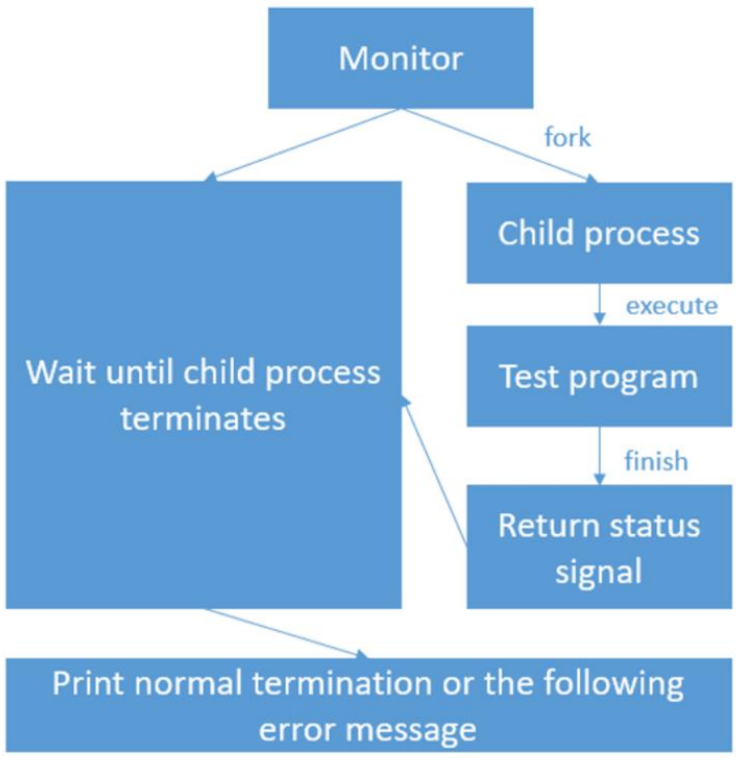
Assignment1 Report

Chen Qixu (120090643)

1. How to design the program

This project of Assignment1 includes two parts: task 1 and task2.

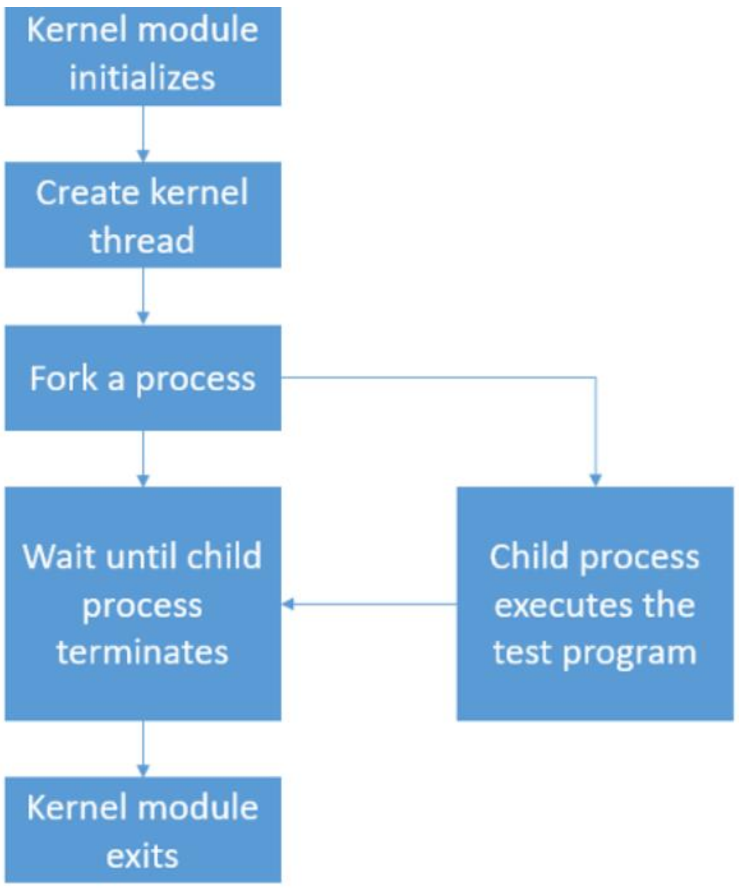
For task1, the main flow chart is:


Monitor
fork
Child process
execute
Test program
finish
Return status signal
Wait until child process terminates
Print normal termination or the following error message

This program includes the following process:

1. Fork a process with the fork () function. The fork will return the pid of the parent process and the child process.
2. Then the program will use the pid to identify the process: If the pid is negative, then fork error occurs in the program. If pid equals to 0, the process is child process. Else.it is the parent process.
3. In the child process, the program will execute the given file.
4. The parent process will wait for the child process to finish and return the status signal. The function will wait for the signal with the function: waitpid (-1, &status, WUNTRACED). The first argument "-1" means the function will wait for any child process created. WUNTRACED allows your parent to be returned from waitpid () if a child gets stopped as well as exiting or being killed. The parent process will finally print normal termination or error message.

For task2, the flow chart is


Kernel module initializes
Create kernel thread
Fork a process
Wait until child process terminates
Child process executes the test program
Kernel module exits

This program includes the following process:

0. To use the API from the kernel, we have to go into the kernel file to export some functions. Under the lines where kernel_clone, do_execve, do_wait, and getname_kernel are defined, we write the EXPORT_SYMBOL () to export these functions. (They are in file: /kernel/fork.c, /fs/exec.c, /kernel/exit.c, /fs/namei.c.) After recompiling the kernel file, we can start writing program2.c.
1. Module_init calls program2_ init (). The initialization code will then call kthread_create() and supply the parameter to my_fork. After wake_up_process (), the program will go to my_fork().
2. In my_fork(), the program will call the function kernel_clone to start a child process.The argument for kernel_clone is a special struct called kernel_clone_args. In this struct , the exit_signal and exit_flag is set to SIGCHLD;
3. The parent process will wait for the child process to terminate using the function my_wait(). In my_wait, the function will capture the return signal from the child process.
4. For my_wait, we have to implement on a struct called wait_opts,which requires lots of complex variable. The most important value is wo.wo_stat, it contains the return value of child process after do_wait(wo).This value is set to be 0 initially.
5. In the child process, it will execute the function my_exec()(which has been included in the .stack of the kernel_clone_args.). Inside the function, a const char storing the path of the test file is defined. The function will then call the

