LAB 5 : Linux Command Line Utilities Simulator in C Programming

Nirajan Bekoju
PUL076BCT039
2^{nd} March, 2023

I. Introduction

A Linux Command Line Utilities Simulator in C Programming is a collection of programs that simulate the behavior of various Linux commands in a command line environment. The purpose of this simulator is to provide a way for users to learn and practice using Linux commands without having to install Linux or access a Linux terminal.
Each program in the simulator is written in the C programming language and emulates the functionality of a specific Linux command. For example, one program may simulate the behavior of the ”ls” command, while another program may simulate the behavior of the ”cd” command.
The programs in the simulator typically use standard input and output to communicate with the user, allowing the user to input command line arguments and see the output of the simulated command. The programs may also use system calls and libraries to access the file system and perform other operations required by the simulated command.
The Linux Command Line Utilities Simulator in C Programming can be a useful tool for individuals who are new to Linux and want to learn how to use Linux commands, as well as for experienced Linux users who want to test out different command options or troubleshoot issues in a safe and controlled environment. It can also be used by educators as a teaching tool in computer science or information technology courses.

II. Implementation of ls Command

Aim : To write a C program to simulate the operation of “ls” command in Unix.
Algorithm :

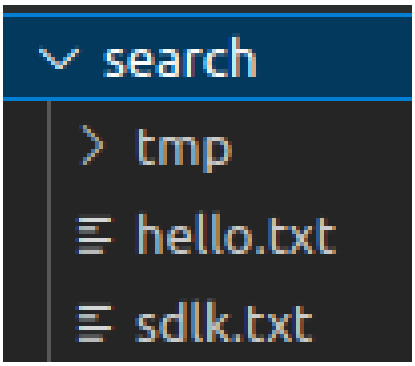
STEP 1 : Check if the number of command line arguments is less than 2. If yes, Print error and exit.
STEP 2 : Check if the second argument (i.e. directory to be listed) is valid or not. If not then exit.
STEP 3 : Print the content of the directory
STEP 4 : Close the directory entry file.

Program :

```
#include <stdio.h>
#include <sys/types.h>
#include <dirent.h>

// struct dirent {
//     ino_t          d_ino;       /* Inode number */
//     off_t          d_off;       /* Not an offset; see below */
//     unsigned short d_reclen;    /* Length of this record */
//     unsigned char  d_type;      /* Type of file; not supported
//                                    by all filesystem types */
//     char           d_name[256]; /* Null-terminated filename */
// };

int main(int argc, char *argv[]){
    DIR *dp;

    struct dirent *dirp;

    if(argc < 2){
        printf("\n You have provided only 1 argument \n");
        return -1;
    }
    dp = opendir(argv[1]);
```

```
    if(dp == NULL){
        printf("\n Cannot open %s file!\n", argv[1]);
        return -1;
    }
    dirp = readdir(dp);
    while (dirp != NULL)
    {
        printf("%s\n", dirp->d_name);
        dirp = readdir(dp);
    }
    closedir(dp);
}
```

The folder structure for ”search” folder is as follow:



Ouput :

```
$ ./ls search

tmp
hello.txt
sdlk.txt
.
..
```

III. Conclusion

In this lab, we studied how we can simulate the Linux Command Line Utilities usinig C programming. And we also learn the basic of dirent in c programming.
In C programming language, the dirent.h header file provides a standard interface for accessing directory entries. The directory entry structure, struct dirent, is defined in this header file.
The struct dirent contains information about each directory entry, including the file name, the file type, and the file size. Here’s an example of the struct definition:

```
struct dirent {
  ino_t          d_ino;       /* inode number */
  off_t          d_off;       /* offset to the next dirent */
  unsigned short d_reclen;    /* length of this record */
  unsigned char  d_type;      /* type of file; not supported by
                                 all file system types */
  char           d_name[256]; /* filename */
};
```

The d_ino field is the inode number of the directory entry, which is a unique identifier for the file within the file system. The d_off field is the offset to the next directory entry in the directory stream. The d_reclen field is the length of the directory entry record, and the d_type field indicates the type of the file, such as a regular file, a directory, a symbolic link, or other file types. The d_name field is a character array that contains the name of the directory entry. It is limited to 256 characters to ensure compatibility across different file systems.

The dirent.h header file provides a set of functions for working with directory entries, including opendir(), readdir(), and closedir(). These functions allow you to open a directory stream, read the directory entries, and close the directory stream when you are done.
Overall, dirent in C provides a simple and standardized way to work with directory entries in a cross-platform manner.