ASSIGNMENT 1
08/08/23

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

```c
//Q2
/*
 * Write C programs to simulate UNIX commands like ls,
grep, etc.
 */

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

// argv[2] = pattern
// argv[1] = file

void grep(char **pattern, char **filename) {
    char temp[200];
    FILE *fp;
    fp = fopen(filename, "r");
    while (!feof(fp)) {
        fgets(temp, 1000, fp);
        if (strstr(temp, pattern))
            printf("%s", temp);
    }
    fclose(fp);
}

void ls(char **dirname) {
    DIR *p;
    struct dirent *d;
    p = opendir(dirname);
    if (p == NULL) {
        perror("Cannot find directory");
        exit(-1);
    }
    while (d = readdir(p))
        printf("%s\n", d->d_name);
}


int main(int argc, char **argv) {
    grep(argv[2], argv[1]);
    ls(argv[1]);
    return 0;
}
```

```c
//Q3
/*
 * Write a program to implement
 * 1. Create a file
 * 2. Read contents of a file
 * 3. Write to a file
 * 4. Link and unlink a file
 * 5. Copy file
 * 6. Read contents of a file in a reverse order
 * Using the system calls: open( ), close( ), read( ),
write( ), lseek( ), link( ), unlink( ).
 */

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

void createFile(char *filename) {
    int fd;
    if ((fd = creat(filename, S_IRUSR | S_IWUSR)) < 0)
        perror("create() error\n");
    else {
        close(fd);
        printf("File %s created successfully\n",
filename);
    }
}

void readFile(char *filename) {
    char str;
    int fd = open(filename, O_RDONLY);

    if (fd) {
        printf("\nFile contents : \n");
        while (read(fd, &str, 1) == 1) {
            printf("%c", str);
        }
    } else printf("File not found!\n");

    close(fd);
}
```

```c
void writeFile(char *filename) {
    int fd = open("example.txt", O_WRONLY | O_CREAT,
0644);
    if (fd == -1) {
        perror("File not found!\n");
        return 1;
    }

    char data[];
    printf("Enter data to be written\n");
    scanf("%*[^\n]%*c", &data);
    ssize_t bytes_written = write(fd, data,
sizeof(data) - 1);

    if (bytes_written == -1) {
        perror("Error writing to the file\n");
        close(fd);
        return 1;
    }
    close(fd);
    printf("Data has been written to the file.\n");
}

void LinkUnlinkFile(char *filename, char *lnk) {
    char* source_filename;
    char* link_filename;

    strcpy(source_filename, filename);
    strcpy(link_filename, lnk);

    if (linkAndUnlink(source_filename, link_filename) !
= 0) {
        return 1;
    }

    if (link(source, linkName) == -1) {
        perror("Error linking the files");
        return 1;
    }
    if (unlink(linkName) == -1) {
        perror("Error unlinking the file");
        return 1;
    }
```

```
    printf("Linking and unlinking completed
successfully.\n");
}

void copyFile(char *filename) {
    printf("Enter file to be copied : ");
    char src[];
    scanf("%*[^\n]%*c", &src);

    printf("Enter file to be pasted on : ");
    char dst[];
    scanf("%*[^\n]%*c", &dst);

    int source_fd = open(src, O_RDONLY);
    if (source_fd == -1) {
        perror("Error opening the source file\n");
        return 1;
    }

    int dest_fd = open(dst, O_WRONLY | O_CREAT |
O_TRUNC, 0644);
    if (dest_fd == -1) {
        perror("Error opening/creating the destination
file\n");
        close(source_fd);
        return 1;
    }

    char buffer[BUFFER_SIZE];
    ssize_t bytes_read, bytes_written;
    while ((bytes_read = read(source_fd, buffer,
BUFFER_SIZE)) > 0) {
        bytes_written = write(dest_fd, buffer,
bytes_read);

        if (bytes_written == -1) {
            perror("Error writing to the destination
file\n");
            close(source_fd);
            close(dest_fd);
            return 1;
        }
    }

    if (bytes_read == -1) {
```

```
        perror("Error reading the source file\n");
        close(source_fd);
        close(dest_fd);
        return 1;
    }

    close(source_fd);
    close(dest_fd);
    printf("File copied successfully.\n");
}

void readRevFile(char *filename) {
    int fd = open("example.txt", O_RDONLY);
    if (fd == -1) {
        perror("Error opening the file");
        return 1;
    }

    off_t file_size = lseek(fd, 0, SEEK_END);
    if (file_size == -1) {
        perror("Error getting file size");
        close(fd);
        return 1;
    }

    char buffer[BUFFER_SIZE];
    for (off_t offset = file_size - BUFFER_SIZE; offset
>= 0; offset -= BUFFER_SIZE) {
        ssize_t bytes_read;

        if (lseek(fd, offset, SEEK_SET) == -1) {
            perror("Error seeking file");
            close(fd);
            return 1;
        }

        bytes_read = read(fd, buffer, BUFFER_SIZE);
        if (bytes_read == -1) {
            perror("Error reading file");
            close(fd);
            return 1;
        }

        for (ssize_t i = bytes_read - 1; i >= 0; i--) {
            putchar(buffer[i]);
```

```
        }
    }

    close(fd);
}

int main(int argc, char **argv) {
    while (1) {
        printf("Press 0. EXIT\n"
               "Press 1. Create a file\n"
               "Press 2. Read contents of a file\n"
               "Press 3. Write to a file\n"
               "Press 4. Link and unlink a file\n"
               "Press 5. Copy file\n"
               "Press 6. Read contents of a file in a
reverse order\n");

        int ch;
        printf("Enter choice : ");
        scanf("%d", &ch);
        char filename[1024] = "hello.txt";
        char lnk[1024] = "test.txt";

        switch (ch) {
            case 0 :
                exit(ch);
                break;
            case 1 :
                printf("Enter filename to Create : ");
                scanf("%*[^\n]%*c", &filename);
                createFile(filename);
                break;
            case 2 :
                printf("Enter filename to Read : ");
                scanf("%*[^\n]%*c", &filename);
                readFile(filename);
                break;
            case 3 :
                printf("Enter filename to Write : ");
                scanf("%*[^\n]%*c", &filename);
                writeFile(filename);
                break;
            case 4 :
                printf("Enter filename to link and
unlink : ");
```

```
                scanf("%*[^\n]%*c", &filename);
                scanf("%*[^\n]%*c", &lnk);
                LinkUnlinkFile(filename, lnk);
                break;
            case 5 :
                printf("Enter filename to copy : ");
                scanf("%*[^\n]%*c", &filename);
                readFile(filename);
                break;
            case 6 :
                printf("Enter filename to read : ");
                scanf("%*[^\n]%*c", &filename);
                readFile(filename);
                break;
            default:
                printf("INVALID INPUT!\n");
                break;
        }
        printf("\n");
    }
    return 0;
}
```

```c
//Q4
/*
 * Determine the size of a file using the lseek
command. Once you found out the size, calculate the
number of blocks assigned for the file. Compare these
results with the similar results obtained when using
the function stat.
