****

**LAB EXERCISE 2**

**Implementation of System calls**

Name: Jayannthan P T

Dept: CSE ‘A’

Roll No.: 205001049

Implementing cp command in C using system calls

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <fcntl.h>

**int** main(**int** argc, **char** \*argv**[]**)

{

    if (argc > 4)

        printf("Too many arguements\n");

    else if (argc < 2)

        printf("Atleast Two arguements required\n");

    else

    {

        printf("Opening file1:\n");

**int** file\_descriptor1 = open(argv[1], O\_RDONLY);

        if (file\_descriptor1 == -1)

            printf("Source File does not exist\n");

        else

        {

**char** contents[100];

            printf("Reading file1:\n");

**int** re = read(file\_descriptor1, contents, 100);

            printf("Checking for i:\n");

            if (argc > 3 && strcmp(argv[3], "i") == 0)

            {

                printf("Found i:\n");

                printf("Creating file2:\n");

**int** file\_descriptor2 = creat(argv[2], S\_IRUSR | S\_IWUSR);

                if (file\_descriptor2 < 0)

                {

                    printf("!!!ERROR!!!\n");

                }

                else

                {

                    printf("copying into file2:\n");

**int** wr = write(file\_descriptor2, contents, sizeof(contents));

                    printf("Closing file2:\n");

                    close(file\_descriptor2);

                    printf("\nSuccessfully Copied\n");

                }

                close(file\_descriptor1);

            }

            else

            {

                printf("i not found:\n");

                printf("Checking file2:\n");

**int** file\_descriptor2 = open(argv[2], O\_WRONLY);

                if (!(file\_descriptor2 < 0))

                {

**char** ch;

                    printf("Overwrite %s file?(y/n) ", argv[2]);

                    scanf(" %c", &ch);

                    if (!(ch == 'y' || ch == 'Y'))

                    {

                        close(file\_descriptor2);

                        close(file\_descriptor1);

                    }

                    else

                    {

                        printf("Creating file2:\n");

**int** file\_descriptor2 = creat(argv[2], S\_IRUSR | S\_IWUSR);

                        if (file\_descriptor2 < 0)

                        {

                            printf("!!!ERROR!!!\n");

                        }

                        else

                        {

                            printf("writing into file2:\n");

**int** wr = write(file\_descriptor2, contents, sizeof(contents));

                            close(file\_descriptor2);

                            printf("\nSuccessfully Copied\n");

                        }

                        close(file\_descriptor1);

                    }

                }

                else

                {

                    printf("Creating file2:\n");

**int** file\_descriptor2 = creat(argv[2], S\_IRUSR | S\_IWUSR);

                    if (file\_descriptor2 < 0)

                    {

                        printf("!!!ERROR!!!\n");

                    }

                    else

                    {

                        printf("copying into file2:\n");

**int** wr = write(file\_descriptor2, contents, sizeof(contents));

                        printf("Closing file2:\n");

                        close(file\_descriptor2);

                        printf("\nSuccessfully Copied\n");

                    }

                    close(file\_descriptor1);

                }

            }

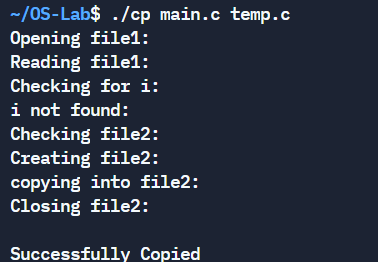
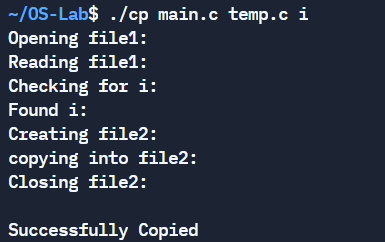
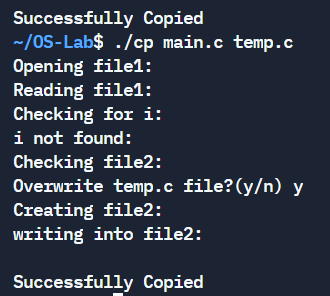
        }

    }

    return 0;

}

**Output:**

Implementing ls command in C using system calls

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <dirent.h>

#include <fcntl.h>

DIR \*dir, \*temp;

**struct** dirent \*tmp;

**void** recursive(**struct** dirent \*entry)

{

    if (entry == NULL)

    {

        return;

    }

    recursive(readdir(dir));

    printf("  %s\n", entry->d\_name);

}

*// void normal(struct dirent \*entry, int n)*

*// {*

*//     if (entry == NULL)*

*//     {*

*//         return;*

*//     }*

*//     for (int i = 0; i < n; i++)*

*//     {*

*//         printf("\t");*

*//     }*

*//     printf("  %s\n", entry->d\_name);*

*//     if (entry->d\_type == DT\_DIR && !(strcmp(entry->d\_name, ".") == 0 || strcmp(entry->d\_name, "..") == 0 || (entry->d\_name[0] == '.')))*

*//     {*

*//         temp = opendir(entry->d\_name);*

*//         tmp = readdir(temp);*

*//         normal(&tmp, n + 1);*

*//         closedir(temp);*

*//     }*

*//     normal(readdir(dir), n);*

*// }*

**int** main(**int** argc, **char** \*argv**[]**)

{

    if (argc > 4)

        printf("Too many arguements\n");

    else if (argc < 1)

        printf("Atleast one arguement required\n");

    else

    {

*// printf("%d\n",argc);*

        if (argc == 2)

        {

**struct** dirent \*entry;

            if ((dir = opendir(argv[1])) == NULL)

            {

                printf("CANNOT OPEN GIVEN DIRECTORY");

            }

            else

            {

                printf("Contents of the given:\n");

                while ((entry = readdir(dir)) != NULL)

                {

                    if (strcmp(entry->d\_name, ".") == 0 || strcmp(entry->d\_name, "..") == 0 || (entry->d\_name[0] == '.'))

