UNIX BASICs

This exercise is designed to provide the basic skills required for working in the UNIX environment, using plenty of relevant examples, specifically for biologists. If you are using your personal computer, make sure that you have downloaded the files required for the workshop. This exercise will provide you information regarding navigation, files and directory creation/modification and some administrative things related to file permissions.

Naviagation

This section will introduce you to some basic file/directory navigation and manipulation techniques.

Changing directories

To jump from one directory to another we use the cd (*c*hange *d*irectory) command.

cd ..

Changes your present location to the parent directory

cd DIRECTORY

This changes your location back to your DIRECTORY.

**Task 1.1: Now change your directory to the** WORKSHOP\_FILES **directory present in your home directory.**

**NOTE:** You can type in first few letters of the directory name and then press tab to auto complete rest of the name (especially useful when the file/directory name is long). This only works when there are unique matches for the starting letters you have typed. If there is more than one matching files/directories, pressing tab twice will list all the matching names. You can also recall your previous commands by pressing up/down arrow or browse all your previously used commands by typing history on your terminal (typically, last 500 commands will be saved in this file).

To know the present location of your command

pwd

/home/username

Returns you the present working directory (*p*rint *w*orking *d*irectory)

This means, you are now working in the username directory, which is located in home directory. The directory that you will be in after logging in is your home directory. You can also avoid writing the full path by using ~ in front of your username or simply ~.

~ or ~username same as /home/username

Present directory is represented as . (dot) and parent directory is represented as .. (dot dot)

Viewing the contents of the directory

The contents of a directory can be viewed using ls (*l*i*s*t) command.

ls DIRECTORY *# now try it with* tutorials *directory*

If no directory name is provided then ls will list all the contents of the present directory.

Like any other command, you can use absolute path or abbreviated path. There are also various options available for ls command.

Some very useful options include:

ls –l

Lists all the files in lengthy or detailed view

ls –t

Lists all the files, sorted based on creation time

ls –S

Lists all the files, sorted based on size

You can also combine these options together for getting more focused results.

*Looking at the manual for ls, what option can you use to view hidden files in a directory (files starting with dot)? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Can you sort the files based on its extension? How? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**Task 1.5: Examine the contents of the** tutorials **directory. Try options such as** -l**,** -t**,** -a **and** -X**. Also check if you can combine many options together (like** -la **or** -lh **etc).** Try these:

ls -l tutorials

ls -a

Directories and Files

Copying Directories

To copy a file, cp (*c*o*p*y) command is used. When using this command you have to provide both source file and destination file.

cp SOURCE DESTINATION

You can also specify the absolute path of the source and/or destination file. To know more about any command you can use man command, which opens the manual of the command you ask (referred as 'man page').

man cp

This opens the manual for the cp command. Take a look at the manual of cp command (use arrow keys to move top or bottom of the page). OPTIONS are optional arguments that can be used to accomplish more from the same command. *Eg.,* by using option –i with the regular cp command, you can always make sure that you are not overwriting the existing file while copying. The syntax for using the options will also be provided in the manual. **To exit, press** q.

*Looking at the man page for cp command, what options can be used to copy a directory (including all files within it)? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*How else you can get help on cp command (other than ‘man’)? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**Task 1.3: Now change your directory back to the home directory. Create a copy of** WORKSHOP\_FILES **and name it as** BACKUP\_WORKSHOP**).** This will serve as a backup copy of all files that are required for the workshop (in case you accidentally modify the contents while working).

cp -r WORKSHOP\_FILES BACKUP\_WORKSHOP

Moving Directories

To move a file or a directory, mv (*m*o*v*e) command is used. Again, like the cp command you need to provide both source file and destination file.

mv SOURCE DESTINATION

Absolute path also works fine. Some of the options used by cp command also work with mv command. mv can also be used to rename files and directories

mv OLDNAME NEWNAME

**Task 1.4: Rename WORKSHOP\_FILES as tutorials.**

mv WORKSHOP\_FILES tutorials

ls -1 tutorials

ls -lh tutorials

ls -t tutorials

Making directories

To create a directory, mkdir (*m*a*k*e *dir*ectory) can be used.

mkdir DIRECTORY

Unlike PC/Mac folders, here you can’t have space in your directory name (but some special characters are okay). You can also specify the path where you want to create your new folder.

