CSC3150 Fall 2022

Project 1 Report

Student ID: 120090437 Name: HU, Wenxi
2022.10.10

Program design

Task 1.

The purpose of Task 1 is to fork multiple process based on user mode, execute test programs and print signal information. We can do this in user mode by calling *fork(), execve()*, and *waitpid()*.

In the main body of the program, the *fork()* function is used to execute the child and parent processes simultaneously, and the pid information about each process is printed out. The parent process uses the *waitpid()* function to wait for the child process to execute first.

The child process obtains the execution file name from the command line, executes the file using the *execve()* function, and prints the program execution information at the terminal.

After waiting for the execution of the child process, the parent process receives the status returned by the child process, obtains the signal in normal/SIG*/stop, determines the specific signal value through the *switch case*, and prints the information on the terminal.

Task 2.

Task 2 is to fork multiple processes in kernel mode, which means the implementation of fork, execute, and wait will be more closely related to the kernel, mainly through *kthread_create, kernel_thread, do_execve, do_wait, getname_kernel*.

First, in the *module_init* function, create a kernel thread using *kthread_create* and execute the *my_fork* function.

In the *my_fork* function, generate a child process using the *kernel_thread()* function and have it execute the *my_exec* function. The parent is waiting for the signal passed by the child by the function *my_wait*.

In *my_exec*, I use the *do_execve()* function and *kernel_getname()* function to execute the specified executable file.

In *my_wait*, I use *do_wait()*. The information of the child process's status is stored in * *wo.wo_stat* variable. After the child is finished, the parent will receive the child's signal and use *switch case* to distinguish the signal, print out the corresponding information to kernel log.

Finally, exit the module.

Bonus.

To perform the function of pstree, that is, to print out the process tree in the system and execute the option function.

I first need to get the end user's instruction options through the *getopt_long ()* function and execute the different option functions separately.

In the second step, to obtain the process and thread information in the system, you need to use *opendir* and *readdir* to obtain the pid information of all processes in /proc. Open these pid folders to read the process name and ppid of the corresponding parent process. To get information about the thread, and to distinguish the thread name by curly braces in the print, we need to get information such as the thread name and ppid from */proc/[pid]/task/ tid /stat* and mark it as a thread. **(These two steps get succeeded)**

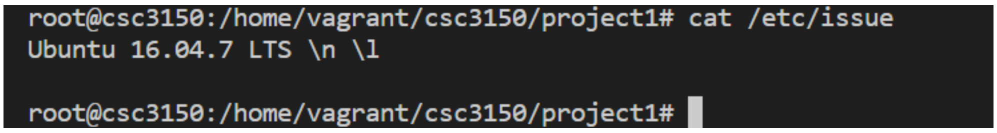
The third step is to connect the read information through a linked list, marking the relationship between parent and child processes. **(This step still gets error in my implementation)**

Finally, according to the specific information of each process, the result of the process tree is formatted and printed in the terminal.

Environment

OS Version:

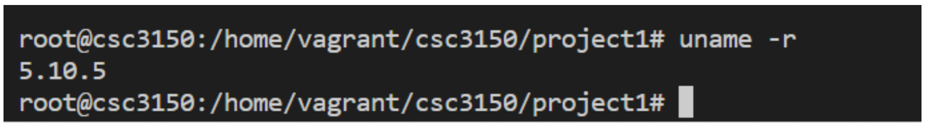
Linux Ubuntu 16.04.7



```
root@csc3150:/home/vagrant/csc3150/project1# cat /etc/issue
Ubuntu 16.04.7 LTS \n \l

root@csc3150:/home/vagrant/csc3150/project1#
```

Kernel Version:

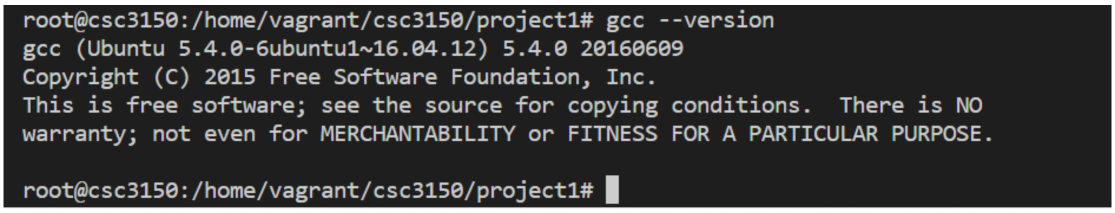
5.10.5



```
root@csc3150:/home/vagrant/csc3150/project1# uname -r
5.10.5
root@csc3150:/home/vagrant/csc3150/project1#
```

Gcc Version

5.4.0



```
root@csc3150:/home/vagrant/csc3150/project1# gcc --version
gcc (Ubuntu 5.4.0-6ubuntu1~16.04.12) 5.4.0 20160609
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

root@csc3150:/home/vagrant/csc3150/project1#
```

Set up environment

Download 5.10.5 kernel source code from http://www.kernel.org

Install Dependency and development tools:

sudo apt-get install libncurses-dev gawk flex bison openssl libssl-dev dkms libelf-devlibudev-dev libpci-dev libiberty-dev autoconf llvm dwarves

Extract the source file to /home/vagrant/csc3150:

cp KERNEL_FILE.tar.xz /home/vagrant/csc3150

cd /home/seed/work

$sudo tar xvf KERNEL_FILE.tar.xz

Copy config from /boot to /home/vagrant/csc3150/linux-5.10.5

Login root account and go to kernel source directory

$sudo su

$cd /home/vagrant/csc3150/linux-5.10.5

Clean previous setting and start configuration

$make mrproper

$make clean

$make menuconfig

save the config and exit

Update the kernel source code (add EXPORT_SYMBOL() for 4 functions invoked)

Build kernel Image and modules

$make –j$(nproc)

Install kernel modules

$make modules_install

Install kernel

$make install

Reboot to load new kernel

$reboot

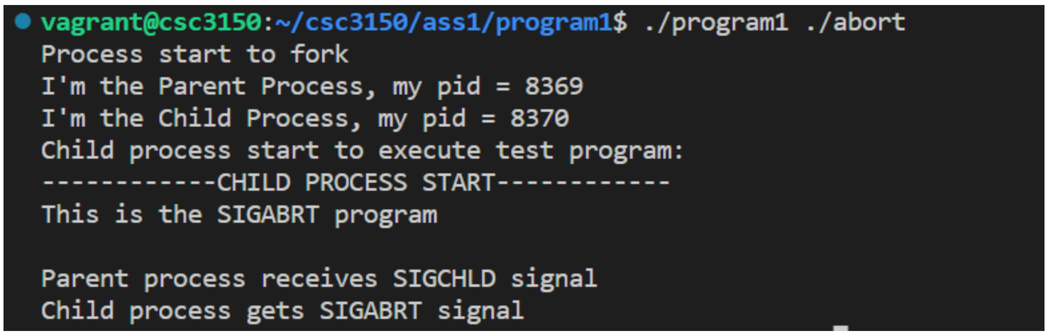
To execute task2, you should use *gcc -o test test.c* to compile the test file, use *insmod program2.ko, rmmod program2.ko,* and *dmesg* to install, remove the module and check the kernel log.

