

THE CNINESE UNIVERSITY OF HONG KONG, SHENZHEN

CSC3150

Operating Systems

---

## Report for Assignment 1

---

Author Name

梁智昊

Student Number

120090780

# Contents

1. Design .....	3
2. Environment .....	4
3. Execution .....	5
4. Output .....	7
5. Feeling .....	17

# 1 Design

Assignment 1 requires us to do some programming about process in user mode and kernel mode. We need to set up the environment for the virtual machine, write the code to deal with process operations, compile the kernel and modify its source code.

## Task 1

Task 1 aims to fork a process, execute the test program and print out the signal information. Since it is based on user mode, we can use the API like `fork()`, `execve()` and `waitpid()` conveniently. Following the instructions from tut sildes, I fistly fork the process. Then, I print out the pid of parent process and child process. After that, I execute the test program in child process. The parent process will wait until the terminated of the child process and judge which condition is based on the (normal/SIG/stop) signal. Here I use switch-case to do the classification. The whole procedure is clear and easy for task 1

## Task 2

Task 2 is similar with task 1 in procedure. The difference is that it is written in kernel mode, which means the implementation of fork, execute and wait is more underlying/basic. Here I used `my_fork()` to build a `kernel_thread`, which is used to fork a child process to run the test program through `my_execve()` function. In the `my_execve()` function, the executable file name is specified, and the kernel API function `do_execve()` is responsible for running the .exe file. At the same time, the parent process will wait for the end of its child due to `my_wait()`. In `my_wait()` function, the actual waiting action is implemented by the `do_wait()` function. The information

of the child process' s status is stored in `*wo.wo_stat` variable. Different from the task 1, I used `if -else if -else` structure to classify the situation based on the value which is stored in the `wo.wo_stat`. After judging, the program printed out the corresponding information to kernel log. Finally, exit the module. (Code part).

As for the part beyond the code, we need to export the symbol of `getname_kernel()`, `do_execve()`, `kernel_clone` and `do_wait()` functions to make use of them since they belong to the kernel API. After the modification, recompilation of the kernel is needed. The updated modules need to be installed. Although many times of failure were met during the compilation of the kernel, I still successfully finished it in 3 days.

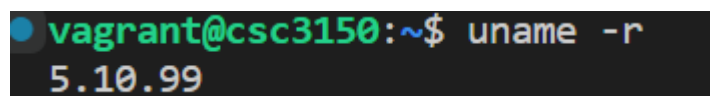
### Some details:

1. The flag of `wo` (struct `wait_opts`) should be set to `WEXITED | WUNTRACED` to deal with the stop signal.
2. The `wo.wo_stat` can be changed to the int numbers which refers to the standard signals through `0xff` and `0x7f`.

## 2. Environment

OS version: Linux Ubuntu 5.4.0-6ubuntu1~16.04.12

Kernel version:



```
vagrant@csc3150:~$ uname -r
5.10.99
```

# 3 Execution

This part shows the execution steps of the program.

## Task 1:

1. Cd to the directory of program 1 (it contains source codes and a makefile)
2. Type `gcc -o program1 program1.c` in the terminal to gain the .exe file of program1.
3. Type `gcc -o TESTFILE TESTFILE.c` in the terminal to gain the .exe file of test files.
4. Type `./program1 ./TESTFILE` in the terminal to run the program.

## Task 2:

Before the execution, certain operations to kernel need to be done.

1. Download the kernel of 5.10.x edition from the internet.
2. Compile the kernel:
  1. `sudo su;`
  2. `cd` the folder where `KERNEL_FILE.tar.xz` exist.
  3. `sudo tar xvf KERNEL_FILE.tar.xz`
  4. `cd KERNEL_FILE`
  5. add `EXPORT_SYMBOL()` for 4 functions invoked
  6. `make mrproper;`
  7. `make clean;`

8. make menuconfig (need to install a tool here);

9. make bzImage;

10. make modules;

11. make modules\_install;

12. make install;

3. reboot

4. (Start execution) Cd to the directory of program 2 (containing all the source codes and makefile).

5. Compile the test program(s): type gcc -o test test.c in the terminal. You can change the Raise signal in the test.c to test different conditions.

6. Type "make" in the terminal to make the file.(make clean can clean the rebuild file)

7. Type "sudo insmod program2.ko" to install the module.

8. Type "rmmod program2" to remove the module.

9.Type "dmesg" to print the result and check.

## 4 Output

This part shows some outputs of the programs.