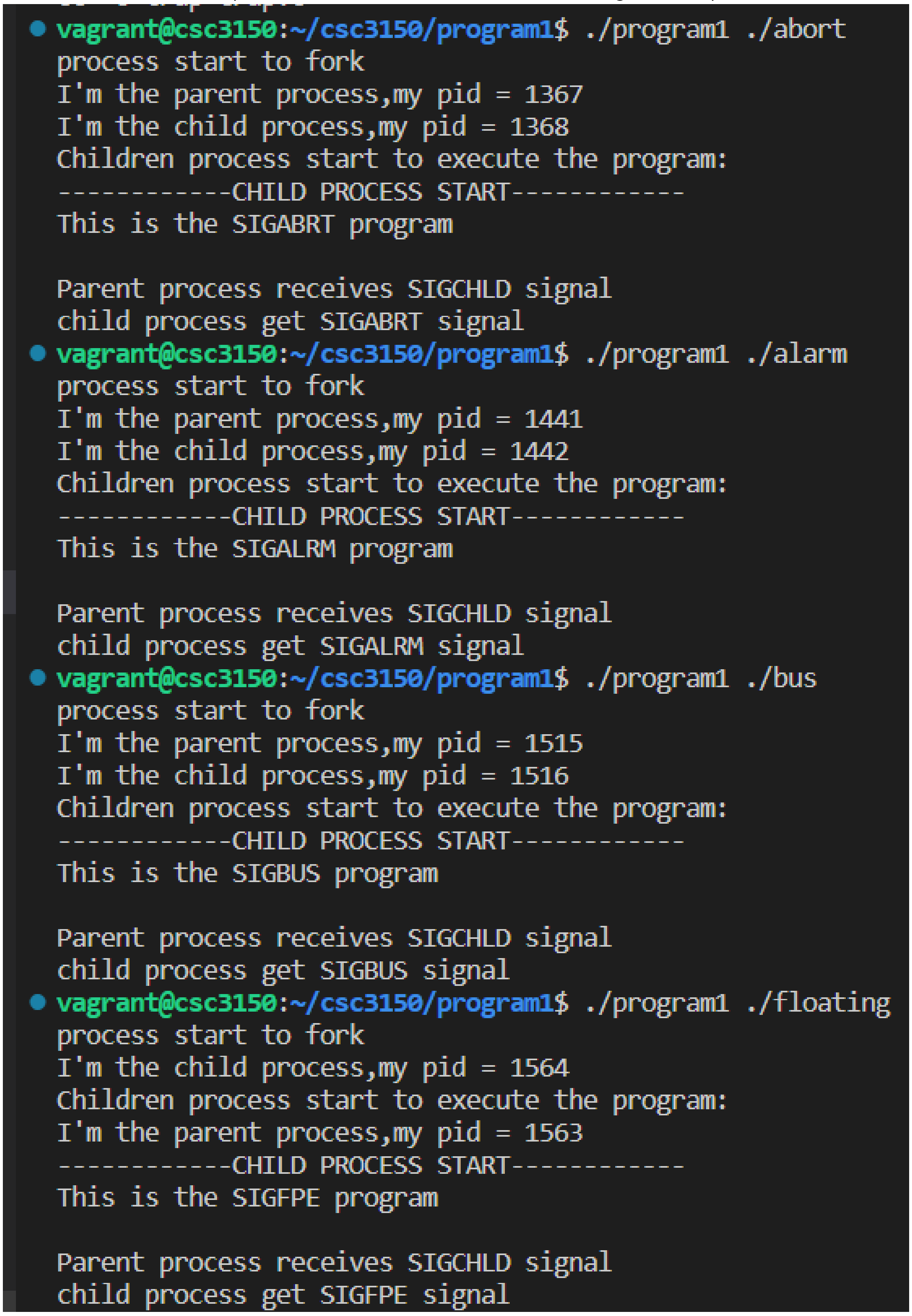
“do_execve” function to execute the test file.

6. After the parent process has received the signal returned from the child process(returned from the my_wait() function), the parent process use printk() to write all information acquired from the status information from the child process

2. How to set up your development environment and test the program
   1. How to set up the development environment.
      a. Install vitrualbox and vagrant. After installation, follow the instructions in the tutorial to set up the virtual machine
      b. I downloaded the kernel file whose version is Linux 5-10.146, Extract the source file to the directory /home/seed/work. Unpack the kernel file using $sudo tar xvf linux-5.10.146.tar.xz.
      c. How to compile the kernel:
         i. Copy config from /boot to /home/seed/work/ linux-5.10.146
         ii. Login root account and go to kernel source directory and then input the following instructions in the terminal: make mrproper; Make clean;make menuconfig;
         iii. Save the config and exit
         iv. Build kernel Image and modules by inputting the following instructions: $make bzImage -j$(nproc); make modules -j$(nproc);make modules install; make install
         v. Reboot to use the new kernel
   2. How to test task1
      a. Enter the terminal: input “Make” to make all the files.
      b. Enter the name of the file you want to test. For example, of you want to test the abort.c, input ./program1 ./abort in the terminal.
   3. How to test task2
      a. Enter the terminal: input “Make clean” then “make”
      b. Input the following instructions:
         i. gcc test.c -o test (other C files are the same)
         ii. cd file_path
         iii. sudo make
         iv. sudo insmod program2.ko
         v. sudo rmmod program2
         vi. dmesg
      c. you will see the output in the terminal.

