CIS 415 Operating Systems

Assignment<1> Report Collection

Submitted to:

Prof. Allen Malony

Author:

<John (Ziyuan) Zhou>

Introduction:

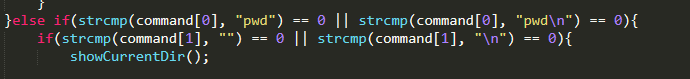
In this project, we basically make a pseudo-shell that able to execute some UNIX system command. This shell actually has two mode to execute, one is file mode and other one interactive mode. The difference between these two modes is one is able to read and execute the command in the input file and write the output in the output file (file mode) and other one simply types the command and print out the result in the console (interactive mode).

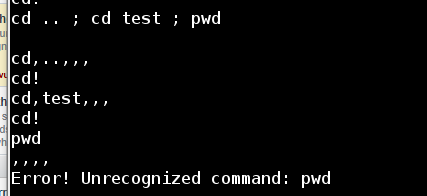
Background:

For this project, I actually used method strtok () and strtok\_r () a lot. In the lab, strtok () be introduced by TA and significantly help me to split the commands in the format I prefer. However, this method actually has some restriction if I want to split the string in different format many times. For strtok (), the string in the buffer will be split base on what I need, but I will have the problem if I want to split the string which I received from strtok (). The reason for that is strtok save a static pointer for reuse in the next time, but strtok\_r doesn’t. In the easy way to say it, I can’t parse 2 strings in parallel if I use strtok. Strtok\_r is a reentrant version of strtok. The non-reentrant versions often use global state, so if I call them from multiple threads, I might will invoke undefined behavior.

Implementation:

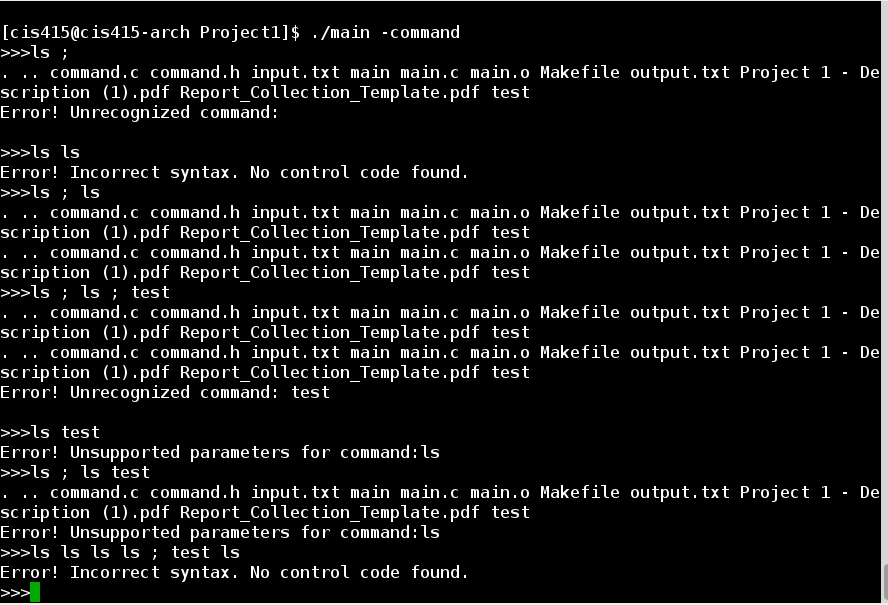
For the implementation of this project, I actually had some trouble with the file mode, and I am still working on that. The structures of my file mode and interactive mode are similar. All the commands received from either console or file split by semicolon and space step by step, which means I can read each command and then extract the cmd and parameters in order to check they are valid or not. If they are valid, I can input the variable to the function and return the right result. Everything is able to execute in my interactive mode, but I can’t read my command correctly in my file mode for some reasons. When I pass my command[0] which is pwd\n to check is it meet the requirement, something occurred and cause the condition didn’t meet and finally return as unrecognized command. The output shows in the console is pwd\n, and this unmatching should not happened.

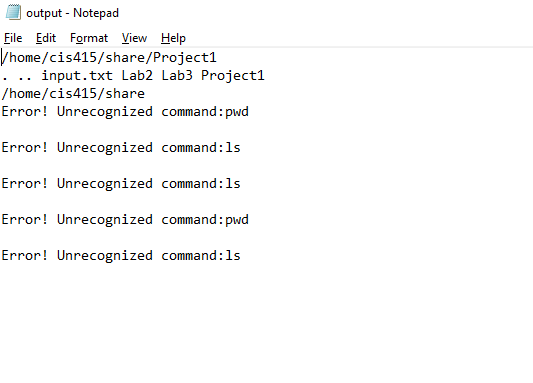




Performance Result and Discussion:

Like what I said in the pervious section, my interactive mode work well but file mode doesn’t because of input reading.





Conclusion:

After doing this project, like professor mentioned “let your C skill back online”, I feel familiar to the C currently, even learned something new. For instance, I actually don’t know the reentrant version of strtok and the restriction of this method before I do this project. It is interesting that how C different than java or other languages handles all the data it received. I never consider about the problem of buffer when I use something equivalent to strtok in other programming languages. This project let me feel that C is a very detailed and precise language, every small problem may cause my program crashed and I have to be cautious when I dealing with the memory.

Code:

/\*

\* Project 1 main.c

\*

\* Author: <John Zhou>

\*

\* Date: 10/20/2019

\*

\* Notes: <write anything you would like us to keep in mind when grading.>

\* Proper code documentation includes:

\* 1. Using a readable formatting.

\* 2. Adequate use of comments and white space.

\* 3. Explanation attached to code blocks that do not work.

