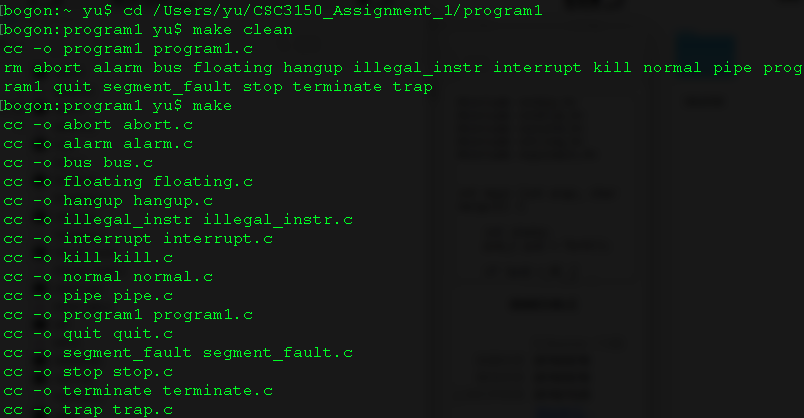
**CSC3150 Assignment1 Report (116010274 俞佳含)**

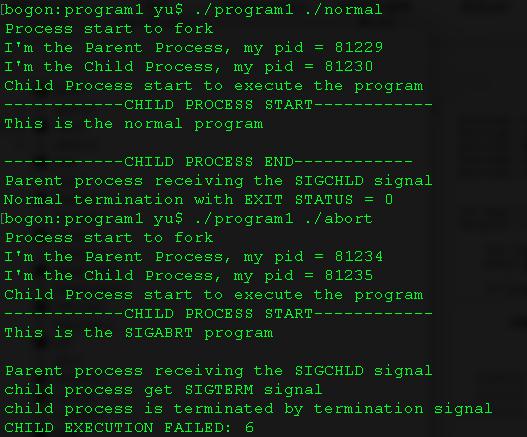
**[Program 1]**

1. Program 1 can be divided into several parts.
2. Fork a child process;
3. Child process execute test program;
4. Parent wait for the child to terminate;
5. Parent check child’s process’s termination status
6. Execute program1 in mac terminal.
   1. Go into the directory where contains program.c file.
   2. Preparation work: type “make clean”, then type “make”.

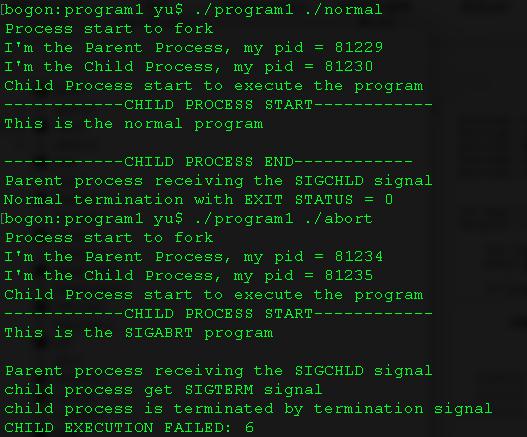


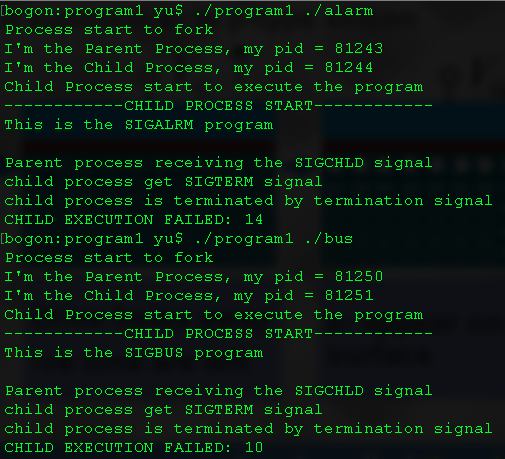
1. Screenshot of program output.

