UNIX Commands

1. Basic Linux Command

- `pwd`: print working directory
- `ls`: list files and directories
- `cd <directory>`: change directory
- `mkdir <directory>`: make a new directory
- `touch <file>`: make a new file
- `cp <file> <directory>`: copy file(s) to directory
- `mv <file> <directory>`: move or rename file(s)
- `rm <file>`: delete file(s)
- `curl -O <url/to/file>`: download

1.1 Exercises

- Use the command `pwd` to see what's your current directory

```
pwd
```

- Use `mkdir` to create a new directory `stat`

```
mkdir stat
```

- Change directory to `stat`

```
cd stat
```

- Use the command `curl` to download the following text file: http://textfiles.com/food/bread.txt

```
curl -O http://textfiles.com/food/bread.txt
```

- Use the command `ls` to list the contents in your current directory

```
ls
```

- Use the command `curl` to download these other text files:
  - http://textfiles.com/food/btaco.txt
  - http://textfiles.com/food/1st_aid.txt
  - http://textfiles.com/food/beesherb.txt
  - http://textfiles.com/food/bakebred.txt

```
curl -O http://textfiles.com/food/btaco.txt
curl -O http://textfiles.com/food/1st_aid.txt
curl -O http://textfiles.com/food/beesherb.txt
curl -O http://textfiles.com/food/bakebred.txt
```

- Use the command curl to download the following csv files:
  - http://archive.ics.uci.edu/ml/machine-learning-databases/forest-fires/forestfires.csv
  - http://web.pdx.edu/~gerbing/data/cars.csv
  - http://web.pdx.edu/~gerbing/data/color.csv
  - http://web.pdx.edu/~gerbing/data/snow.csv
  - http://web.pdx.edu/~gerbing/data/mid1.csv
  - http://web.pdx.edu/~gerbing/data/mid2.csv
  - http://web.pdx.edu/~gerbing/data/minutes1.csv
  - http://web.pdx.edu/~gerbing/data/minutes2.csv

```
curl -O http://archive.ics.uci.edu/ml/machine-learning-databases/forest-fires/forestfires.csv
curl -O http://web.pdx.edu/~gerbing/data/cars.csv
curl -O http://web.pdx.edu/~gerbing/data/color.csv
curl -O http://web.pdx.edu/~gerbing/data/snow.csv
curl -O http://web.pdx.edu/~gerbing/data/mid1.csv
curl -O http://web.pdx.edu/~gerbing/data/mid2.csv
curl -O http://web.pdx.edu/~gerbing/data/minutes1.csv
curl -O http://web.pdx.edu/~gerbing/data/minutes2.csv
```

- Use the command `ls` to list the contents in your current directory

```
ls
```

- Now try `ls -l` to list the contents in your current directory in long format

```
ls -l
```

- Look at the `man` documentation of `ls` to find out how to list the contents in reverse order

```
man ls -r
```

- How would you list the contents in long format arranged by time?

```
ls -lt
```

- Find out how to use the wildcard `*` to list all the files with extension `.txt`

```
ls *.txt
```

- Use the wildcard `*` to list all the files with extension `.csv` in reverse order

```
ls -r *.csv
```

- You can use the character `?` to represent a single character: e.g. `ls mid?.csv`. Find out how to use the wilcard `?` to list `.csv` files with names made of 4 characters (e.g. `mid1.csv`, `snow.csv`)

```
ls ????.csv
```

- The command `ls *[1]*.csv` should list `.csv` files with names containing the number `1` (e.g. `mid1.csv`, `minutes1.csv`). Adapt the command to list `.csv` files with names containing the number `2`.

```
ls *[2]*.csv
```

- Find out how to list files with names containing any number.

```
ls *[0-9]*.*
```

- Invoke the command `history` and see what happens.

`history` will show you the commands that have been used so far.

- Inside `stat` create a directory `data`

```
mkdir data
```

- Change directory to `data`

```
cd data
```

- Create a directory `txt-files`

```
mkdir txt-files
```

- Create a directory `csv-files`

```
mkdir csv-files
```

- Use the command `mv` to move the `bread.txt` file to the folder `txt-files`. Without changing directories, use `ls` to confirm that `bread.txt` is now inside `txt-files`.

```
mv ../bread.txt txt-files
ls txt-files
```

- Use the wildcard `*` to move all the `.txt` files to the directory `txt-files`. Without changing directories, use `ls` to confirm that all the `.txt` files are inside `txt-files`.

```
mv ../*.txt txt-files
ls txt-files
```

- Use the wildcard `*` to move all the `.csv` files to the directory `csv-files`. Without changing directories, use `ls` to confirm that all the `.csv` files are inside `csv-files`.

```
mv ../*.csv csv-files
ls csv-files
```

