Report

Chen Xuanwen 120090582

1. Program 1

   a. Basic idea

      The program simulates how the process fork a child process and execute a test program, wait for its return signal in user mode. By using corresponding system call function print the related information.

   b. Implement

      At start, the program uses fork() to create a child process. Then check the pid to distinguish whether it is child or parent process. If child process, execute file and return corresponding signals when terminates. If parent process, wait for the terminated signal of child process and print the information.

   c. Environment

      Version of kernel: 5.10.30

   d. Execution

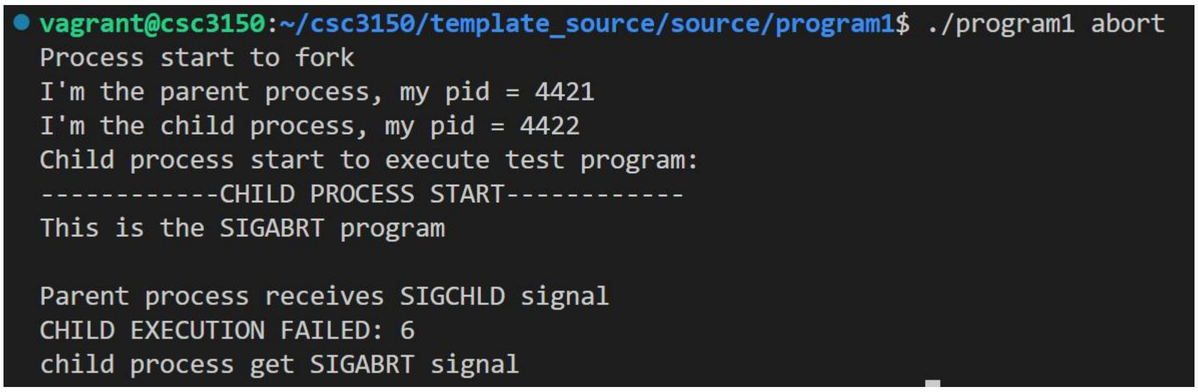
      i. Us command “cd” in terminal to go to the folder program1.

      ii. Type the command “sudo make” in the terminal.

      iii. Type “./program1 testfile” in the terminal(where the test file is abort, alarm…..), then check the result.

   e. Output samples

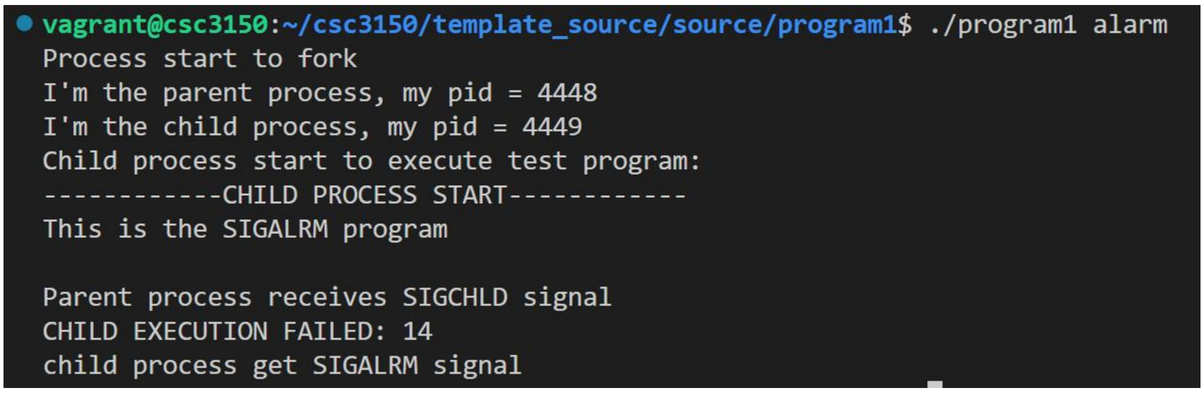
1. abort



```
vagrant@csc3150:~/csc3150/template_source/source/program1$ ./program1 abort
Process start to fork
I'm the parent process, my pid = 4421
I'm the child process, my pid = 4422
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGABRT program

Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 6
child process get SIGABRT signal
```

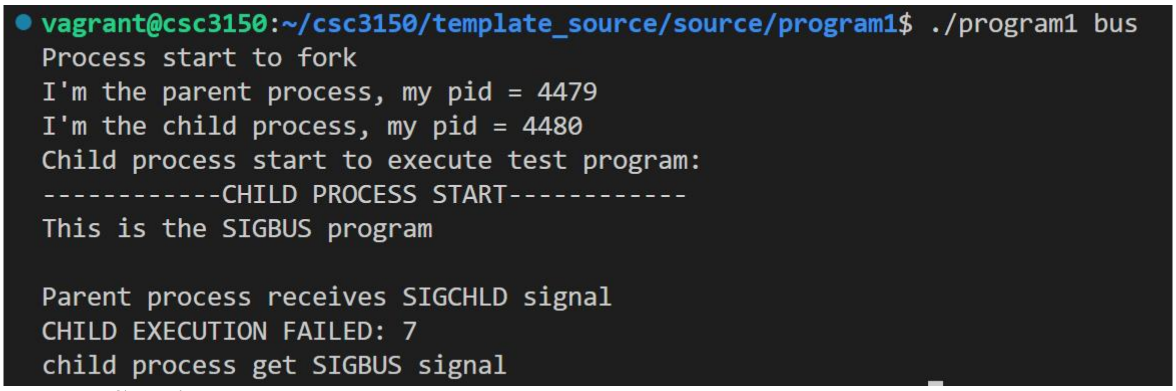
2. alarm



```
vagrant@csc3150:~/csc3150/template_source/source/program1$ ./program1 alarm
Process start to fork
I'm the parent process, my pid = 4448
I'm the child process, my pid = 4449
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGALRM program

Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 14
child process get SIGALRM signal
```

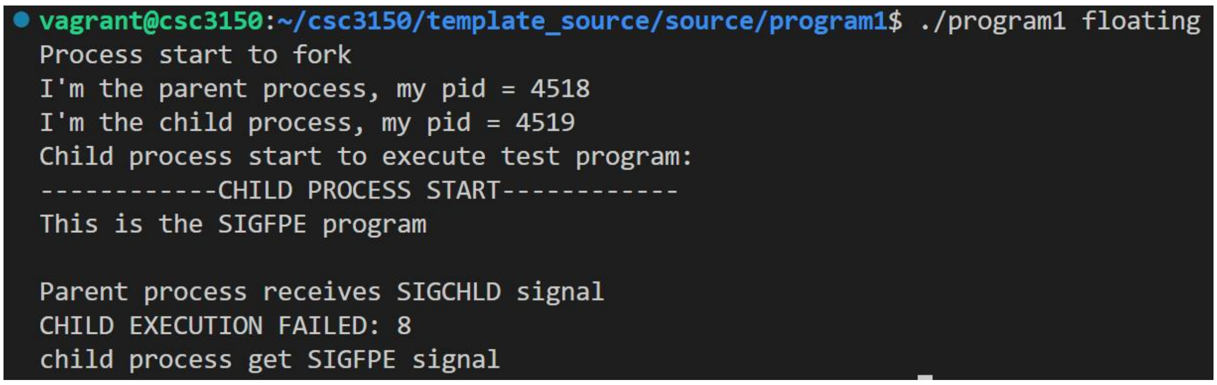
3. bus



```
vagrant@csc3150:~/csc3150/template_source/source/program1$ ./program1 bus
Process start to fork
I'm the parent process, my pid = 4479
I'm the child process, my pid = 4480
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGBUS program

Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 7
child process get SIGBUS signal
```