Before executing all the programs, you should type *make* to compile the programs.

<u>Program Output</u>

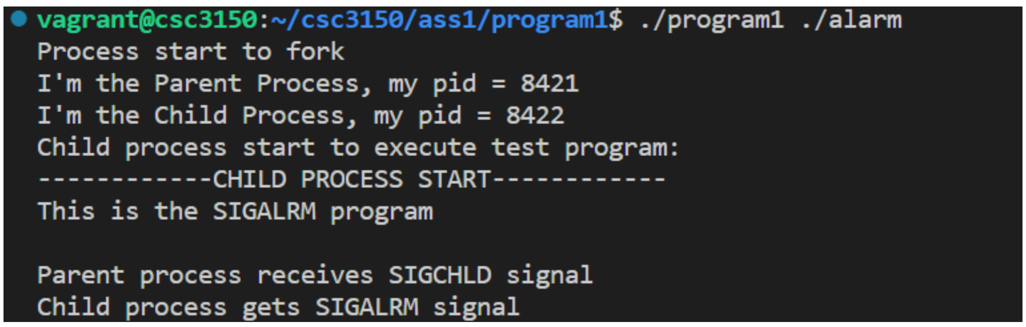
Task1.

SIGABRT



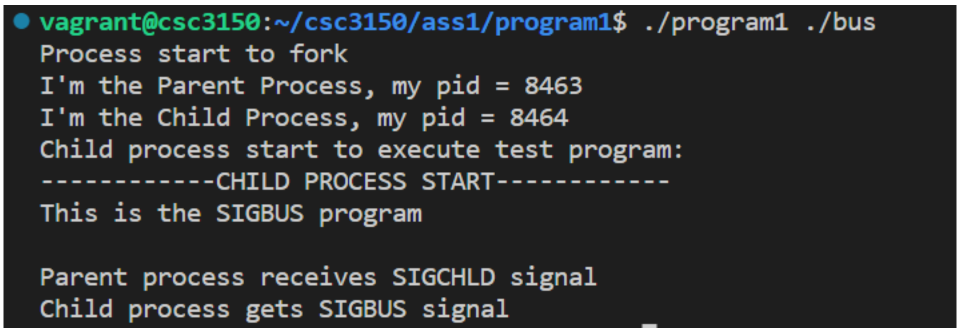
```
vagrant@csc3150:~/csc3150/ass1/program1$ ./program1 ./abort
Process start to fork
I'm the Parent Process, my pid = 8369
I'm the Child Process, my pid = 8370
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGABRT program

Parent process receives SIGCHLD signal
Child process gets SIGABRT signal
```

SIGALRM



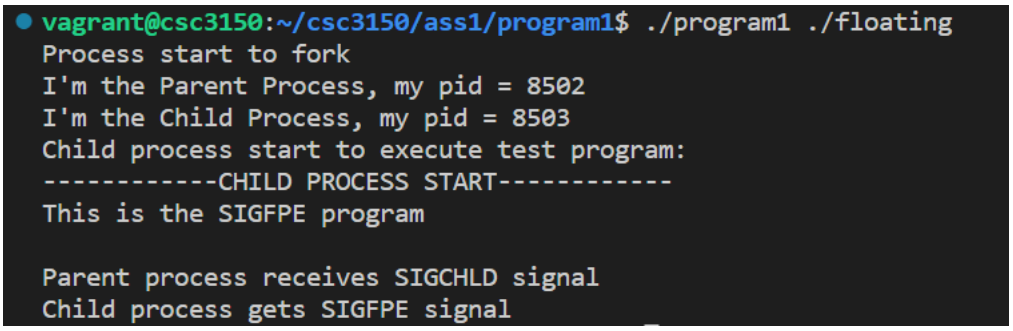
```
vagrant@csc3150:~/csc3150/ass1/program1$ ./program1 ./alarm
Process start to fork
I'm the Parent Process, my pid = 8421
I'm the Child Process, my pid = 8422
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGALRM program

Parent process receives SIGCHLD signal
Child process gets SIGALRM signal
```

SIGBUS



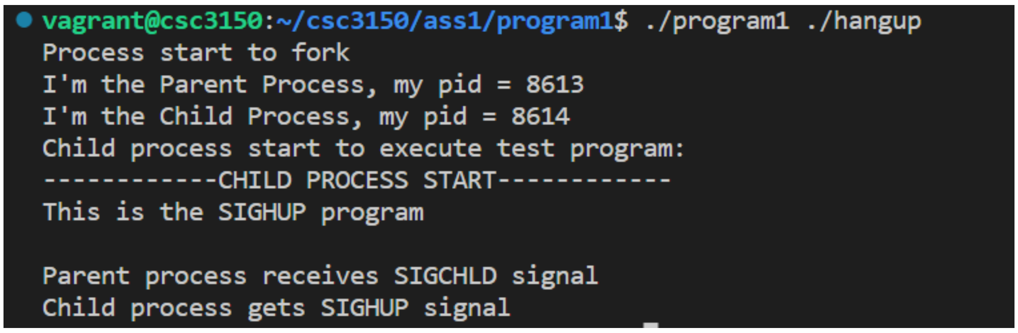
```
vagrant@csc3150:~/csc3150/ass1/program1$ ./program1 ./bus
Process start to fork
I'm the Parent Process, my pid = 8463
I'm the Child Process, my pid = 8464
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGBUS program

Parent process receives SIGCHLD signal
Child process gets SIGBUS signal
```

SIGFPE



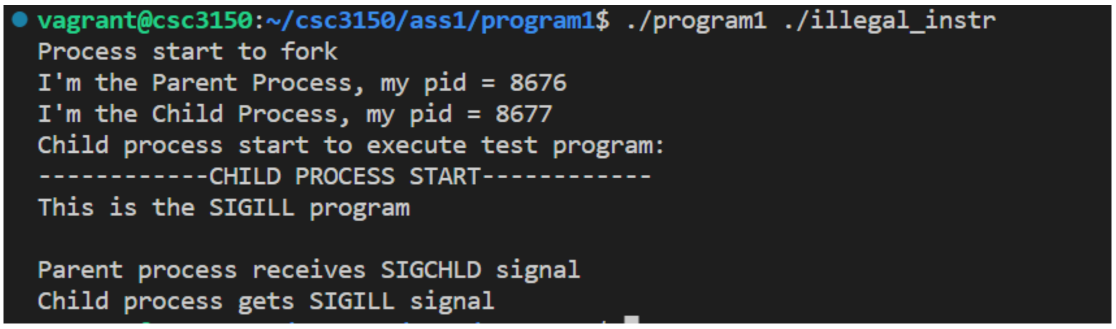
```
vagrant@csc3150:~/csc3150/ass1/program1$ ./program1 ./floating
Process start to fork
I'm the Parent Process, my pid = 8502
I'm the Child Process, my pid = 8503
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGFPE program

Parent process receives SIGCHLD signal
Child process gets SIGFPE signal
```

