J-La Kimbrough
DATA-413 | Data Science
Due: 9.15.24

HW 1 – Bash

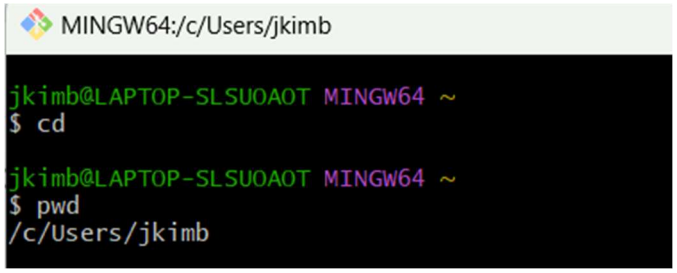
Question 1: (10 Points)

What is the difference between shell and bash?

While the terms "shell" and "bash" can be used interchangeably, there is a subtle difference between the two. The term "shell" simply refers to any program that provides a command-line interface for interacting with an operating system. The term "bash" stands for Bourne-Again SHell and is a commonly used Unix/Linux shell. Bash is simply a type of shell.

Question 2: (10 Points) To respond to this question, you need to use terminal/Bash and have a screenshot of your terminal/bash.

A. What is your home directory?



B. What files/folders exist in it?

J-La Kimbrough
DATA-413 | Data Science
Due: 9.15.24

HW 1 – Bash

```
jkimb@LAPTOP-SLSUOAOT MINGW64 ~
$ ls
 02-lab_pandas_guided_exercise.ipynb
 0_functions.ipynb
 1_Array_creation_routines_Solution.ipynb
 2_Array_manipulation_routines_Solutions.ipynb
 3_Mathematical_functions_solutions.ipynb
 4_Sorting_searching_and_counting_Solutions.ipynb
 AppData/
'Application Data'@
 CIFAR10_data_loader.ipynb
 CSC-208/
 CSC-256_HW-03_V1.ipynb
 CSC-256_HW5.ipynb
 CSC-256_HW6.ipynb
 CSC-420_HW-03_V1.ipynb
 CSC-420_HW-04_unedited.ipynb
 CSC-420_HW03_V2.ipynb
'CSC-420_HW03_V3 (test codes).ipynb'
 CSC-420_HW03_V4.ipynb
 CSC-420_HW5-Copy1.ipynb
 CSC-420_HW5.ipynb
 CSC-420_HW6-Copy1.ipynb
 CSC-420_HW6.ipynb
```

Question 3: (10 Points) To respond to this question, you need to use terminal/Bash and have a screenshot of your terminal/bash.

A. **Where does the command `cd../../` take you? Run the command `pwd` and explain the output!**

```
jkimb@LAPTOP-SLSUOAOT MINGW64 ~
$ cd ../../

jkimb@LAPTOP-SLSUOAOT MINGW64 /c
$ pwd
/c
```

The `cd ../../` moved me up two directory levels from my current location, which was my home directory. After running `pwd`, it confirmed that I had navigated to the root directory of my C: drive. The output is /c, indicating that I am now at the root directory of my C: drive.

B. **What does the command `cd` do? Run the command `pwd` and explain the output!**

J-La Kimbrough
DATA-413 | Data Science
Due: 9.15.24

HW 1 – Bash

```
jkimb@LAPTOP-SLSUOAOT MINGW64 /c
$ cd

jkimb@LAPTOP-SLSUOAOT MINGW64 ~
$ pwd
/c/Users/jkimb
```

Running the `cd` command without arguments brings you back to your home directory, regardless of where you are in the file system. Running the `pwd` command shows you what your current working directory is, in this case, it is my home directory.

Question 4: (10 Points) To respond to this question, you need to use terminal/Bash and have a screenshot of your terminal/bash.

Read the manual page of ls. What does the `a` flag do? What does the `l` flag do?

