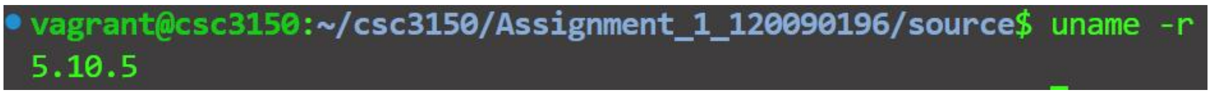
Report

Name: Zhenyu Pan

Student ID: 120090196

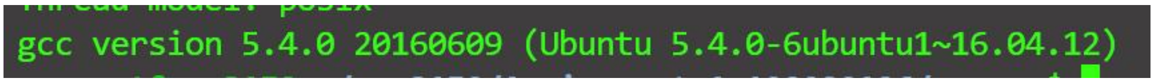
Environment

Linux Kernel Version: 5.10.5



```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source$ uname -r
5.10.5
```

GCC Version: 5.4.0



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

Steps to set up development environment

1. cd ~ // enter home directory
2. wget https://mirror.tuna.tsinghua.edu.cn/kenel/v5.x/linux-5.10.5.tar.xz
3. sudo tar -xvf linux-5.10.5.tar.xz   // decompress files
4. cd linux-5.10.5 // enter the source code directory
5. sudo root   // enter the root directory
6. sudo apt-get install libncurses5-dev libssl-dev bc .... // install tools
7. make mrproper
8. make clean
9. make menuconfig -- save config -- exit // generate default configuration
10. make j8
11. kernel/fork.c EXPORT_SYMBOL(kernel_clone);

kernel/exit.c EXPORT_SYMBOL(do_exit);

kernel/exit.c EXPORT_SYMBOL(do_wait);

fs/exec.c EXPORT_SYMBOL(do_execve);

fs/namei.c EXPORT_SYMBOL(kernel_getname)

remove keyword static;

(all above using vim to complete)

12. make -j8 // recompile
13. make modules_install // install kernel module
14. make install // install kernel (actually just copy the header file(.h) to the some directories of PATH in the system)
15. reboot

一、Task 1

a) Design of the program

In this program, we run a process under user mode. The main process will fork a new child process using function ***fork()***. Then, the child process will execute the given test file and receive the signal by function ***execve()***; the parent process will wait until the child process is finished using ***waitpid()***. After that, the parent process will identify the status in function ***waitpid()*** and output the corresponding information according to the value of ***status***.

b) Steps to execute the program

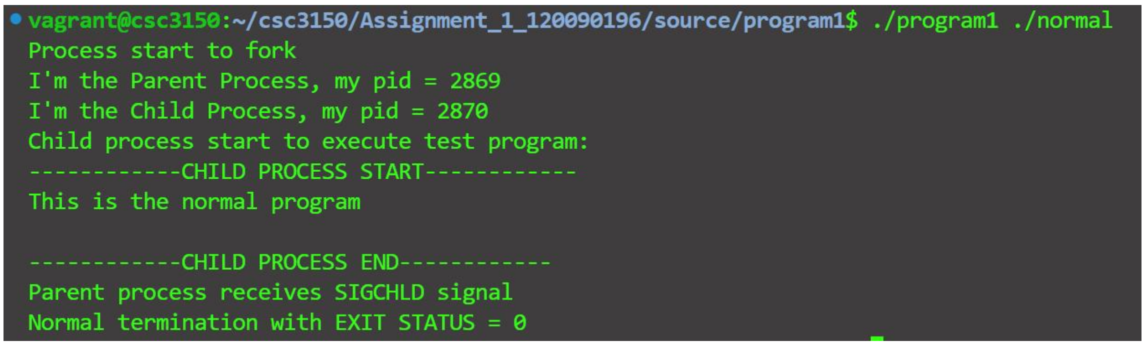
1. Go to the folder of the program1 (cd .../source/program1)

2. Type "sudo make" in the terminal

3. Type "./program1 ./TEST_FILE" in the terminal

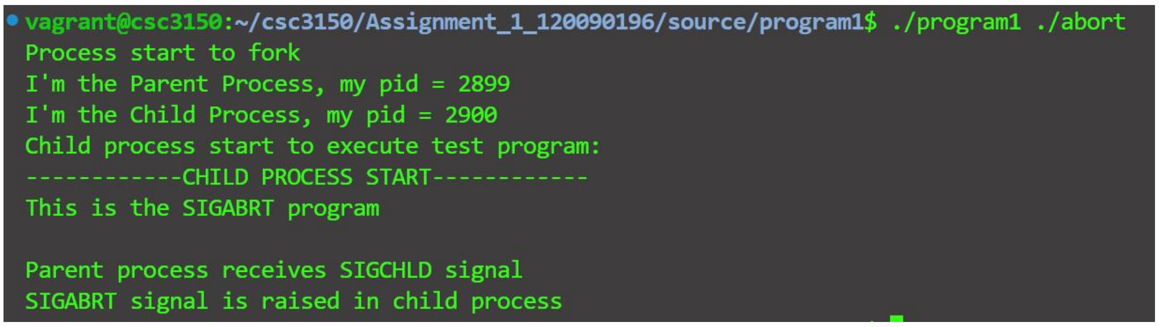
c) Screenshots

1. normal



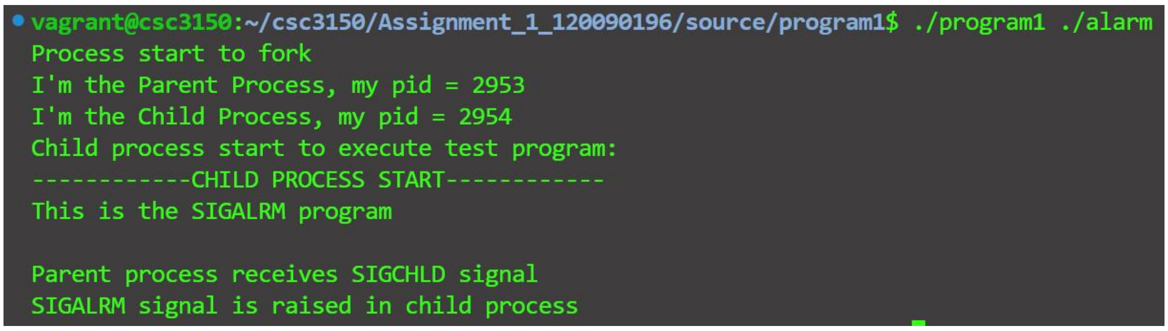
```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program1$ ./program1 ./normal
Process start to fork
I'm the Parent Process, my pid = 2869
I'm the Child Process, my pid = 2870
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the normal program

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

2. abort



```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program1$ ./program1 ./abort
Process start to fork
I'm the Parent Process, my pid = 2899
I'm the Child Process, my pid = 2900
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGABRT program

Parent process receives SIGCHLD signal
SIGABRT signal is raised in child process
```

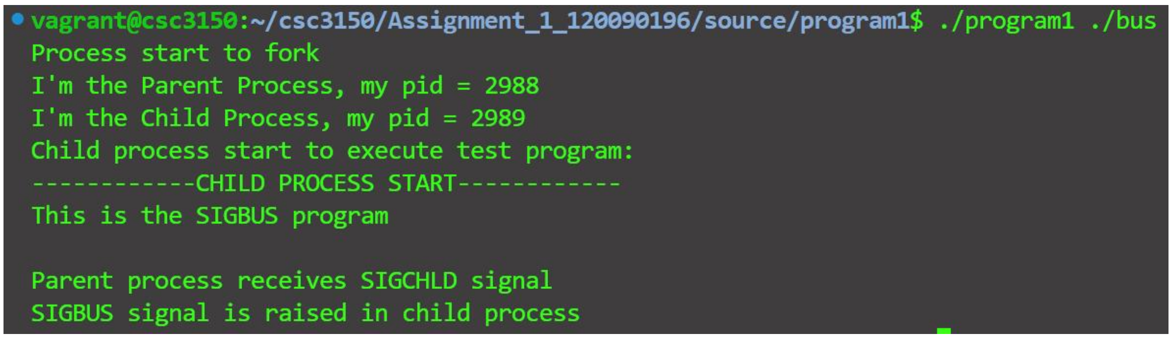
3. alarm



```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program1$ ./program1 ./alarm
Process start to fork
I'm the Parent Process, my pid = 2953
I'm the Child Process, my pid = 2954
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGALRM program

