OS-Assignment 1 (UNIX Shell) - Report by Arnav Goel (2021519)

This is a report for OS-Assignment 1 by Arnav Goel (Roll No: 2021519) from CSAI, IIITD 25. In this report, I am gonna cover how I implemented my Shell and I am going to elaborate on how I implemented my Linux Commands. Through explaining how I implemented my Linux Commands, I am also going to cover how I handled corner cases/bugs/attacks.

General Description:

The shell is named *"pabloo"* and is authored by me. It supports 5 external commands i.e. mkdir, rm, cat, date and ls and these external commands are handled in 2 ways. If there is '&t' at the end of my command these external commands are handled by the "*pthread_create()" and "system()"* calls and if not, these are handled by *"fork()" and "execl()"* calls. Additionally, the shell handles three internal commands i.e. pwd, echo and cd. To exit the shell, one needs to enter the *"exit"* command into the shell.

External Commands:

1) mkdir:

- This command creates a directory as input by the user.
- I have handled the flags -v and -pv for this command.
- **Input format: mkdir [-v|-p] filename1 filename2…… || mkdir filename1 filename2……**
- As one can see, this can handle multiple directory creation through one line of command.
- **mkdir -v:** This command displays the file which is created with a message for it
- **mkdir -pv:** This command recursively creates directories one within the other. Eg if input is mkdir -pv arnav/goel/csai —> it creates arnav as a parent directory, followed by goel as its child and csai as its child.
- **Corner Cases handled:**
    1) If someone enters only the command mkdir, it throws an error saying that the operand is missing.
    2) If one enters the name of a file which already exists, it throws an error saying it can't be made.
    3) If you enter **mkdir -v** or **mkdir -pv** to the shell only, it reports an error saying illegal usage.

2) <u>rm:</u>

- This command deletes a file or directory depending on the flag set by the user.
- I have handled the flags -d and -v for this command.
- If one calls rm filename, "filename" has to be a file and can't be a directory. For deleting directories, -d flag should be set.
- **Input format: rm [-v|-d] filename1 filename2…… || rm filename1 filename2……**
- As one can see, this can handle multiple directory/file deletion through one line of command.
- **rm -v:** This command displays the file which is deleted with a message for it.
- **rm -d:** This command handles directory deletion only (assumption) and deletes the directories you enter after it.
- **Corner Cases handled:**
    1) If someone enters only the command **rm**, it throws an error saying that the operand is missing.
    2) If one enters the name of a file which doesn't exist, it throws an error saying the file does not exist.
    3) If one enters a directory with rm without the -d flag, it throws an error saying it is not a directory.

3) <u>cat:</u>

- This command reads a file and writes it to the stdout.
- I have handled the flags -n and -b for this command.
- If one enters only the command "cat" it starts taking input from the user on stdin and printing it out to stdout.
- **Input format: cat [-n|-b] filename**
- If one enters **cat filename** without any flag, it reads the file and prints its contents as it is to stdout.
- The stated above happens in the case of entering only **cat -n** and **cat -b** also. (assumption)
- **cat -n:** This command reads the contents of a file and numbers all the lines of the file and outputs the contents with the appropriate line numbers.
- **cat -b:** This command reads the contents of a file and numbers all the **non-blank** lines of the file and outputs the contents with the appropriate line numbers.
- **Corner Cases handled:**

    1) The **cat** command can only handle one file at a time and this assumption has been made to ensure the cleanliness of the stdout. So in case of more than one file, it gives an error message.
    2) If you call a file with **cat** which doesn't exist, it throws out an error message.
    3) Will give an error message if you call some other flag as 2nd argument.

4) <u>date:</u>

- This command prints the system date and time to the stdout with the timezone.
- I have handled the flags -r and -u for this command.
- If one enters only the command "date", it prints the current system date and time with IST to the stdout.
- **Input format: date || date [-u] || date [-r] filename**
- **date -u:** It prints the UTC system date and time to the stdout.
- **date -r filename:** This outputs the last modified date and time of the filename which is inputted.
- **Corner Cases handled:**
    1) The **date** command would give an error if you enter more than three commands as more arguments.
    2) Will give an error if you enter something after the **date -u.**
    3) Will give an error if the filename you enter does not exist in the current working directory.

5) <u>ls:</u>

- This command lists the contents of a directory on stdout.
- I have handled the flags -a and -i for this command.
- If one enters only the command "ls", it prints the contents of the current working directory. My assumption is that ls pathname is not supported and doesn't give any output.
- **Input format: ls || ls [-a|-i] ||**
- **ls -a:** It also prints the contents of the cwd which start with a dot.
- **ls -i:** This prints all the contents of the cwd with their inode numbers.
- **Corner Cases handled:**
    1) Will give an error if you enter more than 2 arguments in the terminal.
    2) Will give an error if you enter some other flag or option as your 2nd argument.

3) Will give an error if the filename you enter does not exist in the current working directory.

<u>Internal Commands:</u>

1) **cd:**

- This changes your directory.
- **Input format: cd || cd path_name || cd ..**
- **cd:** Changes directory to home directory
- **cd .. :** Goes to previous directory
- **cd path_name :** Goes to the path
- **Corner Cases handled:**
    1) Will give an error if the given path is not found from the current directory i.e. not a child directory.
    2) Will give an error if you enter a filename as a path which can't be entered.
    3) Will give an error if there are too many arguments.

2) **pwd:**

- This prints the current working directory of your terminal.
- I have handled flags **-L and -P** for this.
- **Input format: pwd || pwd [-L|-P]**
- -L and -P both remove white spaces and symbolic links to print cwd.
- **Corner Cases handled:**
    1) Gives too many arguments error
    2) Gives error if some other flag is entered.

