CSC3150 Assignment 1 Report

Yinggan XU 120090375

Design of the program

Program 1

Program 1 of mine basically follows what I've been taught in the tutorial. The process is like:

- Fork a child process
- Do something in the child process
- Raise the signals in child process and go back to parent process

I use `fork()` function to fork a child process. Then several logical statement and a `print_status()` function is used to report the child process' status.

Program 2

The second program is essentially the same as the first program, but is relatively complex. Starting from the `program2_init`, the program follows the process:

- Create a kernel thread
- Use `my_fork()` to fork a child process
- Use `my_exec()` to execute test program
- Use `my_wait()` to wait for the child process to terminate
- Analysis the signal of child process
  The most important thing in program 2 design is that all the functions/macros/components consists of kernel mode api/methods.

Bonus

The idea of doing bonus is tree search. First I scan the directory of `/proc/` and find all the numerical directories and store them in an array. Later, starting from `/proc/1/`, I find all of its child and recursively construct a process tree. Finally print out the tree.

Environment Setup

The environment setup consists of several steps.

VM Setup

I set up the VM using vgrant and VirtualBox, following tutorial 1. Basically I install a specific Ubuntu system with VirtualBox.

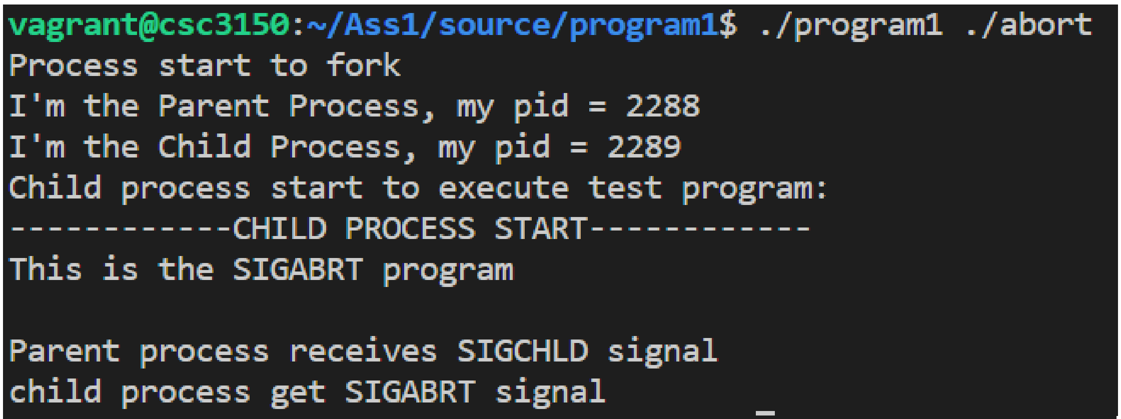
Kernel Compile

I first compile the kernel following the tutorial 2. Basically I downloaded kernel code of version `5.10.50` and put it under home directory of root user. The I make the kernel through series of make command and later install the new kernel. After my kernel is upgraded to `5.10.x`, I have later, in implementing program 2, found that the kernel code required modification. So I added `extern` and applied EXPORT_SYMBOL operation to several function, `do_wait()`, `kernel_clone()`, `do_execve()` etc. After doing so, the kernel is re-compiled.

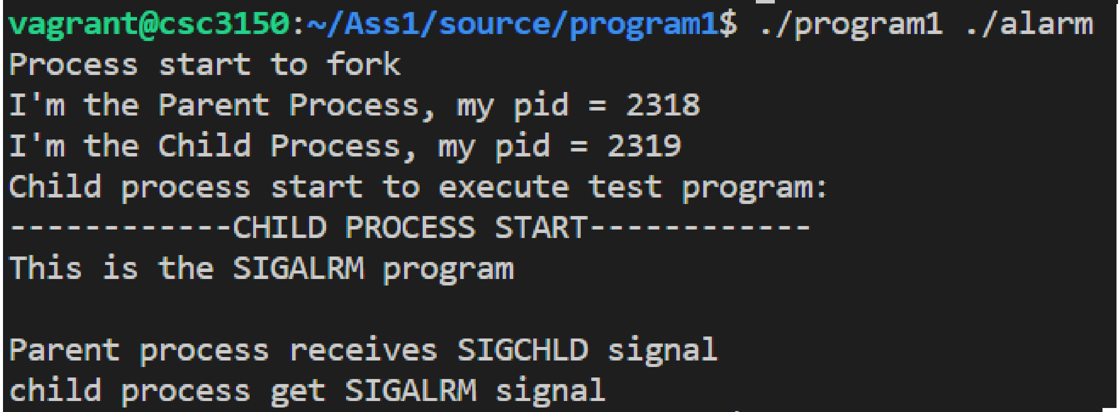
Test Output

Program 1

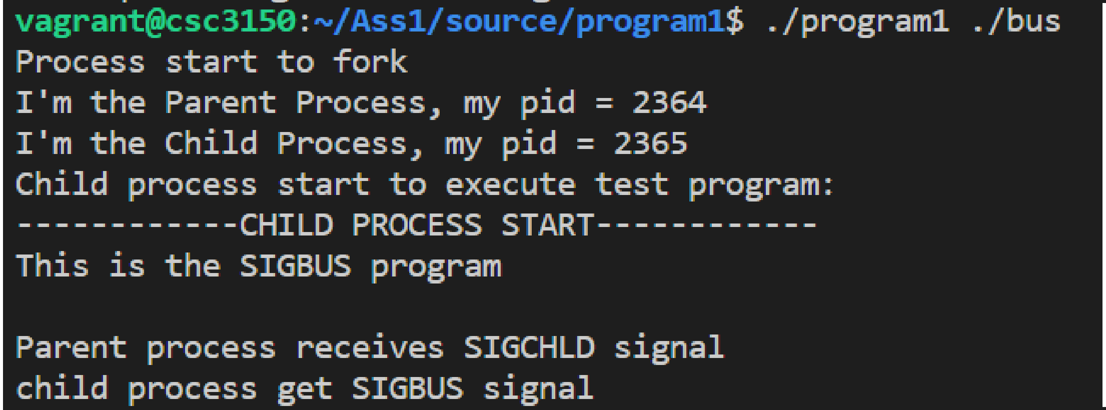
Here are the testing result of different signals' test program.