 */

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

void fileSizeLSEEK(char **filename) {
    int fd = open(filename, O_RDONLY);
    int size = lseek(fd, 0, SEEK_END);

    printf("LSEEK Size of %s = %d\n", filename, size);
    close(fd);
}

void fileSizeSTAT(char **filename) {
    struct stat stbuf;
    stat(filename, &stbuf);

    printf("STAT Size of %s = %d\n", filename,
stbuf.st_size);
    printf("BLOCK Size of %s = %d\n", filename,
stbuf.st_blocks);
    printf("BLOCK Size of %s = %d\n", filename,
stbuf.st_ino);
}

int main(int argc, char **argv) {
    fileSizeLSEEK(argv[1]);
    fileSizeSTAT(argv[1]);
    return 0;
}
```

```c
//Q5
/*
 * Write a program to change current working directory
and display the inode details for each file in the new
directory using the system calls: opendir( ),
readdir( ), closedir( ), getcwd( ), chdir( ).
 */

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

void printInodes(char **dirname) {
    DIR *dir;
    struct dirent *d;
    struct stat stbuf;
    char filename[1024];

    dir = opendir(dirname);
    if (dir == NULL) {
        perror("Cannot find directory");
        exit(-1);
    }

    while (d = readdir(dir)) {
        strcpy(filename, d->d_name);
        stat(filename, &stbuf);
        printf("Inode of %s : %d\n", filename,
stbuf.st_ino);
    }

    closedir(dir);
}

int main(int argc, char **argv) {
    char cwd[1024];
    if (getcwd(cwd, sizeof(cwd)) == NULL)
        perror("getcwd() error\n");
    else {
        chdir(cwd);
        printInodes(cwd);
    }
```

```
    return 0;
}
```

```
.../sem5/os/08-08-23
 ➜ ~/desktop/cse/ASSGN/sem5/os/08-08-23 $ clang q2.c -o q2 2> /dev/null
 ➜ ~/desktop/cse/ASSGN/sem5/os/08-08-23 $ ./q2 q2.c "argv"
// argv[2] = pattern
// argv[1] = file
int main(int argc, char **argv) {
    grep(argv[2], argv[1]);
//    ls(argv[1]);
 ➜ ~/desktop/cse/ASSGN/sem5/os/08-08-23 $ cd test; ls; cd ..;
f1.txt f2.txt f3.txt
 ➜ ~/desktop/cse/ASSGN/sem5/os/08-08-23 $ clang q2.c -o q2 2> /dev/null
 ➜ ~/desktop/cse/ASSGN/sem5/os/08-08-23 $ ./q2 test
.
..
f1.txt
f3.txt
f2.txt
 ➜ ~/desktop/cse/ASSGN/sem5/os/08-08-23 $
```

```
.../sem5/os/08-08-23
 ➜ ~/desktop/cse/ASSGN/sem5/os/08-08-23 $ clang q4.c -o q4 2> /dev/null
 ➜ ~/desktop/cse/ASSGN/sem5/os/08-08-23 $ ./q4 hello.txt
LSEEK Size of hello.txt = 32
STAT Size of hello.txt = 32
BLOCK Size of hello.txt = 8
BLOCK Size of hello.txt = 51517805
 ➜ ~/desktop/cse/ASSGN/sem5/os/08-08-23 $ clang q5.c -o q5 2> /dev/null
 ➜ ~/desktop/cse/ASSGN/sem5/os/08-08-23 $ ./q5
Inode of . : 51508948
Inode of .. : 51508434
Inode of q5.c : 51512736
Inode of .DS_Store : 51556694
Inode of test : 51508954
Inode of q2.c : 51508951
Inode of t : 51514091
Inode of q3.c : 51514701
Inode of q4.c : 51508953
Inode of hello.txt : 51517805
Inode of t.c : 51513967
Inode of q2 : 52322124
Inode of q5 : 52322355
Inode of q4 : 52322313
Inode of q3 : 51518988
```