\* 4. Keep the width of the program under 80 columns to avoid overwrap.

\*/

/\*-------------------------Preprocessor Directives---------------------------\*/

#define \_GNU\_SOURCE

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "command.h"

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

/\*-----------------------------Program Main----------------------------------\*/

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

/\* Variables \*/

FILE \*stream;

FILE \*file;

setbuf(stdout, NULL);

char \*buffer;

size\_t bufsize = 30;

const char space[2] = " ";

const char semi[4] = ";";

char \*line;

char \*token;

char\* command[4];

int count;

char\* laste[1];

int stop = 1;

buffer = (char \*)malloc(bufsize\*sizeof(char));

if(strcmp(argv[1],"-command") == 0){

while(stop){

printf(">>>");

getline(&buffer,&bufsize,stdin);

// buffer = strtok(buffer, "\n");

line = strtok(buffer, semi);

while(line != NULL){

count = 0;

for(int i = 0; i<4; i++){

command[i] = "";

}

while(token = strtok\_r(line,space,&line)){

command[count] = token;

count ++;

}

if(strcmp(command[0], "\n") == 0){

printf("Error! Unrecognized command: %s\n", command[0]);

break;

}else if(strcmp(command[0], "ls") == 0 || strcmp(command[0], "ls\n") == 0){

if(strcmp(command[1], "") == 0 || strcmp(command[1], "\n") == 0){

listDir();

}else if(strcmp(command[1], "ls") == 0 || strcmp(command[1], "ls\n") == 0 ||

strcmp(command[1], "pwd") == 0 || strcmp(command[1], "pwd\n") == 0 ||

strcmp(command[1], "mkdir") == 0 || strcmp(command[1], "mkdir\n") == 0 ||

strcmp(command[1], "cd") == 0 || strcmp(command[1], "cd\n") == 0 ||

strcmp(command[1], "cp") == 0 || strcmp(command[1], "cp\n") == 0 ||

strcmp(command[1], "mv") == 0 || strcmp(command[1], "mv\n") == 0 ||

strcmp(command[1], "rm") == 0 || strcmp(command[1], "rm\n") == 0 ||

strcmp(command[1], "cat") == 0 || strcmp(command[1], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else{

printf("Error! Unsupported parameters for command:%s\n",command[0]);

break;

}

}else if(strcmp(command[0], "pwd") == 0 || strcmp(command[0], "pwd\n") == 0){

if(strcmp(command[1], "") == 0 || strcmp(command[1], "\n") == 0){

showCurrentDir();

}else if(strcmp(command[1], "ls") == 0 || strcmp(command[1], "ls\n") == 0 ||

strcmp(command[1], "pwd") == 0 || strcmp(command[1], "pwd\n") == 0 ||

strcmp(command[1], "mkdir") == 0 || strcmp(command[1], "mkdir\n") == 0 ||

strcmp(command[1], "cd") == 0 || strcmp(command[1], "cd\n") == 0 ||

strcmp(command[1], "cp") == 0 || strcmp(command[1], "cp\n") == 0 ||

strcmp(command[1], "mv") == 0 || strcmp(command[1], "mv\n") == 0 ||

strcmp(command[1], "rm") == 0 || strcmp(command[1], "rm\n") == 0 ||

strcmp(command[1], "cat") == 0 || strcmp(command[1], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else{

printf("Error! Unsupported parameters for command:%s\n",command[0]);

break;

}

}else if(strcmp(command[0], "exit") == 0 || strcmp(command[0], "exit\n") == 0){

stop = 0;

break;

}else if(strcmp(command[0], "mkdir") == 0 || strcmp(command[0], "mkdir\n") == 0){

if(strcmp(command[1], "") == 0 || strcmp(command[1], "\n") == 0){

printf("Error! Incorrect parameters. Missing parameters\n");

break;

}else if(strcmp(command[1], "ls") == 0 || strcmp(command[1], "ls\n") == 0 ||

strcmp(command[1], "pwd") == 0 || strcmp(command[1], "pwd\n") == 0 ||

strcmp(command[1], "mkdir") == 0 || strcmp(command[1], "mkdir\n") == 0 ||

strcmp(command[1], "cd") == 0 || strcmp(command[1], "cd\n") == 0 ||

strcmp(command[1], "cp") == 0 || strcmp(command[1], "cp\n") == 0 ||

strcmp(command[1], "mv") == 0 || strcmp(command[1], "mv\n") == 0 ||

strcmp(command[1], "rm") == 0 || strcmp(command[1], "rm\n") == 0 ||

strcmp(command[1], "cat") == 0 || strcmp(command[1], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else {

if(strcmp(command[2], "") == 0 || strcmp(command[2], "\n") == 0){

command[1] = strtok\_r(command[1], "\n", &command[1]);

makeDir(command[1]);

}else{

if(strcmp(command[2], "ls") == 0 || strcmp(command[2], "ls\n") == 0 ||

strcmp(command[2], "pwd") == 0 || strcmp(command[2], "pwd\n") == 0 ||

strcmp(command[2], "mkdir") == 0 || strcmp(command[2], "mkdir\n") == 0 ||

strcmp(command[2], "cd") == 0 || strcmp(command[2], "cd\n") == 0 ||

strcmp(command[2], "cp") == 0 || strcmp(command[2], "cp\n") == 0 ||

strcmp(command[2], "mv") == 0 || strcmp(command[2], "mv\n") == 0 ||

strcmp(command[2], "rm") == 0 || strcmp(command[2], "rm\n") == 0 ||

strcmp(command[2], "cat") == 0 || strcmp(command[2], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else{

printf("Error! Incorrect parameters. Too many parameters\n");

break;