Parent process receives SIGCHLD signal
SIGALRM signal is raised in child process
```

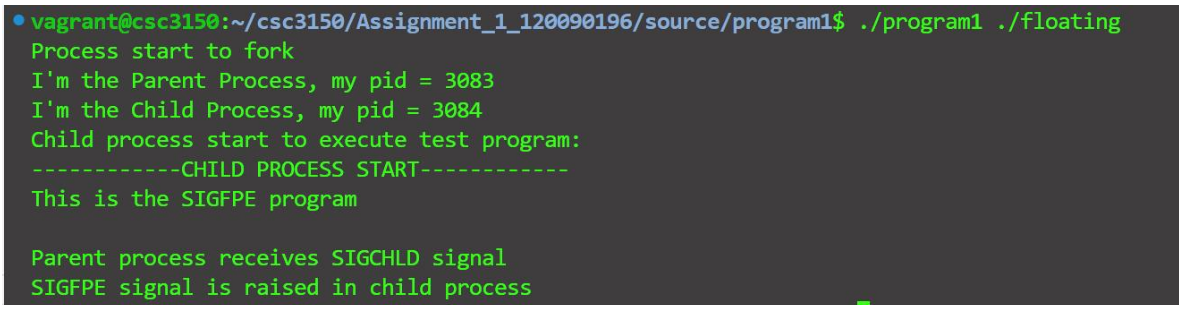
Please see next page

4. bus



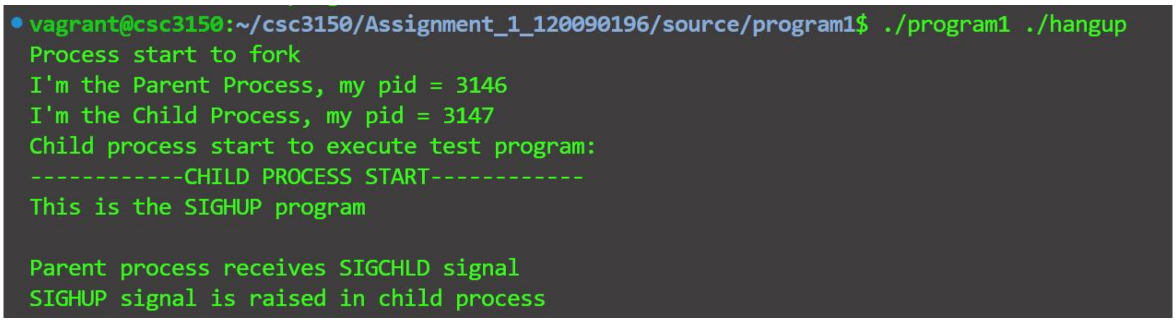
```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program1$ ./program1 ./bus
Process start to fork
I'm the Parent Process, my pid = 2988
I'm the Child Process, my pid = 2989
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGBUS program

Parent process receives SIGCHLD signal
SIGBUS signal is raised in child process
```

5. floating



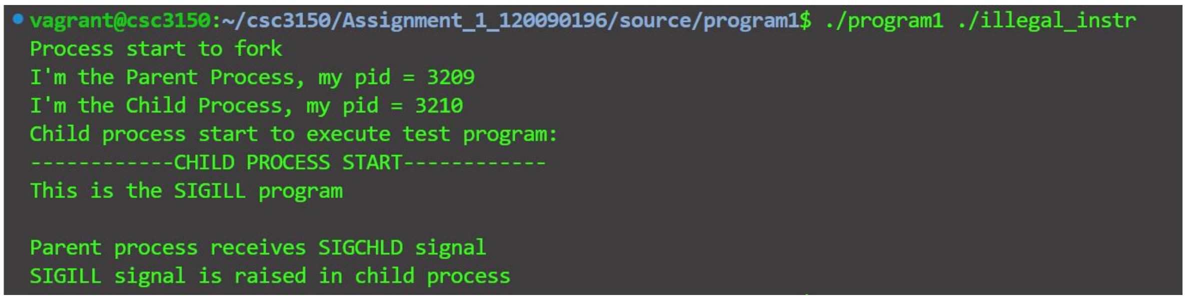
```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program1$ ./program1 ./floating
Process start to fork
I'm the Parent Process, my pid = 3083
I'm the Child Process, my pid = 3084
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGFPE program

Parent process receives SIGCHLD signal
SIGFPE signal is raised in child process
```

6. hangup



```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program1$ ./program1 ./hangup
Process start to fork
I'm the Parent Process, my pid = 3146
I'm the Child Process, my pid = 3147
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGHUP program

Parent process receives SIGCHLD signal
SIGHUP signal is raised in child process
```

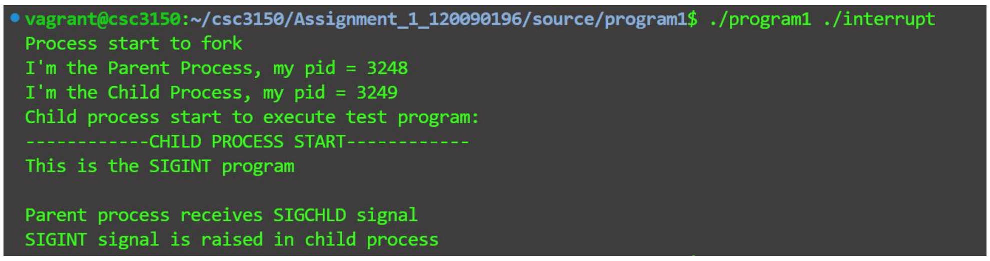
7. illegal_instr



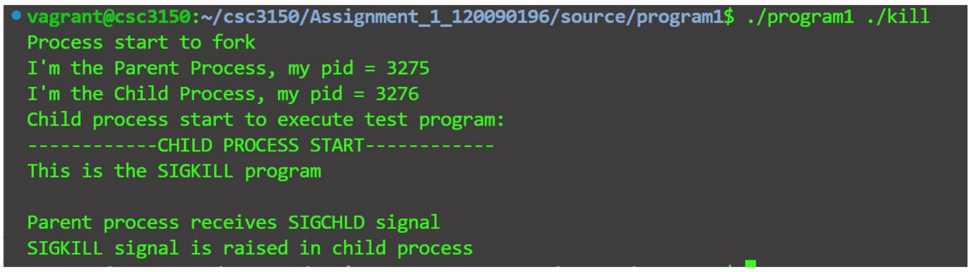
```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program1$ ./program1 ./illegal_instr
Process start to fork
I'm the Parent Process, my pid = 3209
I'm the Child Process, my pid = 3210
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGILL program

Parent process receives SIGCHLD signal
SIGILL signal is raised in child process
```

Please see next page

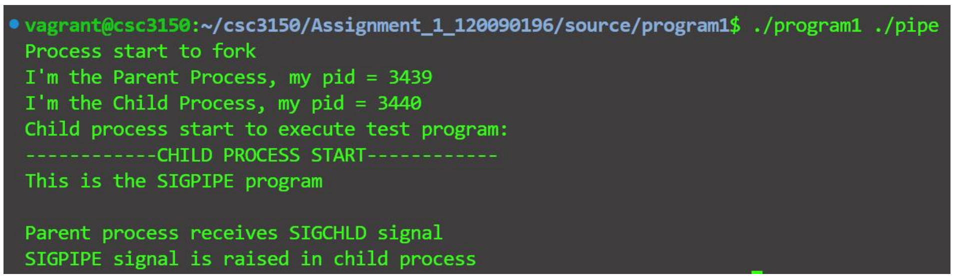
8. interrupt


```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program1$ ./program1 ./interrupt
Process start to fork
I'm the Parent Process, my pid = 3248
I'm the Child Process, my pid = 3249
Child process start to execute test program:
------------CHILD PROCESS START-----------
This is the SIGINT program

Parent process receives SIGCHLD signal
SIGINT signal is raised in child process
```

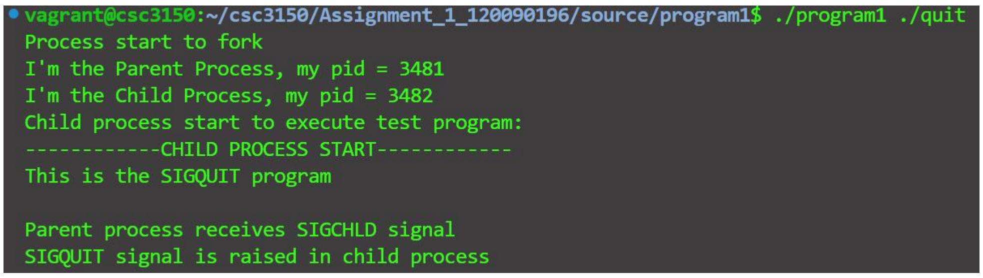
9. kill


```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program1$ ./program1 ./kill
Process start to fork
I'm the Parent Process, my pid = 3275
I'm the Child Process, my pid = 3276
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGKILL program

Parent process receives SIGCHLD signal
SIGKILL signal is raised in child process
```

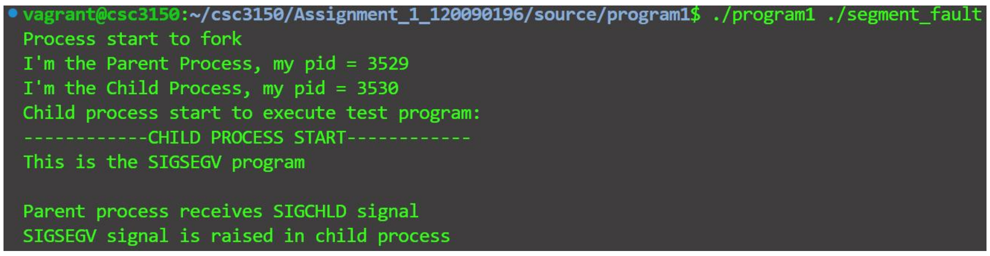
10. pipe


```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program1$ ./program1 ./pipe
Process start to fork
I'm the Parent Process, my pid = 3439
I'm the Child Process, my pid = 3440
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGPIPE program