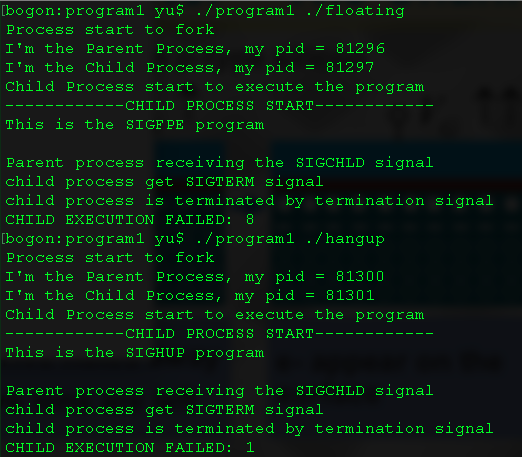
* Demo output for normal termination:

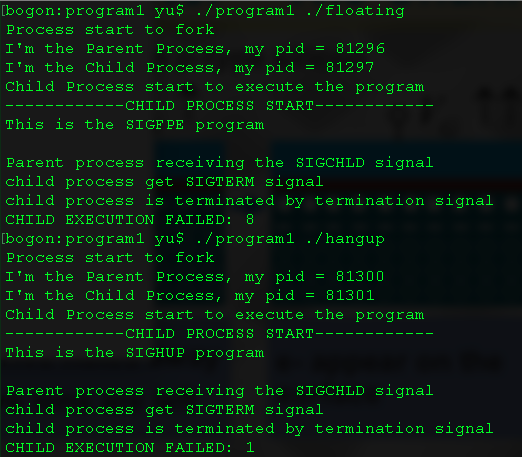


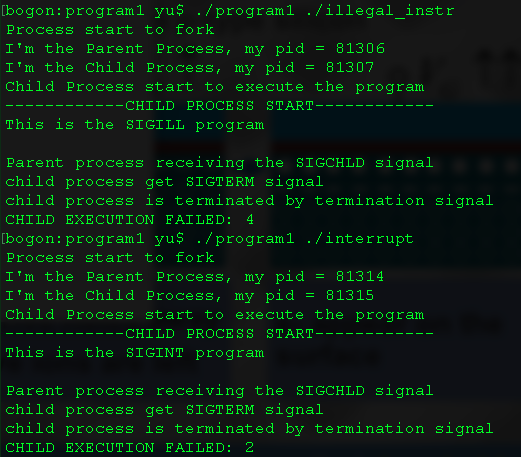
* Demo output for signaled termination:

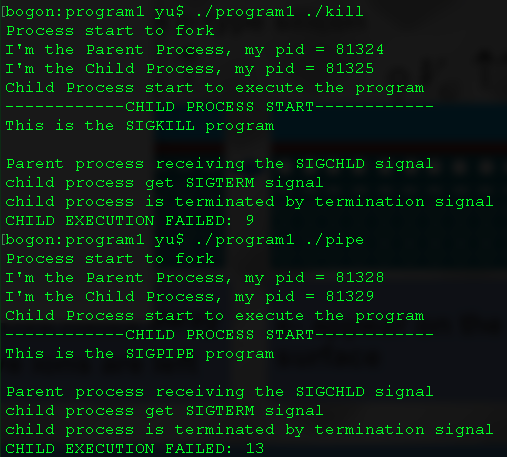


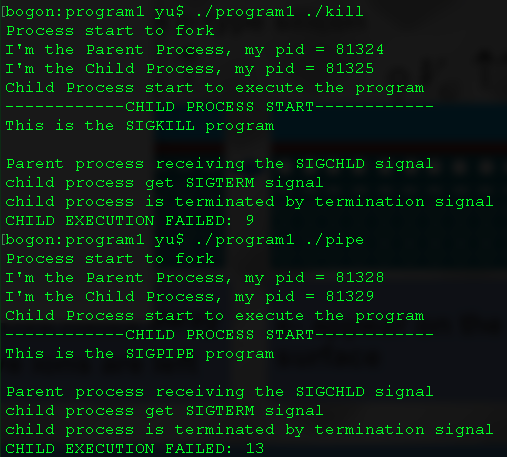


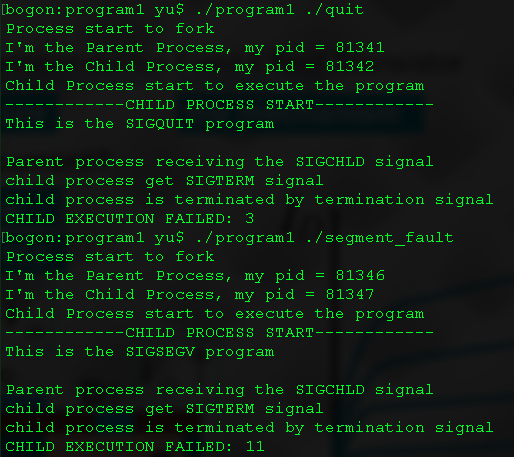


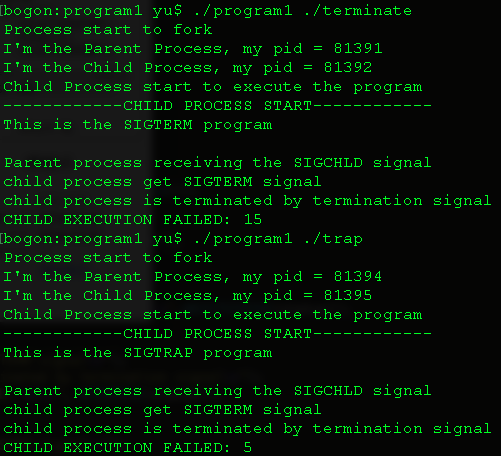


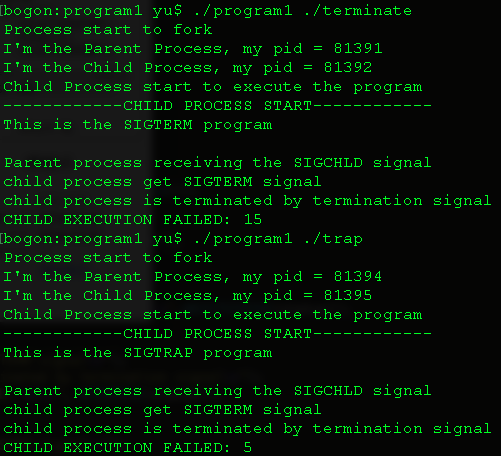




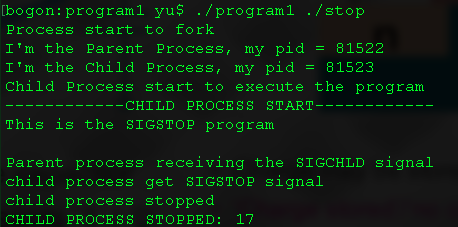








* Demo output for stopped:



1. What I learn.

* Process Creation
* Parent and Child Processes
* Process Termination
* Process Signal
* Executing a file

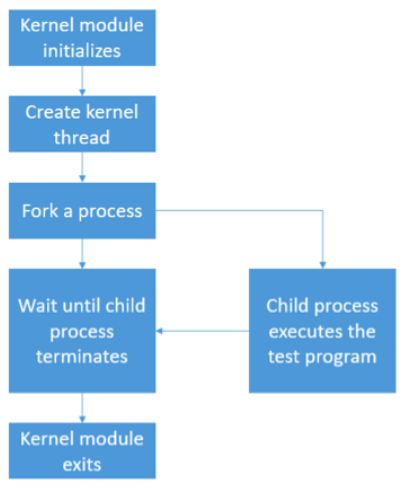
**[Program 2]**

1. Preparation work: revising Linux (4.10.4) kernel

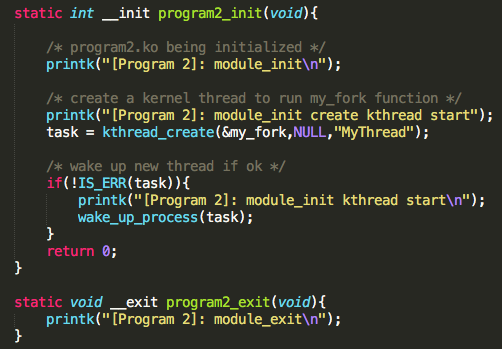
Update the Linux source code: export symbol so that \_do\_fork(), do\_execve(), and getname() functions could be used in the kernel module;

Then compile the kernel and boot image, replace the boot image with new one, then reboot.