}

}

}

}else if(strcmp(command[0], "cd") == 0 || strcmp(command[0], "cd\n") == 0){

if(strcmp(command[1], "") == 0 || strcmp(command[1], "\n") == 0){

printf("Error! Incorrect parameters. Missing parameters\n");

break;

}else if(strcmp(command[1], "ls") == 0 || strcmp(command[1], "ls\n") == 0 ||

strcmp(command[1], "pwd") == 0 || strcmp(command[1], "pwd\n") == 0 ||

strcmp(command[1], "mkdir") == 0 || strcmp(command[1], "mkdir\n") == 0 ||

strcmp(command[1], "cd") == 0 || strcmp(command[1], "cd\n") == 0 ||

strcmp(command[1], "cp") == 0 || strcmp(command[1], "cp\n") == 0 ||

strcmp(command[1], "mv") == 0 || strcmp(command[1], "mv\n") == 0 ||

strcmp(command[1], "rm") == 0 || strcmp(command[1], "rm\n") == 0 ||

strcmp(command[1], "cat") == 0 || strcmp(command[1], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else {

if(strcmp(command[2], "") == 0 || strcmp(command[2], "\n") == 0){

command[1] = strtok\_r(command[1], "\n", &command[1]);

changeDir(command[1]);

}else{

if(strcmp(command[2], "ls") == 0 || strcmp(command[2], "ls\n") == 0 ||

strcmp(command[2], "pwd") == 0 || strcmp(command[2], "pwd\n") == 0 ||

strcmp(command[2], "mkdir") == 0 || strcmp(command[2], "mkdir\n") == 0 ||

strcmp(command[2], "cd") == 0 || strcmp(command[2], "cd\n") == 0 ||

strcmp(command[2], "cp") == 0 || strcmp(command[2], "cp\n") == 0 ||

strcmp(command[2], "mv") == 0 || strcmp(command[2], "mv\n") == 0 ||

strcmp(command[2], "rm") == 0 || strcmp(command[2], "rm\n") == 0 ||

strcmp(command[2], "cat") == 0 || strcmp(command[2], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else{

printf("Error! Incorrect parameters. Too many parameters\n");

break;

}

}

}

}else if(strcmp(command[0], "rm") == 0 || strcmp(command[0], "rm\n") == 0){

if(strcmp(command[1], "") == 0){

printf("Error! Incorrect parameters. Missing parameters\n");

break;

}else if(strcmp(command[1], "ls") == 0 || strcmp(command[1], "ls\n") == 0 ||

strcmp(command[1], "pwd") == 0 || strcmp(command[1], "pwd\n") == 0 ||

strcmp(command[1], "mkdir") == 0 || strcmp(command[1], "mkdir\n") == 0 ||

strcmp(command[1], "cd") == 0 || strcmp(command[1], "cd\n") == 0 ||

strcmp(command[1], "cp") == 0 || strcmp(command[1], "cp\n") == 0 ||

strcmp(command[1], "mv") == 0 || strcmp(command[1], "mv\n") == 0 ||

strcmp(command[1], "rm") == 0 || strcmp(command[1], "rm\n") == 0 ||

strcmp(command[1], "cat") == 0 || strcmp(command[1], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else {

if(strcmp(command[2], "") == 0 || strcmp(command[2], "\n") == 0){

command[1] = strtok\_r(command[1],"\n",&command[1]);

deleteFile(command[1]);

}else{

if(strcmp(command[2], "ls") == 0 || strcmp(command[2], "ls\n") == 0 ||

strcmp(command[2], "pwd") == 0 || strcmp(command[2], "pwd\n") == 0 ||

strcmp(command[2], "mkdir") == 0 || strcmp(command[2], "mkdir\n") == 0 ||

strcmp(command[2], "cd") == 0 || strcmp(command[2], "cd\n") == 0 ||

strcmp(command[2], "cp") == 0 || strcmp(command[2], "cp\n") == 0 ||

strcmp(command[2], "mv") == 0 || strcmp(command[2], "mv\n") == 0 ||

strcmp(command[2], "rm") == 0 || strcmp(command[2], "rm\n") == 0 ||

strcmp(command[2], "cat") == 0 || strcmp(command[2], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else{

printf("Error! Incorrect parameters. Too many parameters\n");

break;

}

}

}

}else if(strcmp(command[0], "cat") == 0 || strcmp(command[0], "cat\n") == 0){

if(strcmp(command[1], "") == 0 || strcmp(command[1], "\n") == 0){

printf("Error! Incorrect parameters. Missing parameters\n");

break;

}else if(strcmp(command[1], "ls") == 0 || strcmp(command[1], "ls\n") == 0 ||

strcmp(command[1], "pwd") == 0 || strcmp(command[1], "pwd\n") == 0 ||

strcmp(command[1], "mkdir") == 0 || strcmp(command[1], "mkdir\n") == 0 ||

strcmp(command[1], "cd") == 0 || strcmp(command[1], "cd\n") == 0 ||

strcmp(command[1], "cp") == 0 || strcmp(command[1], "cp\n") == 0 ||

strcmp(command[1], "mv") == 0 || strcmp(command[1], "mv\n") == 0 ||

strcmp(command[1], "rm") == 0 || strcmp(command[1], "rm\n") == 0 ||

strcmp(command[1], "cat") == 0 || strcmp(command[1], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else {

if(strcmp(command[2], "") == 0 || strcmp(command[2], "\n") == 0){

command[1] = strtok\_r(command[1], "\n", &command[1]);

displayFile(command[1]);