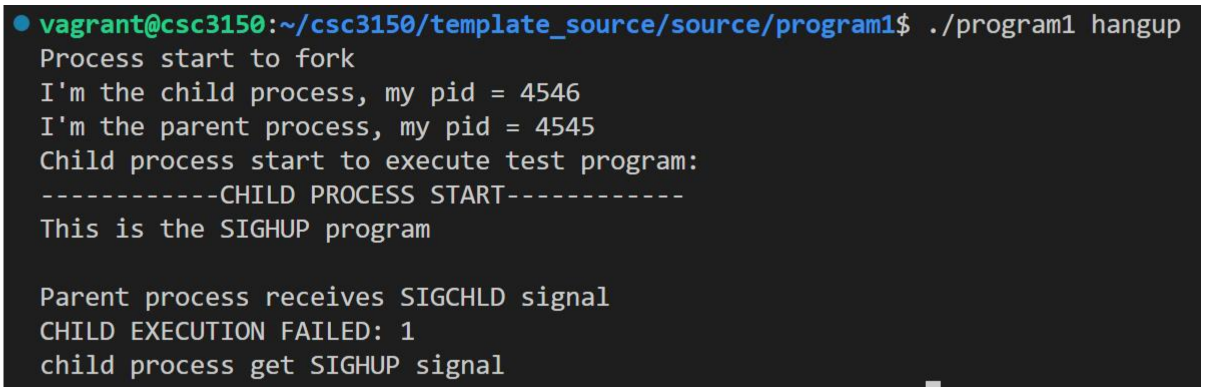
4. floating



```
vagrant@csc3150:~/csc3150/template_source/source/program1$ ./program1 floating
Process start to fork
I'm the parent process, my pid = 4518
I'm the child process, my pid = 4519
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGFPE program

Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 8
child process get SIGFPE signal
```

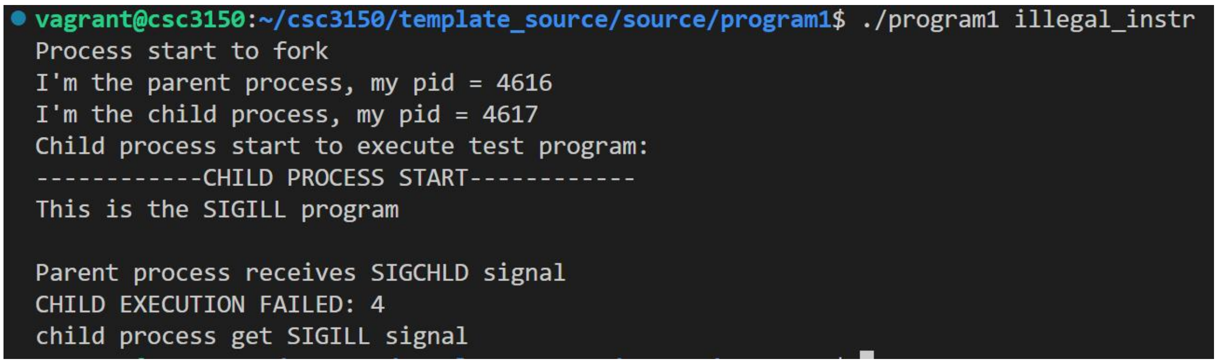
5. hangup



```
vagrant@csc3150:~/csc3150/template_source/source/program1$ ./program1 hangup
Process start to fork
I'm the child process, my pid = 4546
I'm the parent process, my pid = 4545
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGHUP program

Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 1
child process get SIGHUP signal
```

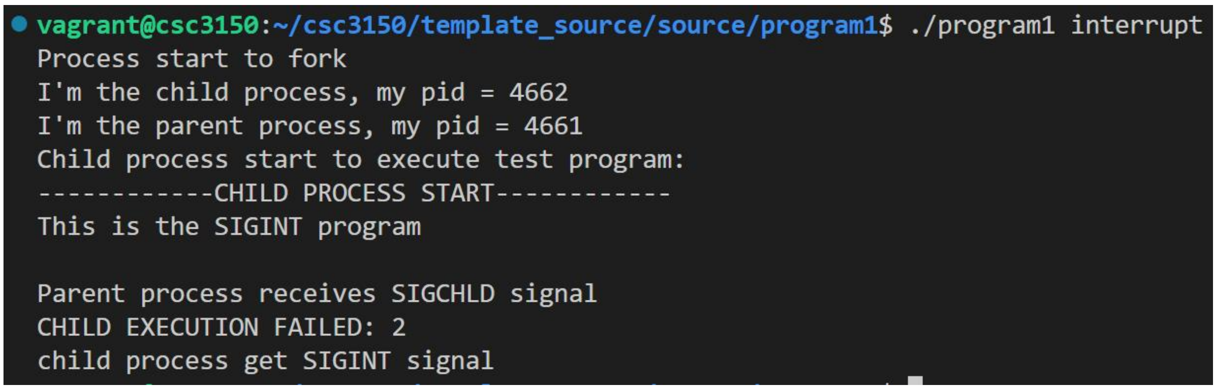
6. illegal_instr



```
vagrant@csc3150:~/csc3150/template_source/source/program1$ ./program1 illegal_instr
Process start to fork
I'm the parent process, my pid = 4616
I'm the child process, my pid = 4617
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGILL program

Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 4
child process get SIGILL signal
```

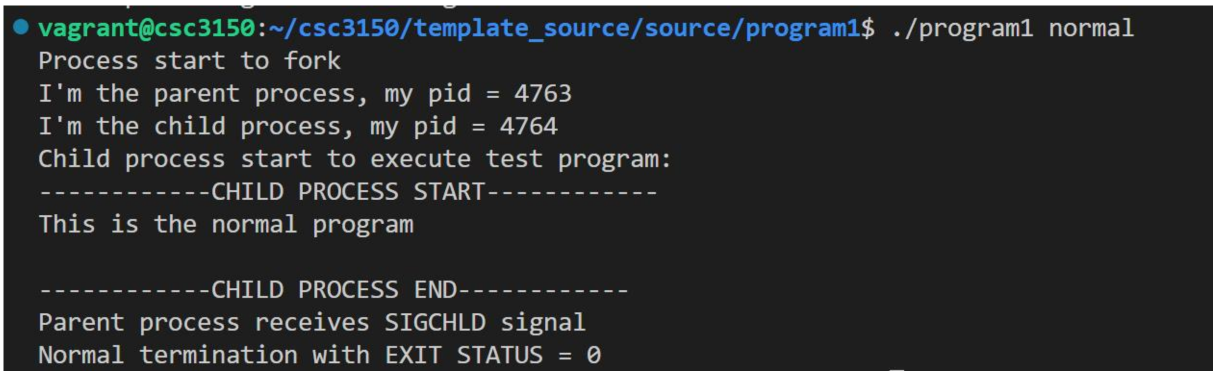
7. interrupt



```
vagrant@csc3150:~/csc3150/template_source/source/program1$ ./program1 interrupt
Process start to fork
I'm the child process, my pid = 4662
I'm the parent process, my pid = 4661
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGINT program

Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 2
child process get SIGINT signal
```

