing file name; the file will be removed from the current directory)*
- **mv /workdir/bukowski/my_file_two ./my_file_three** *(move the file my_file_two from /workdir/bukowski to the current directory changing the name to my_file_three; the file will be removed from /workdir/bukowski)*
- Try **man mv** for all options to the **mv** command….

Removing (deleting) files

rm <file_name>

Examples:

- **rm my_file_one** *(delete file my_file_one from the current directory)*
- **rm /workdir/bukowski/my_file_two** *(delete file my_file_two from directory /workdir/bukowski)*
- **rm –R ./tst5** *(if tst5 is a subdirectory in the current directory, it will be removed with all its files and directories)*
- Try **man rm** for all options to the **rm** command….

Working with files

Since there are no strict naming conventions for various file types, it is not always clear what kind of file we deal with. When in doubt, use the **file** command:

```
cd  /programs/samtools-0.1.11
file samtools
samtools: ELF 64-bit LSB executable, AMD x86-64, version 1 (SYSV), for GNU/Linux 2.6.9,
dynamically linked (uses shared libs), for GNU/Linux 2.6.9, not stripped
```

Working with files

To save disk space, we can compress large files if we do not intend to use them for a while. A lot of files downloaded from the web are compressed and need to be uncompressed before any processing can take place.

Common compressed formats:

- gzip (gz)
  - **gzip my_file** *(compresses file* my_file*, producing its compressed version,* my_file.gz*)*
  - **gzip –d my_file.gz** *(decompress* my_file.gz*, producing its original version* my_file*)*
- bzip2
  - **bzip2 my_file** *(compresses file* my_file*, producing its compressed version,* my_file.bz2*)*
  - **bunzip2 my_file.bz2** *(decompress* my_file.bz2*, producing its original version* my_file*)*

Working with files

Common compressed formats (continued):

- zip
  - **zip my_file.zip my_file1 my_file2 my_file3**   *(create a compressed archive called* my_files.zip*, containing three files:* my_file1, my_file2, my_file3*)*
  - **zip   -r   my_file.zip my_file1 my_dir**   *(if* my_dir *is a directory, create an archive* my_file.zip *containing the file* my_file1 *and the directory* my_dir *with all its content)*
  - **zip –l   my_file.zip**   *(list contents of the zip archive* my_file.zip*)*
  - **unzip   my_files.zip**   *(decompress the archive into the constituent files and directories*
- tar
  - **tar   -cvf my_file.tar my_file1 my_file2 my_dir**   *(create a compressed archive called* my_files.tar*, containing files* my_file1, my_file2 *and the directory* my_dir *with all its content)*
  - **tar –tvf   my_file.tar**   *(list contents of the tar archive* my_file.tar*)*
  - **tar - xvf my_files.tar**   *(decompress the archive into the constituent files and directories)*

Working with files

Common compressed formats (continued):

- tgz (also, tar.gz – essentially a combo of “tar” and “gzip”)
  - **tar -czvf my_file.tgz my_file1 my_file2 my_dir** *(create a compressed archive called* my_files.tgz*, containing files* my_file1, my_file2 *and the directory* my_dir *with all its content)*
  - **tar –tzvf my_file.tgz** *(list contents of the tar archive* my_file.tar*)*
  - **tar -xzvf my_files.tgz** *(decompress the archive into the constituent files and directories)*

Compression works best (i.e., saves most disk space) for text files (e.g., large FASTQ files).