}else{

if(strcmp(command[2], "ls") == 0 || strcmp(command[2], "ls\n") == 0 ||

strcmp(command[2], "pwd") == 0 || strcmp(command[2], "pwd\n") == 0 ||

strcmp(command[2], "mkdir") == 0 || strcmp(command[2], "mkdir\n") == 0 ||

strcmp(command[2], "cd") == 0 || strcmp(command[2], "cd\n") == 0 ||

strcmp(command[2], "cp") == 0 || strcmp(command[2], "cp\n") == 0 ||

strcmp(command[2], "mv") == 0 || strcmp(command[2], "mv\n") == 0 ||

strcmp(command[2], "rm") == 0 || strcmp(command[2], "rm\n") == 0 ||

strcmp(command[2], "cat") == 0 || strcmp(command[2], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else{

printf("Error! Incorrect parameters. Too many parameters\n");

break;

}

}

}

}else if(strcmp(command[0], "cp") == 0 || strcmp(command[0], "cp\n") == 0){

if(strcmp(command[1], "") == 0 || strcmp(command[1], "\n") == 0){

printf("Error! Incorrect parameters. Missing parameters\n");

break;

}else if(strcmp(command[1], "ls") == 0 || strcmp(command[1], "ls\n") == 0 ||

strcmp(command[1], "pwd") == 0 || strcmp(command[1], "pwd\n") == 0 ||

strcmp(command[1], "mkdir") == 0 || strcmp(command[1], "mkdir\n") == 0 ||

strcmp(command[1], "cd") == 0 || strcmp(command[1], "cd\n") == 0 ||

strcmp(command[1], "cp") == 0 || strcmp(command[1], "cp\n") == 0 ||

strcmp(command[1], "mv") == 0 || strcmp(command[1], "mv\n") == 0 ||

strcmp(command[1], "rm") == 0 || strcmp(command[1], "rm\n") == 0 ||

strcmp(command[1], "cat") == 0 || strcmp(command[1], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else{

if(strcmp(command[2], "") == 0 || strcmp(command[2], "\n") == 0){

printf("Error! Incorrect parameters. Missing parameters\n");

break;

}else if(strcmp(command[2], "ls") == 0 || strcmp(command[2], "ls\n") == 0 ||

strcmp(command[2], "pwd") == 0 || strcmp(command[2], "pwd\n") == 0 ||

strcmp(command[2], "mkdir") == 0 || strcmp(command[2], "mkdir\n") == 0 ||

strcmp(command[2], "cd") == 0 || strcmp(command[2], "cd\n") == 0 ||

strcmp(command[2], "cp") == 0 || strcmp(command[2], "cp\n") == 0 ||

strcmp(command[2], "mv") == 0 || strcmp(command[2], "mv\n") == 0 ||

strcmp(command[2], "rm") == 0 || strcmp(command[2], "rm\n") == 0 ||

strcmp(command[2], "cat") == 0 || strcmp(command[2], "cat\n") == 0){

printf("Error! Unsupported parameters for command:%s\n",command[0]);

break;

}else{

if(strcmp(command[3], "") == 0 || strcmp(command[3], "\n") == 0){

command[2] = strtok\_r(command[2], "\n", &command[2]);

copyFile(command[1],command[2]);

}else{

if(strcmp(command[3], "ls") == 0 || strcmp(command[3], "ls\n") == 0 ||

strcmp(command[3], "pwd") == 0 || strcmp(command[3], "pwd\n") == 0 ||

strcmp(command[3], "mkdir") == 0 || strcmp(command[3], "mkdir\n") == 0 ||

strcmp(command[3], "cd") == 0 || strcmp(command[3], "cd\n") == 0 ||

strcmp(command[3], "cp") == 0 || strcmp(command[3], "cp\n") == 0 ||

strcmp(command[3], "mv") == 0 || strcmp(command[3], "mv\n") == 0 ||

strcmp(command[3], "rm") == 0 || strcmp(command[3], "rm\n") == 0 ||

strcmp(command[3], "cat") == 0 || strcmp(command[3], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else{

printf("Error! Incorrect parameters. Too many parameters\n");

break;

}

}

}

}

}else if(strcmp(command[0], "mv") == 0 || strcmp(command[0], "mv\n") == 0){

if(strcmp(command[1], "") == 0 || strcmp(command[1], "\n") == 0){

printf("Error! Incorrect parameters. Missing parameters\n");

break;

}else if(strcmp(command[1], "ls") == 0 || strcmp(command[1], "ls\n") == 0 ||

strcmp(command[1], "pwd") == 0 || strcmp(command[1], "pwd\n") == 0 ||

strcmp(command[1], "mkdir") == 0 || strcmp(command[1], "mkdir\n") == 0 ||

strcmp(command[1], "cd") == 0 || strcmp(command[1], "cd\n") == 0 ||

strcmp(command[1], "cp") == 0 || strcmp(command[1], "cp\n") == 0 ||

strcmp(command[1], "mv") == 0 || strcmp(command[1], "mv\n") == 0 ||

strcmp(command[1], "rm") == 0 || strcmp(command[1], "rm\n") == 0 ||

strcmp(command[1], "cat") == 0 || strcmp(command[1], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else{

if(strcmp(command[2], "") == 0 || strcmp(command[2], "\n") == 0){

printf("Error! Incorrect parameters. Missing parameters\n");

break;

}else if(strcmp(command[2], "ls") == 0 || strcmp(command[2], "ls\n") == 0 ||

strcmp(command[2], "pwd") == 0 || strcmp(command[2], "pwd\n") == 0 ||

strcmp(command[2], "mkdir") == 0 || strcmp(command[2], "mkdir\n") == 0 ||

strcmp(command[2], "cd") == 0 || strcmp(command[2], "cd\n") == 0 ||

strcmp(command[2], "cp") == 0 || strcmp(command[2], "cp\n") == 0 ||

strcmp(command[2], "mv") == 0 || strcmp(command[2], "mv\n") == 0 ||

strcmp(command[2], "rm") == 0 || strcmp(command[2], "rm\n") == 0 ||

strcmp(command[2], "cat") == 0 || strcmp(command[2], "cat\n") == 0){

printf("Error! Unsupported parameters for command:%s\n",command[0]);

break;