**Task 1.2: Make a new directory named** FirstDirectory **within the** WORKSHOP\_FILES **directory. Then change your directory to the** FirstDirectory**.**

mkdir FirstDirectory

Deleting files and directories

To delete directories from the system, you can use rmdir (*r*e*m*ove *dir*ectory) command. You can also use rm command to delete file(s).

rmdir DIRECTORY

The directory should be empty before you use the rmdir command.

rm FILE

To delete a file rm command can be used

Some useful options include

–r recursively delete files

-f delete forcefully

rm –rf DIRECTORY **[DO NOT USE THIS NOW!]**

When you want to delete a folder, with all its content

**Task 1.8: Delete the directory named** delete\_me **inside the** tutorials **directory (to do this you may first want to delete the** sample.txt **file inside this directory).**

cd delete\_me

rm sample.txt

cd ..

rmdir delete\_me

Creating and editing files

touch FILENAME

Creates a new file in the present location

nano FILENAME

Like notepad/textedit, this text editor lets you edit a file.

**Task 1.6: Create a new file named** firstfile **inside the** tutorials **directory. You can create using touch or using** nano**. Then add some contents (Your name and email address) to the** firstfile **(using** nano**). After editing, press** Ctrl + X **to exit, then enter** y **to save changes and confirm the file name**.

touch firstfile

nano firstfile

Viewing contents of the files

There are various commands to print the contents of the file in bash. Most of these commands are often used in specific contexts. All these commands when executed with filenames displays the contents on the screen. Most common ones are less, more, cat, head and tail.

less FILENAME *try this:* less AT\_cDNA.fa

Displays file contents on the screen with line scrolling (to scroll you can use arrow keys, PgUp/PgDn keys, space bar or Enter key). **When you are done press** q **to exit**.

more FILENAME *try this:* more AT\_cDNA.fa

Like less command, also, displays file contents on the screen with line scrolling but uses only space bar or Enter key to scroll. **When you are done press** q **to exit**.

cat FILENAME *try this:* cat AT\_cDNA.fa

Simplest form of displaying contents. It *cat*alogs the entire contents of the file on the screen. In case of large files, entire file will scroll on the screen without pausing

head FILENAME *try this:* head AT\_cDNA.fa

Displays only the starting lines of a file. The default is first ten lines. But, any number of lines can be displayed using –n option (followed by required number of lines).

tail FILENAME *try this:* tail AT\_cDNA.fa

Similar to head, but displays the last 10 lines. Again –n option can be used to change this.

More information about any of these commands can be found in man pages (man command)

**Task 1.7: Try using all these commands on the** RefSeq.faa**. You are also welcome to try these commands on various other files that are present in the** tutorials **directory.** These commands don’t change the contents of the file; they just display them on the screen.

compressing files

There are several options for archiving and compressing groups of files or directories. Compressed files are not only easier to handle (copy/move) but also occupy less size on the disk (less than 1/3 of the original size). In Linux systems you can use zip, tar or gz for archiving and compressing files/directories.

ZIP compression/extraction

zip OUTFILE.zip INFILE.txt

Compress INFILE.txt

zip -r OUTDIR.zip DIRECTORY

Compress all files in a DIRECTORY into one archive file (OUTDIR.zip)

zip -r OUTFILE.zip . -i \*.txt

Compress all txt files in a DIRECTORY into one archive file (OUTFILE.zip)

unzip SOMEFILE.zip

Decompress a file

**Task 1.9: Zip** AT\_genes.gff **file located in the** tutorials **directory. Check the file size before and after zip compression (Hint: use** ls –lh **to check file sizes).**

zip AT\_genes.gff.zip AT\_genes.gff

*Is there any size difference before and after compressing? Y/N*

tar (*t*ape *ar*chive) utility saves many files together into a single archive file, and restores individual files from the archive. It also includes automatic archive compression/decompression options and special features for incremental and full backups.