- Go back to the parent directory `stat`

```
cd ..
```

- Create a directory `copies`

```
mkdir copies
```

- Use the command `cp` to copy the `bread.txt` file (the one inside the folder `txt-files`) to the `copies` directory

```
cp data/txt-files/bread.txt copies
```

- Without changing directories, use `ls` to confirm that `bread.txt` is now inside `copies`.

```
ls copies
```

- Use the wildcard `*` to copy all the `.txt` files in the directory `copies`

```
cp data/txt-files/*.txt copies
```

- Without changing directories, use `ls` to confirm that all the `.txt` files are now inside `copies`.

```
ls copies
```

- Use the wildcard `*` to copy all the `.csv` files in the directory `copies`

```
cp data/csv-files/*.csv copies
```

- Change to the directory `copies`

```
cd copies
```

- Use the command `mv` to rename the file `bread.txt` as `bread-recipe.txt`

```
mv bread.txt bread-recipe.txt
```

- Rename the file `cars.csv` as `autos.csv`

```
mv cars.csv autos.csv
```

- Rename the file `btaco.txt` as `breakfast-taco.txt`

```
mv btaco.txt breakfast-taco.txt
```

- Change to the parent directory (i.e. `stat`)

```
cd ..
```

- Rename the directory `copies` as `copy-files`

```
mv copies copy-files
```

- Find out how to use the `rm` command to delete the `.csv` files that are in `copy-files`

```
rm copy-files/*.csv
```

- Find out how to use the `rm` command to delete the directory `copy-files`

```
rm -r copy-files
```

- List the contents of the directory `txt-files` displaying the results in reverse (alphabetical) order

```
ls -r data/txt-files
```

2. Unix Filters and Pipes Basic Bash Command (*B*ourne *A*gain *Sh*ell)

- `cut` allows you to select columns
- `paste` allows you to merge columns
- `sort` can be used to arrange lines
- `sort` can also be used to group by lines
- `sort` and uniq can be used to count occurrences
- `grep` allows you to filter rows

Redirect `>`: save command output to a new file (existing file will be overrided) *Redirect* `>>` : update command output to a file (added after last line of existing file)

2.1. Unix Filters Cheat Sheet

Command	Description	R alternative
`wc nba2017-players.csv`	count lines, words, and bytes	`object.size()`, `nrow()`
`wc -l nba2017-players.csv`	count number of lines	`nrow()`
`head nba2017-players.csv`	inspect first 10 rows	`head()`
`tail nba2017-players.csv`	inspect last 10 rows	`tail()`
`less nba2017-players.csv`	see contents with a paginator	`View()`

Extracting columns with **`cut`**

Option	Description
`-f` 1,5	return columns 1 and 5, delimited by tabs.
`-f` 1-5	return columns 1 through 5, delimited by tabs.
`-d ","`	use commas as the delimiters.
`-c 2-7`	return characters 2 through 7 from the file.

Sorting lines with **`sort`**

Option	Description
`-n`	sort in numerical order rather than alphabetically.
`-r`	sort in reverse order, z to a or decreasing numbers.
`-f`	fold uppercase into lowercase (i.e. ignore case).
`-u`	return a unique representative of repeated items.
`-k 3`	sort lines based on column 3 (tab or space delimiters)
`-t ","`	use commas for delimiters.
`-b`	ignore leading blanks.

Option	Description
`-d`	sort in dictionary order.

Isolating unique lines with **`uniq`**

Option	Description
`-c`	adds a count of how many times each line occurred.
`-u`	lists only lines that are not repeated.
`-d`	lists only lines that are duplicated.
`-i`	ignore case when determining uniqueness
`-f 4`	ignore the first 4 fields (space delimiter)

Showing first 10 lines with **`head`**

Option	Description
`-3`	shows first 3 lines (same as `-n 3`)
`-n -3`	shows first lines up to last 3 lines
`-v`	shows file names

Showing last 10 lines with **`tail`**

Option	Description
`-3`	shows last 3 lines (same as `-n 3`)
`-n +3`	shows lines after the 3rd line
`-f`	keeps reading lines
`-F`	make a new file if change occurs
`-r`	reverse contents
`-v`	shows file names

Counting number of lines, words, bytes with **`wc`**

Option	Description
`-c`	counts bytes
`-l`	counts number of lines
`-w`	counts number of words

2.2. Exercises
The data set we use: `tents.csv`

```
             name     brand  price weight height     bestuse  seasons capacity
1   fly-creek-ul2 big-agnes 349.95    960     96 Backpacking 3-season 2-person
2   fly-creek-ul3 big-agnes 449.95   1450    107 Backpacking 3-season 3-person
3        salida-2     kelty 159.95   1700    102 Backpacking 3-season 2-person
4 jack-rabbit-sl3 big-agnes 359.95   2160    107 Backpacking 3-season 3-person
5       passage-2       rei 149.00   2210    107 Backpacking 3-season 2-person
6 copper-spur-ul2 big-agnes 399.95   1530    107 Backpacking 3-season 2-person
```