SIGHUP



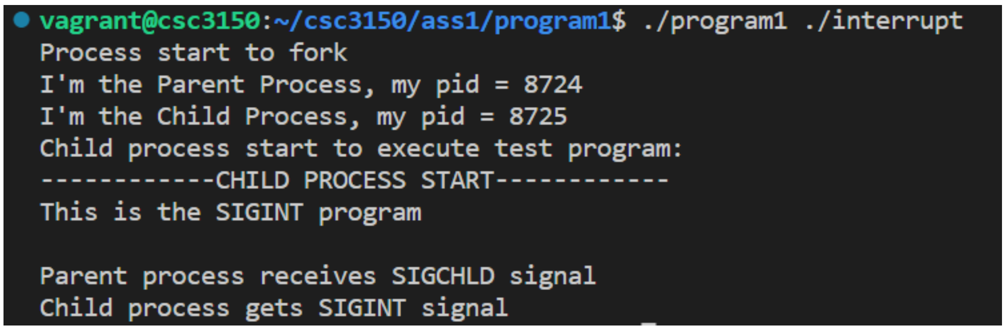
```
vagrant@csc3150:~/csc3150/ass1/program1$ ./program1 ./hangup
Process start to fork
I'm the Parent Process, my pid = 8613
I'm the Child Process, my pid = 8614
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGHUP program

Parent process receives SIGCHLD signal
Child process gets SIGHUP signal
```

SIGILL

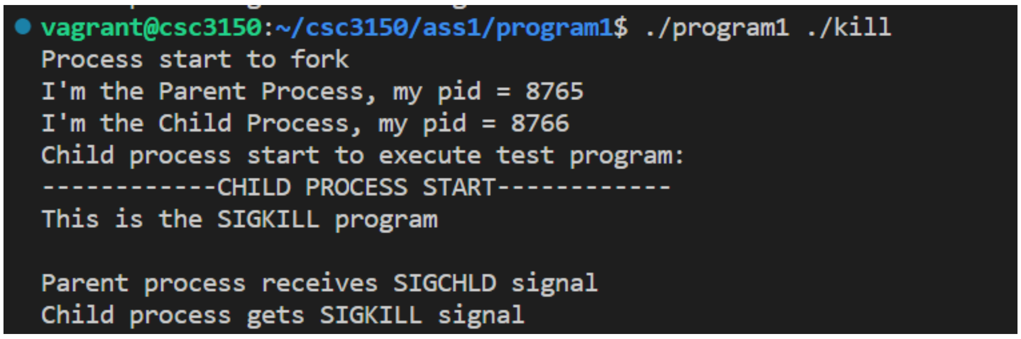


```
vagrant@csc3150:~/csc3150/ass1/program1$ ./program1 ./illegal_instr
Process start to fork
I'm the Parent Process, my pid = 8676
I'm the Child Process, my pid = 8677
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGILL program

Parent process receives SIGCHLD signal
Child process gets SIGILL signal
```

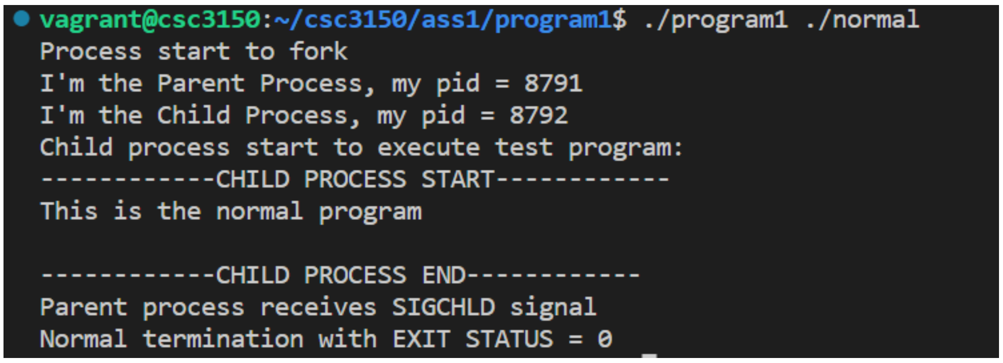
SIGINT


```
vagrant@csc3150:~/csc3150/ass1/program1$ ./program1 ./interrupt
Process start to fork
I'm the Parent Process, my pid = 8724
I'm the Child Process, my pid = 8725
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGINT program

Parent process receives SIGCHLD signal
Child process gets SIGINT signal
```

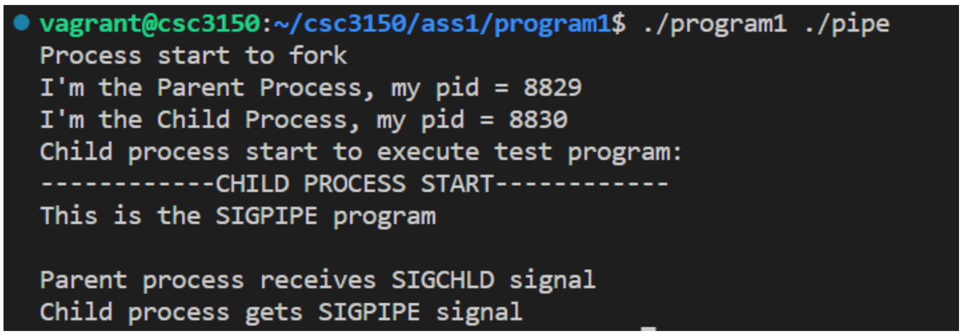
SIGKILL


```
vagrant@csc3150:~/csc3150/ass1/program1$ ./program1 ./kill
Process start to fork
I'm the Parent Process, my pid = 8765
I'm the Child Process, my pid = 8766
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGKILL program

Parent process receives SIGCHLD signal
Child process gets SIGKILL signal
```

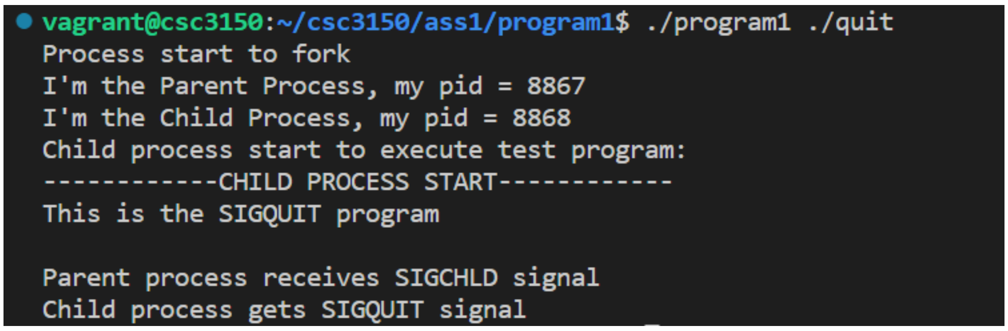
normal


```
vagrant@csc3150:~/csc3150/ass1/program1$ ./program1 ./normal
Process start to fork
I'm the Parent Process, my pid = 8791
I'm the Child Process, my pid = 8792
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
```

