

## UNIX cheat sheet for CPSC 499 class, day 1

Command	English interpretation
ssh	open a remote, secure connection
passwd	change my password
script	save everything I type in a file
ctrl-z	stop a program
kill %1	get rid of it for good
fg %	start it up again
ls	list files and directories
ls -a	show hidden files
ls -lh	show details, such as file sizes
cd fred	change current directory to fred
cd ..	change to one directory "higher"
cd /	change to "root" directory
cd ~	go to home directory
mkdir alice	make a new directory, alice
rmdir fred	delete an existing directory, fred
pwd	print the directory you're in
cp text fred	copies the file "text" to a file called "text" in the directory fred: If there is no directory called fred: copies the file text to a <b>file</b> called fred <b>If there is already a file called fred: it will be automatically over-written!</b>
mv text fred	Move, not copy. Same thing, except it <b>deletes the original file. And same caveat.</b>
rm text	Deletes the file "text". <b>Forever.</b>
nano text	view or edit contents of file "text" nano is not on all UNIX computers, but is almost always available on Linux.
more text	see file "text" in screen-size chunks
less text	see file with arrows to move about
head text	see top of file "text"
tail text	see bottom of file "text"
gunzip	unzip a .gz compressed file
unzip	unzip a .zip compressed file
tar -xvf	un-tar a tar archive
man tar	get help about program "tar"
blastall	the blast command
nice -n 10	run a command with "nice" priority 10
ls > list	redirect output of "ls" to file "list" <b>file "list" will be overwritten if it already exists!</b>

## UNIX cheet sheet for CPSC 499 class, day 2

top	see all system processes
formatdb	make a BLAST database
clustalw	perform multiple sequence alignment
hmmpfam	search protein with pfam database
hmmbuild	make an hmm from an alignment
hmmsearch	search a protein file with an hmm
grep this text	find lines matching "this" in file text
ftp	get a remote file
ftp>cd	same as unix
ftp>ls	same as unix
ftp>get	get a remote file
ftp>put	deposit a file
sftp	same as ftp but more secure
wget	get from a web address
./configure	
make	GNU compile commands

```
export PATH=$PATH:/home/me/mydir/
```

adds mydir to your command path

chmod	change permissions of a file
-------	------------------------------

./myscript	run a script
------------	--------------

perl myscript.pl	run a perl script
------------------	-------------------

```
perl -e 'print "this is a command line script\n";'
```

## Perl cheet sheet, day 2

#!/usr/bin/perl	tells the UNIX shell this is a perl script
-----------------	--

print	output text to terminal (NOT printer!)
-------	--

" "	encloses a string (e.g. to print)
-----	-----------------------------------

;	end of command
---	----------------

\n	newline
----	---------

\	backslash, escape character
---	-----------------------------

## Perl cheat sheet, day 3

my	initialize a variable
+	add two things
-	subtract two things
*	multiply two things
/	divide two things
**	raise one thing to the power of another
\$this	a variable
=	set the variable equal to something
\n	a newline
\	the escape character
<> (<STDIN>)	input from the keyboard
chomp	remove a newline
if	if this is true, do something
{print "a";}	a conditional block (something to do)
(1 == 1)	an expression (something that might be true)
else	do something else if not true
elsif	do something else if not true and this is
==	is this number equal to that one?
!=	not equal?
>	bigger than?
>=	bigger than or equal to?
<	smaller than?
<=	smaller than or equal to?
eq	is this string the same as that one?
ne	is this string different from that one?
die	exit the program safely
@this	an array called this
split	split a variable into bits in an array
foreach	do something for everything in an array
for	do something for some numbers
substr	get me part of a string

## Perl cheat sheet, day 4

<code>%this</code>	a hash called this
<code>\$this{"that"}</code>	the part of the hash with key "that"
<code>(keys %this)</code>	the keys of the hash as an array
<code>open INFILE, "file"</code>	open a file for reading
<code>open OUTFILE, "&gt;file"</code>	open a file for writing
<code>close INFILE</code>	close the file
<code>while</code>	while an expression is true, do stuff
<code>&lt;INFILE&gt;</code>	read the file one line at a time
<code>\$_</code>	the current line in a loop
<code>=~</code>	binding operator
<code>/this/</code>	regular expression match
<code>.</code>	any character
<code>\d</code>	any digit
<code>\w</code>	any "word" character
<code>\s</code>	any space
<code>\t</code>	a tab
<code>^</code>	the beginning of a line
<code>\$</code>	the end of a line
<code>@ARGV</code>	where the arguments live
<code>use Bio::Perl</code>	use a module
<code>my @seq_object_array = read_all_sequences(\$file, 'fasta');</code>	get sequences from a fasta
<code>my \$sequence = \$object-&gt;seq();</code>	the sequence
<code>my \$name = \$object-&gt;display_id;</code>	the ID
<code>system</code>	ways of passing commands to the shell
<code>exec</code>	ways of passing commands to the shell
<code>` `</code>	ways of passing commands to the shell
<code>tr///</code>	translate
<code>s///</code>	substitute
<code>reverse</code>	reverse order of characters (in string)