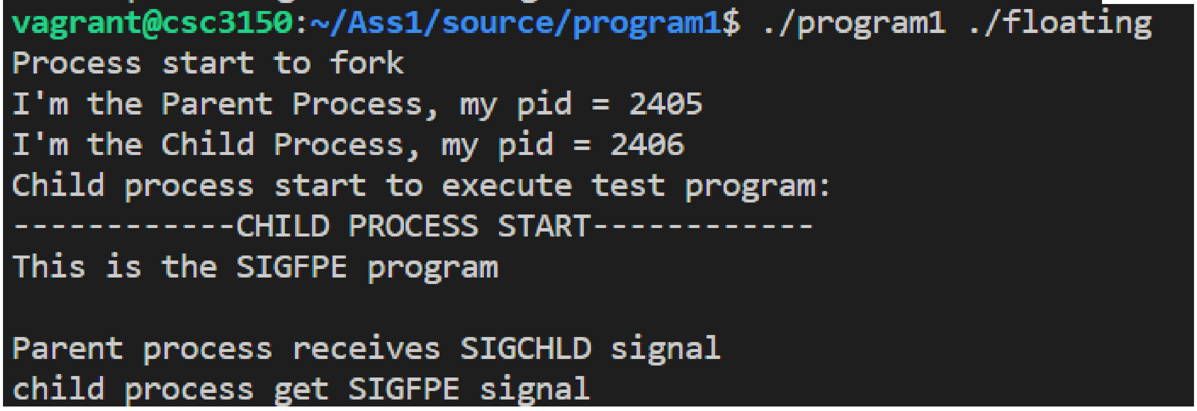
```
vagrant@csc3150:~/Ass1/source/program1$ ./program1 ./abort
Process start to fork
I'm the Parent Process, my pid = 2288
I'm the Child Process, my pid = 2289
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGABRT program

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



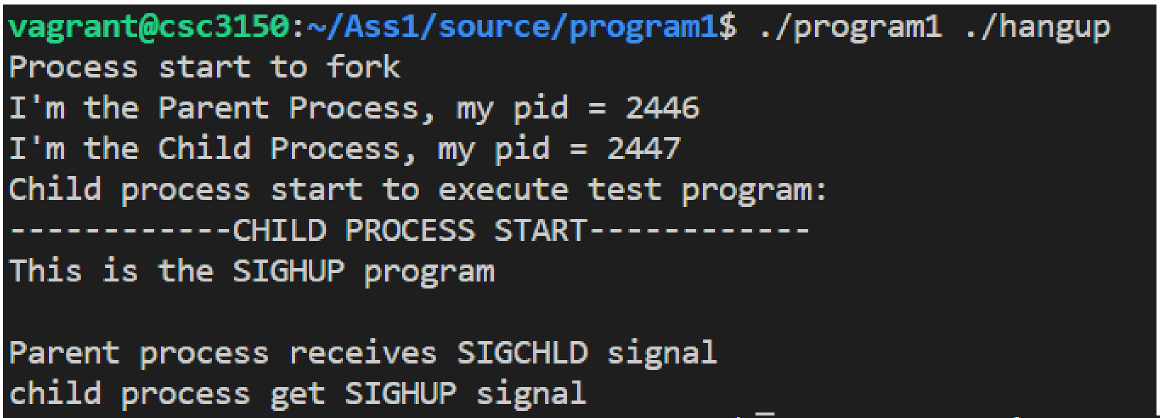
```
vagrant@csc3150:~/Ass1/source/program1$ ./program1 ./alarm
Process start to fork
I'm the Parent Process, my pid = 2318
I'm the Child Process, my pid = 2319
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGALRM program

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



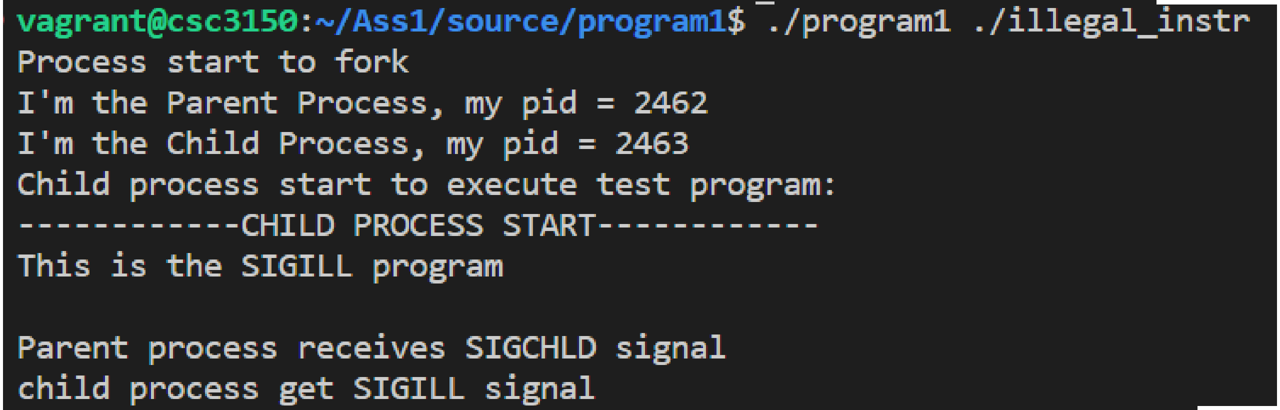
```
vagrant@csc3150:~/Ass1/source/program1$ ./program1 ./bus
Process start to fork
I'm the Parent Process, my pid = 2364
I'm the Child Process, my pid = 2365
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGBUS program

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



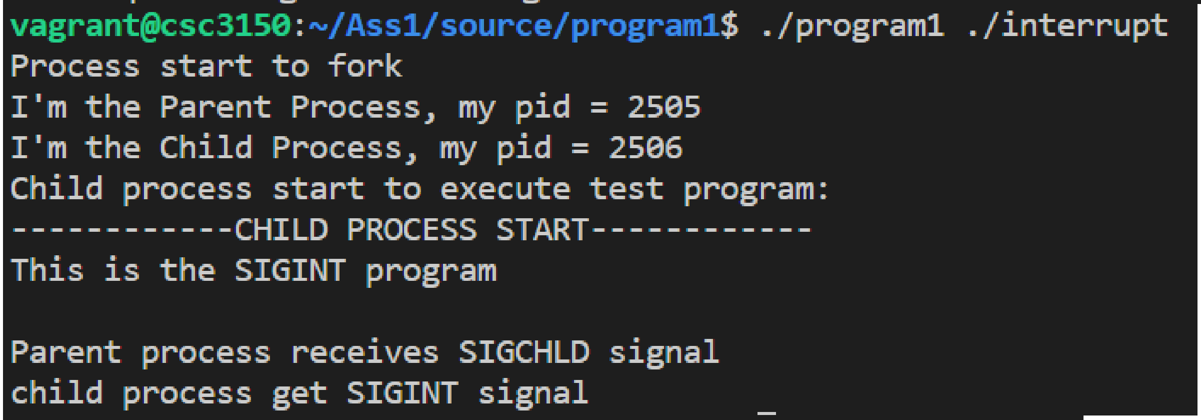
```
vagrant@csc3150:~/Ass1/source/program1$ ./program1 ./floating
Process start to fork
I'm the Parent Process, my pid = 2405
I'm the Child Process, my pid = 2406
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGFPE program