SIGPIPE

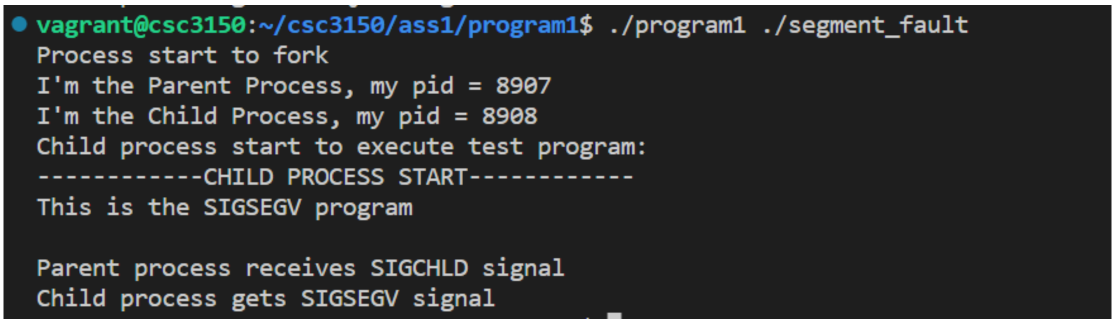

```
vagrant@csc3150:~/csc3150/ass1/program1$ ./program1 ./pipe
Process start to fork
I'm the Parent Process, my pid = 8829
I'm the Child Process, my pid = 8830
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGPIPE program

Parent process receives SIGCHLD signal
Child process gets SIGPIPE signal
```

SIGQUIT



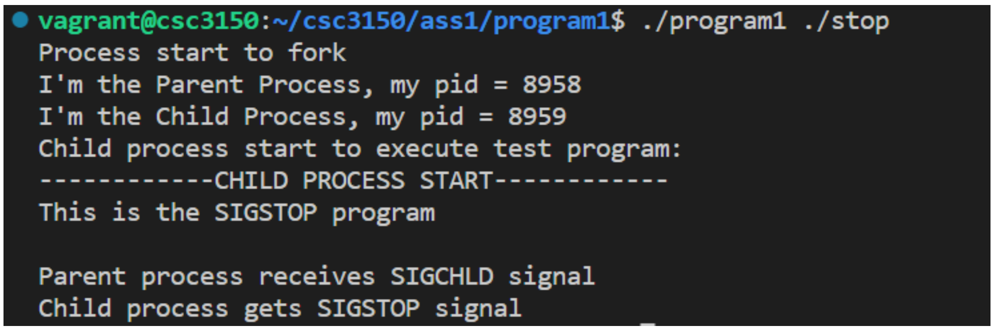
```
vagrant@csc3150:~/csc3150/ass1/program1$ ./program1 ./quit
Process start to fork
I'm the Parent Process, my pid = 8867
I'm the Child Process, my pid = 8868
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

Parent process receives SIGCHLD signal
Child process gets SIGQUIT signal
```

SIGSEGV



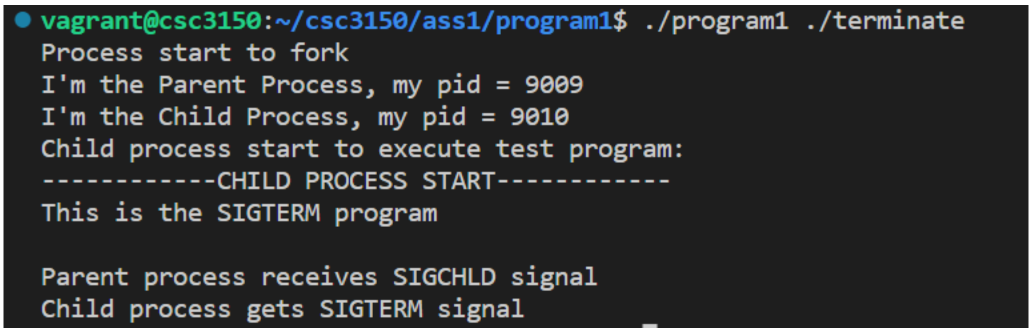
```
vagrant@csc3150:~/csc3150/ass1/program1$ ./program1 ./segment_fault
Process start to fork
I'm the Parent Process, my pid = 8907
I'm the Child Process, my pid = 8908
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

Parent process receives SIGCHLD signal
Child process gets SIGSEGV signal
```

SIGSTOP



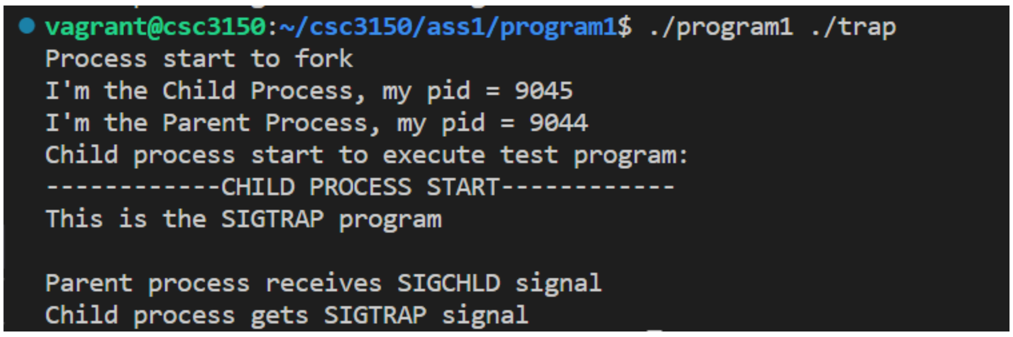
```
vagrant@csc3150:~/csc3150/ass1/program1$ ./program1 ./stop
Process start to fork
I'm the Parent Process, my pid = 8958
I'm the Child Process, my pid = 8959
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

Parent process receives SIGCHLD signal
Child process gets SIGSTOP signal
```

SIGTERM



```
vagrant@csc3150:~/csc3150/ass1/program1$ ./program1 ./terminate
Process start to fork
I'm the Parent Process, my pid = 9009
I'm the Child Process, my pid = 9010
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTERM program

Parent process receives SIGCHLD signal
Child process gets SIGTERM signal
```

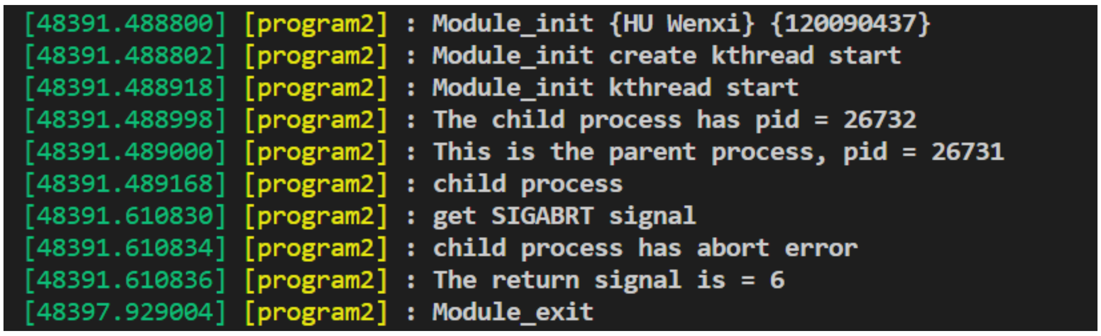
SIGTRAP



```
vagrant@csc3150:~/csc3150/ass1/program1$ ./program1 ./trap
Process start to fork
I'm the Child Process, my pid = 9045
I'm the Parent Process, my pid = 9044
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTRAP program

Parent process receives SIGCHLD signal
Child process gets SIGTRAP signal
```

