UNIX SHELL

Done By:

Adithyan Sukumar (CB.EN.U4AIE19004)

Arun Prakash J (CB.EN.U4AIE19014)

Challa Sai Harsha Reddy (CB.EN.U4AIE19018)

Deepthi Sudharsan (CB.EN.U4AIE19022)



B.TECH CSE-AI, CEN DEPARTMENT, AMRITA SCHOOL OF ENGINEERING

ACKNOWLEDGEMENT

This project would not have been possible without the support and guidance of Mr. Premjith B and Mr. Sachin Kumar S, who have provided us with sufficient knowledge and guidelines even during these unprecedented and trying times.

We would also like to thank our Computer Science and Engineering (Artificial Intelligence) department for giving us this opportunity to nurture and hone our skills.

Furthermore, we would like to thank the Amrita Vishwa Vidyapeetham management for providing ample resources to avail our project needs and a platform for online lectures during this lockdown.

TABLE OF CONTENTS

ABSTRACT4

UNIX SHELL5

UNIX COMMANDS5

FEW UNIX COMMAND IMPLEMENTATIONS IN C18

FEW UNIX COMMAND IMPLEMENTATIONS IN PYTHON25

FEW UNIX COMMAND IMPLEMENTATIONS IN JAVA27

REFERENCES29

ABSTRACT

A Unix shell is a command-line interpreter or shell that provides a command line user interface for Unix-like operating systems. The shell is both an interactive command language and a scripting language and is used by the operating system to control the execution of the system using shell scripts.

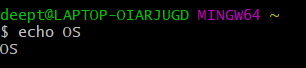
This project provides the basic understanding and implementation of different Basic UNIX shell commands and their functions. The project also gives a basic insight on the implementation of few of the UNIX commands in C, Python and Java.

*Keywords*: UNIX, Shell, Operating Systems

UNIX SHELL

UNIX COMMANDS:

1. **echo** : displays arguments on the screen



1. **uptime :** shows how long our system has been running and the number of users



1. **users** : displays currently logged users

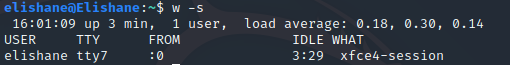


1. **W** : displays users logged in and their process .