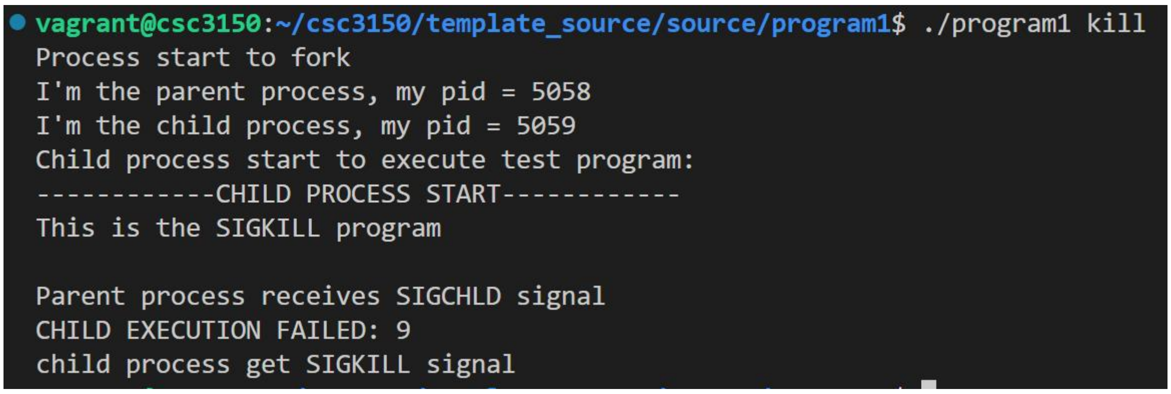
8. normal



```
vagrant@csc3150:~/csc3150/template_source/source/program1$ ./program1 normal
Process start to fork
I'm the parent process, my pid = 4763
I'm the child process, my pid = 4764
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the normal program

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

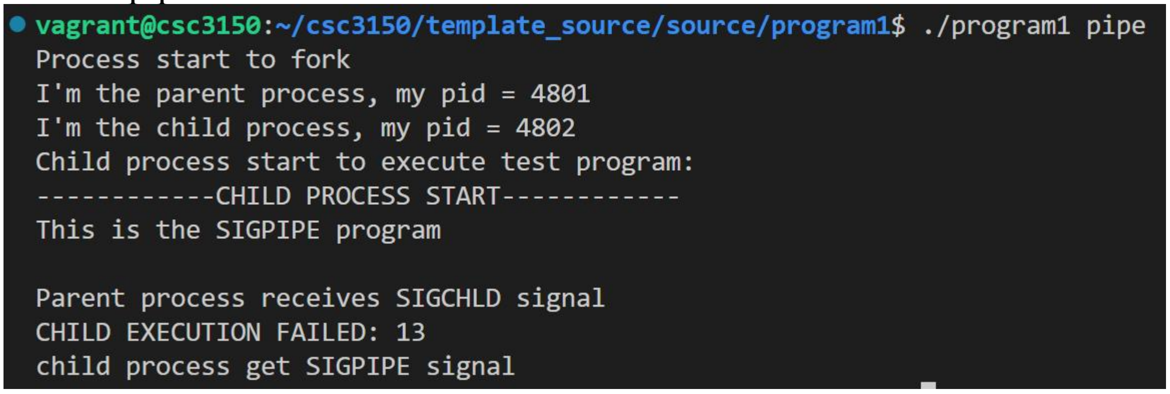
9. kill



```
vagrant@csc3150:~/csc3150/template_source/source/program1$ ./program1 kill
Process start to fork
I'm the parent process, my pid = 5058
I'm the child process, my pid = 5059
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGKILL program

Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 9
child process get SIGKILL signal
```

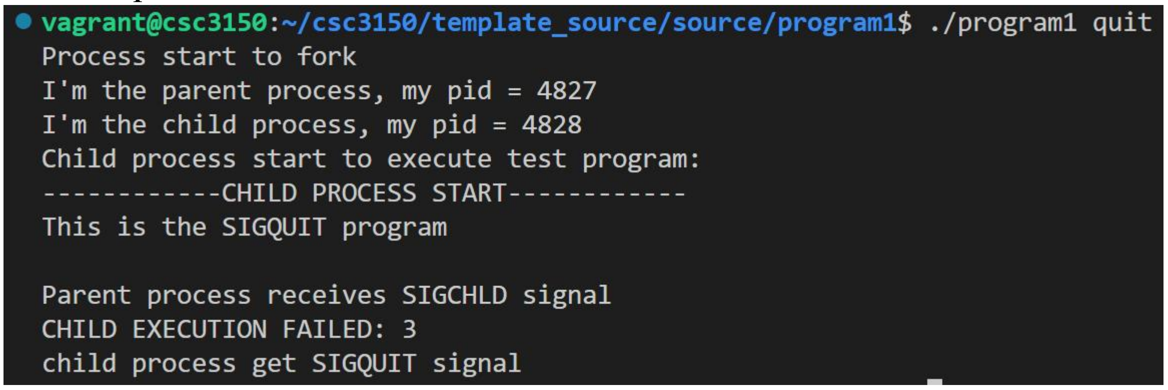
10.pipe



```
vagrant@csc3150:~/csc3150/template_source/source/program1$ ./program1 pipe
Process start to fork
I'm the parent process, my pid = 4801
I'm the child process, my pid = 4802
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGPIPE program

Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 13
child process get SIGPIPE signal
```

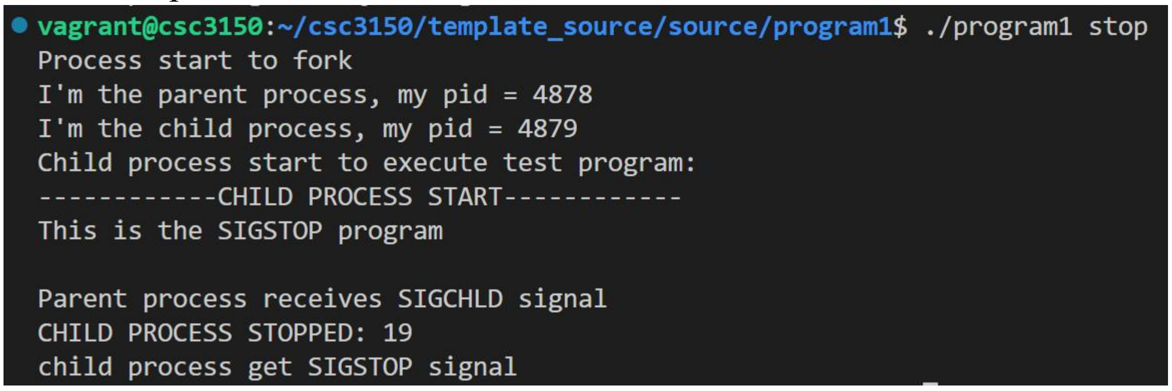
11.quit



```
vagrant@csc3150:~/csc3150/template_source/source/program1$ ./program1 quit
Process start to fork
I'm the parent process, my pid = 4827
I'm the child process, my pid = 4828
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 3
child process get SIGQUIT signal
```

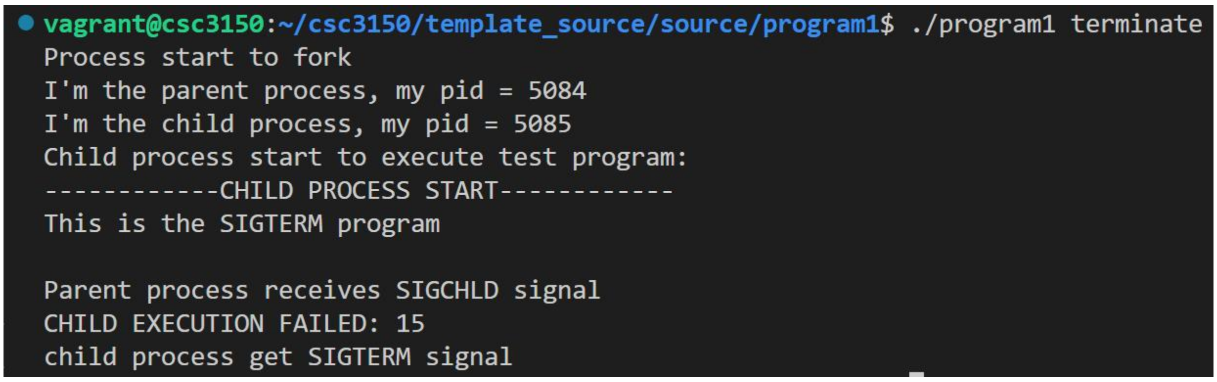
12.stop



```
vagrant@csc3150:~/csc3150/template_source/source/program1$ ./program1 stop
Process start to fork
I'm the parent process, my pid = 4878
I'm the child process, my pid = 4879
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