1. Main flow of program2:

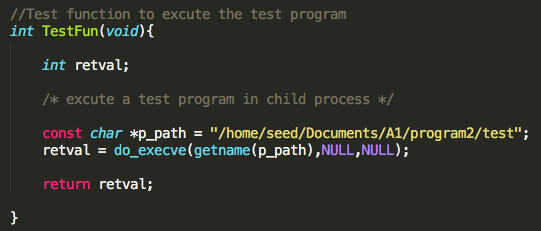


Below are functions of program2.ko being initialized and exited. Inside the initializing function, use kthread\_create() to create a kernel thread and run my\_fork function.



Within my\_fork function, use \_do\_fork() function to fork a process, and in the function TestFun(i.e. test function), the test program (i.e. test.c) will be executed.





By using function do\_wait(), the parent process will wait until child process terminates.



Print out the process id for both parent and child process.



Within this test program, it will raise signal. Below set default signal.

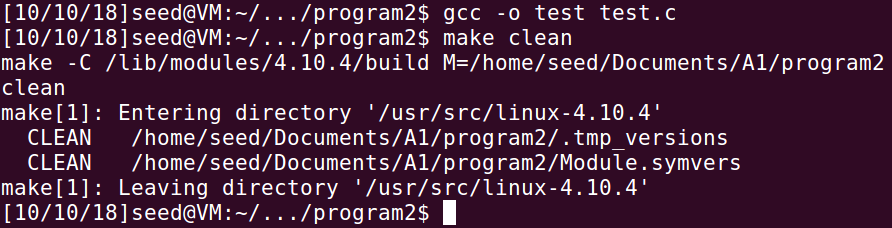


The signal could be caught and related message should be printed out in kernel log.

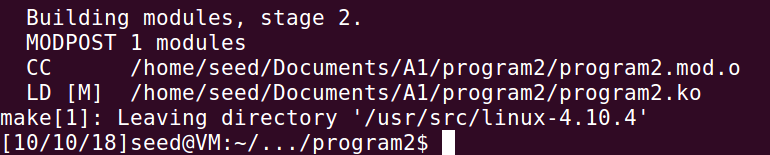


1. Steps to execute my program

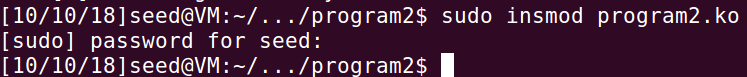
First, compile program2 steps:



* To compile to test program, first type 'gcc -o test test.c'.
* (In the 'program2' directory, type 'make clean' command and enter.)
* In the 'program2' directory, type 'make' command and enter

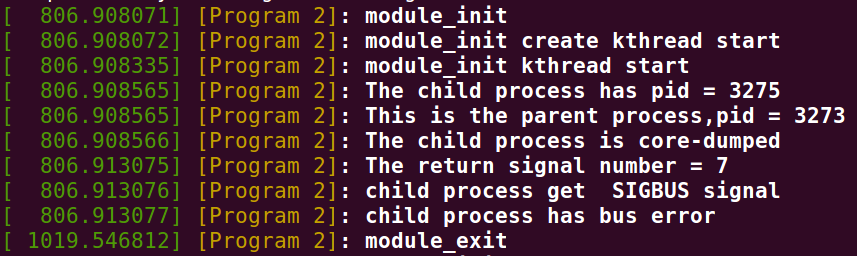


Second, execute program2 steps:

* + Type 'sudo insmod program2.ko' under 'program2' directory and enter
  + Type ‘dmesg’ command to see messages between 'module init' and 'module exit'.
  + Type 'sudo rmmod program2' and enter to remove the program2 module.

*Remark for my program.c file: in my test function, I used absolute path. If you want to run my code, you need to transfer to your local path where the test file test.c is.*

1. Screenshot of program2 output



1. What have I learned from program2

* Compile kernel
* Export symbol
* Build kernel object
* Insert and remove kernel module
* Create kernel thread
* Process creation
* Program execution
* Wait for signal

**[Bonus]**