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



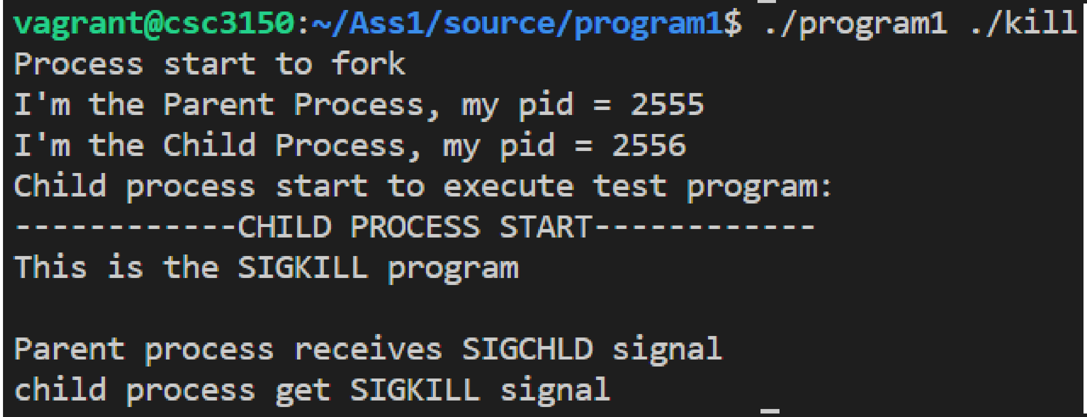
```
vagrant@csc3150:~/Ass1/source/program1$ ./program1 ./hangup
Process start to fork
I'm the Parent Process, my pid = 2446
I'm the Child Process, my pid = 2447
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGHUP program

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



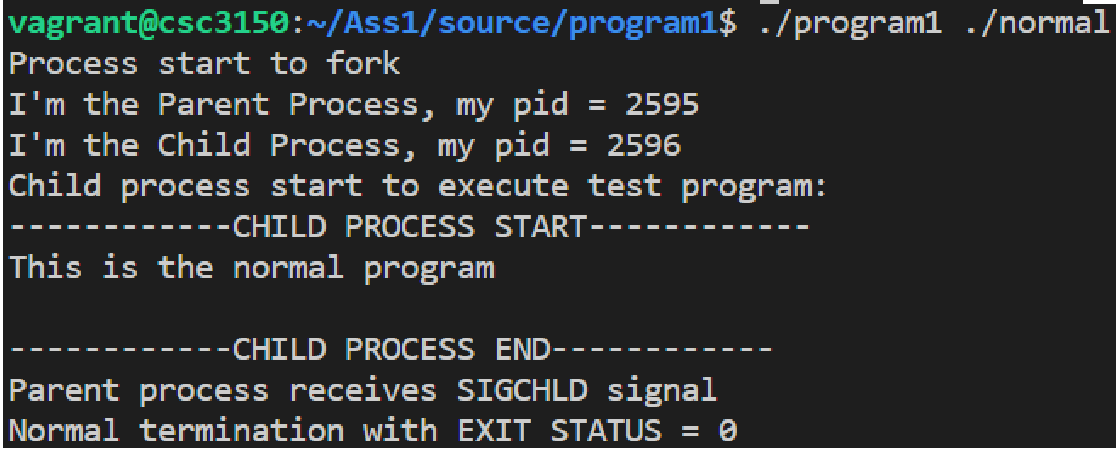
```
vagrant@csc3150:~/Ass1/source/program1$ ./program1 ./illegal_instr
Process start to fork
I'm the Parent Process, my pid = 2462
I'm the Child Process, my pid = 2463
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGILL program

