Albert Wu

11573507

CptS 360

Lab 3 Code

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/wait.h>

#include <sys/stat.h>

#include <fcntl.h>

int executeCommand(char \*onecmd, int readfd, int writefd, char \*env[]) {

char \*myargvs[100];

char str[100][16]; //stores argvs

char cmdfile[64];

char \*token = strtok(onecmd, " ");

int i = 1;

while (token) {

myargvs[i] = str[i];

strcpy(myargvs[i], token);

token = strtok(NULL, " ");

i++;

}

myargvs[i] = NULL;

int fail = 0;

//check for sh file

int isELFFile = 1;

strcat(cmdfile, myargvs[1]);

FILE \*binfile = fopen(cmdfile, "r");

if (binfile) {

char buffer[4];

fread(buffer, 4, 1, binfile);

if(strcmp(buffer, "0x7F") != 0) { //sh file

strcpy(cmdfile, "/bin/bash");

strcpy(myargvs[0], "bash");

isELFFile = 0;

}

fclose(binfile);

}

if (isELFFile){

//search through PATH for ELF file

char \*actualpath = getenv("PATH");

char pathvar[strlen(actualpath)];

strcpy(pathvar, actualpath);

token = strtok(pathvar, ":");

while(token) {

cmdfile[0] = '\0';

strcpy(cmdfile, token);

strcat(cmdfile, "/");

strcat(cmdfile, myargvs[1]);

FILE \*file = fopen(cmdfile, "r");

if (file != NULL) {

// cmd file exists

fclose(file);

break;

}

token = strtok(NULL, ":");

}

if(!token) {

// elf file not found

printf("cmd not found\n");

return -1;

}

}

//Pipe redirection

if (readfd != -1) {

close(0);

dup(readfd);

}

if (writefd != -1) {

close(1);

dup(writefd);

}

//cmd IO redirection (input must be correct - will overwrite pipe redirection)

if (i > 3 && (strcmp(myargvs[i - 2], ">") == 0 || strcmp(myargvs[i - 2], ">>") == 0)) {

char \*ofiles = myargvs[i-1];

FILE \*outfile = fopen(ofiles, "w");

if (outfile) {

fclose(outfile);

close(1);

if (strcmp(myargvs[i - 2], ">") == 0) {

open(ofiles, O\_WRONLY|O\_CREAT, 0644);

} else {

open(ofiles, O\_WRONLY|O\_APPEND, 0644);

}

}

myargvs[i - 2] = NULL;

myargvs[i - 1] = NULL;

} else if(i > 3 && strcmp(myargvs[i - 2], "<<") == 0) {

char \*ifiles = myargvs[i-1];

FILE \*infile = fopen(ifiles, "w");

if (infile) {

fclose(infile);

close(0);

open(ifiles, O\_RDONLY);

}

myargvs[i - 2] = NULL;

myargvs[i - 1] = NULL;

}

//if is shfile, argsv will start 1 earlier to accomadate extra argument

exit(execve(cmdfile, myargvs + isELFFile, env));

}

int main(int c, char \*v[], char \*env[])

{

char line[128];

char \*myargvs[100];

char str[100][16]; //stores argvs

while(1){

printf("input a command line: ");

fgets(line,128,stdin);

line[strlen(line)-1] = 0;

char pipedcmds[100][100];

int p = 0;

char \*token = strtok(line, "|");

while (token) {

strcpy(pipedcmds[p], token);

p++;

token = strtok(NULL, "|");

}

pipedcmds[p][0] = '\0';

// check for simple commands

char firstcopy[100];

strcpy(firstcopy, \*pipedcmds);

token = strtok(firstcopy, " ");

int i = 0;

if(!token) {

printf("no command entered\n");

continue;

}

while (token) {

myargvs[i] = str[i];

strcpy(myargvs[i], token);

token = strtok(NULL, " ");

i++;

}

if(strcmp(myargvs[0], "cd") == 0) {

if (i > 1) {

chdir(myargvs[1]);

} else {

chdir(getenv("HOME"));

}

} else if (strcmp(myargvs[0], "exit") == 0) {

exit(0);

}

else { //executable command

int pid = fork();

if(pid) {

int status = 0;

int pid = wait(&status);

printf("command child completed, exit code: %d\n", status);

} else { // child

// iteratively spawn child for each command, linked by pipes

int fd[2];

int readfd = -1;

int nextreadfd = -1;

int writefd = -1;

int cp = 0;

while(1) {

readfd = nextreadfd;

if (pipedcmds[cp + 1][0] == '\0') {

writefd = -1;

} else {

int r = pipe(fd);

nextreadfd = fd[0];

writefd = fd[1];

}

int pid = fork();

if(!pid) {

executeCommand(pipedcmds[cp], readfd, writefd, env);

}

close(readfd);

close(writefd);

if (pipedcmds[cp + 1][0] == '\0') {

int status = 0;

while (wait(&status) > 0); // wait for all children

exit(0);

}

cp++;

}

}

}

}

}