CSC3150_assignment1_report

GongQian_120090587

2022/10/10

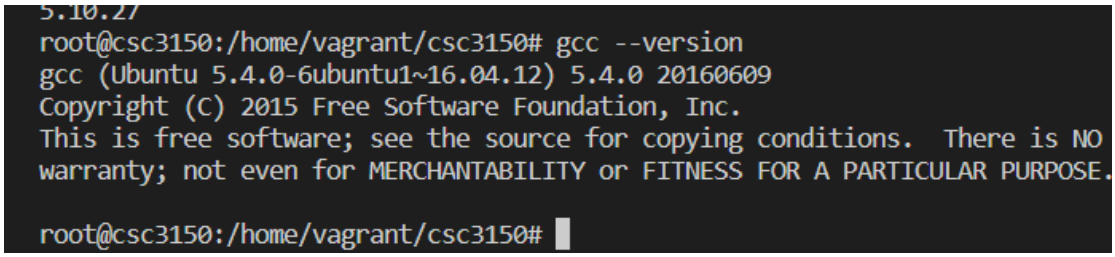
Contents

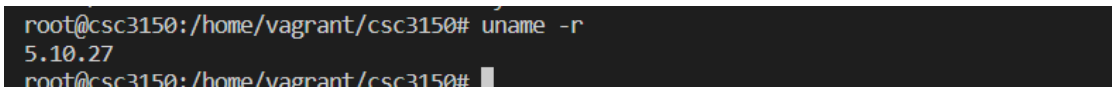
Overview

The CSC3050 assignment focuses on helping students grasp the details of multi-process programming. The content can be divided into two parts, one is the user mode multi-process programming in program 1, and the other is the kernel mode multi-process programming in program 2. Program 1 mainly focuses on the functions of dividing the fork process, executing the test program, while the parent process waits for the child process to terminate, and processing and outputting different signals sent by the child process in user mode. The functions implemented in program 2 include inserting a new model and creating a kernel thread, creating a new process in kernel mode, and the parent process waiting for the child process and processing some signals. In addition, this is a multi-process programming problem, mainly using a recursive way to execute files on the terminal, but also requires us to combine the various processes.

Environment

The environment of running my programs is the following:





Program Design

Set up Development Environment

To get a proper environment for us to build our program. First, we should build a visual machine to let us cooperate in linux system which helps us better understand the user mode and kernel mode. And then use VS to connect the virtual machine to give us an interface. To get a linux kernel version 5.10.xx, we also need to download a proper tar.gz file to the visual machine then complile and install it. Finally we can get a proper Development environment for us to practice.

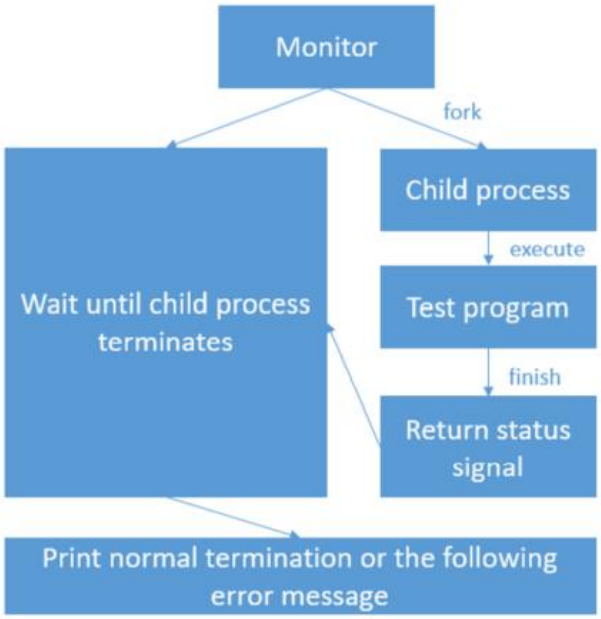
About the compilation process, we need to decompress the tar file to the VM directory and enter the corresponding command, such as “make menuconfig” to compile the file. Finally, install the kernel through the install command. Of course, it also needs to compile while we test program 2, but we can save a lot of time by using instructions, such as “make bzImage”, to compile only the changed parts.

Program

Here is the program design, it offers the basic idea for each program and some important detailed code.