4. Output of program1

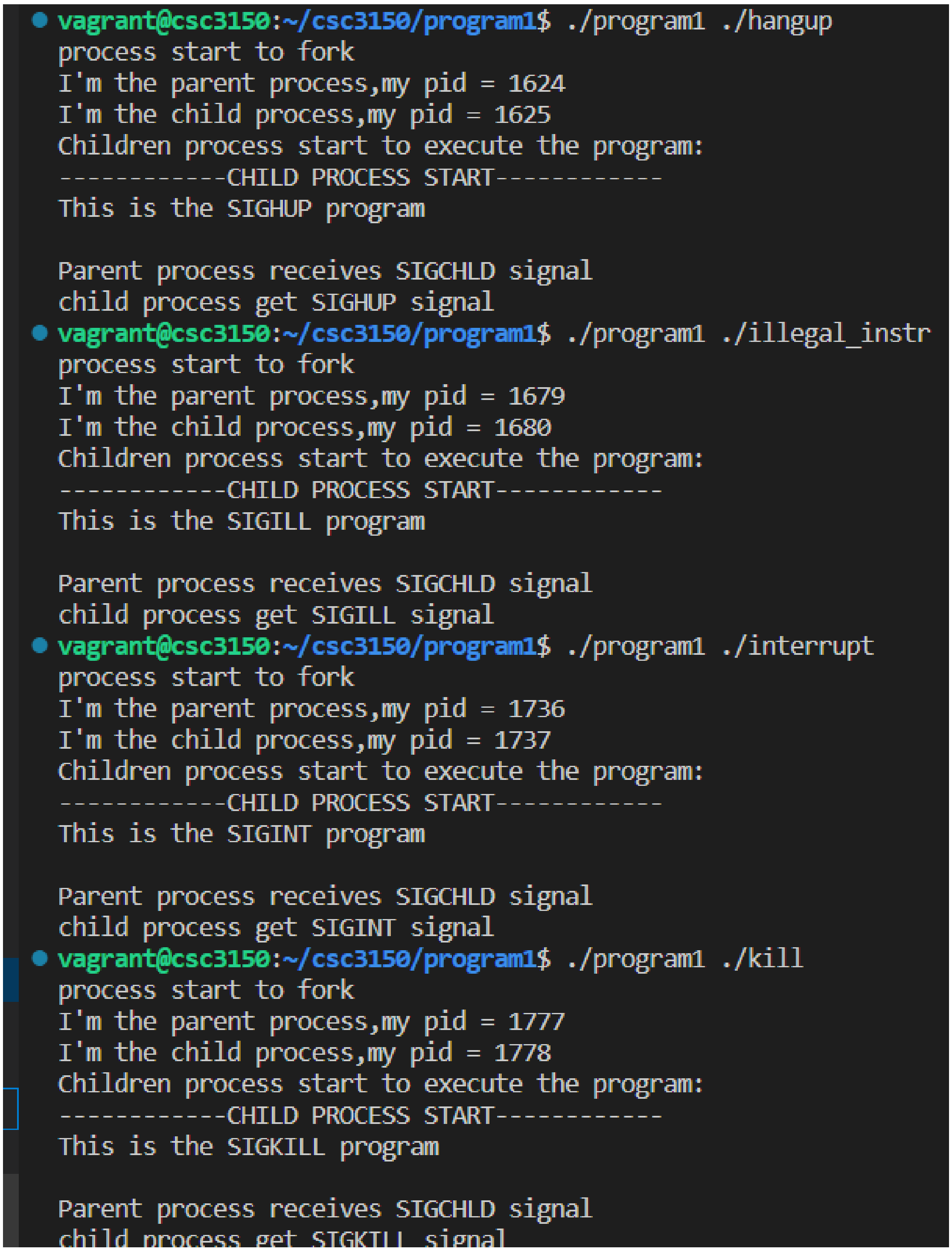
You will see the output of the abort.c, alarm.c,bus.c, floating.c in the picture



```
vagrant@csc3150:~/csc3150/program1$ ./program1 ./abort
process start to fork
I'm the parent process,my pid = 1367
I'm the child process,my pid = 1368
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGABRT program

Parent process receives SIGCHLD signal
child process get SIGABRT signal
vagrant@csc3150:~/csc3150/program1$ ./program1 ./alarm
process start to fork
I'm the parent process,my pid = 1441
I'm the child process,my pid = 1442
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGALRM program

Parent process receives SIGCHLD signal
child process get SIGALRM signal
vagrant@csc3150:~/csc3150/program1$ ./program1 ./bus
process start to fork
I'm the parent process,my pid = 1515
I'm the child process,my pid = 1516
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGBUS program

Parent process receives SIGCHLD signal
child process get SIGBUS signal
vagrant@csc3150:~/csc3150/program1$ ./program1 ./floating
process start to fork
I'm the child process,my pid = 1564
Children process start to execute the program:
I'm the parent process,my pid = 1563
------------CHILD PROCESS START------------
This is the SIGFPE program

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

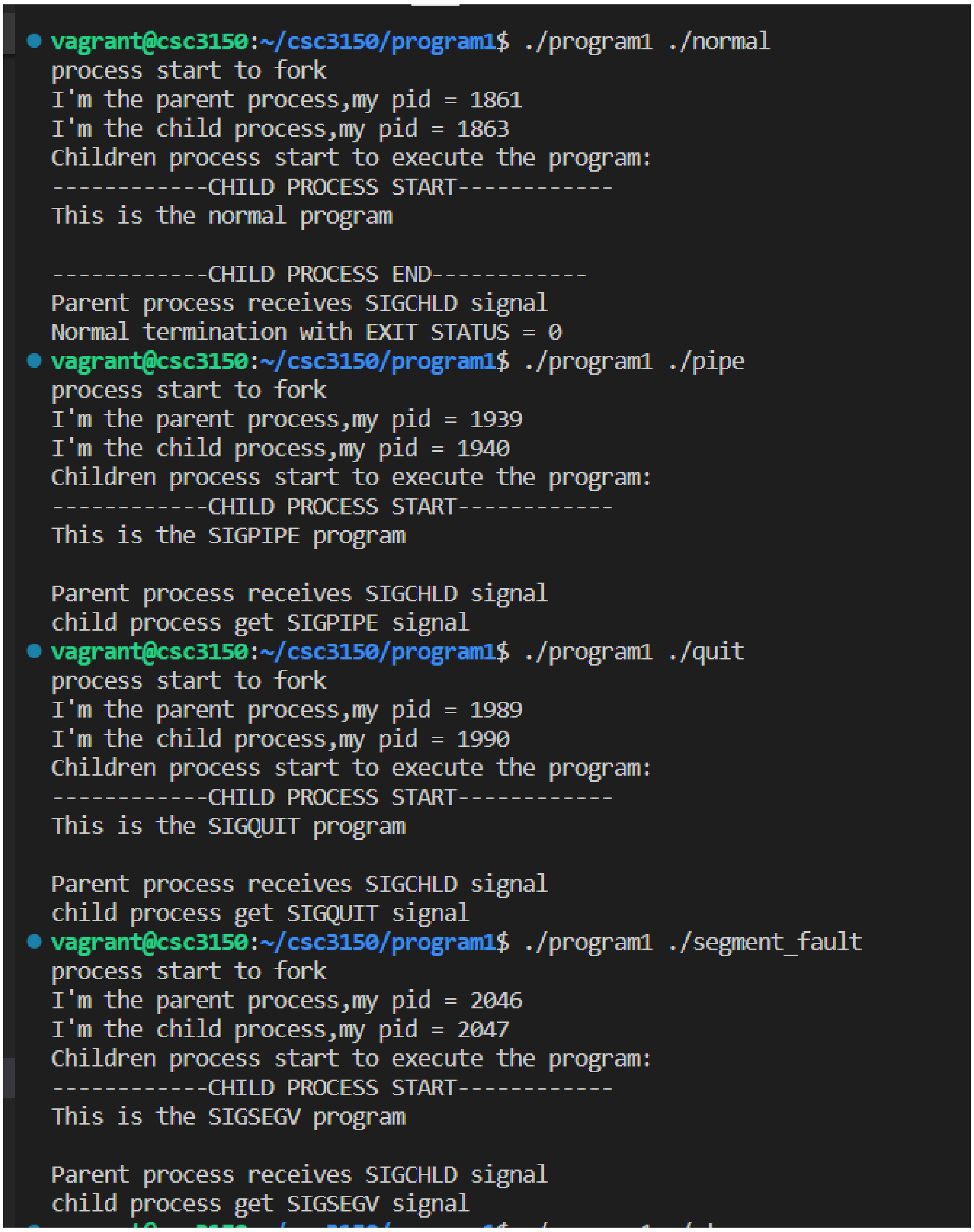
vi.

You will see the output of
Hangup.c, illegal_instr.c,interrupt.c, kill.c



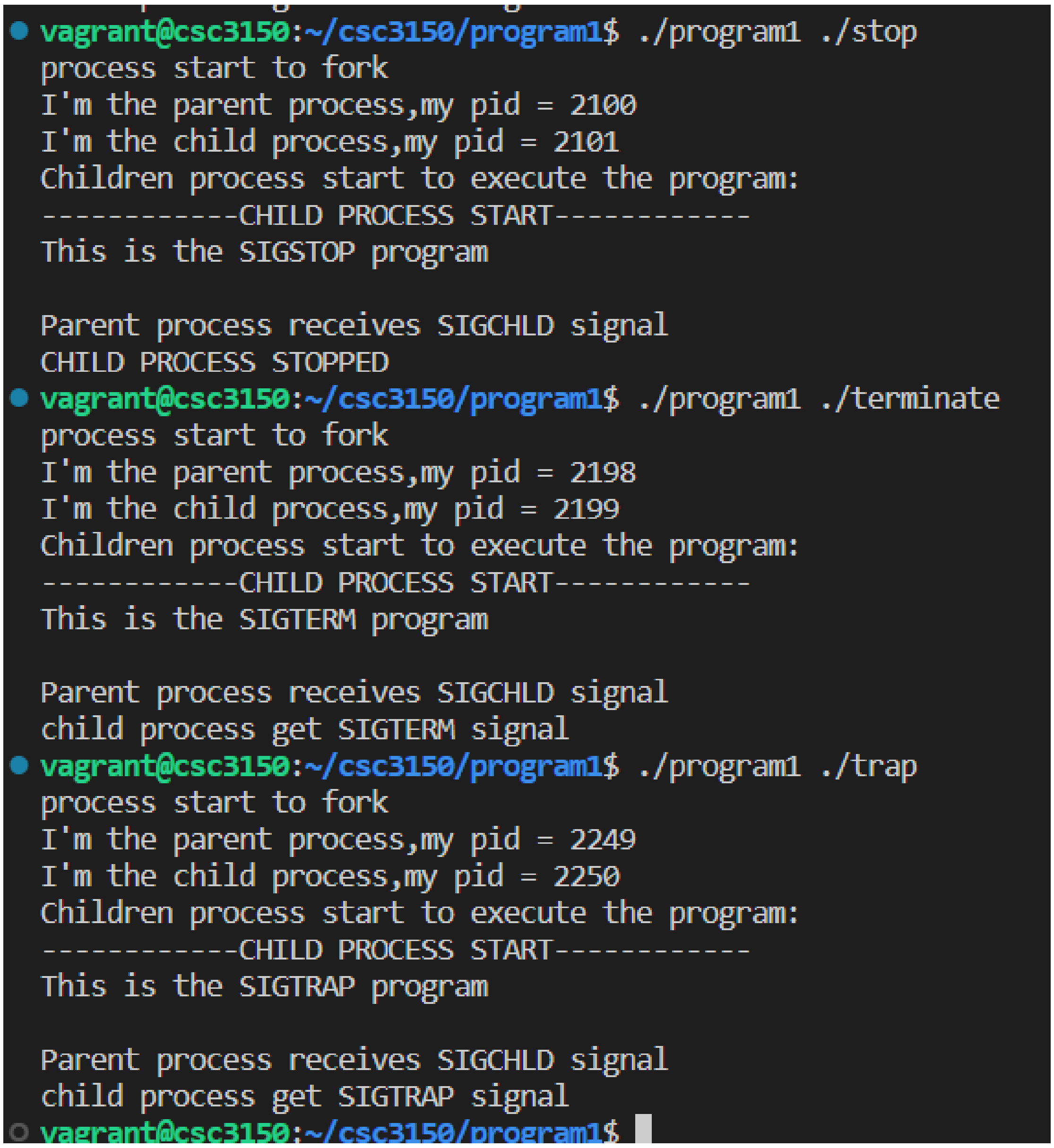
```
vagrant@csc3150:~/csc3150/program1$ ./program1 ./hangup
process start to fork
I'm the parent process,my pid = 1624
I'm the child process,my pid = 1625
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGHUP program

