jpombert@Newton:~

File Edit View Search Terminal Help

```
top - 17:49:16 up 4 days,  8:23,  4 users,  load average: 1.29, 1.24, 1.15
Tasks: 390 total,   2 running, 388 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3.4 us,  0.2 sy,  0.0 ni, 96.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:  26414072+total, 26356950+used,   571228 free,   263368 buffers
KiB Swap:  4194300 total,    39964 used,  4154336 free, 19046982+cached

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
15938 ncorradi  20   0 65.004g 0.060t   2872 R  99.5 24.4  10751:03 mira
  821 root      20   0  355548 112212  58172 S   7.0  0.0  58:19.29 Xorg
 1774 jpombert  20   0 2635440 811380  47468 S   6.3  0.3   1510:51 cinnamon
29543 jpombert  20   0  710648  22648  12716 S   1.0  0.0   0:44.69 gnome-terminal-
  104 root      rt   0       0      0      0 S   0.3  0.0   0:33.96 watchdog/19
29478 root      20   0       0      0      0 S   0.3  0.0   0:01.92 kworker/2:0
29671 root      20   0       0      0      0 S   0.3  0.0   0:12.96 kworker/24:2
    1 root      20   0   50856   6832   3352 S   0.0  0.0   0:09.61 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.29 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.20 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kworker/u64:0
    8 root      rt   0       0      0      0 S   0.0  0.0   0:06.99 migration/0
    9 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
   10 root      20   0       0      0      0 S   0.0  0.0   9:36.68 rcu_sched
   11 root      rt   0       0      0      0 S   0.0  0.0   0:07.63 watchdog/0
   12 root      rt   0       0      0      0 S   0.0  0.0   0:22.73 watchdog/1
   13 root      rt   0       0      0      0 S   0.0  0.0   0:04.17 migration/1
   14 root      20   0       0      0      0 S   0.0  0.0   0:00.04 ksoftirqd/1
   16 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:0H
   17 root      rt   0       0      0      0 S   0.0  0.0   0:20.45 watchdog/2
   18 root      rt   0       0      0      0 S   0.0  0.0   0:08.29 migration/2
   19 root      20   0       0      0      0 S   0.0  0.0   0:00.05 ksoftirqd/2
   21 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/2:0H
   22 root      rt   0       0      0      0 S   0.0  0.0   0:24.09 watchdog/3
   23 root      rt   0       0      0      0 S   0.0  0.0   0:10.16 migration/3
   24 root      20   0       0      0      0 S   0.0  0.0   0:00.15 ksoftirqd/3
   26 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/3:0H
   27 root      rt   0       0      0      0 S   0.0  0.0   0:25.88 watchdog/4
   28 root      rt   0       0      0      0 S   0.0  0.0   0:07.24 migration/4
   29 root      20   0       0      0      0 S   0.0  0.0   0:00.05 ksoftirqd/4
   31 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/4:0H
   32 root      rt   0       0      0      0 S   0.0  0.0   0:05.97 watchdog/5
   33 root      rt   0       0      0      0 S   0.0  0.0   0:03.91 migration/5
   34 root      20   0       0      0      0 S   0.0  0.0   0:00.03 ksoftirqd/5
   36 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/5:0H
   37 root      rt   0       0      0      0 S   0.0  0.0   0:24.67 watchdog/6
   38 root      rt   0       0      0      0 S   0.0  0.0   0:08.86 migration/6
   39 root      20   0       0      0      0 S   0.0  0.0   0:00.04 ksoftirqd/6
   41 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/6:0H
   42 root      rt   0       0      0      0 S   0.0  0.0   0:30.20 watchdog/7
   43 root      rt   0       0      0      0 S   0.0  0.0   0:05.15 migration/7
   44 root      20   0       0      0      0 S   0.0  0.0   0:00.01 ksoftirqd/7
   46 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/7:0H
   47 root      rt   0       0      0      0 S   0.0  0.0   0:24.67 watchdog/8
   48 root      rt   0       0      0      0 S   0.0  0.0   0:22.09 migration/8
   49 root      20   0       0      0      0 S   0.0  0.0   0:08.45 ksoftirqd/8
   51 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/8:0H
```

The Bash Shell

Jean-François Pombert, Ph.D.
Office PS 296 (Lab PS 340)

Menu jpombert@Newt... [perl.pdf — A (ve... scripts Notes (~/Deskto... jpombert@Newt... GNU Image Mani... Create a New Im... Mon Aug 12, 17:49

Connect to Mozart by SSH

Replace **$ID** by your username

ssh $ID@216.47.151.148 (Use the IP, Mozart not implemented in DNS)

The optional -X switch permits GUI over SSH, requires bandwidth

Your home folder (~)

```
pwd                    ## lists your current directory
cd ~                   ## returns to your home from anywhere
cd $HOME               ## same thing
cd /home/username/     ## same; may change according to Linux setup
```

Getting around (absolute paths)

Absolute paths start from the root (/)

cd /folder1/folder2/folder3

You can only access folders for which you have the right permissions

Tab autocompletion will save you a lot of time

Getting around (relative paths)

```
cd ./          ## moves to current folder, not very useful
cd ../         ## moves back one folder
cd ../../      ## moves back two folders
```

Ls – Listing the content of a folder

```
ls /path_to_dir/        ## lists content of specified folder basic listing
ls -la /path_to_dir/    ## shows hidden files and file permissions
```

File extensions are not required in Linux/Unix

.files~ – Hidden in plain sight

Removing clutter

Using a dot (.) before a file/folder name hides it

Temporary file/folders ending with a tilde (~) are hidden

ls -la will show the hidden files

Man – integrated manuals

man program

e.g. man ls

Works with standard Bash programs, others may use -h or -help

Globs – Wildcards

```
ls *.txt        ## lists all files/folders ending with .txt
ls Dr*          ## lists all files/folders starting with Dr
```

Other useful wildcards exist, type *man 7 glob* for more info

Reading a text file

```
less /etc/netconfig        ## reads file interactively, can move up/down
head /etc/netconfig        ## reads only the first part
tail /etc/netconfig        ## reads only the last part
                           ## head/tail -n XX; reads the first/last XX lines instead
```