Parent process receives SIGCHLD signal
SIGPIPE signal is raised in child process
```

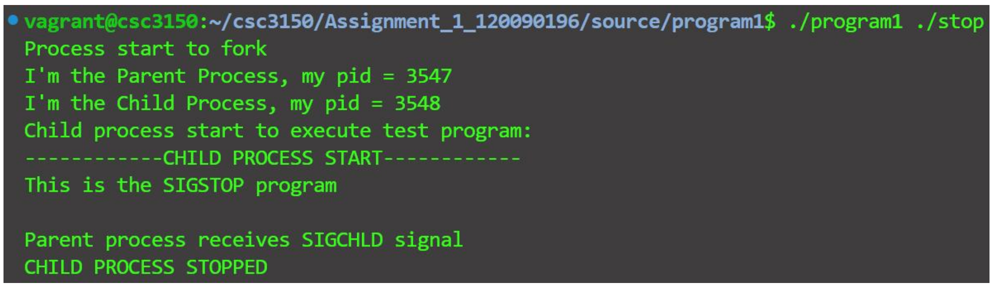
11. quit


```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program1$ ./program1 ./quit
Process start to fork
I'm the Parent Process, my pid = 3481
I'm the Child Process, my pid = 3482
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

Parent process receives SIGCHLD signal
SIGQUIT signal is raised in child process
```

12. segment_fault

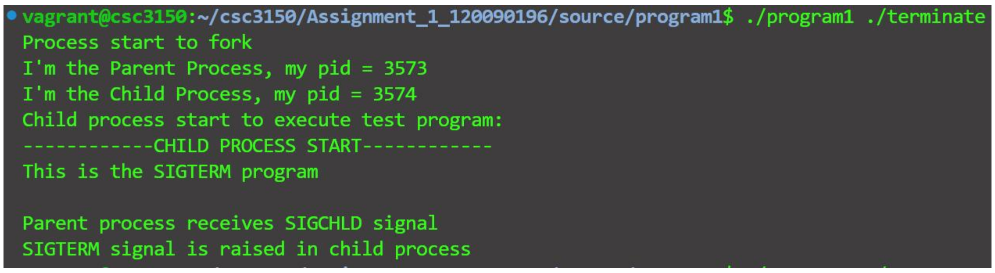

```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program1$ ./program1 ./segment_fault
Process start to fork
I'm the Parent Process, my pid = 3529
I'm the Child Process, my pid = 3530
Child process start to execute test program:
------------CHILD PROCESS START-----------
This is the SIGSEGV program

Parent process receives SIGCHLD signal
SIGSEGV signal is raised in child process
```

13. stop



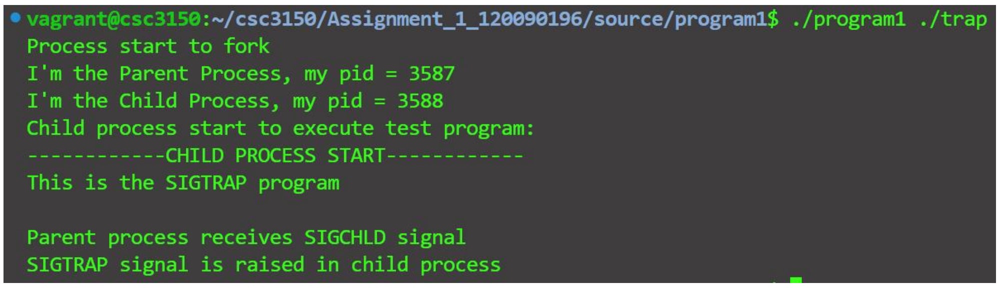
```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program1$ ./program1 ./stop
Process start to fork
I'm the Parent Process, my pid = 3547
I'm the Child Process, my pid = 3548
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

Parent process receives SIGCHLD signal
CHILD PROCESS STOPPED
```

14. terminate



```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program1$ ./program1 ./terminate
Process start to fork
I'm the Parent Process, my pid = 3573
I'm the Child Process, my pid = 3574
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTERM program

Parent process receives SIGCHLD signal
SIGTERM signal is raised in child process
```

15. trap



```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program1$ ./program1 ./trap
Process start to fork
I'm the Parent Process, my pid = 3587
I'm the Child Process, my pid = 3588
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTRAP program

