CSC3150 Assignment 1

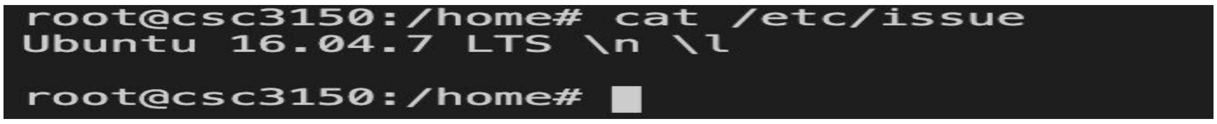
1. Overview

This project is composed of three programs. Program 1 implement to fork a child process in the user mode and execute the test program. Then the parent process waits for the termination of child process and output the signals sent by the children process. Program 2 implement to create a kernel threat and fork a new process in the kernel mode. Also, the parent process wait for the termination of child process and handle and output different signals. Moreover, the bonus program implement the linux command pstree that can print out the process tree of linux. In conclusion, this project is aims to implement how to fork a new process in the user mode and kernel mode and how parent process waits and handles the signals sent back from child process.

2. Development Environment

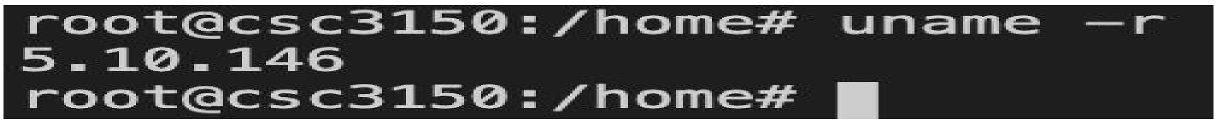
The detailed information of development environment is as following:

Version of OS: **Ubuntu 16.04.7**



```
root@csc3150:/home# cat /etc/issue
Ubuntu 16.04.7 LTS \n \l

root@csc3150:/home# 
```

Version of kernel: Linux-5.10.146



```
root@csc3150:/home# uname -r
5.10.146
root@csc3150:/home# 
```

Version of GCC: 5.4.0

```
gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.12)
```

The steps to compile kernel: Firstly, download the linux kernel source code and install dependency and development tools. Secondly, extract and unzip the source file to certain folder and copy the config file to the folder. Thirdly, login in root account, clean the previous setting and start configuration. Then build kernel image and modules, install kernel modules and install kernel. Finally, reboot to load the new compiled kernel.

3. Program Information

(1) Program 1

(a) Flow of the program 1

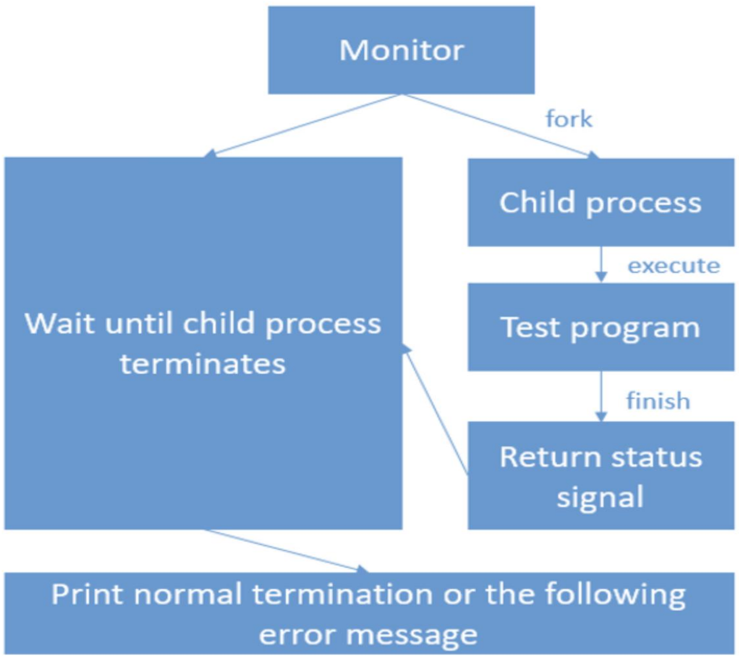

Monitor
fork
Child process
execute
Test program
finish
Return status signal
Wait until child process terminates
Print normal termination or the following error message

Figure 1. The flow chart of program 1

This program is to fork a child process to execute the test program. Parent process waits and receives the signals from child process, and then handle and output the information of the signal.

(b) Implementation

First, the program use fork( ) function to fork a child process. After checking the process is forked, the program make it execute the test program. And the program let parent process wait for the signal sent from the child process and output the information of the signal.

(c) Execution steps of the program

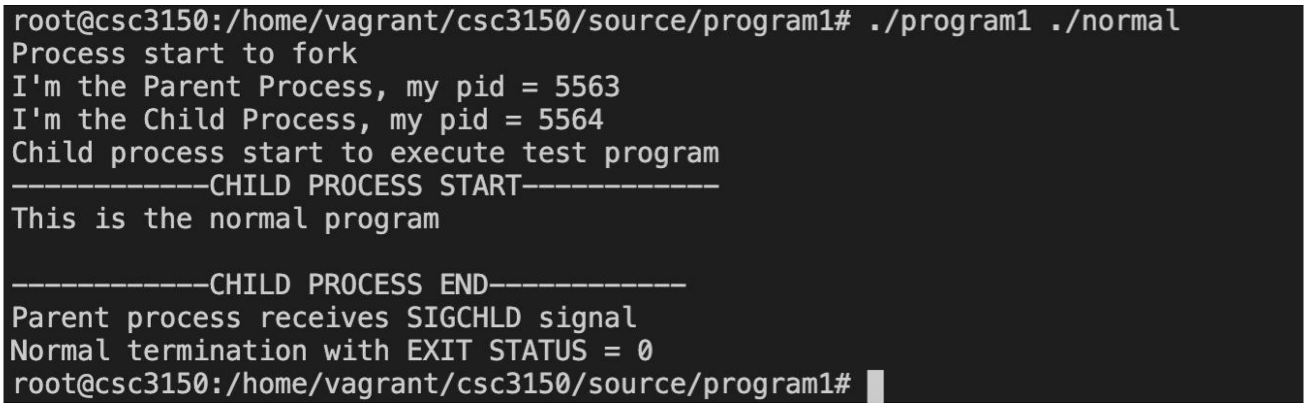
I. Enter cd command to the location of the program folder

II. Enter make in the terminal to compile the code.

III. Enter "./program1 ./testprogram" in the terminal. The {testprogram} is the name of the compiled test program.

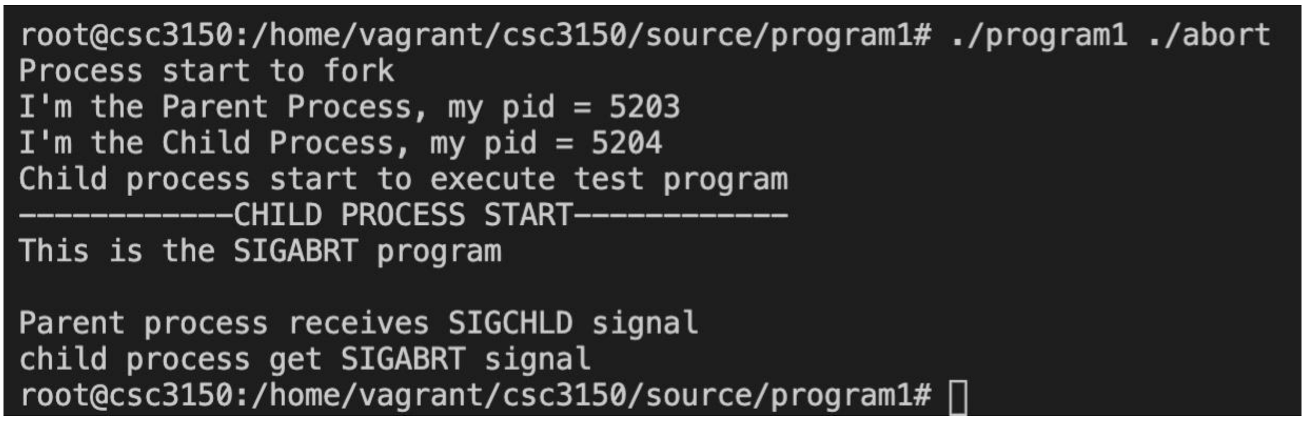
(d) Outputs of the program

1. normal



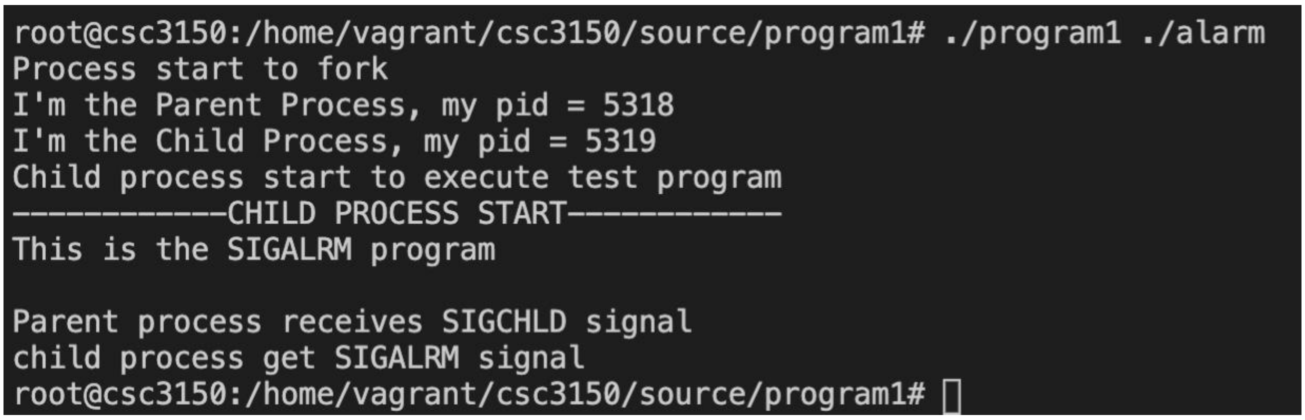
```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./normal
Process start to fork
I'm the Parent Process, my pid = 5563
I'm the Child Process, my pid = 5564
Child process start to execute test program
------------CHILD PROCESS START------------
This is the normal program

------------CHILD PROCESS END------------
Parent process receives SIGCHLD signal
Normal termination with EXIT STATUS = 0
root@csc3150:/home/vagrant/csc3150/source/program1# 
```

2. abort



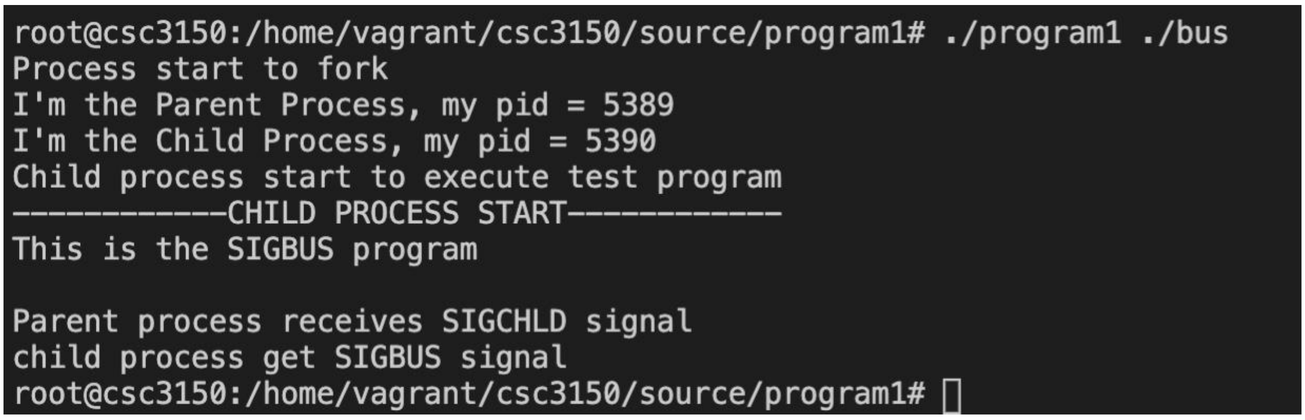
```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./abort
Process start to fork
I'm the Parent Process, my pid = 5203
I'm the Child Process, my pid = 5204
Child process start to execute test program
------------CHILD PROCESS START------------
This is the SIGABRT program

Parent process receives SIGCHLD signal
child process get SIGABRT signal
root@csc3150:/home/vagrant/csc3150/source/program1# 
```

3. **alarm**



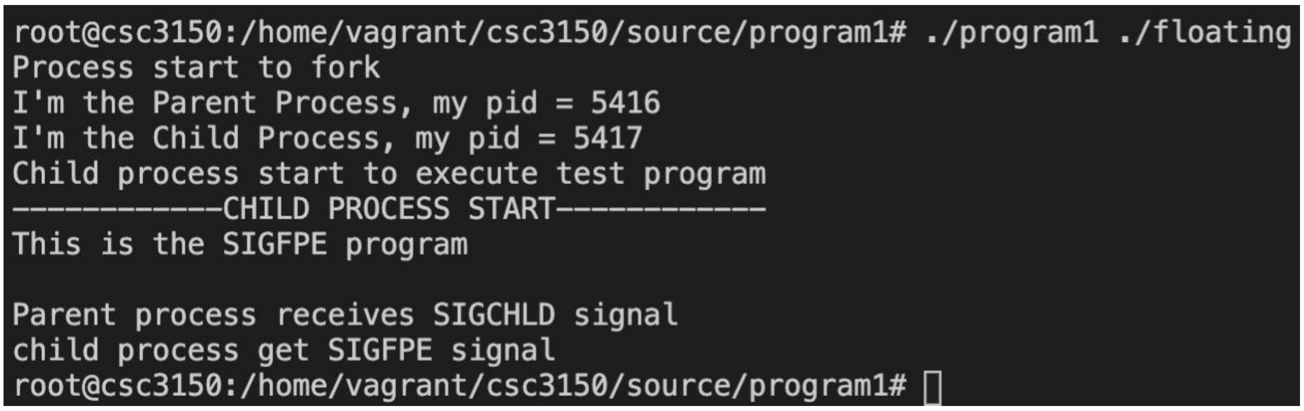
```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./alarm
Process start to fork
I'm the Parent Process, my pid = 5318
I'm the Child Process, my pid = 5319
Child process start to execute test program
------------CHILD PROCESS START------------
This is the SIGALRM program

Parent process receives SIGCHLD signal
child process get SIGALRM signal
root@csc3150:/home/vagrant/csc3150/source/program1# 
```