Parent process receives SIGCHLD signal
CHILD PROCESS STOPPED: 19
child process get SIGSTOP signal
```

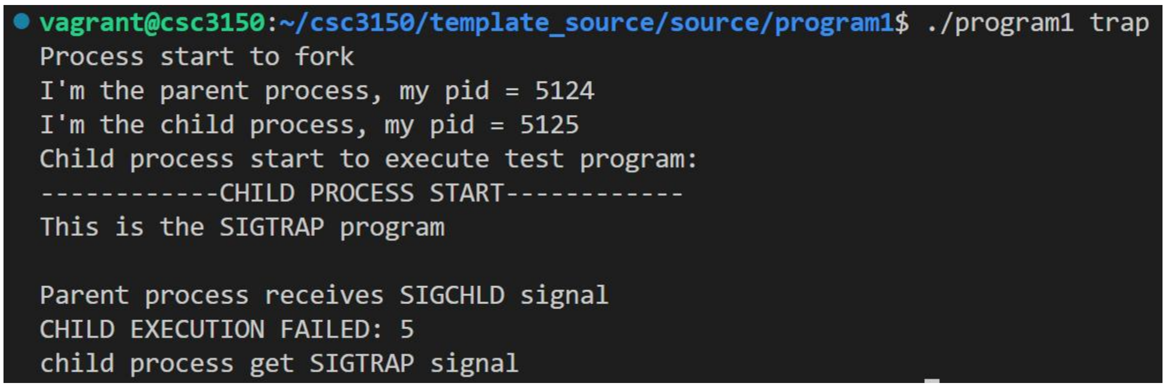
13.terminate



```
vagrant@csc3150:~/csc3150/template_source/source/program1$ ./program1 terminate
Process start to fork
I'm the parent process, my pid = 5084
I'm the child process, my pid = 5085
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTERM program

Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 15
child process get SIGTERM signal
```

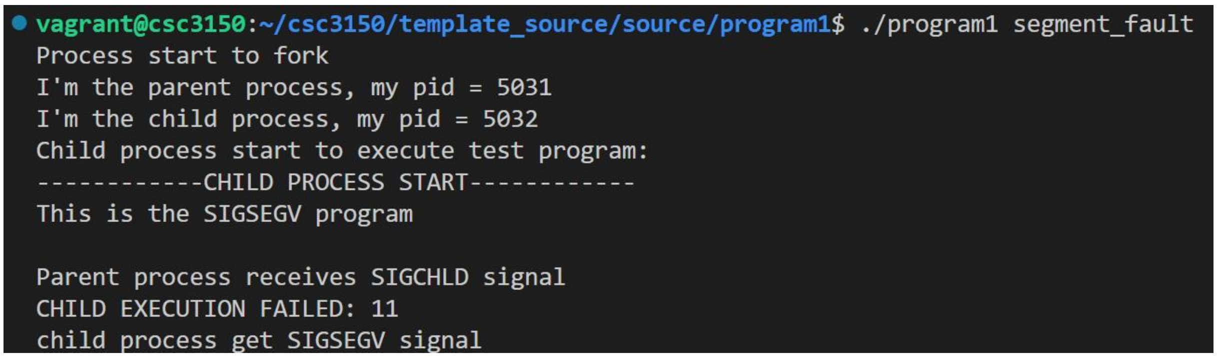
14.trap



```
vagrant@csc3150:~/csc3150/template_source/source/program1$ ./program1 trap
Process start to fork
I'm the parent process, my pid = 5124
I'm the child process, my pid = 5125
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTRAP program

Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 5
child process get SIGTRAP signal
```

15.segment_fault



```
vagrant@csc3150:~/csc3150/template_source/source/program1$ ./program1 segment_fault
Process start to fork
I'm the parent process, my pid = 5031
I'm the child process, my pid = 5032
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

Parent process receives SIGCHLD signal
CHILD EXECUTION FAILED: 11
child process get SIGSEGV signal
```

f. What I learnt from the project

I learnt how to use fork to create a child process and use it to do other tasks. I also have a better understanding about how parent and child process working mechanism, that when child process terminated, the parent process would receive a signal and know that child process ends.

2. Program 2

   a. How to set up the development environment

      i. At first, this assignment requires us to use linux kernel that version is later than linux-5.10. Go to the linux kernel website and download one of the latest version, mine is 5.10.30.

      ii. After download, we only get a linux-5.10.30.tar.xz file, grab it into the VSCode connected with VM by mouse. Typing command "sudo apt-get install libncurses-dev gawk flex bison openssl libssl-dev dkms libelf-dev libudev-dev libpci-dev libiberty-dev autoconf llvm dwarves" in the terminal to install dependency and development tools (First use cd and go to the directory where the linux.tar.xz file at)

      iii. Type "tar xvf kernel_file.tar.xz" in the terminal. (the kernel_file.tar.xz is the linux file, like linux-5.10.30.tar.xz)

      iv. Since there are several functions in the linux kernel that we are going to use in this assignment, like do_wait (/kernel/exit.c), kernel_clone(/kernel/fork.c) , do_execve(/fs/exec.c), getname_kernel(/fs/namei.c). At first we need to find the files containing these functions, then add EXPORT_SYMBOL(funcname) after where those functions are defined. To expose them so that we can use them in program2.c.

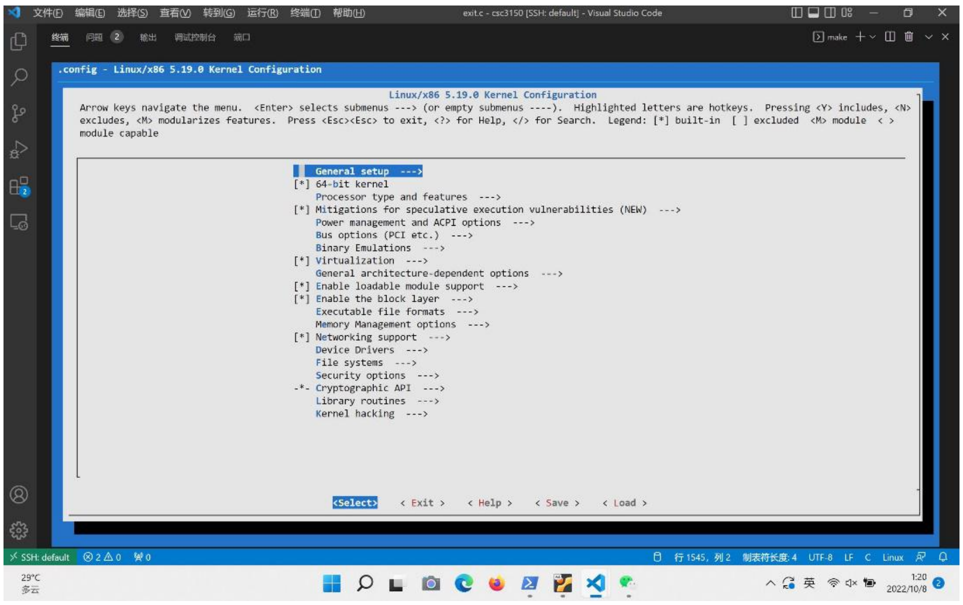
v. After modifying the files, we need to recompile the kernel.

1. Copy config file from /boot to the directory where the kernel folder is.
2. Log in root account and go to kernel source directory

   Type “sudo su”

   cd to the specific directory
3. Clean previous setting and start configuration

   Type “make mrproper”, “make clean”, “make menuconfig”,



```
.config - Linux/x86 5.19.0 Kernel Configuration
                          Linux/x86 5.19.0 Kernel Configuration
 Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N>
 excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < >
 module capable

                    General setup  --->
                [*] 64-bit kernel
                    Processor type and features  --->
                [*] Mitigations for speculative execution vulnerabilities (NEW)  --->
                    Power management and ACPI options  --->
                    Bus options (PCI etc.)  --->
                    Binary Emulations  --->
                [*] Virtualization  --->
                    General architecture-dependent options  --->
                [*] Enable loadable module support  --->
                [*] Enable the block layer  --->
                    Executable file formats  --->
                    Memory Management options  --->
                [*] Networking support  --->
                    Device Drivers  --->
                    File systems  --->
                    Security options  --->
                -*- Cryptographic API  --->
                    Library routines  --->
                    Kernel hacking  --->