Parent process receives SIGCHLD signal
SIGTRAP signal is raised in child process
```

二、Task 2

a) Design of the program

This program will create a kernel thread using function ***kthread_create()***, which is like a kernel version of program_1. In the thread, the program will fork a new child process to execute another program by function ***my_fork()***. After that, the parent process will wait for the child process's terminated signal and print out the corresponding information about it by using the function ***my_wait()***.

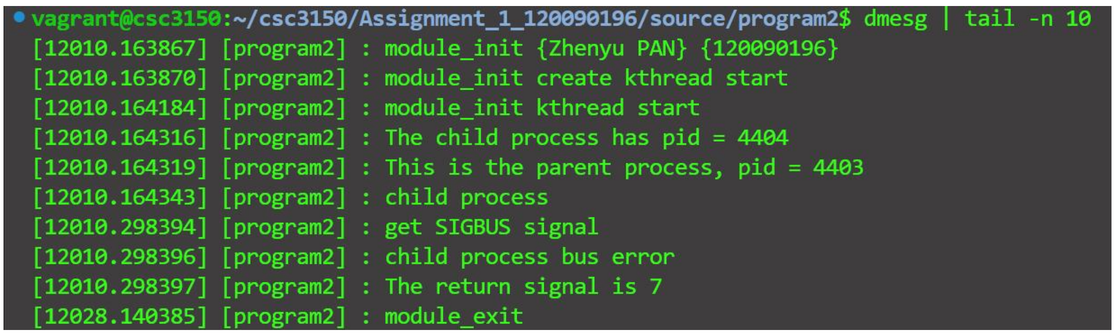
Note: Since we need to use functions *kernel_clone()*, *do_execve()*, *kernel_getname()* and *do_wait(),* we have to find their position in kernel and export them using ***EXPORT_SYMBOL.*** Additionally, we have to remove the keyword *static* before the function. After that, we can use extern to make them available to invoke.

b) Steps to execute the program

1. Make sure you modify the kernel(EXORT_SYMBOL), remove the keyword *static*(using vim) , rebuild the module and *extern* those functions.
2. Go to the program folder(cd .../source/program2).
3. Type "gcc test.c -o test" in the terminal to compile the test file.
4. Type "sudo make" in the terminal.
5. Type "sudo insmod program2.ko" in the terminal.
6. Type "sudo rmmod program2.ko" in the terminal.
7. Type "dmesg | tail -n 10" in terminal to display the message.

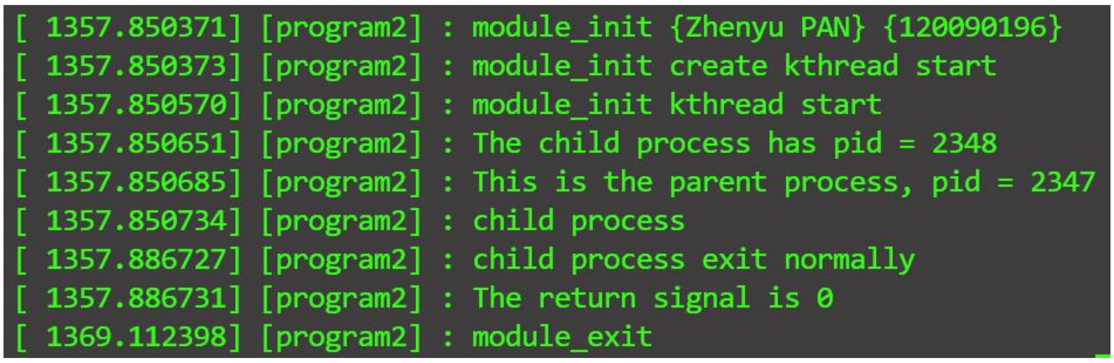
c) Screenshots

1. test.c (SIGBUS)



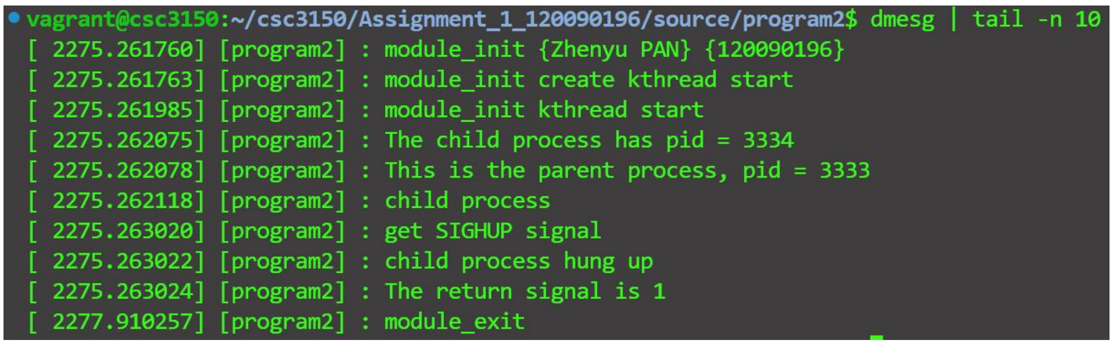
```
• vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program2$ dmesg | tail -n 10
  [12010.163867] [program2] : module_init {Zhenyu PAN} {120090196}
  [12010.163870] [program2] : module_init create kthread start
  [12010.164184] [program2] : module_init kthread start
  [12010.164316] [program2] : The child process has pid = 4404
  [12010.164319] [program2] : This is the parent process, pid = 4403
  [12010.164343] [program2] : child process
  [12010.298394] [program2] : get SIGBUS signal
  [12010.298396] [program2] : child process bus error
  [12010.298397] [program2] : The return signal is 7
  [12028.140385] [program2] : module_exit
```

2. normal



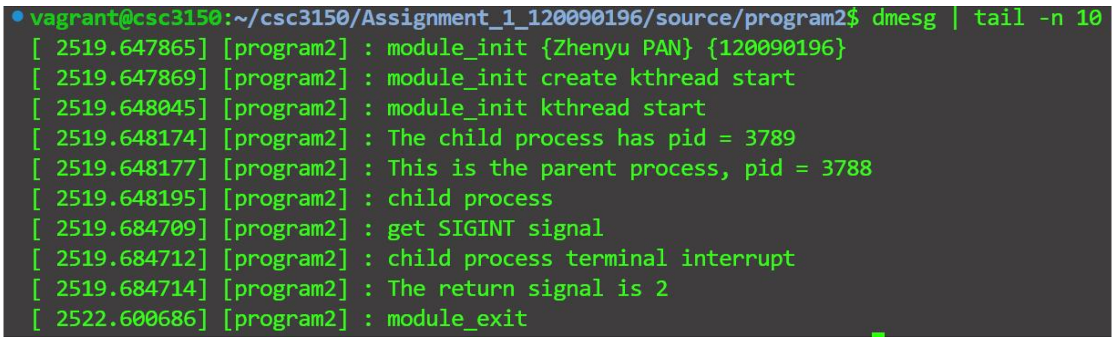
```
[ 1357.850371] [program2] : module_init {Zhenyu PAN} {120090196}
[ 1357.850373] [program2] : module_init create kthread start
[ 1357.850570] [program2] : module_init kthread start
[ 1357.850651] [program2] : The child process has pid = 2348
[ 1357.850685] [program2] : This is the parent process, pid = 2347
[ 1357.850734] [program2] : child process
[ 1357.886727] [program2] : child process exit normally
[ 1357.886731] [program2] : The return signal is 0
[ 1369.112398] [program2] : module_exit
```

3. hangup



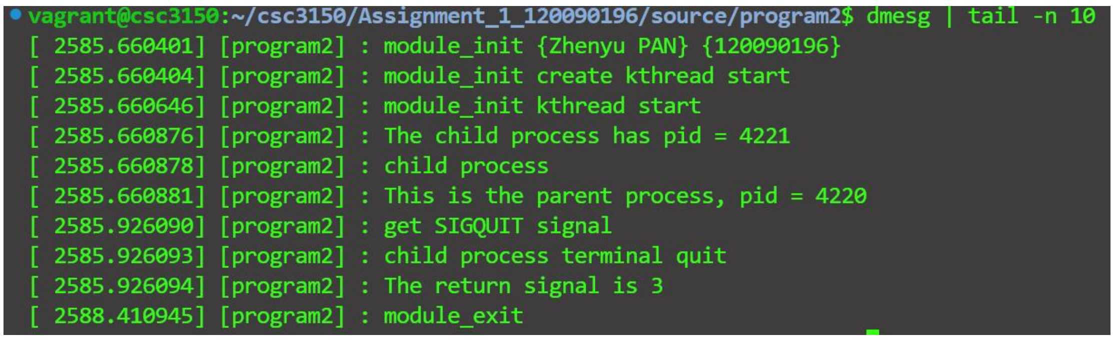
```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program2$ dmesg | tail -n 10
 [ 2275.261760] [program2] : module_init {Zhenyu PAN} {120090196}
 [ 2275.261763] [program2] : module_init create kthread start
 [ 2275.261985] [program2] : module_init kthread start
 [ 2275.262075] [program2] : The child process has pid = 3334
 [ 2275.262078] [program2] : This is the parent process, pid = 3333
 [ 2275.262118] [program2] : child process
 [ 2275.263020] [program2] : get SIGHUP signal
 [ 2275.263022] [program2] : child process hung up
 [ 2275.263024] [program2] : The return signal is 1
 [ 2277.910257] [program2] : module_exit
```

4. interrupt



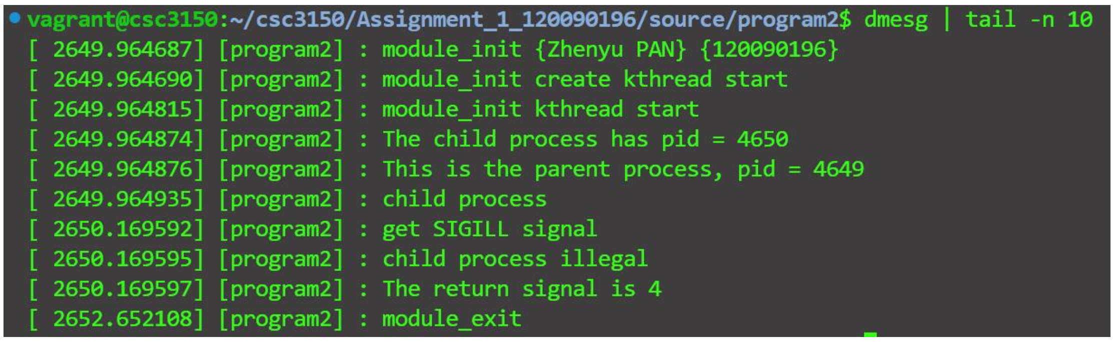
```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program2$ dmesg | tail -n 10
 [ 2519.647865] [program2] : module_init {Zhenyu PAN} {120090196}
 [ 2519.647869] [program2] : module_init create kthread start
 [ 2519.648045] [program2] : module_init kthread start
 [ 2519.648174] [program2] : The child process has pid = 3789
 [ 2519.648177] [program2] : This is the parent process, pid = 3788
 [ 2519.648195] [program2] : child process
 [ 2519.684709] [program2] : get SIGINT signal
 [ 2519.684712] [program2] : child process terminal interrupt
 [ 2519.684714] [program2] : The return signal is 2
 [ 2522.600686] [program2] : module_exit
```

5. quit



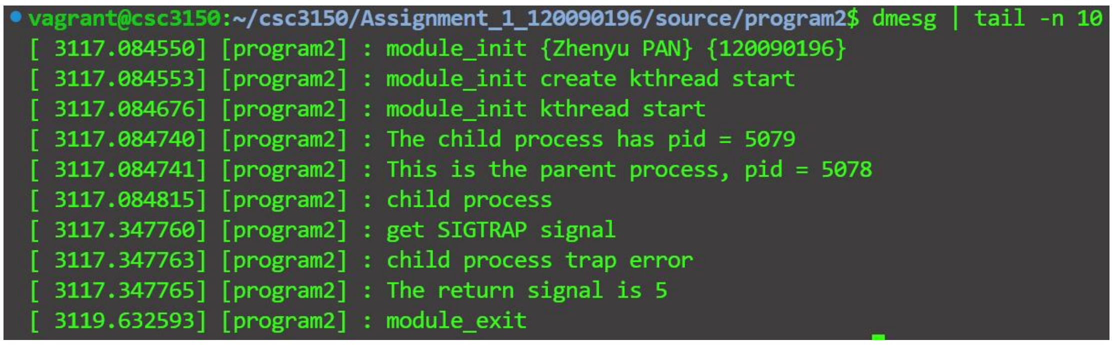
```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program2$ dmesg | tail -n 10
 [ 2585.660401] [program2] : module_init {Zhenyu PAN} {120090196}
 [ 2585.660404] [program2] : module_init create kthread start
 [ 2585.660646] [program2] : module_init kthread start
 [ 2585.660876] [program2] : The child process has pid = 4221
 [ 2585.660878] [program2] : child process
 [ 2585.660881] [program2] : This is the parent process, pid = 4220
 [ 2585.926090] [program2] : get SIGQUIT signal
 [ 2585.926093] [program2] : child process terminal quit
 [ 2585.926094] [program2] : The return signal is 3
 [ 2588.410945] [program2] : module_exit
```

6. illegal_instr



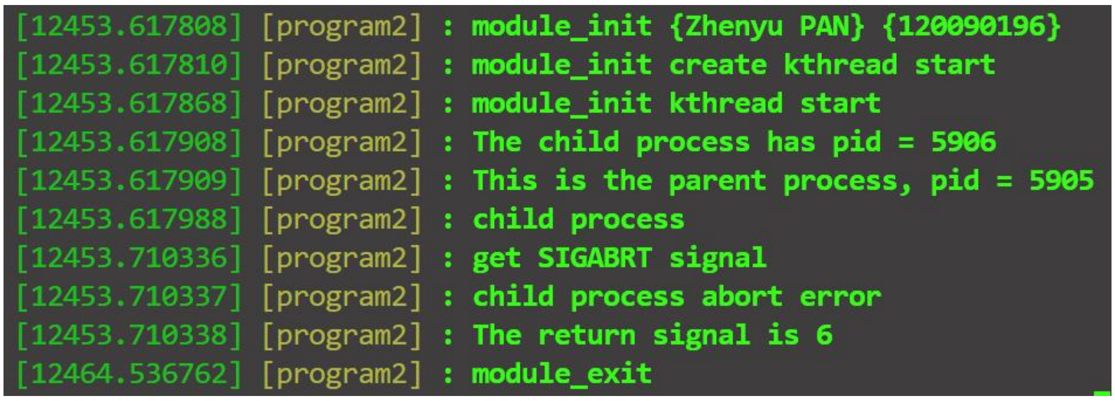
```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program2$ dmesg | tail -n 10
[ 2649.964687] [program2] : module_init {Zhenyu PAN} {120090196}
[ 2649.964690] [program2] : module_init create kthread start
[ 2649.964815] [program2] : module_init kthread start