tar -cvf OUTFILE.tar INFILE

archive INFILE

tar -czvf OUTFILE.tar.gz INFILE

archive and compress file INFILE

tar -tvf SOMEFILE.tar

list contents of archive SOMEFILE.tar

tar -xvf SOMEFILE.tar

extract contents of SOMEFILE.tar

tar -xzvf SOMEFILE.tar.gz

extract contents of gzipped archive SOMEFILE.tar.gz

tar -czvf OUTFILE.tar.gz DIRECTORY

archive and compress all files in a directory into one archive file

tar -czvf OUTFILE.tar.gz \*.txt

archive and compress all ".txt" files in current directory into one archive file

**Task 1.10: Archive and compress the** BACKUP\_WORKSHOP **directory you created in Task 1.3 (you can name it as** backup.tar.gz **or anything you want)**

tar -czvf backup.tar.gz BACKUP\_WORKSHOP

gzip (*g*nu *zip*) compression utility designed as a replacement for compress, with much better compression and no patented algorithms. The standard compression system for all GNU software.

gzip SOMEFILE

compress SOMEFILE (also removes uncompressed file)

gunzip SOMEFILE.gz

uncompress SOMEFILE.gz (also removes compressed file)

**Task 1.11:** gzip **the file** AT\_genes.gff **and examine the size.** gunzip **it back so that you can use this file for the later exercises.**

gzip AT\_genes.gff

ls -lh

gunzip AT\_genes.gff.gz

ls –lh

Pipes and REDIRECTS

Many UNIX commands use some input file/data and display the output on the screen. This is feasible when the data being displayed is small enough to fit the screen or if it is the endpoint of your analysis. But for large data outputs, it is efficient to redirect to a file instead of screen. This can be done very easily in UNIX using > (greater than) or < (lesser than) or >> signs.

* < redirects the data to the command for processing
* > redirects the data from the command's output to a file. The file will be created if it is non-existing and if present it will overwrite the contents with the new output data (you will lose the original file).
* >> unlike > this redirection lets user append the data to an already existing file or a new file
* Another special operator | (called pipe) is used sometimes to pass the output from a command to another command (as input) before sending it to an output file or display.

Some *eg.*

cat FILE1 > FILE2

Creates a new file, FILE2 with same contents as old file, FILE1

cat FILE1 >> FILE2

Appends the contents for FILE1 to FILE2, equivalent to opening FILE1, copying all the contents, pasting the copied contents to the end of the FILE2 and saving it!

cat FILE1 | less

Here, cat command displays the contents of the FILE1, but instead of sending it to standard output (screen) it sends it through the pipe to the next command 'less' so that contents of the file are now displayed on the screen with line scrolling.

**Task 2.1: The Sequences directory contains a number of files and each of these files contain a single FASTA formatted nucleotide sequence. Combine them all together to make a single file sequences.fasta using redirects.**

cat \*.fa >> sequences.fasta

this command will combine all .fa files into one.

Regular Expressions

When working with the sequences (protein or DNA) we are often interested to see if a particular feature is present or not. This could be various things like a start codon, restriction site or even a motif. In UNIX all strings of text that follow some pattern can be searched using some formula called regular expressions. *eg.* If you are looking for a particular motif in large number of sequences, then you can create a regular expression in UNIX and search all the sequences having that motif relatively easily. Regular expression consists of normal and metacharacters. Commonly used characters include

|  |  |
| --- | --- |
| Expression | Function |
| . | matches any single character |
| $ | matches the end of a line |
| ^ | matches the beginning of a line |
| \* | matches one or more character |
| \ | quoting character, treat the next character followed by this as an ordinary character. |
| [] | matches one or more characters between the brackets |
| ? | match 1 occurrence of the character preceding |

For complete list, type info regex on your terminal.

Administrative Commands

Changing permissions

All files in the UNIX system will have a set of permissions which define what can be done with that file and by whom. (What = read (view contents), write (modify) and execute (run script) Whom=User (owner), group (that account belongs to) and everyone else). They are denoted as

**PERMISSIONS RELATIONS**

read r owner u

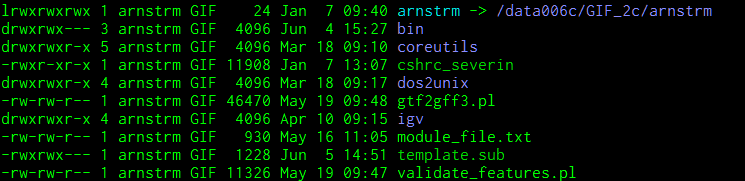
write w group g

execute x others o

all users a

To look at the permissions for any file, you can list the files with l option (ls –l).