4. **bus**

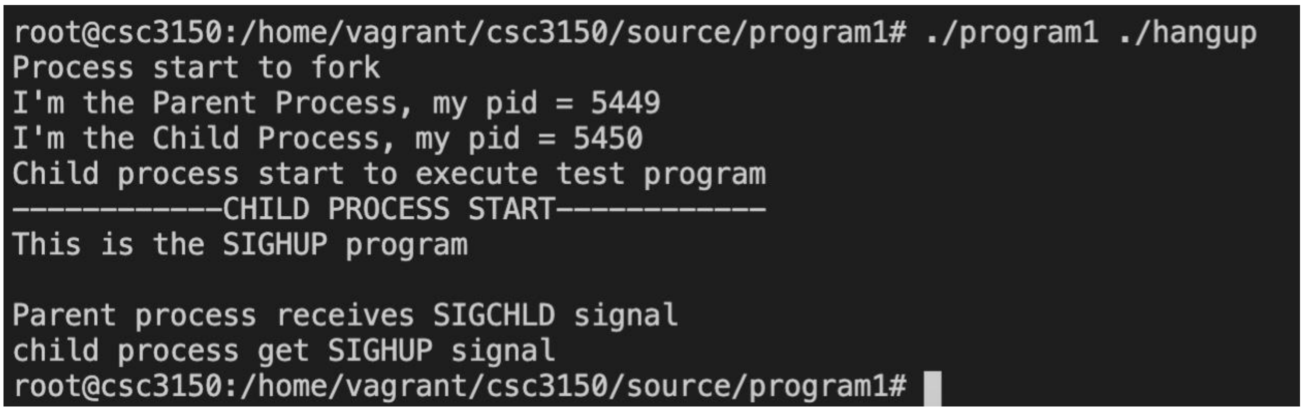


```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./bus
Process start to fork
I'm the Parent Process, my pid = 5389
I'm the Child Process, my pid = 5390
Child process start to execute test program
------------CHILD PROCESS START------------
This is the SIGBUS program

Parent process receives SIGCHLD signal
child process get SIGBUS signal
root@csc3150:/home/vagrant/csc3150/source/program1# 
```

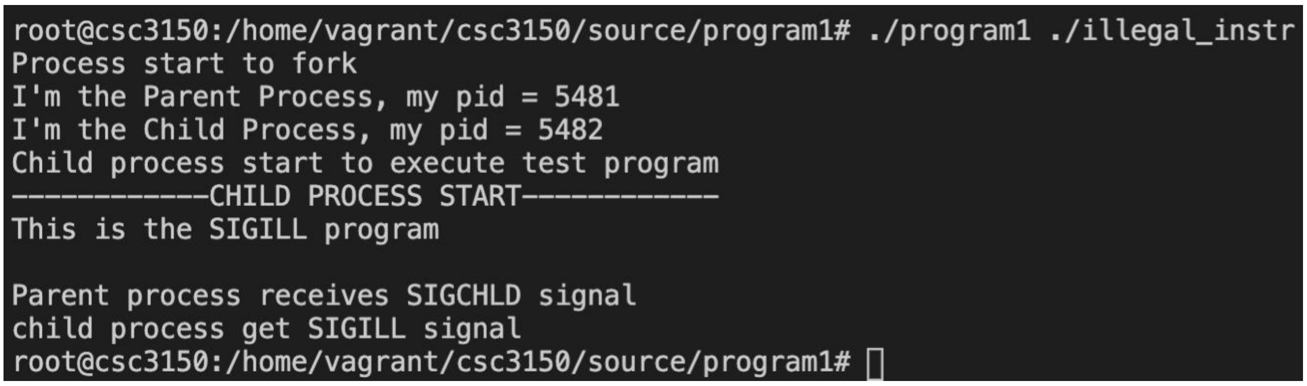
5. **floating**


```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./floating
Process start to fork
I'm the Parent Process, my pid = 5416
I'm the Child Process, my pid = 5417
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
root@csc3150:/home/vagrant/csc3150/source/program1# 
```

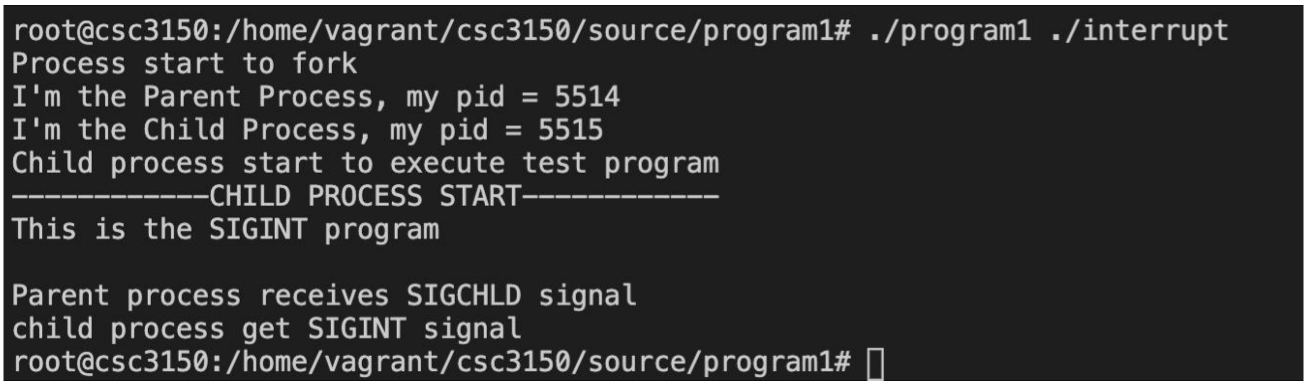
6. **hangup**


```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./hangup
Process start to fork
I'm the Parent Process, my pid = 5449
I'm the Child Process, my pid = 5450
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
root@csc3150:/home/vagrant/csc3150/source/program1# 
```

7. **illegal_instr**

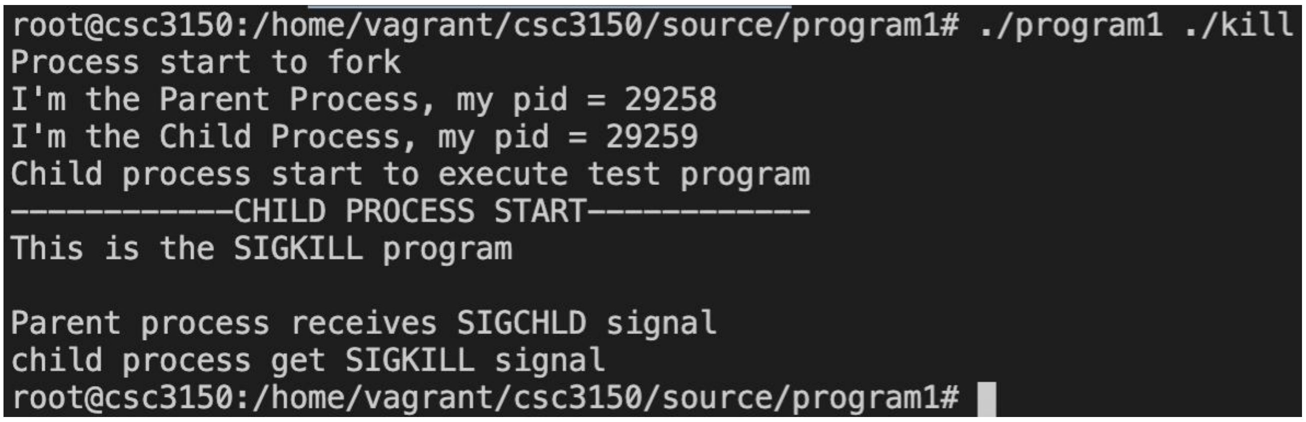

```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./illegal_instr
Process start to fork
I'm the Parent Process, my pid = 5481
I'm the Child Process, my pid = 5482
Child process start to execute test program
------------CHILD PROCESS START------------
This is the SIGILL program

Parent process receives SIGCHLD signal
child process get SIGILL signal
root@csc3150:/home/vagrant/csc3150/source/program1# 
```

8. **interrupt**



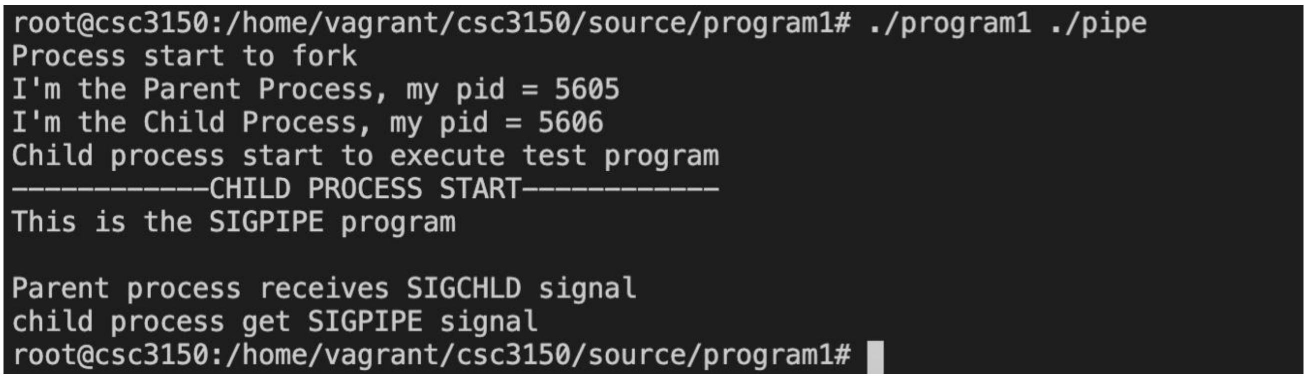
```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./interrupt
Process start to fork
I'm the Parent Process, my pid = 5514
I'm the Child Process, my pid = 5515
Child process start to execute test program
------------CHILD PROCESS START------------
This is the SIGINT program

Parent process receives SIGCHLD signal
child process get SIGINT signal
root@csc3150:/home/vagrant/csc3150/source/program1# 
```

9. **kill**



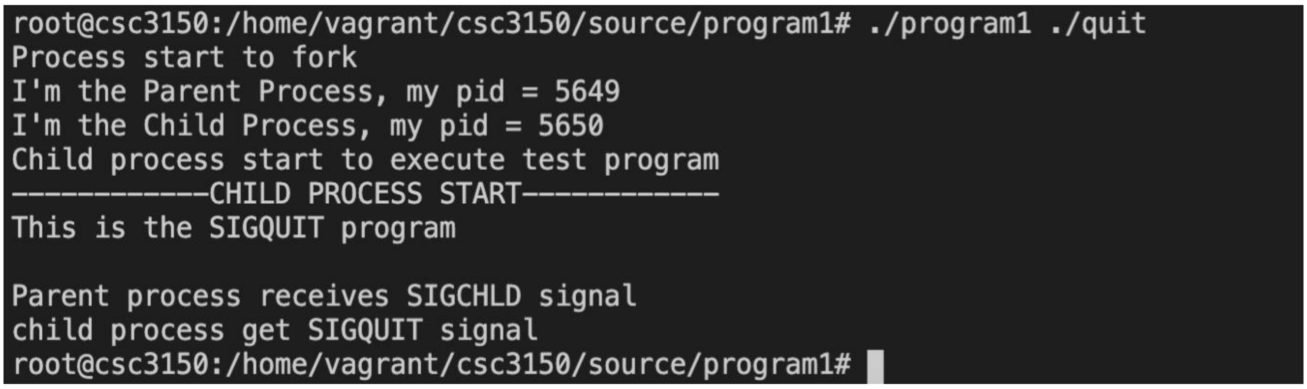
```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./kill
Process start to fork
I'm the Parent Process, my pid = 29258
I'm the Child Process, my pid = 29259
Child process start to execute test program
------------CHILD PROCESS START------------
This is the SIGKILL program

Parent process receives SIGCHLD signal
child process get SIGKILL signal
root@csc3150:/home/vagrant/csc3150/source/program1# 
```

10. **pipe**



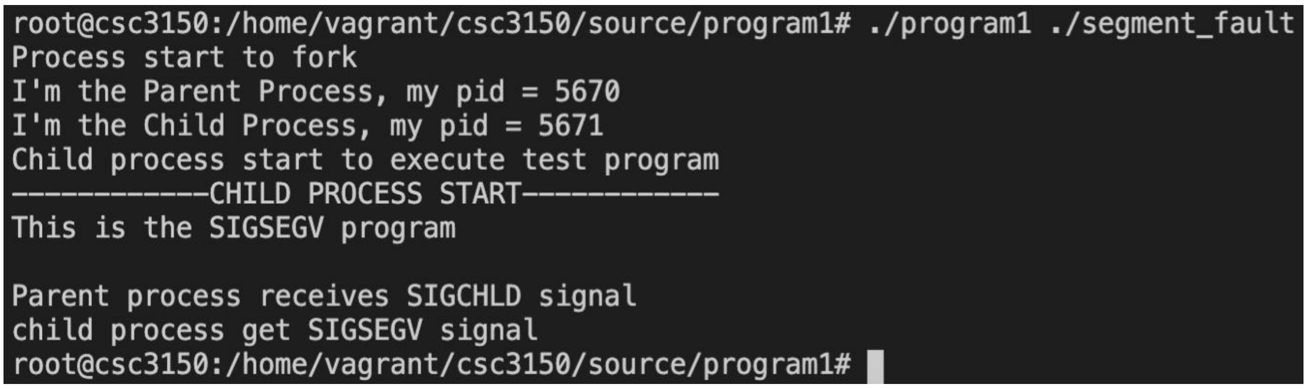
```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./pipe
Process start to fork
I'm the Parent Process, my pid = 5605
I'm the Child Process, my pid = 5606
Child process start to execute test program
------------CHILD PROCESS START------------
This is the SIGPIPE program

Parent process receives SIGCHLD signal
child process get SIGPIPE signal
root@csc3150:/home/vagrant/csc3150/source/program1# 
```