Creating/editing a text file

Avoid using spaces and weird characters in file/folder names

```
Underscores are OK          ## e.g. This_is_ok.txt
touch empty.txt             ## creates an empty file
nano empty.txt              ## command line text editor; sudo dnf install nano
```

Removing files

Files & folders are deleted permanently, proceed with caution!

```
rm file_to_remove.txt          ## removes file
rm -R Folder_To_Trash          ## removes the folder and all files within; recursive
```

Making/removing folders

Files & folders are deleted permanently, proceed with caution!

```
mkdir Test          ## creates Test directory
rmdir Test          ## removes Test directory, works only if empty
rm -R test          ## works even if folder not empty, careful!
```

cp – Copying files

```
cp file.txt /where/to/file.txt        ## Useful for backups/testing purposes
-R/-r/--recursive                     ## Recursive
-t directory                          ## Copies files to existing directory
```

mv – Moving files

```
mv file.txt /where/to/file.txt        ## Moves file.txt to new location
mv file.txt newname.txt               ## Renames file instead
-n/--no-clobber                       ## Prevents overwriting existing file
```

ln – Creating shortcuts

The function ln makes links between files

```
ln -s /path/to/original_file /path/to/shortcut_name        ## The name can be anything
ln -s /path/to/original_file          ## Not entering a second operand will create the shortcut
                                      ## in the current directory
  -s                                  ## -s means symbolic link; should always be used
```

Copy & Paste

Ctrl + shift + C ## Copy

Ctrl + shift + V ## Paste

Ctrl + C cancels running jobs, be careful using it in the shell

NOTE: These commands are for Linux; SSH terminals may have other shortcuts

Env – Environmental variables

env	## lists all current active variables
global *vs.* local	## system = global; user = local
session-specific variables	## exporting a variable is valid only for the active session

$HOME – Where the heart is

Links to your main folder (echo $HOME)

Remember to back it up

Working folders can be set elsewhere

$PATH – Environmental variable

echo $PATH

Directory listing which folders to search for executables

Negates the need to type in the absolute/relative paths

$LD_LIBRARY_PATH – Env. var.

echo $LD_LIBRARY_PATH

Directory listing which folders to search for libraries

Required for program-specific dependencies

$PERL5LIB – Perl 5 libraries

echo $PERL5LIB

A “log” listing which folders to search for Perl libraries

Required for specific Perl modules

Setting environment variables

Adding installed software to your local $PATH

```
export                ## Sets the specified variable in the environment; temporary
export POTATO="I’m a potato"; echo $POTATO    ## Defining the variable
export POTATO=$POTATO:"and a cucumber"; echo $POTATO ## Adding to the variable
```

```
jpombert@fedora:~ — nano /home/jpombert/.bash_profile
  GNU nano 5.3                    /home/jpombert/.bash_profile
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

^G Help      ^O Write Out  ^W Where Is   ^K Cut      ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit      ^R Read File  ^\ Replace    ^U Paste    ^J Justify    ^_ Go To Line M-E Redo      M-6 Copy
```

Ubuntu => .profile
MacOS => .zprofile

The .bash_profile

Adding installed software to **your** $PATH

Can also be used to set other environment variables and/or shortcuts

Sets the specified variables in the environment; read at login

Editing your .bash_profile

```
nano ~/.bash_profile          ## modify your settings
Ctrl+X to save
source ~/.bash_profile        ## update your settings
```

```
  GNU nano 2.8.7                    File: /etc/profile.d/bash.sh

PATH=$PATH:/opt/artemis                                  ## Artemis
PATH=$PATH:/opt/bandage                                  ## Bandage
PATH=$PATH:/opt/barrnap                                  ## Barnap
PATH=$PATH:/opt/bcftools                                 ## Bcftools
PATH=$PATH:/opt/bowtie2                                  ## Bowtie2
PATH=$PATH:/opt/canu-1.7/bin                             ## Canu
PATH=$PATH:/opt/cdhit                                    ## CD-Hit
PATH=$PATH:/opt/clustalx                                 ## ClustalO/W2/X
PATH=$PATH:/opt/DEXTRACTOR                               ## Dextractor
PATH=$PATH:/opt/dfast_core                               ## DFast
                        [ File '/etc/profile.d/bash.sh' is unwritable ]
^G Get Help   ^O Write Out   ^W Where Is   ^K Cut Text   ^J Justify   ^C Cur Pos   M-U Undo   M-A Mark Text
^X Exit       ^R Read File   ^\ Replace    ^U Uncut Text ^T To Linter ^_ Go To Line M-E Redo   M-6 Copy Text
```

Scripts in /etc/profile.d/

Require super-user privileges ## Run system-wide

Can be used to set variables for everyone

sudo nano /etc/profile.d/bash.sh ## Creating a shell script to set our variables

or perform other tasks

Aliases – (and .bashrc)

Shorten/remember commands

```
Not permanent unless in ~/.bashrc    ## or ~/.bash_profile
alias name="my full command"         ## creates alias substituting for longer command
unalias alias_name                   ## removes alias; only valid if alias isn't permanent
```

Find – Command line search

```
find ./                                  ## lists all files in current folder; searches are recursive
find ./ -name 'xxx*'                     ## searches for file with name containing xxx in ./
find ./ -iname 'xxx*'                    ## same, but case insensitive
find ./ -iname 'xxx*' -user 'uname'      ## same, but owned by user “uname”
```

Locate – Command line search

```
locate *.txt                ## lists all txt files; searches are recursive
locate *.txt | less         ## same but pipes into less for better readability
locate -i evol              ## searches for pattern evol; case insensitive
```

Locate uses indexed searches; files are indexed with updatedb via the OS cron jobs

Exercise 1 – Add bowtie2 to ~/.bash_profile

1) Look into /opt/

2) Find bowtie2

3) Add the folder containing the program to your $PATH variable. To make it permanent, add it to your ~/.bash_profile.