### Task 1

T1. abort

```
Process start to fork
I'm the Parent Process, my pid = 20542
I'm the Child Rrocess, my pid = 20541
Child process start to execute test program:
-----CHILD PROCESS START-----
This is the SIGABRT program

Parent process receive SIGCHLD signal
child process get SIGABRT signal
```

T1. alarm

```
Process start to fork
I'm the Parent Process, my pid = 20593
I'm the Child Rrocess, my pid = 20592
Child process start to execute test program:
-----CHILD PROCESS START-----
This is the SIGALRM program

Parent process receive SIGCHLD signal
child process get SIGALRM signal
```

T1. bus

```
Process start to fork
I'm the Parent Process, my pid = 20671
I'm the Child Rrocess, my pid = 20670
Child process start to execute test program:
-----CHILD PROCESS START-----
This is the SIGBUS program

Parent process receive SIGCHLD signal
child process get SIGBUS signal
```

T1. floating

```
Process start to fork
I'm the Parent Process, my pid = 20719
I'm the Child Rrocess, my pid = 20718
Child process start to execute test program:
-----CHILD PROCESS START-----
This is the SIGFPE program

Parent process receive SIGCHLD signal
child process get SIGFPE signal
```

T1. hang up

```
Process start to fork
I'm the Parent Process, my pid = 20767
I'm the Child Rrocess, my pid = 20766
Child process start to execute test program:
-----CHILD PROCESS START-----
This is the SIGHUP program

Parent process receive SIGCHLD signal
child process get SIGHUP signal
```



#### T1. illegal\_instr

```
Process start to fork
I'm the Parent Process, my pid = 20845
I'm the Child Rrocess, my pid = 20844
Child process start to execute test program:
-----CHILD PROCESS START-----
This is the SIGILL program

Parent process receive SIGCHLD signal
child process get SIGILL signal
```

#### T1. interrupt

```
Process start to fork
I'm the Parent Process, my pid = 20912
I'm the Child Rrocess, my pid = 20911
Child process start to execute test program:
-----CHILD PROCESS START-----
This is the SIGINT program

Parent process receive SIGCHLD signal
child process get SIGINT signal
```

#### T1. kill

```
Process start to fork
I'm the Parent Process, my pid = 21005
I'm the Child Rrocess, my pid = 21004
Child process start to execute test program:
-----CHILD PROCESS START-----
This is the SIGKILL program

Parent process receive SIGCHLD signal
child process get SIGKILL signal
```

T1. normal

```
Process start to fork
I'm the Parent Process, my pid = 21045
I'm the Child Rrocess, my pid = 21044
Child process start to execute test program:
-----CHILD PROCESS START-----
This is the normal program

-----CHILD PROCESS END-----
Parent process receive SIGCHLD signal
Normal termination with EXIT STATUS = 0
```

T1. pipe

```
Process start to fork
I'm the Parent Process, my pid = 21091
I'm the Child Rrocess, my pid = 21090
Child process start to execute test program:
-----CHILD PROCESS START-----
This is the SIGPIPE program

Parent process receive SIGCHLD signal
child process get SIGPIPE signal
```

T1. quit

```
Process start to fork
I'm the Parent Process, my pid = 21142
I'm the Child Rrocess, my pid = 21141
Child process start to execute test program:
-----CHILD PROCESS START-----
This is the SIGQUIT program

Parent process receive SIGCHLD signal
child process get SIGQUIT signal
```

## T1. segment\_fault

```
Process start to fork
I'm the Parent Process, my pid = 21335
I'm the Child Rrocess, my pid = 21334
Child process start to execute test program:
-----CHILD PROCESS START-----
This is the SIGSEGV program

Parent process receive SIGCHLD signal
child process get SIGSEGV signal
```

## T1. stop

```
Process start to fork
I'm the Parent Process, my pid = 21402
I'm the Child Rrocess, my pid = 21401
Child process start to execute test program:
-----CHILD PROCESS START-----
This is the SIGSTOP program

Parent process receive SIGCHLD signal
child process get SIGSTOP signal
```

## T1. terminal

```
Process start to fork
I'm the Parent Process, my pid = 21481
I'm the Child Rrocess, my pid = 21480
Child process start to execute test program:
-----CHILD PROCESS START-----
This is the SIGTERM program

Parent process receive SIGCHLD signal
child process get SIGTERM signal
```

## T1. trap

```
Process start to fork
I'm the Parent Process, my pid = 21533
I'm the Child Rrocess, my pid = 21532
Child process start to execute test program:
-----CHILD PROCESS START-----
This is the SIGTRAP program

Parent process receive SIGCHLD signal
child process get SIGTRAP signal
```

## Task 2

### T2. abort

```
[12312.913165] [program2] : Module_init {梁智昊} {120090780}
[12312.913166] [program2] : Module_init create kthread start
[12312.913247] [program2] : Module_init Kthread starts
[12312.913303] [program2] : The child process has pid = 22181
[12312.913304] [program2] : This is the parent process, pid = 22180
[12312.913305] [program2] : child process
[12313.054307] [program2] : child process get SIGABRT signal
[12313.054309] [program2] : child process terminated
[12313.054310] [Kernel Thread] : The return signal is 6
[12321.437705] [program2] : Module_exit
```