Permissions User Group Size Date modified Name



u g o

(d=directory, l=link, r=read, w=write, x=execute, -=blank, u=user, g=group, o=others)

To set/modify a file's permissions you need to use the chmod command (*ch*ange *mod*e). Only the owner of a file can alter a file's permissions. The syntax:

chmod [OPTIONS] RELATIONS[+ or -]PERMISSIONS FILE

Add permissions

chmod RELATIONS+PERMISSIONS FILENAME

chmod g+rwx FILENAME grants read, write and execute permissions for group

chmod g+r FILENAME grants read permission for group

chmod a+rwx FILENAME makes the file public (don’t do this to any file/directory unless you want to share)

Remove permissions

chmod RELATIONS-PERMISSIONS FILENAME

chmod g-wx FILENAME removes write and execute permissions for group

chmod g-rwx FILENAME removes all permissions for group

chmod a-rwx FILENAME removes all permissions for others

chmod a-x FILENAME removes execution permissions for others

OPTIONS include

-R recursively (the permissions are applied to all the files, directories present inside the directory)

**Task 1.12: Check the permissions for the files located in the** tutorials **directory. Do**

ls -l

*What permissions does the group have on these files? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Which group does your account belong to? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

TR

The tr (*tr*anslate) utility in UNIX can translate or transliterate the input to produce a modified output. It uses 2 sets of parameters, and replaces occurrences of the characters in the first set with the corresponding elements from the other set.

tr [OPTIONS] "STRING1" "STRING2" <INFILE >OUTFILE

Useful options are

-c complements the set of characters specified by string1

-d delete occurrences of string1 (string2 not needed)

-s squeeze repeats or multiple occurrences found in string1 will be replaced with one string2

Common uses of tr command are:

tr "a-z" "A-Z" or tr "[:lower:]" "[:upper:]"

Convert lower case to upper case (or upper to lower case)

tr -s '\n'

Convert each sequence of repeated newlines to a single newline

Files generated in both Mac and Windows OS will have a different newline character (to mark the end of the line) that is not recognized by the UNIX OS. Similarly files generated in UNIX will have a different newline that can't be read in Windows or Mac OS. The 'tr' command provides an easy way to convert these 'newlines' to different forms.

tr '\r' '\n' <MAC.TXT >UNIX.TXT

Convert Mac text file to UNIX text file

tr '\n' '\r' <UNIX.txt >MAC.TXT

Convert UNIX text file to MAC text file

tr -d '\015' <WIN.TXT >UNIX.TXT

Convert Windows text file to UNIX text file

tr '\n' '\015'<UNIX.txt >WIN.TXT

Convert UNIX text file to windows text file

NOTE: There are in built commands like dos2unix, mac2unix and unix2dos to do these conversions automatically in most recent versions of UNIX.

There are several utilities that can mask low complexity regions of the genomes such as repeats. They do that either by converting the bases/residues to lower case (soft masking) or converting them to N or X (hard masking). The public databases often store these soft masked genomes. When downloaded it might be useful to remove the masking, if your analysis doesn't require it (pattern searching etc.). It can be easily done by changes cases

tr "ATGC" "atgc" <AT\_cDNA.fa >AT\_cDNA\_tr.fa

Converts masking from the sequences and saves them in a new file

tr "ATGC" "AUGC" <AT\_cDNA.fa > AT\_rna.fa

Converts cDNA to mRNA sequence and saves them in a new file

Word count

wc (*w*ord *c*ount) is another useful command that lets you count the number of words (and lines) in a file

wc FILENAME *try this* wc AT\_genes.gff

This outputs both number of words as well as lines in a file.

wc -l FILENAME *try this* wc -l AT\_genes.gff

Outputs only number of lines in file

Often these commands are "piped" with other commands to count certain things. *eg.*: Counting the number of files in a directory, counting the number of sequences etc.

**Task 2.6: Count how many files with .fa extensions are present in** sequences **directory.**

ls Sequences| wc -l

Sort

sort command can be used to arrange things in a file. Simplest way to use this command is:

sort FILE1 > SORTED\_FILE1

Useful options include

-n numerical sort

-r reverse sort

-k N,N sort the Nth field (column), where N is a number. Sorting can also be done on the exact character on a particular field *eg.* -k 4.3,4.4 sorts based on 3rd and 4th character of the 4th field. Additionally you can supply additional -k for resolving ties.