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



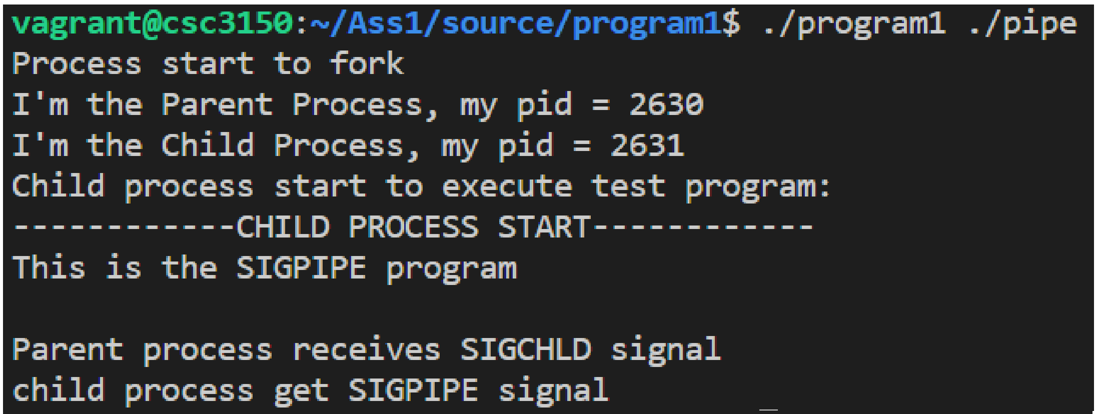
```
vagrant@csc3150:~/Ass1/source/program1$ ./program1 ./interrupt
Process start to fork
I'm the Parent Process, my pid = 2505
I'm the Child Process, my pid = 2506
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGINT program

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



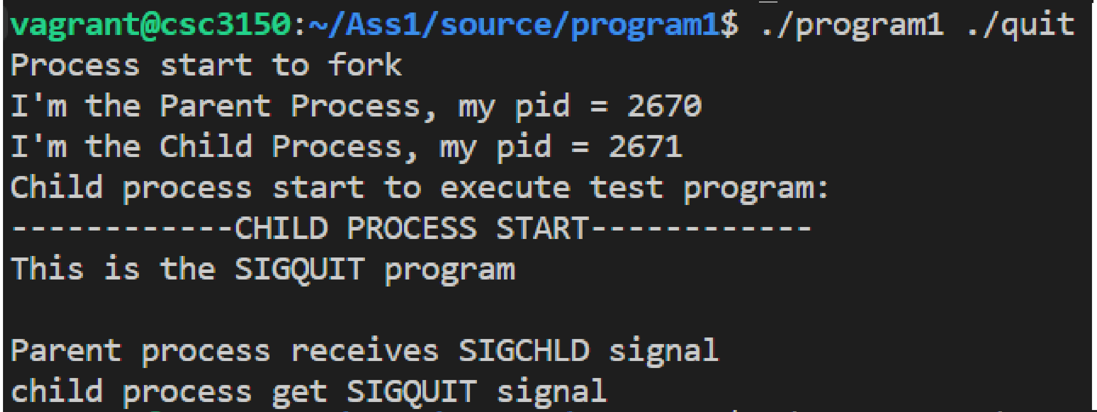
```
vagrant@csc3150:~/Ass1/source/program1$ ./program1 ./kill
Process start to fork
I'm the Parent Process, my pid = 2555
I'm the Child Process, my pid = 2556
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGKILL program

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



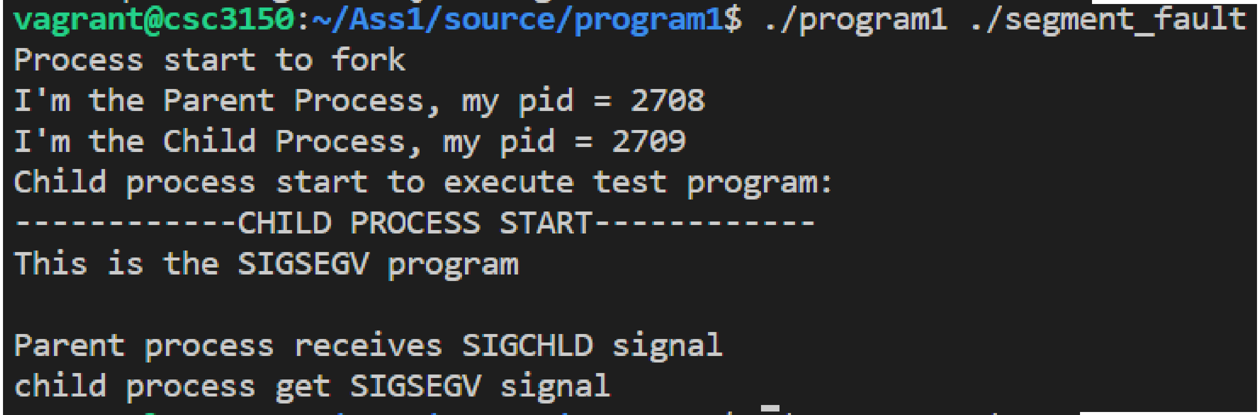
```
vagrant@csc3150:~/Ass1/source/program1$ ./program1 ./normal
Process start to fork
I'm the Parent Process, my pid = 2595
I'm the Child Process, my pid = 2596
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the normal program

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



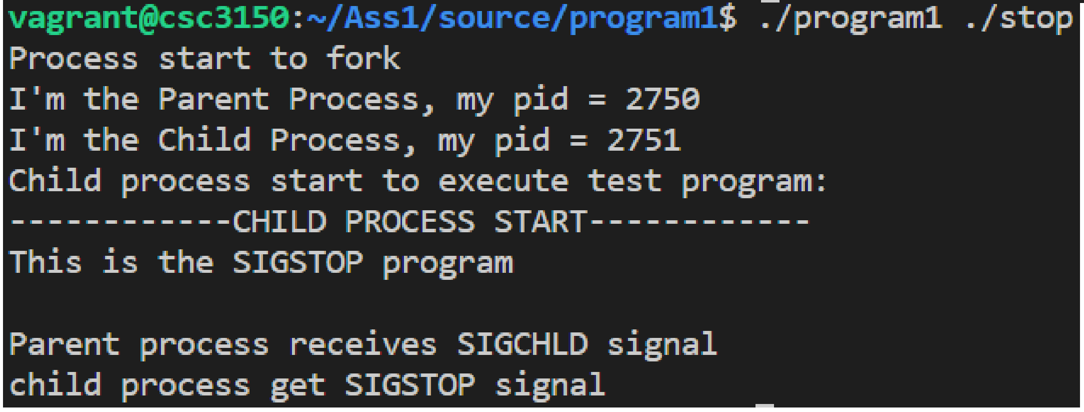
```
vagrant@csc3150:~/Ass1/source/program1$ ./program1 ./pipe
Process start to fork
I'm the Parent Process, my pid = 2630
I'm the Child Process, my pid = 2631
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGPIPE program