Parent process receives SIGCHLD signal
child process get SIGHUP signal
vagrant@csc3150:~/csc3150/program1$ ./program1 ./illegal_instr
process start to fork
I'm the parent process,my pid = 1679
I'm the child process,my pid = 1680
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGILL program

Parent process receives SIGCHLD signal
child process get SIGILL signal
vagrant@csc3150:~/csc3150/program1$ ./program1 ./interrupt
process start to fork
I'm the parent process,my pid = 1736
I'm the child process,my pid = 1737
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGINT program

Parent process receives SIGCHLD signal
child process get SIGINT signal
vagrant@csc3150:~/csc3150/program1$ ./program1 ./kill
process start to fork
I'm the parent process,my pid = 1777
I'm the child process,my pid = 1778
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGKILL program

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

You will see the output of
normal.c pipe.c quit.c segment_fault.c



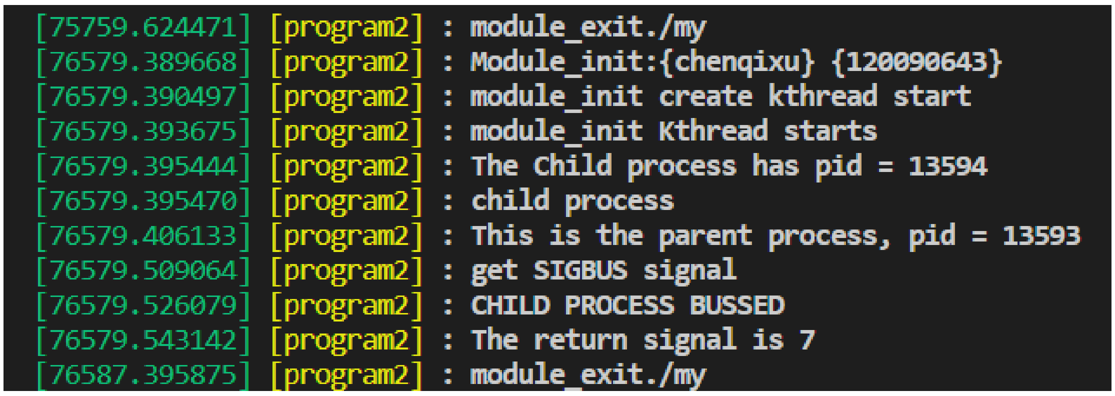
```
vagrant@csc3150:~/csc3150/program1$ ./program1 ./normal
process start to fork
I'm the parent process,my pid = 1861
I'm the child process,my pid = 1863
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the normal program

------------CHILD PROCESS END------------
Parent process receives SIGCHLD signal
Normal termination with EXIT STATUS = 0
vagrant@csc3150:~/csc3150/program1$ ./program1 ./pipe
process start to fork
I'm the parent process,my pid = 1939
I'm the child process,my pid = 1940
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGPIPE program

Parent process receives SIGCHLD signal
child process get SIGPIPE signal
vagrant@csc3150:~/csc3150/program1$ ./program1 ./quit
process start to fork
I'm the parent process,my pid = 1989
I'm the child process,my pid = 1990
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

Parent process receives SIGCHLD signal
child process get SIGQUIT signal
vagrant@csc3150:~/csc3150/program1$ ./program1 ./segment_fault
process start to fork
I'm the parent process,my pid = 2046
I'm the child process,my pid = 2047
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

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

You will see the output of
stop.c terminate.c trap.c



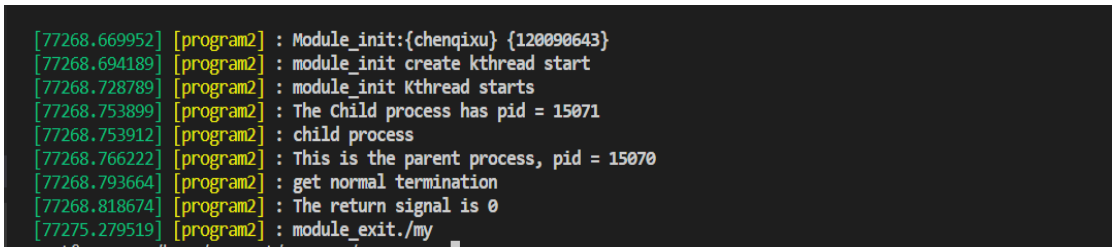
```
vagrant@csc3150:~/csc3150/program1$ ./program1 ./stop
process start to fork
I'm the parent process,my pid = 2100
I'm the child process,my pid = 2101
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