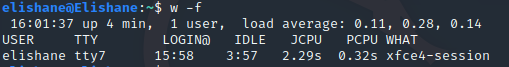
**$w -h**



**$w -s**



**$w -f**



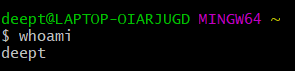
**$w -V**



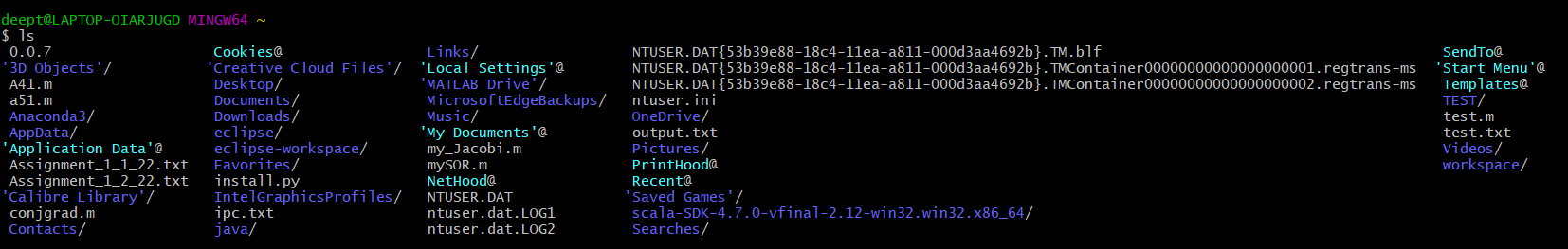
1. **who** : returns user name, date



1. **whoami** : it displays the current user name only

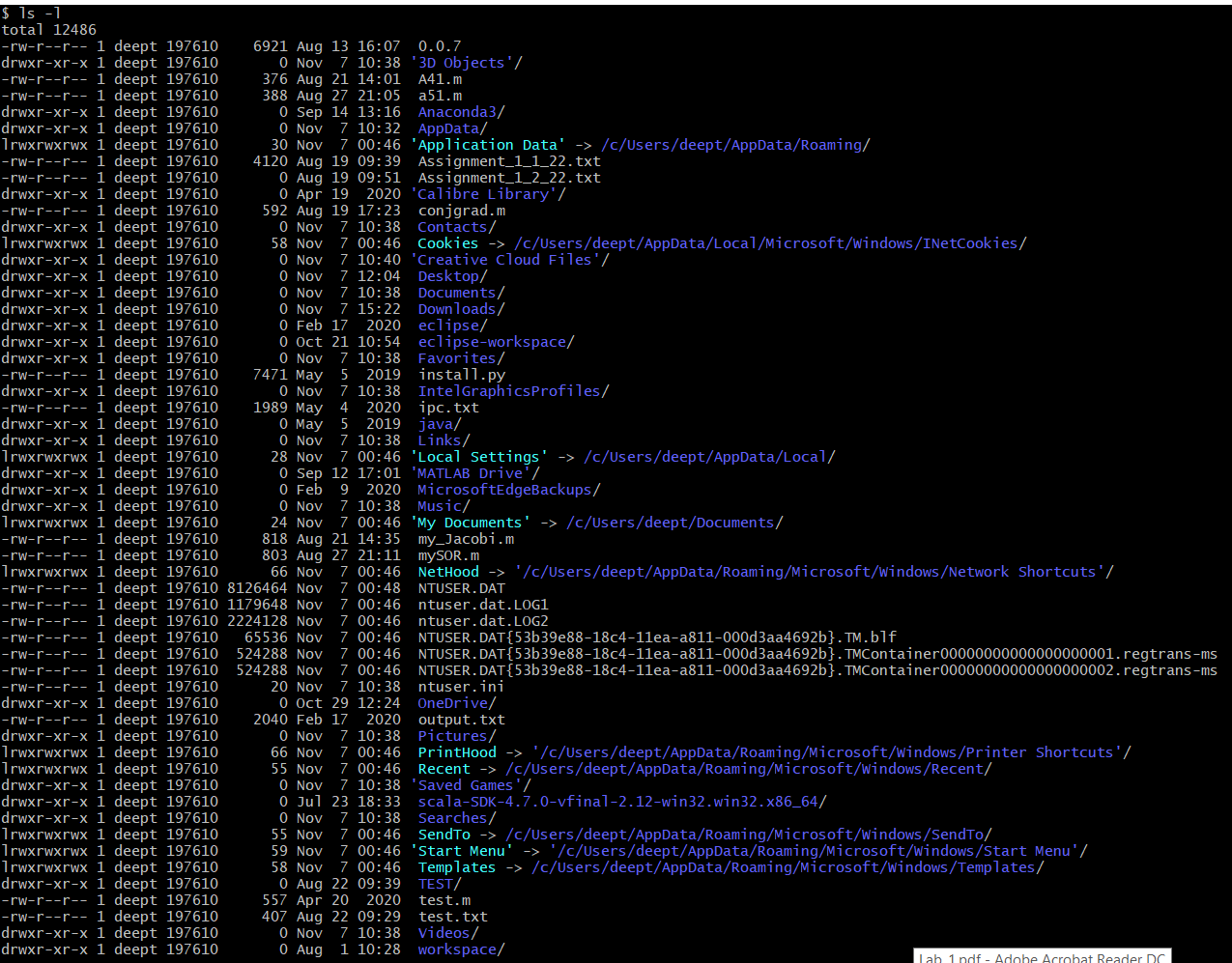


1. **ls** : lists the files and directories



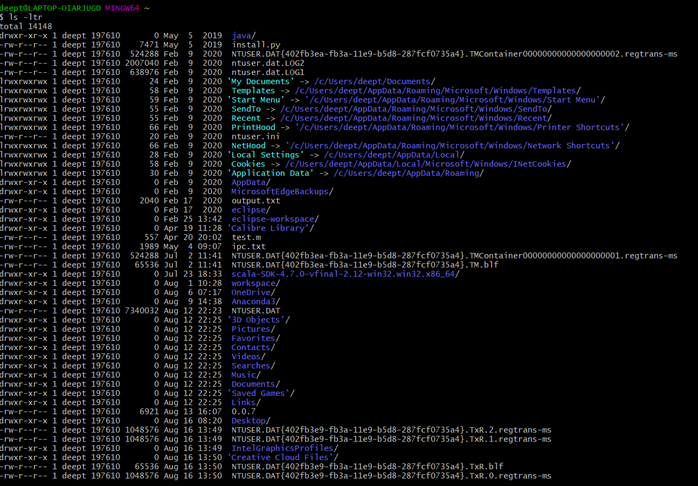
**$ls -l :** Shows file or directory, size, modified date and time, file

or folder name and owner of file and its permission.



**$ls -ltr**

Sorts file according to the last modified time.