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



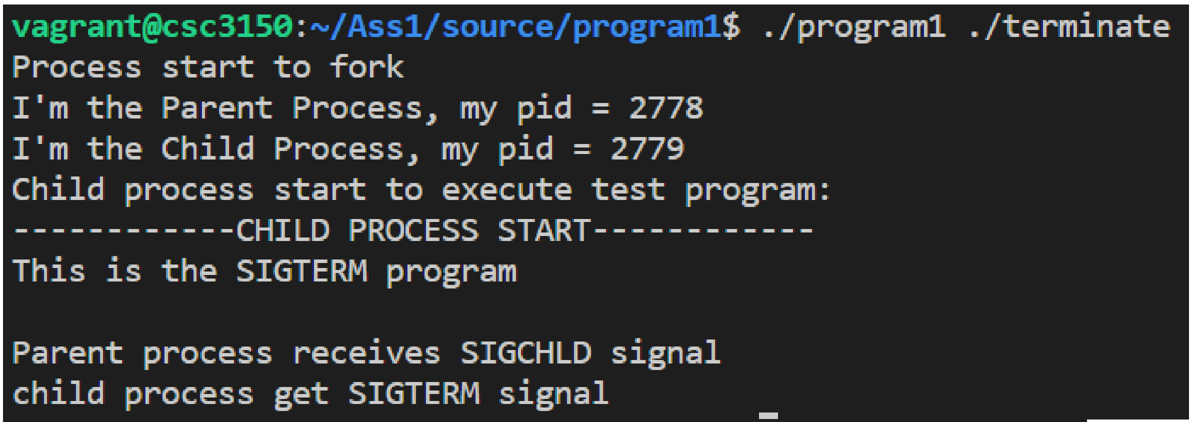
```
vagrant@csc3150:~/Ass1/source/program1$ ./program1 ./quit
Process start to fork
I'm the Parent Process, my pid = 2670
I'm the Child Process, my pid = 2671
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

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



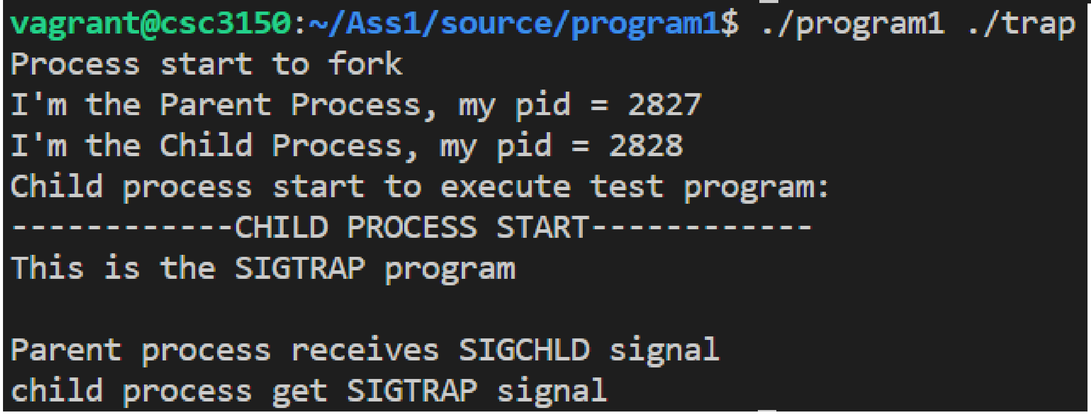
```
vagrant@csc3150:~/Ass1/source/program1$ ./program1 ./segment_fault
Process start to fork
I'm the Parent Process, my pid = 2708
I'm the Child Process, my pid = 2709
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

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



```
vagrant@csc3150:~/Ass1/source/program1$ ./program1 ./stop
Process start to fork
I'm the Parent Process, my pid = 2750
I'm the Child Process, my pid = 2751
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

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



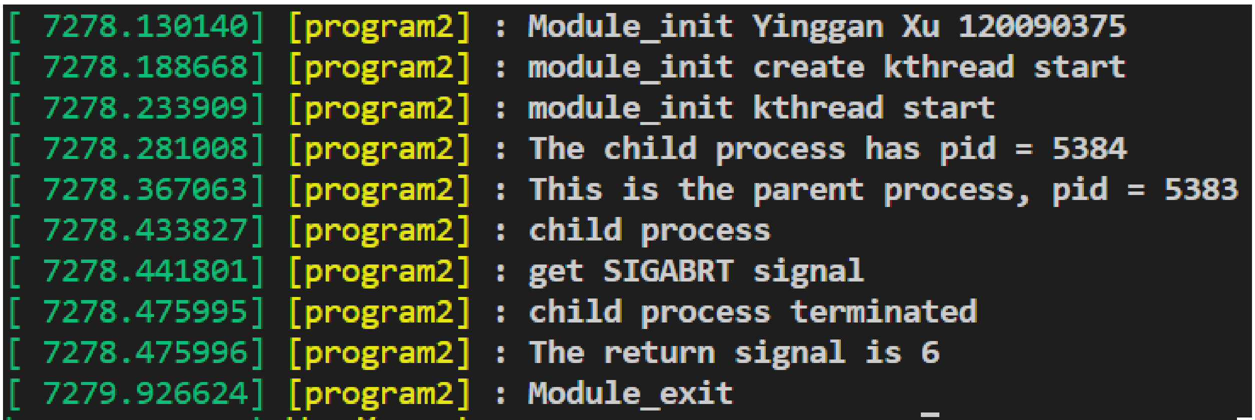
```
vagrant@csc3150:~/Ass1/source/program1$ ./program1 ./terminate
Process start to fork
I'm the Parent Process, my pid = 2778
I'm the Child Process, my pid = 2779
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTERM program