J-La Kimbrough
DATA-413 | Data Science
Due: 9.15.24

HW 1 – Bash

```
jkimb@LAPTOP-SLSUOAOT MINGW64 ~
$ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

Mandatory arguments to long options are mandatory for short options too.
  -a, --all                  do not ignore entries starting with .
  -A, --almost-all           do not list implied . and ..
      --author               with -l, print the author of each file
  -b, --escape               print C-style escapes for nongraphic characters
      --block-size=SIZE      with -l, scale sizes by SIZE when printing them;
                               e.g., '--block-size=M'; see SIZE format below
  -B, --ignore-backups       do not list implied entries ending with ~
  -c                         with -lt: sort by, and show, ctime (time of last
                               modification of file status information);
                               with -l: show ctime and sort by name;
                               otherwise: sort by ctime, newest first
  -C                         list entries by columns
      --color[=WHEN]         colorize the output; WHEN can be 'always' (default
                               if omitted), 'auto', or 'never'; more info below
  -d, --directory            list directories themselves, not their contents
  -D, --dired                generate output designed for Emacs' dired mode
  -f                         do not sort, enable -aU, disable -ls --color
  -F, --classify             append indicator (one of */=>@|) to entries
      --file-type            likewise, except do not append '*'
      --format=WORD          across -x, commas -m, horizontal -x, long -l,
                               single-column -1, verbose -l, vertical -C
      --full-time            like -l --time-style=full-iso
  -g                         like -l, but do not list owner
      --group-directories-first
                             group directories before files;
                               can be augmented with a --sort option, but any
                               use of --sort=none (-U) disables grouping
  -G, --no-group             in a long listing, don't print group names
  -h, --human-readable       with -l and -s, print sizes like 1K 234M 2G etc.
      --si                   likewise, but use powers of 1000 not 1024
  -H, --dereference-command-line
                             follow symbolic links listed on the command line
      --dereference-command-line-symlink-to-dir
                             follow each command line symbolic link
                               that points to a directory
      --hide=PATTERN         do not list implied entries matching shell PATTERN
                               (overridden by -a or -A)
      --hyperlink[=WHEN]     hyperlink file names; WHEN can be 'always'
                               (default if omitted), 'auto', or 'never'
      --indicator-style=WORD  append indicator with style WORD to entry names:
                               none (default), slash (-p),
                               file-type (--file-type), classify (-F)
  -i, --inode                print the index number of each file
  -I, --ignore=PATTERN       do not list implied entries matching shell PATTERN
  -k, --kibibytes            default to 1024-byte blocks for disk usage;
                               used only with -s and per directory totals
  -l                         use a long listing format
  -L, --dereference          when showing file information for a symbolic
                               link, show information for the file the link
                               references rather than for the link itself
```

The `a` flag represents all files, including hidden files and directories in the listing. The `l` flag displays detailed information about files and directories, it is also known as a long format.

J-La Kimbrough
DATA-413 | Data Science
Due: 9.15.24

HW 1 – Bash

Question 5: (A and B each have 5 points, and C has 10 points. The total is 20 points.) To respond to this question, you need to use terminal/Bash and have a screenshot of your terminal/bash.

A. Create a folder within your home directory, which was identified in Question 2, and name it 'temp_bash'.

```
jkimb@LAPTOP-SLSUOAOT MINGW64 ~
$ cd

jkimb@LAPTOP-SLSUOAOT MINGW64 ~
$ mkdir temp_bash
```

B. Create a new file using the command `touch` and name it `myfile.txt` inside the new folder `temp_bash` and run `ls` to show that the file is inside the folder.

```
jkimb@LAPTOP-SLSUOAOT MINGW64 ~
$ cd temp_bash

jkimb@LAPTOP-SLSUOAOT MINGW64 ~/temp_bash
$ touch myfile.txt

jkimb@LAPTOP-SLSUOAOT MINGW64 ~/temp_bash
$ ls
myfile.txt
```

C. Run the `stat myfile.txt` command and explain the information retrieved from the output. Here is an example of what should be included in the output and a brief explanation for each part.