Program 1

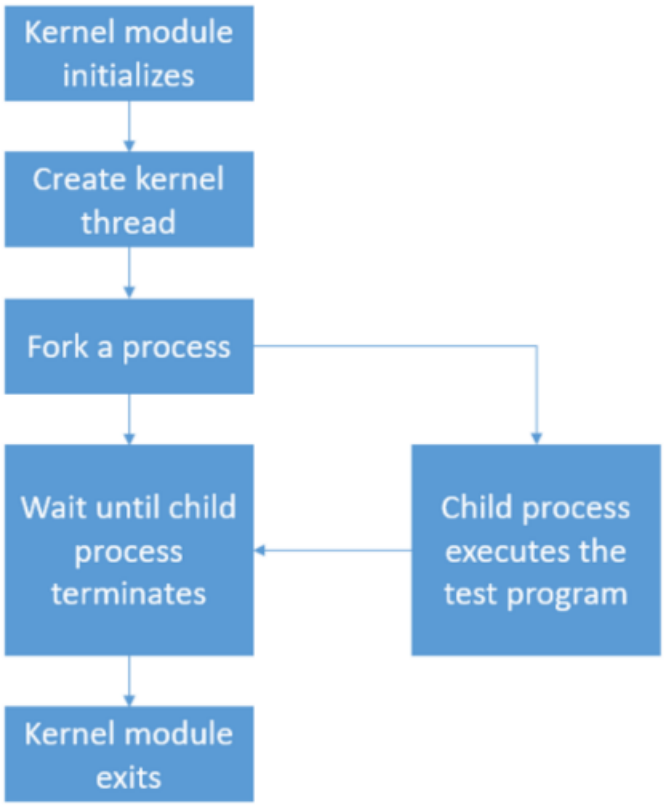
The program 1 chart flow is:



For This problem, we need to fork the process in user mode. The fork() function can help us build two different processes which have different pid. In the child process, we use the execve() function to test different test files. Then return the status signal to the parent process. To get better results, we use wait_pid(), which accepts the signal. we also get some different cases for different send-back signals. In short, as shown in the picture, first the parent and child processes are built, then the child processes run the test files, and finally, the signal is passed to the parent process to print out the results.

Program 2

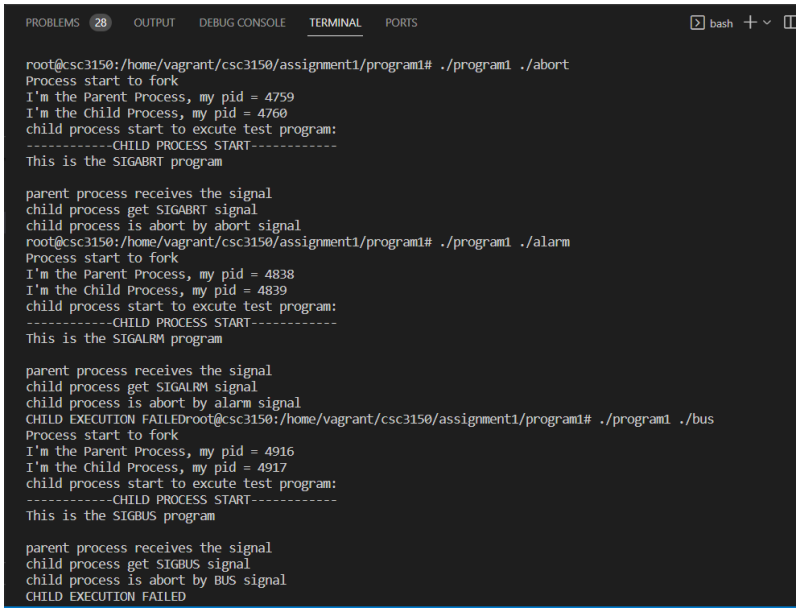
The program 2 chart flow is:

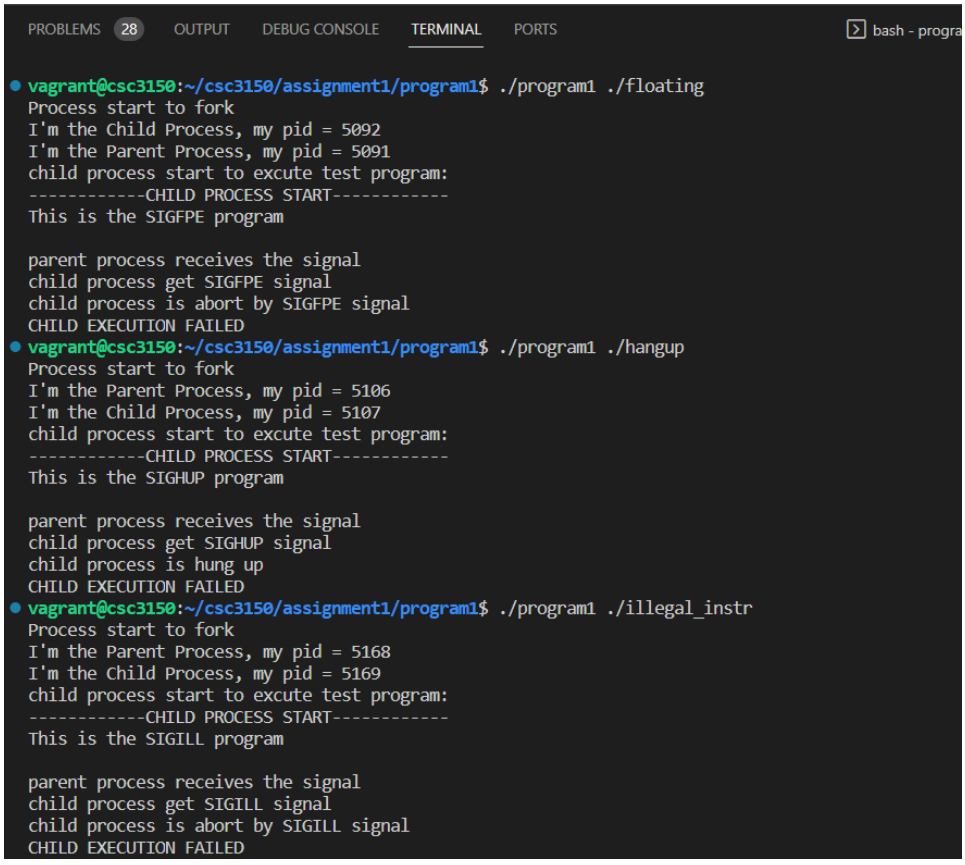


For this problem, we need to fork() the process in kernel mode. First, we shall make the kernel and kernel modules in the computer before we type the make command. program_init() function is used to initialize the kernel and create the kernel thread. my_exec() is used to test the kernel. my_fork() is used to fork the process and get the pid for the parent and child process. the my_wait() function is used to wait for the child process to terminate and send some signal to the parent. moreover, do_fork() is used to allocate a new process resources area in my_fork() with given function names and other parameters. my_exec() function called by do fork, will then comes in and start the execution of another test file. my_exec() takes care of locating the test file, passing in the arguments, and starting the execution. my_wait( )does its job in the parent process. Within the my_wait() function, struct wait_opts is constructed and passed to do_wait() as a parameter. Therefore, the parent process can check whether the child process is finished through the given child's pid.