                <Select>    < Exit >    < Help >    < Save >    < Load >
```

文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R) 终端(T) 帮助(H)
exit.c - csc3150 [SSH: default] - Visual Studio Code
终端 问题 2 输出 调试控制台 端口
make
SSH: default ⊗ 2 ⚠ 0 0
行 1545，列 2 制表符长度: 4 UTF-8 LF C Linux
29°C 多云
英 1:20 2022/10/8

   Save the config and exit.
4. Build kernel Image and modules

   Type “make -j$(nproc)”
5. Compile kernel

   “make modules_install”, “make install”, “reboot”

   Then the recompiling is down.

b. Implement

   i. When initiating the module, create a kernel thread using kthread_create() to run my_fork() function, where the program will fork a child process to execute another program.

   ii. To implement my_fork , there are a few functions needed to be implemented.

      a. my_exec():

         This function will use getname_kernel() to get the information of the execute file. After that, use do_execve() to execute the file.

      b. my_wait():

         This function gets terminated signal of child process using do_wait()

         *One thing needed to know that the return signal of do_wait function is not the same as the terminated signal like in task 1. Therefore, some functions needed to be implemented first.

         _WEXITSTATUS(), _WTERMSIG(), _WSTOPSIG(),

         _WIFEXITED(), _WIFSIGNALED(), _WIFSTOPPED()

         These functions are like the macro features in linux kernel, that are used to generate signals that are same in the task 1, just for convenience. The realization of these functions are the same in /usr/include/i386-linux-gne/bits/waitstatus.h

c. With all the functions above implemented, firstly use my_fork() to generate a child process to run my_exec() function. Then it will use my_wait() to get the terminated signal of the child process and print out corresponding information.

c. Environment

version of kernel: 5.10.30

d. Execting the program

cd to the directory /program2

type “sudo su” to log in the root account

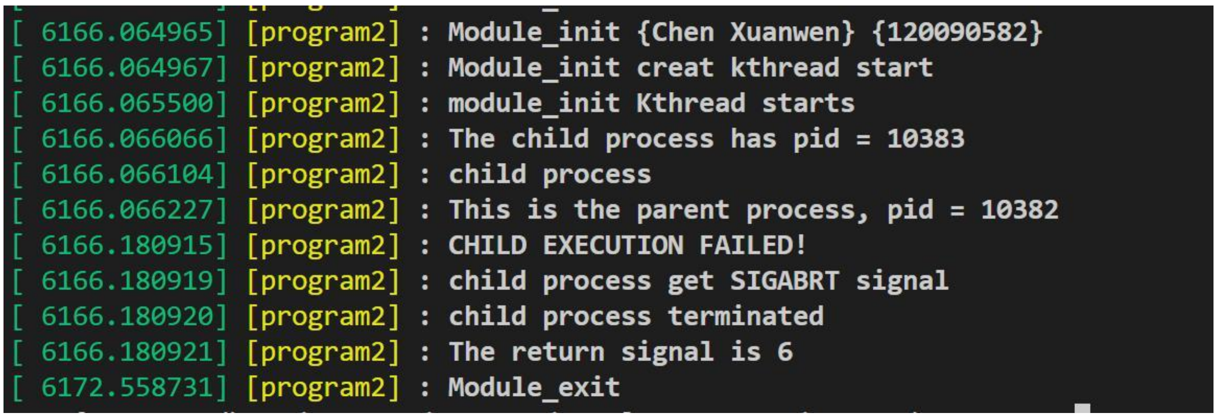
type “make clean”, “make”, “insmod program2.ko”, “rmmod program2” in the terminal to make files, insert program2.ko module, and remove the module.

Use “gcc -o test test.c” to compile the test file.

Type “dmesg” to display the corresponding message.

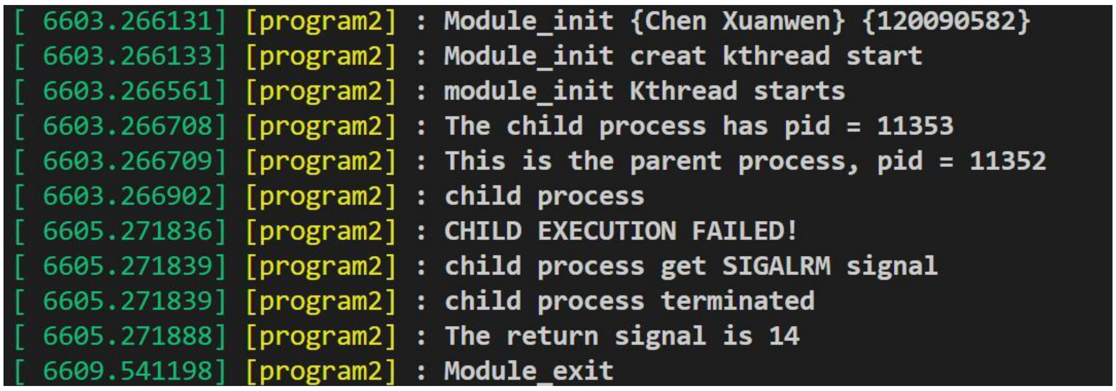
e. Sample outputs

1. abort



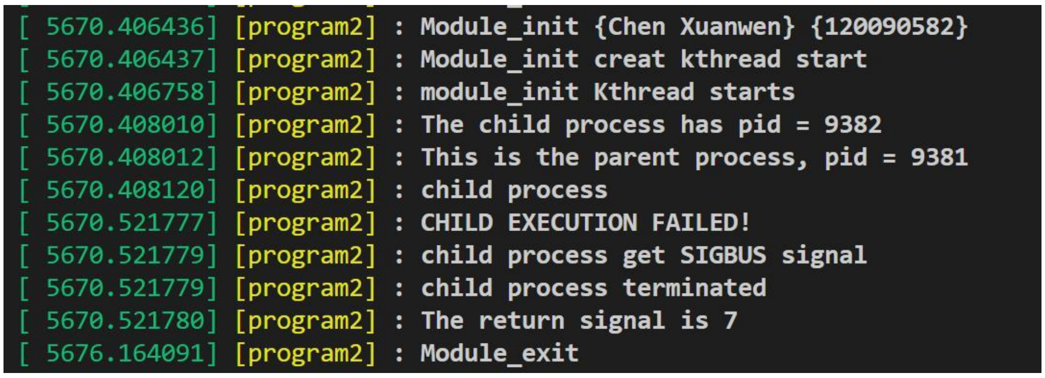
```
[ 6166.064965] [program2] : Module_init {Chen Xuanwen} {120090582}
[ 6166.064967] [program2] : Module_init creat kthread start
[ 6166.065500] [program2] : module_init Kthread starts
[ 6166.066066] [program2] : The child process has pid = 10383
[ 6166.066104] [program2] : child process
[ 6166.066227] [program2] : This is the parent process, pid = 10382
[ 6166.180915] [program2] : CHILD EXECUTION FAILED!
[ 6166.180919] [program2] : child process get SIGABRT signal
[ 6166.180920] [program2] : child process terminated
[ 6166.180921] [program2] : The return signal is 6
[ 6172.558731] [program2] : Module_exit
```

2.alarm



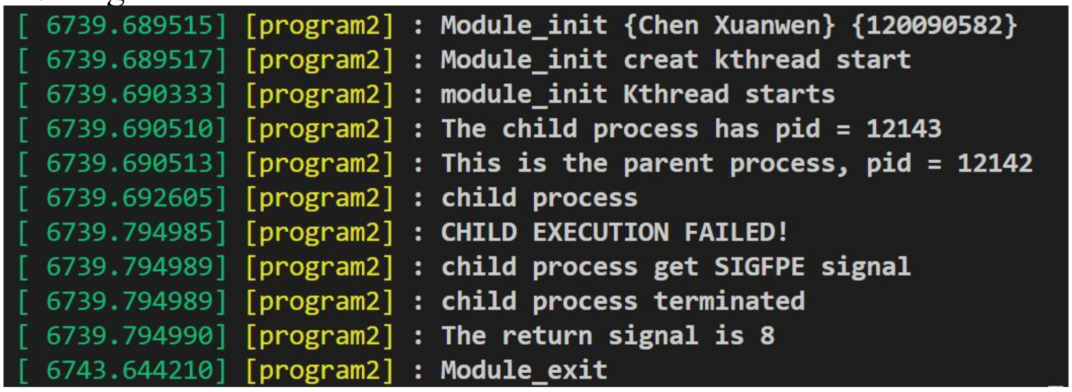
```
[ 6603.266131] [program2] : Module_init {Chen Xuanwen} {120090582}
[ 6603.266133] [program2] : Module_init creat kthread start
[ 6603.266561] [program2] : module_init Kthread starts
[ 6603.266708] [program2] : The child process has pid = 11353
[ 6603.266709] [program2] : This is the parent process, pid = 11352
[ 6603.266902] [program2] : child process
[ 6605.271836] [program2] : CHILD EXECUTION FAILED!
[ 6605.271839] [program2] : child process get SIGALRM signal
[ 6605.271839] [program2] : child process terminated
[ 6605.271888] [program2] : The return signal is 14
[ 6609.541198] [program2] : Module_exit
```

3. bus



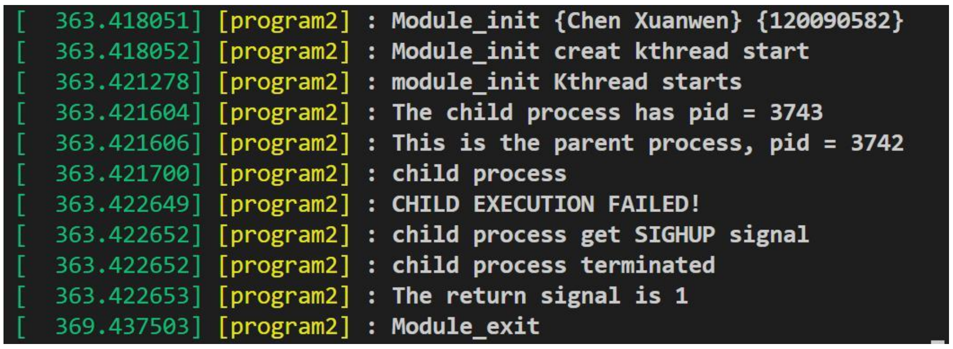
```
[ 5670.406436] [program2] : Module_init {Chen Xuanwen} {120090582}
[ 5670.406437] [program2] : Module_init creat kthread start
[ 5670.406758] [program2] : module_init Kthread starts
[ 5670.408010] [program2] : The child process has pid = 9382
[ 5670.408012] [program2] : This is the parent process, pid = 9381
[ 5670.408120] [program2] : child process
[ 5670.521777] [program2] : CHILD EXECUTION FAILED!
[ 5670.521779] [program2] : child process get SIGBUS signal
[ 5670.521779] [program2] : child process terminated
[ 5670.521780] [program2] : The return signal is 7
[ 5676.164091] [program2] : Module_exit
```

4. floating



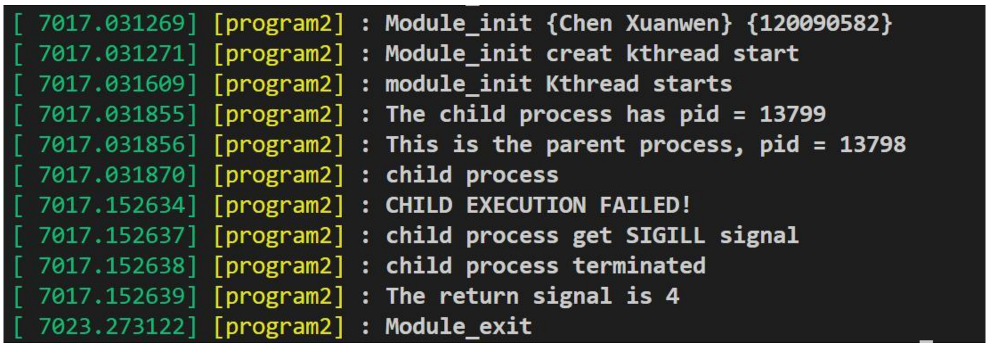
```
[ 6739.689515] [program2] : Module_init {Chen Xuanwen} {120090582}
[ 6739.689517] [program2] : Module_init creat kthread start
[ 6739.690333] [program2] : module_init Kthread starts
[ 6739.690510] [program2] : The child process has pid = 12143
[ 6739.690513] [program2] : This is the parent process, pid = 12142
[ 6739.692605] [program2] : child process
[ 6739.794985] [program2] : CHILD EXECUTION FAILED!
[ 6739.794989] [program2] : child process get SIGFPE signal
[ 6739.794989] [program2] : child process terminated
[ 6739.794990] [program2] : The return signal is 8
[ 6743.644210] [program2] : Module_exit
```

5. hangup



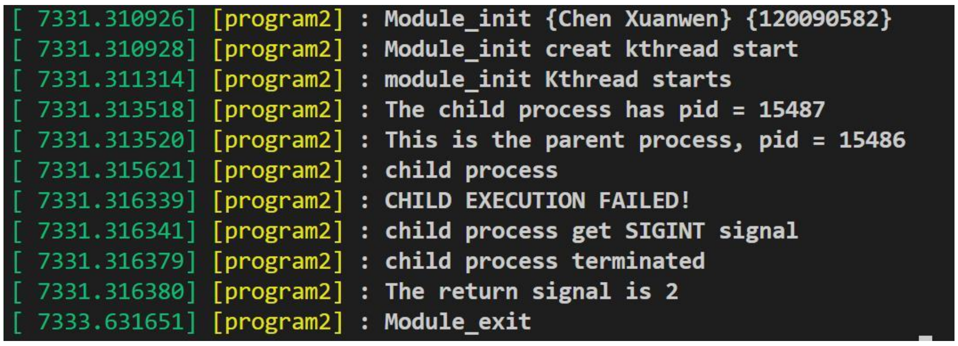
```
[  363.418051] [program2] : Module_init {Chen Xuanwen} {120090582}
[  363.418052] [program2] : Module_init creat kthread start
[  363.421278] [program2] : module_init Kthread starts
[  363.421604] [program2] : The child process has pid = 3743
[  363.421606] [program2] : This is the parent process, pid = 3742
[  363.421700] [program2] : child process
[  363.422649] [program2] : CHILD EXECUTION FAILED!
[  363.422652] [program2] : child process get SIGHUP signal
[  363.422652] [program2] : child process terminated
[  363.422653] [program2] : The return signal is 1
[  369.437503] [program2] : Module_exit
```

6. illegal_instr



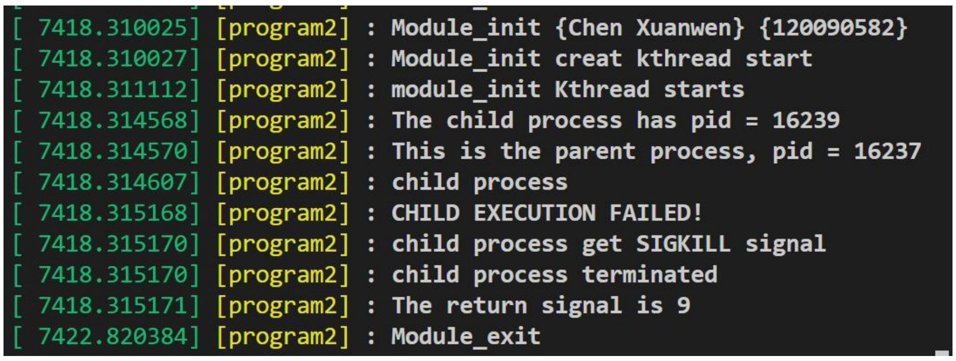
```
[ 7017.031269] [program2] : Module_init {Chen Xuanwen} {120090582}
[ 7017.031271] [program2] : Module_init creat kthread start
[ 7017.031609] [program2] : module_init Kthread starts
[ 7017.031855] [program2] : The child process has pid = 13799
[ 7017.031856] [program2] : This is the parent process, pid = 13798
[ 7017.031870] [program2] : child process
[ 7017.152634] [program2] : CHILD EXECUTION FAILED!
[ 7017.152637] [program2] : child process get SIGILL signal
[ 7017.152638] [program2] : child process terminated
[ 7017.152639] [program2] : The return signal is 4
[ 7023.273122] [program2] : Module_exit
```

7. interrupt



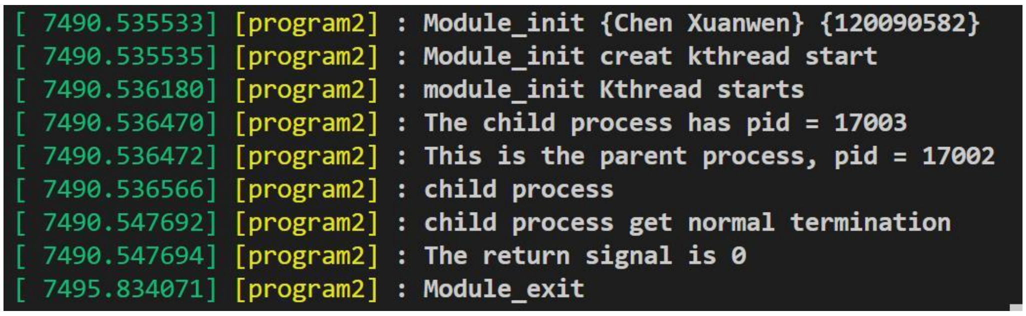
```
[ 7331.310926] [program2] : Module_init {Chen Xuanwen} {120090582}
[ 7331.310928] [program2] : Module_init creat kthread start
[ 7331.311314] [program2] : module_init Kthread starts
[ 7331.313518] [program2] : The child process has pid = 15487
[ 7331.313520] [program2] : This is the parent process, pid = 15486
[ 7331.315621] [program2] : child process
[ 7331.316339] [program2] : CHILD EXECUTION FAILED!
[ 7331.316341] [program2] : child process get SIGINT signal
[ 7331.316379] [program2] : child process terminated
[ 7331.316380] [program2] : The return signal is 2
[ 7333.631651] [program2] : Module_exit
```

8. kill

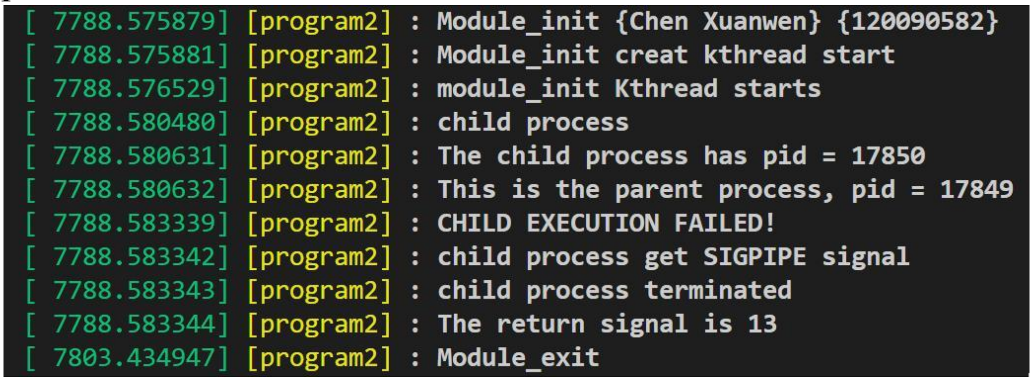


