Name : Dev Adnani  
SID : 202212012

Subject : Systems Programming

Assignment : 5

Problem 1: Write a program (StudentID\_Lab5\_1.c) to implement following 3 commands as different functions

* cat <filename> - it takes parameters as file name including path and displays the content.
* ls <directory> – it takes a parameter as directory including path and lists all the files in that directory (single dot “.” Indicates current directory and double dots “..” indicates parent directory
* mv <source> <destination> – it takes 2 parameters: source and destination. source is the filename including path which you need to move, destination is either a directory or a new filename including path. If destination is a directory files is moved to this directory but if destination is filename then you move the file with a new name (i.e. move and rename operations combined), for example
  + mv ~/mydir/abc.txt ~/mydir/def.txt → only rename file abc.txt to def.txt in the same directory ~/mydir (~ indicates home folder) but don’t perform move.
  + mv ~/mydir/abc.txt ~/mydir\_new/def.txt → rename abc.txt to def.txt but also move the file from ~/mydir to ~/mydir\_new
  + mv ~/mydir/abc.txt ~/mydir\_new/ → only move the file abc.txt from ~/mydir to ~/mydir\_new but don’t rename
* Test each of the functions by calling them from the main() function.

Code :

#include <stdio.h>

#include <stdlib.h>

#include <fcntl.h>

#include <dirent.h>

#include <libgen.h>

#include <string.h>

#include <unistd.h>

void cat(char \*file)

{

if (strlen(file) <= 1)

{

printf("\nEnter File name : ");

return; }

char buffer[500];

int filename = open(file, O\_RDONLY); int size;

if (filename == -1)

{

printf("\nError while opening file...");

return; }

while ((size = read(filename, buffer, 500)) > 0) {

buffer[size] = '\0';

printf("%s", buffer); }

close(filename);

}

void ls(char \*dname)

{

DIR \*d;

struct dirent \*dir;

d = opendir(dname);

if (d == NULL)

{

printf("\nDirectory not found...");

return; }

while ((dir = readdir(d)))

{

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

} }

void mv(char \*src, char \*des) {

int size = strlen(des);

if (des[size - 1] == '/')

{

char \*filename = basename(src);

strcat(des, filename); }

if (rename(src, des) == -1)

{

printf("\nCan't move...");

} }

int main() {

char file[100];

char src[100];

char des[100];

int choice;

while (1)

{

printf("\nEnter 1: cat"); printf("\nEnter 2: ls"); printf("\nEnter 3: mv"); printf("\nEnter 4: Exit"); printf("\nEnter your choice : "); scanf("%d", &choice);

switch (choice)

{

case 1:

printf("Enter file path : "); scanf("%s", file); printf("\n");

cat(file);

break;

case 2:

printf("Enter directory path : "); scanf("%s", file);

printf("\n");

ls(file);

break;

case 3:

printf("Enter Source file : "); scanf("%s", src);

printf("Enter Destination Path/file : "); scanf("%s", des);

printf("\n");

mv(src, des);

break;

case 4:

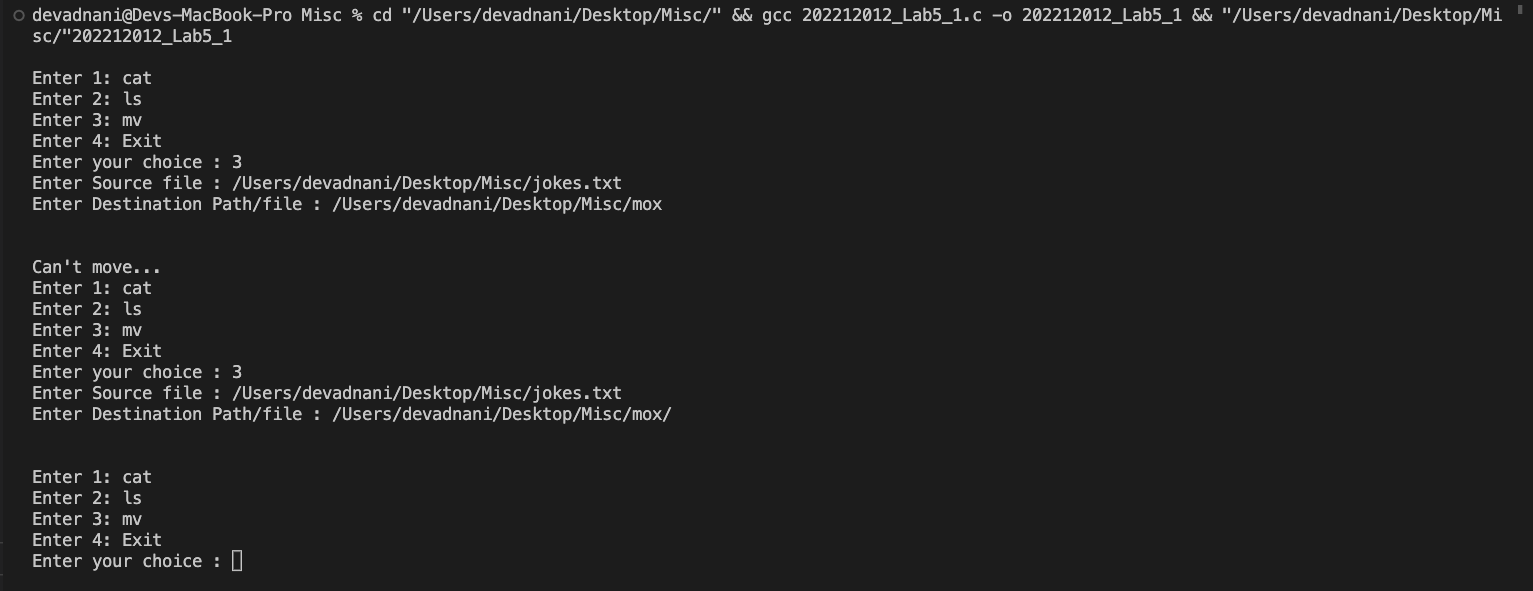
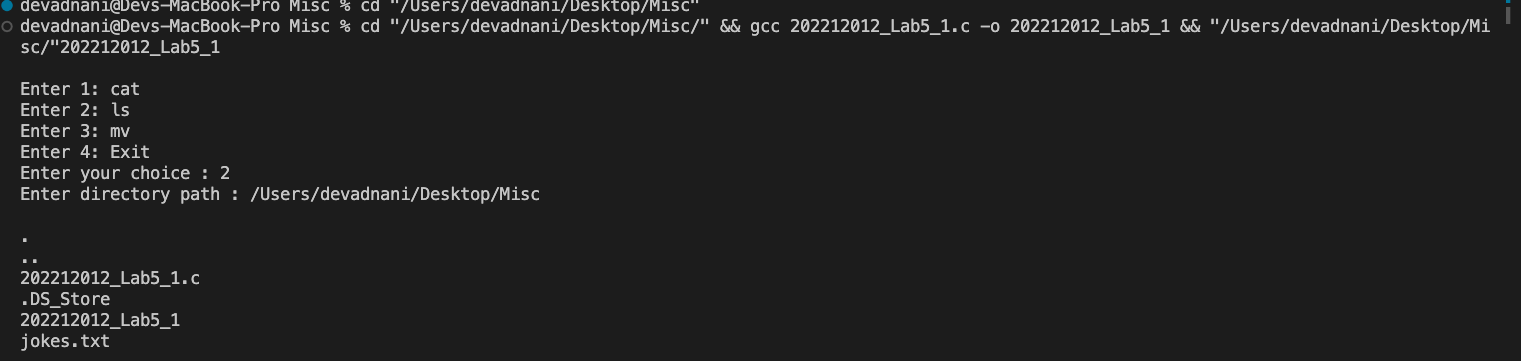
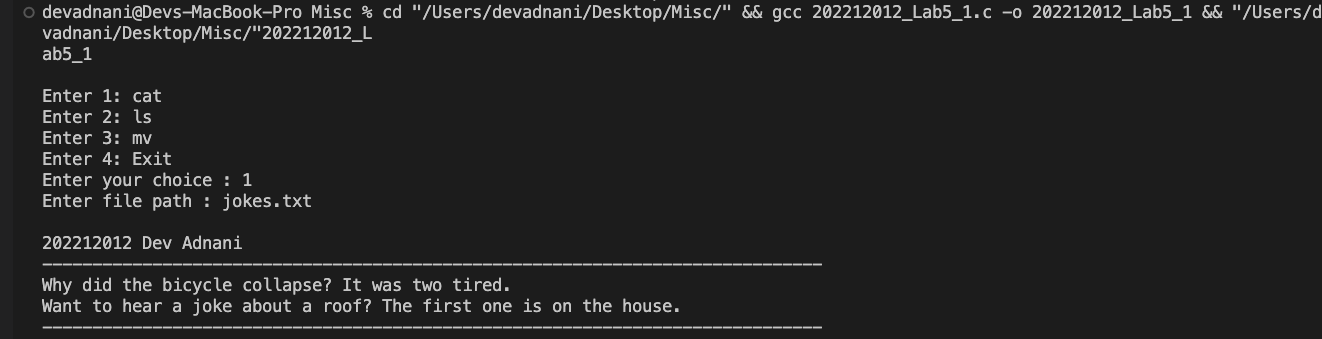
exit(0);

default:

printf("Invalid choice......"); break;

} }

}  
  
Output :



Problem 2: Implement your own shell in StudentID\_Lab5\_2.c

* Give Prompt myshell@<user>@<hostname>$ for user to type in command.
* E.g. user who has logged in is “student” on hostname “lab004\_15” then the prompt should be displayed as myshell@student@lab004\_15$
* Prompt waits for user to enter the command. It will accept commands listed below. For the first 3 commands, you will be calling the functions implemented in Problem1. For the forth command exit, you should exit from your own shell and return back to Parent Shell prompt.
  + cat <filename>
  + ls <directory>
  + mv <source> <destination> o exit