Output Screenshots Demo

Program1





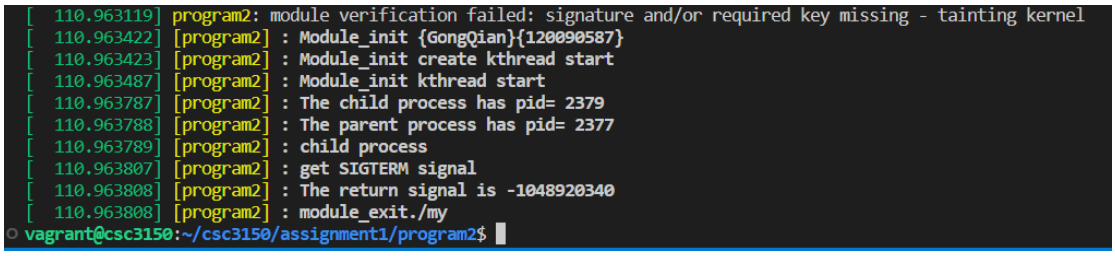
```
vagrant@csc3150:~/csc3150/assignment1/program1$ ./program1 ./stop
Process start to fork
I'm the Parent Process, my pid = 5473
I'm the Child Process, my pid = 5474
child process start to excute test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

parent process receives the signal
child process get SIGSTOP signal
vagrant@csc3150:~/csc3150/assignment1/program1$ ./program1 ./terminate
Process start to fork
I'm the Parent Process, my pid = 5556
I'm the Child Process, my pid = 5557
child process start to excute test program:
------------CHILD PROCESS START------------
This is the SIGTERM program

parent process receives the signal
child process get SIGTERM signal
child process is abort by SIGTERM signal
CHILD EXECUTION FAILED
vagrant@csc3150:~/csc3150/assignment1/program1$ ./program1 ./trap
Process start to fork
I'm the Parent Process, my pid = 5573
I'm the Child Process, my pid = 5574
child process start to excute test program:
------------CHILD PROCESS START------------
This is the SIGTRAP program

parent process receives the signal
child process get SIGTRAP signal
child process is abort by SIGTRAP signal
CHILD EXECUTION FAILED
vagrant@csc3150:~/csc3150/assignment1/program1$
```

```
vagrant@csc3150:~/csc3150/assignment1/program1$ ./program1 ./interrupt
Process start to fork
I'm the Parent Process, my pid = 5232
I'm the Child Process, my pid = 5233
child process start to excute test program:
------------CHILD PROCESS START------------
This is the SIGINT program

parent process receives the signal
child process get SIGINT signal
child process is abort by SIGINT signal
CHILD EXECUTION FAILED
vagrant@csc3150:~/csc3150/assignment1/program1$ ./program1 ./kill
Process start to fork
I'm the Parent Process, my pid = 5258
I'm the Child Process, my pid = 5259
child process start to excute test program:
------------CHILD PROCESS START------------
This is the SIGKILL program

parent process receives the signal
child process get SIGKILL signal
child process is abort by SIGKILL signal
CHILD EXECUTION FAILED
vagrant@csc3150:~/csc3150/assignment1/program1$ ./program1 ./normal
Process start to fork
I'm the Parent Process, my pid = 5305
I'm the Child Process, my pid = 5306
child process start to excute test program:
------------CHILD PROCESS START------------
This is the normal program

------------CHILD PROCESS END------------
parent process receives the signal
Normal termination with EXIT STATUS = 0
vagrant@csc3150:~/csc3150/assignment1/program1$
```

```
vagrant@csc3150:~/csc3150/assignment1/program1$ ./program1 ./stop
Process start to fork
I'm the Parent Process, my pid = 5473
I'm the Child Process, my pid = 5474
child process start to excute test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

parent process receives the signal
child process get SIGSTOP signal
vagrant@csc3150:~/csc3150/assignment1/program1$ ./program1 ./terminate
Process start to fork
I'm the Parent Process, my pid = 5556
I'm the Child Process, my pid = 5557
child process start to excute test program:
------------CHILD PROCESS START------------
This is the SIGTERM program

parent process receives the signal
child process get SIGTERM signal
child process is abort by SIGTERM signal
CHILD EXECUTION FAILED
vagrant@csc3150:~/csc3150/assignment1/program1$ ./program1 ./trap
Process start to fork
I'm the Parent Process, my pid = 5573
I'm the Child Process, my pid = 5574
child process start to excute test program:
------------CHILD PROCESS START------------
This is the SIGTRAP program

parent process receives the signal
child process get SIGTRAP signal
child process is abort by SIGTRAP signal
CHILD EXECUTION FAILED
vagrant@csc3150:~/csc3150/assignment1/program1$
```

```
vagrant@csc3150:~/csc3150/assignment1/program1$ ./program1 ./quit
Process start to fork
I'm the Parent Process, my pid = 5358
I'm the Child Process, my pid = 5359
child process start to excute test program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

parent process receives the signal
child process get SIGQUIT signal
child process is abort by SIGQUIT signal
CHILD EXECUTION FAILED
vagrant@csc3150:~/csc3150/assignment1/program1$ ./programm1 ./segment_fault
bash: ./programm1: No such file or directory
vagrant@csc3150:~/csc3150/assignment1/program1$ ./program1 ./segment_fault
Process start to fork
I'm the Parent Process, my pid = 5446
I'm the Child Process, my pid = 5447
child process start to excute test program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

parent process receives the signal
child process get SIGSEGV signal
child process is abort by SIGSEGV signal
CHILD EXECUTION FAILED
```

Program2



Summarize

I must admit that I have gained more from self-study and teamwork than from learning knowledge from books. This assignment is a great challenge for me. The unfamiliarity with c language and linux system makes it very difficult. But finally, with the help of my friends and the answer of my teaching assistant, the problem was gradually solved. I also through consulting materials and self-study to obtain more extensive knowledge outside the textbook, and a more solid foundation. In terms of knowledge, through hands-on practice, I have gained a new understanding of the operating principle of the operating system, the conversion between user mode and kernel mode, and so on