[ 2649.964874] [program2] : The child process has pid = 4650
[ 2649.964876] [program2] : This is the parent process, pid = 4649
[ 2649.964935] [program2] : child process
[ 2650.169592] [program2] : get SIGILL signal
[ 2650.169595] [program2] : child process illegal
[ 2650.169597] [program2] : The return signal is 4
[ 2652.652108] [program2] : module_exit
```

7. trap



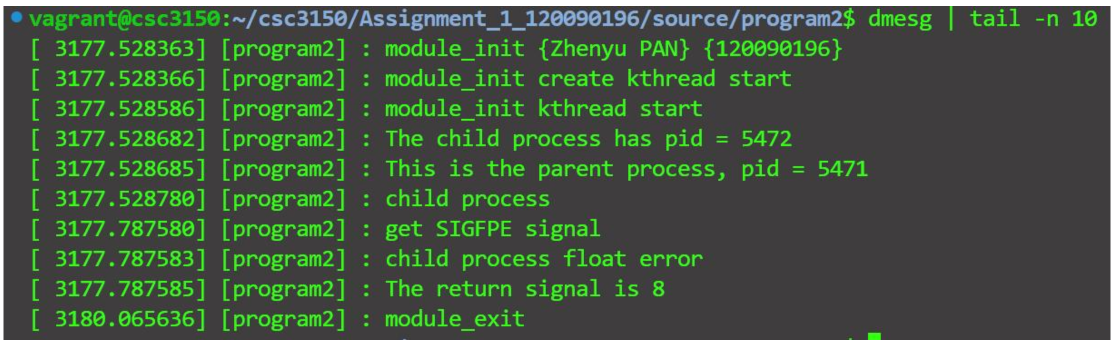
```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program2$ dmesg | tail -n 10
[ 3117.084550] [program2] : module_init {Zhenyu PAN} {120090196}
[ 3117.084553] [program2] : module_init create kthread start
[ 3117.084676] [program2] : module_init kthread start
[ 3117.084740] [program2] : The child process has pid = 5079
[ 3117.084741] [program2] : This is the parent process, pid = 5078
[ 3117.084815] [program2] : child process
[ 3117.347760] [program2] : get SIGTRAP signal
[ 3117.347763] [program2] : child process trap error
[ 3117.347765] [program2] : The return signal is 5
[ 3119.632593] [program2] : module_exit
```

8. abort



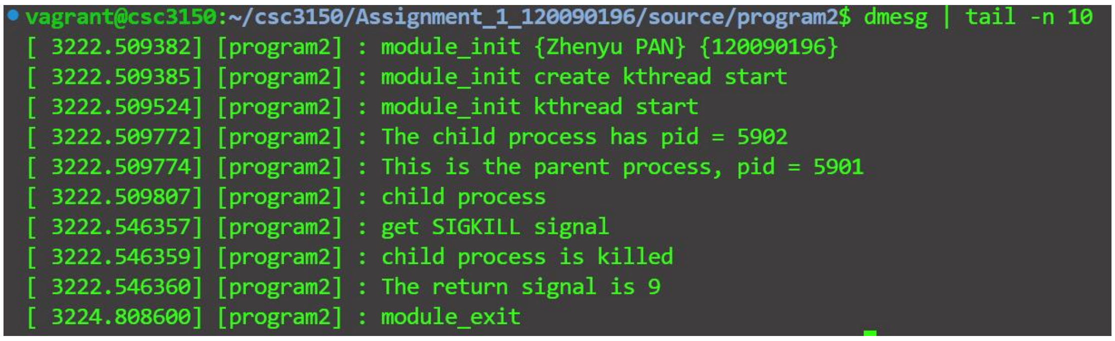
```
[12453.617808] [program2] : module_init {Zhenyu PAN} {120090196}
[12453.617810] [program2] : module_init create kthread start
[12453.617868] [program2] : module_init kthread start
[12453.617908] [program2] : The child process has pid = 5906
[12453.617909] [program2] : This is the parent process, pid = 5905
[12453.617988] [program2] : child process
[12453.710336] [program2] : get SIGABRT signal
[12453.710337] [program2] : child process abort error
[12453.710338] [program2] : The return signal is 6
[12464.536762] [program2] : module_exit
```

9. floating



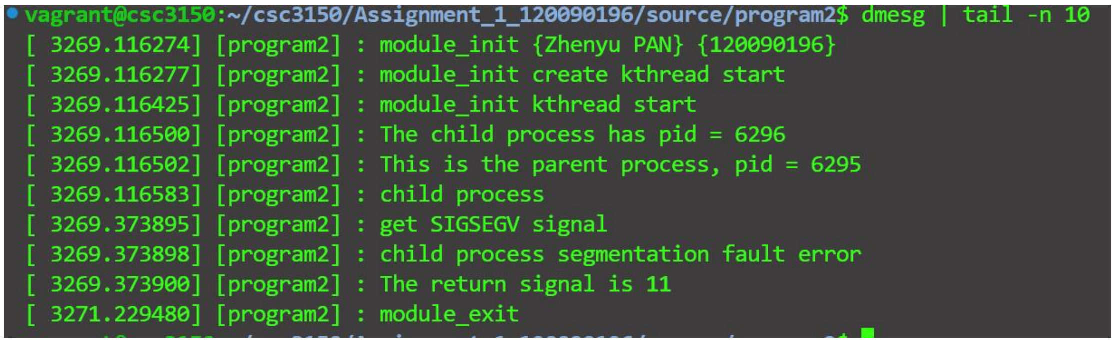
```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program2$ dmesg | tail -n 10
[ 3177.528363] [program2] : module_init {Zhenyu PAN} {120090196}
[ 3177.528366] [program2] : module_init create kthread start
[ 3177.528586] [program2] : module_init kthread start
[ 3177.528682] [program2] : The child process has pid = 5472
[ 3177.528685] [program2] : This is the parent process, pid = 5471
[ 3177.528780] [program2] : child process
[ 3177.787580] [program2] : get SIGFPE signal
[ 3177.787583] [program2] : child process float error
[ 3177.787585] [program2] : The return signal is 8
[ 3180.065636] [program2] : module_exit
```

10. kill



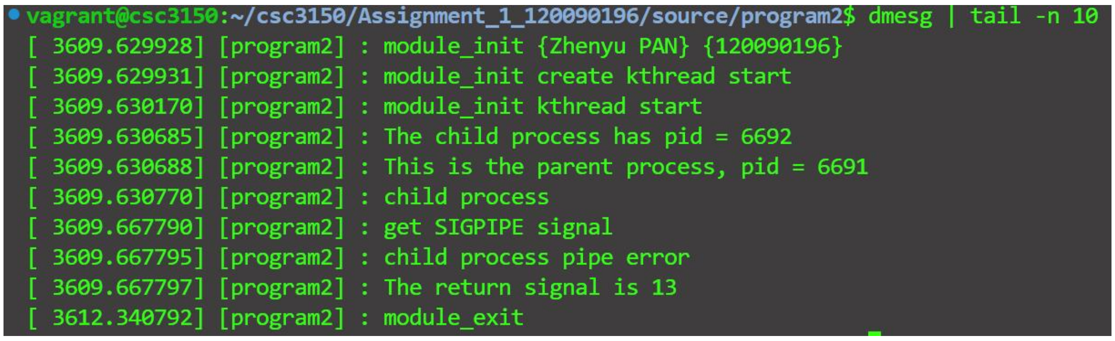
```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program2$ dmesg | tail -n 10
[ 3222.509382] [program2] : module_init {Zhenyu PAN} {120090196}
[ 3222.509385] [program2] : module_init create kthread start
[ 3222.509524] [program2] : module_init kthread start
[ 3222.509772] [program2] : The child process has pid = 5902
[ 3222.509774] [program2] : This is the parent process, pid = 5901
[ 3222.509807] [program2] : child process
[ 3222.546357] [program2] : get SIGKILL signal
[ 3222.546359] [program2] : child process is killed
[ 3222.546360] [program2] : The return signal is 9
[ 3224.808600] [program2] : module_exit
```

11. segemnt_fault



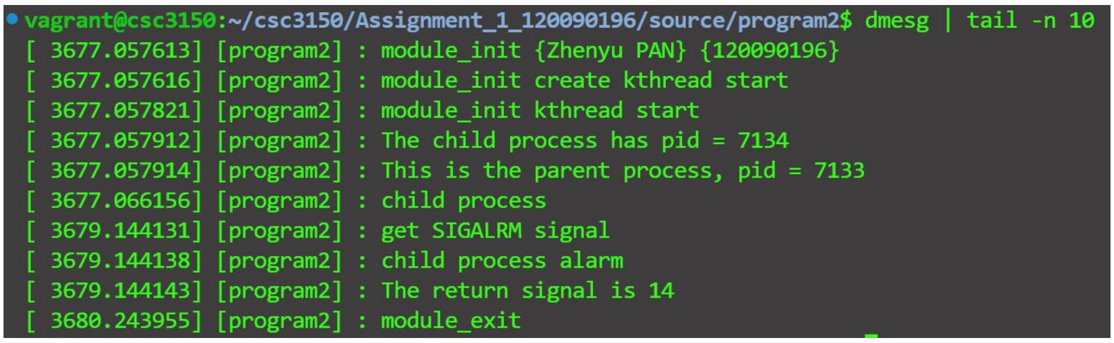
```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program2$ dmesg | tail -n 10
[ 3269.116274] [program2] : module_init {Zhenyu PAN} {120090196}
[ 3269.116277] [program2] : module_init create kthread start
[ 3269.116425] [program2] : module_init kthread start
[ 3269.116500] [program2] : The child process has pid = 6296
[ 3269.116502] [program2] : This is the parent process, pid = 6295
[ 3269.116583] [program2] : child process
[ 3269.373895] [program2] : get SIGSEGV signal
[ 3269.373898] [program2] : child process segmentation fault error
[ 3269.373900] [program2] : The return signal is 11
[ 3271.229480] [program2] : module_exit
```

12. pipe



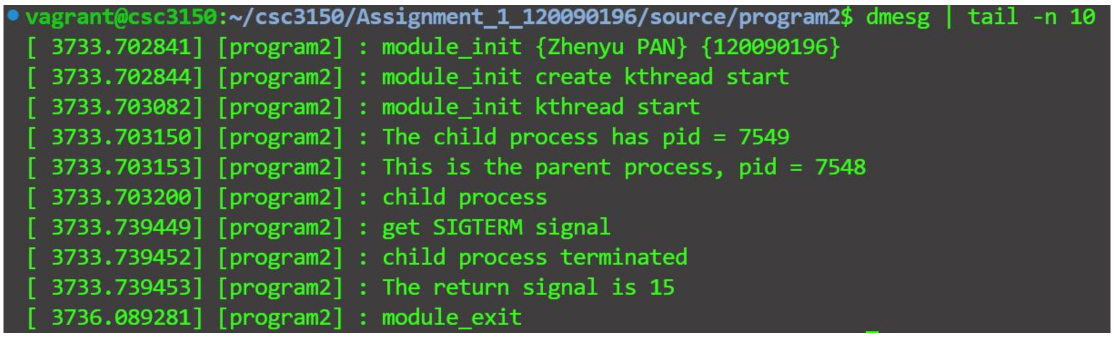
```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program2$ dmesg | tail -n 10
[ 3609.629928] [program2] : module_init {Zhenyu PAN} {120090196}
[ 3609.629931] [program2] : module_init create kthread start
[ 3609.630170] [program2] : module_init kthread start
[ 3609.630685] [program2] : The child process has pid = 6692
[ 3609.630688] [program2] : This is the parent process, pid = 6691
[ 3609.630770] [program2] : child process
[ 3609.667790] [program2] : get SIGPIPE signal
[ 3609.667795] [program2] : child process pipe error
[ 3609.667797] [program2] : The return signal is 13
[ 3612.340792] [program2] : module_exit
```

13. alarm



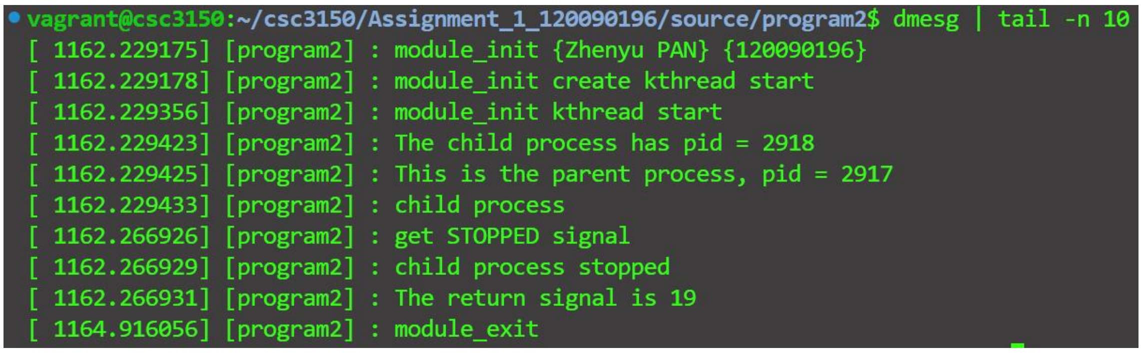
```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program2$ dmesg | tail -n 10
[ 3677.057613] [program2] : module_init {Zhenyu PAN} {120090196}
[ 3677.057616] [program2] : module_init create kthread start
[ 3677.057821] [program2] : module_init kthread start
[ 3677.057912] [program2] : The child process has pid = 7134
[ 3677.057914] [program2] : This is the parent process, pid = 7133
[ 3677.066156] [program2] : child process
[ 3679.144131] [program2] : get SIGALRM signal
[ 3679.144138] [program2] : child process alarm
[ 3679.144143] [program2] : The return signal is 14
[ 3680.243955] [program2] : module_exit
```

14. terminate