11. **quit**



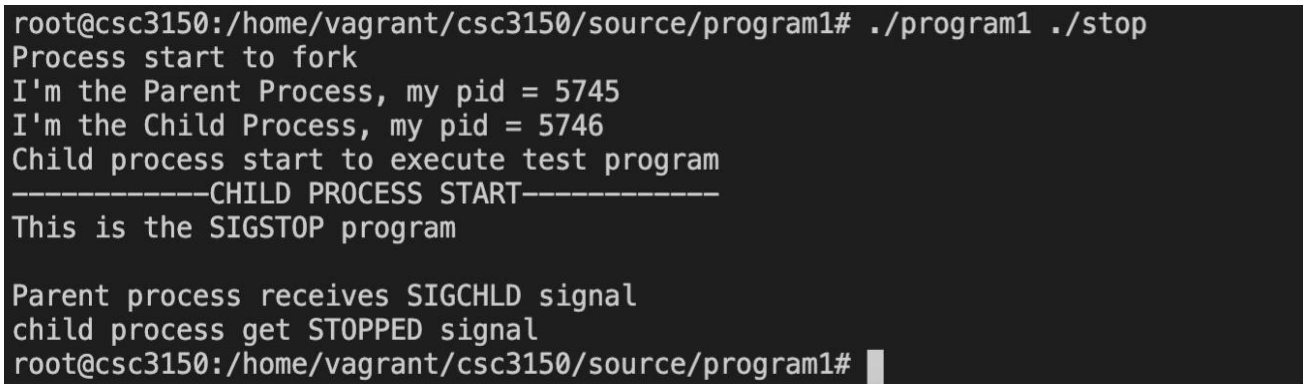
```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./quit
Process start to fork
I'm the Parent Process, my pid = 5649
I'm the Child Process, my pid = 5650
Child process start to execute test program
------------CHILD PROCESS START------------
This is the SIGQUIT program

Parent process receives SIGCHLD signal
child process get SIGQUIT signal
root@csc3150:/home/vagrant/csc3150/source/program1# 
```

12. **segment_fault**



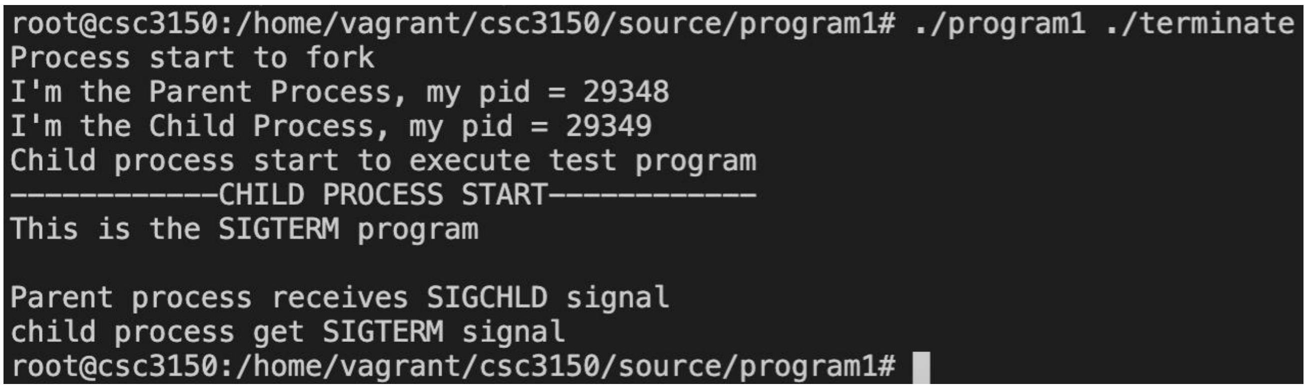
```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./segment_fault
Process start to fork
I'm the Parent Process, my pid = 5670
I'm the Child Process, my pid = 5671
Child process start to execute test program
------------CHILD PROCESS START------------
This is the SIGSEGV program

Parent process receives SIGCHLD signal
child process get SIGSEGV signal
root@csc3150:/home/vagrant/csc3150/source/program1# 
```

13. **stop**



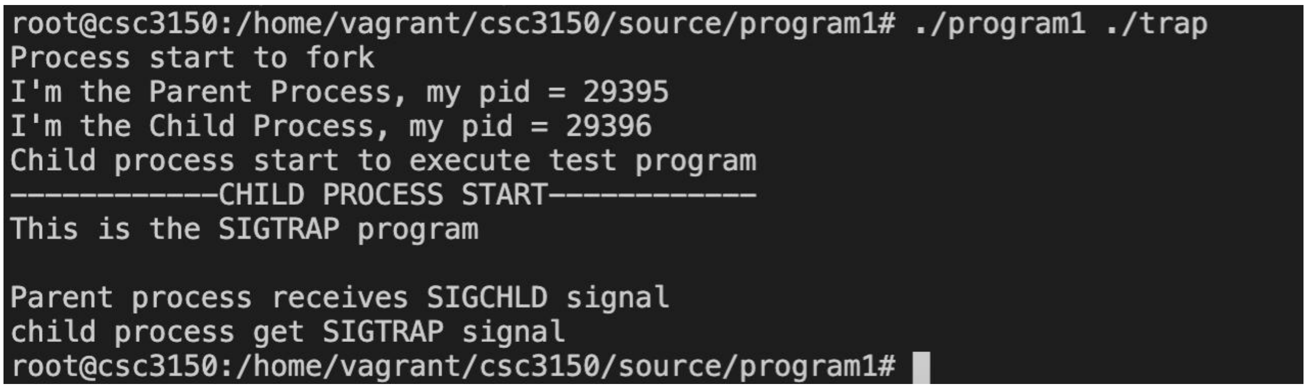
```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./stop
Process start to fork
I'm the Parent Process, my pid = 5745
I'm the Child Process, my pid = 5746
Child process start to execute test program
------------CHILD PROCESS START------------
This is the SIGSTOP program

Parent process receives SIGCHLD signal
child process get STOPPED signal
root@csc3150:/home/vagrant/csc3150/source/program1# 
```

14. **terminate**



```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./terminate
Process start to fork
I'm the Parent Process, my pid = 29348
I'm the Child Process, my pid = 29349
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
root@csc3150:/home/vagrant/csc3150/source/program1# 
```

15. **trap**



```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./trap
Process start to fork
I'm the Parent Process, my pid = 29395
I'm the Child Process, my pid = 29396
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
root@csc3150:/home/vagrant/csc3150/source/program1#
```

(2) Program 2

(a) Flow of the program 2

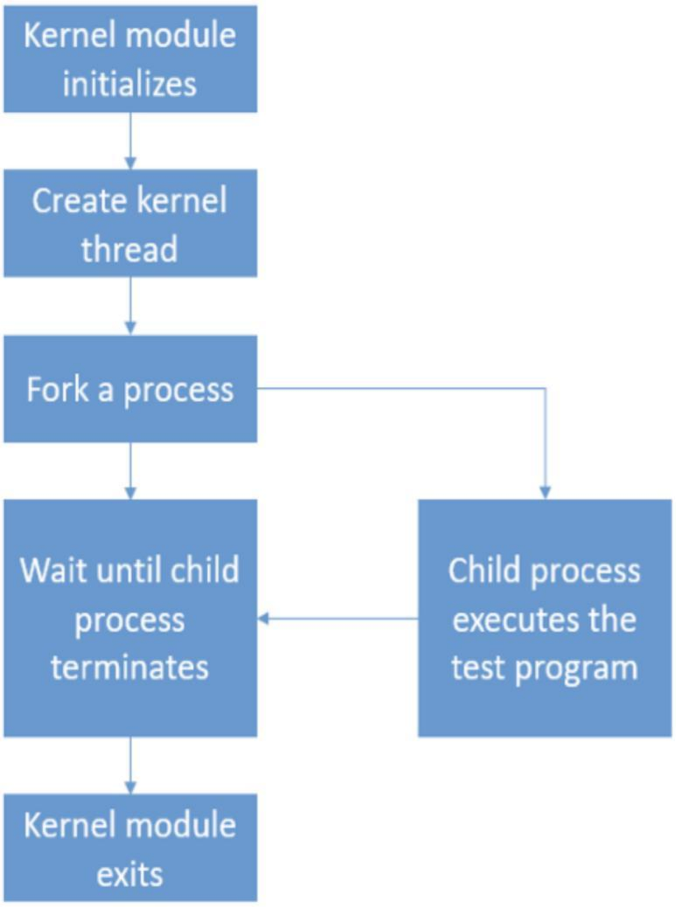

Kernel module initializes
Create kernel thread
Fork a process
Wait until child process terminates
Child process executes the test program
Kernel module exits

Figure 2. The flow chart of program 2

This program is to create a kernel thread. In the thread, the program fork a process to execute the another program. Then Parent process waits and receives the signals sent back from child process, and handle and output the information of the signal.

(b) Implementation

To begin with, we need to find and export 4 functions in kernel(“kernel_clone” (/kernel/fork.c) , “do_execve” (/fs/exec.c), “getname_kernel” (/fs/namei.c) and “do_wait”(/kernel/exit.c)) using EXPORT_SYMBOL . After modifications, we need to recompile the kernel. Also, we need to use extern to import them in program2.c. after recompiling the kernel, the program use kernel_clone( ) to call my_fork( ) which will fork a new process.

To implement my_fork( ), my_exec( ) and my_wait( ) need to be implemented in advance. The my_exec( ) use getname_kernel( ) to get the information of the test program by the its address. The it use do_execve( ) to execute the file. The my_wait( ) is used to get the signal sent back from child process using do_wait( ). After calling these 2 functions, my_fork( ) will handle the signal from child process and output its information.

(c) Execution steps of the program

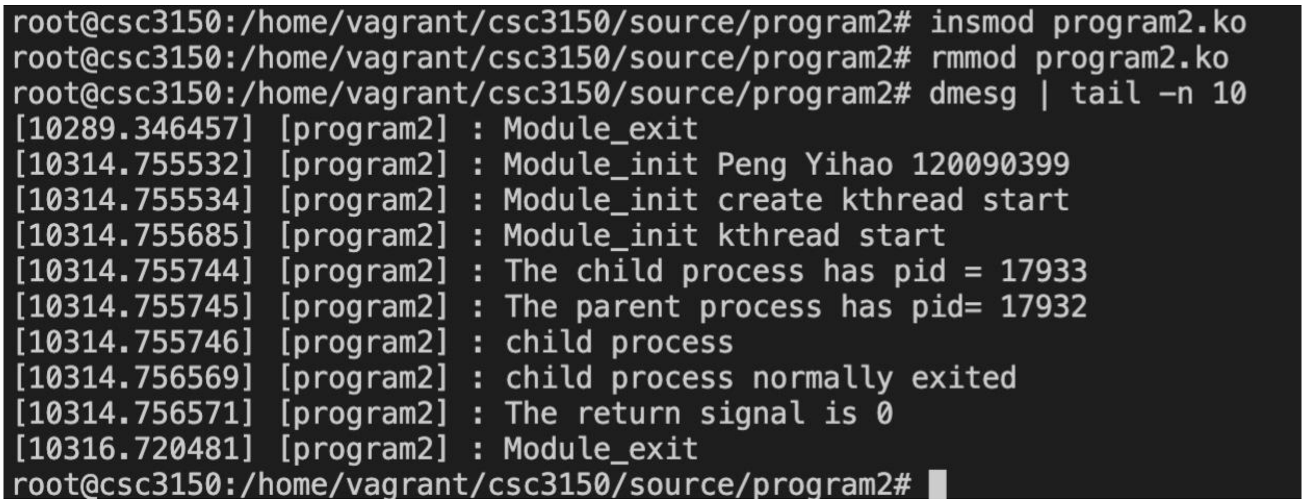
I. Modify the kernel files with EXPORT_SYMBOL to export “kernel_clone”, “do_execve”, “getname_kernel” and “do_wait” and rebuild the module and recompiled the kernel.

II. Enter the folder of program2 and enter make in the terminal to compile the code.

III. Type and enter in order “insmod program2.ko”, “rmmod program2.ko” in the terminal to insert and remove the module. Finally, using “dmesg | tail -n 10” to display the messages.