- `Blocks: 0` ` The number of blocks for the file.
- `IO Block: 65536` The size of each block.

J-La Kimbrough
DATA-413 | Data Science
Due: 9.15.24

HW 1 – Bash

```
jkimb@LAPTOP-SLSUOAOT MINGW64 ~/temp_bash
$ stat myfile.txt
  File: myfile.txt
  Size: 0               Blocks: 0          IO Block: 65536  regular empty file
Device: 1a3715b9h/439817657d    Inode: 4222124650679226  Links: 1
Access: (0644/-rw-r--r--)  Uid: (197609/   jkimb)   Gid: (197609/ UNKNOWN)
Access: 2024-09-11 21:33:48.680816000 -0400
Modify: 2024-09-11 21:33:48.680816000 -0400
Change: 2024-09-11 21:33:48.680448200 -0400
 Birth: 2024-09-11 21:33:48.680448200 -0400
```

- Blocks: 0 = Shows the number of blocks allocated for the file. Since this file is empty, no disk blocks are needed.
- IO Block: 65536 = Indicates the size of each block used for input/output operations on this file.
- Size: 0 = Represents the file size in bytes. Since the file is empty, it shows 0 bytes.
- Device/Inode: Both are unique identifiers for the file on the file system.
- Access/Modify/Changes Times: These are the timestamps indicating the last access, modification, and change times for the file.

Question 6: (40 Points) To respond to this question, you need to use terminal/Bash and have a screenshot of your terminal/bash.

A. Use the command `>>` and add the following line `This line is my first line`. Now add the following line `This line is my second line`. Then, run `cat myfile.txt` to show that the line has been added.

```
jkimb@LAPTOP-SLSUOAOT MINGW64 ~/temp_bash
$ echo "This line is my first line." >> myfile.txt

jkimb@LAPTOP-SLSUOAOT MINGW64 ~/temp_bash
$ echo "This line is my second line." >> myfile.txt

jkimb@LAPTOP-SLSUOAOT MINGW64 ~/temp_bash
$ cat myfile.txt
This line is my first line.
This line is my second line.
```

B. Copy the file `myfile.txt to file copy_myfile.txt with the command `cp`

J-La Kimbrough
DATA-413 | Data Science
Due: 9.15.24

HW 1 – Bash

```
jkimb@LAPTOP-SLSUOAOT MINGW64 ~/temp_bash
$ cp myfile.txt copy_myfile.txt

jkimb@LAPTOP-SLSUOAOT MINGW64 ~/temp_bash
$ cat copy_myfile.txt
This line is my first line.
This line is my second line.
```

C. Use the command `>` and add the following line `This line is a new line` to `copy_myfile.txt`. Then run `cat copy_myfile.txt` to show the line is added.

```
jkimb@LAPTOP-SLSUOAOT MINGW64 ~/temp_bash
$ echo "This line is a new line." > copy_myfile.txt

jkimb@LAPTOP-SLSUOAOT MINGW64 ~/temp_bash
$ cat copy_myfile.txt
This line is a new line.
```

D. Explain the difference between `>` and `>>` based on the result of the Question 6.

The `>>` command appends new content to an existing file, adding new lines while preserving the original content. This is useful when you want to add to a file without losing any of the existing data. In contrast, the `>` command overwrites the file entirely, replacing all existing content with the new data. This is helpful when you want to start fresh and no longer need the old content.

J-La Kimbrough
DATA-413 | Data Science
Due: 9.15.24

HW 1 – Bash

References

Geeks for Geeks. (2024, August 5). *ls Command in Linux.* Geeksforgeeks. https://www.geeksforgeeks.org/ls-command-in-linux/

Hira, Z. (2023, March 20). *Bash Scripting Tutorial – Linux Shell Script and Command Line for Beginners.* freeCodeCamp. https://www.freecodecamp.org/news/bash-scripting-tutorial-linux-shell-script-and-command-line-for-beginners/