```
[ 7418.310025] [program2] : Module_init {Chen Xuanwen} {120090582}
[ 7418.310027] [program2] : Module_init creat kthread start
[ 7418.311112] [program2] : module_init Kthread starts
[ 7418.314568] [program2] : The child process has pid = 16239
[ 7418.314570] [program2] : This is the parent process, pid = 16237
[ 7418.314607] [program2] : child process
[ 7418.315168] [program2] : CHILD EXECUTION FAILED!
[ 7418.315170] [program2] : child process get SIGKILL signal
[ 7418.315170] [program2] : child process terminated
[ 7418.315171] [program2] : The return signal is 9
[ 7422.820384] [program2] : Module_exit
```

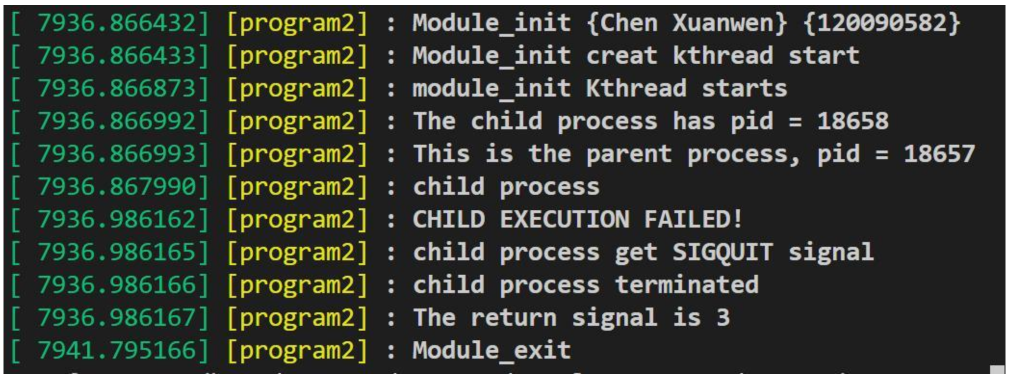
9. normal