```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program2$ dmesg | tail -n 10
[ 3733.702841] [program2] : module_init {Zhenyu PAN} {120090196}
[ 3733.702844] [program2] : module_init create kthread start
[ 3733.703082] [program2] : module_init kthread start
[ 3733.703150] [program2] : The child process has pid = 7549
[ 3733.703153] [program2] : This is the parent process, pid = 7548
[ 3733.703200] [program2] : child process
[ 3733.739449] [program2] : get SIGTERM signal
[ 3733.739452] [program2] : child process terminated
[ 3733.739453] [program2] : The return signal is 15
[ 3736.089281] [program2] : module_exit
```

15. stop



```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/program2$ dmesg | tail -n 10
[ 1162.229175] [program2] : module_init {Zhenyu PAN} {120090196}
[ 1162.229178] [program2] : module_init create kthread start
[ 1162.229356] [program2] : module_init kthread start
[ 1162.229423] [program2] : The child process has pid = 2918
[ 1162.229425] [program2] : This is the parent process, pid = 2917
[ 1162.229433] [program2] : child process
[ 1162.266926] [program2] : get STOPPED signal
[ 1162.266929] [program2] : child process stopped
[ 1162.266931] [program2] : The return signal is 19
[ 1164.916056] [program2] : module_exit
```

三、Bonus

a) Design of program

This program simulate the Linux command ***pstree***, which could print out the process tree of Linux. First, since I set the default format is UTF-8, I use the function ***setlocale()*** to change the entire locale from C to C.UTF-8 so that we can print out information in default format. Then, the function ***get_opt()*** will check whether the input is valid or not (in my implementation, there are five options available: -A, -p, -n, -V, -U); if not, the function ***Usage()*** will tell the user which options are valid and the function of them to guide the user to input correctly. After that, the ***print_version()*** function will print out the version and the author name if user type "-V"; or the

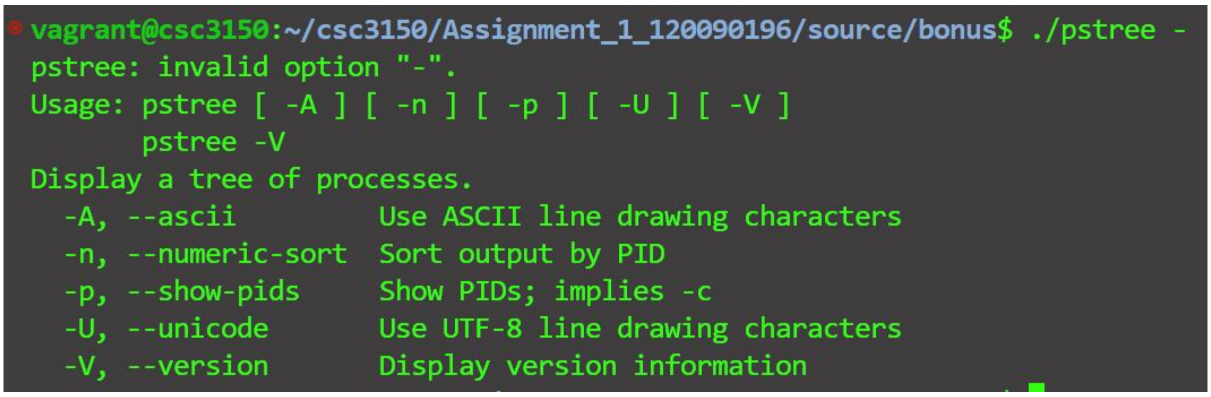
print_ps_tree() function will print out corresponding information up to what command the user type.

b) Steps to execute the program

1. Go to the folder of the bonus (cd .../source/bonus)
2. Type "sudo make" in the terminal
3. Tyep "./pstree"
4. Type "./pstree -option in {A, p, n, V, U}"

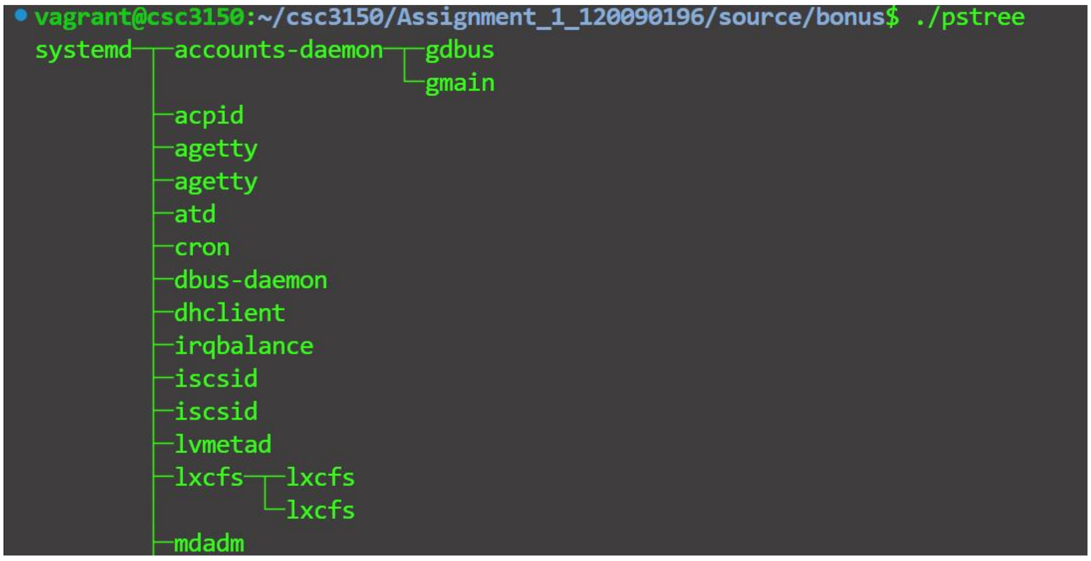
c) Screenshots

Guide Information



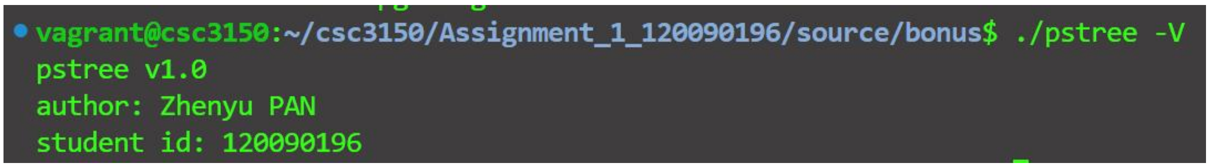
```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/bonus$ ./pstree -
pstree: invalid option "-".
Usage: pstree [ -A ] [ -n ] [ -p ] [ -U ] [ -V ]
       pstree -V
Display a tree of processes.
  -A, --ascii          Use ASCII line drawing characters
  -n, --numeric-sort   Sort output by PID
  -p, --show-pids      Show PIDs; implies -c
  -U, --unicode        Use UTF-8 line drawing characters
  -V, --version        Display version information
```

1. ./pstree(since the terminal is limited, just show part of it)



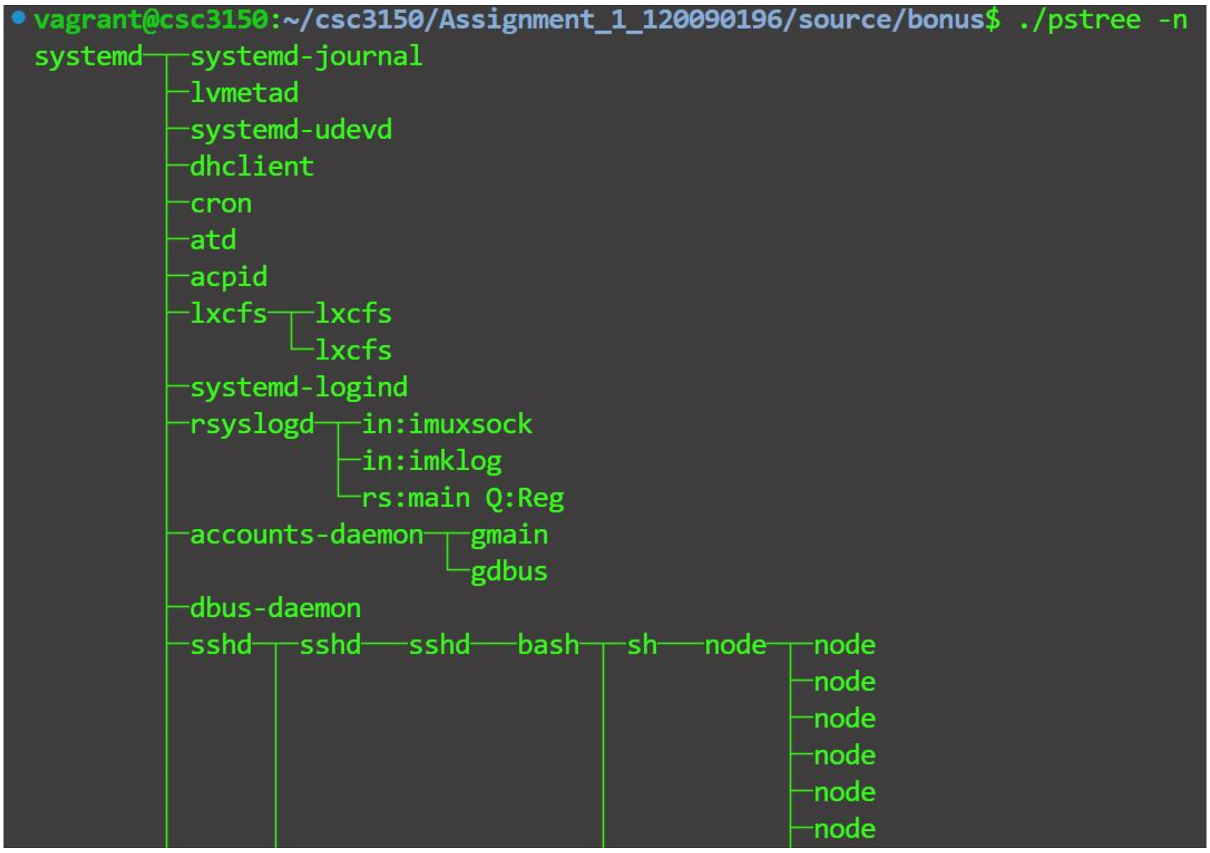
```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/bonus$ ./pstree
systemd─┬─accounts-daemon─┬─gdbus
        │                 └─gmain
        ├─acpid
        ├─agetty
        ├─agetty
        ├─atd
        ├─cron
        ├─dbus-daemon
        ├─dhclient
        ├─irqbalance
        ├─iscsid
        ├─iscsid
        ├─lvmetad
        ├─lxcfs─┬─lxcfs
        │       └─lxcfs
        ├─mdadm
```

2. ./pstree -V



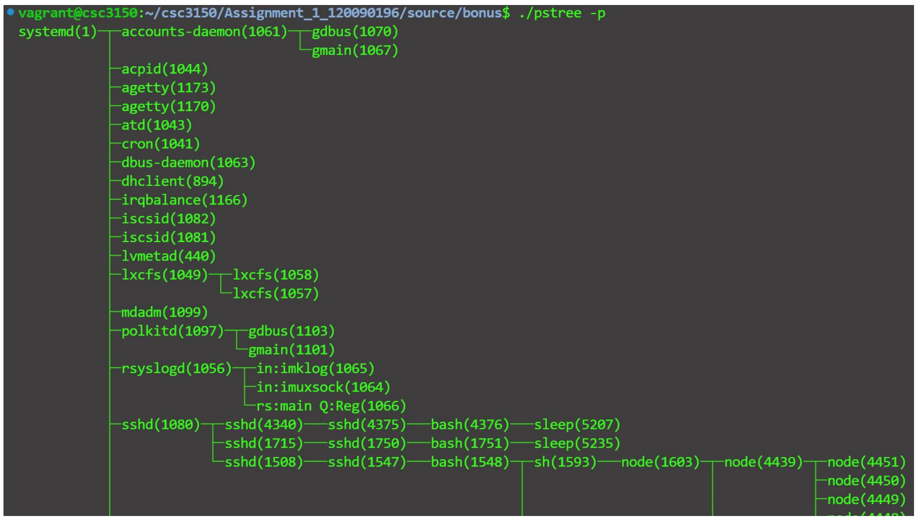
```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/bonus$ ./pstree -V
pstree v1.0
author: Zhenyu PAN
student id: 120090196
```

3. ./pstree -n (since the terminal is limited, just show part of it)



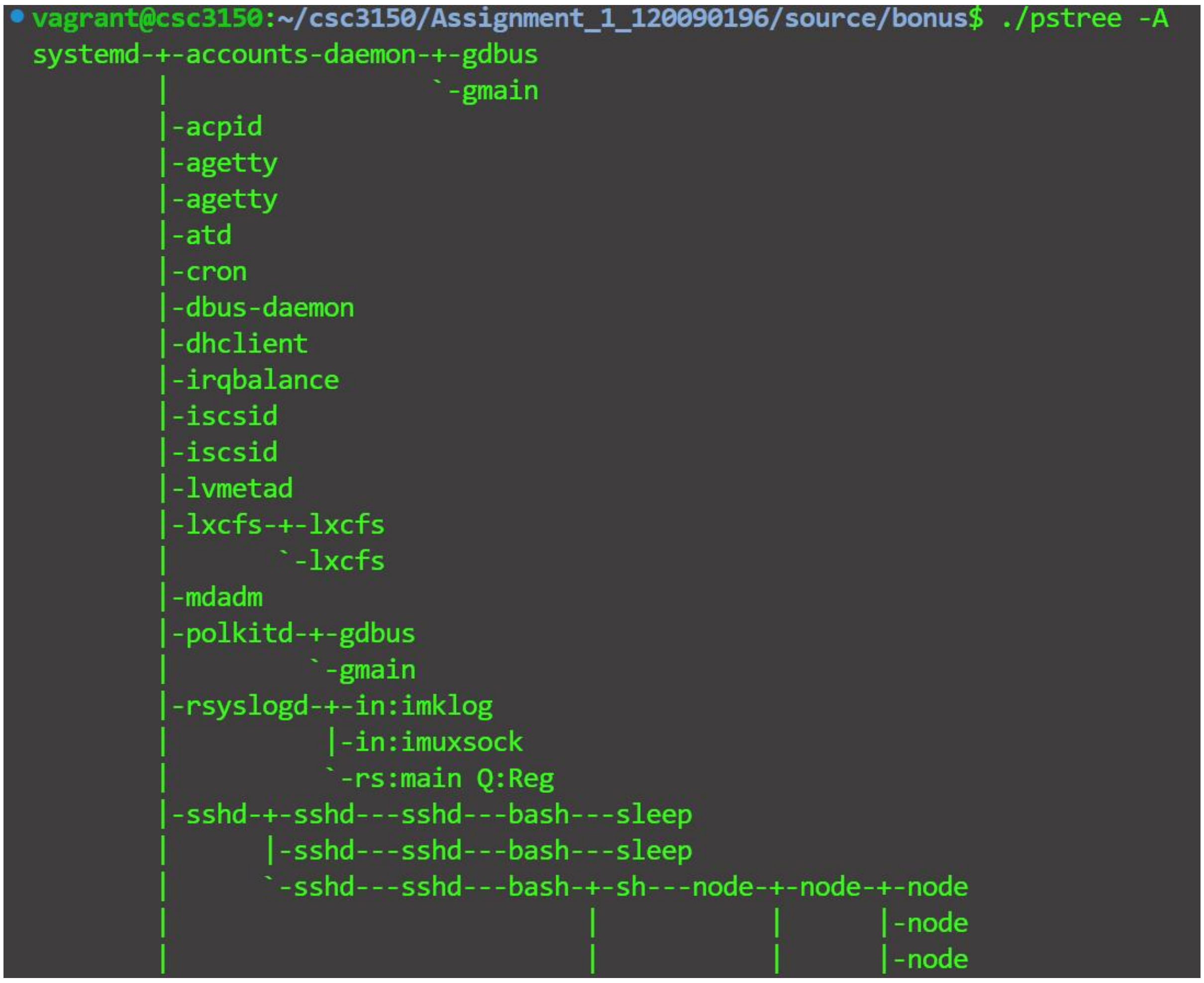
```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/bonus$ ./pstree -n
systemd─┬─systemd-journal
        ├─lvmetad
        ├─systemd-udevd
        ├─dhclient
        ├─cron
        ├─atd
        ├─acpid
        ├─lxcfs─┬─lxcfs
        │       └─lxcfs
        ├─systemd-logind
        ├─rsyslogd─┬─in:imuxsock
        │          ├─in:imklog
        │          └─rs:main Q:Reg
        ├─accounts-daemon─┬─gmain
        │                 └─gdbus
        ├─dbus-daemon
        ├─sshd─┬─sshd───sshd───bash─┬─sh───node─┬─node
        │      │                    │           ├─node
        │      │                    │           ├─node
        │      │                    │           ├─node
        │      │                    │           ├─node
        │      │                    │           ├─node
```

4. ./pstree -p(since the terminal is limited, just show part of it)

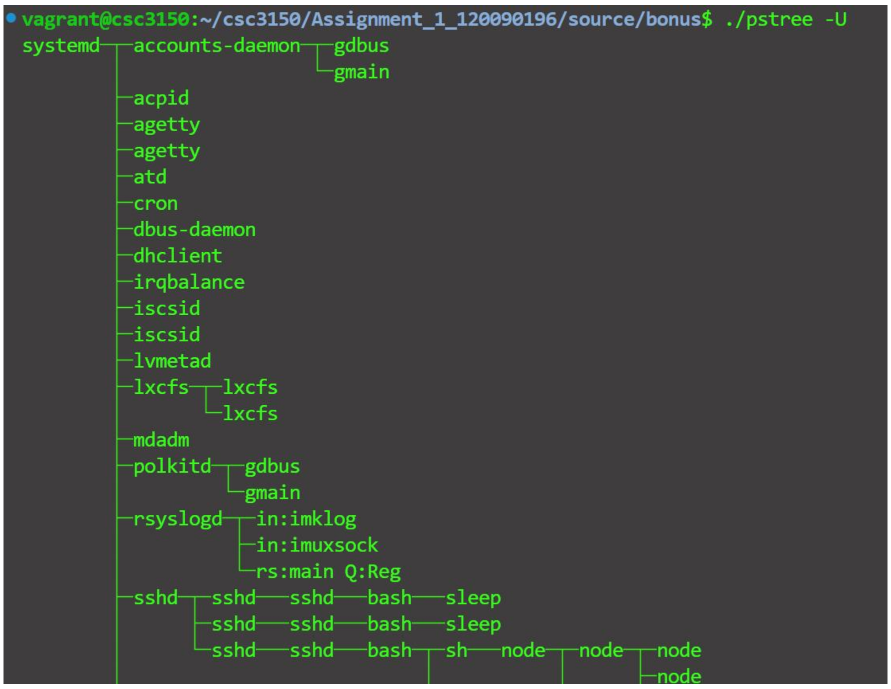