4) Source your ~/.bash_profile

5) Launch bowtie2

Exercise 2 – Add Ray to ~/.bash_profile

1) Look into /opt/

2) Find Ray

3) Add the folder containing the program to your $PATH variable. To make it permanent, add it to your ~/.bash_profile.

4) Source your ~/.bash_profile

5) Launch Ray, what happens?

6) Libraries? (export LD_LIBRARY_PATH=/usr/lib64/openmpi/lib) ## You may encounter an error if the libraries are not set

IBM Knowledge Center https://www.ibm.com/support/knowledgecenter/SSB27H_6.2.0/fl2cp_standard_streams_redirection.html

Home > z/VSE 6.2.0 > Developing C Language Programs to run under z/VSE with LE/VSE > Input and Output >

Previous Next

Search in all products

Standard Streams and Redirection

Search in this product...

Table of Contents | Change version or product | Print | PDF | Help | Take a tour

z/VSE with LE/VSE
+ About This Publication
− Input and Output
+ Introduction to C Input and Output
+ Models of C I/O
+ Opening Files
Buffering of C Streams
+ ASA Text Files
+ LE/VSE C Run-Time Support for the Double-Byte Character Set (DBCS)
− Standard Streams and Redirection
Default Open Modes
Using the Redirection Symbols

A C program has associated with it *standard streams*. You do not have to open these streams, because they are automatically set up for you by C when you include the stdio.h header file. Table 1 below shows three standard streams for C and the functions that implicitly use them.

Table 1. C Standard Streams

Name of stream	Purpose	Functions that use it
stdin	The input device from which your C program usually retrieves its data.	**getchar() scanf() gets()**
stdout	The output device to which your C program normally directs its output.	**printf() puts() putchar()**
stderr	The output device to which your C program directs its diagnostic messages. LE/VSE C Run-Time uses stderr to collect error messages about exceptions that occur.	**perror()**

About STDOUT and STDERR

Standard input/output (I/O) streams

STDIN	Standard input	## descriptor 0
STDOUT	Standard output	## descriptor 1
STDERR	Standard error log	## descriptor 2

The > and >> signs – Outputs

> Redirects outputs to files

```
ls -la ~ > home_content.txt                          ## Writes STDOUT to file
ls -la ~ &> home_content.txt                         ## Writes STDOUT/STDERR to file
ls -la ~ 1> home_content.std 2> home_content.err     ## Writes STDOUT/STDERR to distinct files
```

>> Appends outputs to files

```
ls -la ~ 1>> home_content.std 2>> home_content.err   ## Appends to file
```

The < sign – Inputs

< specifies inputs to your command

```
desired_command < file.txt      ## Feeds the file to the your command line
                                ## Does not work with every command
```

e.g. less < home_content.txt

History – Remember what you did

```
history                          ## shows your command history
history >> my_commands.txt       ## writes previous commands to a file
↑↓ arrows                        ## scrolls through previous command
```

History – environment variables

HISTFILE ## File where commands are saved: *e.g.*, ~/.bash_history

HISTFILESIZE ## Number of commands to store in file. Default: 1000

HISTSIZE ## Number of cached commands. Default: 1000

echo $HISTFILESIZE

On MacOS: /etc/zshrc

The pipe | character

Feeds (redirects) the STDOUT of a program into another

Computer pipelines are often a line of pipes, literally

e.g. dmesg | less ## dmesg shows kernel messages

Grep – Search text input

Globally search a Regular Expression and Print ## Searches through text

```
grep -e 'motif1' -e 'motif2' file    ## Searches for specified motifs (expression) in file
grep -i -f 'input_file' file         ## Searches for motifs in input file (one motif
                                     ## per line); case insensitive search
grep -P -e 'motif' file              ## Searches for a Perl regular expression
```

Exercise 3 – Grep

In /opt/scripts/

1) Find all .pl files containing the word 'perl' in /opt/scripts/

2) Count the number of time perl is mentioned in those files

3) Try with 'fasta'

4) Try with 'Fasta'

5) Try with 'Fa*'. Notice something weird?

Cat – Concatenating files

```
cat *.txt > big.txt                    ## Extremely useful for concatenating text files
Can also be used to display files      ## Default output is the shell
```

Careful if one or more files do not end with a return sign when concatenating!

Sed – Text manipulation

Big files? No problem

Example: sed 's/ /_/g' file.txt > newfile.txt ## replaces all whitespaces with underscores

Not compatible with Perl regex

Exercise 4 – dmesg, pipeline, GREP & more

1) In one command, print out all lines containing 'linux' from kernel messages in a file named linux_boot.txt. *These kernel messages can be displayed using dmesg*. ## dmesg (daemon messages); prints info from the Linux kernel
2) Try again with 'PCI'. Append the output to linux_boot.txt
3) Replace all 'PCI' words with 'AGP'. Save to new file called boot_test.txt
4) Grep 'PCI' again on the boot_test.txt
5) Do you find it?

The Bash loop

Your weekend's best friend

```
for i in {0..25}; do touch file_$i; done
for value in {1,7,22,44,57}; do …
for file in /opt/*; do …
```

Exercise 5 – Loops!

In one command line ## Hint -> ;

1) Create a loop that:

- Generates 100 files named Wonderful_$value
- Generates 100 folders named Loops_$value
- Moves the 100 files to the corresponding folders

2) Then, use a command that tells you a message to say that the job is done

If/elif .. then .. else – Conditionals

```
for x in {1..5}; do
  if [ $x = 2 ]; then
    echo "2 = $x";
  elif [ $x = 4 ]; then
    echo "4 = $x";
  else
    echo "$x isn't equal to 2 or 4";
  fi
done
```

Bash scripts

Your weekend's NEW best friend

Bash scripts are text files, one command per line (*e.g.* bash.sh)

Must be made executables to work

#!/bin/bash (where is your bash shell, if unsure type: which bash)