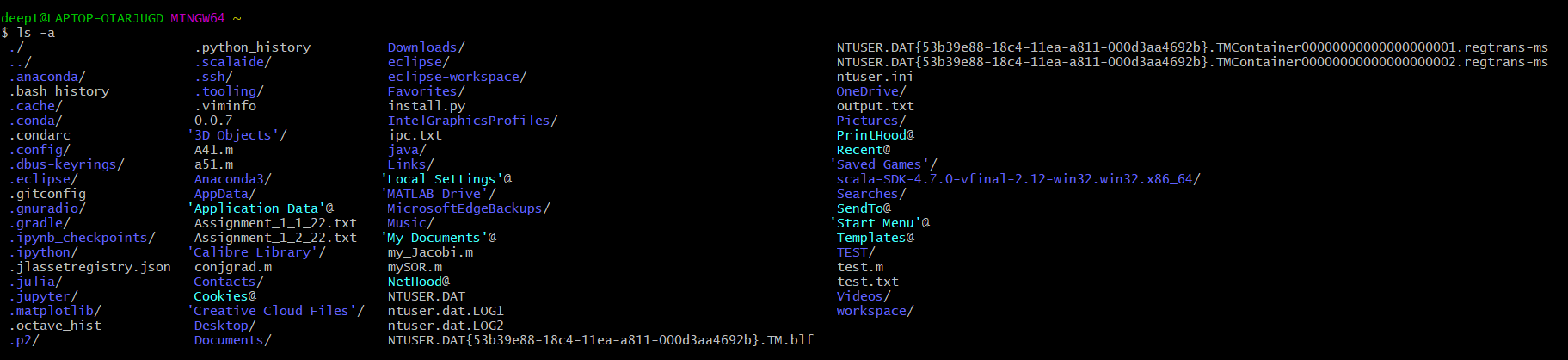


**ls -ltr > Assignment\_1\_1\_22.txt**

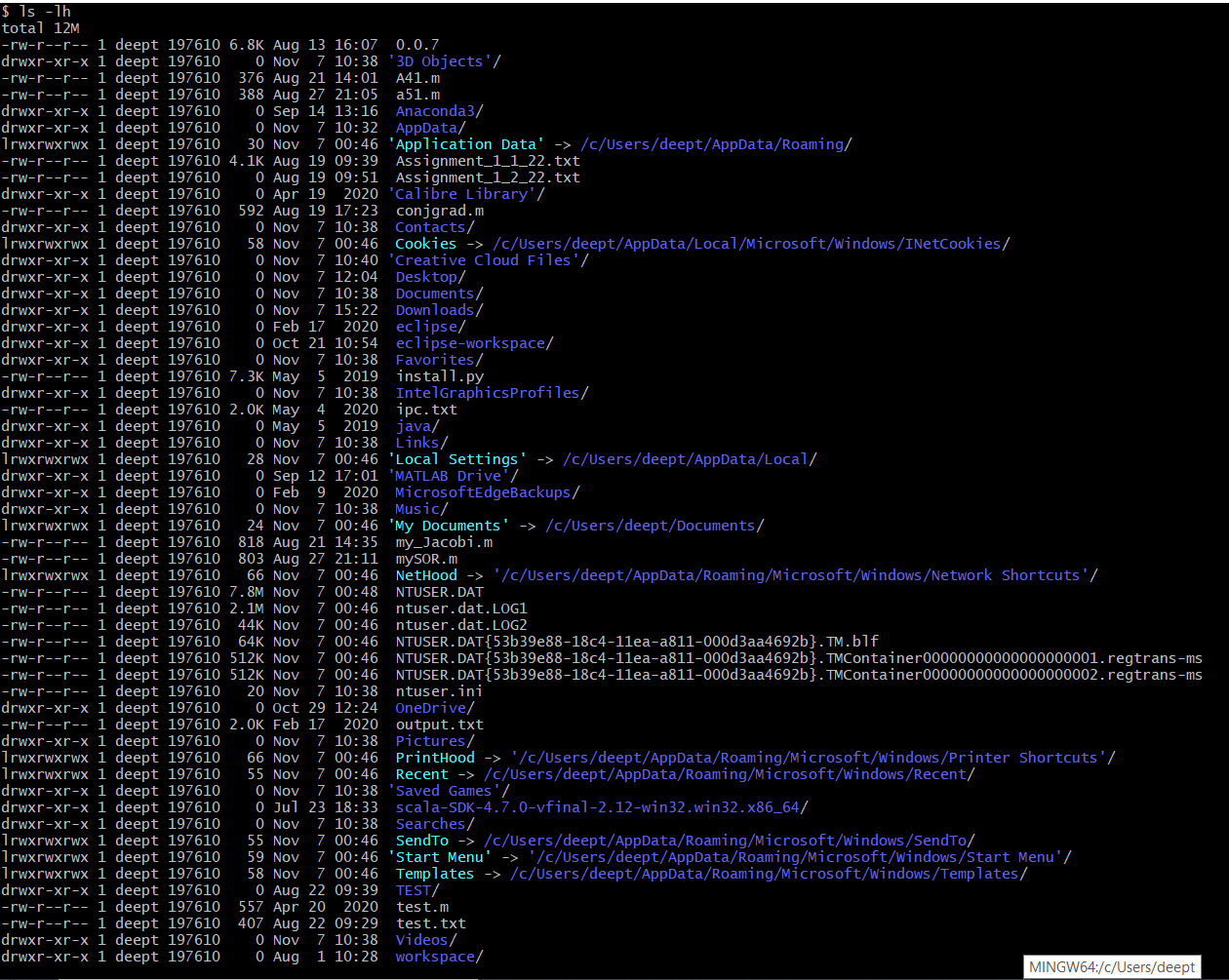
Redirecting the result to a file called Assignment\_1\_1\_22.txt