}else{

if(strcmp(command[3], "") == 0 || strcmp(command[3], "\n") == 0){

command[2] = strtok\_r(command[2], "\n", &command[2]);

moveFile(command[1],command[2]);

}else{

if(strcmp(command[3], "ls") == 0 || strcmp(command[3], "ls\n") == 0 ||

strcmp(command[3], "pwd") == 0 || strcmp(command[3], "pwd\n") == 0 ||

strcmp(command[3], "mkdir") == 0 || strcmp(command[3], "mkdir\n") == 0 ||

strcmp(command[3], "cd") == 0 || strcmp(command[3], "cd\n") == 0 ||

strcmp(command[3], "cp") == 0 || strcmp(command[3], "cp\n") == 0 ||

strcmp(command[3], "mv") == 0 || strcmp(command[3], "mv\n") == 0 ||

strcmp(command[3], "rm") == 0 || strcmp(command[3], "rm\n") == 0 ||

strcmp(command[3], "cat") == 0 || strcmp(command[3], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else{

printf("Error! Incorrect parameters. Too many parameters\n");

break;

}

}

}

}

}else{

printf("Error! Unrecognized command: %s\n", command[0]);

break;

}

line = strtok(NULL, semi);

}

}

}

else if(strcmp(argv[1],"-f") == 0){

stream = fopen(argv[2],"r");

file = freopen("output.txt","w",stdout);

while(getline(&buffer,&bufsize,stream) != -1){

line = strtok(buffer, semi);

while(line != NULL){

count = 0;

for(int i = 0; i<4; i++){

command[i] = "";

}

while(token = strtok\_r(line,space,&line)){

command[count] = token;

count ++;

}

if(strcmp(command[0], "\n") == 0){

printf("Error! Unrecognized command: %s\n", command[0]);

break;

}else if(strcmp(command[0], "ls") == 0 || strcmp(command[0], "ls\n") == 0){

if(strcmp(command[1], "") == 0 || strcmp(command[1], "\n") == 0){

listDir();

}else if(strcmp(command[1], "ls") == 0 || strcmp(command[1], "ls\n") == 0 ||

strcmp(command[1], "pwd") == 0 || strcmp(command[1], "pwd\n") == 0 ||

strcmp(command[1], "mkdir") == 0 || strcmp(command[1], "mkdir\n") == 0 ||

strcmp(command[1], "cd") == 0 || strcmp(command[1], "cd\n") == 0 ||

strcmp(command[1], "cp") == 0 || strcmp(command[1], "cp\n") == 0 ||

strcmp(command[1], "mv") == 0 || strcmp(command[1], "mv\n") == 0 ||

strcmp(command[1], "rm") == 0 || strcmp(command[1], "rm\n") == 0 ||

strcmp(command[1], "cat") == 0 || strcmp(command[1], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else{

printf("Error! Unsupported parameters for command:%s\n",command[0]);

break;

}

}else if(strcmp(command[0], "pwd") == 0 || strcmp(command[0], "pwd\n") == 0){

if(strcmp(command[1], "") == 0 || strcmp(command[1], "\n") == 0){

showCurrentDir();

}else if(strcmp(command[1], "ls") == 0 || strcmp(command[1], "ls\n") == 0 ||

strcmp(command[1], "pwd") == 0 || strcmp(command[1], "pwd\n") == 0 ||

strcmp(command[1], "mkdir") == 0 || strcmp(command[1], "mkdir\n") == 0 ||

strcmp(command[1], "cd") == 0 || strcmp(command[1], "cd\n") == 0 ||

strcmp(command[1], "cp") == 0 || strcmp(command[1], "cp\n") == 0 ||

strcmp(command[1], "mv") == 0 || strcmp(command[1], "mv\n") == 0 ||

strcmp(command[1], "rm") == 0 || strcmp(command[1], "rm\n") == 0 ||

strcmp(command[1], "cat") == 0 || strcmp(command[1], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else{

printf("Error! Unsupported parameters for command:%s\n",command[0]);

break;

}

}else if(strcmp(command[0], "exit") == 0 || strcmp(command[0], "exit\n") == 0){

stop = 0;

break;

}else if(strcmp(command[0], "mkdir") == 0 || strcmp(command[0], "mkdir\n") == 0){

if(strcmp(command[1], "") == 0 || strcmp(command[1], "\n") == 0){

printf("Error! Incorrect parameters. Missing parameters\n");

break;

}else if(strcmp(command[1], "ls") == 0 || strcmp(command[1], "ls\n") == 0 ||

strcmp(command[1], "pwd") == 0 || strcmp(command[1], "pwd\n") == 0 ||

strcmp(command[1], "mkdir") == 0 || strcmp(command[1], "mkdir\n") == 0 ||

strcmp(command[1], "cd") == 0 || strcmp(command[1], "cd\n") == 0 ||

strcmp(command[1], "cp") == 0 || strcmp(command[1], "cp\n") == 0 ||

strcmp(command[1], "mv") == 0 || strcmp(command[1], "mv\n") == 0 ||

strcmp(command[1], "rm") == 0 || strcmp(command[1], "rm\n") == 0 ||

strcmp(command[1], "cat") == 0 || strcmp(command[1], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else {

if(strcmp(command[2], "") == 0 || strcmp(command[2], "\n") == 0){

command[1] = strtok\_r(command[1], "\n", &command[1]);

makeDir(command[1]);