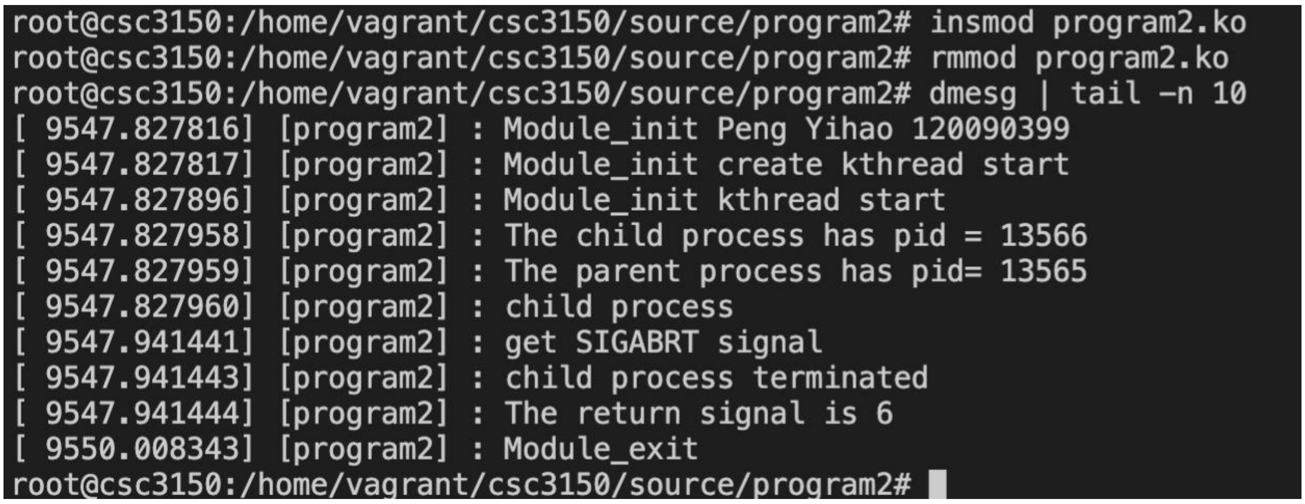
(d) Outputs of the program

1. normal



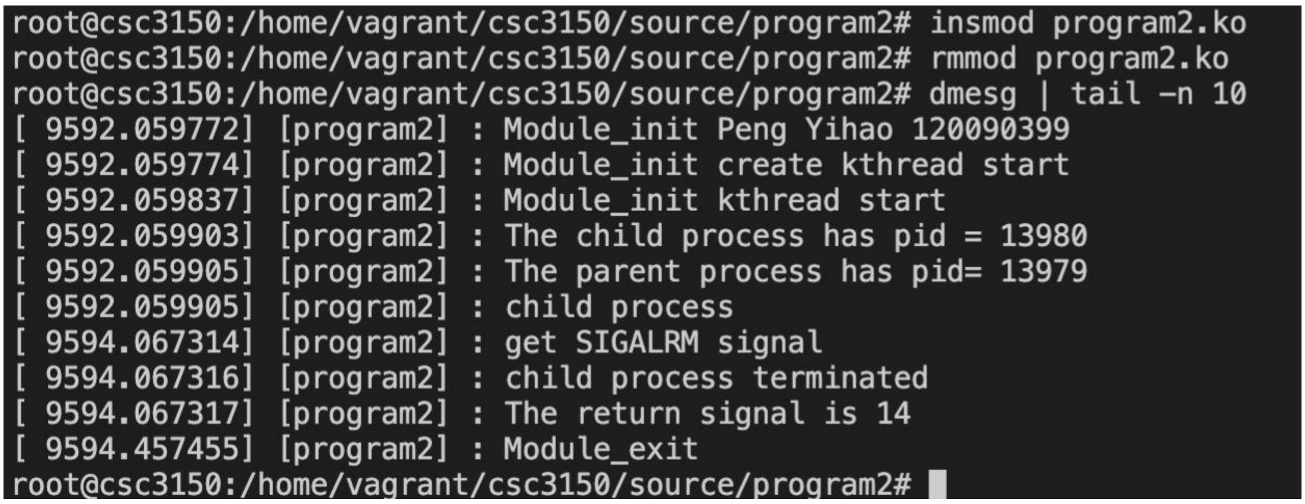
```
root@csc3150:/home/vagrant/csc3150/source/program2# insmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# dmesg | tail -n 10
[10289.346457] [program2] : Module_exit
[10314.755532] [program2] : Module_init Peng Yihao 120090399
[10314.755534] [program2] : Module_init create kthread start
[10314.755685] [program2] : Module_init kthread start
[10314.755744] [program2] : The child process has pid = 17933
[10314.755745] [program2] : The parent process has pid= 17932
[10314.755746] [program2] : child process
[10314.756569] [program2] : child process normally exited
[10314.756571] [program2] : The return signal is 0
[10316.720481] [program2] : Module_exit
root@csc3150:/home/vagrant/csc3150/source/program2# 
```

2. abort



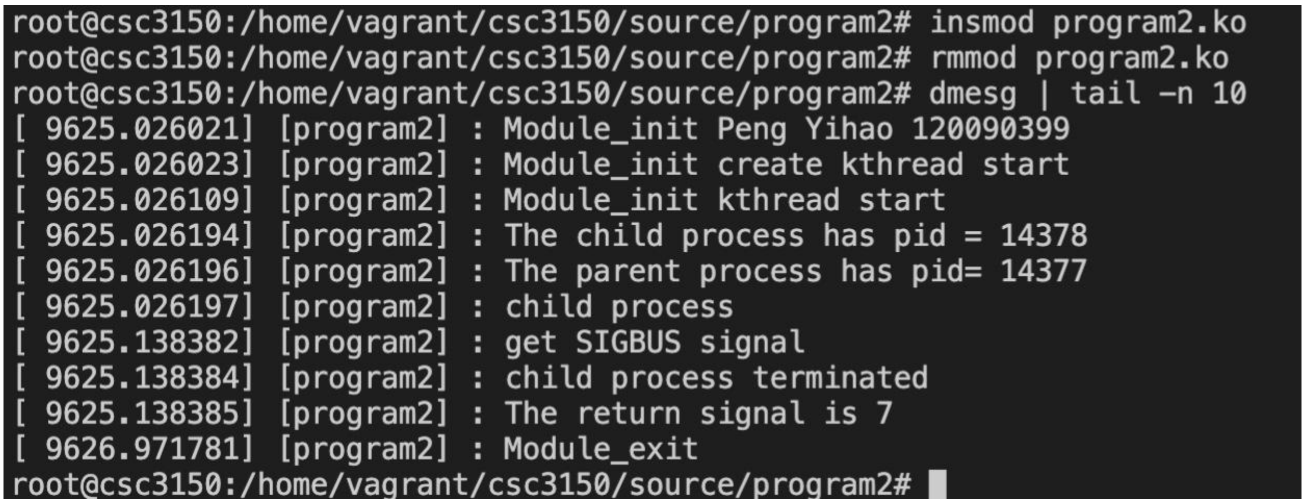
```
root@csc3150:/home/vagrant/csc3150/source/program2# insmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# dmesg | tail -n 10
[ 9547.827816] [program2] : Module_init Peng Yihao 120090399
[ 9547.827817] [program2] : Module_init create kthread start
[ 9547.827896] [program2] : Module_init kthread start
[ 9547.827958] [program2] : The child process has pid = 13566
[ 9547.827959] [program2] : The parent process has pid= 13565
[ 9547.827960] [program2] : child process
[ 9547.941441] [program2] : get SIGABRT signal
[ 9547.941443] [program2] : child process terminated
[ 9547.941444] [program2] : The return signal is 6
[ 9550.008343] [program2] : Module_exit
root@csc3150:/home/vagrant/csc3150/source/program2# 
```

3. alarm



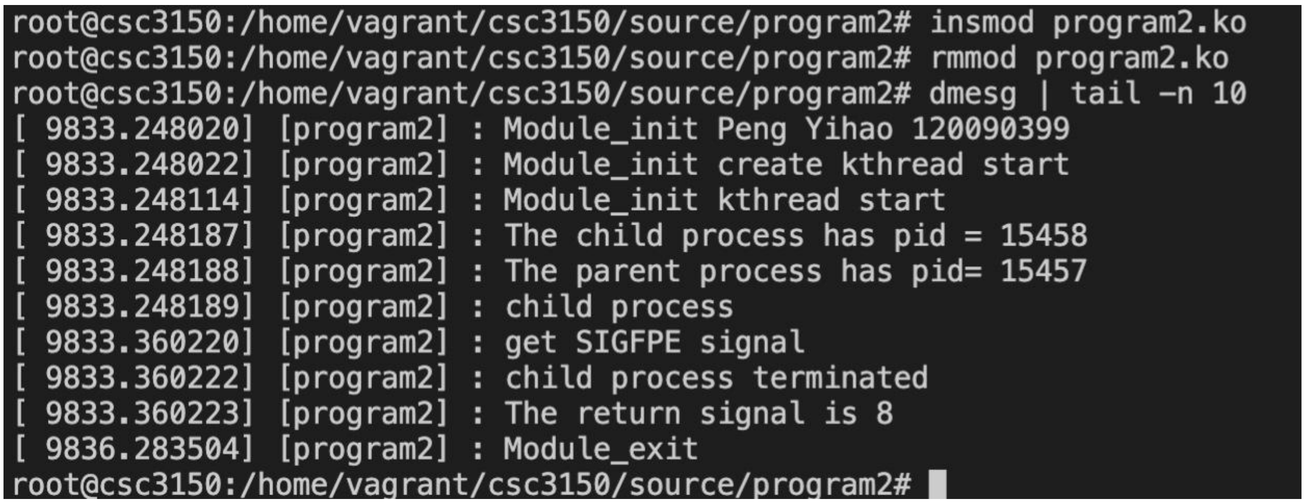
```
root@csc3150:/home/vagrant/csc3150/source/program2# insmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# dmesg | tail -n 10
[ 9592.059772] [program2] : Module_init Peng Yihao 120090399
[ 9592.059774] [program2] : Module_init create kthread start
[ 9592.059837] [program2] : Module_init kthread start
[ 9592.059903] [program2] : The child process has pid = 13980
[ 9592.059905] [program2] : The parent process has pid= 13979
[ 9592.059905] [program2] : child process
[ 9594.067314] [program2] : get SIGALRM signal
[ 9594.067316] [program2] : child process terminated
[ 9594.067317] [program2] : The return signal is 14
[ 9594.457455] [program2] : Module_exit
root@csc3150:/home/vagrant/csc3150/source/program2# 
```

4. **bus**



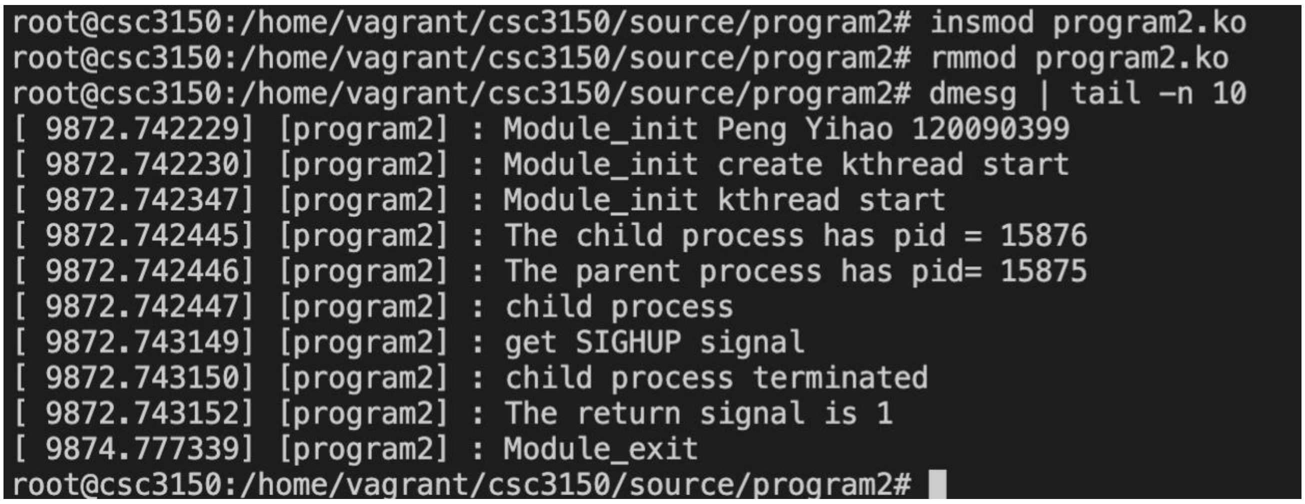
```
root@csc3150:/home/vagrant/csc3150/source/program2# insmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# dmesg | tail -n 10
[ 9625.026021] [program2] : Module_init Peng Yihao 120090399
[ 9625.026023] [program2] : Module_init create kthread start
[ 9625.026109] [program2] : Module_init kthread start
[ 9625.026194] [program2] : The child process has pid = 14378
[ 9625.026196] [program2] : The parent process has pid= 14377
[ 9625.026197] [program2] : child process
[ 9625.138382] [program2] : get SIGBUS signal
[ 9625.138384] [program2] : child process terminated
[ 9625.138385] [program2] : The return signal is 7
[ 9626.971781] [program2] : Module_exit
root@csc3150:/home/vagrant/csc3150/source/program2# 
```

5. **floating**



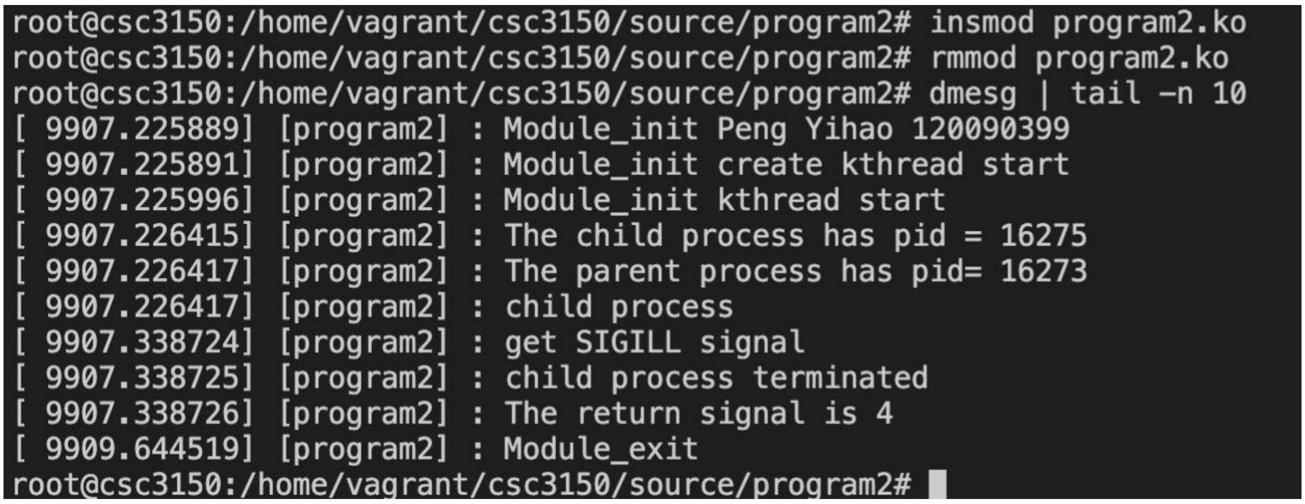
```
root@csc3150:/home/vagrant/csc3150/source/program2# insmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# dmesg | tail -n 10
[ 9833.248020] [program2] : Module_init Peng Yihao 120090399
[ 9833.248022] [program2] : Module_init create kthread start
[ 9833.248114] [program2] : Module_init kthread start
[ 9833.248187] [program2] : The child process has pid = 15458
[ 9833.248188] [program2] : The parent process has pid= 15457
[ 9833.248189] [program2] : child process
[ 9833.360220] [program2] : get SIGFPE signal
[ 9833.360222] [program2] : child process terminated
[ 9833.360223] [program2] : The return signal is 8
[ 9836.283504] [program2] : Module_exit
root@csc3150:/home/vagrant/csc3150/source/program2# 
```