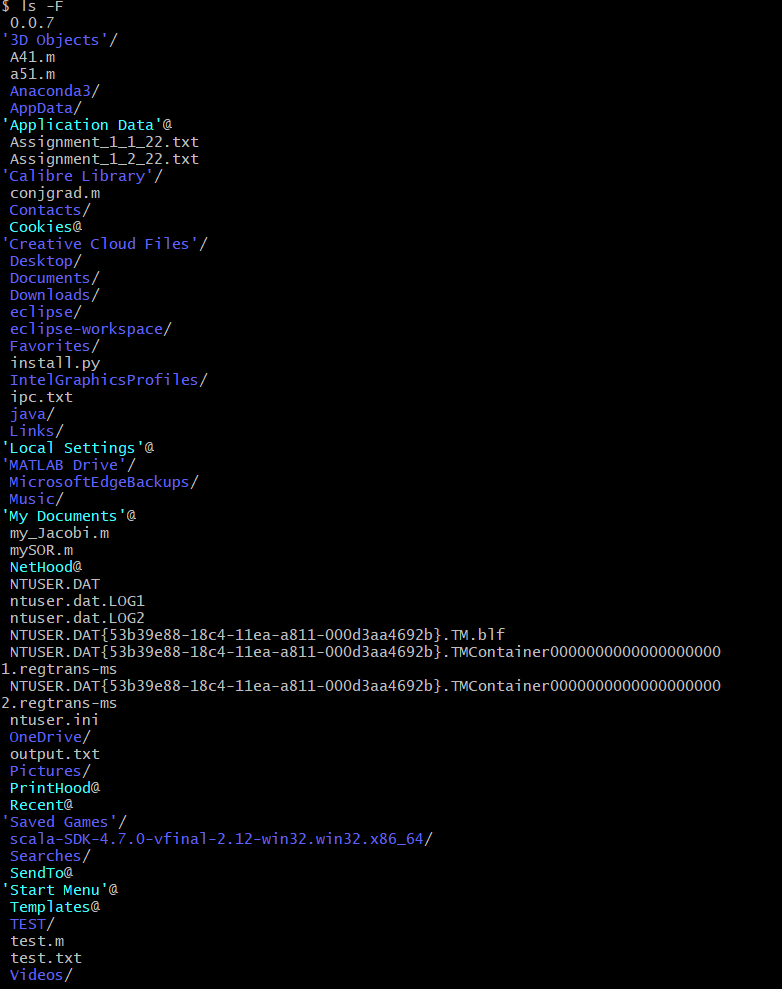
**$ls -a :** List all files including hidden file starting with `.`