```
[ 7490.535533] [program2] : Module_init {Chen Xuanwen} {120090582}
[ 7490.535535] [program2] : Module_init creat kthread start
[ 7490.536180] [program2] : module_init Kthread starts
[ 7490.536470] [program2] : The child process has pid = 17003
[ 7490.536472] [program2] : This is the parent process, pid = 17002
[ 7490.536566] [program2] : child process
[ 7490.547692] [program2] : child process get normal termination
[ 7490.547694] [program2] : The return signal is 0
[ 7495.834071] [program2] : Module_exit
```

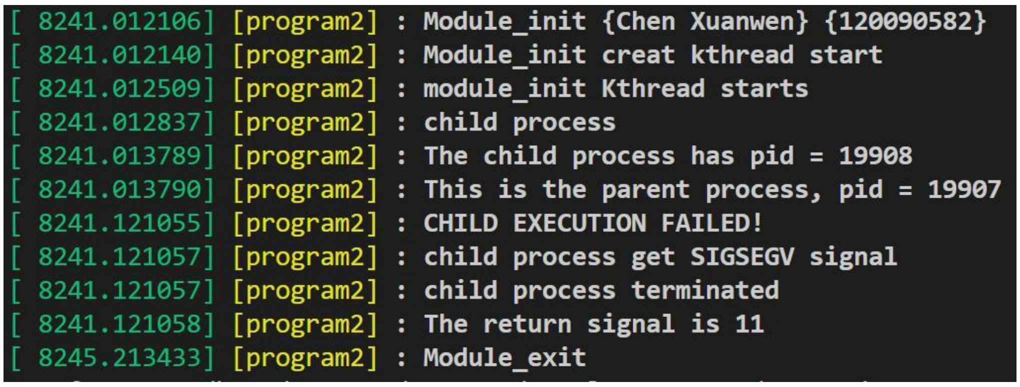
10.pipe


