ASSIGNMENT 2
22/08/23

NAME : SHRESTH SONKAR
REGNO : 20214272
GROUP : CS5D
TOPIC : OS LAB
CODE : CS-15203

```c
//Q1
/*
In this assignment we will start writing a command
interpreter (Shell). The shell will give a prompt for
the user to type in a command (from a set of commands),
take the command, execute it, and then give the prompt
back for the next command (i.e., actually give the
functionality of a shell).
Your program should do the following:
Give a prompt "myshell$" for the user to type in a
command Implement the following built in commands:
(a) cd < dir > : changes the directory to "dir"
(b) pwd : prints the current directory
(c) mkdir < dir > : creates a directory called "dir"
(d) rmdir < dir > : removes the directory called "dir"
(e) ls: lists the files in the current directory. It
should support both ls without any option and with the
option \-l"
(f) exit: exits the shell
The commands are the same as the corresponding Linux
commands by the same name. Do "man" to see the
descriptions. You can use the standard system calls
chdir, getcwd, mkdir, rmdir, readdir etc. toimplement
the calls (standard C library functions are available
for these; look them up). These commands are called
built in commands since your shell program will have a
function correspondingto each of these commands to
execute.
*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <dirent.h>
#include <sys/types.h>
#include <sys/stat.h>

#define PROMPT "myshell $ "
#define BUFFERSIZE 150

void CD(char *command) {
    char dirname[BUFFERSIZE / 3];
    strcpy(dirname, command + 3);
    int ret = chdir(dirname);
```

```
    if (ret)
        printf("%s: No such directory\n", dirname);
}

void PWD(char *command) {
    char cwd[BUFFERSIZE];
    getcwd(cwd, BUFFERSIZE);
    printf("%s\n", cwd);
}

void MKDIR(char *command) {
    char dirname[BUFFERSIZE / 3];
    strcpy(dirname, command + 6);
    int status = mkdir(dirname, S_IRWXU | S_IRGRP |
S_IXGRP | S_IROTH | S_IXOTH);
    if (status)
        printf("%s: Directory could not be created due
to some error!\n", dirname);
}

void RMDIR(char *command) {
    char dirname[BUFFERSIZE / 3];
    strcpy(dirname, command + 6);
    int status = rmdir(dirname);
    if (status)
        printf("%s: Directory could not be removed due
to some error!\n", dirname);
}

void LS(char *command) {
    struct dirent *dp;
    DIR *dirp = opendir(".");
    int count = 0;
    if (dirp != NULL) {
        while ((dp = readdir(dirp)) != NULL) {
            if (dp->d_name[0] != '.') {
                if (++count % 4)
                    printf("%-16s", dp->d_name);
                else
                    printf("%-16s\n", dp->d_name);
            }
        }

        closedir(dirp);
        if (count % 4 != 0)
```

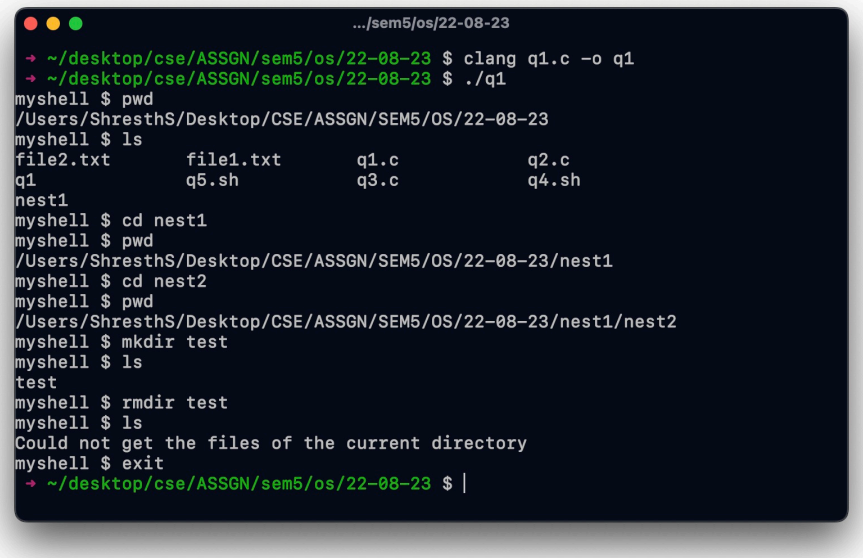
```
            printf("\n");
        else
            printf("Could not get the files of the
current directory\n");
    }
}

int main() {
    char command[BUFFERSIZE];
    int status;
    pid_t pid;

    while (1) {
        printf("%s", PROMPT);
        fgets(command, BUFFERSIZE, stdin);
        command[strlen(command) - 1] = '\0';

        if (strcmp(command, "exit") == 0)
            break;

        else if (strstr(command, "cd") != NULL) {
            CD(command);
        } else if (strcmp(command, "pwd") == 0) {
            PWD(command);
        } else if (strstr(command, "mkdir") != NULL) {
            MKDIR(command);
        } else if (strstr(command, "rmdir") != NULL) {
            RMDIR(command);
        } else if (strcmp(command, "ls") == 0) {
            LS(command);
        }
    }
}
```



```c
//Q2
/*
 * Write a C program that finds a file in a file-tree
starting from a given directory. The name of the file
for which we are searching for, as well as the name of
the starting directory should be read from the command
line. Optionally, the name of the file can be specified
as a pattern using the `*'character.