**$ls -lh :** Shows the sizes of files and folders



**$ls -F :** Adds the `/' Character at the end each directory.

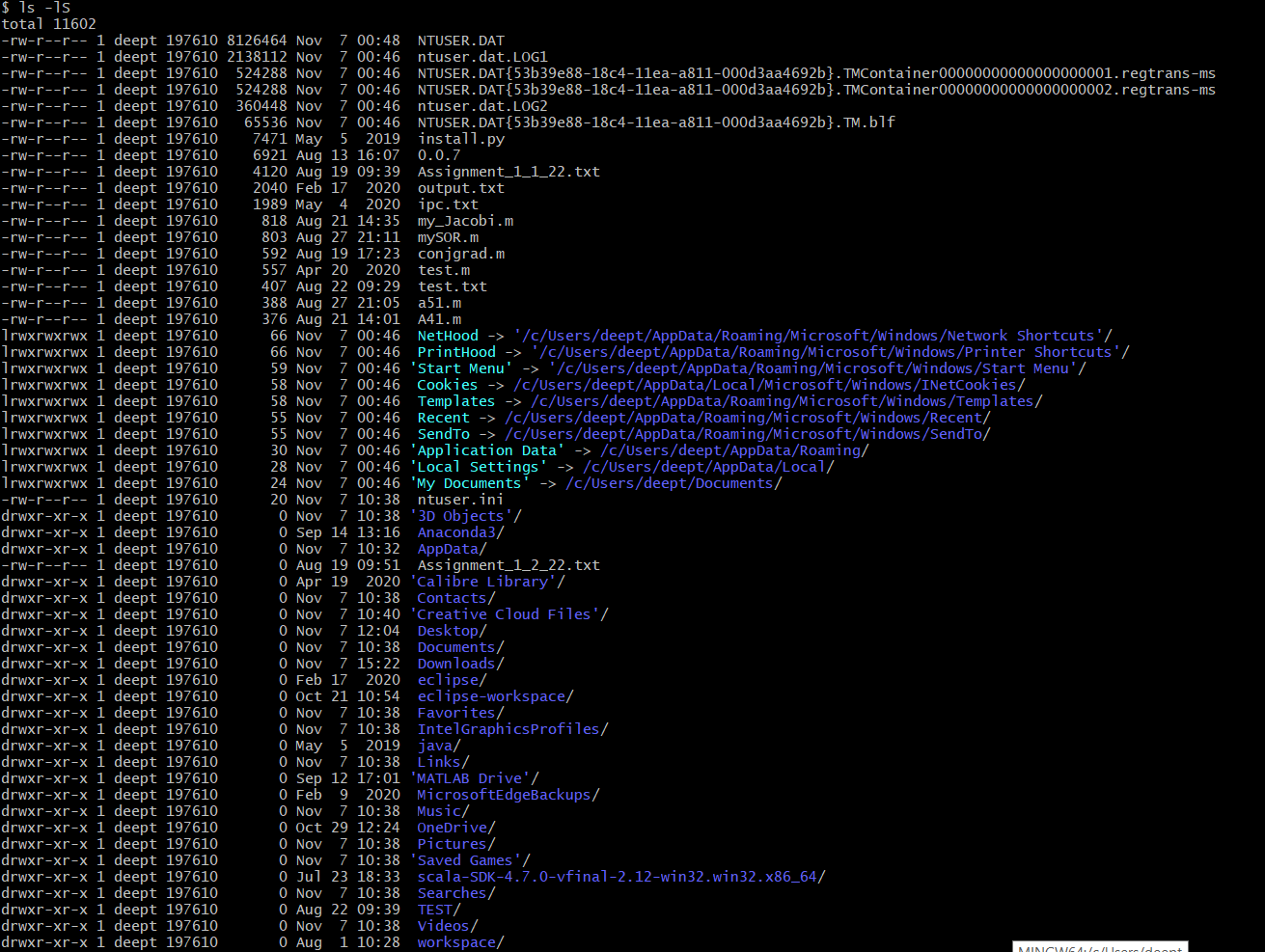


**$ls -R**

Lists folders and subfolders.

**$ls -lS**

Sorts the files according to their size in descending order.



**$ls --help**

Lists all the possible parameters and their definition



1. **less** : gives quick view of a file



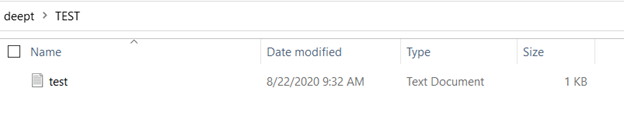


1. **more** : gives details about the files



1. **cp** : it its used to copy the file from one location to another





**$cp -i**

Prompt before overwrite.

**$cp -f**

If an existing destination file cannot be opened, remove it and

try again.

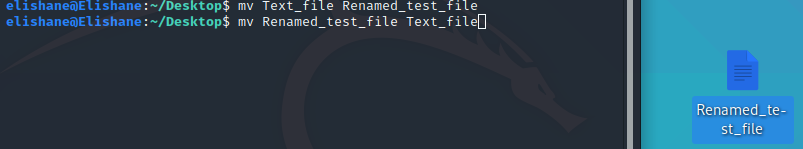
**$cp -n**

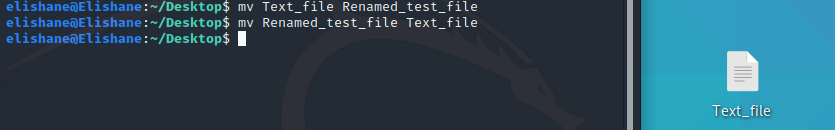
Do not overwrite an existing file.

**$cp --help**

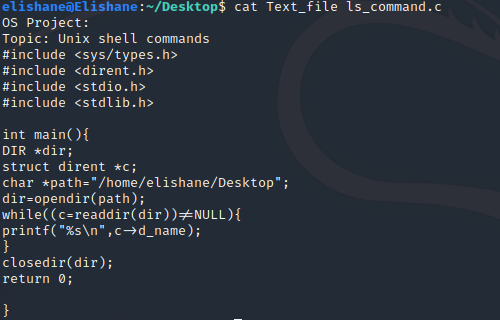
Gives details about other parameters.

1. **mv** : it is used to rename a file





1. **cat** : used to view multiple files at the same time



1. **mkdir** : it is used to make a directory



1. **cd** : it is used to change the directory



1. **rm** : it is used to remove a file



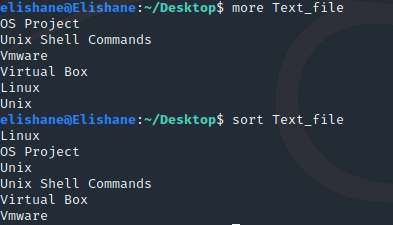
1. **rmdir** : it is used to remove a directory



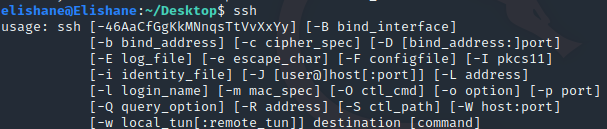
1. **pwd** : returns the present working directory



1. **sort** : prints the lines of text in ascending order



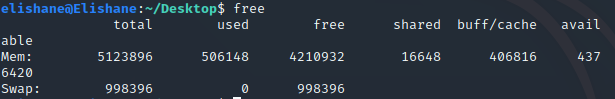
1. **ssh** : it is used to login into a remote system