                        continue;

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

                }

                closedir(dir);

            }

        }

        else if (argc > 1 && strcmp(argv[2], "r") == 0)

        {

            printf("ls -R\n");

**struct** dirent \*entry;

            if ((dir = opendir(argv[1])) == NULL)

            {

                printf("CANNOT OPEN GIVEN DIRECTORY");

            }

            else

            {

                printf("Contents of the given:\n");

                recursive(&entry);

                closedir(dir);

            }

        }

        else if (argc > 1 && strcmp(argv[2], "a") == 0)

        {

*// DIR \*dir;*

**struct** dirent \*entry;

            if ((dir = opendir(argv[1])) == NULL)

            {

                printf("CANNOT OPEN GIVEN DIRECTORY");

            }

            else

            {

                printf("Contents of the given:\n");

                while ((entry = readdir(dir)) != NULL)

                {

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

                }

                closedir(dir);

            }

        }

*// else if (argc > 1 && strcmp(argv[2], "R") == 0)*

*// {*

*//     // DIR \*dir;*

*//     struct dirent \*entry;*

*//     if ((dir = opendir(argv[1])) == NULL)*

*//     {*

*//         printf("CANNOT OPEN GIVEN DIRECTORY");*

*//     }*

*//     else*

*//     {*

*//         printf("Contents of the given:\n");*

*//         if ((entry = readdir(dir)) != NULL)*

*//         {*

*//             normal(&entry, 0);*

*//         }*

*//         closedir(dir);*

*//     }*

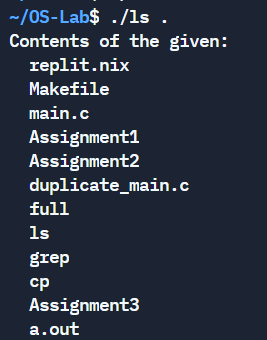
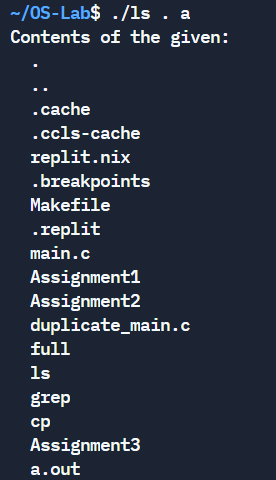
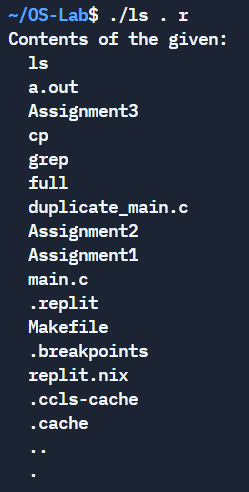
*// }*

    }

    return 0;

}

**Output:**

Implementing grep command in C using system calls

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

#include <dirent.h>

#include <fcntl.h>

**int** main(**int** argc, **char** \*argv**[]**)

{

    if (argc > 4)

        printf("Too many arguements\n");

    else if (argc < 2)

        printf("Two arguements required\n");

    else

    {

**int** file\_descriptor = open(argv[2], O\_RDONLY);

        if (file\_descriptor == -1)

            printf("File does not exist\n");

        else

        {

            if (argc == 3)

            {

*// printf("Normal Grep\n");*

**char** line[100], buf[1024];

**int** l = 0, i = 0, nr, count = 0;

                nr = read(file\_descriptor, buf, 1024);

                close(file\_descriptor);

                while (l < nr)

                {

                    for (i = 0; buf[l] != '\n' && l < nr; i++, l++)

                    {

                        line[i] = buf[l];

                    }

                    line[i] = '\0';

                    l++;

                    if (strstr(line, argv[1]))

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

                }

            }

            else if (argc > 3 && strcmp(argv[3], "c") == 0)

            {

*// printf("Grep -c\n");*

**char** line[100], buf[1024];

**int** l = 0, i = 0, nr, count = 0;

                nr = read(file\_descriptor, buf, 1024);

                close(file\_descriptor);

                while (l < nr)

                {

                    for (i = 0; buf[l] != '\n' && l < nr; i++, l++)

                    {

                        line[i] = buf[l]; *// extracting lines*

                    }

                    line[i] = '\0';

                    l++;

                    if (strstr(line, argv[1]))

                    {

                        count++;

                    }

                }

                printf("%d\n", count);

            }

        }

    }

    return 0;

}

**Output:**