Getting help about compression tools:

- **gzip -h, bzip2 --help, zip, tar --help**
- **man gzip, man bzip2, man zip, man tar** (may be intimidating…)

Working with text files

Viewing text files

Examples:

- **more README.txt** *(display the content of the file README.txt in the current directory dividing the file into pages; press SPACE bar to go to the next page)*
- **head -100 my_reads.fastq** *(display first 100 lines of the file my_reads.fastq in the current directory)*
- **tail -100 my_reads.fastq** *(display last 100 lines of the file my_reads.fastq in the current directory)*
- **tail -1000 my_eads.fastq | more** *(extract the last 1000 lines of the file my_reads.fastq and display them page by page)*
- **head -1000 my_reads.fastq | tail -100** *(display lines 901 through 1000 of the file my_reads.fastq)*
- **cat my_reads.fastq** *(print the file on screen – good for small files)*
- **wc my_reads.fastq** *(display the number of lines, words, and characters in a file)*

Looking for a string in a group of text files:

- **grep "Error: lane" *.out** *(display all files *.out from the current directory which contain the string "Error: lane"; also display the lines containing that string)*

Editing text files

vi
- Available on all UNIX-like systems (Linux included), i.e., also on lab workstations
- Free Windows implementation available (once you learn vi, you can just use one editor everywhere)
- User interface rather peculiar (no nice buttons to click, need to remember quite a few keyboard commands instead)
- Some love it, some hate it

gedit (installed on lab workstations; just type "gedit" to invoke)
- X-windows application – need to have X-ming running on client PC.
- May be slow on slow networks…

edit+ (http://www.editplus.com/)
- Commercial product
- Runs on a local machine (laptop) and transfers data to/from Linux workstation as needed
- Can browse Linux directories in a Windows-like file explorer
- May be slow on slow networks
- Some people swear by it

vi basics

Opening a file:

vi my_reads.fastq (open the file my_reads.fastq in the current directory for editing; if the file does not exist, it will be created)

Command mode: typing will issue commands to the editor (rather than change text itself)

Edit mode: typing will enter/change text in the document

<Esc>	exit edit mode and enter command mode (this is the most important key – use it whenever you are lost)

The following commands will **take you to edit mode**:

i	enter insert mode
r	single replace
R	multiple replace
a	move one character right and enter insert mode
o	start a new line under current line
O	start a new line above the current line

The following commands **operate in command mode (hit <Esc> before using them)**

x	delete one character at cursor position
dd	delete the current line
G	go to end of file
1G	go to beginning of file
154G	go to line 154
$	go to end of line
1	go to beginning of line
:q!	exit without saving
:w	save (but not exit)
:wq!	save and exit

Arrow keys: move cursor around (in both modes)

Disk usage guidelines

(3CPG lab specific)

Your home directory (e.g., /home/bukowski)

- Is network-mounted and therefore access to it is slow
- Visible from each workstation – no matter which one you log in to
- 200 GB quota will be imposed (may change depending on conditions)
- Use it to store files which you use frequently (reference genomes, index files) or which are small and hard to replace (scripts and executables)
- Never run any disk intensive applications (all Next-Gen tools are disk intensive) with your home directory (or any of its subdirectories) as the "current directory". Work on **/workdir** instead.

The **/workdir** directory

- Is local to its workstation (located on disks physically attached to the machine's controller)
- Not visible from other workstations
- Temporary – the content of /workdir may be erased after you log out. When you log in again, your files may be no longer there
- After you log in, create your own subdirectory in /workdir (if not already there)
- All the files to be used in processing have to be moved to that subdirectory
- Applications have to be started in that subdirectory
- Important output files have to be copied back to the home directory or (better yet) out of the machine.

Checking disk space

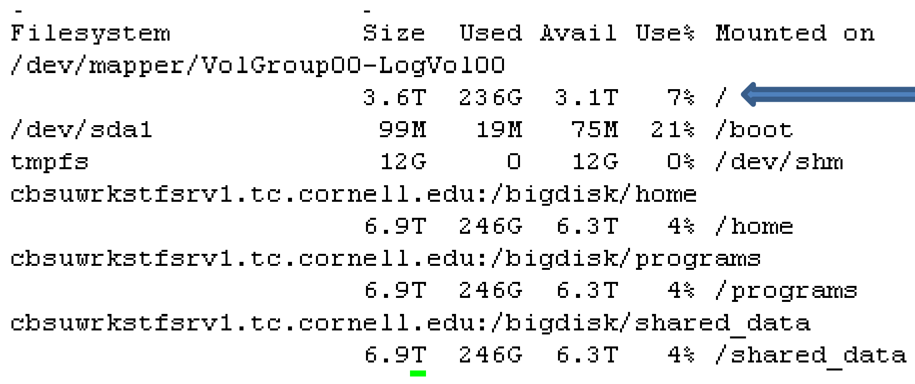
<u>How much disk space is taken by my files?</u>