1. **ftp** : to connect to remote ftp



1. **free** : shows free memory



1. **grep** : searches for given string in a file

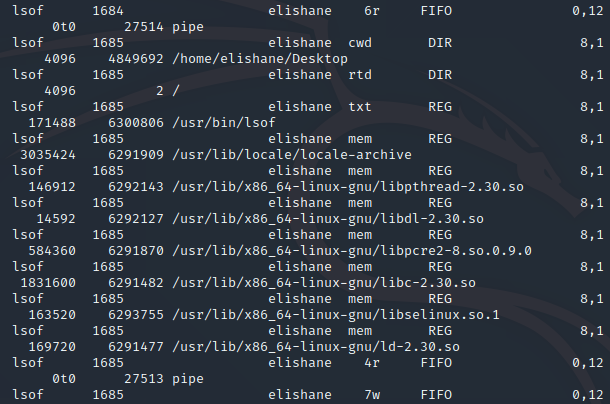


1. **find** : used for searching files

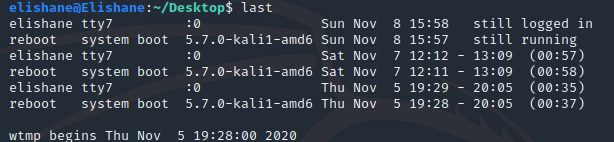


1. **lsof** : lists of opened files

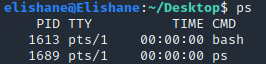




1. **last** : to watch users activity in the systems



1. **ps** : displays the process in the system



1. **stime()** : it is used for setting the time



1. **pause()** : it is used to suspend a file

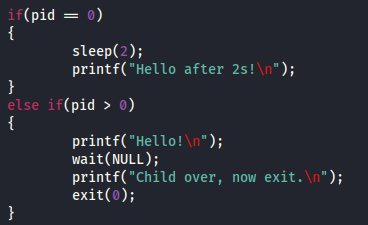
****

1. **fork()** : a new process can be created using fork



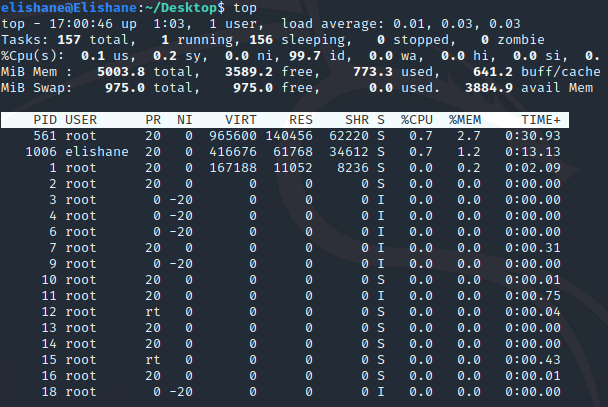


1. **exit()** : it is used to exit a process





1. **top** : Display and update information about the top CPU processes



**Few other commonly used UNIX commands and their functions:**

|  |  |
| --- | --- |
| **COMMAND** | **FUNCTION** |
| kill() | To terminate a process |
| access() | Checks access to a file |
| chmod() | Changes the mode of a file |
| chown() | Changes the ownership of a file |
| link() | Used for linking file names |
| open() | Used for opening files |
| chroot() | Changes the root directory of the file |

FEW UNIX COMMAND IMPLEMENTATIONS IN C:

1. **echo()**

#include <stdio.h>

int main(){

char str[50];

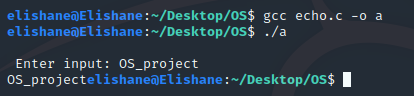
printf("\n Enter input: ");

scanf("%[^\n]+", str);

printf(str);

return 0;

}



1. **cd()**

#include<stdio.h>

// chdir function is declared

// inside this header

#include<unistd.h>

int main()

{

char s[100];

// printing current working directory

printf("%s\n", getcwd(s, 100));

// using the command

chdir("..");

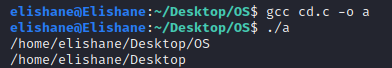
// printing current working directory

printf("%s\n", getcwd(s, 100));

// after chdir is executed

return 0;

}



1. **uptime()**

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int main(void) {

FILE \* uptimefile;

char uptime\_chr[28];

long uptime = 0;

if((uptimefile = fopen("/proc/uptime", "r")) == NULL)

perror("supt"), exit(EXIT\_FAILURE);

fgets(uptime\_chr, 12, uptimefile);

fclose(uptimefile);

uptime = strtol(uptime\_chr, NULL, 10);

printf("System up for %ld seconds, %ld hours\n", uptime, uptime / 3600);

exit(EXIT\_SUCCESS);