#include <stdio.h>

#include <stdlib.h>

#include <fcntl.h>

#include <dirent.h>

#include <unistd.h>

#include <string.h>

#include <libgen.h>

void cat(char \*str)

{

FILE \*fp = fopen(str, "r");

if (fp == NULL)

printf("\nCan't open the file...");

else

{

char ch;

while ((ch = fgetc(fp)) != EOF)

{

printf("%c", ch);

}

fclose(fp);

}

}

void ls(char \*dname)

{

DIR \*d;

struct dirent \*dir;

d = opendir(dname);

if (d == NULL)

{

printf("\nDirctory can't be accessed....");

return;

}

while ((dir = readdir(d)))

{

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

}

}

void mv(char \*src, char \*des)

{

int size = strlen(des);

if (des[size - 1] == '/')

{

char \*file = basename(src);

strcat(des, file);

}

if (rename(src, des) == -1)

{

printf("Can't move...");

}

}

int main()

{

char \*user = "student", \*hostname = "lab004\_15", input[100];

while (1)

{

printf("\n@%s@%s $", user, hostname);

scanf(" %[^\n]s", input);

char \*cmd = strtok(input, " ");

char \*srcAdd = strtok(NULL, " ");

char \*destAdd = strtok(NULL, " ");

if (strcmp(cmd, "ls") == 0)

{

ls(srcAdd);

printf("\n");

}

else if (strcmp(cmd, "cat") == 0)

{

cat(srcAdd);

printf("\n");

}

else if (strcmp(cmd, "mv") == 0)

{

mv(srcAdd, destAdd);

printf("\n");

}

else if (strcmp(cmd, "exit") == 0)

{

exit(0);

}

else

{

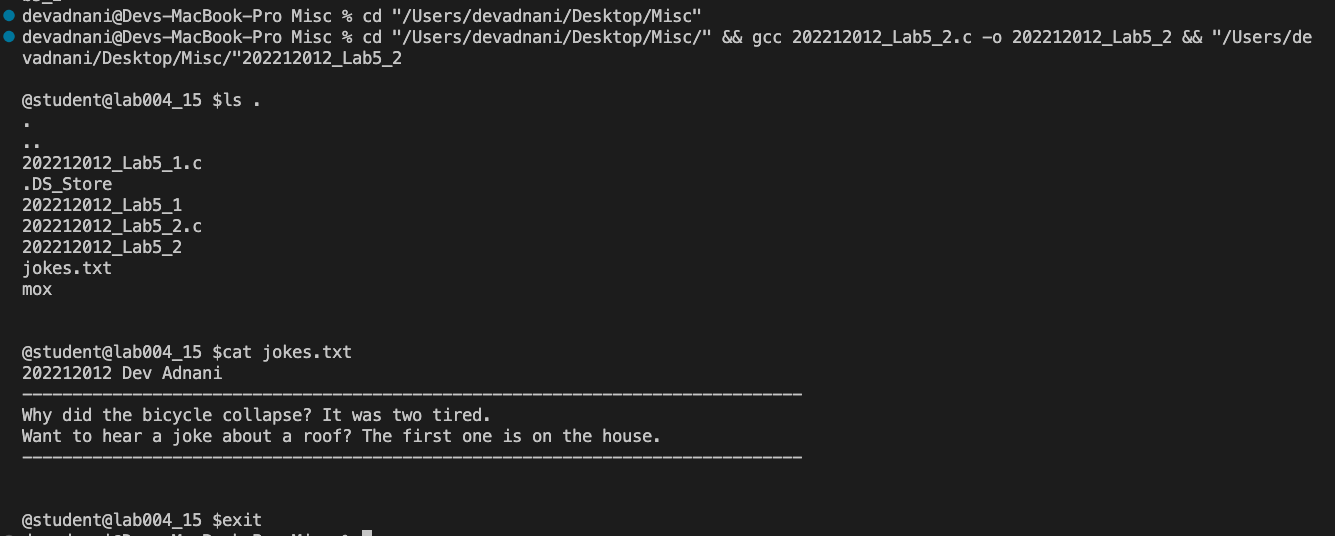
printf("\nYou have entered Wrong command !");

}

}

}

Output :



Problem 3: Write a program StudentID\_Lab5\_3.c to see the difference between hard links and symbolic links

* Using link() system call create hard link to a given text file as a command line argument
* Using symlink() system call create symbolic link to the same text file as above.
* Using ls command display the value of inodes for original file, hardlink and symbolic link and understand the output.

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

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

{

int hardLink = link(argv[1], "hard.txt");

printf("\nHardLink return value: %d", hardLink);

if (hardLink == 0)

{

printf("\nCongratulations, Hard Link has been created succuessfuly!\n");

}

else

{

printf("\nThere is an error while creating hard link\n\n ");

}

int softLink = symlink(argv[1], "soft.txt");

printf("SoftLink return value: %d", softLink);

if (softLink == 0)

{

printf("\ncongratulations, Soft Link has been created succuessfuly !\n");

}

else

{

printf("\nThere is an error while creating soft link\n\n");

}

return 0;

}

Screenshot :

