**Problem Set 3 – The Shell**

1. **Source Code**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

#include <sys/stat.h>

#include <sys/resource.h>

#include <fcntl.h>

#include <errno.h>

#include <sys/time.h>

int ioRedirect(char\* path, int stream, int type){

int fd;

switch(type){

case 0:

fd = open(path, O\_RDONLY);

break;

case 1:

fd = open(path, O\_WRONLY|O\_CREAT|O\_TRUNC, 0666);

break;

case 2:

fd = open(path, O\_WRONLY|O\_CREAT|O\_APPEND, 0666);

break;

}

if(fd<0){

fprintf(stderr, "Failed to open file %s: %s\n", path, strerror(errno));

return -1;

}

if(dup2(fd,stream) < 0){

fprintf(stderr, "Failed to dup2 %s to stream %i: %s\n", path, stream, strerror(errno));

return -1;

}

close(fd);

return 0;

}

int processRedirect(char \*token, char \*\*fileRedirect, int \*stream, int \*type){

char \*c = token;

int i = STDIN\_FILENO;

if(\*c == '<'){

\*fileRedirect = &token[1];

\*stream = \*type = STDIN\_FILENO;

return 1;

} else if (\*c == '>'){

if(\*(++c) == '>'){

\*fileRedirect = &token[2];

\*stream = STDOUT\_FILENO;

\*type = STDERR\_FILENO;

} else {

\*fileRedirect = &token[1];

\*stream = \*type = STDOUT\_FILENO;

}

return 1;

} else if (\*c == '2'){

if(\*(++c) == '>'){

if(\*(++c) == '>'){

\*fileRedirect = &token[3];

\*stream = STDERR\_FILENO;

\*type = STDOUT\_FILENO;

} else {

\*fileRedirect = &token[2];

\*stream = \*type = STDERR\_FILENO;

}

return 1;

}

}

return 0;

}

void process(char\* line){

char \*token, \*command, \*arg1[1024], \*argr[1024];

char \*\*argi = arg1, \*\*argn = argr;

int status;

struct rusage ru;

struct timeval start, end, diff;

if((token = strtok(line, " \t\n")) == NULL) return;

\*argi++ = command = token;

printf("Executing command %s with arguments ", command);

while((token = strtok(NULL, " \t\n")) != NULL){

char \*fileRedirect;

int stream = -1, type = -1;

if(processRedirect(token, &fileRedirect, &stream, &type)){

\*argn++ = token;

continue;

} else {

\*argi++ = token;

printf("\"%s\" ",token);

}

}

\*argi++ = \*argn++ = NULL;

printf("\n");

if(strcmp(command, "cd") == 0){

if(chdir(arg1[1]) == -1){

fprintf(stderr, "Failed to change directory to %s: %s\n", arg1[1], strerror(errno));

return;

}

}else if(strcmp(command, "exit") == 0){

(arg1[1]!= NULL)?\_exit(atoi(arg1[1])):\_exit(EXIT\_SUCCESS);

}else{

pid\_t pid = fork();

switch(pid){

case -1:

fprintf(stderr, "Failed to fork on command %s: %s\n", command, strerror(errno));

exit(-1);

break;

case 0:

for(argn = argr; \*argn != NULL; \*argn++){

char \*fileRedirect;

int stream = -1, type = -1;

processRedirect(\*argn, &fileRedirect, &stream, &type);

ioRedirect(fileRedirect, stream, type);

}

execvp(command, arg1);

fprintf(stderr, "Execv returned on command %s: %s\n", command, strerror(errno));

exit(-1);

break;

default:

gettimeofday(&start, NULL);

if(wait3(&status, 0, &ru) == -1){

fprintf(stderr, "Wait3 failed for child process pid %d: %s\n", pid, strerror(errno));

}else{

gettimeofday(&end, NULL);

if(WIFSIGNALED(status)){

printf("Command returned with signal %d\n", WTERMSIG(status));

}else{

printf("Command returned with code %d\n", WEXITSTATUS(status));

}

timersub(&end, &start, &diff);

printf("consuming %lu.%03lu real seconds, %lu.%03lu user, %lu.%03lu system\n", diff.tv\_sec, diff.tv\_usec, ru.ru\_utime.tv\_sec, ru.ru\_utime.tv\_usec, ru.ru\_stime.tv\_sec, ru.ru\_stime.tv\_usec);

}

break;

}

}

return;

}

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

char \*line = NULL;

FILE \*input = stdin;

size\_t read = 0, length = 0;

do{

printf("$ ");

if(read = getline(&line, &length, input) == -1) break;

if(\*line == '#') continue;

if(line[read-1] == '\n'){

line[read-1] == '\0';