}else{

if(strcmp(command[2], "ls") == 0 || strcmp(command[2], "ls\n") == 0 ||

strcmp(command[2], "pwd") == 0 || strcmp(command[2], "pwd\n") == 0 ||

strcmp(command[2], "mkdir") == 0 || strcmp(command[2], "mkdir\n") == 0 ||

strcmp(command[2], "cd") == 0 || strcmp(command[2], "cd\n") == 0 ||

strcmp(command[2], "cp") == 0 || strcmp(command[2], "cp\n") == 0 ||

strcmp(command[2], "mv") == 0 || strcmp(command[2], "mv\n") == 0 ||

strcmp(command[2], "rm") == 0 || strcmp(command[2], "rm\n") == 0 ||

strcmp(command[2], "cat") == 0 || strcmp(command[2], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else{

printf("Error! Incorrect parameters. Too many parameters\n");

break;

}

}

}

}else if(strcmp(command[0], "cd") == 0 || strcmp(command[0], "cd\n") == 0){

if(strcmp(command[1], "") == 0 || strcmp(command[1], "\n") == 0){

printf("Error! Incorrect parameters. Missing parameters\n");

break;

}else if(strcmp(command[1], "ls") == 0 || strcmp(command[1], "ls\n") == 0 ||

strcmp(command[1], "pwd") == 0 || strcmp(command[1], "pwd\n") == 0 ||

strcmp(command[1], "mkdir") == 0 || strcmp(command[1], "mkdir\n") == 0 ||

strcmp(command[1], "cd") == 0 || strcmp(command[1], "cd\n") == 0 ||

strcmp(command[1], "cp") == 0 || strcmp(command[1], "cp\n") == 0 ||

strcmp(command[1], "mv") == 0 || strcmp(command[1], "mv\n") == 0 ||

strcmp(command[1], "rm") == 0 || strcmp(command[1], "rm\n") == 0 ||

strcmp(command[1], "cat") == 0 || strcmp(command[1], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else {

if(strcmp(command[2], "") == 0 || strcmp(command[2], "\n") == 0){

command[1] = strtok\_r(command[1], "\n", &command[1]);

changeDir(command[1]);

}else{

if(strcmp(command[2], "ls") == 0 || strcmp(command[2], "ls\n") == 0 ||

strcmp(command[2], "pwd") == 0 || strcmp(command[2], "pwd\n") == 0 ||

strcmp(command[2], "mkdir") == 0 || strcmp(command[2], "mkdir\n") == 0 ||

strcmp(command[2], "cd") == 0 || strcmp(command[2], "cd\n") == 0 ||

strcmp(command[2], "cp") == 0 || strcmp(command[2], "cp\n") == 0 ||

strcmp(command[2], "mv") == 0 || strcmp(command[2], "mv\n") == 0 ||

strcmp(command[2], "rm") == 0 || strcmp(command[2], "rm\n") == 0 ||

strcmp(command[2], "cat") == 0 || strcmp(command[2], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else{

printf("Error! Incorrect parameters. Too many parameters\n");

break;

}

}

}

}else if(strcmp(command[0], "rm") == 0 || strcmp(command[0], "rm\n") == 0){

if(strcmp(command[1], "") == 0){

printf("Error! Incorrect parameters. Missing parameters\n");

break;

}else if(strcmp(command[1], "ls") == 0 || strcmp(command[1], "ls\n") == 0 ||

strcmp(command[1], "pwd") == 0 || strcmp(command[1], "pwd\n") == 0 ||

strcmp(command[1], "mkdir") == 0 || strcmp(command[1], "mkdir\n") == 0 ||

strcmp(command[1], "cd") == 0 || strcmp(command[1], "cd\n") == 0 ||

strcmp(command[1], "cp") == 0 || strcmp(command[1], "cp\n") == 0 ||

strcmp(command[1], "mv") == 0 || strcmp(command[1], "mv\n") == 0 ||

strcmp(command[1], "rm") == 0 || strcmp(command[1], "rm\n") == 0 ||

strcmp(command[1], "cat") == 0 || strcmp(command[1], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else {

if(strcmp(command[2], "") == 0 || strcmp(command[2], "\n") == 0){

command[1] = strtok\_r(command[1], "\n", &command[1]);

deleteFile(command[1]);

}else{

if(strcmp(command[2], "ls") == 0 || strcmp(command[2], "ls\n") == 0 ||

strcmp(command[2], "pwd") == 0 || strcmp(command[2], "pwd\n") == 0 ||

strcmp(command[2], "mkdir") == 0 || strcmp(command[2], "mkdir\n") == 0 ||

strcmp(command[2], "cd") == 0 || strcmp(command[2], "cd\n") == 0 ||

strcmp(command[2], "cp") == 0 || strcmp(command[2], "cp\n") == 0 ||

strcmp(command[2], "mv") == 0 || strcmp(command[2], "mv\n") == 0 ||

strcmp(command[2], "rm") == 0 || strcmp(command[2], "rm\n") == 0 ||

strcmp(command[2], "cat") == 0 || strcmp(command[2], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else{

printf("Error! Incorrect parameters. Too many parameters\n");

break;

}

}

}

}else if(strcmp(command[0], "cat") == 0 || strcmp(command[0], "cat\n") == 0){

if(strcmp(command[1], "") == 0 || strcmp(command[1], "\n") == 0){

printf("Error! Incorrect parameters. Missing parameters\n");

break;