3) **echo:**

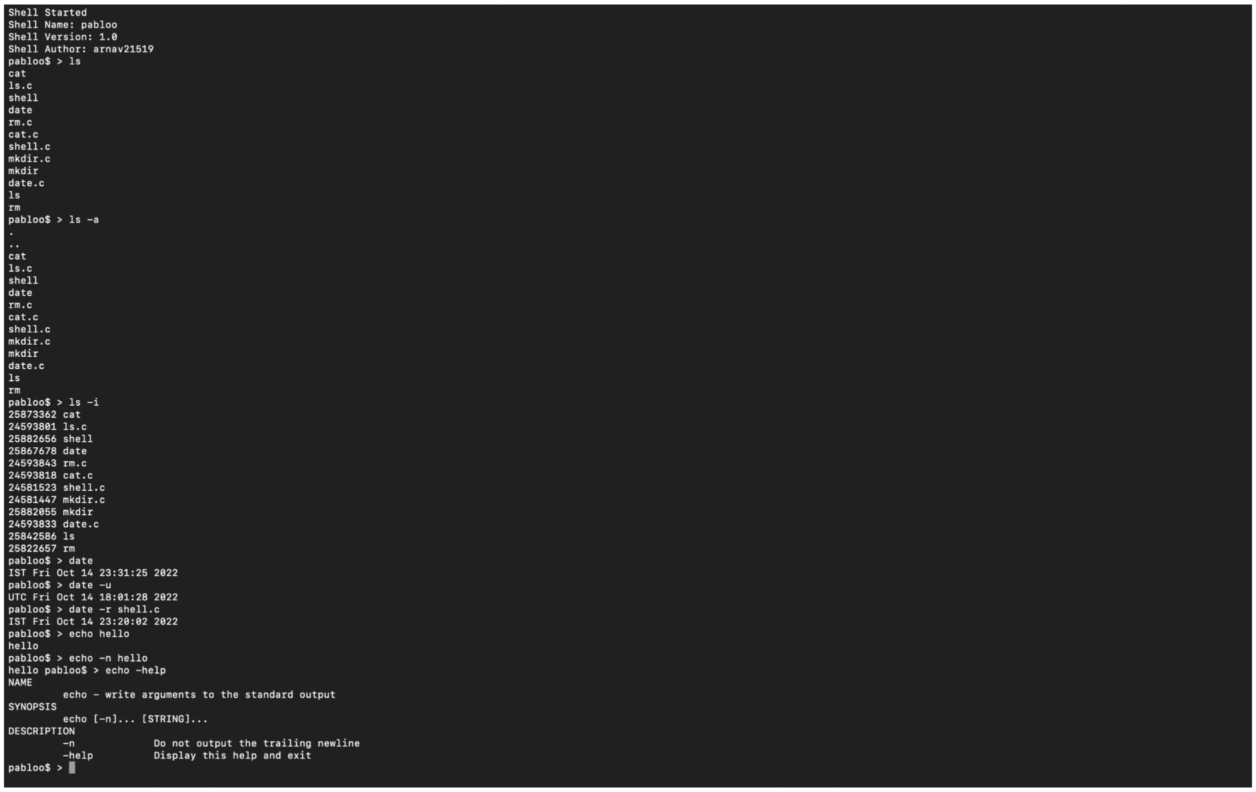
- Prints the input string with echo to the output.
- I have implemented -n and -help flags with this.
- **Input format: echo [-n] string || echo -help**
- echo -n removes the newline at the end of the output and echo -help prints the help page.
- **Corner Cases handled:**
    1) Gives too many arguments error
    2) Gives error if some other flag is entered.

General Error Handling:

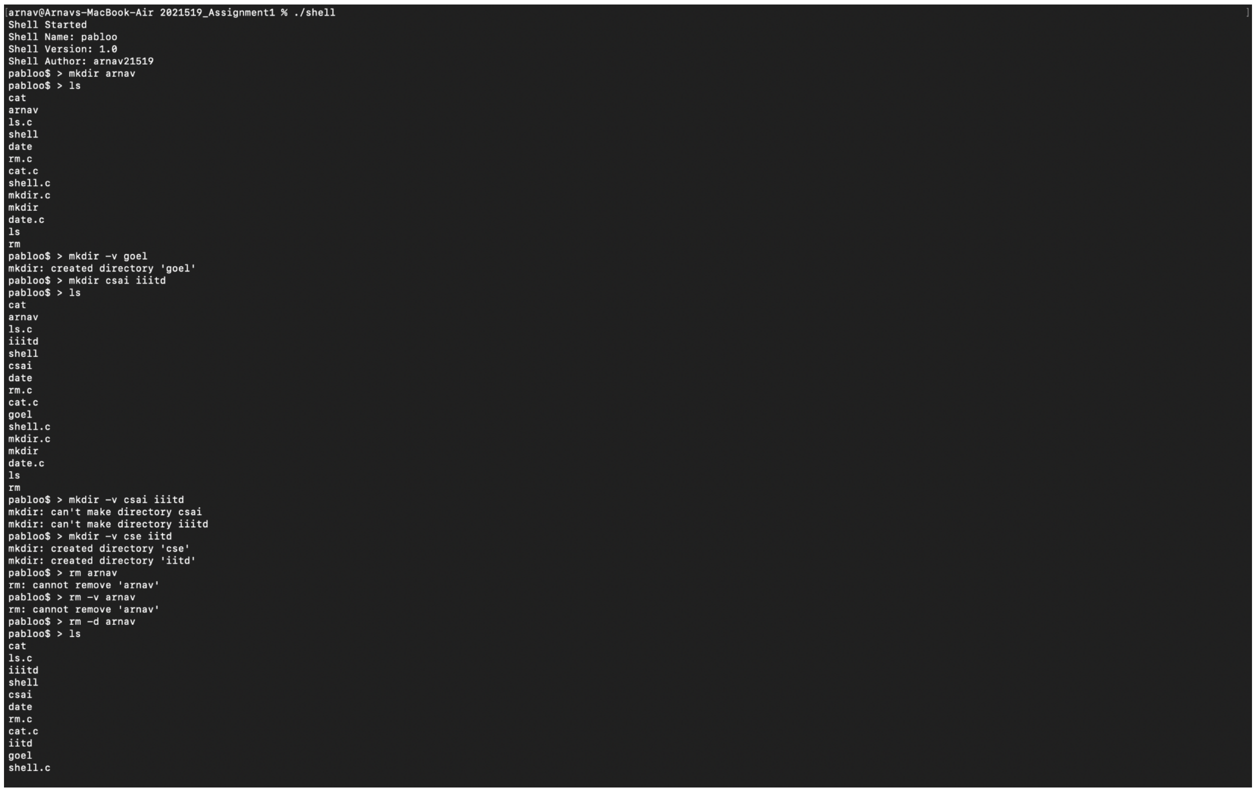
1) **No input to shell**
2) **Wrong command out of these 8 commands**
3) **Prints cwd with cd to notify the user of the current directory.**

Test Cases:

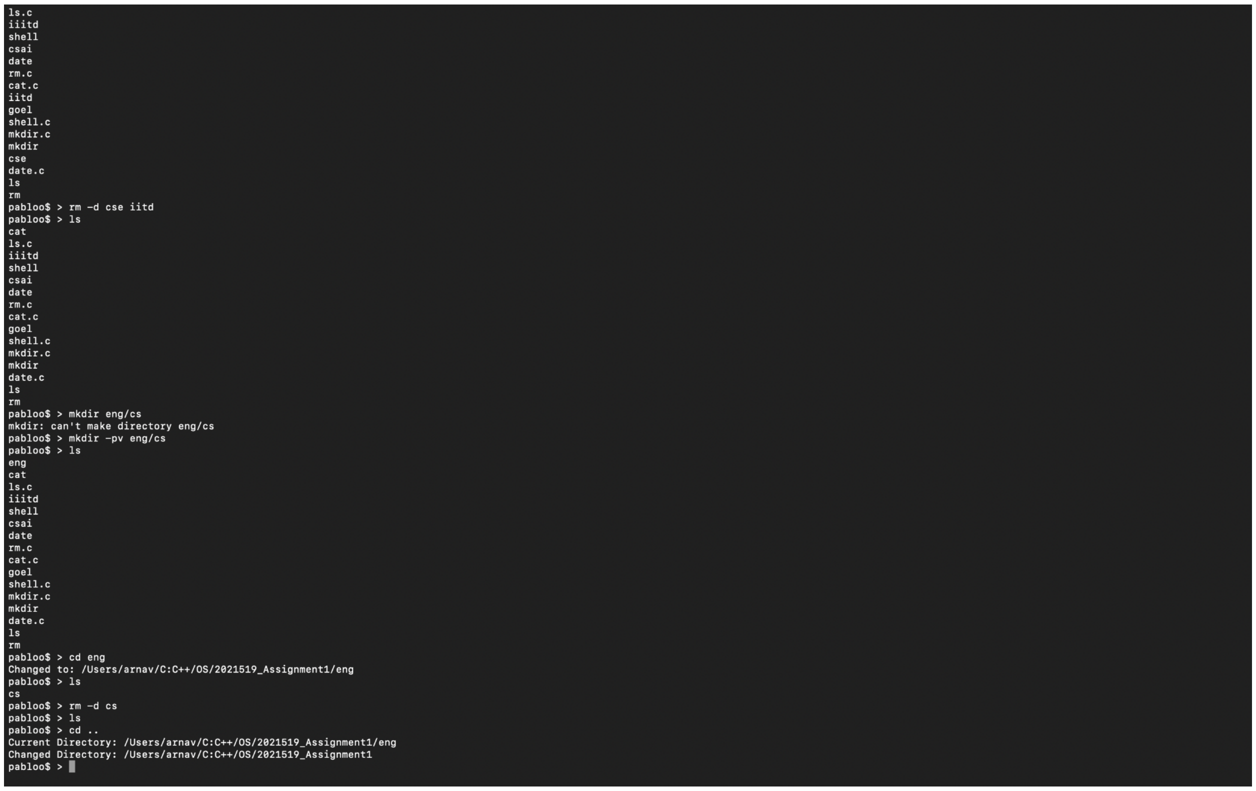
Using fork-exec:



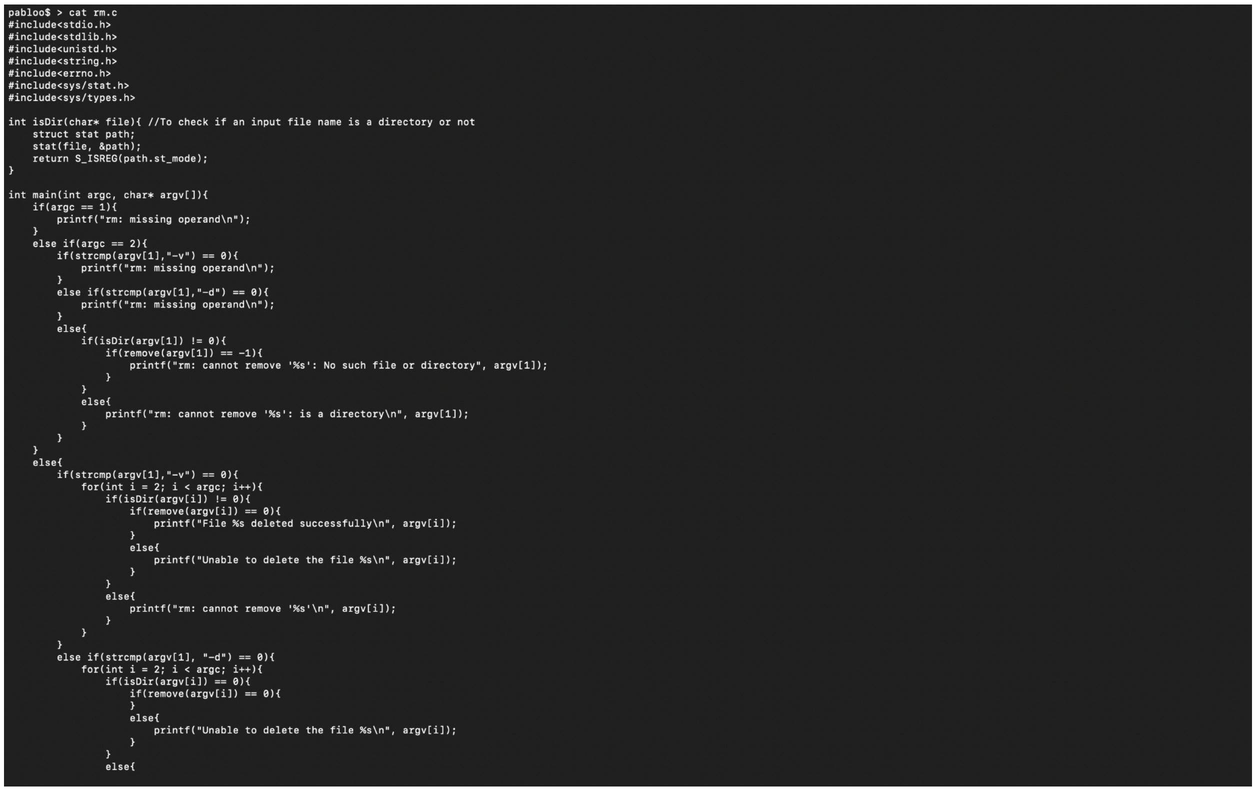
```
Shell Started
Shell Name: pabloo
Shell Version: 1.0
Shell Author: arnav21519
pabloo$ > ls
cat
ls.c
shell
date
rm.c
cat.c
shell.c
mkdir.c
mkdir
date.c
ls
rm
pabloo$ > ls -a
.
..
cat
ls.c
shell
date
rm.c
cat.c
shell.c
mkdir.c
mkdir
date.c
ls
rm
pabloo$ > ls -i
25873362 cat
24593801 ls.c
25882656 shell
25867678 date
24593843 rm.c
24593818 cat.c
24581523 shell.c
24581447 mkdir.c
25882055 mkdir
24593833 date.c
25842586 ls
25822657 rm
pabloo$ > date
IST Fri Oct 14 23:31:25 2022
pabloo$ > date -u
UTC Fri Oct 14 18:01:28 2022
pabloo$ > date -r shell.c
IST Fri Oct 14 23:20:02 2022
pabloo$ > echo hello
hello
pabloo$ > echo -n hello
hello pabloo$ > echo -help
NAME
        echo - write arguments to the standard output
SYNOPSIS
        echo [-n]... [STRING]...
DESCRIPTION
        -n              Do not output the trailing newline
        -help           Display this help and exit
pabloo$ > 
```