```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/bonus$ ./pstree -p
systemd(1)─┬─accounts-daemon(1061)─┬─gdbus(1070)
           │                       └─gmain(1067)
           ├─acpid(1044)
           ├─agetty(1173)
           ├─agetty(1170)
           ├─atd(1043)
           ├─cron(1041)
           ├─dbus-daemon(1063)
           ├─dhclient(894)
           ├─irqbalance(1166)
           ├─iscsid(1082)
           ├─iscsid(1081)
           ├─lvmetad(440)
           ├─lxcfs(1049)─┬─lxcfs(1058)
           │             └─lxcfs(1057)
           ├─mdadm(1099)
           ├─polkitd(1097)─┬─gdbus(1103)
           │               └─gmain(1101)
           ├─rsyslogd(1056)─┬─in:imklog(1065)
           │                ├─in:imuxsock(1064)
           │                └─rs:main Q:Reg(1066)
           ├─sshd(1080)─┬─sshd(4340)───sshd(4375)───bash(4376)───sleep(5207)
           │            ├─sshd(1715)───sshd(1750)───bash(1751)───sleep(5235)
           │            └─sshd(1508)───sshd(1547)───bash(1548)─┬─sh(1593)───node(1603)─┬─node(4439)─┬─node(4451)
           │                                                   │                       │            ├─node(4450)
           │                                                   │                       │            ├─node(4449)
```

5. ./pstree -A(since the terminal is limited, just show part of it))