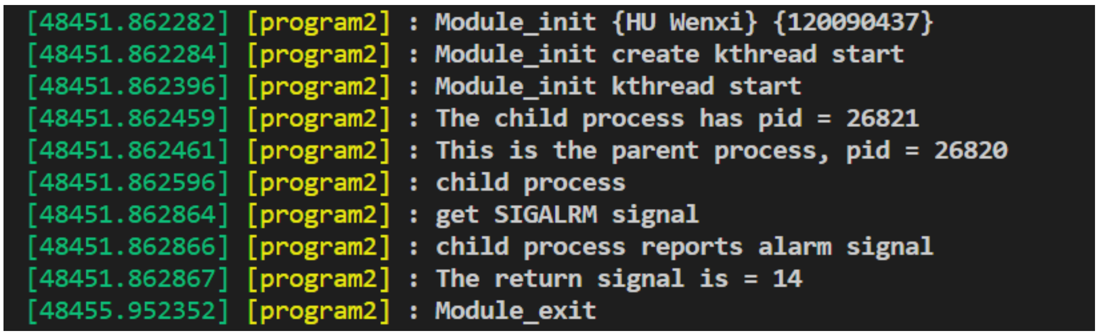
Task2.

SIGABRT



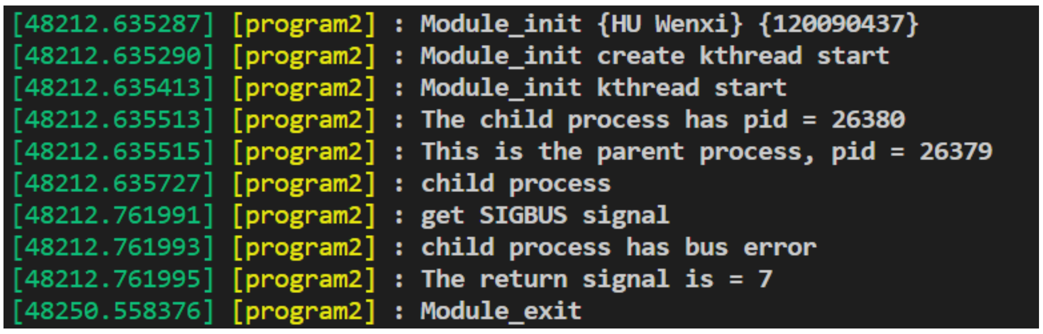
```
[48391.488800] [program2] : Module_init {HU Wenxi} {120090437}
[48391.488802] [program2] : Module_init create kthread start
[48391.488918] [program2] : Module_init kthread start
[48391.488998] [program2] : The child process has pid = 26732
[48391.489000] [program2] : This is the parent process, pid = 26731
[48391.489168] [program2] : child process
[48391.610830] [program2] : get SIGABRT signal
[48391.610834] [program2] : child process has abort error
[48391.610836] [program2] : The return signal is = 6
[48397.929004] [program2] : Module_exit
```

SIGALRM



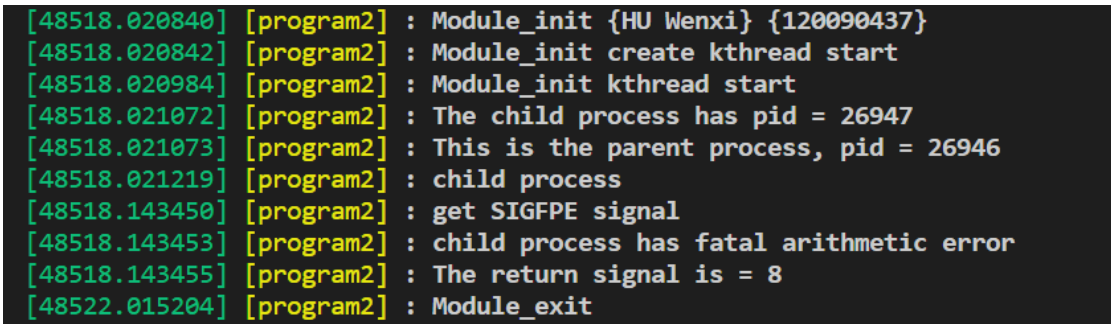
```
[48451.862282] [program2] : Module_init {HU Wenxi} {120090437}
[48451.862284] [program2] : Module_init create kthread start
[48451.862396] [program2] : Module_init kthread start
[48451.862459] [program2] : The child process has pid = 26821
[48451.862461] [program2] : This is the parent process, pid = 26820
[48451.862596] [program2] : child process
[48451.862864] [program2] : get SIGALRM signal
[48451.862866] [program2] : child process reports alarm signal
[48451.862867] [program2] : The return signal is = 14
[48455.952352] [program2] : Module_exit
```

SIGBUS

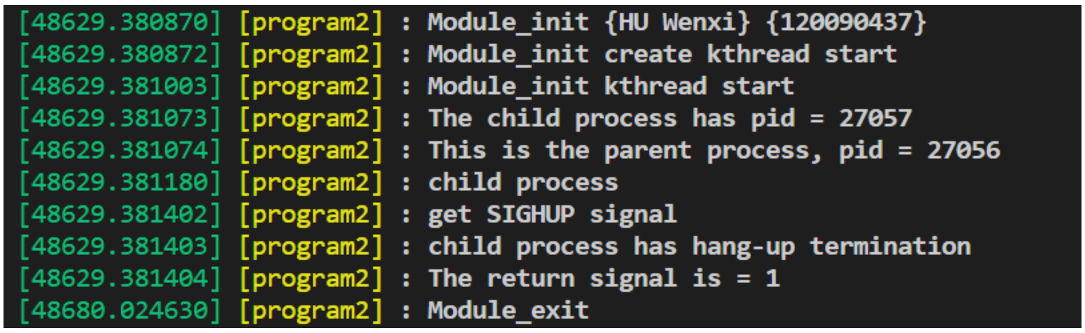


```
[48212.635287] [program2] : Module_init {HU Wenxi} {120090437}
[48212.635290] [program2] : Module_init create kthread start
[48212.635413] [program2] : Module_init kthread start
[48212.635513] [program2] : The child process has pid = 26380
[48212.635515] [program2] : This is the parent process, pid = 26379
[48212.635727] [program2] : child process
[48212.761991] [program2] : get SIGBUS signal
[48212.761993] [program2] : child process has bus error
[48212.761995] [program2] : The return signal is = 7
[48250.558376] [program2] : Module_exit
```