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



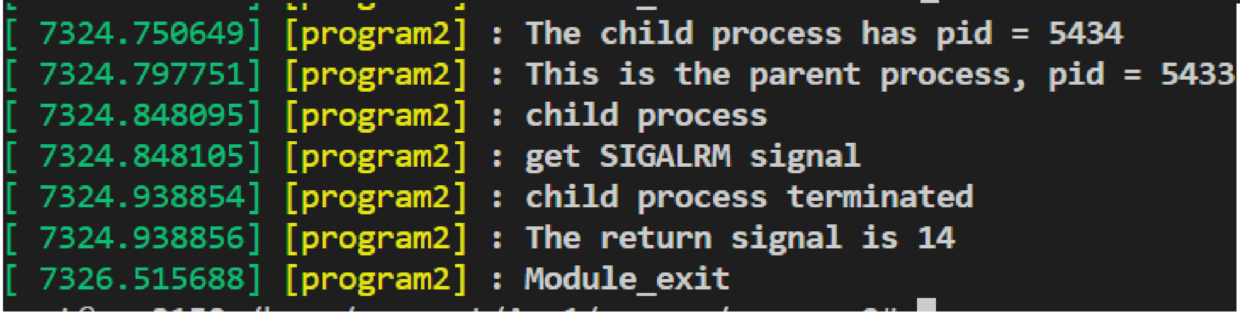
```
vagrant@csc3150:~/Ass1/source/program1$ ./program1 ./trap
Process start to fork
I'm the Parent Process, my pid = 2827
I'm the Child Process, my pid = 2828
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTRAP program

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

Program 2

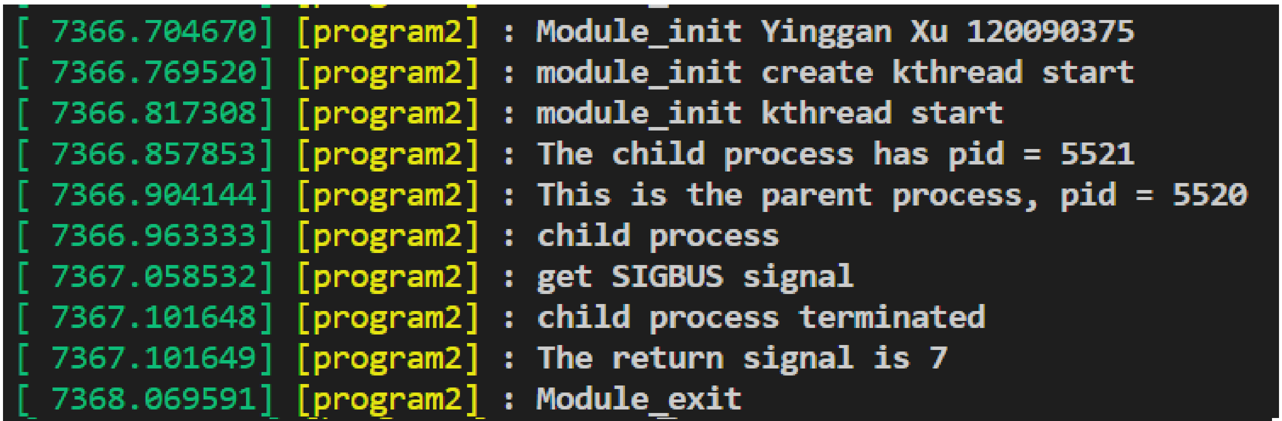
Here are the testing result of program 2 testing. Due to the log information is way too long, so the results only show different part rather than whole picture, except for the first one.



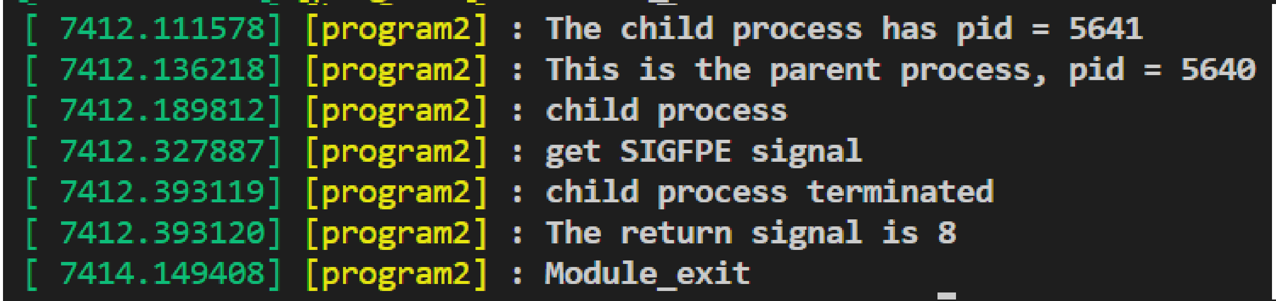
```
[ 7278.130140] [program2] : Module_init Yinggan Xu 120090375
[ 7278.188668] [program2] : module_init create kthread start
[ 7278.233909] [program2] : module_init kthread start
[ 7278.281008] [program2] : The child process has pid = 5384
[ 7278.367063] [program2] : This is the parent process, pid = 5383
[ 7278.433827] [program2] : child process
[ 7278.441801] [program2] : get SIGABRT signal
[ 7278.475995] [program2] : child process terminated
[ 7278.475996] [program2] : The return signal is 6
[ 7279.926624] [program2] : Module_exit
```



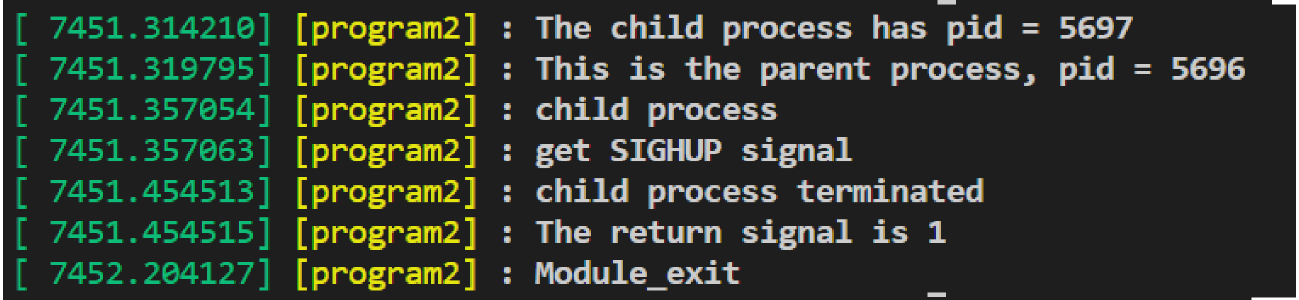
```
[ 7324.750649] [program2] : The child process has pid = 5434
[ 7324.797751] [program2] : This is the parent process, pid = 5433
[ 7324.848095] [program2] : child process
[ 7324.848105] [program2] : get SIGALRM signal
[ 7324.938854] [program2] : child process terminated
[ 7324.938856] [program2] : The return signal is 14
[ 7326.515688] [program2] : Module_exit
```



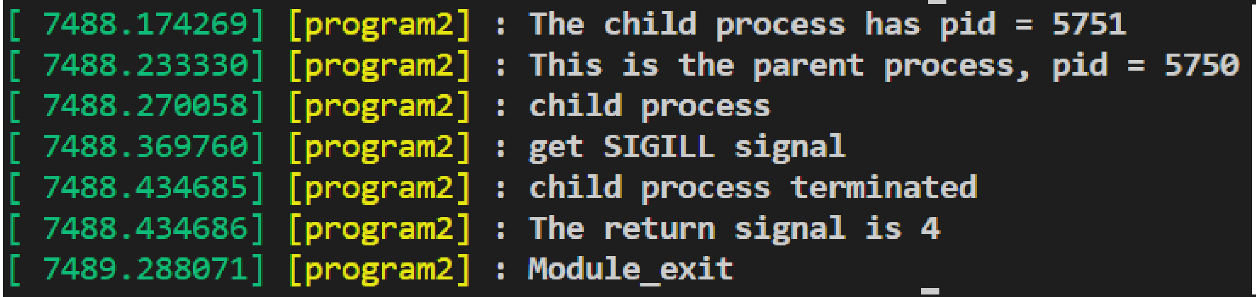
```
[ 7366.704670] [program2] : Module_init Yinggan Xu 120090375
