

Exercise-Bash

Question:

What is difference between shell and bash?

- “shell” is a broad term that refers to any program that provides a command-line interface, “Bash” is a specific type of shell that is widely used in Unix/Linux systems.

Exercise 1:

What is your home directory? What files/folders exist in your home directory? Navigate to it and then navigate back to your notes.

- The home directory is the default directory assigned to us where we store personal files and directories.
- To view the files in home directory we can use the command **ls** (screenshot attached below)

```
kriti@kritika MINGW64 ~
$ cd
kriti@kritika MINGW64 ~
$ cd /c/Users/kriti/OneDrive/Desktop/DataScience
kriti@kritika MINGW64 ~/OneDrive/Desktop/DataScience
$ #looking up files in home directory
kriti@kritika MINGW64 ~/OneDrive/Desktop/DataScience
$ cd
kriti@kritika MINGW64 ~
$ ls
AppData/      Favorites/      NTUSER.DAT{96538bec-e9a5-11ee-a1d6-de23eb79529b}.tm.blf  Recent@  Untitled.ipynb  ntuser.ini
Application Data/  Links/          NTUSER.DAT{96538bec-e9a5-11ee-a1d6-de23eb79529b}.tm.container00000000000000000001.regtrans-ms  Saved Games/  Videos/
Contact/         'Local Settings'/  NTUSER.DAT{96538bec-e9a5-11ee-a1d6-de23eb79529b}.tm.container00000000000000000002.regtrans-ms  Searches/     WA-Fin-UseC...HR-Employee-Attrition.csv
Cookies@         Music/           NetHood@
Documents/       'My Documents'@  OneDrive/
Downloads/       NTUSER.DAT       PrintHood@
kriti@kritika MINGW64 ~
$ |
```

- Navigate to it and then navigate back to your notes.

```
MINGW64:/c/Users/kriti/OneDrive/Desktop/DataScience

kriti@Kritika MINGW64 ~
$ cd
kriti@Kritika MINGW64 ~
$ cd /c/Users/kriti/OneDrive/Desktop/DataScience
kriti@Kritika MINGW64 ~/OneDrive/Desktop/DataScience
$ |
```

Exercise 2:

Where does the following command take you? How does it work?

```
cd ../../../../..
```

- The command moves back 3 folders

```
kriti@Kritika MINGW64 ~/OneDrive/Desktop/DataScience
$ cd ../../../../..

kriti@Kritika MINGW64 /
$ |
```

Exercise 3:

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

- **a flag:**

`-a, --all` do not ignore entries starting with `.`

`-a`: Lists all files, including hidden files (those starting with a dot `.`).

- **l flag:**

`-l` list one file per line. Avoid `'\n'` with `-q` or `-b`

`-l`: Displays detailed information about each file, such as permissions, number of links, owner, group, size, and modification time.

```
MINGW64/
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 -l, 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:
```

Exercise 4:

Create a new file with `touch` command. for instance `touch myfile.txt`. Run `stat myfile.txt` what information do you get?

Information Provided:

- File: `myfile.txt`
- Size: The size of the file
- Blocks: Number of blocks allocated
- IO Block: Block size
- Device: Device number
- Inode: Inode number
- Links: Number of links
- Access: File permissions
- Uid: User ID of the file's owner
- Gid: Group ID of the file's owner
- Access: Last access time
- Modify: Last modification time
- Change: Last status change time
- Birth: Created time

```
kriti@Kritika MINGW64 ~/OneDrive/Desktop/DataScience
$ touch myfile.txt

kriti@Kritika MINGW64 ~/OneDrive/Desktop/DataScience
$ stat myfile.txt
  File: myfile.txt
  Size: 0                Blocks: 0          IO Block: 65536  regular empty file
Device: 28d3f43ch/684979260d    Inode: 4785074604211258  Links: 1
Access: (0644/-rw-r--r--)  Uid: (197609/  kriti)   Gid: (197609/ UNKNOWN)
Access: 2024-07-06 19:54:12.389430400 -0400
Modify: 2024-07-06 19:54:12.389430400 -0400
Change: 2024-07-06 19:54:13.355456000 -0400
 Birth: 2024-07-06 19:54:12.386740900 -0400

kriti@Kritika MINGW64 ~/OneDrive/Desktop/DataScience
$ |
```

Exercise 5:

Run `ls` and from there list select a file. Now run '`ls -l`' to display the details of the files, showing that it has been created or updated. what information does it give you regarding the `myfile.txt` and your selected file.

```
kriti@Kritika MINGW64 ~/OneDrive/Desktop/DataScience
$ ls
'DS Files'/  HW1.htm  HW1_files/  myfile.txt

kriti@Kritika MINGW64 ~/OneDrive/Desktop/DataScience
$ ls -l
total 96
drwxr-xr-x 1 kriti 197609 0 Jul 6 19:23 'DS Files'/
-rw-r--r-- 1 kriti 197609 92290 Jul 1 17:48 HW1.htm
drwxr-xr-x 1 kriti 197609 0 Jul 6 19:23 HW1_files/
-rw-r--r-- 1 kriti 197609 0 Jul 6 19:54 myfile.txt
```

```
kriti@Kritika MINGW64 ~/OneDrive/Desktop/DataScience
$ ls -l HW1.htm
-rw-r--r-- 1 kriti 197609 92290 Jul 1 17:48 HW1.htm
```

- Details provided:
 - 1: Number of links to the file.
 - kriti: The owner of the file.
 - 197609: The group ID associated with the file.
 - 92290: Size of the file in bytes.
 - Jul 1 17:48: Last modification date and time of the file.
 - HW1.htm: Name of the file. (HW1.htm was my selected file)

Details provided by `ls -l` regarding `myfile.txt`. The `ls -l` command provides the information for all the files in that directory.

- 1: This indicates the number of links to this file.
- kriti: This is my username, indicating that I own this file.
- 197609: This is the group ID associated with my user account.
- 0: This is the size of the file in bytes.
- Jul 6 19:54: This is the last modification date and time of the file.
- myfile.txt: This is the name of the file.

The `ls -l` command output confirms that the files and directories have been created or updated, as shown by the timestamps. The `myfile.txt` file was last modified on Jul 6 at 19:54, indicating that it has been updated recently. The permissions and ownership details provide additional information about access control for each file and directory.

Exercise 6:

Add the following line `This line is my first line` to `myfile.txt`. Then run `cat myfile.txt` to show the line is added.

```
kriti@Kritika MINGW64 ~/OneDrive/Desktop/DataScience
$ echo "This line is my first line" >myfile.txt

kriti@Kritika MINGW64 ~/OneDrive/Desktop/DataScience
$ cat myfile.txt
This line is my first line

kriti@Kritika MINGW64 ~/OneDrive/Desktop/DataScience
$ |
```

Exercise 7:

Run `touch myfile.txt` then run `ls -l myfile.txt` does the “timestamp” for the file `myfile.txt` is updated? Show the output. *Note: Another common use of the `touch` command is to update the timestamps of an existing file.*

- Yes the timestamp was updated

```
kriti@Kritika MINGW64 ~/OneDrive/Desktop/DataScience
$ touch myfile.txt

kriti@Kritika MINGW64 ~/OneDrive/Desktop/DataScience
$ ls -l myfile.txt
-rw-r--r-- 1 kriti 197609 27 Jul  6 20:10 myfile.txt

kriti@Kritika MINGW64 ~/OneDrive/Desktop/DataScience
$ |
```