Parent process receives SIGCHLD signal
CHILD PROCESS STOPPED
vagrant@csc3150:~/csc3150/program1$ ./program1 ./terminate
process start to fork
I'm the parent process,my pid = 2198
I'm the child process,my pid = 2199
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGTERM program

Parent process receives SIGCHLD signal
child process get SIGTERM signal
vagrant@csc3150:~/csc3150/program1$ ./program1 ./trap
process start to fork
I'm the parent process,my pid = 2249
I'm the child process,my pid = 2250
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGTRAP program

Parent process receives SIGCHLD signal
child process get SIGTRAP signal
vagrant@csc3150:~/csc3150/program1$
```

5. Output of program2
   For the test.c provided(raise SIGBUS)



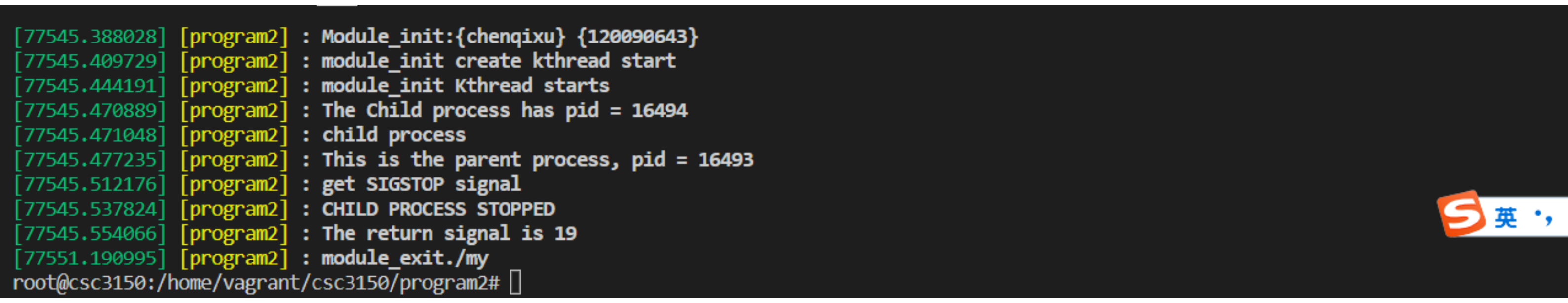
```
[75759.624471] [program2] : module_exit./my
[76579.389668] [program2] : Module_init:{chenqixu} {120090643}
[76579.390497] [program2] : module_init create kthread start
[76579.393675] [program2] : module_init Kthread starts
[76579.395444] [program2] : The Child process has pid = 13594
[76579.395470] [program2] : child process
[76579.406133] [program2] : This is the parent process, pid = 13593
[76579.509064] [program2] : get SIGBUS signal
[76579.526079] [program2] : CHILD PROCESS BUSSED
[76579.543142] [program2] : The return signal is 7
[76587.395875] [program2] : module_exit./my
```

For normal termination, the program will output



```
[77268.669952] [program2] : Module_init:{chenqixu} {120090643}
[77268.694189] [program2] : module_init create kthread start
[77268.728789] [program2] : module_init Kthread starts
[77268.753899] [program2] : The Child process has pid = 15071
[77268.753912] [program2] : child process
[77268.766222] [program2] : This is the parent process, pid = 15070
[77268.793664] [program2] : get normal termination
[77268.818674] [program2] : The return signal is 0
[77275.279519] [program2] : module_exit./my
```

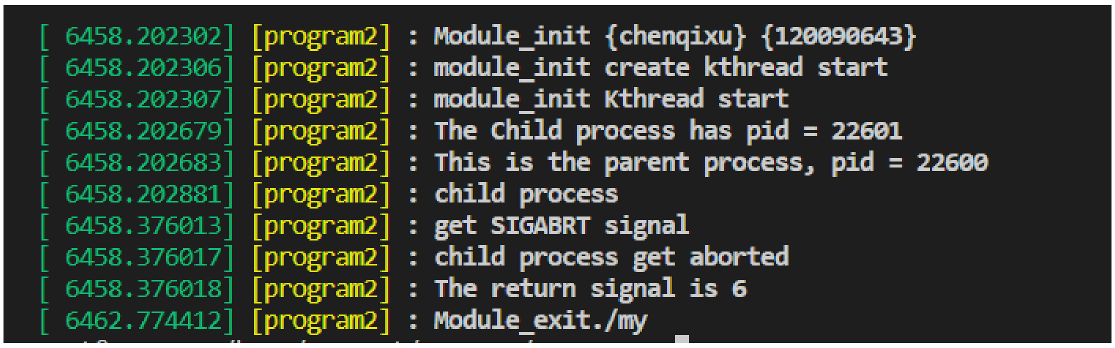
If the child process stop, the program will output:



```
[77545.388028] [program2] : Module_init:{chenqixu} {120090643}
[77545.409729] [program2] : module_init create kthread start
[77545.444191] [program2] : module_init Kthread starts
[77545.470889] [program2] : The Child process has pid = 16494
[77545.471048] [program2] : child process
[77545.477235] [program2] : This is the parent process, pid = 16493
[77545.512176] [program2] : get SIGSTOP signal
[77545.537824] [program2] : CHILD PROCESS STOPPED
[77545.554066] [program2] : The return signal is 19
[77551.190995] [program2] : module_exit./my
root@csc3150:/home/vagrant/csc3150/program2# 
```

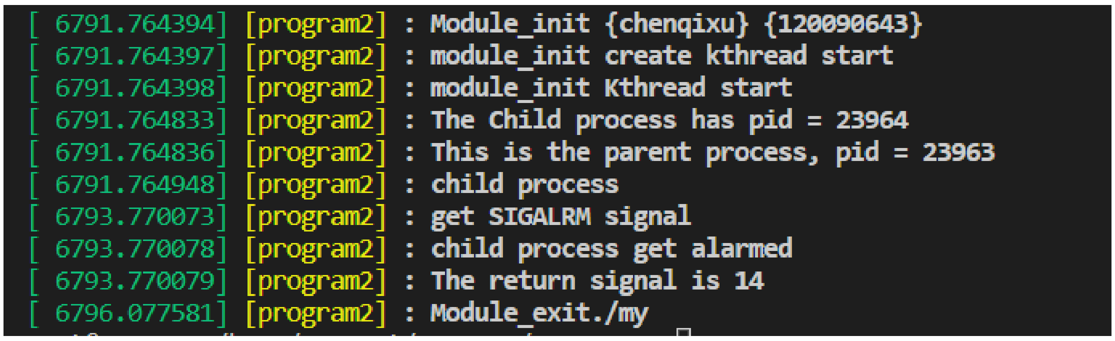
For other signal:

SIGABRT:



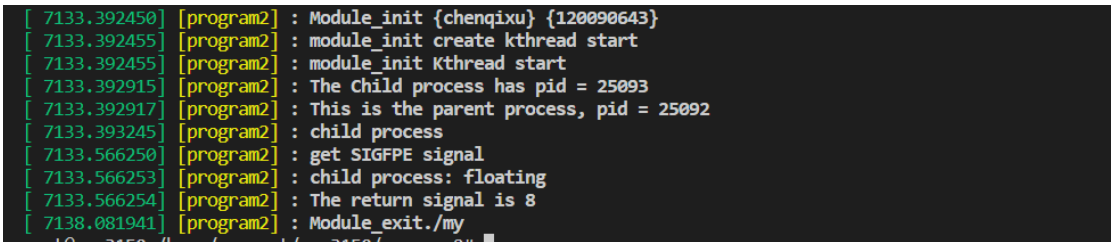
```
[ 6458.202302] [program2] : Module_init {chenqixu} {120090643}
[ 6458.202306] [program2] : module_init create kthread start
[ 6458.202307] [program2] : module_init Kthread start
[ 6458.202679] [program2] : The Child process has pid = 22601
[ 6458.202683] [program2] : This is the parent process, pid = 22600
[ 6458.202881] [program2] : child process
[ 6458.376013] [program2] : get SIGABRT signal
[ 6458.376017] [program2] : child process get aborted
[ 6458.376018] [program2] : The return signal is 6
[ 6462.774412] [program2] : Module_exit./my
```

SIGALRM:



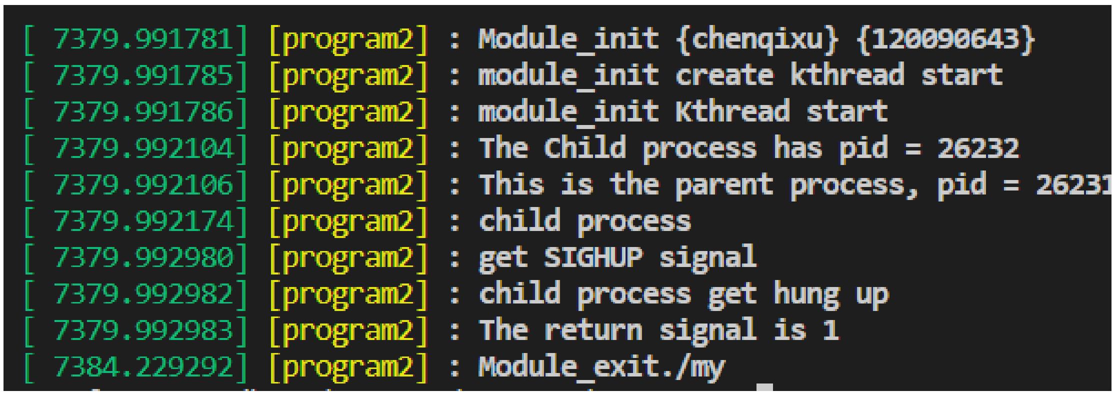
```
[ 6791.764394] [program2] : Module_init {chenqixu} {120090643}
[ 6791.764397] [program2] : module_init create kthread start
[ 6791.764398] [program2] : module_init Kthread start
[ 6791.764833] [program2] : The Child process has pid = 23964
[ 6791.764836] [program2] : This is the parent process, pid = 23963
[ 6791.764948] [program2] : child process
[ 6793.770073] [program2] : get SIGALRM signal
[ 6793.770078] [program2] : child process get alarmed
[ 6793.770079] [program2] : The return signal is 14
[ 6796.077581] [program2] : Module_exit./my
```

SIGFPE



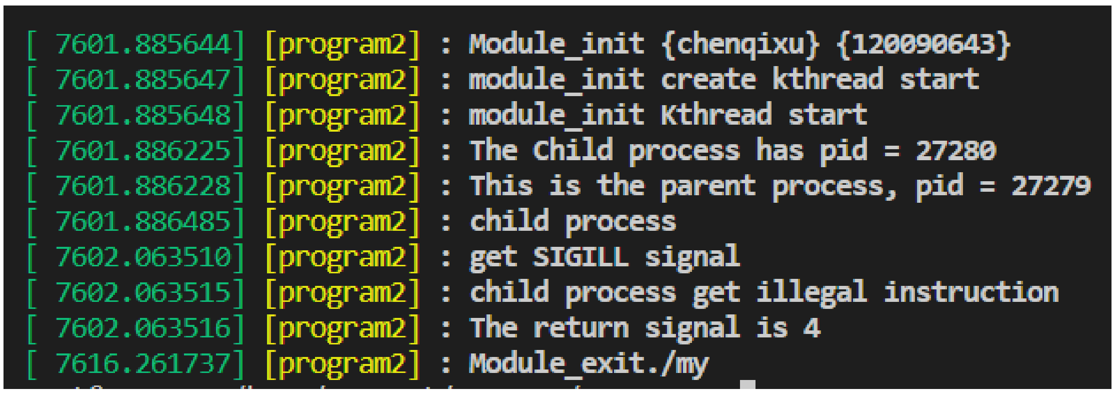
```
[ 7133.392450] [program2] : Module_init {chenqixu} {120090643}
[ 7133.392455] [program2] : module_init create kthread start
[ 7133.392455] [program2] : module_init Kthread start
[ 7133.392915] [program2] : The Child process has pid = 25093
[ 7133.392917] [program2] : This is the parent process, pid = 25092
[ 7133.393245] [program2] : child process
[ 7133.566250] [program2] : get SIGFPE signal
[ 7133.566253] [program2] : child process: floating
[ 7133.566254] [program2] : The return signal is 8
[ 7138.081941] [program2] : Module_exit./my
```

SIGHUP



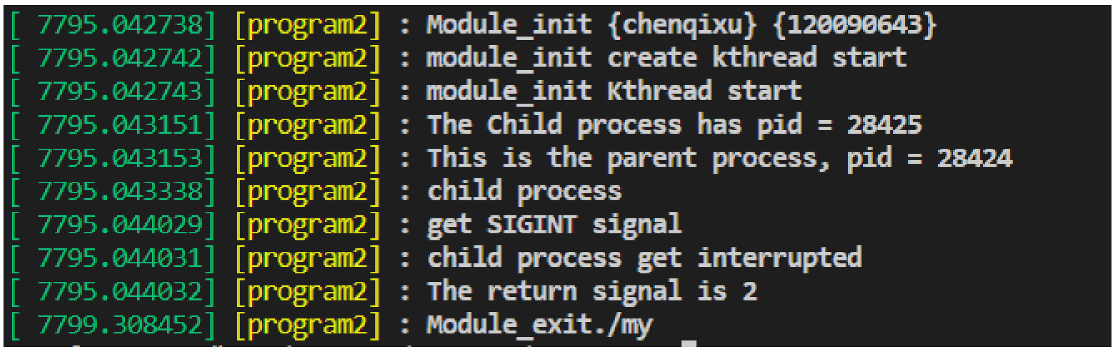
```
[ 7379.991781] [program2] : Module_init {chenqixu} {120090643}
[ 7379.991785] [program2] : module_init create kthread start
[ 7379.991786] [program2] : module_init Kthread start
[ 7379.992104] [program2] : The Child process has pid = 26232
[ 7379.992106] [program2] : This is the parent process, pid = 26231
[ 7379.992174] [program2] : child process
[ 7379.992980] [program2] : get SIGHUP signal
[ 7379.992982] [program2] : child process get hung up
[ 7379.992983] [program2] : The return signal is 1
[ 7384.229292] [program2] : Module_exit./my
```

SIGILL



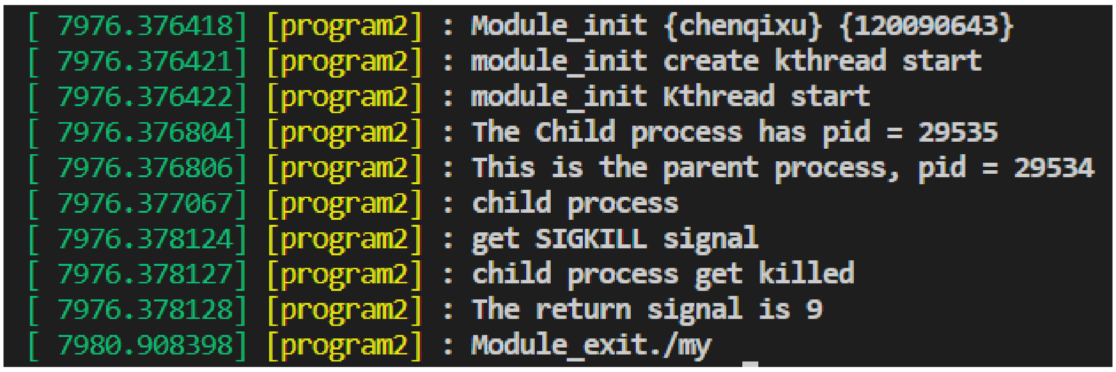
```
[ 7601.885644] [program2] : Module_init {chenqixu} {120090643}
[ 7601.885647] [program2] : module_init create kthread start
[ 7601.885648] [program2] : module_init Kthread start
[ 7601.886225] [program2] : The Child process has pid = 27280