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>
#include <sys/stat.h>

int match(const char *pattern, const char *filename) {

    if (*pattern == '\0' && *filename == '\0') {
        return 1;
    }

    if (*pattern == '*' && *(pattern + 1) != '\0' &&
*filename == '\0') {
        return match(pattern + 1, filename);
    }

    if (*pattern == *filename) {
        return match(pattern + 1, filename + 1);
    }

    return 0;
}

void searchFile(const char *dirname, const char
*filename) {
    DIR *dir = opendir(dirname);
    if (dir == NULL) {
        perror("Error opening directory");
        exit(EXIT_FAILURE);
    }

    struct dirent *entry;
    while ((entry = readdir(dir)) != NULL) {
```

```
        if (strcmp(entry->d_name, ".") == 0 ||
strcmp(entry->d_name, "..") == 0) {
            continue;
        }

        char path[1024];
        snprintf(path, sizeof(path), "%s/%s", dirname,
entry->d_name);

        struct stat statbuf;
        if (stat(path, &statbuf) == -1) {
            perror("Error stat");
            continue;
        }

        if (S_ISDIR(statbuf.st_mode)) {
            searchFile(path, filename);
        } else if (S_ISREG(statbuf.st_mode)) {
            if (match(filename, entry->d_name)) {
                printf("Found: %s/%s\n", dirname,
entry->d_name);
            }
        }
    }

    closedir(dir);
}

int main(int argc, char *argv[]) {
    if (argc != 3) {
        fprintf(stderr, "Usage: %s <filename>
<starting_directory>\n", argv[0]);
        return EXIT_FAILURE;
    }

    const char *filename = argv[1];
    const char *startingDir = argv[2];

    searchFile(startingDir, filename);

    return EXIT_SUCCESS;
}
```

```
.../22-08-23/nest1/nest2

➜ ~/desktop/cse/ASSGN/sem5/os/22-08-23 $ clang q2.c -o q2
➜ ~/desktop/cse/ASSGN/sem5/os/22-08-23 $ ./q2 tofind.txt nest1
Found: nest1/nest2/tofind.txt
➜ ~/desktop/cse/ASSGN/sem5/os/22-08-23 $ ./q2 thisfiledoesnotexist.txt nest1
➜ ~/desktop/cse/ASSGN/sem5/os/22-08-23 $ ls
file1.txt nest1     q1.c      q2.c      q4.sh
file2.txt q1        q2        q3.c      q5.sh
➜ ~/desktop/cse/ASSGN/sem5/os/22-08-23 $ cd nest1/nest2
➜ ~/desktop/cse/ASSGN/sem5/os/22-08-23/nest1/nest2 $ ls
tofind.txt
➜ ~/desktop/cse/ASSGN/sem5/os/22-08-23/nest1/nest2 $
```

```c
//Q3
/*
 * Write a C program that deletes a directory with all
its subfolders. The name of the directory should be
read from the command line.