6. **hangup**



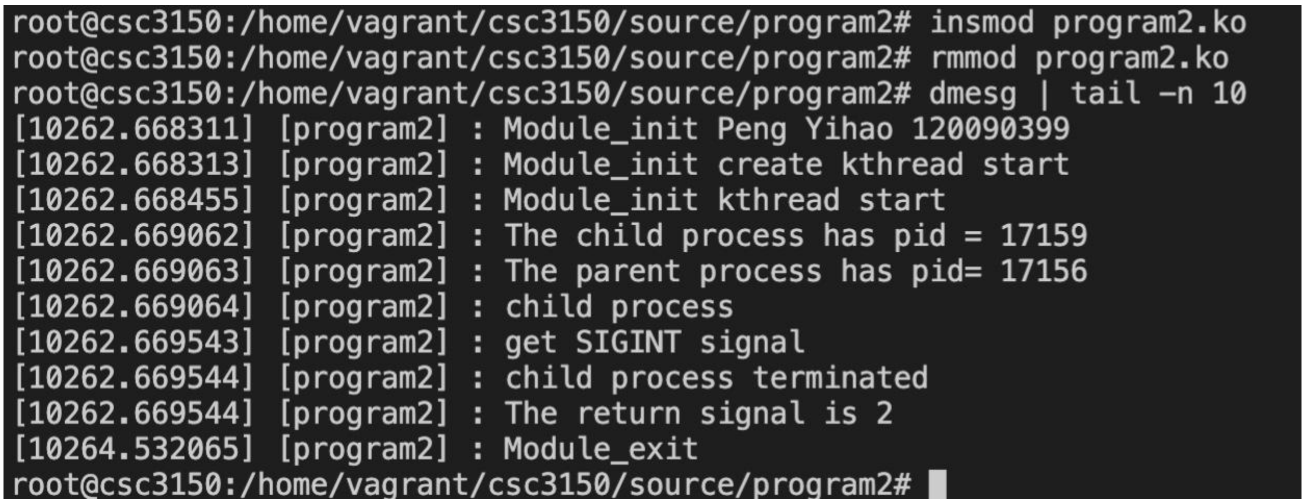
```
root@csc3150:/home/vagrant/csc3150/source/program2# insmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# dmesg | tail -n 10
[ 9872.742229] [program2] : Module_init Peng Yihao 120090399
[ 9872.742230] [program2] : Module_init create kthread start
[ 9872.742347] [program2] : Module_init kthread start
[ 9872.742445] [program2] : The child process has pid = 15876
[ 9872.742446] [program2] : The parent process has pid= 15875
[ 9872.742447] [program2] : child process
[ 9872.743149] [program2] : get SIGHUP signal
[ 9872.743150] [program2] : child process terminated
[ 9872.743152] [program2] : The return signal is 1
[ 9874.777339] [program2] : Module_exit
root@csc3150:/home/vagrant/csc3150/source/program2# 
```

7. **illegal_instr**



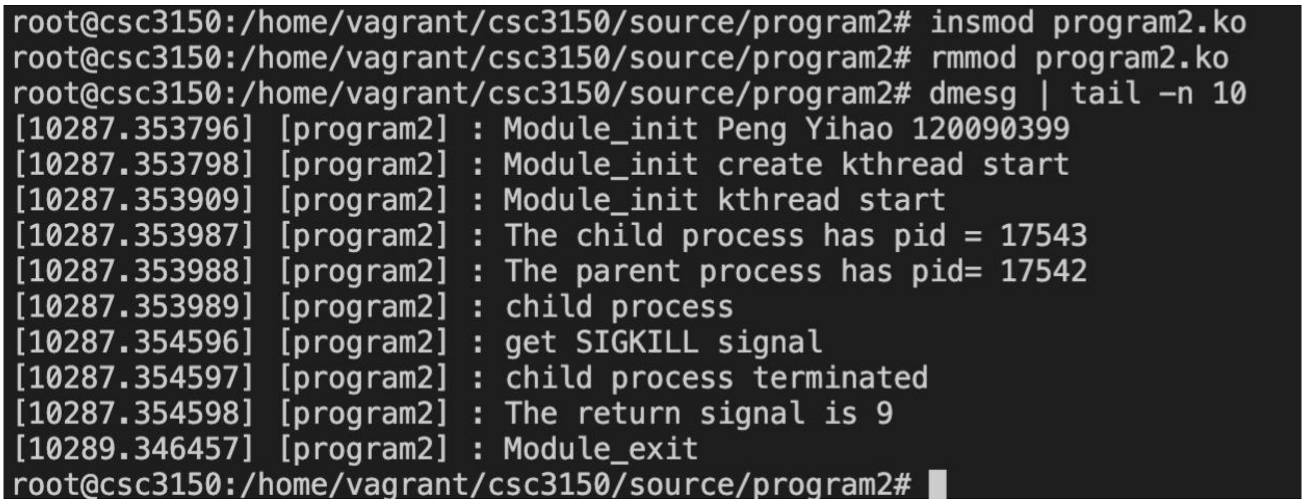
```
root@csc3150:/home/vagrant/csc3150/source/program2# insmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# dmesg | tail -n 10
[ 9907.225889] [program2] : Module_init Peng Yihao 120090399
[ 9907.225891] [program2] : Module_init create kthread start
[ 9907.225996] [program2] : Module_init kthread start
[ 9907.226415] [program2] : The child process has pid = 16275
[ 9907.226417] [program2] : The parent process has pid= 16273
[ 9907.226417] [program2] : child process
[ 9907.338724] [program2] : get SIGILL signal
[ 9907.338725] [program2] : child process terminated
[ 9907.338726] [program2] : The return signal is 4
[ 9909.644519] [program2] : Module_exit
root@csc3150:/home/vagrant/csc3150/source/program2# 
```

8. **interrupt**



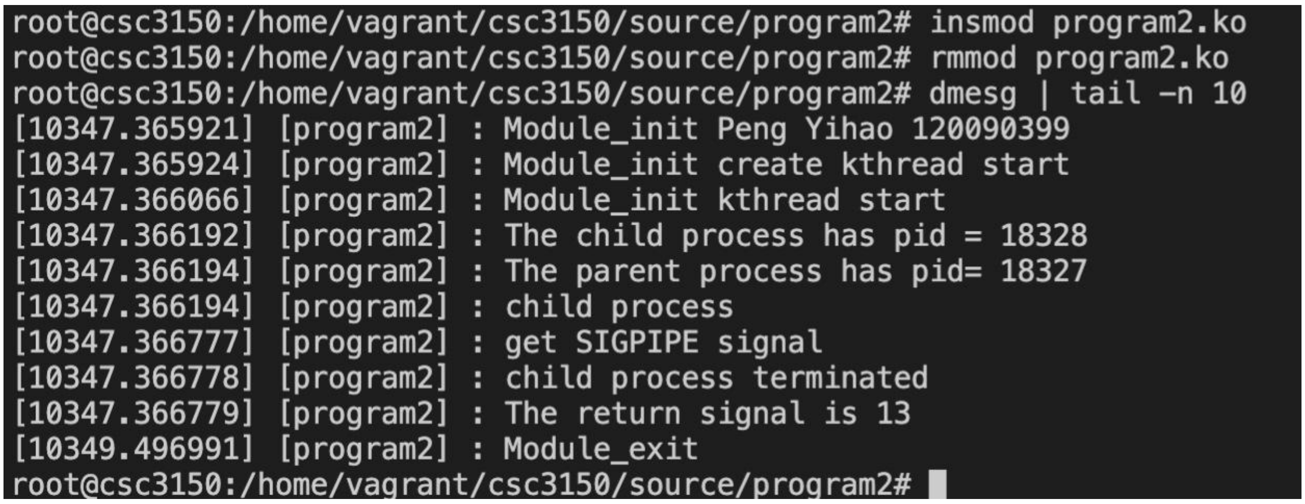
```
root@csc3150:/home/vagrant/csc3150/source/program2# insmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# dmesg | tail -n 10
[10262.668311] [program2] : Module_init Peng Yihao 120090399
[10262.668313] [program2] : Module_init create kthread start
[10262.668455] [program2] : Module_init kthread start
[10262.669062] [program2] : The child process has pid = 17159
[10262.669063] [program2] : The parent process has pid= 17156
[10262.669064] [program2] : child process
[10262.669543] [program2] : get SIGINT signal
[10262.669544] [program2] : child process terminated
[10262.669544] [program2] : The return signal is 2
[10264.532065] [program2] : Module_exit
root@csc3150:/home/vagrant/csc3150/source/program2# 
```

9. **kill**



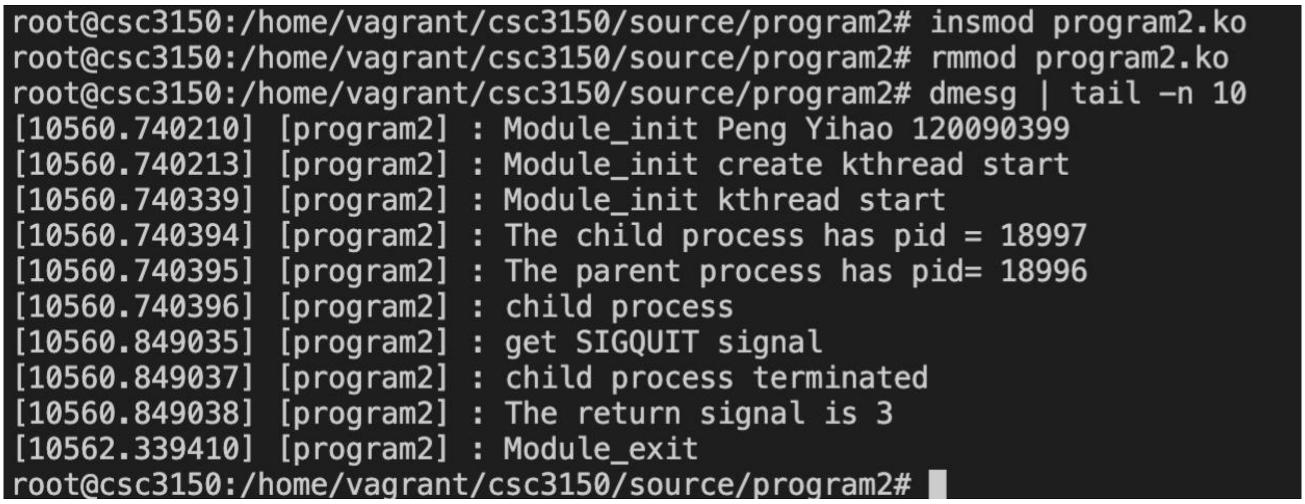
```
root@csc3150:/home/vagrant/csc3150/source/program2# insmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# dmesg | tail -n 10
[10287.353796] [program2] : Module_init Peng Yihao 120090399
[10287.353798] [program2] : Module_init create kthread start
[10287.353909] [program2] : Module_init kthread start
[10287.353987] [program2] : The child process has pid = 17543
[10287.353988] [program2] : The parent process has pid= 17542
[10287.353989] [program2] : child process
[10287.354596] [program2] : get SIGKILL signal
[10287.354597] [program2] : child process terminated
[10287.354598] [program2] : The return signal is 9
[10289.346457] [program2] : Module_exit
root@csc3150:/home/vagrant/csc3150/source/program2# 
```

10. **pipe**



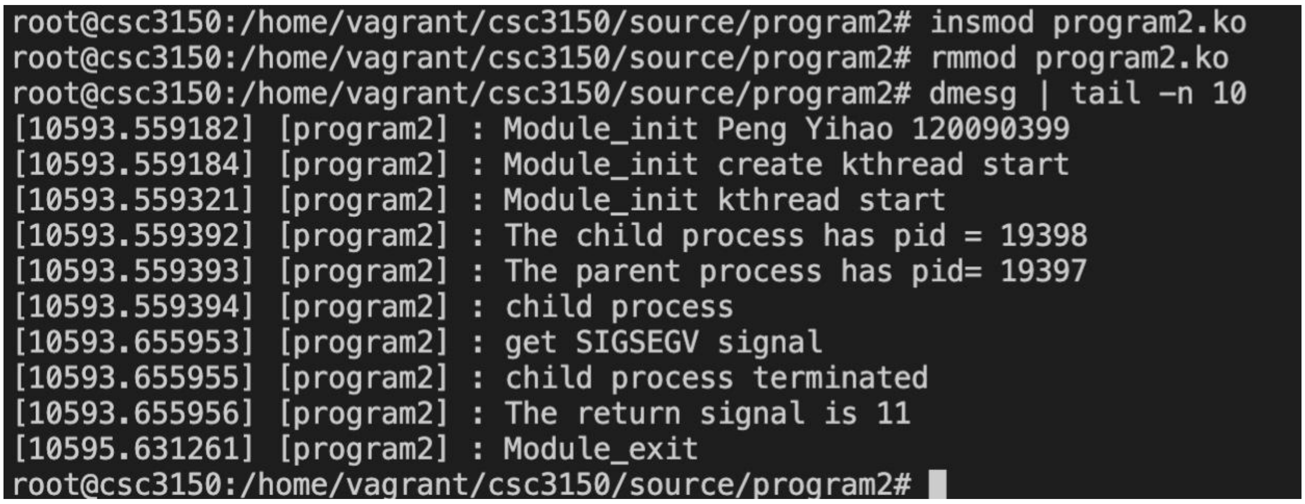
```
root@csc3150:/home/vagrant/csc3150/source/program2# insmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# dmesg | tail -n 10
[10347.365921] [program2] : Module_init Peng Yihao 120090399
[10347.365924] [program2] : Module_init create kthread start
[10347.366066] [program2] : Module_init kthread start
[10347.366192] [program2] : The child process has pid = 18328
[10347.366194] [program2] : The parent process has pid= 18327
[10347.366194] [program2] : child process
[10347.366777] [program2] : get SIGPIPE signal
[10347.366778] [program2] : child process terminated
[10347.366779] [program2] : The return signal is 13
[10349.496991] [program2] : Module_exit
root@csc3150:/home/vagrant/csc3150/source/program2# 
```