Bash script anatomy

Indicates where is the Bash binary

Setting variables

Executing commands

```
#! /bin/bash -x
## This is a comment!
IN1=../L1_sickle.fastq
IN2=../R1_sickle.fastq
SGA_BIN=sga
BWA_BIN=bwa
SAMTOOLS_BIN=samtools
BAM2DE_BIN=sga-bam2de.pl
ASTAT_BIN=sga-astat.py
DISTANCE_EST=DistanceEst
CPU=8
CORRECTION_K=41
MIN_OVERLAP=85
ASSEMBLE_OVERLAP=111
TRIM_LENGTH=400
MIN_CONTIG_LENGTH=500
MIN_PAIRS=10
# Check the required programs are installed and executable
prog_list="$SGA_BIN $BWA_BIN $SAMTOOLS_BIN $BAM2DE_BIN $DISTANCE_EST $ASTAT_BIN"
for prog in $prog_list; do
   hash $prog 2>/dev/null || { echo "Error $prog not found. Please place $prog on your PATH or update the *_BIN variables in this script"; exit 1; }
done
# Create and change Directory, create symlinks to preprocessed reads
mkdir k$CORRECTION_K
cd k$CORRECTION_K
ln -s ../reads.pp.fastq reads.pp.fastq
ln -s ../reads.pp.bwt reads.pp.bwt
ln -s ../reads.pp.sai reads.pp.sai

file_list="$IN1 $IN2"
for input in $file_list; do if [ ! -f $input ]; then echo "Error input file $input not found"; exit 1; fi done

$SGA_BIN correct -k $CORRECTION_K --learn -t $CPU -o reads.ss.fastq reads.pp.fastq
$SGA_BIN index -a ropebwt -t $CPU reads.ss.fastq
$SGA_BIN filter -x 2 -t $CPU reads.ss.fastq
$SGA_BIN overlap -m $MIN_OVERLAP -t $CPU reads.ss.filter.pass.fa
$SGA_BIN assemble -m $ASSEMBLE_OVERLAP --min-branch-length $TRIM_LENGTH -o primary reads.ss.filter.pass.asqg.gz
PRIMARY_CONTIGS=primary-contigs.fa
PRIMARY_GRAPH=primary-graph.asqg.gz
$BWA_BIN index $PRIMARY_CONTIGS
$BWA_BIN aln -t $CPU $PRIMARY_CONTIGS $IN1 > $IN1.sai
$BWA_BIN aln -t $CPU $PRIMARY_CONTIGS $IN2 > $IN2.sai
$BWA_BIN sampe $PRIMARY_CONTIGS $IN1.sai $IN2.sai $IN1 $IN2 | $SAMTOOLS_BIN view -Sb - > libPE.bam
$BAM2DE_BIN -n $MIN_PAIRS -m $MIN_CONTIG_LENGTH --prefix libPE libPE.bam
$ASTAT_BIN -m $MIN_CONTIG_LENGTH libPE.bam > libPE.astat
$SGA_BIN scaffold -m $MIN_CONTIG_LENGTH -a libPE.astat -o scaffolds.scaf --pe libPE.de $PRIMARY_CONTIGS
$SGA_BIN scaffold2fasta --use-overlap --write-unplaced -m $MIN_CONTIG_LENGTH -a $PRIMARY_GRAPH -o sga-scaffolds.fa scaffolds.scaf
```

Variables must be set before usage (obviously)

```
drwxr-xr-x. 1 jpombert jpombert  0 Jan 25 10:07 Desktop
drwxr-xr-x. 1 jpombert jpombert 20 Jan 30 10:53 Documents
drwxr-xr-x. 1 jpombert jpombert  0 Jan 28 10:06 Downloads
drwxr-xr-x. 1 jpombert jpombert  0 Jan 25 10:07 Music
drwxr-xr-x. 1 jpombert jpombert  0 Jan 25 10:07 Pictures
drwxr-xr-x. 1 jpombert jpombert  0 Jan 25 10:07 Public
drwxrwxr-x. 1 jpombert jpombert 50 Jan 31 10:21 Software
drwxr-xr-x. 1 jpombert jpombert  0 Jan 25 10:07 Templates
drwxr-xr-x. 1 jpombert jpombert  0 Jan 25 10:07 Videos
```

permissions user group

File permissions

User (U) drwxrwxrwx

Group (G) drwxrwxrwx

Others (O) drwxrwxrwx

r = read w = write x = execute - = permission denied

```
[jpombert@localhost ~]$ chmod 775 make_kmers.pl
[jpombert@localhost ~]$ ls -l make_kmers.pl
-rwxrwxr-x. 1 jpombert jpombert 649 Feb  3 10:10 make_kmers.pl
```

7 7 5

r	w	x
4	2	1

#		r		w		x		triplet
1	=	0	+	0	+	1	=	--x
2	=	0	+	2	+	0	=	-w-
3	=	0	+	2	+	1	=	-wx
4	=	4	+	0	+	0	=	r--
5	=	4	+	0	+	1	=	r-x
6	=	4	+	2	+	0	=	rw-
7	=	4	+	2	+	1	=	rwx

Chmod – Changing permissions

The UGO (and A)	## u = user; g = group; o = others; a = all
The RWX	## r = read; w = write; x = execute
The numbers	## 4 = read; 2 = write; 1 = execute
The (not so) lucky 777	## read + write + execute for everyone!!! -> BAD

Chown/chgrp – Changing ownership

chown OWNER folder/file ## Changing owner (user)

chown OWNER:GROUP folder/file ## Changing owner + group

chgrp GROUP folder/file ## Changing group only

-R, --recursive ## You must have the proper rights to change owners/permissions

Exercise 6 – Permissions

1) Create a text file called pwners.txt containing “Can you read it?”
2) Make the file readable, executable but not writable for the user and the group
3) Open it with nano, modify it and try to save it
4) Give yourself back writing permission

Exercise 7 – Bash scripts

Create a script that will use a loop to create 20 folders named FD_01 to FD_20

The loop should, per iteration:

1) Generate the folder

2) Change directory to the created folder

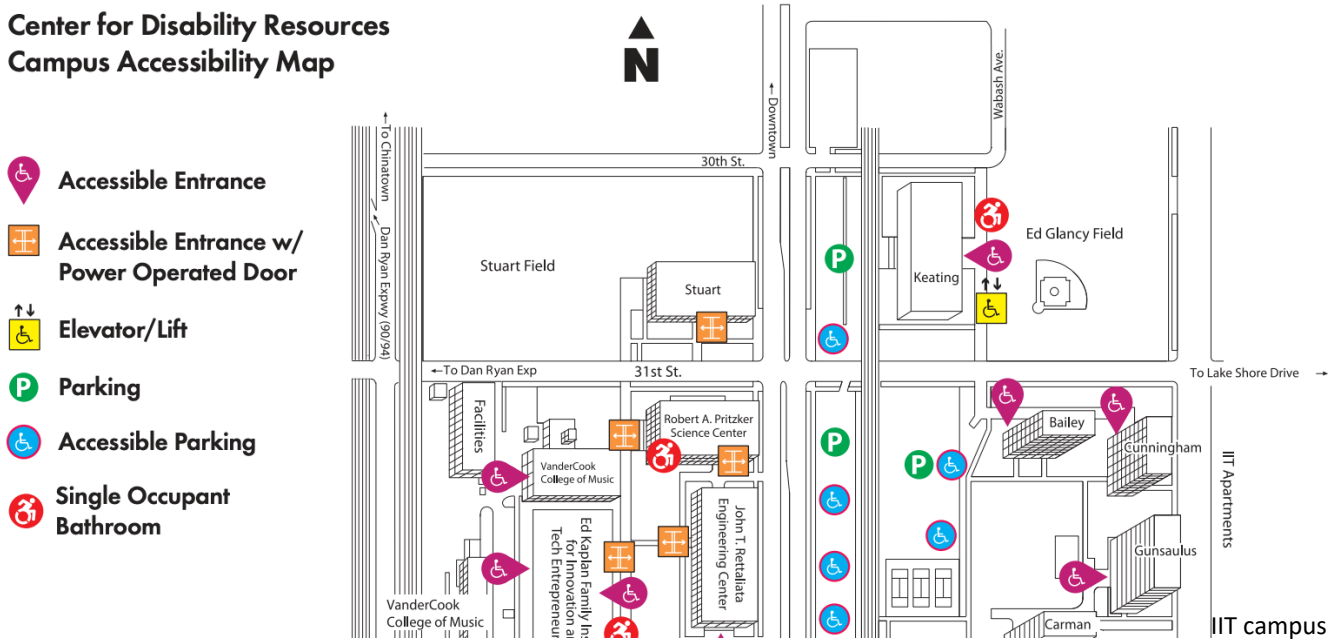
3) Create a file with the same folder number inside it ## *e.g.* rg01.txt

4) Get back out of the directory ## e.g. cd ../

5) Change permission of the folder and its content recursively so that only the owner can read/execute/overwrite it ## e.g. 700

Once the loop is completed:

Have the shell script tell you that the job is done! ## e.g. using echo



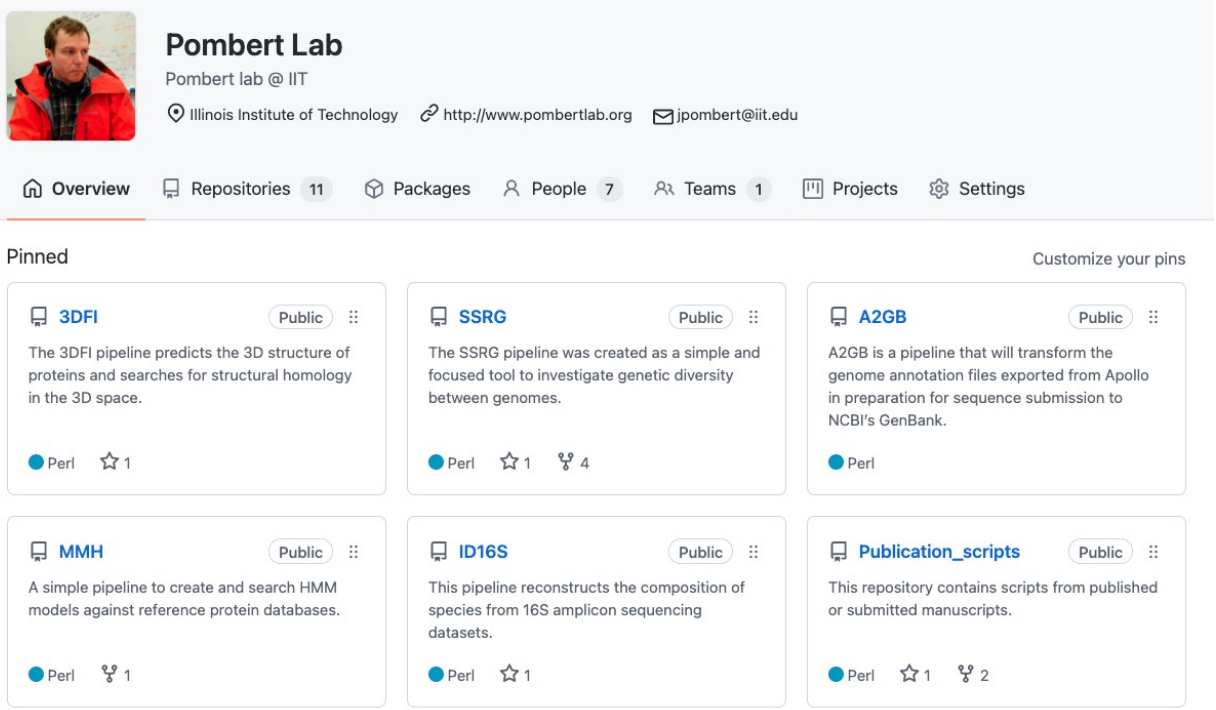
IIT campus map
https://www.iit.edu/sites/default/files/2019-11/mies_campus_map.pdf

