Introduction to Shell

Basic Commands

`pwd` → "**P**rint **W**orking **D**irectory"

- `pwd` (Print Working Directory)

`ls` → "Li**s**ting"

- `ls` (Listing on the Working Directory)
- `ls /home/folder` (Listing of a directory)
- `ls folder`
- `ls -R` (Recursive → shows everything)

`cd` → "**C**hange **D**irectory"

- `cd folder`
- `cd /home/folder`

Absolute Path vs Relative Path → The shell decides if a path is absolute or relative by looking at its first character: If it begins with /, it is absolute. If it does not begin with /, it is relative.

`..` → "the directory above the one I'm currently in"

`.` → "the current directory"

`~` → "your home directory"

`cp` → "Copy"

- `cp original.txt duplicate.txt`
- `cp folder/original.txt backup/duplicate.txt`

`mv` → "Move"

- `mv original.txt backup`
- `mv course.txt old-course.txt` (Rename file)

`rm` → "Remove"

- `rm sometext.txt`
- `rm folder/sometext.txt`

`mkdir` → "Make directory"

`mkdir folder`

`rmdir` → “Remove directory”

`rmdir folder`

`history` → (See all history commands)

`history`

`!34` (Rerun command 34 in the history)

`!head` (Rerun last head command)

Manipulating Data

`cat` → “Concatenate” (Read File)

`cat sometext.txt`

`less` → Read file :n nextPage :q

`less sometext1.txt sometext2.txt`

`head` → View first 10 lines

`head somecsv.csv`

`head -n 5  somecsv.csv` (Show only five lines)

`tail` → view Last 10 lines

`man` → Manual

`cut` → Select Columns

`cut -f 2-5,8 -d , values.csv` (-f “fields”, -d “delimiter”)(This comand shows columns 2-5 and 6 with delimiter “,”)

`wc` → Word count / Count lines (-l lines, -w words, -c caracters)

`wc -l sometext.txt` (Count lines in file)

`grep` → global regular expression print

- `c`: print a count of matching lines rather than the lines themselves
- `h`: do *not* print the names of files when searching multiple files
- `i`: ignore case (e.g., treat "Regression" and "regression" as matches)
- `l`: print the names of files that contain matches, not the matches

- `n`: print line numbers for matching lines
- `v`: invert the match, i.e., only show lines that *don't* match

`grep molar seasonal/winter.csv` (print all lines with molar)

`grep -v -n molar seasonal/winter.csv` (print all lines without molar and show lines number)

`sort` → Sort values

`cat sometext.txt | sort -r` (Show lines in reverse alphabetical order)

`uniq` → Unique values

`cat sometxt.txt | sort | uniq -c` (Show all lines with unique values and count them)

Combining tools

`>` → Store/redirect a command's output in a file.

`head -n 5 seasonal/summer.csv > top.csv` (Created a file with comand output)

`> result.txt | head -n seasonal/summer.csv`

`|` → Combine commands (Pipe) Use the output of the command on the left as the input to the command on the right.

`head -n 5 seasonal/summer.csv | tail -n 3` (Print lines 3-5)

`*` → Wildcard (Match zero or more characters)

`cut -d , -f 2 seasonal/s*.csv` (Use cut in all files inside seasonal folder that start with s and end with .csv)

`?` → Wildcard Match a single character

`head -n 2 201?.txt` (Match with 2018.txt, 2019.txt but not 2011-2.txt)

`[...]` → Wildcard Matches any one of the characters inside the square brackets

`head -n 2 201[5,6].txt` (Match with 2015 and 2016.txt)

`{...}` → Wildcard Matches any of the comma-separated patterns inside the curly brackets

`head -n {2*.txt, 2*.csv}` (Match with 2015.txt and 2015.csv)

Batch processing

environment variables

`echo $HOME` (Print variable value inside home)

`echo` → print

`echo hello world`

shell variable

`testing=seasonal/winter.csv` (Creating variables)

`head -n 1 $testing` (Head -n 1 use variables)

`for x in pdf odf docx; do echo $x; done` (Prints pdf, odf, docx)

`for file in seasonal/* ; do echo $file ; done` (Prints all files inside folder)

`for f in seasonal/*csv; do grep 2017 $f | tail -n 1; done` (Print last value of 2017 of each file in folder)

Creating new tools

`nano` → File editing

`nano filename`

- `Ctrl` + `K`: delete a line.
- `Ctrl` + `U`: un-delete a line.
- `Ctrl` + `O`: save the file ('O' stands for 'output'). *You will also need to press Enter to confirm the filename!*
- `Ctrl` + `X`: exit the editor.

Example save commands : `history | tails -n 3 > steps.txt`

Example creating simples sh script :

`vim dates.sh`

`cut -d , -f 1 seasonal/*.csv`

`bash dates.sh`

`$@` → Special expression to pass arguments to script

```
vim cols.sh
```

```
cut -d , -f $2 $1
```

```
bash cols.sh seasonal/*.csv 2
```