du –hs . *(displays combined size of all files in the current directory and recursively in all its subdirectories)*

du –h --max-depth=1 . *(as above, but sizes of each subdirectory are also displayed)*

How much disk space is available?

df -h



```
-                      -
Filesystem             Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup00-LogVol00
                       3.6T  236G  3.1T   7% /
/dev/sda1               99M   19M   75M  21% /boot
tmpfs                   12G     0   12G   0% /dev/shm
cbsuwrkstfsrv1.tc.cornell.edu:/bigdisk/home
                       6.9T  246G  6.3T   4% /home
cbsuwrkstfsrv1.tc.cornell.edu:/bigdisk/programs
                       6.9T  246G  6.3T   4% /programs
cbsuwrkstfsrv1.tc.cornell.edu:/bigdisk/shared_data
                       6.9T  246G  6.3T   4% /shared_data
```

/workdir is a part of it

File transfer

between PC or Mac and a lab workstation

On Windows PC: install and use your favorite **sftp client** program, such as

- **winscp**: http://winscp.net/eng/index.php
- **CoreFTP LE**: http://www.coreftp.com/
- **FileZilla** (client): http://filezilla-project.org/
- … others…
- When connecting to Lab workstations from a client, use the **sftp** protocol. You will be asked for your user name and password (the same you use to log in to the lab workstations).
- Transfer text file in text mode, binary files in binary mode (the "default" not always right).
- All clients feature
  - File explorer-like graphical interface to files on both the PC and on the Linux machine
  - Drag-and-drop functionality

On a Mac: recommended file transfer program is **fetch** (recommended by Cornell CIT)

- http://www2.cit.cornell.edu/services/systems_support/filefetch.html#fetchinst
- graphical user interface
- Drag-and-drop functionality

Large files (> 0.5 GB) should be transferred to your subdirectory under /workdir (e.g., /workdir/bukowski). Avoid storing and processing such files in your home directory.

File transfer

fixing Windows/Mac – Linux text file conversion problems

unix2dos my_file (convert a text file in linux format my_file to Windows/Mac format, i.e., change line endings)

dos2unix my_file (convert a text file my_file in Windows/Mac format to Linux format, i.e., change line endings)

File transfer

between a lab workstation and another Linux machine

Suppose we want to transfer a file from **cbsuss04.tc.cornell.edu** (another Linux machine; substitute "your" Linux machine here) and **cbsuwrkst2** lab workstation.

Option 1: when logged in to **cbsuwrkst2**, sftp to **cbsuss04** by running the following commands:

cd /workdir/bukowski *(this is where we want the file to be on cbsuwrkst2)*
sftp bukowski@cbsuss04.tc.cornell.edu *(instead of "bukowski", use your own user name on cbsuss04; you will be asked for password)*
cd /data/bukowski/reads *(on cbsuss04, go to the directory where the file is)*
get my_read.fastq *(transfer, or "get" the file from cbsuss04)*
quit *(exit sftp client and disconnect from cbsuss04 – we are back on cbsuwrkst2)*

Option 2: when logged in to **cbsuss04**, sftp to **cbsuwrkst2** by running the following commands:

cd /data/bukowski/reads *(this is where the file is on cbsuss04)*
sftp bukowski@cbsuwrkst2.tc.cornell.edu *(instead of "bukowski", use your own user name on cbsuss04; you will be asked your lab password)*
cd /workdir/bukowski *(on cbsuwrkst2, go to the directory where the file is supposed to be stored)*
put my_read.fastq *(transfer, or "put" the file on cbsuwrkst2)*
quit *(exit sftp client and disconnect from cbsuwrkst2– we are back on cbsuss04)*