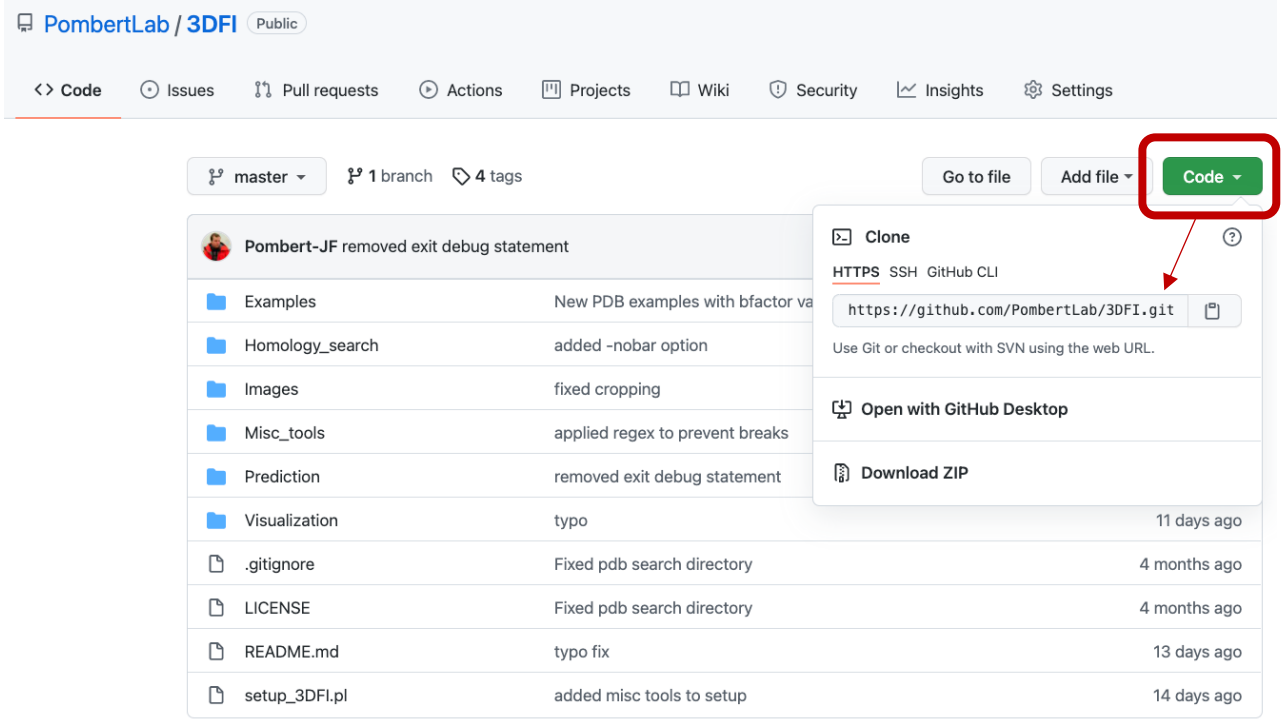
Wget – Getting files from the Web

Command-line downloader

wget https://www.iit.edu/sites/default/files/2019-11/mies_campus_map.pdf

Very useful when installing things remotely





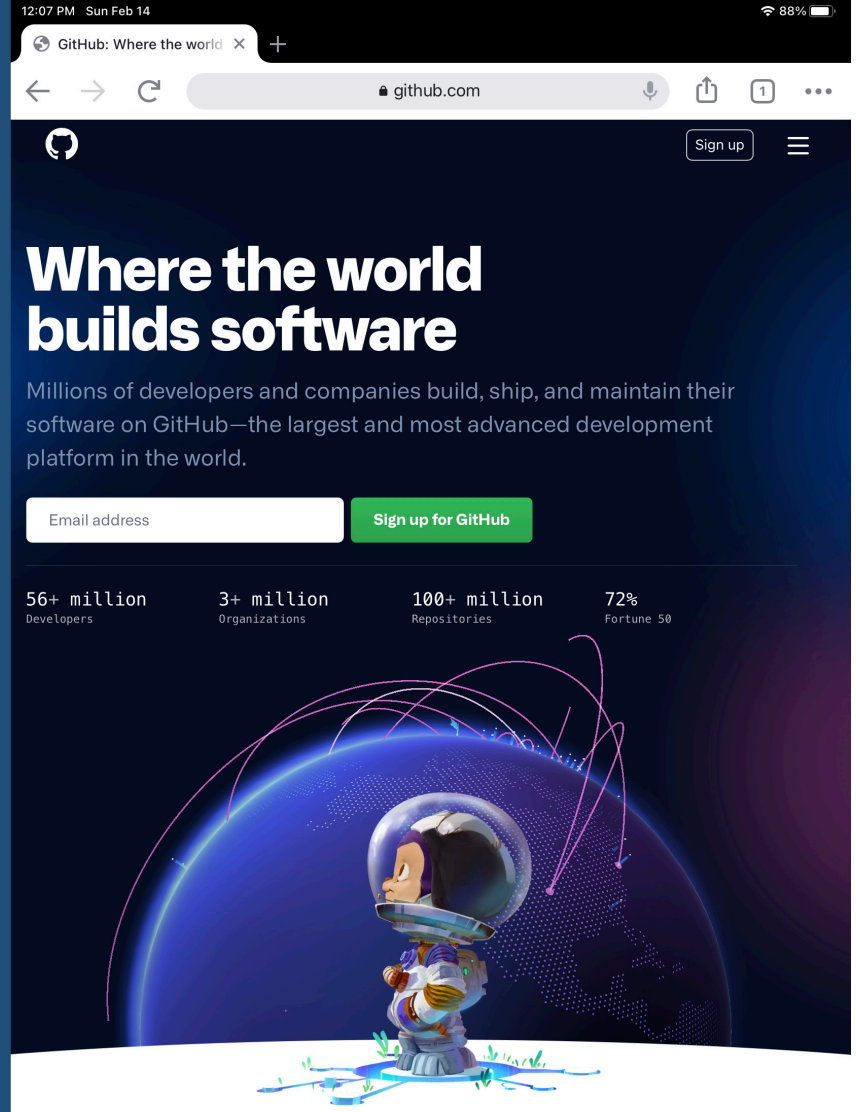
Git – Getting tools from the Web

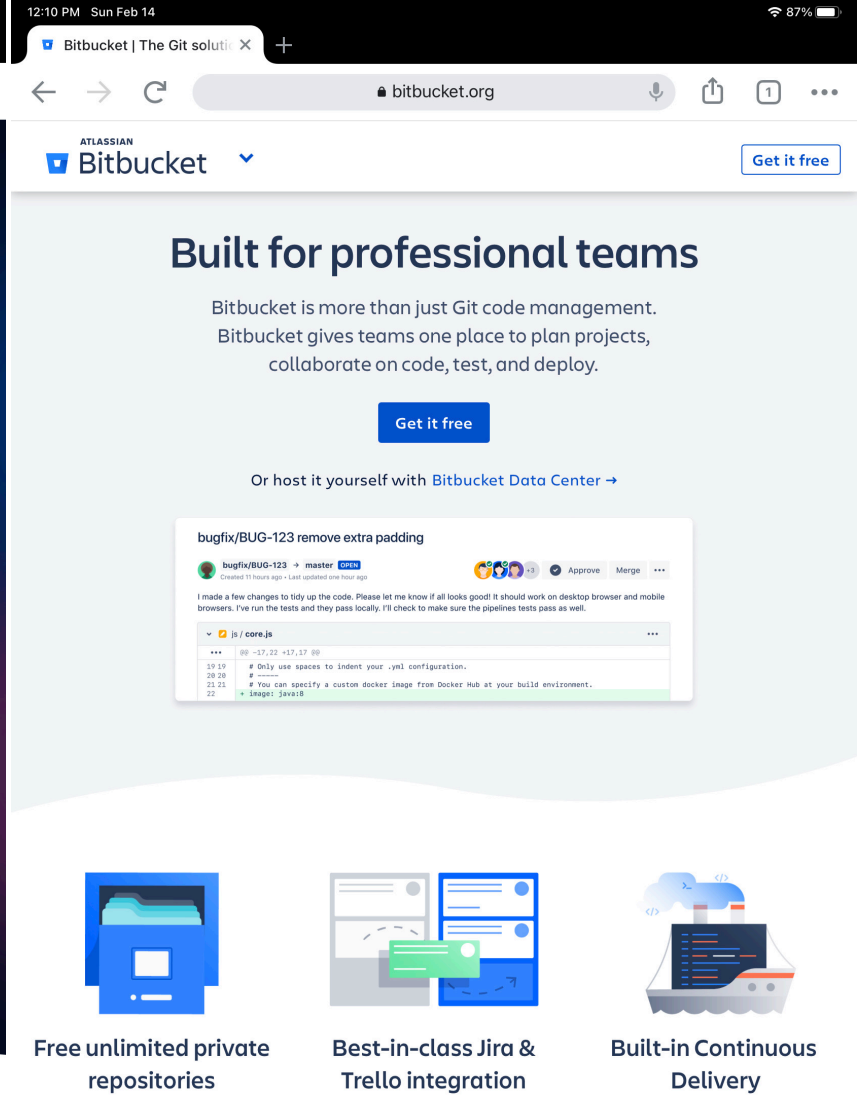