}

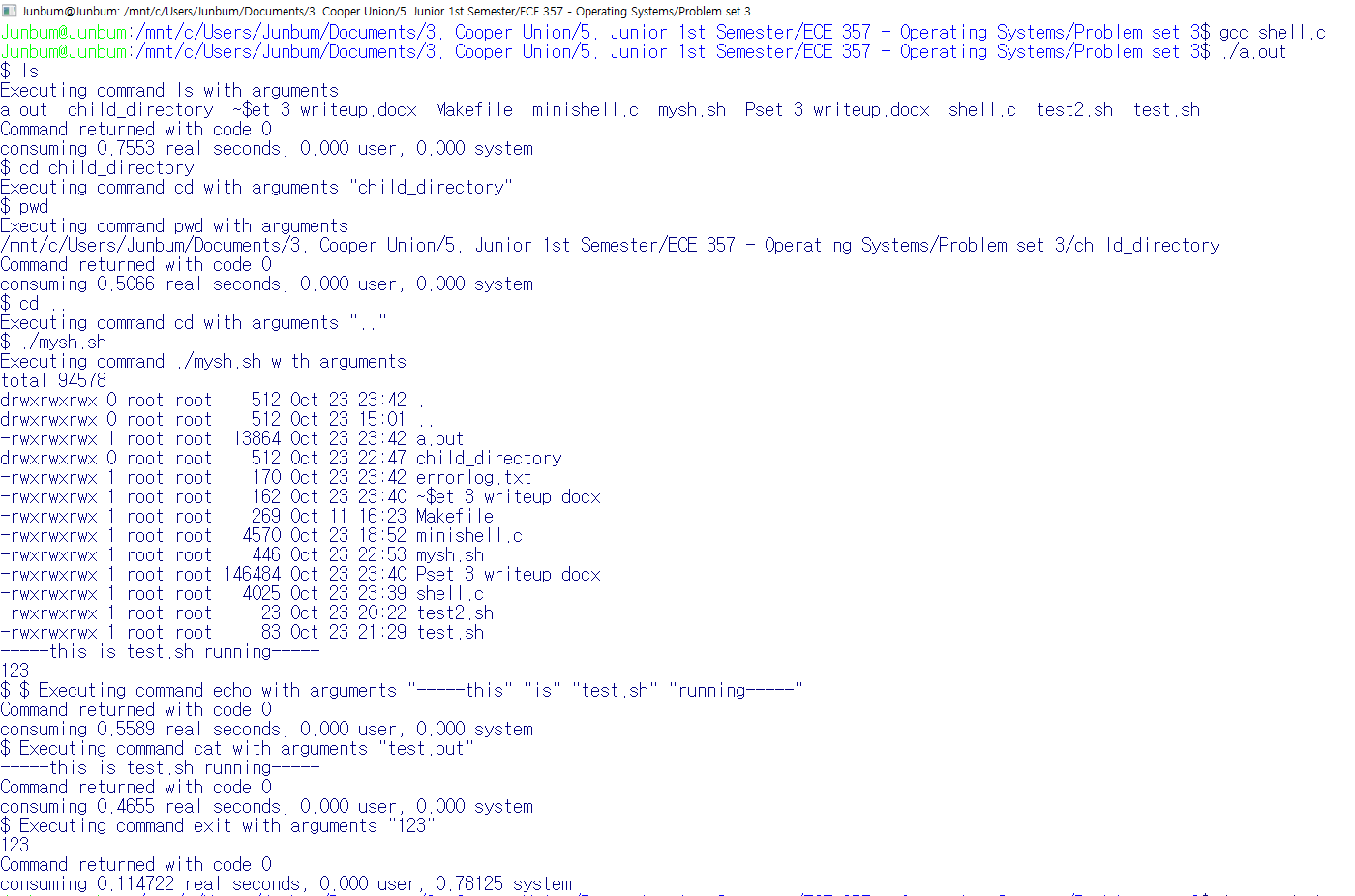
process(line);

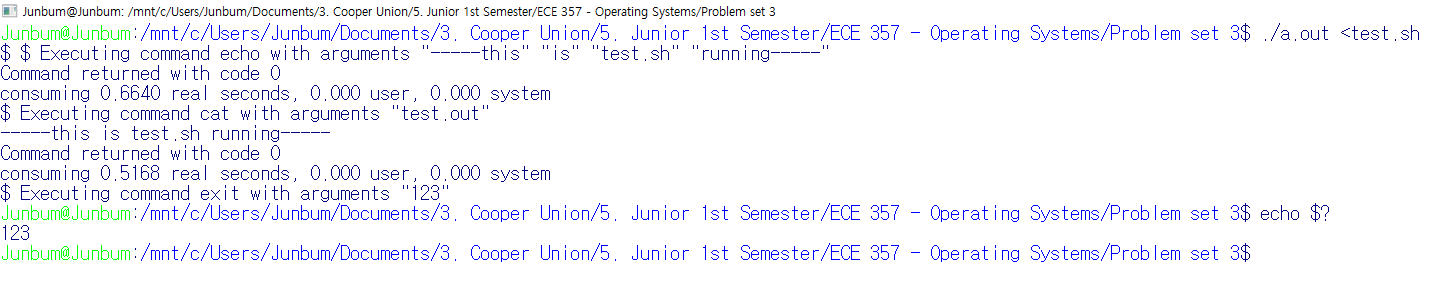
}while(1);

return 0;

}

1. **Screenshot of a Sample Run**



****

1. **Shell Scripts used for Testing**
2. mysh.sh

#!/bin/bash

cat shell.c >shellcopy

diff shellcopy shell.c

rm shellcopy

#this command will return error as there is no ll command found by this shell as ll is not aliased as ls -la

ll 2>errorlog.txt

#cd to non existent file

cd filethatdoesnotexist 2>>errorlog.txt

rm 2>>errorlog.txt

#prints out list of file in the directory

ls -la 2>>errorlog.txt

./test.sh

echo $?

#test.sh returns with value 123

./a.out <test.sh

echo $?

#./a.out returns 0

exit

1. test.sh

#!/bin/bash

echo -----this is test.sh running----- >test.out

cat test.out

exit 123