[ 7366.769520] [program2] : module_init create kthread start
[ 7366.817308] [program2] : module_init kthread start
[ 7366.857853] [program2] : The child process has pid = 5521
[ 7366.904144] [program2] : This is the parent process, pid = 5520
[ 7366.963333] [program2] : child process
[ 7367.058532] [program2] : get SIGBUS signal
[ 7367.101648] [program2] : child process terminated
[ 7367.101649] [program2] : The return signal is 7
[ 7368.069591] [program2] : Module_exit
```



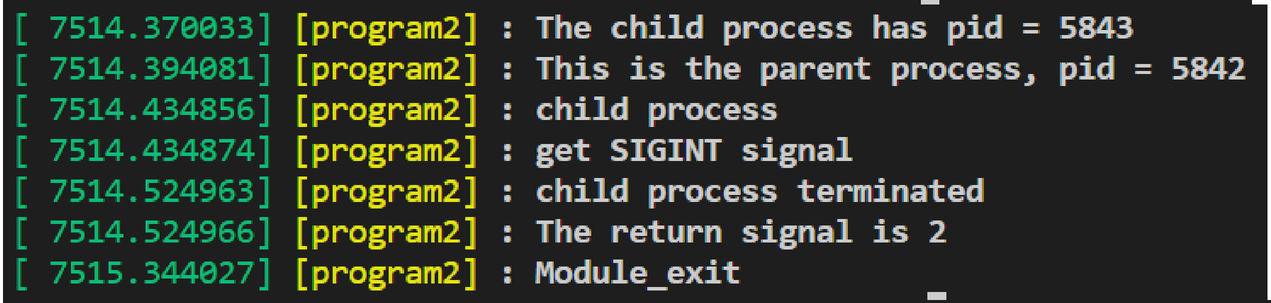
```
[ 7412.111578] [program2] : The child process has pid = 5641
[ 7412.136218] [program2] : This is the parent process, pid = 5640
[ 7412.189812] [program2] : child process
[ 7412.327887] [program2] : get SIGFPE signal
[ 7412.393119] [program2] : child process terminated
[ 7412.393120] [program2] : The return signal is 8
[ 7414.149408] [program2] : Module_exit
```



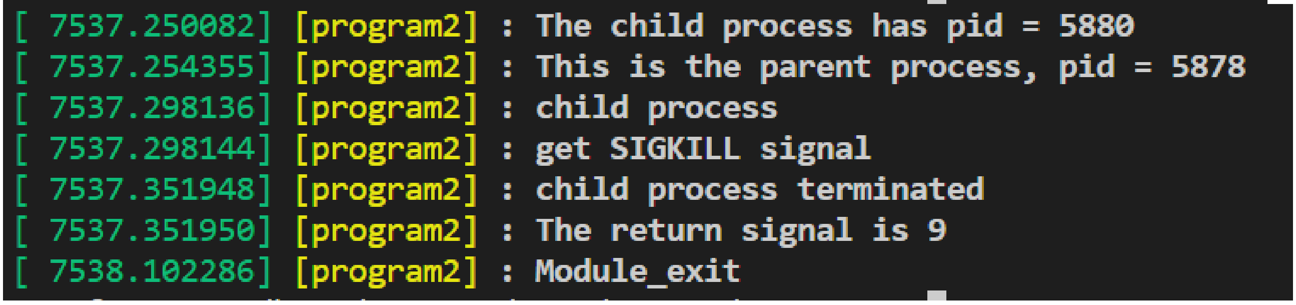
```
[ 7451.314210] [program2] : The child process has pid = 5697
[ 7451.319795] [program2] : This is the parent process, pid = 5696
[ 7451.357054] [program2] : child process
[ 7451.357063] [program2] : get SIGHUP signal
[ 7451.454513] [program2] : child process terminated
[ 7451.454515] [program2] : The return signal is 1
[ 7452.204127] [program2] : Module_exit
```



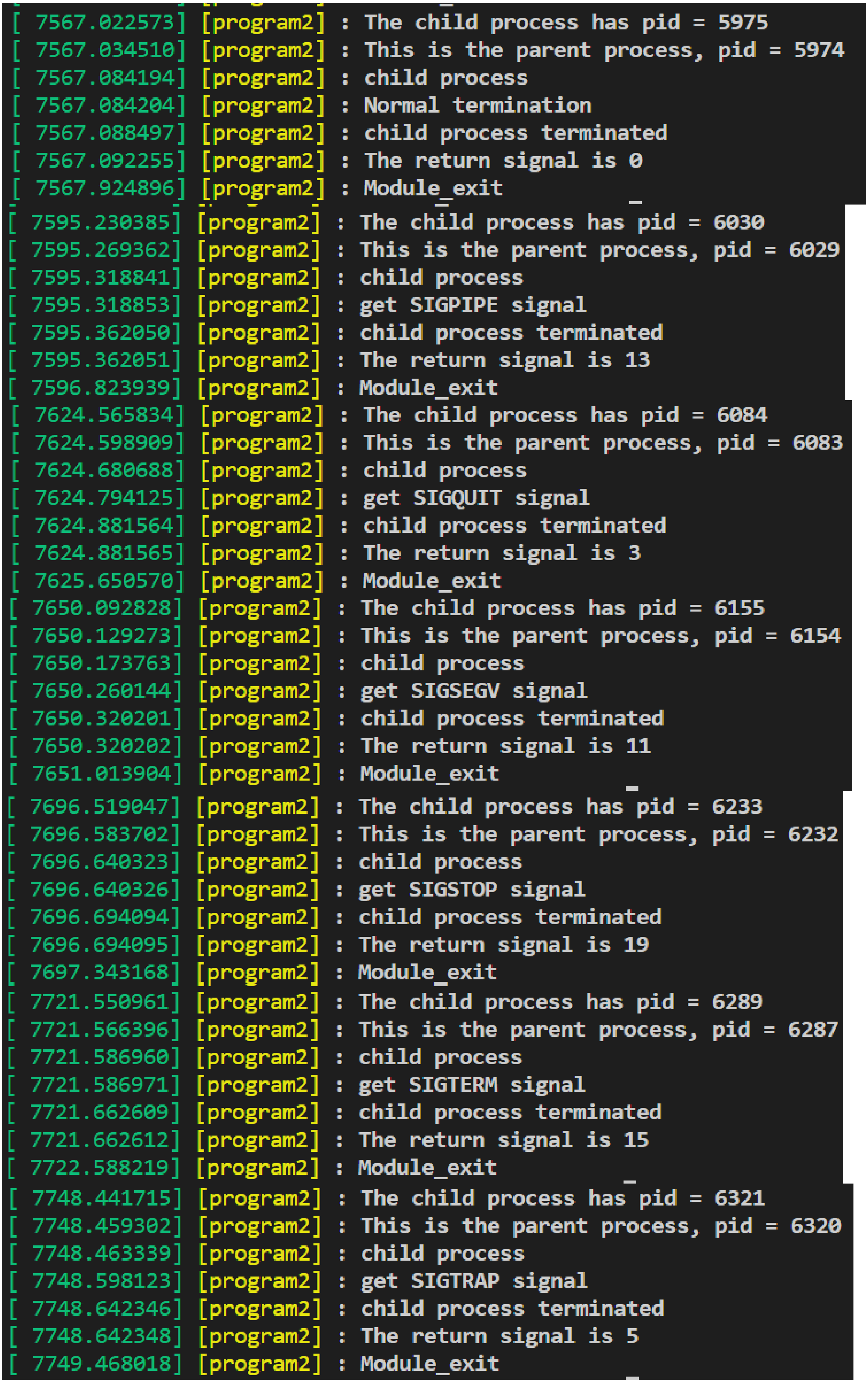
```
[ 7488.174269] [program2] : The child process has pid = 5751
[ 7488.233330] [program2] : This is the parent process, pid = 5750
[ 7488.270058] [program2] : child process
[ 7488.369760] [program2] : get SIGILL signal
[ 7488.434685] [program2] : child process terminated
[ 7488.434686] [program2] : The return signal is 4
[ 7489.288071] [program2] : Module_exit
```