-t specify the delimiters to be used to identify fields (default is TAB) *eg.* -t : to use ':' as delimiter

**Task 2.7: The** Sequences **directory consists of numerically labeled files. UNIX can sort either alphabetically or numerically (not both) and hence they are arranged in** NT1.fa**,** NT10.fa**,** NT11.fa ***etc*. In order to sort them in an easy to read way, try using**

ls |sort -n -k 1.3,1.4

This command lists all the files in sequences directory and then passes it to sort command. Sort command then sorts it numerically but only using 3rd and 4th letters of the first field (file name)

Try using sort on AT\_genes.gff file

sort -r -k 1,1 AT\_genes.gff

sort -r -k 4,4 AT\_genes.gff

Uniq

uniq (*uniq*ue) command removes duplicate lines from a sorted file, retaining only one instance of the running matching lines. Optionally, it can show only lines that appear exactly once, or lines that appear more than once. uniq requires sorted input since it compares only consecutive lines.

uniq [OPTIONS] INFILE OUTFILE

Useful options include

-c count; prints lines by the number of occurrences

-d only print duplicate lines

-u only print unique lines

-i ignore differences in case when comparing

-s N skip comparing the first N characters (N=number)

**Task 2.8: Number each lines based on number of occurrences:**

uniq -c ids.txt

**Print only duplicated lines.**

uniq -d ids.txt

**Print only unique lines.**

uniq -u ids.txt

Comparing files

diff (*diff*erence) reports differences between files. A simple example for diff usage would be

diff [OPTIONS] FILE1 FILE2

Useful options include

-b ignore blanks

-w ignore white space (spaces and tabs)

-i ignore case

-r recursively compare all files (when comparing folders)

-s list all similar files (when comparing folders)

-y side by side comparison of files

The differences reported will be in the form of corrections that are required to change the first file to second file

Generate diffIDs.txt by comparing the differences between ids\_a.txt and ids\_b.txt

diff -y ids\_a.txt ids\_b.txt > diffIDs.txt

*Are these files different?*

comm (*comm*on) command compares two sorted files line by line.

comm [OPTIONS] FILE1 FILE2

-1 suppress lines unique to FILE1

-2 suppress lines unique to FILE2

-3 suppress lines that appear in both files

**Task 2.9: Compare the same files (ids\_a.txt and ids\_b.txt) again with 'comm' command and see how the outputs differ**

comm -1 ids\_a.txt ids\_b.txt

comm -2 ids\_a.txt ids\_b.txt

comm -3 ids\_a.txt ids\_b.txt

DIVIDING FILES

cut divides the file into several parts and displays selected columns or fields from each line of a file. Normally cut command requires how the fields are separated and what fields needs to be displayed.

cut -d "," -f 2-4 FILE displays columns 2,3 and 4 of a file separated by ","

cut -d "|" -f 1,10 FILE displays 1st and 10th columns of a file separated by "|"

cut -f 1 FILE displays 1st column of a file, assumes TAB as delimiter

split generates output files of a fixed size (bytes or lines). Useful when huge file needs to be processed. *eg.,*

split -d -l 100 FILENAME SUFFIX

here -d specifies numeric suffix only (suffix00, sufix01, suffix02 *etc*.) while -l specifies number of lines in each file (100 in this case). If you want to split based on bytes, you can use -b option (*eg*. -b 1k or -b 1m for 1 KB and 1 MB respectively)

*From the commands that you have learned, can you combine the all the split files into a single file again?*

**Task 2.10: Display only first column of the** AT\_genes.gff **file using cut**

cut -f 1 AT\_genes.gff (press ctrl + c to exit) or

cut -f 1 AT\_genes.gff | less

**Similarly, display 1st, 4th and 5th column of the** AT\_genes.gff **file**

cut -f 1,4,5 AT\_genes.gff (press ctrl + c to exit) or

cut -f 1,4,5 AT\_genes.gff | less

**Verify if all the columns in** AT\_genes.gff **file has same number of entries in every field**

cut -f 1 AT\_genes.gff |wc -l

cut -f 2 AT\_genes.gff |wc -l

cut -f 3 AT\_genes.gff |wc -l

**Split the file** AT\_genes.gff **every 100,000 lines. Use** gff\_split **as suffix for the files and use numerical suffix.**

split -d -l 100000 AT\_genes.gff gff\_split

*How many split files are generated: \_\_\_\_\_\_\_\_\_*

ls gff\_split\* |wc -l

COMBINING files

paste prints lines consisting of sequentially corresponding lines of each specified file. eg.,

paste FILE1 FILE2 > FILE3

Combines the contents of FILE1 and FILE2, side by side generating a new file, FILE3.