```
[ 7788.575879] [program2] : Module_init {Chen Xuanwen} {120090582}
[ 7788.575881] [program2] : Module_init creat kthread start
[ 7788.576529] [program2] : module_init Kthread starts
[ 7788.580480] [program2] : child process
[ 7788.580631] [program2] : The child process has pid = 17850
[ 7788.580632] [program2] : This is the parent process, pid = 17849
[ 7788.583339] [program2] : CHILD EXECUTION FAILED!
[ 7788.583342] [program2] : child process get SIGPIPE signal
[ 7788.583343] [program2] : child process terminated
[ 7788.583344] [program2] : The return signal is 13
[ 7803.434947] [program2] : Module_exit
```

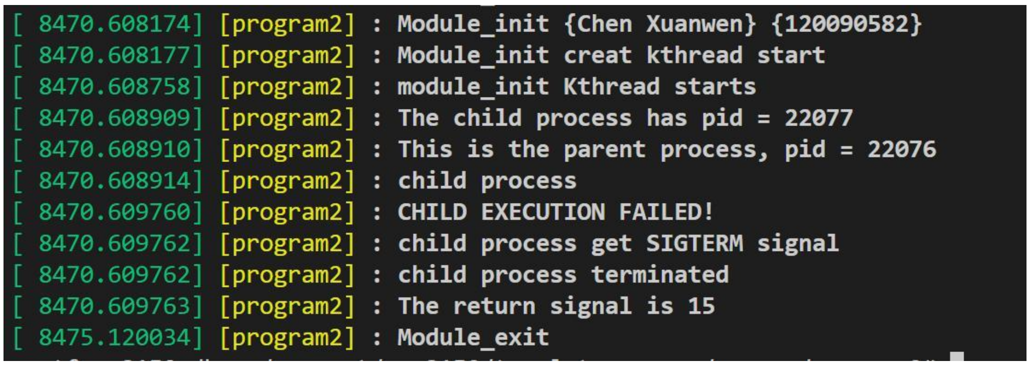
11.quit


```
[ 7936.866432] [program2] : Module_init {Chen Xuanwen} {120090582}
[ 7936.866433] [program2] : Module_init creat kthread start
[ 7936.866873] [program2] : module_init Kthread starts
[ 7936.866992] [program2] : The child process has pid = 18658
[ 7936.866993] [program2] : This is the parent process, pid = 18657
[ 7936.867990] [program2] : child process
[ 7936.986162] [program2] : CHILD EXECUTION FAILED!
[ 7936.986165] [program2] : child process get SIGQUIT signal
[ 7936.986166] [program2] : child process terminated
[ 7936.986167] [program2] : The return signal is 3
[ 7941.795166] [program2] : Module_exit
```

12.segment_fault

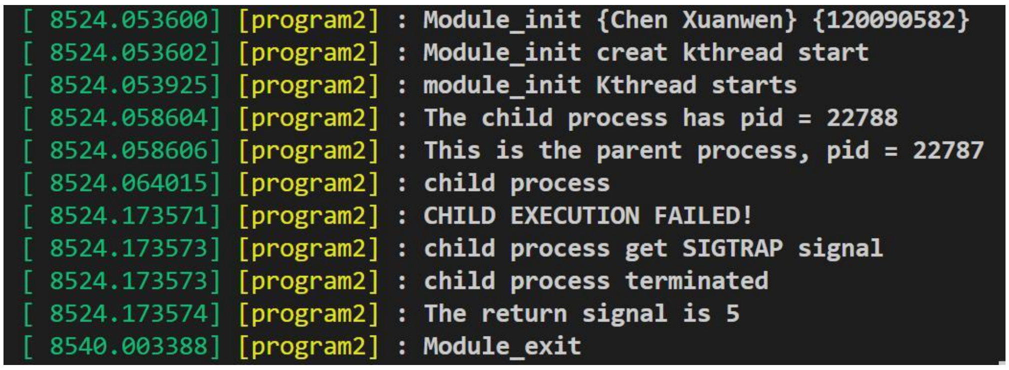

```
[ 8241.012106] [program2] : Module_init {Chen Xuanwen} {120090582}
[ 8241.012140] [program2] : Module_init creat kthread start
[ 8241.012509] [program2] : module_init Kthread starts
[ 8241.012837] [program2] : child process
[ 8241.013789] [program2] : The child process has pid = 19908
[ 8241.013790] [program2] : This is the parent process, pid = 19907
[ 8241.121055] [program2] : CHILD EXECUTION FAILED!
[ 8241.121057] [program2] : child process get SIGSEGV signal
[ 8241.121057] [program2] : child process terminated
[ 8241.121058] [program2] : The return signal is 11
[ 8245.213433] [program2] : Module_exit
```

13.terminate



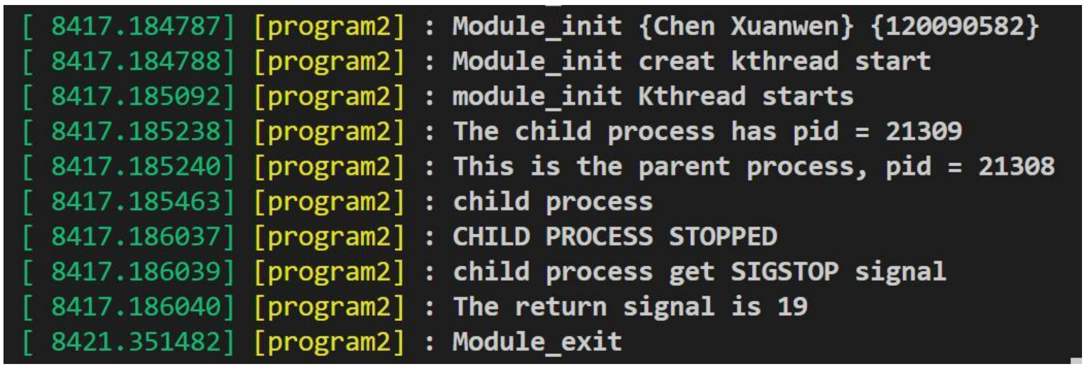
```
[ 8470.608174] [program2] : Module_init {Chen Xuanwen} {120090582}
[ 8470.608177] [program2] : Module_init creat kthread start
[ 8470.608758] [program2] : module_init Kthread starts
[ 8470.608909] [program2] : The child process has pid = 22077
[ 8470.608910] [program2] : This is the parent process, pid = 22076
[ 8470.608914] [program2] : child process
[ 8470.609760] [program2] : CHILD EXECUTION FAILED!
[ 8470.609762] [program2] : child process get SIGTERM signal
[ 8470.609762] [program2] : child process terminated
[ 8470.609763] [program2] : The return signal is 15
[ 8475.120034] [program2] : Module_exit
```

14.trap



```
[ 8524.053600] [program2] : Module_init {Chen Xuanwen} {120090582}
[ 8524.053602] [program2] : Module_init creat kthread start
[ 8524.053925] [program2] : module_init Kthread starts
[ 8524.058604] [program2] : The child process has pid = 22788
[ 8524.058606] [program2] : This is the parent process, pid = 22787
[ 8524.064015] [program2] : child process
[ 8524.173571] [program2] : CHILD EXECUTION FAILED!
[ 8524.173573] [program2] : child process get SIGTRAP signal
[ 8524.173573] [program2] : child process terminated
[ 8524.173574] [program2] : The return signal is 5
[ 8540.003388] [program2] : Module_exit
```

15.stop



```
[ 8417.184787] [program2] : Module_init {Chen Xuanwen} {120090582}
[ 8417.184788] [program2] : Module_init creat kthread start
[ 8417.185092] [program2] : module_init Kthread starts
[ 8417.185238] [program2] : The child process has pid = 21309
[ 8417.185240] [program2] : This is the parent process, pid = 21308
[ 8417.185463] [program2] : child process
[ 8417.186037] [program2] : CHILD PROCESS STOPPED
[ 8417.186039] [program2] : child process get SIGSTOP signal
[ 8417.186040] [program2] : The return signal is 19
[ 8421.351482] [program2] : Module_exit
```

f. What I learnt from the project

In this project, I get more understanding of the LKM, about what it is and how it works. I also leant how to modify the kernel file and how to recompile it. And how to insert and remove the loadable kernel modules.

3. bonus

This part not down yet.