/\*

-------------------------------------------------------------------------------------------------------------------------------------

ATLAS 3.0 Phase III: Classroom Training

Author: Aman Raj (amarajz@amazon.com)

Team: IRONMAN

Module: 1 (Operating System & Sysytem Programming)

-------------------------------------------------------------------------------------------------------------------------------------\*/

/\*-------------------------MINI PROJECT: MINI LINUX SHELL INTERPRETER--------------------------------------------------------------\*/

/\* Library Header Files along with their descriptions\*/

#include<stdio.h> //standard input/output library functions

#include<fcntl.h> //manipulates file descriptor

#include<unistd.h> //standard symbolic constant and types

#include<string.h> //manipulates array of characters

#include<stdlib.h> //defines four variable types, several macros, and various functions for performing general functions

/\* User-defined function for CAT command\*/

void myCatfn(char fname[])

{

int fd;

fd=open(fname,O\_RDONLY); //Opens the file in read-only mode assigning file descriptor in variable "fd"

if(fd == -1) //fd = -1 when no file exists with that file name

printf("%s: No such file exists\n",fname);

else

{

char ch;

int k;

while(1) //Runs a continuous loop where the file is read character by character till k=0

{

k=read(fd,&ch,1);

if(k == 0) //Here integer variable 'k=1' when it reads a char but 'k=0' when no char is left to read

break;

else

printf("%c",ch);

}

printf("\n");

close(fd); //closes the file descriptor or we can say we are closing the file

}

}

/\* User-defined function for HEAD command\*/

void myHeadfn(char fname[])

{

int fd,cnt=1;

char ch;

fd=open(fname,O\_RDONLY); //Opens the file in read-only mode assigning file descriptor in variable "fd"

if(fd == -1) //fd = -1 when no file exists with that file name

printf("%s: No such file exists\n",fname);

else

{

while(1) //Runs a continuous loop

{

if(cnt<=10)

{

read(fd,&ch,1); //where the file is read character by character

printf("%c",ch);

if(ch=='\n')

cnt++; //and also counts the new line character till count=10 means 10 lines fo data from the top of the file

}

else

break;

}

printf("\n");

close(fd); ////closes the file descriptor or we can say we are closing the file

}

}

/\* User-defined function for TAIL command\*/

void myTailfn(char fname[])

{

int k,fd,cnt=0;

char ch;

fd=open(fname,O\_RDONLY); //Opens the file in read-only mode assigning file descriptor in variable "fd"

if(fd == -1) //fd = -1 when no file exists with that file name

printf("%s: No such file exists\n",fname);

else

{

lseek(fd,0,SEEK\_END); // this sets the cursor position to the last character of the file

while(cnt<10)

{ //since read shifts the cursor 1 char forward whenever it reads

lseek(fd,-2,SEEK\_CUR); //this lseek brings back 2 char backward till the new line character reaches the 10 line from the bottom of the file

read(fd,&ch,1);

if(ch == '\n')

cnt++;

}

while(1) //now the cursor is at 10th line from bottom of the file it reads it till the end

{

k=read(fd,&ch,1);

printf("%c",ch);

if(k == 0)

break;

}

printf("\n");

close(fd); //closes the file descriptor or we can say we are closing the file

}

}

/\* User-defined function for CP(COPY) command\*/

void myCpfn(char fname[], char dname[])

{

char ch;

int k,fd1,fd2;

fd1=open(fname,O\_RDONLY); //Open two files in read-only mode assigning file descriptor in variable "fd1" & "fd2"

if(fd1==-1) //fd1 = -1 when no file exists with that file name

{

printf("Source file not available, Cannot copy\n");

exit(0);

}

else

{

fd2=open(dname,O\_WRONLY|O\_CREAT,0666); //Opens the file in write-only mode and file not exists with that name it will create it assigning file descriptor in variable 'fd2'

while(1) //along with file permission read and write for user, group and others

{

k=read(fd1,&ch,1); //where the file is read character by character from file having 'fd1'

if(k==0)

break;

else

write(fd2,&ch,1); //where the file is written character by character from file having 'fd1' to file having 'fd2'

}

close(fd1); //closes the file descriptor or we can say we are closing both of the files

close(fd2);

}

}

/\* User-defined function for MV(MOVE) command\*/

void myMvfn(char fname[], char dname[])

{

char ch;

int k,fd1,fd2;

fd1=open(fname,O\_RDONLY); //Opens the file in read-only mode assigning file descriptor in variable "fd1"

if(fd1==-1) //fd1 = -1 when no file exists with that file name

{

printf("Source file not available, Cannot move/rename\n");

exit(0);

}

else

{

fd2=open(dname,O\_WRONLY|O\_CREAT,0666); //Opens the file in write-only mode and file not exists with that name it will create it assigning file descriptor in variable 'fd2'

while(1) //along with file permission read and write for user, group and others

{

k=read(fd1,&ch,1);

if(k==0)

break;

else

write(fd2,&ch,1); //where the file is written character by character from file having 'fd1' to file having 'fd2'

}

close(fd1); //closes the file descriptor or we can say we are closing both of the files

close(fd2);

unlink(fname); //removes the specified FILE.