Git is an open source version control system

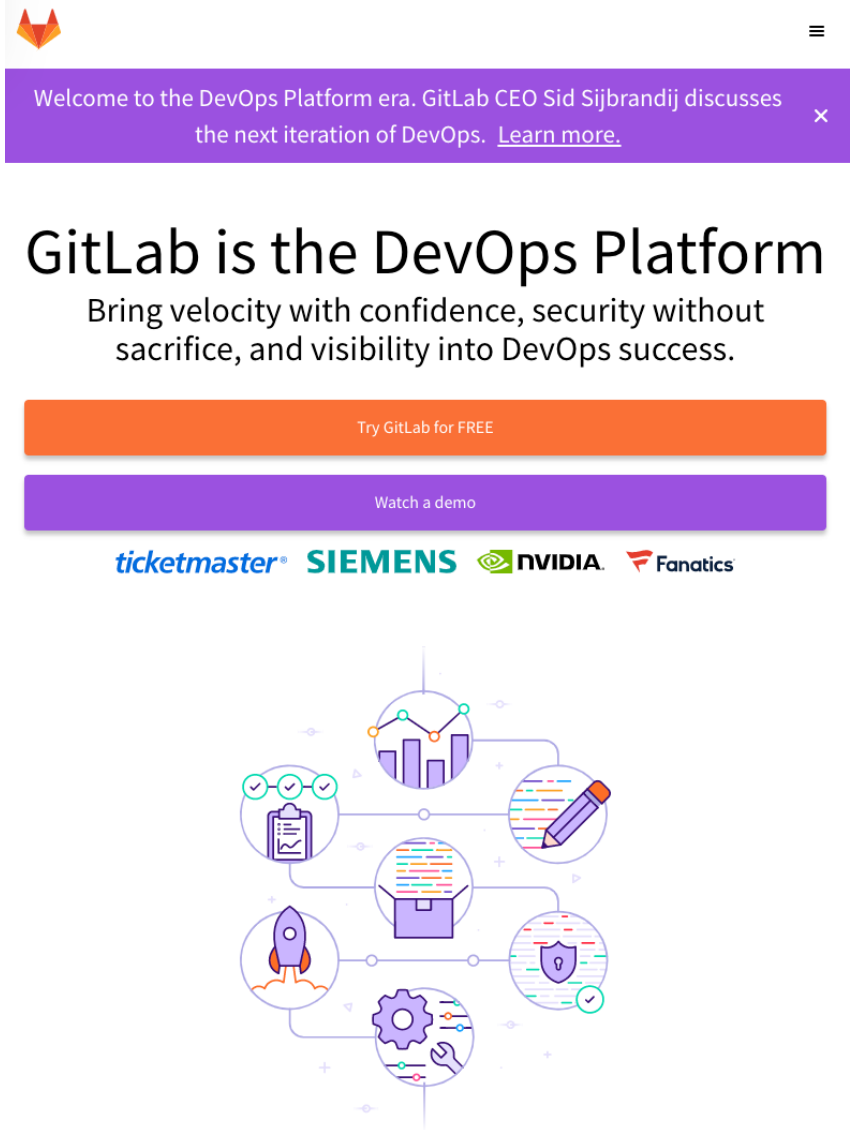
Downloading tools with git is easy

git clone https://github.com/PombertLab/3DFI.git	## clones the directory
git clone --recursive https://github.com/PombertLab/3DFI.git	## clones submodules too (if any)
cd 3DFI; git pull	## updates the code