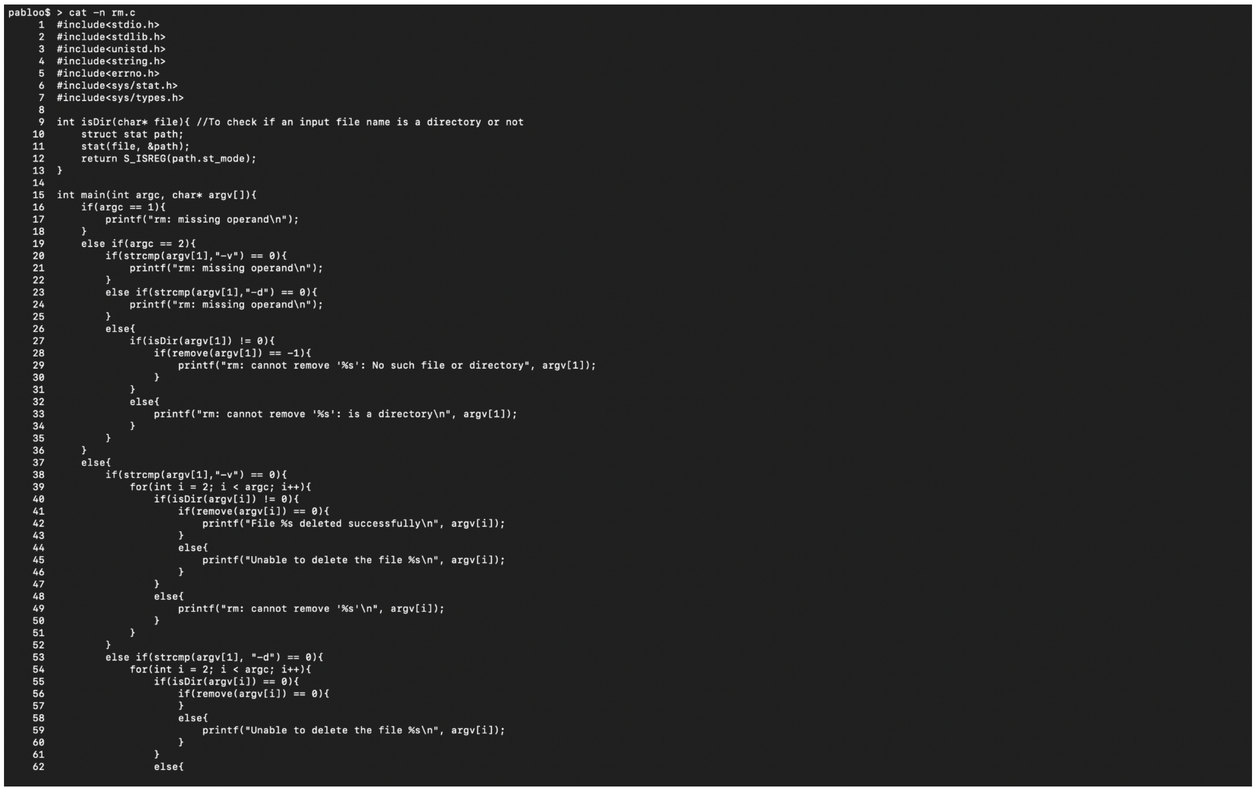
```
arnav@Arnavs-MacBook-Air 2021519_Assignment1 % ./shell
Shell Started
Shell Name: pabloo
Shell Version: 1.0
Shell Author: arnav21519
pabloo$ > mkdir arnav
pabloo$ > ls
cat
arnav
ls.c
shell
date
rm.c
cat.c
shell.c
mkdir.c
mkdir
date.c
ls
rm
pabloo$ > mkdir -v goel
mkdir: created directory 'goel'
pabloo$ > mkdir csai iiitd
pabloo$ > ls
cat
arnav
ls.c
iiitd
shell
csai
date
rm.c
cat.c
goel
shell.c
mkdir.c
mkdir
date.c
ls
rm
pabloo$ > mkdir -v csai iiitd
mkdir: can't make directory csai
mkdir: can't make directory iiitd
pabloo$ > mkdir -v cse iitd
mkdir: created directory 'cse'
mkdir: created directory 'iitd'
pabloo$ > rm arnav
rm: cannot remove 'arnav'
pabloo$ > rm -v arnav
rm: cannot remove 'arnav'
pabloo$ > rm -d arnav
pabloo$ > ls
cat
ls.c
iiitd
shell
csai
date
rm.c
cat.c
iitd
goel
shell.c
```