11. **quit**

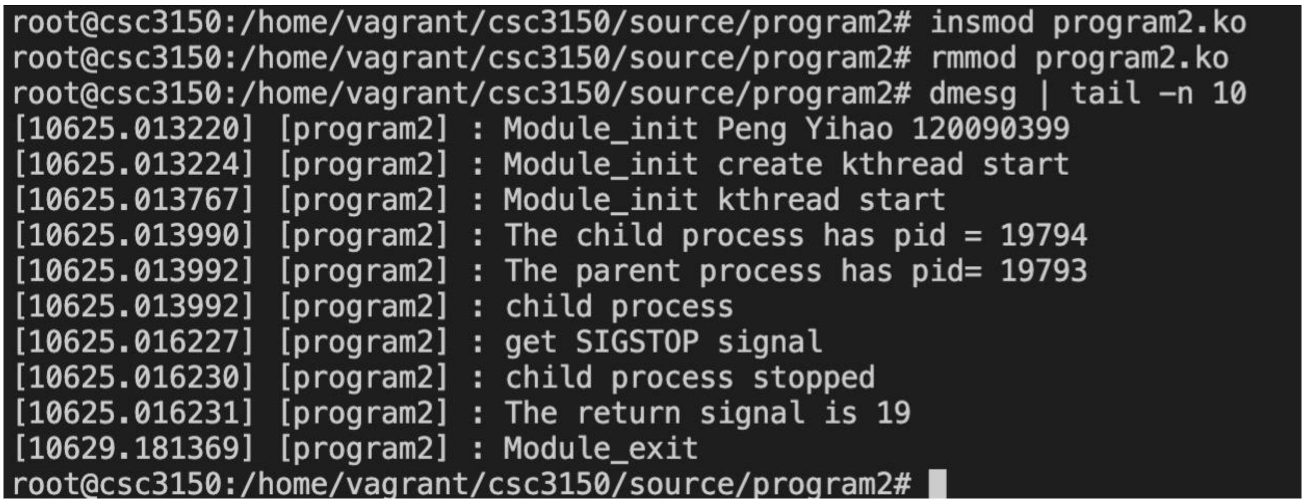


```
root@csc3150:/home/vagrant/csc3150/source/program2# insmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# dmesg | tail -n 10
[10560.740210] [program2] : Module_init Peng Yihao 120090399
[10560.740213] [program2] : Module_init create kthread start
[10560.740339] [program2] : Module_init kthread start
[10560.740394] [program2] : The child process has pid = 18997
[10560.740395] [program2] : The parent process has pid= 18996
[10560.740396] [program2] : child process
[10560.849035] [program2] : get SIGQUIT signal
[10560.849037] [program2] : child process terminated
[10560.849038] [program2] : The return signal is 3
[10562.339410] [program2] : Module_exit
root@csc3150:/home/vagrant/csc3150/source/program2# 
```

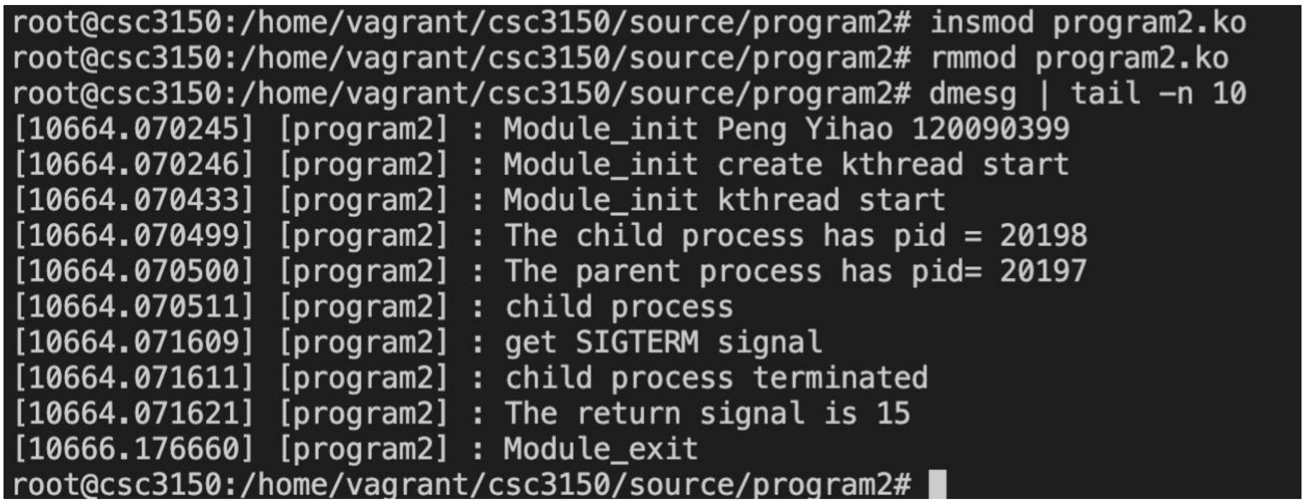
12. **Segment_fault**


```
root@csc3150:/home/vagrant/csc3150/source/program2# insmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# dmesg | tail -n 10
[10593.559182] [program2] : Module_init Peng Yihao 120090399
[10593.559184] [program2] : Module_init create kthread start
[10593.559321] [program2] : Module_init kthread start
[10593.559392] [program2] : The child process has pid = 19398
[10593.559393] [program2] : The parent process has pid= 19397
[10593.559394] [program2] : child process
[10593.655953] [program2] : get SIGSEGV signal
[10593.655955] [program2] : child process terminated
[10593.655956] [program2] : The return signal is 11
[10595.631261] [program2] : Module_exit
root@csc3150:/home/vagrant/csc3150/source/program2#
```

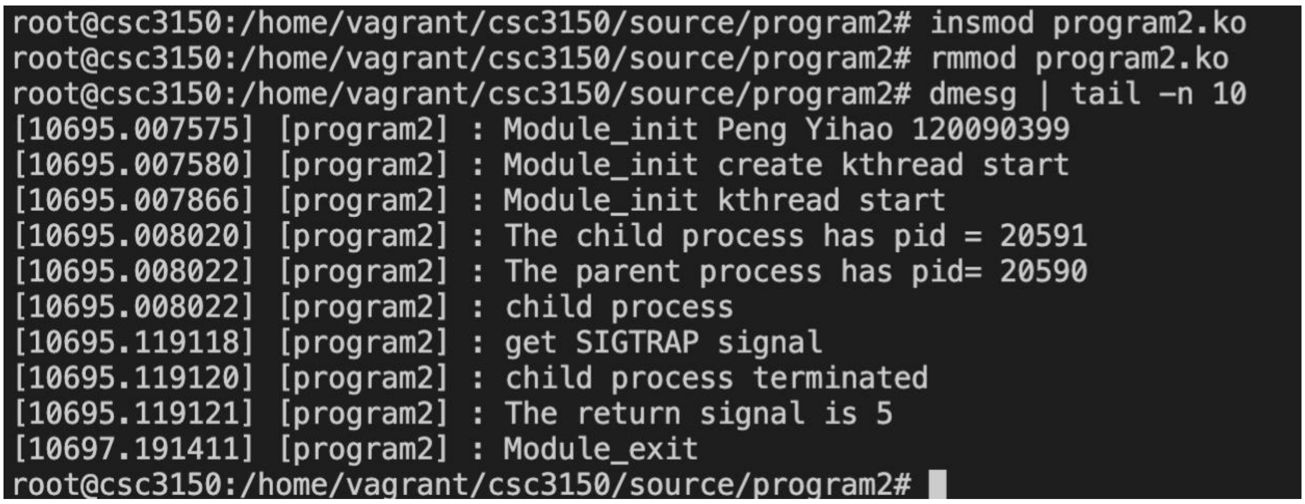
13. **stop**


```
root@csc3150:/home/vagrant/csc3150/source/program2# insmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# dmesg | tail -n 10
[10625.013220] [program2] : Module_init Peng Yihao 120090399
[10625.013224] [program2] : Module_init create kthread start
[10625.013767] [program2] : Module_init kthread start
[10625.013990] [program2] : The child process has pid = 19794
[10625.013992] [program2] : The parent process has pid= 19793
[10625.013992] [program2] : child process
[10625.016227] [program2] : get SIGSTOP signal
[10625.016230] [program2] : child process stopped
[10625.016231] [program2] : The return signal is 19
[10629.181369] [program2] : Module_exit
root@csc3150:/home/vagrant/csc3150/source/program2#
```

14. **terminate**


```
root@csc3150:/home/vagrant/csc3150/source/program2# insmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# dmesg | tail -n 10
[10664.070245] [program2] : Module_init Peng Yihao 120090399
[10664.070246] [program2] : Module_init create kthread start
[10664.070433] [program2] : Module_init kthread start
[10664.070499] [program2] : The child process has pid = 20198
[10664.070500] [program2] : The parent process has pid= 20197
[10664.070511] [program2] : child process
[10664.071609] [program2] : get SIGTERM signal
[10664.071611] [program2] : child process terminated
[10664.071621] [program2] : The return signal is 15
[10666.176660] [program2] : Module_exit
root@csc3150:/home/vagrant/csc3150/source/program2#
```

15. **trap**



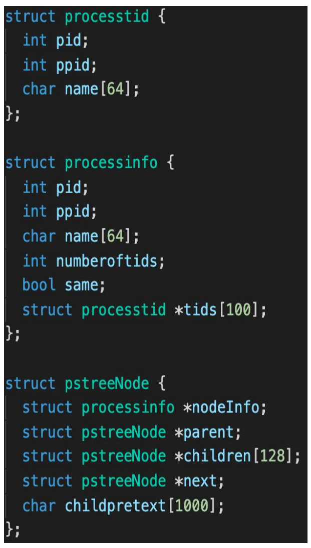
```
root@csc3150:/home/vagrant/csc3150/source/program2# insmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/source/program2# dmesg | tail -n 10
[10695.007575] [program2] : Module_init Peng Yihao 120090399
[10695.007580] [program2] : Module_init create kthread start
[10695.007866] [program2] : Module_init kthread start
[10695.008020] [program2] : The child process has pid = 20591
[10695.008022] [program2] : The parent process has pid= 20590
[10695.008022] [program2] : child process
[10695.119118] [program2] : get SIGTRAP signal
[10695.119120] [program2] : child process terminated
[10695.119121] [program2] : The return signal is 5
[10697.191411] [program2] : Module_exit
root@csc3150:/home/vagrant/csc3150/source/program2# 
```

(3) Bonus

(a) Ideas of the program

This program is to implement the linux command - pstree that could print out the process tree of linux. First, we need to get the information (name, pid, ppid, threads) of all processes. Then we need to use linked list to assemble all process as nodes to form a process tree. Finally, we need print them from parent node to child node in the format of pstree.

(b) Implementation



```
struct processtid {
  int pid;
  int ppid;
  char name[64];
};

struct processinfo {
  int pid;
  int ppid;
  char name[64];
  int numberoftids;
  bool same;
  struct processtid *tids[100];
};

struct pstreeNode {
  struct processinfo *nodeInfo;
  struct pstreeNode *parent;
  struct pstreeNode *children[128];
  struct pstreeNode *next;
  char childpretext[1000];
};
```

Firstly, we create three kind of structures to store the threads information of a process, the information of a process and the node of process tree respectively. Then the program use getprocessInfo( ) function to set the path of the information files of process in the “/proc/{pid}/status”, and call getpInfos(char *filename) to extract the information of each process. Also, it sets path of the information files of threads of a process in the “/proc/{pid}/task/{tid}/status” and call gettidInfos(char *filename) to extract the information of threads of a process. In these functions, we store information in their corresponding data structure. Then we use create_pstree( ) to create the process tree with processes as nodes.

After extracting the information of all process and creating the process linked list tree, the program use print_pstree(struct pstreeNode *root, int type, char text[], int indexOftid) to print processes’ information in the format of pstree. The main idea of this function is that divides the nodes to be printed into different types, such as the head of subtree or the last child process. Then according to the type of process node, the program print them in different ways to fulfill the format of pstree.

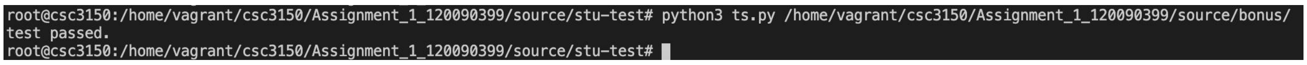
(c) Execution steps of the program

I. Enter the folder of the program and enter make in terminal.

II. First, type “./pstree” to see the printed pstree without any option. Then add some option to run the file, such as “./pstree -c”, “./pstree -p”, “./pstree -V”. The program will print different pstree in corresponding option.

(d) Supplementary information

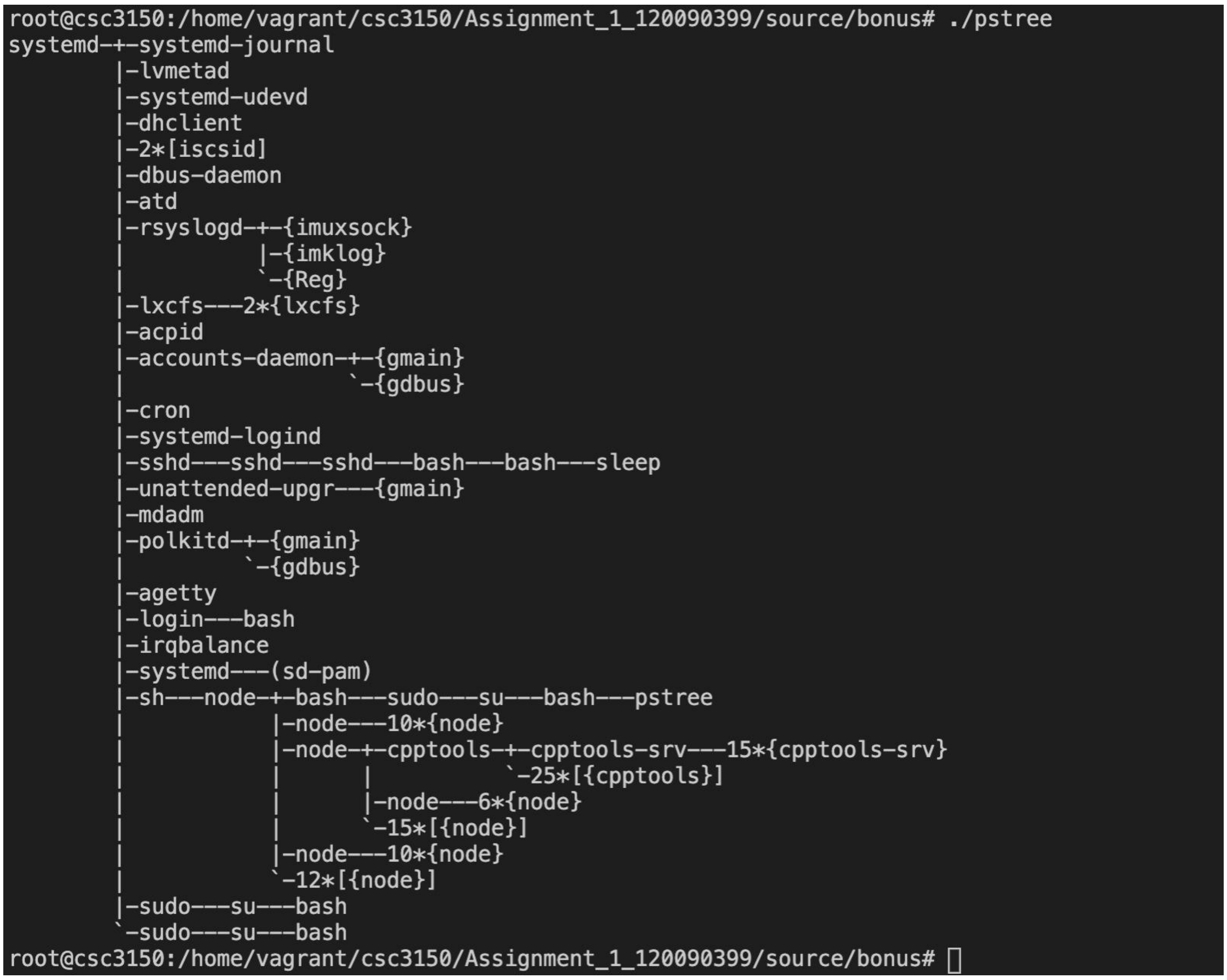
Due to the limitation of lack of some symbols of the extended ASCII in linux, the program use symbols in the standard ASCII table to replace those special symbols. The output format is the same as the pstree. And my program pass the stu-test.



```
root@csc3150:/home/vagrant/csc3150/Assignment_1_120090399/source/stu-test# python3 ts.py /home/vagrant/csc3150/Assignment_1_120090399/source/bonus/
test passed.
root@csc3150:/home/vagrant/csc3150/Assignment_1_120090399/source/stu-test#
```