**Task 2.11: Combine columns of** ids\_a.txt **and** ids\_b.txt **files.**

paste ids\_a.txt ids\_b.txt

*How many columns do you see after combining? \_\_\_\_\_\_\_\_\_\_\_\_*

join combines two files based on the common field that is specified

join -t':' -1 N -2 N FILE1 FILE2

-t':' Specify field separator (here ":" but you can specify anything. Default is TAB)

-1 N Common field number (N) from the 1st file

-2 N Common field number (N) from the 2nd file

**Task 2.12: Join columns based on column 1 in** genes\_a.gff **and column 3 in** genes\_b.gff

join -1 1 -2 3 genes\_a.gff genes\_b.gff

Downloading data

In order to start using the computational power of the HPC cluster, you need to first get the data there. If your data is already in your local computer, you can transfer them easily using WinSCP software or any other software (refer prerequisites). But if the data that you will be using is available in the public databases then you can directly get it from there using curl or wget command (*W*WW *get*)

To download data from NCBI Sequence Read Archive (SRA) or genomics core website or any other website:

curl -O http://website.url

If using wget then,

wget http://website.url

As an example, we will download *Glycine max* (soy bean) annotation information file from Phytozome DB.

curl -O http://goo.gl/CDXx15

This is a single line command and you will see ‘Gmax\_189\_annotation\_info.txt.gz’ file after few seconds. You can extract it and view it or delete it using the commands you have learnt.

Quick Reference Sheet

Commands used in this manual

|  |  |  |
| --- | --- | --- |
| ***Command*** | ***Function*** | ***Syntax/example usage*** |
| *Navigation* | | |
| ls | list contents | ls [OPTIONS] DIRECTORY |
| pwd | print working directory | pwd |
| cd | change directory | cd ~ or cd *#home directory*  cd .. *#previous (parent directory)* |
| *File/Directory operations* | | |
| mkdir | make directory | mkdir DIRECTORY |
| cp | copy files/directories | cp SOURCE DESTINATION |
| man | manual page (help) | man COMMAND |
| mv | move files/directories | mv SOURCE DESTINATION |
| touch | create file | touch FILE |
| nano | edit file | nano FILE |
| less | view file (with more options) | less FILE |
| more | view file (with less options) | more FILE |
| cat | catalog file contents | cat FILE |
| head | show first few lines of a file | head FILE |
| tail | show last few lines of a file | tail FILE |
| rmdir | remove empty directory | rmdir DIRECTORY |
| rm | remove file(s) | rm FILE |
| *Compression/archiving* | | |
| zip | zip compress | zip OUTFILE.zip INFILE.txt  zip -r OUTDIR.zip DIRECTORY |
| unzip | decompress zipped file | unzip ANYTHING.zip |
| tar | archive and compress files/directories | tar -czvf OUTFILE.tar.gz DIRECTORY *#compress*  tar -xzvf OUTFILE.tar.gz *# extract* |
| gzip | gzip files | gzip SOMEFILE |
| gunzip | decompress gzipped files | gunzip SOMEFILE.gz |
| *File permissions* | | |
| chmod | change permissions for files/directories | chmod [OPTIONS] RELATIONS[+/-]PERMISSIONS FILE |
| *File manipulations* | | |
| grep | search a pattern | grep [OPTIONS] "PATTERN" FILENAME |
| sed | stream edit a file | sed 's/search/replace/g' FILENAME |
| awk | multi-purpose command | awk 'PATTERN {ACTION}' FILENAME |
| tr | translate or transliterate a file | tr [OPTIONS] "STRING1" "STRING2" <INFILE |
| wc | word count | wc FILENAME |
| sort | sort files | sort FILE1 > SORTED\_FILE1 |
| uniq | display unique lines | uniq [OPTIONS] INFILE > OUTFILE |
| diff | display difference | diff [OPTIONS] FILE1 FILE2 |
| comm | display common lines among files | comm [OPTIONS] FILE1 FILE2 |
| cut | break files vertically based on fields | cut –d "DELIMITER" –f NUMBER FILE |
| split | break files horizontally | split [OPTIONS] FILENAME |
| paste | combine files side by side | paste FILE1 FILE2 > FILE3 |
| join | join files based on common field | join -t'DELIMITER' -1 N -2 N FILE1 FILE2 |