}else if(strcmp(command[1], "ls") == 0 || strcmp(command[1], "ls\n") == 0 ||

strcmp(command[1], "pwd") == 0 || strcmp(command[1], "pwd\n") == 0 ||

strcmp(command[1], "mkdir") == 0 || strcmp(command[1], "mkdir\n") == 0 ||

strcmp(command[1], "cd") == 0 || strcmp(command[1], "cd\n") == 0 ||

strcmp(command[1], "cp") == 0 || strcmp(command[1], "cp\n") == 0 ||

strcmp(command[1], "mv") == 0 || strcmp(command[1], "mv\n") == 0 ||

strcmp(command[1], "rm") == 0 || strcmp(command[1], "rm\n") == 0 ||

strcmp(command[1], "cat") == 0 || strcmp(command[1], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else {

if(strcmp(command[2], "") == 0 || strcmp(command[2], "\n") == 0){

command[1] = strtok\_r(command[1], "\n", &command[1]);

displayFile(command[1]);

}else{

if(strcmp(command[2], "ls") == 0 || strcmp(command[2], "ls\n") == 0 ||

strcmp(command[2], "pwd") == 0 || strcmp(command[2], "pwd\n") == 0 ||

strcmp(command[2], "mkdir") == 0 || strcmp(command[2], "mkdir\n") == 0 ||

strcmp(command[2], "cd") == 0 || strcmp(command[2], "cd\n") == 0 ||

strcmp(command[2], "cp") == 0 || strcmp(command[2], "cp\n") == 0 ||

strcmp(command[2], "mv") == 0 || strcmp(command[2], "mv\n") == 0 ||

strcmp(command[2], "rm") == 0 || strcmp(command[2], "rm\n") == 0 ||

strcmp(command[2], "cat") == 0 || strcmp(command[2], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else{

printf("Error! Incorrect parameters. Too many parameters\n");

break;

}

}

}

}else if(strcmp(command[0], "cp") == 0 || strcmp(command[0], "cp\n") == 0){

if(strcmp(command[1], "") == 0 || strcmp(command[1], "\n") == 0){

printf("Error! Incorrect parameters. Missing parameters\n");

break;

}else if(strcmp(command[1], "ls") == 0 || strcmp(command[1], "ls\n") == 0 ||

strcmp(command[1], "pwd") == 0 || strcmp(command[1], "pwd\n") == 0 ||

strcmp(command[1], "mkdir") == 0 || strcmp(command[1], "mkdir\n") == 0 ||

strcmp(command[1], "cd") == 0 || strcmp(command[1], "cd\n") == 0 ||

strcmp(command[1], "cp") == 0 || strcmp(command[1], "cp\n") == 0 ||

strcmp(command[1], "mv") == 0 || strcmp(command[1], "mv\n") == 0 ||

strcmp(command[1], "rm") == 0 || strcmp(command[1], "rm\n") == 0 ||

strcmp(command[1], "cat") == 0 || strcmp(command[1], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else{

if(strcmp(command[2], "") == 0 || strcmp(command[2], "\n") == 0){

printf("Error! Incorrect parameters. Missing parameters\n");

break;

}else if(strcmp(command[2], "ls") == 0 || strcmp(command[2], "ls\n") == 0 ||

strcmp(command[2], "pwd") == 0 || strcmp(command[2], "pwd\n") == 0 ||

strcmp(command[2], "mkdir") == 0 || strcmp(command[2], "mkdir\n") == 0 ||

strcmp(command[2], "cd") == 0 || strcmp(command[2], "cd\n") == 0 ||

strcmp(command[2], "cp") == 0 || strcmp(command[2], "cp\n") == 0 ||

strcmp(command[2], "mv") == 0 || strcmp(command[2], "mv\n") == 0 ||

strcmp(command[2], "rm") == 0 || strcmp(command[2], "rm\n") == 0 ||

strcmp(command[2], "cat") == 0 || strcmp(command[2], "cat\n") == 0){

printf("Error! Unsupported parameters for command:%s\n",command[0]);

break;

}else{

if(strcmp(command[3], "") == 0 || strcmp(command[3], "\n") == 0){

command[2] = strtok\_r(command[2], "\n", &command[2]);

copyFile(command[1],command[2]);

}else{

if(strcmp(command[3], "ls") == 0 || strcmp(command[3], "ls\n") == 0 ||

strcmp(command[3], "pwd") == 0 || strcmp(command[3], "pwd\n") == 0 ||

strcmp(command[3], "mkdir") == 0 || strcmp(command[3], "mkdir\n") == 0 ||

strcmp(command[3], "cd") == 0 || strcmp(command[3], "cd\n") == 0 ||

strcmp(command[3], "cp") == 0 || strcmp(command[3], "cp\n") == 0 ||

strcmp(command[3], "mv") == 0 || strcmp(command[3], "mv\n") == 0 ||

strcmp(command[3], "rm") == 0 || strcmp(command[3], "rm\n") == 0 ||

strcmp(command[3], "cat") == 0 || strcmp(command[3], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else{

printf("Error! Incorrect parameters. Too many parameters\n");

break;

}

}

}

}

}else if(strcmp(command[0], "mv") == 0 || strcmp(command[0], "mv\n") == 0){

if(strcmp(command[1], "") == 0 || strcmp(command[1], "\n") == 0){

printf("Error! Incorrect parameters. Missing parameters\n");

break;

}else if(strcmp(command[1], "ls") == 0 || strcmp(command[1], "ls\n") == 0 ||

strcmp(command[1], "pwd") == 0 || strcmp(command[1], "pwd\n") == 0 ||

strcmp(command[1], "mkdir") == 0 || strcmp(command[1], "mkdir\n") == 0 ||

strcmp(command[1], "cd") == 0 || strcmp(command[1], "cd\n") == 0 ||

strcmp(command[1], "cp") == 0 || strcmp(command[1], "cp\n") == 0 ||

strcmp(command[1], "mv") == 0 || strcmp(command[1], "mv\n") == 0 ||

strcmp(command[1], "rm") == 0 || strcmp(command[1], "rm\n") == 0 ||

strcmp(command[1], "cat") == 0 || strcmp(command[1], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else{

if(strcmp(command[2], "") == 0 || strcmp(command[2], "\n") == 0){

printf("Error! Incorrect parameters. Missing parameters\n");

break;