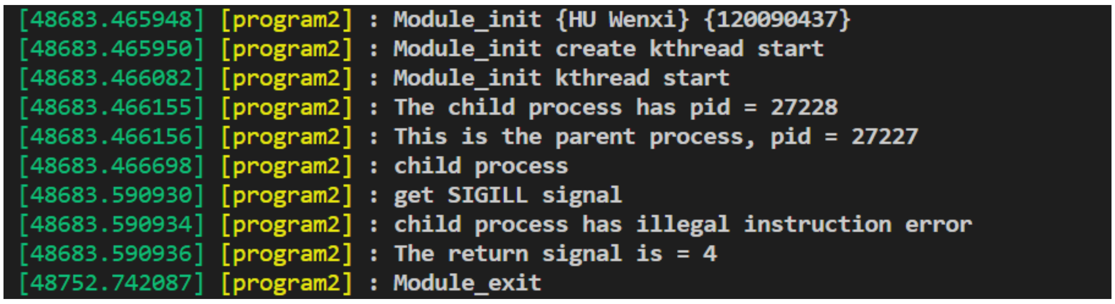
SIGFPE


```
[48518.020840] [program2] : Module_init {HU Wenxi} {120090437}
[48518.020842] [program2] : Module_init create kthread start
[48518.020984] [program2] : Module_init kthread start
[48518.021072] [program2] : The child process has pid = 26947
[48518.021073] [program2] : This is the parent process, pid = 26946
[48518.021219] [program2] : child process
[48518.143450] [program2] : get SIGFPE signal
[48518.143453] [program2] : child process has fatal arithmetic error
[48518.143455] [program2] : The return signal is = 8
[48522.015204] [program2] : Module_exit
```

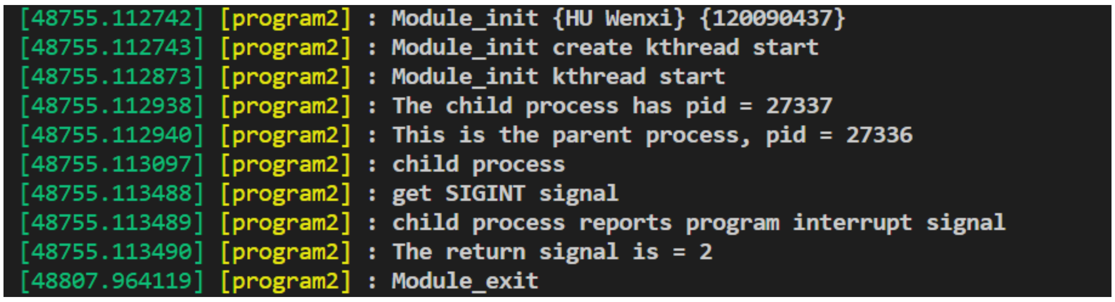
SIGHUP


```
[48629.380870] [program2] : Module_init {HU Wenxi} {120090437}
[48629.380872] [program2] : Module_init create kthread start
[48629.381003] [program2] : Module_init kthread start
[48629.381073] [program2] : The child process has pid = 27057
[48629.381074] [program2] : This is the parent process, pid = 27056
[48629.381180] [program2] : child process
[48629.381402] [program2] : get SIGHUP signal
[48629.381403] [program2] : child process has hang-up termination
[48629.381404] [program2] : The return signal is = 1
[48680.024630] [program2] : Module_exit
```

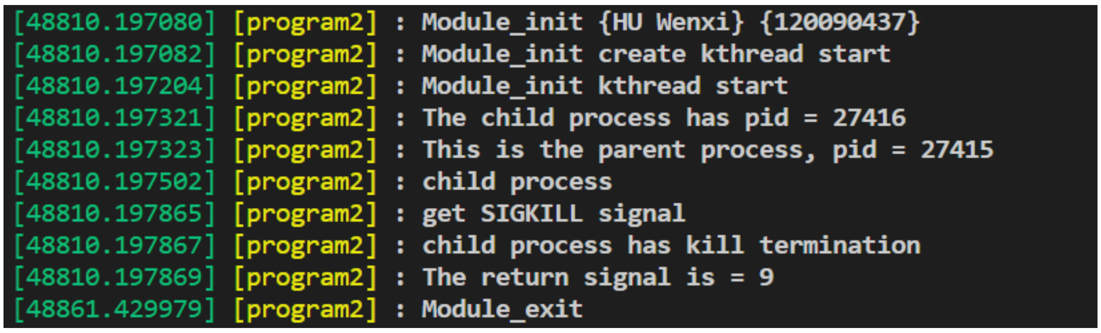
SIGILL


```
[48683.465948] [program2] : Module_init {HU Wenxi} {120090437}
[48683.465950] [program2] : Module_init create kthread start
[48683.466082] [program2] : Module_init kthread start
[48683.466155] [program2] : The child process has pid = 27228
[48683.466156] [program2] : This is the parent process, pid = 27227
[48683.466698] [program2] : child process
[48683.590930] [program2] : get SIGILL signal
[48683.590934] [program2] : child process has illegal instruction error
[48683.590936] [program2] : The return signal is = 4
[48752.742087] [program2] : Module_exit
```

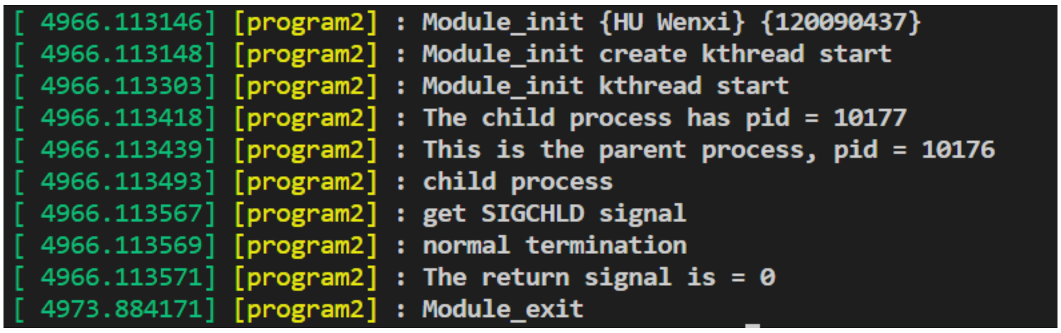
SIGINT


```
[48755.112742] [program2] : Module_init {HU Wenxi} {120090437}
[48755.112743] [program2] : Module_init create kthread start
[48755.112873] [program2] : Module_init kthread start
[48755.112938] [program2] : The child process has pid = 27337
[48755.112940] [program2] : This is the parent process, pid = 27336
[48755.113097] [program2] : child process
[48755.113488] [program2] : get SIGINT signal
[48755.113489] [program2] : child process reports program interrupt signal
[48755.113490] [program2] : The return signal is = 2
[48807.964119] [program2] : Module_exit
```