 */

#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <string.h>

void deleteDirectory(const char *dirname) {
    DIR *dir = opendir(dirname);
    if (dir == NULL) {
        perror("Error opening directory");
        exit(EXIT_FAILURE);
    }

    struct dirent *entry;
    while ((entry = readdir(dir)) != NULL) {
        if (strcmp(entry->d_name, ".") == 0 ||
strcmp(entry->d_name, "..") == 0) {
            continue;
        }

        char path[1024];
        snprintf(path, sizeof(path), "%s/%s", dirname,
entry->d_name);

        if (remove(path) == -1) {
            perror("Error removing file");
        }
    }

    closedir(dir);

    if (rmdir(dirname) == -1) {
        perror("Error removing directory");
        exit(EXIT_FAILURE);
    }
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
```

```c
        fprintf(stderr, "Usage: %s <directory_name>\n",
argv[0]);
        return EXIT_FAILURE;
    }

    const char *dirname = argv[1];

    deleteDirectory(dirname);

    printf("Directory '%s' and its subfolders have been
deleted.\n", dirname);

    return EXIT_SUCCESS;
}
```

```
.../sem5/os/22-08-23
 → ~/desktop/cse/ASSGN/sem5/os/22-08-23 $ ls
file1.txt nest1     q1.c      q2.c      q3.c      q5.sh
file2.txt q1        q2        q3        q4.sh
 → ~/desktop/cse/ASSGN/sem5/os/22-08-23 $ cd nest1
 → ~/desktop/cse/ASSGN/sem5/os/22-08-23/nest1 $ ls
nest2
 → ~/desktop/cse/ASSGN/sem5/os/22-08-23/nest1 $ cd nest2
 → ~/desktop/cse/ASSGN/sem5/os/22-08-23/nest1/nest2 $ ls
 → ~/desktop/cse/ASSGN/sem5/os/22-08-23/nest1/nest2 $ cd ..; cd ..
 → ~/desktop/cse/ASSGN/sem5/os/22-08-23 $ clang q3.c -o q3
 → ~/desktop/cse/ASSGN/sem5/os/22-08-23 $ ./q3 nest1
Directory 'nest1' and its subfolders have been deleted.
 → ~/desktop/cse/ASSGN/sem5/os/22-08-23 $ ls
file1.txt q1        q2        q3        q4.sh
file2.txt q1.c      q2.c      q3.c      q5.sh
 → ~/desktop/cse/ASSGN/sem5/os/22-08-23 $
```

```bash
#!/bin/bash

#Q4
#Write a menu driven shell script for the following
options:
#i. Merging the contents of two files into another
#ii. Searching a pattern from a file
#If the user gives some invalid choice, it should
prompt “Invalid option” message.

while true; do
    echo "Enter : "
    echo "1. For Merging two files"
    echo "2. For Searching pattern from file"
    echo "3. For Exit"
    read -p "Enter your choice: " choice

    case $choice in
    1)
        read -p "Enter 1st filename : " file1
        read -p "Enter 2nd filename : " file2
        read -p "Enter merged filename : " merged_file
        if [[ -f "$file1" ]] && [[ -f "$file2" ]]; then
            cat "$file1" "$file2" >"$merged_file"
            echo "$file1 & $file2 merged into
$merged_file"
            echo
        else
            echo "File 1 OR 2 not found!"
            echo
        fi
        ;;
    2)
        read -p "Enter the filename : " file
        read -p "Enter pattern to search : " ptrn
        if [[ -f "$file" ]]; then
            if grep "$ptrn" "$file"; then
                echo "$ptrn found in $file"
                echo
            else
                echo "$ptrn NOT FOUND!"
                echo
            fi
        else
            echo "File not found!"
```

```
            echo
        fi
        ;;
    3)
        echo "Exiting program"
        echo
        exit 0
        ;;
    *)
        echo "INVALID CHOICE!"
