Final Assignment

awk

Description: awk is a scripting language used for processing and displaying text. awk can work with a text file or from standard output.

formula/syntax: awk + options + {awk command} + file + file to save (optional)

3 examples that you understand well:

- print the first column of every line of a file: awk '{print $1}' ~/Documents.Csv/cars.csv
- print the first field of /etc/passwd: awk -F: '{print $1}' /etc/passwd
- print the last field of the /etc/passwd file: awk -F: '{print $NF}' /etc/passwd

cat

Description: the cta command is used for displaying the content of a file.

formula/syntax: cat + option + file(s) to display

3 examples that you understand well:

- display he content of a file located in the pwd: cat todo.lst
- display the content of a file using absolute path: cat ~/Documents/todo.lst
- display the content of a file with line numbers: cat -n ~/Documents/todo.md

cp

Description: cp copes files/directories from a source to a destination

formula/syntax: files cp + files to copy + destination cp Downloads/wallpapers.zip Pictures/

directories:

- cp -r directory to copy + destination

3 examples that you understand well:

- to copy a directory with absolute path: cp -r ~/Downloads/wallpapers ~/Pictures/
- to copy multiple files in a single command: suco cp -r script.sh program.py home.html assets/ /var/www/html/
- to copy the content of a directory to another directory: cp Downloads/wallpapers/* ~/Pictures/

cut

Description: cut command is used to extract a specific section of each line of a file ad display it to the screen.

formula/syntax: cut + option + file(s) 3 examples that you understand well:

- display a list of all the users in your system: cut -d ':' -f1 /etc/passwd
- display a list of all the users in yoyur system with their login shell: cut -d ":" -f1,7 /etc/passwd
- cut a file using a delimeter but changing the delimeter in the output: cut -d -f1,7 --output-delimeter=' => ' /etc/passwd

grep

Description: grep is used to search text in given file. grep searches line by line.

formula/syntax: grep + option + search criteria + file(s)

3 examples that you understand well:

- search any line that contains the word "dracula" in the given file: grep 'dracula' ~/Documents/dracula.txt
- search for all the lines that do not contain the word 'war': grep -v 'war' ~/Documents/Books/war-and-peace.txt
- search any lines that contains the word dracula regardless of case and with number line: grep -in 'dracula' ~/Documents/Books/dracula.txt

head

Description: the head command displays the top N number of lines of a given file. prints out the first 10 lines.

formula/syntax: head +option + file(s)

3 examples that you understand well:

- display the first 10 lines of a file: head ~/Documents/Book/dracula.txt
- display the first 5 lines of a file: head -5 ~/Documents/Book/dracula.txt
- display the first 3 lines of a file: head -3 ~/Dcouments/Book/dracula.txt

ls

Description: used for displaying all the files inside a given directory. When no directory is specified, ls displays the files in the current working directory.

formula/syntax: ls

3 examples that you understand well:

- list all the files in current directory including hidden files: ls -a
- list all the files in a given directory: ls -a ~/Pictures
- list all the options of the ls command: ls --help

man

Description: man pages are documentation files that describe linux shell commands, executable programs, system calls, special files, and so forth

formula/syntax: man + command

3 examples that you understand well

- man ls
- man passwd
- man cp

mkdir

Description: used for creating a single directory or multiple directories

formula/syntax: mkdir + name of directory

3 examples that you understand well:

- create a directory in a different directory using relative path: mkdir wallpapers/ocean
- create a directory in a different directory using absolute path: mkdir ~/wallapapers/forest
- create multiple directories: mkdir wallpapers/cars wallpapers/cities wallpapers/forest
- create a directory with a parent directory: mkdir -p wallpapers_others/movies

mv

Description: moves and renames directories formula/syntax: mv + source + destination for renaming files/directories the formula remains the same: mv + file/directory to rename + new name 3 examples that you understand well: moving files

- to move file from a directory to another using relative path: mv Downloads/homework.pdf Documents/
- to move a directory from one directory to another using absolute path: sudo mv ~/Downloads/theme /usr/share/themes
- to move multiple directories/files to a different directory: mv games/ wallpapers/ rockmusic/ /media/student/flashdrive/

renaming:

- rename file: mv homework.docx cis106.docx
- rename using absolute path: mv ~Downloads/homework.docx ~Downloads/cis106homework.docx
- move and rename a file in the same command: mv Downloads/cis106.docx Documents/new_cis106homework.docx

tac

Description: the tac command is used for displaying the content of a file in reverse order.