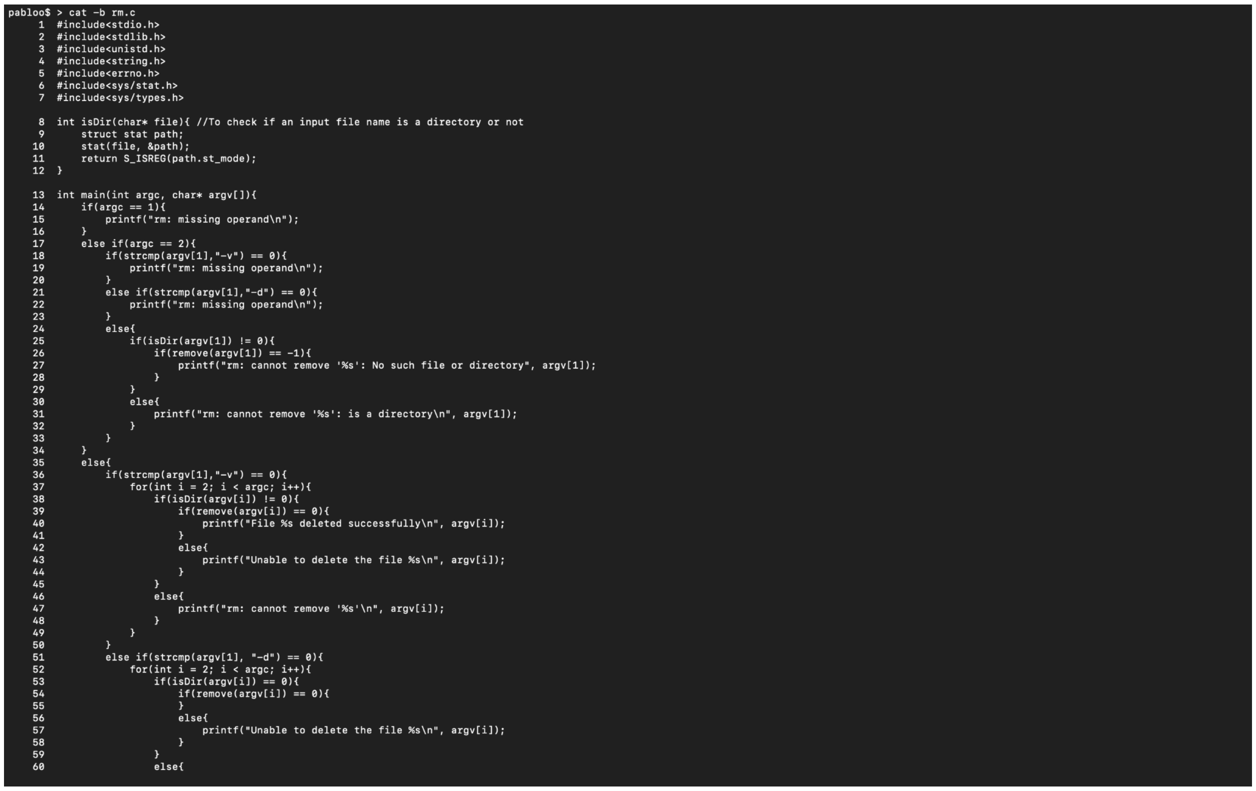
```
ls.c
iiitd
shell
csai
date
rm.c
cat.c
iitd
goel
shell.c
mkdir.c
mkdir
cse
date.c
ls
rm
pabloo$ > rm -d cse iitd
pabloo$ > ls
cat
ls.c
iiitd
shell
csai
date
rm.c
cat.c
goel
shell.c
mkdir.c
mkdir
date.c
ls
rm
pabloo$ > mkdir eng/cs
mkdir: can't make directory eng/cs
pabloo$ > mkdir -pv eng/cs
pabloo$ > ls
eng
cat
ls.c
iiitd
shell
csai
date
rm.c
cat.c
goel
shell.c
mkdir.c
mkdir
date.c
ls
rm
pabloo$ > cd eng
Changed to: /Users/arnav/C:C++/OS/2021519_Assignment1/eng
pabloo$ > ls
cs
pabloo$ > rm -d cs
pabloo$ > ls
pabloo$ > cd ..
Current Directory: /Users/arnav/C:C++/OS/2021519_Assignment1/eng
Changed Directory: /Users/arnav/C:C++/OS/2021519_Assignment1
pabloo$ > 
```



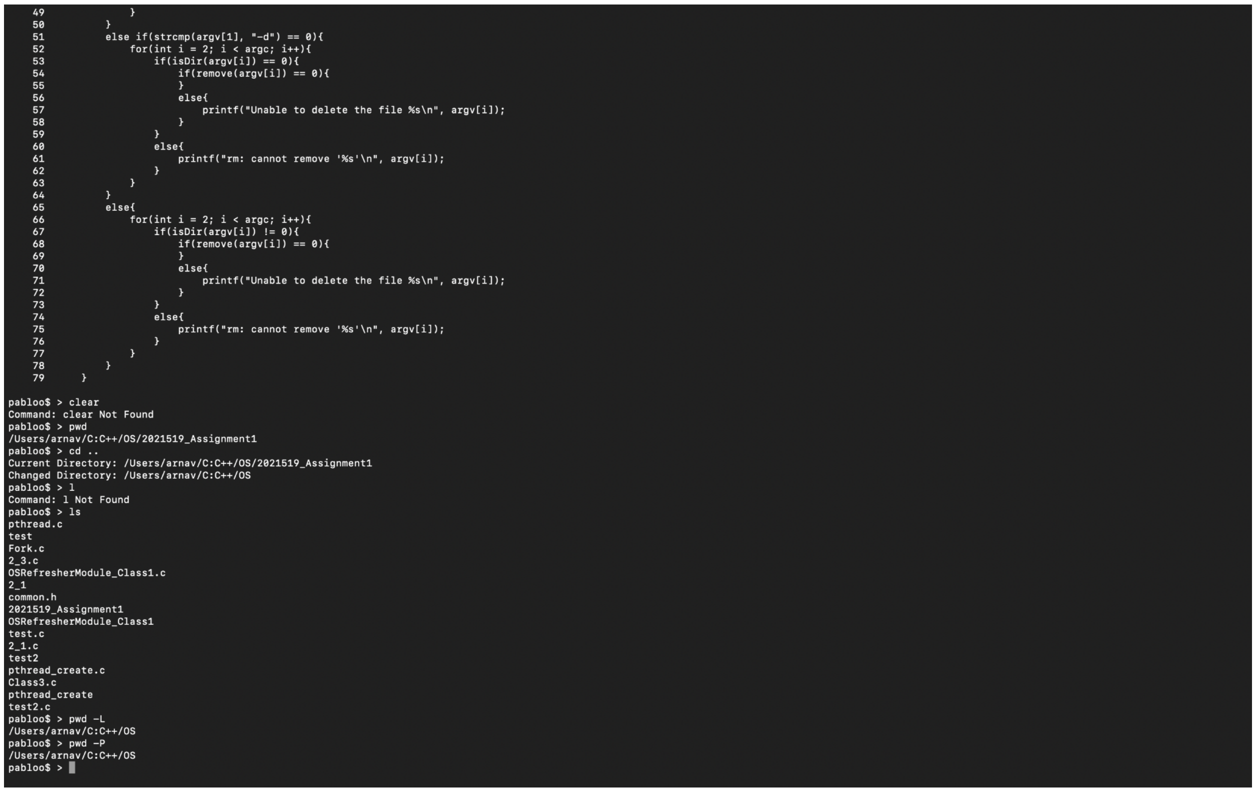
```
pabloo$ > cat rm.c
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>
#include<errno.h>
#include<sys/stat.h>
#include<sys/types.h>

int isDir(char* file){ //To check if an input file name is a directory or not
    struct stat path;
    stat(file, &path);
    return S_ISREG(path.st_mode);
}

int main(int argc, char* argv[]){
    if(argc == 1){
        printf("rm: missing operand\n");
    }
    else if(argc == 2){
        if(strcmp(argv[1],"-v") == 0){
            printf("rm: missing operand\n");
        }
        else if(strcmp(argv[1],"-d") == 0){
            printf("rm: missing operand\n");
        }
        else{
            if(isDir(argv[1]) != 0){
                if(remove(argv[1]) == -1){
                    printf("rm: cannot remove '%s': No such file or directory", argv[1]);
                }
            }
            else{
                printf("rm: cannot remove '%s': is a directory\n", argv[1]);
            }
        }
    }
    else{
        if(strcmp(argv[1],"-v") == 0){
            for(int i = 2; i < argc; i++){
                if(isDir(argv[i]) != 0){
                    if(remove(argv[i]) == 0){
                        printf("File %s deleted successfully\n", argv[i]);
                    }
                    else{
                        printf("Unable to delete the file %s\n", argv[i]);
                    }
                }
                else{
                    printf("rm: cannot remove '%s'\n", argv[i]);
                }
            }
        }
        else if(strcmp(argv[1], "-d") == 0){
            for(int i = 2; i < argc; i++){
                if(isDir(argv[i]) == 0){
                    if(remove(argv[i]) == 0){
                    }
                    else{
                        printf("Unable to delete the file %s\n", argv[i]);
                    }
                }
                else{
```



```
pabloo$ > cat -n rm.c
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>
#include<errno.h>
#include<sys/stat.h>
#include<sys/types.h>

int isDir(char* file){ //To check if an input file name is a directory or not
    struct stat path;
    stat(file, &path);
    return S_ISREG(path.st_mode);
}

int main(int argc, char* argv[]){
    if(argc == 1){
        printf("rm: missing operand\n");
    }
    else if(argc == 2){
        if(strcmp(argv[1],"-v") == 0){
            printf("rm: missing operand\n");
        }
        else if(strcmp(argv[1],"-d") == 0){
            printf("rm: missing operand\n");
        }
        else{
            if(isDir(argv[1]) != 0){
                if(remove(argv[1]) == -1){
                    printf("rm: cannot remove '%s': No such file or directory", argv[1]);
                }
            }
            else{
                printf("rm: cannot remove '%s': is a directory\n", argv[1]);
            }
        }
    }
    else{
        if(strcmp(argv[1],"-v") == 0){
            for(int i = 2; i < argc; i++){
                if(isDir(argv[i]) != 0){
                    if(remove(argv[i]) == 0){
                        printf("File %s deleted successfully\n", argv[i]);
                    }
                    else{
                        printf("Unable to delete the file %s\n", argv[i]);
                    }
                }
                else{
                    printf("rm: cannot remove '%s'\n", argv[i]);
                }
            }
        }
        else if(strcmp(argv[1], "-d") == 0){
            for(int i = 2; i < argc; i++){
                if(isDir(argv[i]) == 0){
                    if(remove(argv[i]) == 0){
                    }
                    else{
                        printf("Unable to delete the file %s\n", argv[i]);
                    }
                }
                else{
```