File transfer

from web- and ftp sites to lab workstations

Option 1: download using a web browser on workstation. While logged in to the workstation, execute the following:

- **firefox** *(this will start the Firefox browser on the workstation)*
  - *If you are working remotely from a PC, you will need to have Xming running. Note: Firefox browser you just started is* ***running on Linux workstation****, your PC is just displaying the browser's window. May be slow on slow networks…*
- Navigate to the site you want to download the file from, click on download link. The browser will ask for destination directory (on the workstation !) to put the file in. Select a directory (should be in /workdir if the file is large) and let the browser complete the download.
- Close Firefox browser if no longer needed.

Option 2: run **wget** command on the workstation (if you know the URL of the file)

- **wget ftp://ftp.ncbi.nih.gov/blast/matrices/BLOSUM100** *(will download the file BLOSUM100 from the NCBI FTP site and deposit it in the current directory under the name BLOSUM100)*
- **wget -O e_coli_1000_1.fq "http://cbsuapps.tc.cornell.edu/Sequencing/showseqfile.aspx?cntrl=646698859&laneid=487&mode=http&file=e_coli_1000_1.fq"**
- *(the command above can be used to download files given by complicated URLs; note the "" marks around the link and the* **–O** *option which specifies the name you want to give the downloaded file)*

More about commands

- Each command is, in fact, an executable program stored somewhere on disk, usually in places like **/bin**, **/usr/bin**, or **/usr/local/bin**
  - **which mv** *(tells us where on disk the command mv is located)*
- Why can we just use **mv** rather than the full name **/bin/mv** ? Because of the search path environment variable which is defined for everybody. The you type **mv**, the system tries each directory on the search path one by one until it finds the corresponding executable.
  - **echo $PATH** *(displays the search path)*
  - Note: the current directory ./ is NOT in the search path. If you need to run a program located, say in your home directory, you need to precede it with **./**, for example, **./my_program**
- The next-gen analysis applications installed on the workstations are also in your $PATH. Thus, you can launch them using just the name rather than the full path:
  - Example: command **/programs/bin/samtools/samtools** is equivalent to **samtools**

Running applications

<path_to_application_executable> <options>

Example: generate BWA index for the D.melanogaster genome

- Study the BWA manual (http://bio-bwa.sourceforge.net/bwa.shtml) to learn what **<options>** are available, what they do, and how to accomplish the task at hand.
- For this example:
  - We will run the program in directory under **/workdir/bukowski/d_melanogaster**
  - We need the FASTA file with the genome, *flygenome.fa,* in that directory
  - We want the index files to end up in **/workdir/bukowski/d_melanogaster/bwaindex** and we want their names to start with "drosophila"

```
cd /workdir/bukowski/d_melanogaster
mkdir bwaindex
```

$$\underbrace{\texttt{bwa}}_{\text{Path to application executable}}\quad \underbrace{\texttt{index}\quad \texttt{-p bwaindex/drosophila}\quad \texttt{flygenome.fa}}_{\text{Program options}}$$