        echo
        ;;
    esac
done
```

```
.../sem5/os/22-08-23
 → ~/desktop/cse/ASSGN/sem5/os/22-08-23 $ cat file1.txt
hello world

 → ~/desktop/cse/ASSGN/sem5/os/22-08-23 $ cat file2.txt
test file 2
 → ~/desktop/cse/ASSGN/sem5/os/22-08-23 $ ./q4.sh
Enter :
1. For Merging two files
2. For Searching pattern from file
3. For Exit
Enter your choice: 1
Enter 1st filename : file1.txt
Enter 2nd filename : file2.txt
Enter merged filename : m.txt
file1.txt & file2.txt merged into m.txt

Enter :
1. For Merging two files
2. For Searching pattern from file
3. For Exit
Enter your choice: 2
Enter the filename : q4.sh
Enter pattern to search : read
    read -p "Enter your choice: " choice
        read -p "Enter 1st filename : " file1
        read -p "Enter 2nd filename : " file2
        read -p "Enter merged filename : " merged_file
        read -p "Enter the filename : " file
        read -p "Enter pattern to search : " ptrn
read found in q4.sh

Enter :
1. For Merging two files
2. For Searching pattern from file
3. For Exit
Enter your choice: e
INVALID CHOICE!

Enter :
1. For Merging two files
2. For Searching pattern from file
3. For Exit
Enter your choice: 3
Exiting program

 → ~/desktop/cse/ASSGN/sem5/os/22-08-23 $ cat m.txt
hello world

test file 2
```

```bash
#!/bin/bash

#Write a menu driven shell script for the following
options:
#i. Number of presently active users
#ii. Displaying some desired number of lines from top
of a file
#iii. Updating the access time of a given file to
current time
#If the user gives some invalid choice, it should
prompt “Invalid option” message.

while true; do
    echo "Enter : "
    echo "1. For Number of Active users"
    echo "2. For displaying number of lines from top of
file"
    echo "3. For Updating access time of file to
current time"
    echo "4. For Exit"
    read -p "Enter your choice: " choice

    case $choice in
    1)
        active_users=$(who | wc -l)
        echo "Active users : $active_users"
        echo
        ;;
    2)
        read -p "Enter filename : " filename
        read -p "Enter number of lines to display : "
num
        if [ -f "$filename" ]; then
            echo "Top $num lines from $filename : "
            head -n $num "$filename"
            echo
        else
            echo "File not found!"
            echo
        fi
        ;;
    3)
        read -p "Enter filename : " filename
        if [ -f "$filename" ]; then
            touch "$filename"
```

```
            echo "Access time updated succesfully!"
            echo
        else
            echo "File not found!"
            echo
        fi
        ;;
    4)
        echo "Exiting program"
        echo
        exit 0
        ;;
    *)
        echo "INVALID CHOICE!"
        echo
        ;;
    esac
done
```

```
.../sem5/os/22-08-23
ShresthS         console      Jul 13 21:23
 → ~/desktop/cse/ASSGN/sem5/os/22-08-23 $ stat m.txt
16777230 52317912 -rw-r--r-- 1 ShresthS staff 0 25 "Aug 22 22:45:56 2023" "Aug
 22 22:45:54 2023" "Aug 22 22:45:54 2023" "Aug 22 22:42:32 2023" 4096 8 0x40 m
.txt
 → ~/desktop/cse/ASSGN/sem5/os/22-08-23 $ ./q5.sh
Enter :
1. For Number of Active users
2. For displaying number of lines from top of file
3. For Updating access time of file to current time
4. For Exit
Enter your choice: 1
Active users :        1

Enter :
1. For Number of Active users
2. For displaying number of lines from top of file
3. For Updating access time of file to current time
4. For Exit
Enter your choice: 2
Enter filename : q5.sh
Enter number of lines to display : 5
Top 5 lines from q5.sh :
#!/bin/bash

#Write a menu driven shell script for the following options:
#i. Number of presently active users
#ii. Displaying some desired number of lines from top of a file

Enter :
1. For Number of Active users
2. For displaying number of lines from top of file
3. For Updating access time of file to current time
4. For Exit
Enter your choice: 3
Enter filename : m.txt
Access time updated succesfully!

Enter :
1. For Number of Active users
2. For displaying number of lines from top of file
3. For Updating access time of file to current time
4. For Exit
Enter your choice: 4
Exiting program

 → ~/desktop/cse/ASSGN/sem5/os/22-08-23 $ stat m.txt
16777230 52317912 -rw-r--r-- 1 ShresthS staff 0 25 "Aug 22 22:50:41 2023" "Aug
 22 22:50:40 2023" "Aug 22 22:50:40 2023" "Aug 22 22:42:32 2023" 4096 8 0x40 m
```