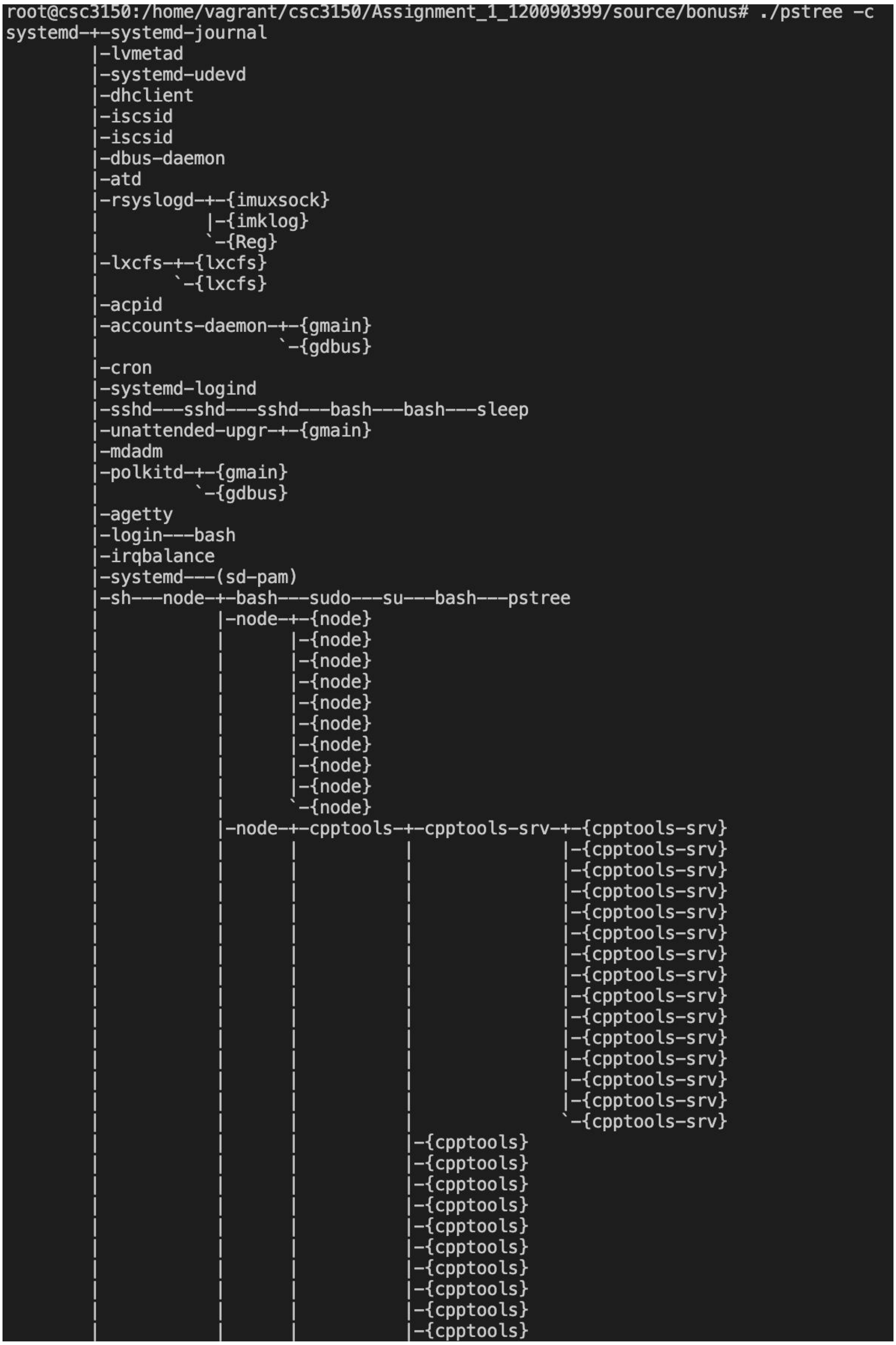
(e) Sample output:

1. **./pstree**

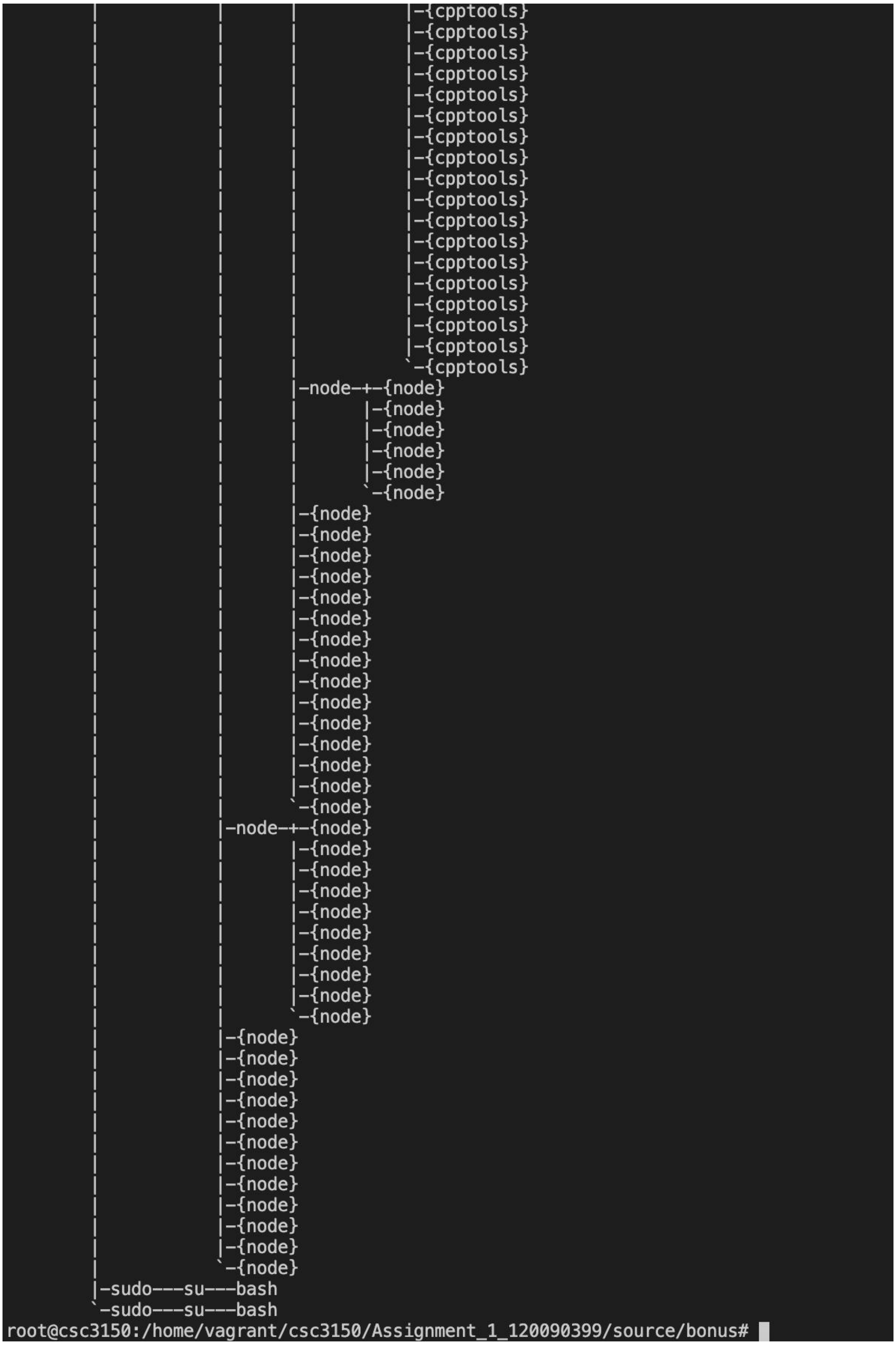


```
root@csc3150:/home/vagrant/csc3150/Assignment_1_120090399/source/bonus# ./pstree
systemd-+-systemd-journal
        |-lvmetad
        |-systemd-udevd
        |-dhclient
        |-2*[iscsid]
        |-dbus-daemon
        |-atd
        |-rsyslogd-+-{imuxsock}
        |          |-{imklog}
        |          `-{Reg}
        |-lxcfs---2*{lxcfs}
        |-acpid
        |-accounts-daemon-+-{gmain}
        |                 `-{gdbus}
        |-cron
        |-systemd-logind
        |-sshd---sshd---sshd---bash---bash---sleep
        |-unattended-upgr---{gmain}
        |-mdadm
        |-polkitd-+-{gmain}
        |         `-{gdbus}
        |-agetty
        |-login---bash
        |-irqbalance
        |-systemd---(sd-pam)
        |-sh---node-+-bash---sudo---su---bash---pstree
        |           |-node---10*{node}
        |           |-node-+-cpptools-+-cpptools-srv---15*{cpptools-srv}
        |           |      |          `-25*[{cpptools}]
        |           |      |-node---6*{node}
        |           |      `-15*[{node}]
        |           |-node---10*{node}
        |           `-12*[{node}]
        |-sudo---su---bash
        `-sudo---su---bash
root@csc3150:/home/vagrant/csc3150/Assignment_1_120090399/source/bonus#
```

2. **./pstree -c**

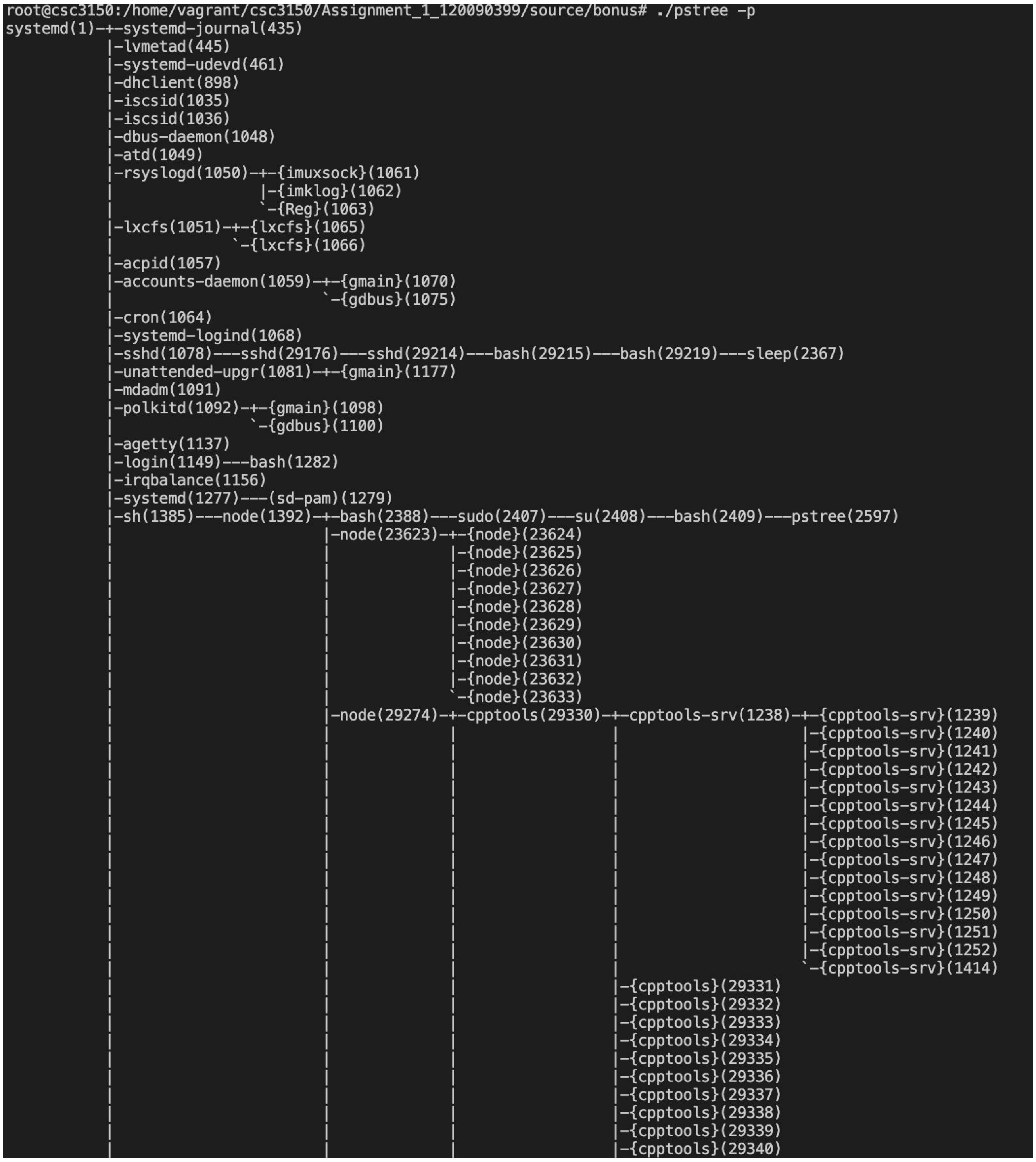


```
root@csc3150:/home/vagrant/csc3150/Assignment_1_120090399/source/bonus# ./pstree -c
systemd-+-systemd-journal
        |-lvmetad
        |-systemd-udevd
        |-dhclient
        |-iscsid
        |-iscsid
        |-dbus-daemon
        |-atd
        |-rsyslogd-+-{imuxsock}
        |          |-{imklog}
        |          `-{Reg}
        |-lxcfs-+-{lxcfs}
        |       `-{lxcfs}
        |-acpid
        |-accounts-daemon-+-{gmain}
        |                 `-{gdbus}
        |-cron
        |-systemd-logind
        |-sshd---sshd---sshd---bash---bash---sleep
        |-unattended-upgr-+-{gmain}
        |-mdadm
        |-polkitd-+-{gmain}
        |         `-{gdbus}
        |-agetty
        |-login---bash
        |-irqbalance
        |-systemd---(sd-pam)
        |-sh---node-+-bash---sudo---su---bash---pstree
        |           |-node-+-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      `-{node}
        |           |-node-+-cpptools-+-cpptools-srv-+-{cpptools-srv}
        |           |      |          |              |-{cpptools-srv}
        |           |      |          |              |-{cpptools-srv}
        |           |      |          |              |-{cpptools-srv}
        |           |      |          |              |-{cpptools-srv}
        |           |      |          |              |-{cpptools-srv}
        |           |      |          |              |-{cpptools-srv}
        |           |      |          |              |-{cpptools-srv}
        |           |      |          |              |-{cpptools-srv}
        |           |      |          |              |-{cpptools-srv}
        |           |      |          |              |-{cpptools-srv}
        |           |      |          |              |-{cpptools-srv}
        |           |      |          |              |-{cpptools-srv}
        |           |      |          |              |-{cpptools-srv}
        |           |      |          |              `-{cpptools-srv}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
```