[ 7601.886228] [program2] : This is the parent process, pid = 27279
[ 7601.886485] [program2] : child process
[ 7602.063510] [program2] : get SIGILL signal
[ 7602.063515] [program2] : child process get illegal instruction
[ 7602.063516] [program2] : The return signal is 4
[ 7616.261737] [program2] : Module_exit./my
```

SIGINT



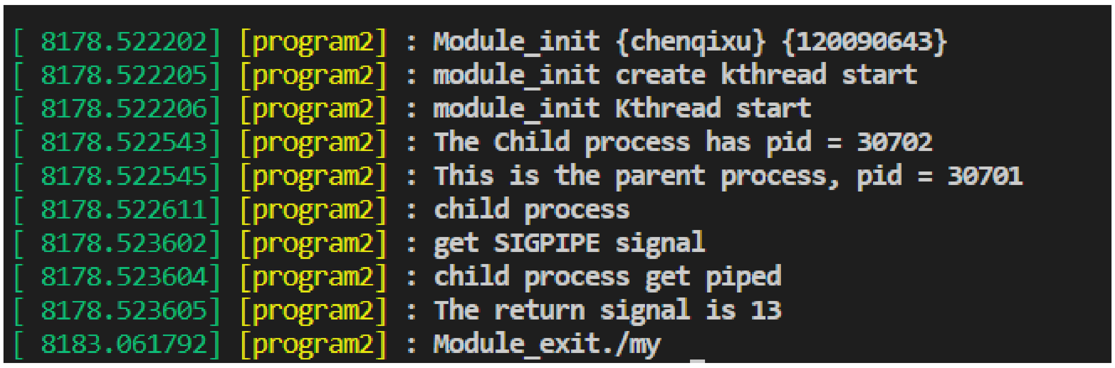
```
[ 7795.042738] [program2] : Module_init {chenqixu} {120090643}
[ 7795.042742] [program2] : module_init create kthread start
[ 7795.042743] [program2] : module_init Kthread start
[ 7795.043151] [program2] : The Child process has pid = 28425
[ 7795.043153] [program2] : This is the parent process, pid = 28424
[ 7795.043338] [program2] : child process
[ 7795.044029] [program2] : get SIGINT signal
[ 7795.044031] [program2] : child process get interrupted
[ 7795.044032] [program2] : The return signal is 2
[ 7799.308452] [program2] : Module_exit./my
```

SIGKILL



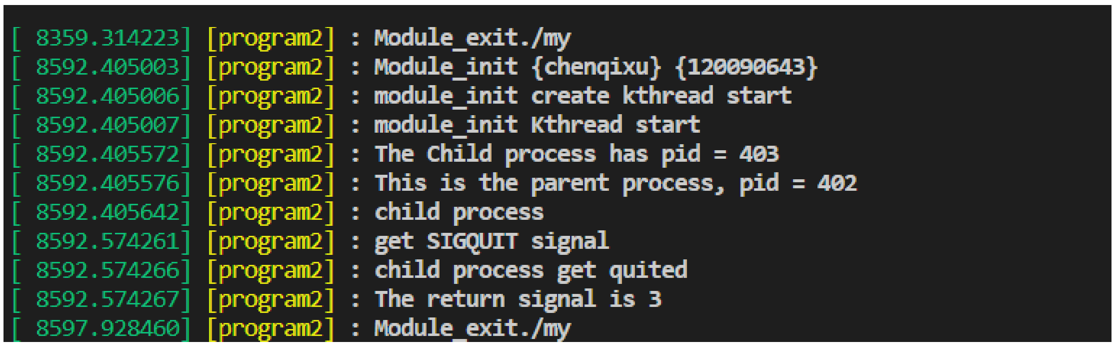
```
[ 7976.376418] [program2] : Module_init {chenqixu} {120090643}
[ 7976.376421] [program2] : module_init create kthread start
[ 7976.376422] [program2] : module_init Kthread start
[ 7976.376804] [program2] : The Child process has pid = 29535
[ 7976.376806] [program2] : This is the parent process, pid = 29534
[ 7976.377067] [program2] : child process
[ 7976.378124] [program2] : get SIGKILL signal
[ 7976.378127] [program2] : child process get killed
[ 7976.378128] [program2] : The return signal is 9
[ 7980.908398] [program2] : Module_exit./my
```

SIGPIPE



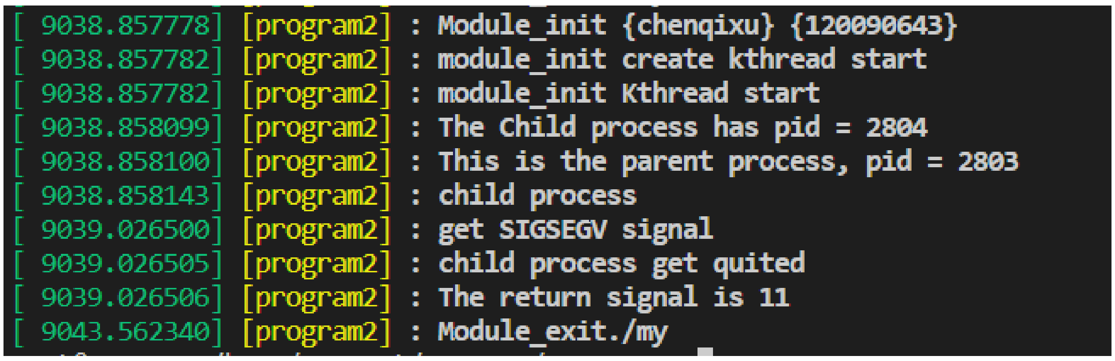
```
[ 8178.522202] [program2] : Module_init {chenqixu} {120090643}
[ 8178.522205] [program2] : module_init create kthread start
[ 8178.522206] [program2] : module_init Kthread start
[ 8178.522543] [program2] : The Child process has pid = 30702
[ 8178.522545] [program2] : This is the parent process, pid = 30701
[ 8178.522611] [program2] : child process
[ 8178.523602] [program2] : get SIGPIPE signal
[ 8178.523604] [program2] : child process get piped
[ 8178.523605] [program2] : The return signal is 13
[ 8183.061792] [program2] : Module_exit./my
```

SIGQUIT



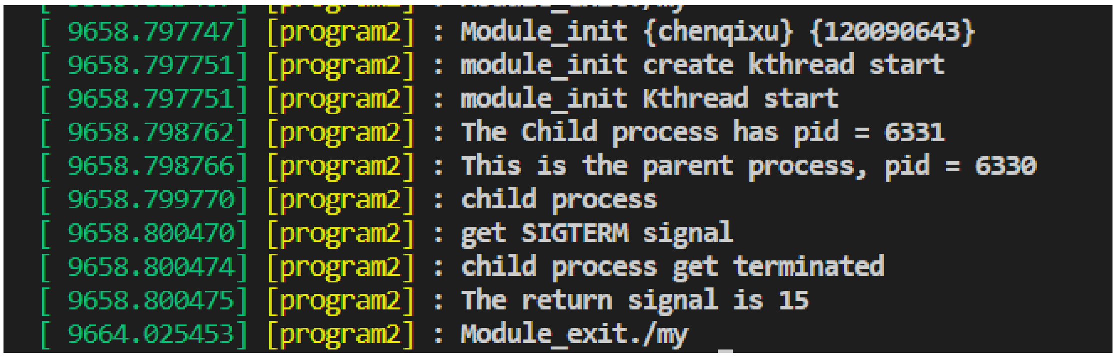
```
[ 8359.314223] [program2] : Module_exit./my
[ 8592.405003] [program2] : Module_init {chenqixu} {120090643}
[ 8592.405006] [program2] : module_init create kthread start
[ 8592.405007] [program2] : module_init Kthread start
[ 8592.405572] [program2] : The Child process has pid = 403
[ 8592.405576] [program2] : This is the parent process, pid = 402
[ 8592.405642] [program2] : child process
[ 8592.574261] [program2] : get SIGQUIT signal
[ 8592.574266] [program2] : child process get quited
[ 8592.574267] [program2] : The return signal is 3
[ 8597.928460] [program2] : Module_exit./my
```

SIGSEGV



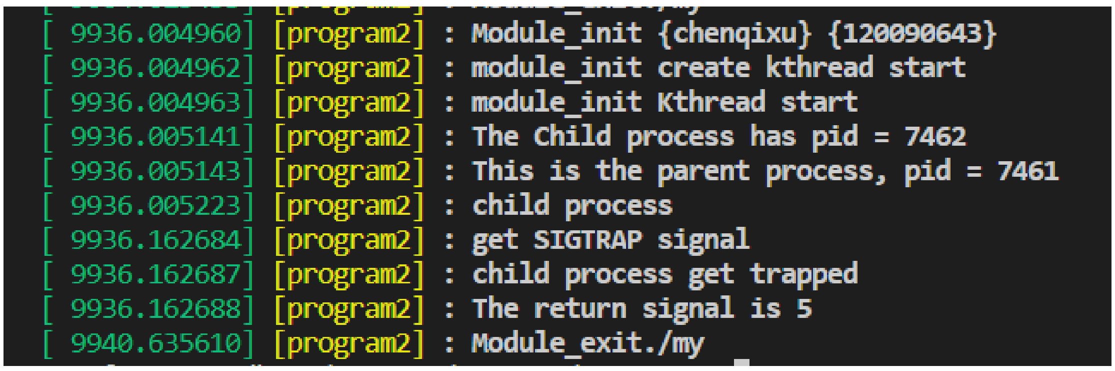
```
[ 9038.857778] [program2] : Module_init {chenqixu} {120090643}
[ 9038.857782] [program2] : module_init create kthread start
[ 9038.857782] [program2] : module_init Kthread start
[ 9038.858099] [program2] : The Child process has pid = 2804
[ 9038.858100] [program2] : This is the parent process, pid = 2803
[ 9038.858143] [program2] : child process
[ 9039.026500] [program2] : get SIGSEGV signal
[ 9039.026505] [program2] : child process get quited
[ 9039.026506] [program2] : The return signal is 11
[ 9043.562340] [program2] : Module_exit./my
```

SIGTERM



