*CIS 3207, Section 4*

*Tuyen Pham*

*TUID: 915591991*

**PROJECT 2A: PSEUDOCODE FOR IMPLEMENTING A LINUX SHELL**

1. **Pseudocode**

/\*\*\*LIBRARY SYSTEM CALLS\*\*\*/

#include<stdio.h>

#include<stdlib.h>

#include<string.h> // for strtok() and strcmp()

#include<unistd.h> //for fork() and pid\_t

#include<sys/wait.h> //for waitpid()

/\*\*\*STRUCT DECLARATIONS\*\*\*/

typdef struct Process{

int in; //where stdin is piped

int out; //where stdout is piped

int status; //execution status

char \*prog; //the program string

char\*\*argv; //argument for execution

} Process

//function to read a line from command into buffer

char \*readLine(){

store the dynamic memory allocation buffer

if buffer allocate fail, exit

while {

update character

if character is end of file or new line, return null

if access the buffer, reallocate it with more spaces or exit

}

}

//function to split a line into commands, which input can be tokenized by method strtok()

char \*\*splitLine(char \*line){

store the pointer token with the memory allocation

char \*token=strtok(line, delim);

int pos = 0; //update pos

while (token != NULL){

update token pos in array

increment pos

compare pos and bufsize

}

token=strtok(NULL, delim); //get next token

}

//Functions built-in Commands ( 8 commands)

char\*builtin\_str[ ]={“cd”, ”clr”, ”dir”, ”environ”, ”echo”, ”help”, “pause”, “quit”};

int(\*builtin\_func())(char\*\*)={&lsh\_cd, &lsh\_clr, &lsh\_dir, &lsh\_environ, &lsh\_echo, &lsh\_help, &lsh\_pause, &lsh\_quit};

//Case 1: build command “cd”

int lsh\_cd(char \*argv){ //if not argument, change to the top dir

char s[100];

print current working director

holds the name of new directory to switch to current

change to newDir

if false, do not exist

else print new working directory

}

//Case 2: build command “clr”

int lsh\_clr(char \*argv){ //print out the magical incantation

printf("\033[H\033[2J");

}

//Case 3: build command “dir”

int lsh\_dir(char \*argv){

pointer for directory entry

opendir() returns a pointer of DIR type

opendir return NULL if could not open directory

for readdir()

}

//Case 4: build command “environ”

int lsh\_environ(char \*argv){

getenv(“USER”);

}

//Case 5: build command “echo”

int lsh\_echo(char \*argv){

store the input

enter the input

scan the input

print the input

}

//Case 6: build command “help”

int lhs\_help(char \*argv){

read a file contains all commands

if success, print the file. else failure message

}

//Case 7: build command “pause”

int lhs\_pause(char \*argv){

getchar() and fflush(stdin)// wait a char input before exit

}

//Case 8: build command “quit”

int lhs\_quit(char \*argv){

return 0; //break the program

}

//function to create child process and run command

int myShell(char \*\*args){

pid\_t pid, wpid;

int status;

pid=fork(); //fork process

if pid =0, the child process

else pid<0, catch fork error

else the parent process

}

//function to execute command from terminal

int execShell (char \*\*args){

if (args[0]==NULL), empty command

loop to check for builtin functions

}

//main function

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

read from myShell configuration files;

parsing commands interactive mode script mode;

exit the shell;

}

1. **Possible errors**

* Pipe stored output failure
* Memory allocation default
* Segmentation false

1. **Test Plan**

This project will be able to handle running multiple commands. By interactive mode, my shell will display a prompt and the user will type in a command at the prompt such as “cd, clr, dir, encho, environ”. The shell will stop accepting the command when it sees the “quit” command. The shell should exit after all running processes have terminated.