- Display names of first 5 tents

```
cut -f 1 -d "," tents.csv | tail +2 | head -n 5
```

```
## "fly-creek-ul2"
## "fly-creek-ul3"
## "salida-2"
## "jack-rabbit-sl3"
## "passage-2"
```

- Redirect the previous operation with redirect operator `>` to the text file `name5.txt`

```
cut -f 1 -d "," tents.csv | tail +2 | head -n 5 > name5.txt
```

- Confirm that `name5.txt` has the tent names

```
head name5.txt
```

```
## "fly-creek-ul2"
## "fly-creek-ul3"
## "salida-2"
## "jack-rabbit-sl3"
## "passage-2"
```

- Sort the names alphabetically

```
cut -f 1 -d "," tents.csv | tail +2 | head -n 5 | sort
```

```
## "fly-creek-ul2"
## "fly-creek-ul3"
## "jack-rabbit-sl3"
## "passage-2"
## "salida-2"
```

- Sort the names alphabetically in reverse order

```
cut -f 1 -d "," tents.csv | tail +2 | head -n 5 | sort -r
```

```
## "salida-2"
## "passage-2"
## "jack-rabbit-sl3"
## "fly-creek-ul3"
## "fly-creek-ul2"
```

- Write a pipe to get the names of the last 5 tents (no sorting required)

```
cut -f 1 -d "," tents.csv | tail +2 | tail -n 5
```

```
## "trail-ridge-4"
## "hula-house-6"
## "docking-station"
## "hacienda-6p"
## "hula-house-4"
```

- Write a pipe to get the name and brand of the first 5 tents

```
cut -f 1,2 -d "," tents.csv | tail +2 | head -n 5
```

```
## "fly-creek-ul2","big-agnes"
## "fly-creek-ul3","big-agnes"
## "salida-2","kelty"
## "jack-rabbit-sl3","big-agnes"
## "passage-2","rei"
```

- Write a pipe to get the name, bestuse, and capacity of the last 5 tents

```
cut -f 1,6,8 -d "," tents.csv | tail +2 | tail -n 5
```

```
## "trail-ridge-4","Carcamping","4-person"
## "hula-house-6","Carcamping","6-person"
## "docking-station","Carcamping","6-person"
## "hacienda-6p","Carcamping","6-person"
## "hula-house-4","Carcamping","4-person"
```

- List all unique `bestuse` category

```
cut -f 6 -d "," tents.csv | tail +2 | sort | uniq
```

```
## "Backpacking"
## "Carcamping"
## "Mountaineering"
```

compare only next lines… use `sort` to extract only unique categories for apart lines

- Count frequencies for each `bestuse` category

```
cut -f 6 -d "," tents.csv | tail +2 | sort | uniq -c
```

```
##   59 "Backpacking"
##   25 "Carcamping"
##    6 "Mountaineering"
```

- Take the previous command and pipe it again with sort in order to display the categories (and their counts) in increasing order

```
cut -f 6 -d "," tents.csv | tail +2 | sort | uniq -c | sort
```

```
##    6 "Mountaineering"
##   25 "Carcamping"
##   59 "Backpacking"
```

- Write a pipe to list (i.e. display) the unique categories of seasons

```
cut -f 7 -d "," tents.csv | tail +2 | sort | uniq
```

```
## "3-4-season"
## "3-season"
## "4-season"
```

- Write a pipe to list (i.e. display) the unique categories of seasons and their counts

```
cut -f 7 -d "," tents.csv | tail +2 | sort | uniq -c
```

```
##    4 "3-4-season"
##   78 "3-season"
##    8 "4-season"
```

- Cut both bestuse and seasons, and pipe them to sort and uniq to get the counts for all possible combinations of categories from bestuse and seasons

```
cut -f 6,7 -d "," tents.csv | tail +2 | sort | uniq -c
```

```
##    3 "Backpacking","3-4-season"
##   55 "Backpacking","3-season"
##    1 "Backpacking","4-season"
##    1 "Carcamping","3-4-season"
##   23 "Carcamping","3-season"
##    1 "Carcamping","4-season"
##    6 "Mountaineering","4-season"
```