### T2. alarm

```
[12439.406494] [program2] : Module_init {梁智昊} {120090780}
[12439.406497] [program2] : Module_init create kthread start
[12439.406631] [program2] : Module_init Kthread starts
[12439.406731] [program2] : The child process has pid = 22544
[12439.406733] [program2] : This is the parent process, pid = 22543
[12439.406734] [program2] : child process
[12439.407563] [program2] : child process get SIGALRM signal
[12439.407566] [program2] : child process terminated
[12439.407568] [Kernel Thread] : The return signal is 14
[12448.798928] [program2] : Module_exit
```

## T2. bus

```
[12584.827102] [program2] : Module_init {梁智昊} {120090780}
[12584.827103] [program2] : Module_init create kthread start
[12584.827173] [program2] : Module_init Kthread starts
[12584.827229] [program2] : The child process has pid = 22842
[12584.827230] [program2] : This is the parent process, pid = 22841
[12584.827231] [program2] : child process
[12584.976210] [program2] : child process get SIGBUS signal
[12584.976211] [program2] : child process terminated
[12584.976212] [program2] : child process terminated
[12584.976213] [Kernel Thread] : The return signal is 7
[12590.316944] [program2] : Module_exit
```

## T2. floating

```
[12671.067327] [program2] : Module_init {梁智昊} {120090780}
[12671.067330] [program2] : Module_init create kthread start
[12671.067569] [program2] : Module_init Kthread starts
[12671.067690] [program2] : The child process has pid = 23055
[12671.067693] [program2] : This is the parent process, pid = 23054
[12671.067694] [program2] : child process
[12671.227985] [program2] : child process get SIGFPE signal
[12671.227987] [program2] : child process terminated
[12671.227988] [Kernel Thread] : The return signal is 8
[12677.050989] [program2] : Module_exit
```

## T2. hang up

```
[12775.195198] [program2] : Module_init {梁智昊} {120090780}
[12775.195201] [program2] : Module_init create kthread start
[12775.195547] [program2] : Module_init Kthread starts
[12775.195643] [program2] : The child process has pid = 23291
[12775.195645] [program2] : This is the parent process, pid = 23290
[12775.195646] [program2] : child process
[12775.196718] [program2] : child process get SIGHUP signal
[12775.196720] [program2] : child process terminated
[12775.196721] [Kernel Thread] : The return signal is 1
[12785.766038] [program2] : Module_exit
```

## T2. illegal\_instr

```
[12846.048795] [program2] : Module_init {梁智昊} {120090780}
[12846.048797] [program2] : Module_init create kthread start
[12846.048949] [program2] : Module_init Kthread starts
[12846.049043] [program2] : The child process has pid = 23469
[12846.049046] [program2] : This is the parent process, pid = 23468
[12846.049048] [program2] : child process
[12846.191749] [program2] : child process get SIGILL signal
[12846.191751] [program2] : child process terminated
[12846.191752] [Kernel Thread] : The return signal is 4
[12848.077443] [program2] : Module_exit
```

## T2. interrupt

```
[12945.712264] [program2] : Module_init {梁智昊} {120090780}
[12945.712266] [program2] : Module_init create kthread start
[12945.712410] [program2] : Module_init Kthread starts
[12945.712476] [program2] : The child process has pid = 23656
[12945.712478] [program2] : This is the parent process, pid = 23655
[12945.712479] [program2] : child process
[12945.713170] [program2] : child process get SIGINT signal
[12945.713172] [program2] : child process terminated
[12945.713174] [Kernel Thread] : The return signal is 2
[12954.851062] [program2] : Module_exit
```

## T2. kill

```
[13117.292276] [program2] : Module_init {梁智昊} {120090780}
[13117.292278] [program2] : Module_init create kthread start
[13117.292355] [program2] : Module_init Kthread starts
[13117.292412] [program2] : The child process has pid = 24016
[13117.292414] [program2] : This is the parent process, pid = 24015
[13117.292415] [program2] : child process
[13117.293288] [program2] : child process get SIGKILL signal
[13117.293289] [program2] : child process terminated
[13117.293290] [program2] : child process terminated
[13117.293291] [Kernel Thread] : The return signal is 9
[13123.404890] [program2] : Module_exit
```



## T2. pipe

```
[13705.389035] [program2] : Module_init {梁智昊} {120090780}
[13705.389036] [program2] : Module_init create kthread start
[13705.389164] [program2] : Module_init Kthread starts
[13705.389213] [program2] : The child process has pid = 25121
[13705.389214] [program2] : This is the parent process, pid = 25120
[13705.389215] [program2] : child process
[13705.389767] [program2] : child process get SIGPIPE signal
[13705.389768] [program2] : child process terminated
[13705.389769] [Kernel Thread] : The return signal is 13
[13713.999368] [program2] : Module_exit
```