https://www.atlassian.com/git/tutorials/what-is-git







Many online git repositories are available

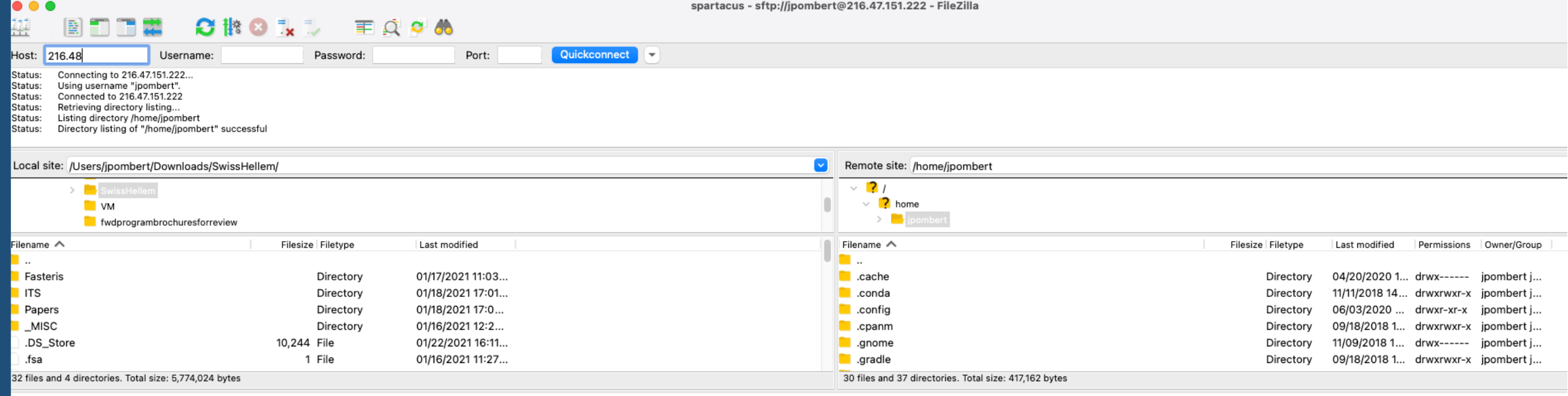
GitHub
Bitbucket
GitLab

SFTP – Transferring files (CMD)

From/to one machine to another

```
sftp username@servername (or IP address)        ## interactive mode
  > put local_file;  get remote_file            ## “put” uploads files; “get” downloads files
  > pwd/lpwd; ls/lls; cd/lcd;                   ## “l” before a command means “local”
  > exit                                        ## quits sftp; “bye” works too
sftp username@servername:/path/to/file          ## downloads file directly
```

https://linuxize.com/post/how-to-use-linux-sftp-command-to-transfer-files/



SFTP – Transferring files (GUI)

Many GUI SFTP clients are possible

FileZilla (https://filezilla-project.org/) is available of most platforms

To install FileZilla in Fedora from the command line:

1) sudo dnf install filezilla ## Installs FileZilla using the DNF package manager
2) filezilla ## To launch FileZilla from the Bash shell

Scp – Transferring files

From anywhere to anywhere

```
scp userID1@remoteserver1:/path/to/dir1/ userID2@remoteserver2:/path/to/dir2/
## Useful when transferring files between servers from a third location
```

Tar – Archiving/compressing files

```
tar -cvf Archive.tar Files_To_Compress          ## -c; creates archives
tar -czvf Archive.tar.gz Files_To_Compress      ## -c; creates archives with GZIP compression
tar -cjvf Archive.tar.bz2 Files_To_Compress     ## -c; creates archives with BZIP2 compression
tar -xvf Archive.tar                            ## -x; decompresses archives
## .tar.gz, .tgz => Tar + GZIP; .tar.bz2, .tbz => Tar + BZIP2
```

.gz => gzip => GNU zip

.gz (gzip files) – Compressed files

```
gzip -k *.txt          ## Compresses all text files; -k keep original files
pigz -k -p 8  *.txt    ## Same but multithreaded; -p => number of processors
gunzip -k *.gz         ## Decompresses all .gz files; -k keep original files
unpigz -k -p 8 *.gz    ## Same but multithreaded; -p => number of processors
```

(Un)zip – (de)compressing

```
zip archive.zip *.txt                    ## Compresses all .txt files into archive.zip
unzip archive.zip                        ## Decompressed .zip file
## (De)compressing .zip files is easy but most Linux folks
## use tar + GZIP or BZIP2 for archiving
```

du/df – Disk usage

```
du                          ## Disk usage; estimates file space usage
du -sh * /path/to/folder    ## s = summarize; h = human-readable
df                          ## Disk file system; estimates space available on drive
```

Split

Created smaller files ## very useful with DNA/RNA data

e.g. FASTQ files = one DNA/RNA sequence per 4 lines

```
split -l 4000000 input.fastq    ## Generate files with one million sequences each
```

System usage

```
top                    ## returns top CPU heavy processes
who                    ## returns who is connected
```

Queuing systems (*e.g.* SGE, Moab) use bash scripts as inputs

Launching long processes

```
screen      ## allows disconnections and reconnections; source ~/.bash_profile
nohup       ## allows disconnections, runs in background
nice        ## sets priorities for a given task, i.e. play nice with other users
```

```
## Running hhblits on multiple evalues independently
run_hhblits.pl \
    -t 10 \
    -f FASTA_OL/ \
    -o HHBLITS/ \
    -d /media/Data_3/Uniclust/UniRef30_2020_06 \
    -e 1e-40 1e-10 1e-03 1e+01
```

https://github.com/PombertLab/3DFI

\ escapes metacharacters, we'll see more about it in the regular expressions section

Breaking down long cmd with \

Very long commands can be annoying to type on one line

You can split across lines using \ ## Type \ before hitting the newline key (Enter)

e.g.

```
top -u \
username \
jpombert
```