}else if(strcmp(command[2], "ls") == 0 || strcmp(command[2], "ls\n") == 0 ||

strcmp(command[2], "pwd") == 0 || strcmp(command[2], "pwd\n") == 0 ||

strcmp(command[2], "mkdir") == 0 || strcmp(command[2], "mkdir\n") == 0 ||

strcmp(command[2], "cd") == 0 || strcmp(command[2], "cd\n") == 0 ||

strcmp(command[2], "cp") == 0 || strcmp(command[2], "cp\n") == 0 ||

strcmp(command[2], "mv") == 0 || strcmp(command[2], "mv\n") == 0 ||

strcmp(command[2], "rm") == 0 || strcmp(command[2], "rm\n") == 0 ||

strcmp(command[2], "cat") == 0 || strcmp(command[2], "cat\n") == 0){

printf("Error! Unsupported parameters for command:%s\n",command[0]);

break;

}else{

if(strcmp(command[3], "") == 0 || strcmp(command[3], "\n") == 0){

command[2] = strtok\_r(command[2], "\n", &command[2]);

moveFile(command[1],command[2]);

}else{

if(strcmp(command[3], "ls") == 0 || strcmp(command[3], "ls\n") == 0 ||

strcmp(command[3], "pwd") == 0 || strcmp(command[3], "pwd\n") == 0 ||

strcmp(command[3], "mkdir") == 0 || strcmp(command[3], "mkdir\n") == 0 ||

strcmp(command[3], "cd") == 0 || strcmp(command[3], "cd\n") == 0 ||

strcmp(command[3], "cp") == 0 || strcmp(command[3], "cp\n") == 0 ||

strcmp(command[3], "mv") == 0 || strcmp(command[3], "mv\n") == 0 ||

strcmp(command[3], "rm") == 0 || strcmp(command[3], "rm\n") == 0 ||

strcmp(command[3], "cat") == 0 || strcmp(command[3], "cat\n") == 0){

printf("Error! Incorrect syntax. No control code found.\n");

break;

}else{

printf("Error! Incorrect parameters. Too many parameters\n");

break;

}

}

}

}

}else{

printf("Error! Unrecognized command:%s\n", command[0]);

break;

}

line = strtok(NULL, semi);

}

}

fclose(stream);

}else{

printf("Error, unkown mode command\n");

}

free(buffer);

/\*

\* Project 1 command.c

\*

\* Author: <John Zhou>

\*

\* Date: 10/20/2019

\*

\* Notes: <write anything you would like us to keep in mind when grading.>

\* Proper code documentation includes:

\* 1. Using a readable formatting.

\* 2. Adequate use of comments and white space.

\* 3. Explanation attached to code blocks that do not work.

\* 4. Keep the width of the program under 80 columns to avoid overwrap.

\*/

#define \_GNU\_SOURCE

#include <stdio.h>

#include <stdlib.h>

#include <dirent.h>

#include <string.h>

#include <unistd.h>

#include <sys/stat.h>

#include <sys/types.h>

#include <fcntl.h>

#include "command.h"

void listDir(){

struct dirent \*entry;

char path[100];

DIR \*dir;

getcwd(path,sizeof(path));

dir = opendir(path);

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

write(1,entry->d\_name,strlen(entry->d\_name));

write(1," ",1);

}

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

closedir(dir);

} /\*for the ls command\*/

void showCurrentDir(){

char path[100];

if(getcwd(path,sizeof(path)) != NULL){

write(1, path,strlen(path));

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

}

} /\*for the pwd command\*/

void makeDir(char \*dirName){

mkdir(dirName, S\_IRWXU);

} /\*for the mkdir command\*/

void changeDir(char \*dirName){

chdir(dirName);

} /\*for the cd command\*/

void copyFile(char \*sourcePath, char \*destinationPath){

int destfile;

int sourfile;

int data;

char \*filename;

char currentdir[100];

char buffer[1024];

getcwd(currentdir,sizeof(currentdir));

sourfile = open(sourcePath, O\_RDONLY);

destfile = open(destinationPath, O\_WRONLY | O\_CREAT);

if(destfile == -1){

filename = strrchr(sourcePath, '/')+1;

chdir(destinationPath);

destfile = open(filename, O\_WRONLY | O\_CREAT);

}

while((data = read(sourfile,buffer,sizeof(buffer))) > 0){

destfile = write(destfile,buffer,data);

}

chdir(currentdir);

close(sourfile);

close(destfile);

} /\*for the cp command\*/

void moveFile(char \*sourcePath, char \*destinationPath){

int destfile;

int sourfile;

int data;

int index;

char buffer[1024];

char destination[100];

char currentdir[100];

char \*filename;

getcwd(currentdir,sizeof(currentdir));

sourfile = open(sourcePath, O\_RDONLY);

destfile = open(destinationPath, O\_WRONLY | O\_CREAT);

if(destfile == -1){

filename = strrchr(sourcePath, '/')+1;

chdir(destinationPath);

destfile = open(filename, O\_WRONLY | O\_CREAT);

}

while((data = read(sourfile,buffer,sizeof(buffer))) > 0){

destfile = write(destfile,buffer,data);

}

chdir(currentdir);

unlink(sourcePath);

close(sourfile);

close(destfile);

} /\*for the mv command\*/