```
[ 7514.370033] [program2] : The child process has pid = 5843
[ 7514.394081] [program2] : This is the parent process, pid = 5842
[ 7514.434856] [program2] : child process
[ 7514.434874] [program2] : get SIGINT signal
[ 7514.524963] [program2] : child process terminated
[ 7514.524966] [program2] : The return signal is 2
[ 7515.344027] [program2] : Module_exit
```



```
[ 7537.250082] [program2] : The child process has pid = 5880
[ 7537.254355] [program2] : This is the parent process, pid = 5878
[ 7537.298136] [program2] : child process
[ 7537.298144] [program2] : get SIGKILL signal
[ 7537.351948] [program2] : child process terminated
[ 7537.351950] [program2] : The return signal is 9
[ 7538.102286] [program2] : Module_exit
```



```
[ 7567.022573] [program2] : The child process has pid = 5975
[ 7567.034510] [program2] : This is the parent process, pid = 5974
[ 7567.084194] [program2] : child process
[ 7567.084204] [program2] : Normal termination
[ 7567.088497] [program2] : child process terminated
[ 7567.092255] [program2] : The return signal is 0
[ 7567.924896] [program2] : Module_exit
[ 7595.230385] [program2] : The child process has pid = 6030
[ 7595.269362] [program2] : This is the parent process, pid = 6029
[ 7595.318841] [program2] : child process
[ 7595.318853] [program2] : get SIGPIPE signal
[ 7595.362050] [program2] : child process terminated
[ 7595.362051] [program2] : The return signal is 13
[ 7596.823939] [program2] : Module_exit
[ 7624.565834] [program2] : The child process has pid = 6084
[ 7624.598909] [program2] : This is the parent process, pid = 6083
[ 7624.680688] [program2] : child process
[ 7624.794125] [program2] : get SIGQUIT signal
[ 7624.881564] [program2] : child process terminated
[ 7624.881565] [program2] : The return signal is 3
[ 7625.650570] [program2] : Module_exit
[ 7650.092828] [program2] : The child process has pid = 6155
[ 7650.129273] [program2] : This is the parent process, pid = 6154
[ 7650.173763] [program2] : child process
[ 7650.260144] [program2] : get SIGSEGV signal
[ 7650.320201] [program2] : child process terminated
[ 7650.320202] [program2] : The return signal is 11
[ 7651.013904] [program2] : Module_exit
[ 7696.519047] [program2] : The child process has pid = 6233
[ 7696.583702] [program2] : This is the parent process, pid = 6232
[ 7696.640323] [program2] : child process
[ 7696.640326] [program2] : get SIGSTOP signal
[ 7696.694094] [program2] : child process terminated
[ 7696.694095] [program2] : The return signal is 19
[ 7697.343168] [program2] : Module_exit
[ 7721.550961] [program2] : The child process has pid = 6289
[ 7721.566396] [program2] : This is the parent process, pid = 6287
[ 7721.586960] [program2] : child process
[ 7721.586971] [program2] : get SIGTERM signal
[ 7721.662609] [program2] : child process terminated
[ 7721.662612] [program2] : The return signal is 15
[ 7722.588219] [program2] : Module_exit
[ 7748.441715] [program2] : The child process has pid = 6321
[ 7748.459302] [program2] : This is the parent process, pid = 6320
[ 7748.463339] [program2] : child process
[ 7748.598123] [program2] : get SIGTRAP signal
[ 7748.642346] [program2] : child process terminated
[ 7748.642348] [program2] : The return signal is 5
[ 7749.468018] [program2] : Module_exit
```

Bonus

Due to time limit and bad health condition, I only implement several preliminary functions for bonus.

What did I learnt from program 2?

First of all, program 2 gives me a chance to get familiar with kernel. I learnt to create kernel thread, fork child process, execute program, and wait for child process in kernel code.
I also learnt:

1. How to utilize kernel code
2. How does `wait()` and other interfaces work
3. I also gained some knowledge about how other program in user mode called kernel mode