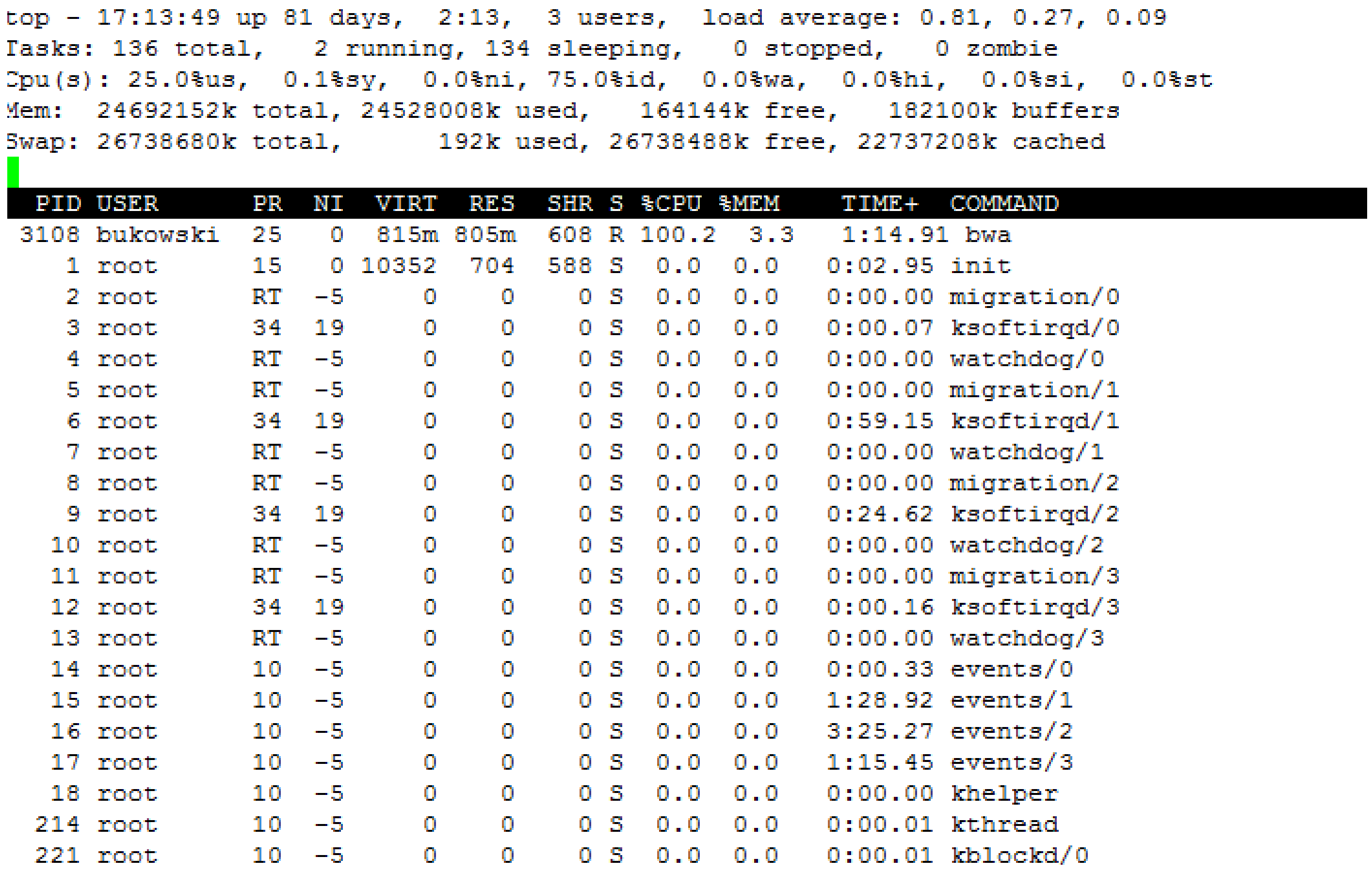
$$\texttt{bwa}\quad \texttt{index}\quad \texttt{-p bwaindex/drosophila}\quad \texttt{flygenome.fa}\quad \underbrace{\texttt{1> run.log}\quad \texttt{2> run.err}}_{\text{Redirect screen output}}$$

$$\texttt{bwa}\quad \texttt{index}\quad \texttt{-p bwaindex/drosophila}\quad \texttt{flygenome.fa}\quad \texttt{1> run.log}\quad \texttt{2> run.err}\quad \underbrace{\texttt{\&}}_{\text{Run in the background}}$$