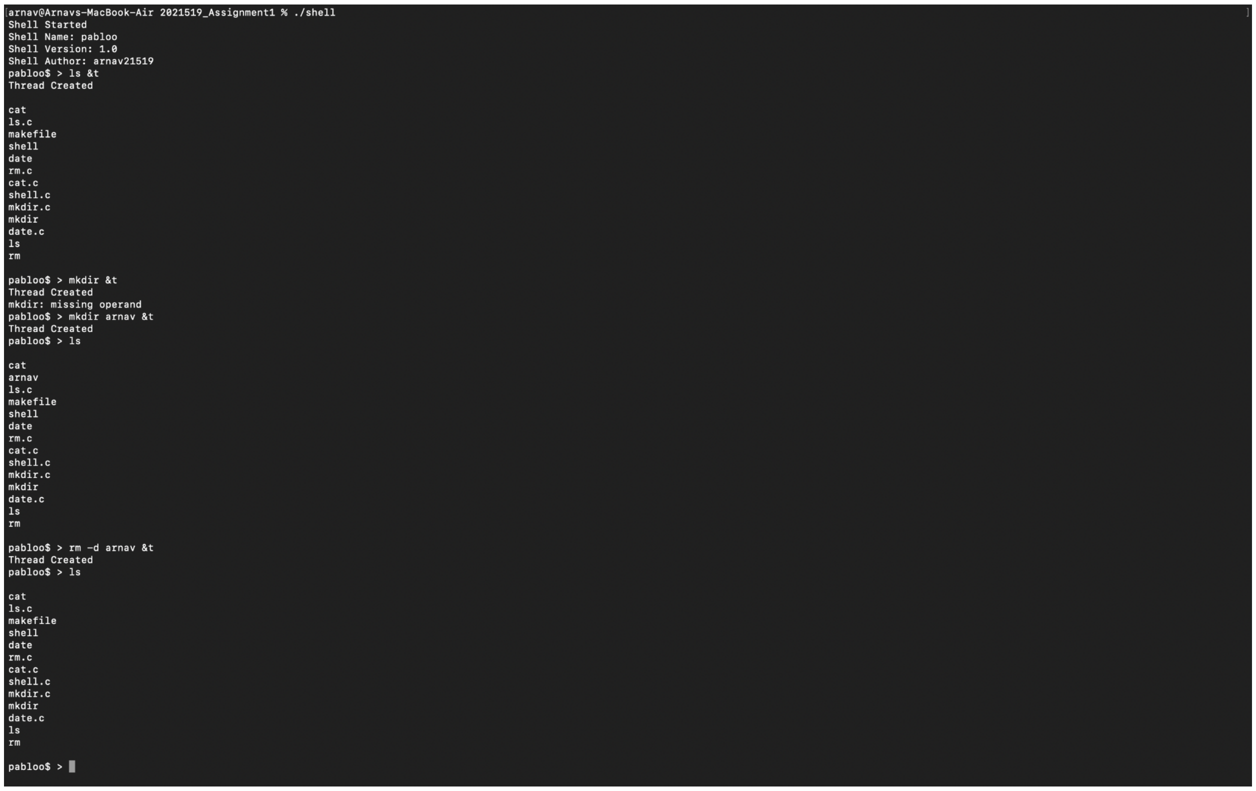


```
pabloo$ > cat -b rm.c
     1	#include<stdio.h>
     2	#include<stdlib.h>
     3	#include<unistd.h>
     4	#include<string.h>
     5	#include<errno.h>
     6	#include<sys/stat.h>
     7	#include<sys/types.h>

     8	int isDir(char* file){ //To check if an input file name is a directory or not
     9	    struct stat path;
    10	    stat(file, &path);
    11	    return S_ISREG(path.st_mode);
    12	}

    13	int main(int argc, char* argv[]){
    14	    if(argc == 1){
    15	        printf("rm: missing operand\n");
    16	    }
    17	    else if(argc == 2){
    18	        if(strcmp(argv[1],"-v") == 0){
    19	            printf("rm: missing operand\n");
    20	        }
    21	        else if(strcmp(argv[1],"-d") == 0){
    22	            printf("rm: missing operand\n");
    23	        }
    24	        else{
    25	            if(isDir(argv[1]) != 0){
    26	                if(remove(argv[1]) == -1){
    27	                    printf("rm: cannot remove '%s': No such file or directory", argv[1]);
    28	                }
    29	            }
    30	            else{
    31	                printf("rm: cannot remove '%s': is a directory\n", argv[1]);
    32	            }
    33	        }
    34	    }
    35	    else{
    36	        if(strcmp(argv[1],"-v") == 0){
    37	            for(int i = 2; i < argc; i++){
    38	                if(isDir(argv[i]) != 0){
    39	                    if(remove(argv[i]) == 0){
    40	                        printf("File %s deleted successfully\n", argv[i]);
    41	                    }
    42	                    else{
    43	                        printf("Unable to delete the file %s\n", argv[i]);
    44	                    }
    45	                }
    46	                else{
    47	                    printf("rm: cannot remove '%s'\n", argv[i]);
    48	                }
    49	            }
    50	        }
    51	        else if(strcmp(argv[1], "-d") == 0){
    52	            for(int i = 2; i < argc; i++){
    53	                if(isDir(argv[i]) == 0){
    54	                    if(remove(argv[i]) == 0){
    55	                    }
    56	                    else{
    57	                        printf("Unable to delete the file %s\n", argv[i]);
    58	                    }
    59	                }
    60	                else{
```