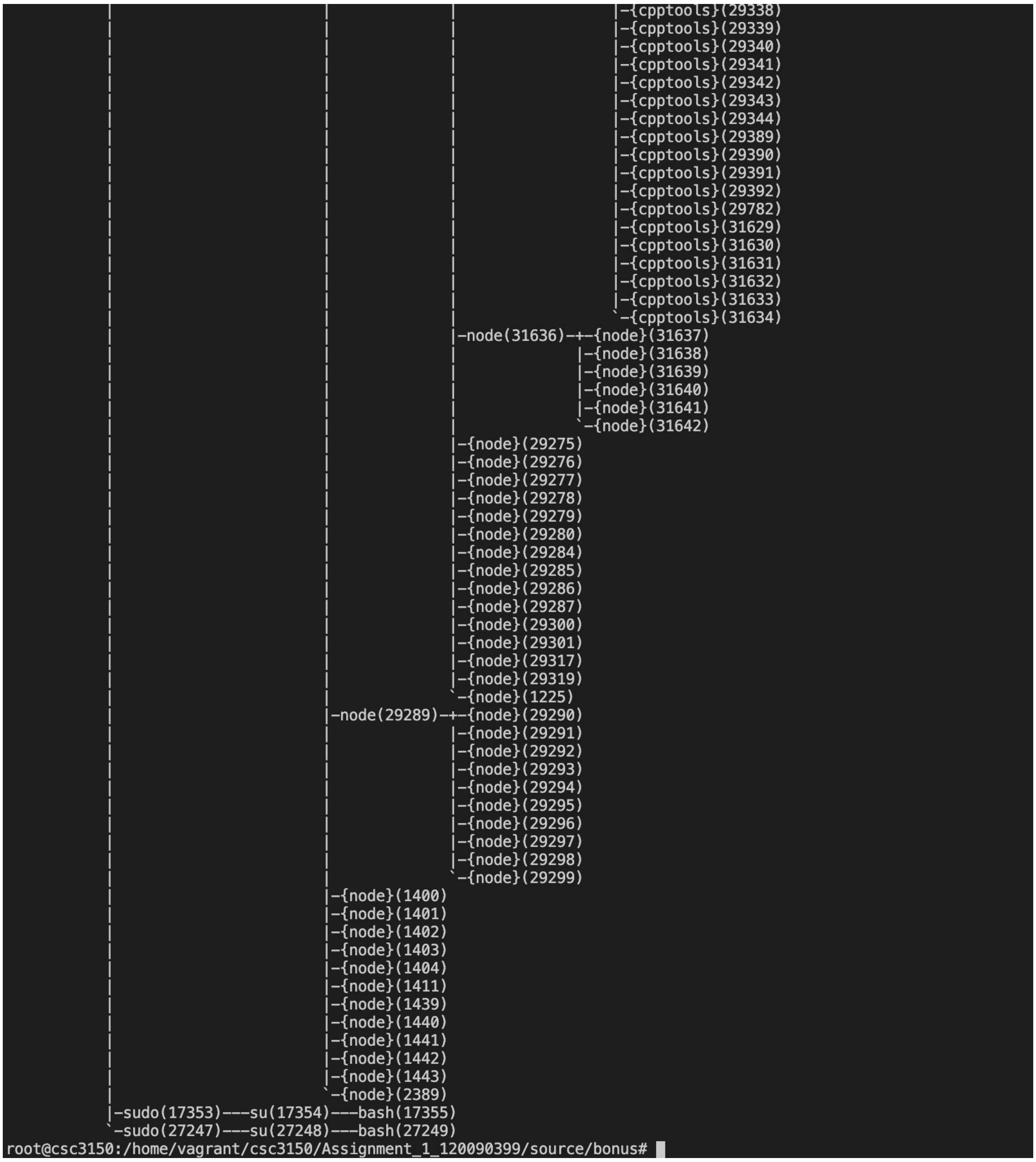


```
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          |-{cpptools}
        |           |      |          `-{cpptools}
        |           |      |-node-+-{node}
        |           |      |      |-{node}
        |           |      |      |-{node}
        |           |      |      |-{node}
        |           |      |      |-{node}
        |           |      |      `-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      `-{node}
        |           |-node-+-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      |-{node}
        |           |      `-{node}
        |           |-{node}
        |           |-{node}
        |           |-{node}
        |           |-{node}
        |           |-{node}
        |           |-{node}
        |           |-{node}
        |           |-{node}
        |           |-{node}
        |           |-{node}
        |           |-{node}
        |           `-{node}
        |-sudo---su---bash
        `-sudo---su---bash
root@csc3150:/home/vagrant/csc3150/Assignment_1_120090399/source/bonus# 
```

3. ./pstree -p

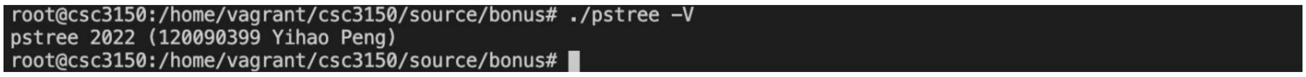


```
root@csc3150:/home/vagrant/csc3150/Assignment_1_120090399/source/bonus# ./pstree -p
systemd(1)-+-systemd-journal(435)
           |-lvmetad(445)
           |-systemd-udevd(461)
           |-dhclient(898)
           |-iscsid(1035)
           |-iscsid(1036)
           |-dbus-daemon(1048)
           |-atd(1049)
           |-rsyslogd(1050)-+-{imuxsock}(1061)
           |                |-{imklog}(1062)
           |                `-{Reg}(1063)
           |-lxcfs(1051)-+-{lxcfs}(1065)
           |             `-{lxcfs}(1066)
           |-acpid(1057)
           |-accounts-daemon(1059)-+-{gmain}(1070)
           |                       `-{gdbus}(1075)
           |-cron(1064)
           |-systemd-logind(1068)
           |-sshd(1078)---sshd(29176)---sshd(29214)---bash(29215)---bash(29219)---sleep(2367)
           |-unattended-upgr(1081)-+-{gmain}(1177)
           |-mdadm(1091)
           |-polkitd(1092)-+-{gmain}(1098)
           |               `-{gdbus}(1100)
           |-agetty(1137)
           |-login(1149)---bash(1282)
           |-irqbalance(1156)
           |-systemd(1277)---(sd-pam)(1279)
           |-sh(1385)---node(1392)-+-bash(2388)---sudo(2407)---su(2408)---bash(2409)---pstree(2597)
           |                       |-node(23623)-+-{node}(23624)
           |                       |             |-{node}(23625)
           |                       |             |-{node}(23626)
           |                       |             |-{node}(23627)
           |                       |             |-{node}(23628)
           |                       |             |-{node}(23629)
           |                       |             |-{node}(23630)
           |                       |             |-{node}(23631)
           |                       |             |-{node}(23632)
           |                       |             `-{node}(23633)
           |                       |-node(29274)-+-cpptools(29330)-+-cpptools-srv(1238)-+-{cpptools-srv}(1239)
           |                       |             |                 |                    |-{cpptools-srv}(1240)
           |                       |             |                 |                    |-{cpptools-srv}(1241)
           |                       |             |                 |                    |-{cpptools-srv}(1242)
           |                       |             |                 |                    |-{cpptools-srv}(1243)
           |                       |             |                 |                    |-{cpptools-srv}(1244)
           |                       |             |                 |                    |-{cpptools-srv}(1245)
           |                       |             |                 |                    |-{cpptools-srv}(1246)
           |                       |             |                 |                    |-{cpptools-srv}(1247)
           |                       |             |                 |                    |-{cpptools-srv}(1248)
           |                       |             |                 |                    |-{cpptools-srv}(1249)
           |                       |             |                 |                    |-{cpptools-srv}(1250)
           |                       |             |                 |                    |-{cpptools-srv}(1251)
           |                       |             |                 |                    |-{cpptools-srv}(1252)
           |                       |             |                 |                    `-{cpptools-srv}(1414)
           |                       |             |                 |-{cpptools}(29331)
           |                       |             |                 |-{cpptools}(29332)
           |                       |             |                 |-{cpptools}(29333)
           |                       |             |                 |-{cpptools}(29334)
           |                       |             |                 |-{cpptools}(29335)
           |                       |             |                 |-{cpptools}(29336)
           |                       |             |                 |-{cpptools}(29337)
           |                       |             |                 |-{cpptools}(29338)
           |                       |             |                 |-{cpptools}(29339)
           |                       |             |                 |-{cpptools}(29340)
```



```
           |                        |             |                 |-{cpptools}(29338)
           |                        |             |                 |-{cpptools}(29339)
           |                        |             |                 |-{cpptools}(29340)
           |                        |             |                 |-{cpptools}(29341)
           |                        |             |                 |-{cpptools}(29342)
           |                        |             |                 |-{cpptools}(29343)
           |                        |             |                 |-{cpptools}(29344)
           |                        |             |                 |-{cpptools}(29389)
           |                        |             |                 |-{cpptools}(29390)
           |                        |             |                 |-{cpptools}(29391)
           |                        |             |                 |-{cpptools}(29392)
           |                        |             |                 |-{cpptools}(29782)
           |                        |             |                 |-{cpptools}(31629)
           |                        |             |                 |-{cpptools}(31630)
           |                        |             |                 |-{cpptools}(31631)
           |                        |             |                 |-{cpptools}(31632)
           |                        |             |                 |-{cpptools}(31633)
           |                        |             |                 `-{cpptools}(31634)
           |                        |             |-node(31636)-+-{node}(31637)
           |                        |             |             |-{node}(31638)
           |                        |             |             |-{node}(31639)
           |                        |             |             |-{node}(31640)
           |                        |             |             |-{node}(31641)
           |                        |             |             `-{node}(31642)
           |                        |             |-{node}(29275)
           |                        |             |-{node}(29276)
           |                        |             |-{node}(29277)
           |                        |             |-{node}(29278)
           |                        |             |-{node}(29279)
           |                        |             |-{node}(29280)
           |                        |             |-{node}(29284)
           |                        |             |-{node}(29285)
           |                        |             |-{node}(29286)
           |                        |             |-{node}(29287)
           |                        |             |-{node}(29300)
           |                        |             |-{node}(29301)
           |                        |             |-{node}(29317)
           |                        |             |-{node}(29319)
           |                        |             `-{node}(1225)
           |                        |-node(29289)-+-{node}(29290)
           |                        |             |-{node}(29291)
           |                        |             |-{node}(29292)
           |                        |             |-{node}(29293)
           |                        |             |-{node}(29294)
           |                        |             |-{node}(29295)
           |                        |             |-{node}(29296)
           |                        |             |-{node}(29297)
           |                        |             |-{node}(29298)
           |                        |             `-{node}(29299)
           |                        |-{node}(1400)
           |                        |-{node}(1401)
           |                        |-{node}(1402)
           |                        |-{node}(1403)
           |                        |-{node}(1404)
           |                        |-{node}(1411)
           |                        |-{node}(1439)
           |                        |-{node}(1440)
           |                        |-{node}(1441)
           |                        |-{node}(1442)
           |                        |-{node}(1443)
           |                        `-{node}(2389)
           |-sudo(17353)---su(17354)---bash(17355)
           `-sudo(27247)---su(27248)---bash(27249)
root@csc3150:/home/vagrant/csc3150/Assignment_1_120090399/source/bonus# 
```

4. ./pstree -V



```
root@csc3150:/home/vagrant/csc3150/source/bonus# ./pstree -V
pstree 2022 (120090399 Yihao Peng)
root@csc3150:/home/vagrant/csc3150/source/bonus# 
```

4. Conclusion

From program 1, I learn how to create child process in the user mode and to execute another program in the child process. Moreover, I learn to make the parent process wait for the signal sent from children and output the information. From program 2, I learn how to build a simple kernel module, to insert and remove modules and call functions of the kernel in the built kernel module by modifying the kernel files and recompiling the kernel. Furthermore, I learn how to create a thread in the kernel to fork a new process to execute program and also make parent process wait and output the signal from child process. In addition, from bonus program, I learn how to get the information like name, pid, ppid, threads of all process in the linux and print them in the format of pstree. I learn how to use linked list and some structures to store information and show the relationship between two process. Also, the bonus program improve my capacities to search documents and make use of documents. In a word, this project help me better understand how process is created in user and kernel mode. I have deeper knowledge of operating system.