Running applications

<u>Checking on your application: the **top** command</u>

To exit – just type **q**



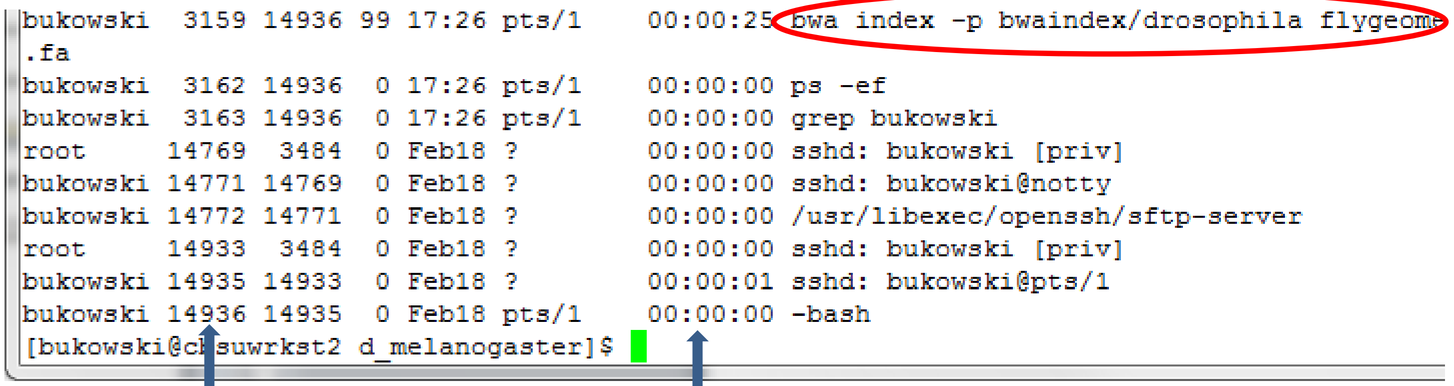
```
top - 17:13:49 up 81 days,  2:13,  3 users,  load average: 0.81, 0.27, 0.09
Tasks: 136 total,   2 running, 134 sleeping,   0 stopped,   0 zombie
Cpu(s): 25.0%us,  0.1%sy,  0.0%ni, 75.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  24692152k total, 24528008k used,   164144k free,   182100k buffers
Swap: 26738680k total,      192k used, 26738488k free, 22737208k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 3108 bukowski  25   0  815m 805m  608 R 100.2  3.3   1:14.91 bwa
    1 root      15   0 10352  704  588 S  0.0  0.0   0:02.95 init
    2 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.07 ksoftirqd/0
    4 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    5 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/1
    6 root      34  19     0    0    0 S  0.0  0.0   0:59.15 ksoftirqd/1
    7 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/1
    8 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/2
    9 root      34  19     0    0    0 S  0.0  0.0   0:24.62 ksoftirqd/2
   10 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/2
   11 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/3
   12 root      34  19     0    0    0 S  0.0  0.0   0:00.16 ksoftirqd/3
   13 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/3
   14 root      10  -5     0    0    0 S  0.0  0.0   0:00.33 events/0
   15 root      10  -5     0    0    0 S  0.0  0.0   1:28.92 events/1
   16 root      10  -5     0    0    0 S  0.0  0.0   3:25.27 events/2
   17 root      10  -5     0    0    0 S  0.0  0.0   1:15.45 events/3
   18 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 khelper
  214 root      10  -5     0    0    0 S  0.0  0.0   0:00.01 kthread
  221 root      10  -5     0    0    0 S  0.0  0.0   0:00.01 kblockd/0
```