}

}

/\* User-defined function for RM(REMOVE) command\*/

void myRmfn(char fname[])

{

int k,fd;

fd=open(fname,O\_RDONLY); //Opens the file in read-only mode assigning file descriptor in variable "fd"

if(fd == -1) //fd1 = -1 when no file exists with that file name

{

printf("No such file exists, Cannot delete\n");

exit(0); //exits

}

else

{

unlink(fname); //removes the specified FILE.

}

}

/\* User-defined function for reading HISTORY command\*/

void showfn(char fname[])

{

int k,fd,cnt=0;

char ch;

fd=open(fname,O\_RDONLY); //Opens the file in read-only mode assigning file descriptor in variable "fd"

if(fd == -1) //fd = -1 when no file exists with that file name

printf("%s: No such file exists\n",fname);

else

{

lseek(fd,0,SEEK\_END); // this sets the cursor position to the last character of the file

while(cnt<50)

{

lseek(fd,-2,SEEK\_CUR); //since read shifts the cursor 1 char forward whenever it reads

read(fd,&ch,1); //this lseek brings back 2 char backward till the new line character reaches the 50 line from the bottom of the file

if(ch == '\n')

cnt++;

}

while(1) //now the cursor is at 50th line from bottom of the file it reads it till the end

{

k=read(fd,&ch,1);

printf("%c",ch);

if(k == 0)

break;

}

printf("\n");

close(fd); //closes the file descriptor or we can say we are closing the file

}

}

/\* User-defined function for saving HISTORY of command to a file\*/

void history(int i,char cmd[], char fname[])

{

write(i,cmd,strlen(cmd)); //writes the cmd with string length of the cmd character array to history file

write(i," ",1); //writes a single space

write(i,fname,strlen(fname)); //writes the cmd with string length of the cmd character array to history file

write(i," ",1); //writes a single space

}

/\* Main function of the mini\_shell\*/

int main()

{

int i;

char cmd[20],fname[20]; //Variables for the command and the filename

system("clear"); //Clears the shell screen

while(1)

{

printf("code@Mini\_Shell$ ");

scanf("%s",cmd); //Reads the input by user the command and filename

scanf("%s",fname);

i=open("history",O\_WRONLY|O\_CREAT|O\_APPEND,0666); //Opens the file in write-only,creation and append mode assigning file descriptor in variable "i"

/\*From here it comapers the command using string compare and send it to the function call for different commands\*/

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

{

history(i,cmd,fname);

write(i,"\n",1);

myCatfn(fname); //CAT command function call

}

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

{

history(i,cmd,fname);

write(i,"\n",1);

myHeadfn(fname); //HEAD command function call

}

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

{

history(i,cmd,fname);

write(i,"\n",1);

myTailfn(fname); //TAIL command function call

}

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

{

char dname[20];

scanf("%s",dname); //Inputs the filename where we want to make the copy

history(i,cmd,fname);

write(i,dname,strlen(dname));

write(i,"\n",1);

myCpfn(fname,dname); //COPY command function call

}

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

{

char dname[20];

scanf("%s",dname); //Inputs the filename to which we want to rename

history(i,cmd,fname);

write(i,dname,strlen(dname));

write(i,"\n",1);

myMvfn(fname,dname); //MOVE command function call

}

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

{

history(i,cmd,fname);

write(i,"\n",1);

myRmfn(fname); //REMOVE command function call

}

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

{

history(i,cmd,fname);

write(i,"\n",1);

showfn(fname); //HISTORY command function call

}

else if((strcmp(cmd,"exit") == 0) && (strcmp(fname,"shell") == 0)) //Here it compares both the string and if matches to 'exit shell' , then it exits the mini\_shell

{

history(i,cmd,fname);

write(i,"\n",1);

break;

}

else

{

printf("Command \'%s\' not found\n",cmd); //If this mini\_shell does not have the command entered by user it gives error.

}

}

close(i); //closes the file descriptor or we can say we are closing the file where history of commands are saved.

}