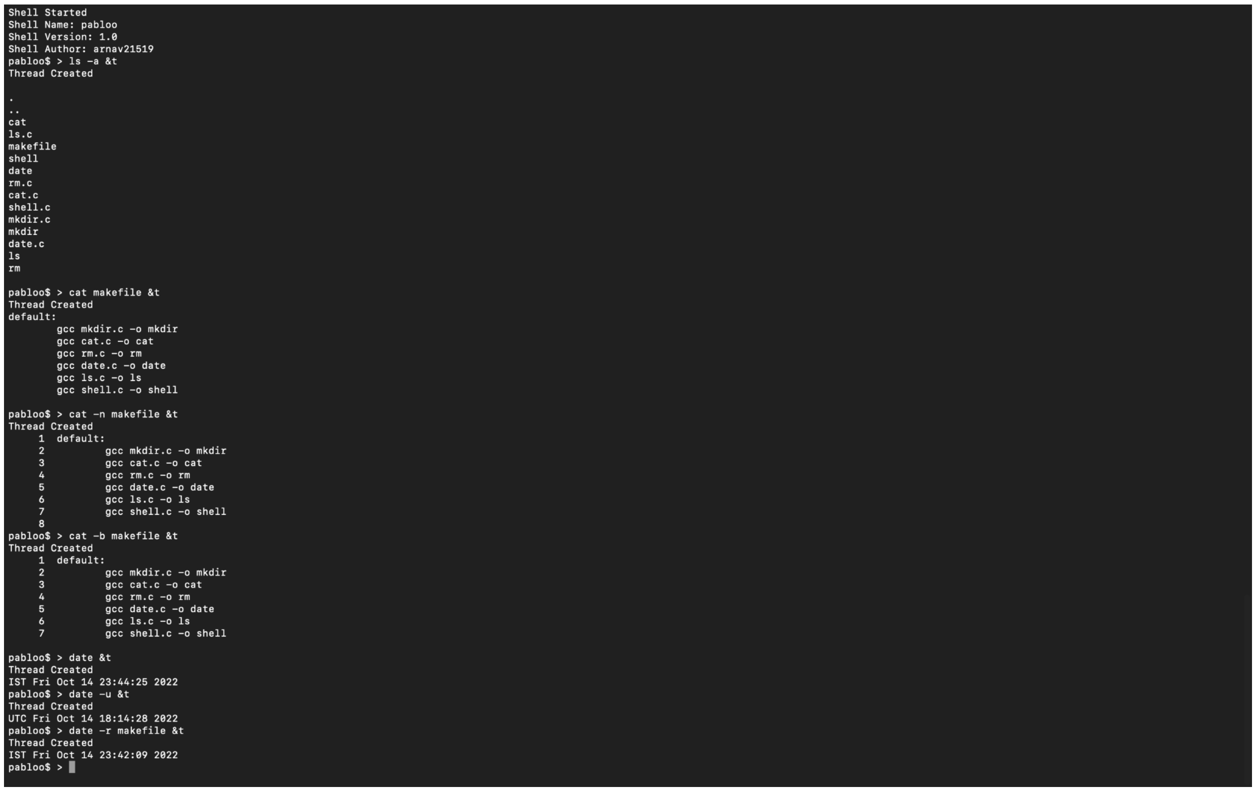


```
    49	            }
    50	        }
    51	        else if(strcmp(argv[1], "-d") == 0){
    52	            for(int i = 2; i < argc; i++){
    53	                if(isDir(argv[i]) == 0){
    54	                    if(remove(argv[i]) == 0){
    55	                    }
    56	                    else{
    57	                        printf("Unable to delete the file %s\n", argv[i]);
    58	                    }
    59	                }
    60	                else{
    61	                    printf("rm: cannot remove '%s'\n", argv[i]);
    62	                }
    63	            }
    64	        }
    65	        else{
    66	            for(int i = 2; i < argc; i++){
    67	                if(isDir(argv[i]) != 0){
    68	                    if(remove(argv[i]) == 0){
    69	                    }
    70	                    else{
    71	                        printf("Unable to delete the file %s\n", argv[i]);
    72	                    }
    73	                }
    74	                else{
    75	                    printf("rm: cannot remove '%s'\n", argv[i]);
    76	                }
    77	            }
    78	        }
    79	    }

pabloo$ > clear
Command: clear Not Found
pabloo$ > pwd
/Users/arnav/C:C++/OS/2021519_Assignment1
pabloo$ > cd ..
Current Directory: /Users/arnav/C:C++/OS/2021519_Assignment1
Changed Directory: /Users/arnav/C:C++/OS
pabloo$ > l
Command: l Not Found
pabloo$ > ls
pthread.c
test
Fork.c
2_3.c
OSRefresherModule_Class1.c
2_1
common.h
2021519_Assignment1
OSRefresherModule_Class1
test.c
2_1.c
test2
pthread_create.c
Class3.c
pthread_create
test2.c
pabloo$ > pwd -L
/Users/arnav/C:C++/OS
pabloo$ > pwd -P
/Users/arnav/C:C++/OS
pabloo$ > █
```