Additional commands

|  |  |
| --- | --- |
| ***Command*** | ***Function*** |
| du –sh DIR | show directory size |
| whoami | display username |
| date | system date/time |
| cal | calendar |
| find . –name FILE | find a file/directory |
| which CMD | display default cmd path |
| whereis CMD | show possible locations of cmd |
| locate FILE | find instances of a file |
| clear | clear screen |
| sleep 5 | pause 5 (any) seconds |
| top | current running processes |
| ps | current running processes |
| wget URL | download specified URL |

Nano shortcuts

|  |  |
| --- | --- |
| ***Commands*** | ***Function*** |
| ctrl+r | read/insert file |
| ctrl+o | save file |
| ctrl+x | close file |
| alt+a | start selecting text |
| ctrl+k | cut selection |
| ctrl+u | uncut (paste) selection |
| alt+/ | go to end of the file |
| ctrl+a | go to start of the line |
| ctrl+e | go to end of the line |
| ctrl+c | show line number |
| ctrl+\_ | go to line number |
| ctrl+w | find matching word |
| alt+w | find next match |
| ctrl+\ | find and replace |

Pre-declared variables

|  |  |
| --- | --- |
| ***Variables\**** | ***Description*** |
| $USER | username |
| $HOME | home path |
| $PWD | working directory path |
| $PATH | path for executables |
| $HOSTNAME | machine name |
| $SHELL | current shell |
| $SSH\_CLIENT | local client's IP address |
| $TERM | type of terminal |
| *\** env *command lists all the assigned variables* | |

Shortcuts

|  |  |
| --- | --- |
| ***Commands*** | ***Function*** |
| TAB | autocomplete names |
| UP/DOWN | browse previous commands |
| ctrl+c | interrupt/kill anything |
| ctrl+l | clear screen |
| ctrl+d | quit, exit |
| ctrl+z | suspend (use fg to restore) |
| !! | repeat last command |
| alt+. | last argument of previous cmd |
| ctrl+insert | copy selection |
| shift+insert | paste copied text |
| ctrl+a | go to start of the line |
| ctrl+e | go to end of the line |
| ctrl+r | reverse search history |
| cd ~ | go to home |

Pipes, redirects

|  |  |
| --- | --- |
| ***Redirects*** | ***Function*** |
| cmd < file | use file as input |
| cmd > file | write output to file |
| cmd >> file | append output to file |
| cmd 2> stderr | error output to file |
| cmd 1>&2 file | send output and error to file |
| cmd1 | cmd2 | send output of cmd1 to cmd2 |

HPC-Cluster specific commands

|  |  |  |
| --- | --- | --- |
| ***Command*** | ***Function*** | ***Syntax/example usage*** |
| qstat | lists current jobs on queues | qstat –an1 queue *# current jobs in specified queue*  qstat –u username *# current jobs by the user*  qstat –f jobid *# information about the job (id#)*  qstat -q *# list all queues*  qstat -a *# list all jobs*  qstat –r *# list running jobs*  qstat -Qf queue *# information about the 'queue'* |
| qdel | delete job from the queue | qdel jobid |
| qsub | submit job to the queue | qsub submissionfile.sub |
| qhold | hold job in queue | qhold jobid |
| qrls | release job for running | qrls jobid |
| qnodes | details about the nodes | qnodes |
| module | use preinstalled programs | module load PROGRAM *# loads program for use*  module list *# lists all loaded modules*  module avail *# lists available modules*  module unload PROGRAM *# unloads module* |

PS: An A-Z Index of the Bash command line for Linux can be found at <http://ss64.com/bash/index.html>