## T2. quit

```
[13163.323779] [program2] : Module_init {梁智昊} {120090780}
[13163.323781] [program2] : Module_init create kthread start
[13163.323957] [program2] : Module_init Kthread starts
[13163.324026] [program2] : The child process has pid = 24215
[13163.324027] [program2] : This is the parent process, pid = 24214
[13163.324029] [program2] : child process
[13163.481483] [program2] : child process get SIGQUIT signal
[13163.481485] [program2] : child process terminated
[13163.481486] [Kernel Thread] : The return signal is 3
[13169.393199] [program2] : Module_exit
```

## T2. segment\_fault

```
[13201.385062] [program2] : Module_init {梁智昊} {120090780}
[13201.385064] [program2] : Module_init create kthread start
[13201.385120] [program2] : Module_init Kthread starts
[13201.385277] [program2] : The child process has pid = 24344
[13201.385278] [program2] : This is the parent process, pid = 24343
[13201.385279] [program2] : child process
[13201.531711] [program2] : child process get SIGSEGV signal
[13201.531714] [program2] : child process terminated
[13201.531716] [Kernel Thread] : The return signal is 11
[13207.918544] [program2] : Module_exit
```

## T2. stop

```
[13244.568120] [program2] : Module_init {梁智昊} {120090780}
[13244.568123] [program2] : Module_init create kthread start
[13244.568300] [program2] : Module_init Kthread starts
[13244.568364] [program2] : The child process has pid = 24448
[13244.568366] [program2] : This is the parent process, pid = 24447
[13244.568367] [program2] : child process
[13244.569623] [program2] : child process get SIGSTOP signal
[13244.569625] [program2] : child process stopped
[13244.569626] [Kernel Thread] : The return signal is 19
[13250.822769] [program2] : Module_exit
```

## T2. trap

```
[13314.854776] [program2] : Module_init {梁智昊} {120090780}  
[13314.854778] [program2] : Module_init create kthread start  
[13314.854923] [program2] : Module_init Kthread starts  
[13314.854994] [program2] : The child process has pid = 24736  
[13314.854996] [program2] : This is the parent process, pid = 24735  
[13314.854997] [program2] : child process  
[13315.008023] [program2] : child process get SIGTRAP signal  
[13315.008025] [program2] : child process terminated  
[13315.008026] [Kernel Thread] : The return signal is 5  
[13320.345038] [program2] : Module_exit
```

## T2. normal

```
[14020.551220] [program2] : Module_init {梁智昊} {120090780}  
[14020.551223] [program2] : Module_init create kthread start  
[14020.551443] [program2] : Module_init Kthread starts  
[14020.551929] [program2] : The child process has pid = 25716  
[14020.551931] [program2] : This is the parent process, pid = 25715  
[14020.551932] [program2] : child process  
[14025.553715] [program2] : Child process terminated  
[14029.462349] [program2] : Module_exit
```



## 5 Feeling

CSC3150 is a hard course taught by professor Zhong. I think I must prepare well for the challenges. Although I was very worried about the homework and almost to give up at first, but finally I successfully finish most of the parts of the assignment only except the bonus. I am very proud of myself.

Assignment 1 itself is not too difficult but troublesome. For the knowledge and coding part, we just need to understand some definitions about the process and kernel and imitate the codes from tutorials. However, we need to set up the environment by ourselves during which there will be mountains of errors and bugs from the system. For me, I am not so familiar with the Linux system and need to search for the solutions of bugs from wechat groups and pizza. However, after doing this project I become more skillful in Linux system and virtual machine. I have learned some operations about process, how to compile and modify kernel, how to execute a program in the codes, and so on. It does enhance my computer ability.

Asking and communicating is the most important part in the course. Some problems can be solve easily if you gain the help from the student who is familiar with it as they might meet the same problems before. For me, I finally used 3 days to compile the kernel, but one of my friend can finish the task with my help within 1.5 hours. Once we communicate with each other, we can share our experience and help walk out of the difficulty. It does reduce many troubles after discussing with some excellent peers. As for the discussion platform, I think wechat group more effective than pizza as more excellent students are more active in we chat. Besides, TA can also help us in the we chat group. But at pizza, few of the answer is efficient. Therefore, I think the professor can give the permission of discussion in we chat group.

Tutorial is important. It is totally different from the lectures because it teaches some practical knowledge highly related to the projects. I think I need to treat tutorials seriously.

I can feel that TAs are hard-working and laborious for this course. They have large workloads and will be disturbed by mountains of questions raised by students. Good luck to you!

That' s all, thank you.