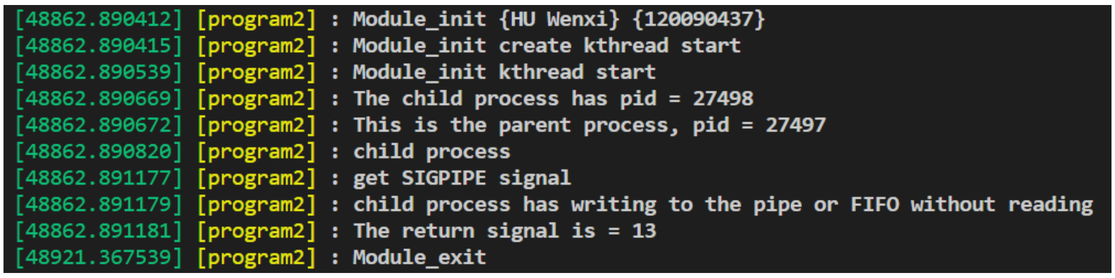
SIGKILL


```
[48810.197080] [program2] : Module_init {HU Wenxi} {120090437}
[48810.197082] [program2] : Module_init create kthread start
[48810.197204] [program2] : Module_init kthread start
[48810.197321] [program2] : The child process has pid = 27416
[48810.197323] [program2] : This is the parent process, pid = 27415
[48810.197502] [program2] : child process
[48810.197865] [program2] : get SIGKILL signal
[48810.197867] [program2] : child process has kill termination
[48810.197869] [program2] : The return signal is = 9
[48861.429979] [program2] : Module_exit
```

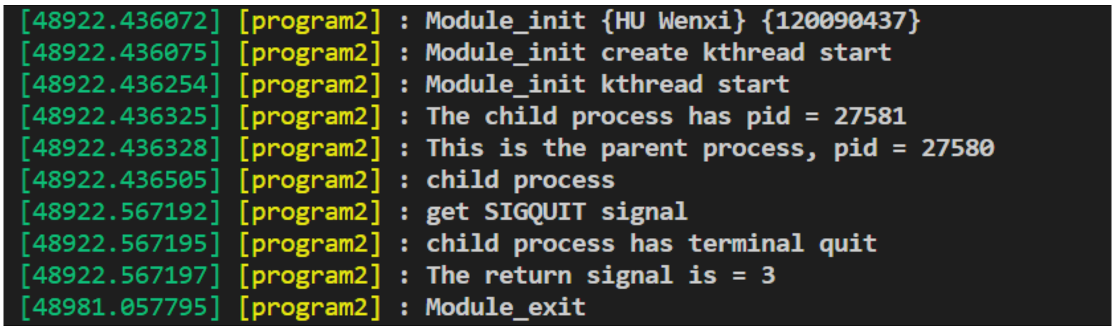
normal


```
[ 4966.113146] [program2] : Module_init {HU Wenxi} {120090437}
[ 4966.113148] [program2] : Module_init create kthread start
[ 4966.113303] [program2] : Module_init kthread start
[ 4966.113418] [program2] : The child process has pid = 10177
[ 4966.113439] [program2] : This is the parent process, pid = 10176
[ 4966.113493] [program2] : child process
[ 4966.113567] [program2] : get SIGCHLD signal
[ 4966.113569] [program2] : normal termination
[ 4966.113571] [program2] : The return signal is = 0
[ 4973.884171] [program2] : Module_exit
```

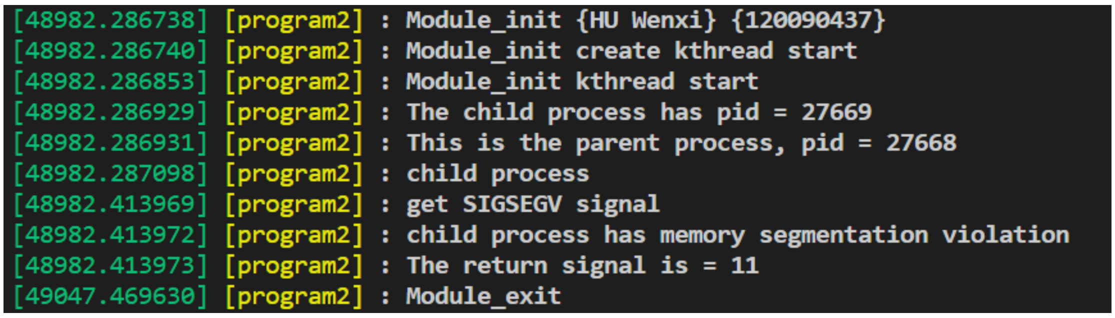
SIGPIPE


```
[48862.890412] [program2] : Module_init {HU Wenxi} {120090437}
[48862.890415] [program2] : Module_init create kthread start
[48862.890539] [program2] : Module_init kthread start
[48862.890669] [program2] : The child process has pid = 27498
[48862.890672] [program2] : This is the parent process, pid = 27497
[48862.890820] [program2] : child process
[48862.891177] [program2] : get SIGPIPE signal
[48862.891179] [program2] : child process has writing to the pipe or FIFO without reading
[48862.891181] [program2] : The return signal is = 13
[48921.367539] [program2] : Module_exit
```

SIGQUIT

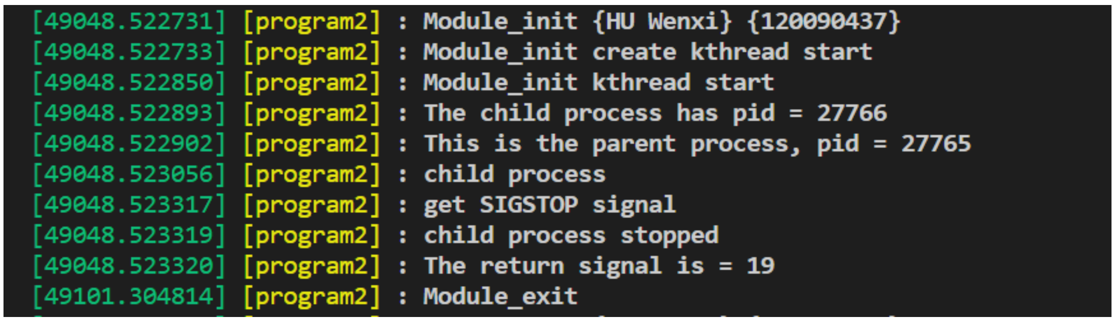

```
[48922.436072] [program2] : Module_init {HU Wenxi} {120090437}
[48922.436075] [program2] : Module_init create kthread start
[48922.436254] [program2] : Module_init kthread start
[48922.436325] [program2] : The child process has pid = 27581
[48922.436328] [program2] : This is the parent process, pid = 27580
[48922.436505] [program2] : child process
[48922.567192] [program2] : get SIGQUIT signal
[48922.567195] [program2] : child process has terminal quit
[48922.567197] [program2] : The return signal is = 3
[48981.057795] [program2] : Module_exit
```