- Find (match) those rows of tents with brand marmot

```
grep "marmot" tents.csv
```

```
## "limelight-3","marmot",249,2690,117,"Backpacking","3-season","3-person"
## "limelight-2","marmot",199,2100,104,"Backpacking","3-season","2-person"
## "limelight-4","marmot",299,3770,127,"Backpacking","3-season","4-person"
## "eos-1p","marmot",219,1250,91,"Backpacking","3-season","1-person"
## "aura-2p","marmot",299,1980,102,"Backpacking","3-season","2-person"
## "alpinist-2","marmot",495,2240,102,"Mountaineering","4-season","2-person"
## "widi-3","marmot",499,3060,90,"Backpacking","3-season","3-person"
## "limestone-4p","marmot",299,4961.2,152,"Carcamping","3-season","6-person"
## "limestone-6p","marmot",389,6945.6,183,"Carcamping","3-season","6-person"
## "hacienda-6p","marmot",599,9553.8,203,"Carcamping","3-season","6-person"
```

- Count how many marmot tents are in tents.csv (i.e. counting number of lines)

```
grep "marmot" tents.csv | wc -l
```

```
##       10
```

- Create a file with the data for marmot tents marmot-tents.csv

```
grep "marmot" tents.csv > marmot-tents.csv
```

- Include column names to the file marmot-tents.csv

```
head -n 1 tents.csv > marmot-tents.csv
grep "marmot" tents.csv >> marmot-tents.csv
```

- Pipe grep with wc to count the number of tents from brand rei

```
grep "rei" tents.csv | wc -l
```

```
##       22
```

- Write a similar pipe to the one above to find how many Backpacking tents are in tents.csv

```
grep "Backpacking" tents.csv | wc -l
```

```
##       59
```

- Write a pipe to create a file big-agnes-tents.csv containing the data for tents of brand big-agnes

```
grep "big-agnes" tents.csv > big-agnes-tents.csv
```

- Write commands to create a file kelty-tents.csv containing the data for tents of brand kelty, arranged by name alphabetically. This file should have column names in the first line (i.e. first row)

```
head -n 1 tents.csv > kelty-tents.csv
grep "kelty" tents.csv | sort >> kelty-tents.csv
```

- Write a pipeline to obtain the unique categories of bestuse (column 6). The output of the pipeline should contain only the categories (NOT the counts). HINT: cut, sort, and uniq are your friends.

```
cut -f 6 -d "," tents.csv | tail +2| sort | uniq
```

```
## "Backpacking"
## "Carcamping"
## "Mountaineering"
```

- Write a pipeline to obtain the counts (i.e. frequencies) of the different bestuse values (column 6), displayed from largest to smallest (i.e. descending order). The first column corresponds to count, the second column corresponds to experience. Redirect the output to a text file bestuse-counts.txt. A HINT: cut, tail, sort, uniq; and redirection > operator, are your friends.

```
cut -f 6 -d "," tents.csv | tail +2| sort |uniq -c
```

```
##   59 "Backpacking"
##   25 "Carcamping"
##    6 "Mountaineering"
```

- Use output redirection commands to create a CSV file msr-tents.csv containing data for the msr brand. Your CSV file should include column names in the first row. HINT: redirection operators > and >>, as well as head and grep are your friends.

```
head -n 1 tents.csv > msr-tents.csv
grep "msr" tents.csv >> msr-tents.csv
```

- Use the previously created file msr-tents.csv to select, separately, the columns name, weight, and price. Store each column in a text file: name.txt, weight.txt, and price.txt

```
cut -f 1 -d "," msr-tents.csv > name.txt
cut -f 4 -d "," msr-tents.csv > weight.txt
cut -f 3 -d "," msr-tents.csv > price.txt
```

- Use the previously created files name.txt, weight.txt, and price.txt to paste them (i.e. merge them), in that order, into a new CSV file msr-prices.csv (comma separated values!!!).

```
paste -d "," name.txt weight.txt price.txt > msr-prices.csv
```

- Write a pipeline to list (display) the five largest prices in decrasing order.

```
cut -f 3 -d "," tents.csv | tail +2 | sort -r -n | head -n 5
```

```
## 699.95
## 599.95
## 599.95
## 599.95
## 599
```

- Write pipelines to create a file top10-tents.csv containing the top10 tents by price from largest to smallest. Your CSV file should include tent name, brand, price

```
head -n 1 tents.csv > top10-tents.csv
cut -f 1-3 -d "," tents.csv | tail +2 | sort -k 3 -t "," -r -n | head -n 10 >> top10-tents.csv
```

```
cut -f 1-3 -d "," tents.csv | tail +2 | sort -k 3 -t "," -n -r | head -n 10
```

```
## "backcountry-barn","msr",699.95
## "soda-mountain-sl","big-agnes",599.95
## "copper-spur-ul4","big-agnes",599.95
## "carbon-reflex-3","msr",599.95
## "hacienda-6p","marmot",599
## "ve-25","north-face",579
## "mountain-manor-6","north-face",529
## "string-ridge-2","big-agnes",499.95
## "fury-2","msr",499.95
## "flying-diamond-6","big-agnes",499.95
```