}



1. **ls()**

#include <sys/types.h>

#include <dirent.h>

#include <stdio.h>

#include <stdlib.h>

int main(){

DIR \*dir;

struct dirent \*c;

char \*path="/home/elishane/Desktop";

dir=opendir(path);

while((c=readdir(dir))!=NULL){

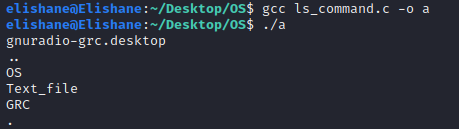
printf("%s\n",c->d\_name);

}

closedir(dir);

return 0;

}



1. **mkdir()**

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

struct stat st = {0};

int main(void)

{

//checked if directory is exists

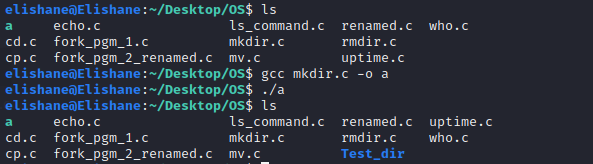
if (stat("Test\_dir", &st) == -1)

{

mkdir("Test\_dir", 0700);

}

}



1. **rmdir()**

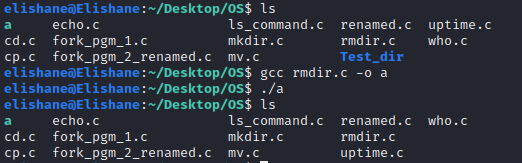
#include <unistd.h>

void main(){

int status;

status = rmdir("/home/elishane/Desktop/OS/Test\_dir");

}



1. **fork()**

#include<stdio.h>

#include<unistd.h>

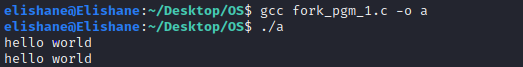
void main()

{

fork();

printf("hello world\n");

}



1. **mv() (renaming a file)**

#include<stdio.h>

int main()

{

// Old file name

char old\_name[] = "fork\_pgm\_2.c";

// Any string

char new\_name[] = "fork\_pgm\_2\_renamed.c";

int value;

// File name is changed here

value = rename(old\_name, new\_name);

// Print the result

if(!value)

{

printf("%s", "File name changed successfully");

}

else

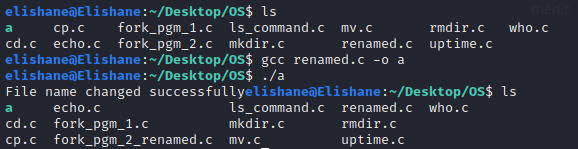
{

perror("Error");

}

return 0;

}



1. **who()**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

void main() {

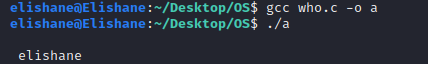
char \*buf;

buf=(char \*)malloc(10\*sizeof(char));

buf=getlogin();

printf("\n %s \n",buf);

}



1. **cp()**

#include <stdio.h>

#include <stdlib.h> // For exit()

int main()

{

FILE \*fptr1, \*fptr2;

char filename[100], c;

printf("Enter the filename to open for reading \n");

scanf("%s", filename);

// Open one file for reading

fptr1 = fopen(filename, "r");

if (fptr1 == NULL)

{

printf("Cannot open file %s \n", filename);

exit(0);

}

printf("Enter the filename to open for writing \n");

scanf("%s", filename);

// Open another file for writing

fptr2 = fopen(filename, "w");

if (fptr2 == NULL)

{

printf("Cannot open file %s \n", filename);

exit(0);

}

// Read contents from file

c = fgetc(fptr1);

while (c != EOF)

{

fputc(c, fptr2);

c = fgetc(fptr1);

}

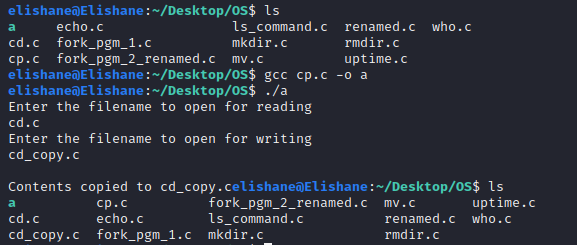
printf("\nContents copied to %s", filename);

fclose(fptr1);

fclose(fptr2);

return 0;

}



1. **mv() (moving a file)**

#include <stdio.h>

int main()

{

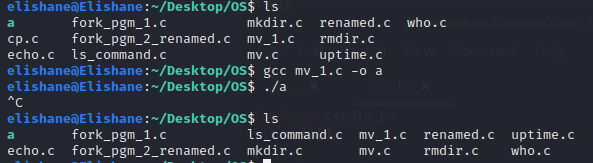
if ( rename("cd.c", "/home/elishane/Desktop/moved.c") )

perror( NULL );

getchar();

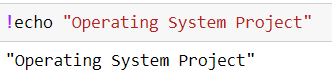
return 0;

}

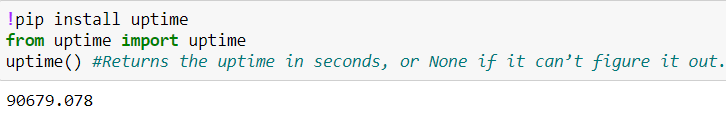


FEW UNIX COMMAND IMPLEMENTATIONS IN PYTHON:

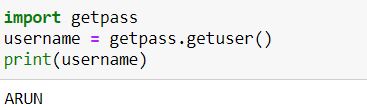
1. **echo :**



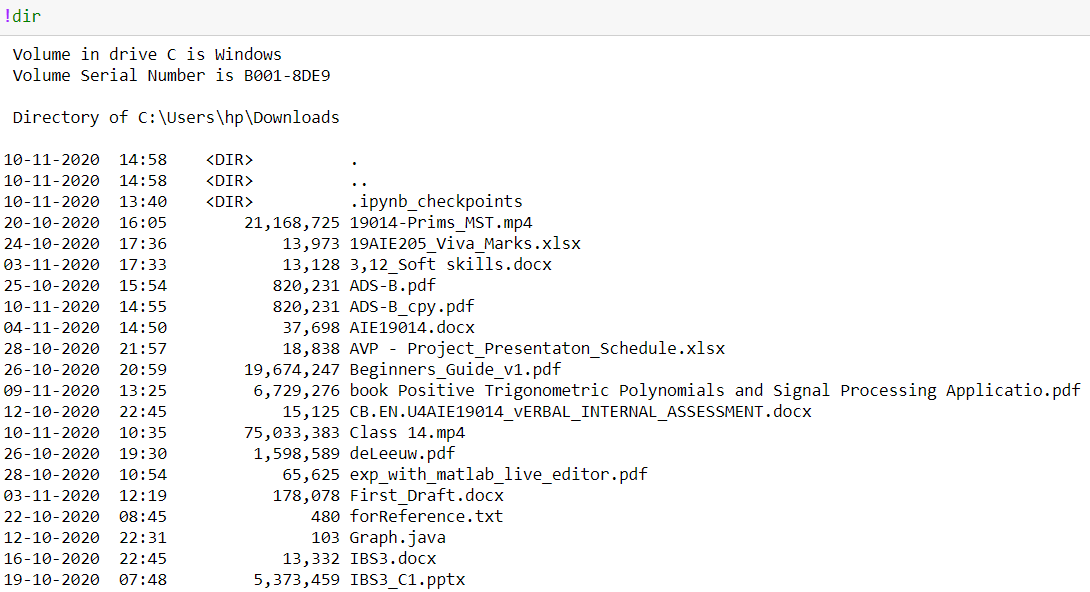
1. **uptime :**



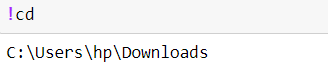
1. **Whoami :**



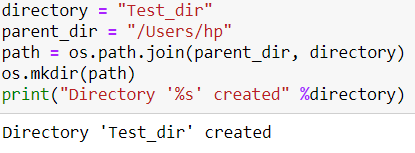
1. **ls :**



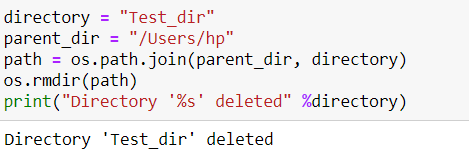
1. **Pwd:**



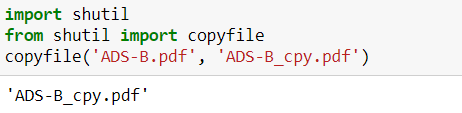
1. **Mkdir:**



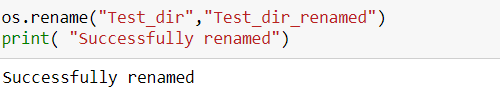
1. **Rmdir:**



1. **Cp:**



1. **Mv(renaming a file):**



FEW UNIX COMMAND IMPLEMENTATIONS IN JAVA:

1. **pwd :**

public class Jpwd {

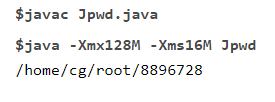
public static void main(String[] args) {

String pwd = System.getProperty("user.dir");

System.out.println(pwd);

}

}



1. **ls :**

import java.io.File;

public class Jls {

public static void main(String[] args) {

File dir = new File(System.getProperty("user.dir"));

String childs[] = dir.list();

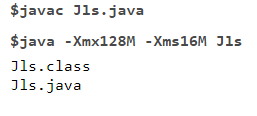
for(String child: childs){

System.out.println(child);

}

}

}



REFERENCES

1. <https://www.tutorialspoint.com/unix_commands/>
2. http://www.mathcs.emory.edu/~valerie/courses/fall10/155/resources/unix\_cheatsheet.html
3. https://stackoverflow.com/questions/7267295/how-can-i-copy-a-file-from-one-directory-to-another-in-c-c
4. https://www.daniweb.com/programming/software-development/threads/361633/copying-a-file-from-one-directory-to-other
5. https://silveiraneto.net/2008/10/07/example-of-unix-commands-implemented-in-java/