```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/bonus$ ./pstree -A
systemd-+-accounts-daemon-+-gdbus
        |                 `-gmain
        |-acpid
        |-agetty
        |-agetty
        |-atd
        |-cron
        |-dbus-daemon
        |-dhclient
        |-irqbalance
        |-iscsid
        |-iscsid
        |-lvmetad
        |-lxcfs-+-lxcfs
        |       `-lxcfs
        |-mdadm
        |-polkitd-+-gdbus
        |         `-gmain
        |-rsyslogd-+-in:imklog
        |          |-in:imuxsock
        |          `-rs:main Q:Reg
        |-sshd-+-sshd---sshd---bash---sleep
        |      |-sshd---sshd---bash---sleep
        |      `-sshd---sshd---bash-+-sh---node-+-node-+-node
        |                           |           |      |-node
        |                           |           |      |-node
```

6. ./pstree -U(since the terminal is limited, just show part of it))


```
vagrant@csc3150:~/csc3150/Assignment_1_120090196/source/bonus$ ./pstree -U
systemd─┬─accounts-daemon─┬─gdbus
        │                 └─gmain
        ├─acpid
        ├─agetty
        ├─agetty
        ├─atd
        ├─cron
        ├─dbus-daemon
        ├─dhclient
        ├─irqbalance
        ├─iscsid
        ├─iscsid
        ├─lvmetad
        ├─lxcfs─┬─lxcfs
        │       └─lxcfs
        ├─mdadm
        ├─polkitd─┬─gdbus
        │         └─gmain
        ├─rsyslogd─┬─in:imklog
        │          ├─in:imuxsock
        │          └─rs:main Q:Reg
        ├─sshd─┬─sshd───sshd───bash───sleep
        │      ├─sshd───sshd───bash───sleep
        │      └─sshd───sshd───bash─┬─sh───node─┬─node─┬─node
        │                           │           │      └─node
```

What I learned from the project

1. Most importantly, start **as early as possible**! The earlier, the better!
2. Get familiar with C, K&R is a good book.
3. When debugging, do not rely on debugger because there might be bug in debugger.
4. Learn more about Linux, the reference book and the source code are both fantastic.
5. From tast1, I learned how to create child processes and how to execute other programs in the process using ***execve()*** function. Also, I learned about what the common signals are and the meaning of them.
6. From task2, I learned how to **EXPORT** the functions we need to invoke from the kernel using vim command. Also, I learned about a little on how to read Linux source code and how to insert and remove modules to kernel.
7. From bonus, I gained a deeper understanding of the Unix system core: everything is a file. The state information of the currently running program is also stored in files. Under a special file directory ***proc***, there are folders whose names are the corresponding process ids. So what ***pstree*** does is parse this folder, which stores all the state information about the process.
   In this program, the main is to parse ***stat*** file, get pid, name, state, ppid and so on.
8. Read piazza from time to time.