formula/syntax: tac + option + file(s) to display

3 examples that you understand well:

- display the content of a file located in pwd: tac todo.md
- display the content of a file using absolute path: tac ~/Documents/todo.md
- display the content of the /etc/passwd file in reverse order: tac /etc/passwd

tail

Description: the tail command displays the last N number of lines of a given file.

formula/syntax: tail + option + file(s)

3 examples that you understand well:

- display the last 10 lines of a file: tail ~/Documents/Book/dracula.txt
- display the last 5 lines of a file: tail -5 ~/documents/Book/dracula.txt
- display the last 5 lines of the /etc/passwd file: tail -5 /etc/passwd

touch

Description: used to create files formula/syntax: touch + name of file 3 examples that you understand well:

- to create several files: touch list_of_cars.txt script.py names.csv
- to create a file using absolute path: touch ~/Downloads/games.txt
- to create file using relative path assuming you are in your home directory: touch Downloads/games2.txt

tr

Description: the tr command is used for translating or deleting characters from standard output.

formula/syntax: Standard output | tr + option + set + set

3 examples that you understand well:

- translate one character to another for example a period with a comma: cat file.txt | tr '.' ','
- translate white space into tabs: cat program.py | tr "[:space:]" '/t'
- translate tabs into space: cat file.py | tr s "[:space:]" ' '

tree

Description: tree is a directory listing program that produces a depth-indented listing of files

formula/syntax: tree or tree + directory

3 examples that you understand well:

- to display current directory structure: tree
- display the tree of a directory: tree -a ./GFG
- to display only files that match the wild-card pattern use the -P and specify the pattern: tree -f -P cata*

How to work with multiple terminals open?

use tilix and use the +[] to add a terminal on either the side or the bottom of the terminal.

How to work with manual pages?

man pages are documentation files that describe linux shell commands, executable programs, system calls, special files, and so forth. to navigate the man page of a command, you can use the arrow key or the man

command internal shortcuts. to exit the man page press letter "q"

How to parse (search) for specific words in the manual page

- man -k file

How to redirect output (> and |)

Description: The pipe allows you to redirect the standard output of a command to use the standard input of another.

formula/syntax: command_1 | command_2 | command_3 | .... | command_N

3 examples that you understand well:

- use grep to look for a string in a particular man page: man ls | grep "human-readable"
- display only the option of the any command from its man page: man ls | grep "^[[:space:]]*[[:punct:]]"
- display only the 2nd line of a file: head -2 file.lst | tail -1

usage: command out put + > + file example:

- save the output of a command to a file: ls -lA ~ > all-files-in-home.txt
- save the error generated by a command to a file: ls -lA downloads/ 2> error-of-ls

How to append the output of a command to a file

Append means t add more to a file instead of overwriting its content. WHen we use > on a file that already exist and contains data, we overwrite whatever is already inside the file. For instance take this example: ls -la > allmyfiles.lst. In this example if the file allmyfiles.lst had any data prior executing the command, that data will be overwritten by the output of ls -la. If we want to keep the old data we would use: ls -la >> allmyfiles.lst

How to use wildcards

For copying and moving multiple files at the same time

- Wildcard represents letters and characters used to specify a file name for searches. FIle gobbling is the processing of pattern matching using wildcards. The wildcards are officially called metacharacter wildcards.
- examples: ls *.txt will match all files that end in .txt regardless of the size of the file name
- ls wildcardpractice/*random* lists all the files that contain random in the name
- the ? wildcard matches precisely one character
- list all the files in a directory that have a 3 letter file extension: ls *.???
- list all the files in a directry that start with a letter v and have 2 characters following and a 3 letter file extension: ls v??*.???

the [] wildcard match a single character in a range

- to match all files tat have a vowel after letter f: ls f[aeiou]*
- to match all files have a range of letters after f: ls f[a-z]*
- to match all files whose name begins with any 3 combination of numbers and the current user's usrnm: ls [0-9][0-9][0-9]$USER

How to use brace expansion

For creating entire directory structures in a single command Brace expansion {} is not a wildcard but another feature of bash that allows you to generate arbitrary strings to use with commands.

- to create a whole directory structure in a single command: mkdir -p music/{jazz,rock}/{mp3files,videos,oggfiles}/new{1..3}
- to remove multiple files in a single directory: rm -r {dir1,dir2,dir3,file.txt,file.py}