Adapted from
http://manuals.bioinformatics.ucr.edu/home/linux-basics#TOC-Orientation

1.Viewing and changing the present working directory:

```
pwd               # Get full path of the present working directory (same as "echo $HOME")

ls                # Content of pwd
ls -l             # Similar as ls, but provides additional info on files and directories
ls -a             # Includes hidden files (.name) as well
ls -R             # Lists subdirectories recursively
ls -t             # Lists files in chronological order

cd <dir_name>     # Switches into specified directory.
cd                # Brings you to the highest level of your home directory.
cd ..             # Moves one directory up
cd ../../         # Moves two directories up (and so on)
cd -             # Go back to you were previously (before the last directory change)
```

2.Files and directories

```
mkdir <dir_name>   # Creates specified directory
rmdir <dir_name>   # Removes empty directory
rm <file_name>     # Removes file name
rm -r <dir_name>   # Removes directory including its content, but asks for confirmation,
                   # 'f' argument turns confirmation off
cp <name> <path>   # Copy file/directory as specified in path (-r to include content in directories)
mv <name1> <name2> # Renames directories or files
mv <name> <path>   # Moves file/directory as specified in path
```

3.Text viewing

```
more <my_file>  # views text, use space bar to browse, hit 'q' to exit
less <my_file>  # a more versatile text viewer than 'more', 'q' exits, 'G' moves to end of text, 'g' to be
cat  <my_file>  # concatenates files and prints content to standard output
```

4.Vim Manual

Basics

```
vim my_file_name # open/create file with vim
```

Once you are in Vim the most important commands are `i` , `:` and `ESC`. The `i` key brings you into the insert mode for typing. The `ESC` brings you out o
starting with `:` need to be typed in the command mode. All other commands are typed in the normal mode after hitting the `ESC` key.

Modifier Keys to Control Vim

- `i # INSERT MODE`
- `ESC # NORMAL (NON-EDITING) MODE`
- `: # commands start with ':'`
- `:w # save command; if you are in editing mode you have to hit ESC first!!`
- `:q # quit file, don't save`
- `:q! # exits WITHOUT saving any changes you have made`
- `:wq # save and quit`
- `R # replace MODE`
- `r # replace only one character under cursor`
- `q: # history of commands (from NORMAL MODE!), to reexecute one of them, select and hit enter!`
- `:w new_filename # saves into new file`
- `:#,#w new_filename # saves specific lines (#,#) to new file`
- `:# go to specified line number`

Useful shell commands

```
cat <file1> <file2> > <cat.out>      # concatenate files in output file 'cat.out'
paste <file1> <file2> > <paste.out>  # merges lines of files and separates them by tabs (useful for table
cmp <file1> <file2>                  # tells you whether two files are identical
diff <fileA> <fileB>                 # finds differences between two files
head -<number> <file>                # prints first lines of a file
tail -<number> <file>                # prints last lines of a file
split -l <number> <file>             # splits lines of file into many smaller ones
csplit -f out fasta_batch "%^>%" "/^>/" "{*}" # splits fasta batch file into many files
                                     # at '>'
sort <file>                          # sorts single file, many files and can merge (-m)
                                     # them, -b ignores leading white space, ...
```

```
sort -k 2,2 -k 3,3n input_file > output_file # sorts in table column 2 alphabetically and
                                             # column 3 numerically, '-k' for column, '-n' for
                                             # numeric
```

```
sort input_file | uniq > output_file # uniq command removes duplicates and creates file/table
                                     # with unique lines/fields
join -1 1 -2 1 <table1> <table2>     # joins two tables based on specified column numbers
                                     # (-1 file1, 1: col1; -2: file2, col2). It assumes
                                     # that join fields are sorted. If that is not the case,
                                     # use the next command:
sort table1 > table1a; sort table2 > table2a; join -a 1 -t "`echo -e '\t'`" table1a table2a > table3
                                     # Default prints only all lines the two tables have in
                                     # common. '-t "`echo -e '\t'`" ->' forces join to
                                     # use tabs as field separator in its output. Default is
                                     # space(s)!!!
cat my_table | cut -d , -f1-3        # cut command prints only specified sections of a table,
                                     # -d specifies here comma as column separator (tab is
                                     # default), -f specifies column numbers.
grep                                 # see chapter 4
egrep                                # see chapter 4
```