Running applications

<u>Checking on your application:</u>
the **ps** command – display info about all your processes – one of them should be **bwa**

ps –ef | grep bukowski



```
bukowski  3159 14936 99 17:26 pts/1    00:00:25 bwa index -p bwaindex/drosophila flygeome
.fa
bukowski  3162 14936  0 17:26 pts/1    00:00:00 ps -ef
bukowski  3163 14936  0 17:26 pts/1    00:00:00 grep bukowski
root     14769  3484  0 Feb18 ?        00:00:00 sshd: bukowski [priv]
bukowski 14771 14769  0 Feb18 ?        00:00:00 sshd: bukowski@notty
bukowski 14772 14771  0 Feb18 ?        00:00:00 /usr/libexec/openssh/sftp-server
root     14933  3484  0 Feb18 ?        00:00:00 sshd: bukowski [priv]
bukowski 14935 14933  0 Feb18 ?        00:00:01 sshd: bukowski@pts/1
bukowski 14936 14935  0 Feb18 pts/1    00:00:00 -bash
[bukowski@cbsuwrkst2 d_melanogaster]$ 
```

Process ID (PID)

Running time

Try **man ps** for more info about the **ps** command.

Running applications

<u>Stopping applications</u>

- If the application is running in the foreground (i.e., without “&”), it can be stopped with Ctrl-C (press and hold the Ctrl key, then press the “C” key) issued from the window (terminal) it is running in.
- If the application is running in the background (i.e., with “&”), it can be stopped with the **kill** command

```
kill   -9   <PID>
```

Where <PID> is the process id obtained rom the **ps** command. For example, to terminate the bwa process form the previous slide, we would use

```
kill   -9  3159
```

Try **man kill** for more info about the **kill** command.

Running applications

basic shell scripting

Typically, data processing requires a “pipeline” – several commands run in succession, so that output from one command is input to the next one.

<u>Example:</u>
Aligning Illumina reads to a genome using BWA and storing the alignment in BAM format requires three steps:

```
bwa  aln  bwaindex/drosophila  short_read.fastq  1> aln.sai  2> log

bwa   samse   bwaindex/drosophila    aln.sai    short_read.fastq   1> aln.sam    2>> log

samtools    view  -bS  -o  aln.bam   aln.sam   2>> log
```

The three commands above may be put in a <u>text file</u>, e.g., **bwascript.sh**, created with a text editor. The script may then be executed:

- **chmod   u+x   bwascript.sh** *(make the file executable; needs to be done only once)*
- **./bwascript.sh   1> script.log   2> script.err &** *(run script in the background)*

Note: use “&” for the whole script rather than in each command (why?)

Running applications

basic shell scripting

Slightly more complicated (and more useful) script

```
#!/bin/bash

INFILE=$1

# Run alignment
bwa  aln  bwaindex/drosophila  ${INFILE}  1> aln.sai  2> log
# Produce alignment in SAM format
bwa   samse   bwaindex/drosophila   aln.sai   ${INFILE}   1> aln.sam   2>> log
# Remove the intermediate sai file to save space
rm aln.sai
# Run samtools to generate the BAM file, name it after the input file
samtools   view  -bS  -o  ${INFILE}_aln.bam   aln.sam   2>> log
# Remove the SAM file to save space
rm   aln.sam
```

Run the script with the following command:

```
./bwascript.sh   short_read.fastq   1> script.log   2> script.err   &
```

More about scripting

Multiple scripting tools available

- shell (bash, tcsh)
- perl (probably the most popular in biology)
- python
- awk (mostly text parsing and processing)
- sed (mostly text parsing and processing)

A separate course on scripting is planned.

Try it

Option 1: 3CPG lab workstations

- Reserve time, log in, practice Linux

Option 2: we set up a student account on one of our older (but still functional) Linux systems, **cbsuss02.tc.cornell.edu**

- ssh to the machine with loginID **cbsuguest** and password **CBSU4ever!**
  - Windows users: use PuTTy
  - Mac users: open terminal window and run
    ssh –X cbsuguest@cbsuss02.tc.cornell.edu
- Create your own subdirectory under /home/cbsuguest and "cd" to it
  - **mkdir /home/cbsuguest/bukowski**
  - **cd /home/cbsuguest/bukowski**
- Try working with Linux commands, files, editors, etc. Note: Next-Gen applications are not available on cbsuss02 – this machine is only for basic Linux training