```
[ 9658.797747] [program2] : Module_init {chenqixu} {120090643}
[ 9658.797751] [program2] : module_init create kthread start
[ 9658.797751] [program2] : module_init Kthread start
[ 9658.798762] [program2] : The Child process has pid = 6331
[ 9658.798766] [program2] : This is the parent process, pid = 6330
[ 9658.799770] [program2] : child process
[ 9658.800470] [program2] : get SIGTERM signal
[ 9658.800474] [program2] : child process get terminated
[ 9658.800475] [program2] : The return signal is 15
[ 9664.025453] [program2] : Module_exit./my
```

SIGTRAP



```
[ 9936.004960] [program2] : Module_init {chenqixu} {120090643}
[ 9936.004962] [program2] : module_init create kthread start
[ 9936.004963] [program2] : module_init Kthread start
[ 9936.005141] [program2] : The Child process has pid = 7462
[ 9936.005143] [program2] : This is the parent process, pid = 7461
[ 9936.005223] [program2] : child process
[ 9936.162684] [program2] : get SIGTRAP signal
[ 9936.162687] [program2] : child process get trapped
[ 9936.162688] [program2] : The return signal is 5
[ 9940.635610] [program2] : Module_exit./my
```

What have I have learned from this project?

TASK1:

1. Get familiar with using the function fork (), execve (), waitpid ()
2. Get familiar about how parent process and child process interact with each other

TASK2:

1. Learned how to use the API in kernel while using the kernel mode.
2. Learned how to search for the target codes among thousands line of source codes
3. DECLARED BEFORE USE IN C LANGUAGE!