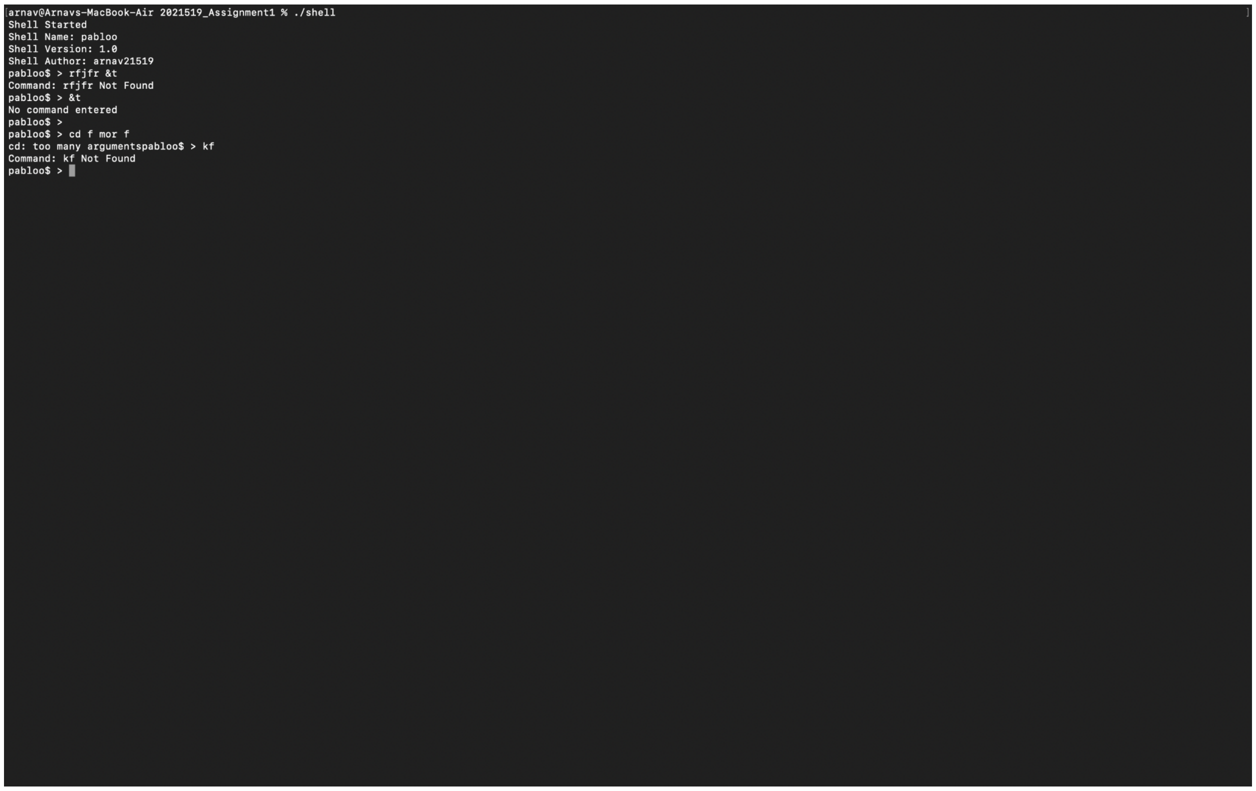
Using pthread_create() and system():



```
arnav@Arnavs-MacBook-Air 2021519_Assignment1 % ./shell
Shell Started
Shell Name: pabloo
Shell Version: 1.0
Shell Author: arnav21519
pabloo$ > ls &t
Thread Created

cat
ls.c
makefile
shell
date
rm.c
cat.c
shell.c
mkdir.c
mkdir
date.c
ls
rm

pabloo$ > mkdir &t
Thread Created
mkdir: missing operand
pabloo$ > mkdir arnav &t
Thread Created
pabloo$ > ls

cat
arnav
ls.c
makefile
shell
date
rm.c
cat.c
shell.c
mkdir.c
mkdir
date.c
ls
rm

pabloo$ > rm -d arnav &t
Thread Created
pabloo$ > ls

cat
ls.c
makefile
shell
date
rm.c
cat.c
shell.c
mkdir.c
mkdir
date.c
ls
rm

pabloo$ >
```



```
Shell Started
Shell Name: pabloo
Shell Version: 1.0
Shell Author: arnav21519
pabloo$ > ls -a &t
Thread Created

.
..
cat
ls.c
makefile
shell
date
rm.c
cat.c
shell.c
mkdir.c
mkdir
date.c
ls
rm

pabloo$ > cat makefile &t
Thread Created
default:
        gcc mkdir.c -o mkdir
        gcc cat.c -o cat
        gcc rm.c -o rm
        gcc date.c -o date
        gcc ls.c -o ls
        gcc shell.c -o shell

pabloo$ > cat -n makefile &t
Thread Created
     1  default:
     2          gcc mkdir.c -o mkdir
     3          gcc cat.c -o cat
     4          gcc rm.c -o rm
     5          gcc date.c -o date
     6          gcc ls.c -o ls
     7          gcc shell.c -o shell
     8
pabloo$ > cat -b makefile &t
Thread Created
     1  default:
     2          gcc mkdir.c -o mkdir
     3          gcc cat.c -o cat
     4          gcc rm.c -o rm
     5          gcc date.c -o date
     6          gcc ls.c -o ls
     7          gcc shell.c -o shell

pabloo$ > date &t
Thread Created
IST Fri Oct 14 23:44:25 2022
pabloo$ > date -u &t
Thread Created
UTC Fri Oct 14 18:14:28 2022
pabloo$ > date -r makefile &t
Thread Created
IST Fri Oct 14 23:42:09 2022
pabloo$ >
```



```
arnav@Arnavs-MacBook-Air 2021519_Assignment1 % ./shell
Shell Started
Shell Name: pabloo
Shell Version: 1.0
Shell Author: arnav21519
pabloo$ > rfjfr &t
Command: rfjfr Not Found
pabloo$ > &t
No command entered
pabloo$ >
pabloo$ > cd f mor f
cd: too many argumentspabloo$ > kf
Command: kf Not Found
pabloo$ >
```