SIGSEGV



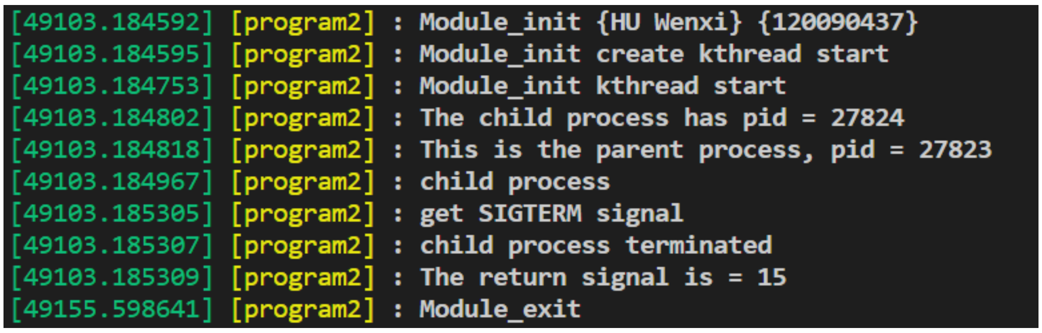
```
[48982.286738] [program2] : Module_init {HU Wenxi} {120090437}
[48982.286740] [program2] : Module_init create kthread start
[48982.286853] [program2] : Module_init kthread start
[48982.286929] [program2] : The child process has pid = 27669
[48982.286931] [program2] : This is the parent process, pid = 27668
[48982.287098] [program2] : child process
[48982.413969] [program2] : get SIGSEGV signal
[48982.413972] [program2] : child process has memory segmentation violation
[48982.413973] [program2] : The return signal is = 11
[49047.469630] [program2] : Module_exit
```

SIGSTOP



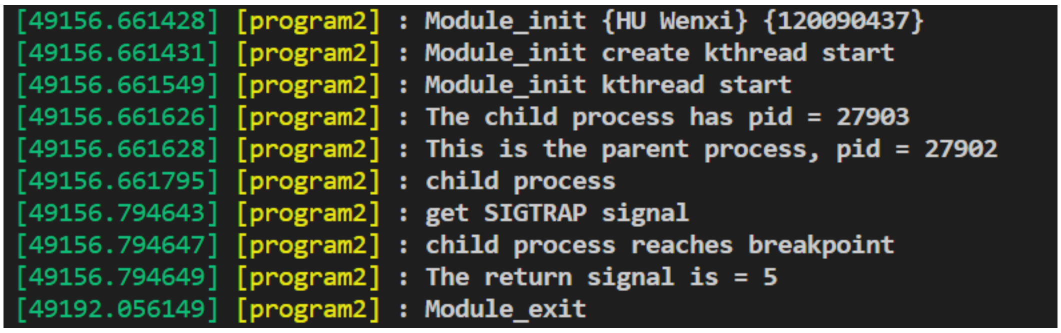
```
[49048.522731] [program2] : Module_init {HU Wenxi} {120090437}
[49048.522733] [program2] : Module_init create kthread start
[49048.522850] [program2] : Module_init kthread start
[49048.522893] [program2] : The child process has pid = 27766
[49048.522902] [program2] : This is the parent process, pid = 27765
[49048.523056] [program2] : child process
[49048.523317] [program2] : get SIGSTOP signal
[49048.523319] [program2] : child process stopped
[49048.523320] [program2] : The return signal is = 19
[49101.304814] [program2] : Module_exit
```

SIGTERM



```
[49103.184592] [program2] : Module_init {HU Wenxi} {120090437}
[49103.184595] [program2] : Module_init create kthread start
[49103.184753] [program2] : Module_init kthread start
[49103.184802] [program2] : The child process has pid = 27824
[49103.184818] [program2] : This is the parent process, pid = 27823
[49103.184967] [program2] : child process
[49103.185305] [program2] : get SIGTERM signal
[49103.185307] [program2] : child process terminated
[49103.185309] [program2] : The return signal is = 15
[49155.598641] [program2] : Module_exit
```

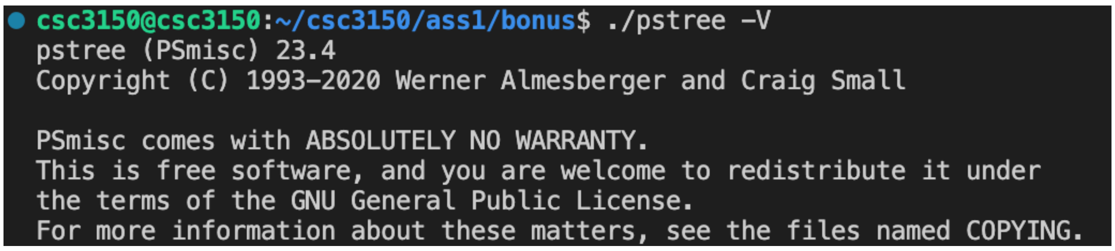
SIGTRAP



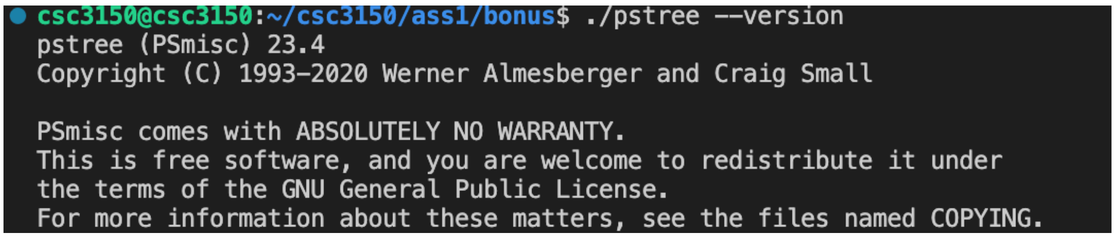
```
[49156.661428] [program2] : Module_init {HU Wenxi} {120090437}
[49156.661431] [program2] : Module_init create kthread start
[49156.661549] [program2] : Module_init kthread start
[49156.661626] [program2] : The child process has pid = 27903
[49156.661628] [program2] : This is the parent process, pid = 27902
[49156.661795] [program2] : child process
[49156.794643] [program2] : get SIGTRAP signal
[49156.794647] [program2] : child process reaches breakpoint
[49156.794649] [program2] : The return signal is = 5
[49192.056149] [program2] : Module_exit
```

Bonus.

./pstree -V (./pstree –version)



```
● csc3150@csc3150:~/csc3150/ass1/bonus$ ./pstree -V
  pstree (PSmisc) 23.4
  Copyright (C) 1993-2020 Werner Almesberger and Craig Small

  PSmisc comes with ABSOLUTELY NO WARRANTY.
  This is free software, and you are welcome to redistribute it under
  the terms of the GNU General Public License.
  For more information about these matters, see the files named COPYING.
```



```
● csc3150@csc3150:~/csc3150/ass1/bonus$ ./pstree --version
  pstree (PSmisc) 23.4
  Copyright (C) 1993-2020 Werner Almesberger and Craig Small

  PSmisc comes with ABSOLUTELY NO WARRANTY.
  This is free software, and you are welcome to redistribute it under
  the terms of the GNU General Public License.
  For more information about these matters, see the files named COPYING.
```

What I have learned

Assignment 1 itself is not particularly technically difficult, but it is still troublesome. For the knowledge and coding part, we need to know some definitions of processes and kernels. The tutorial is clear about this part and provides reference examples so that I can understand the operation of processes more intuitively. But working difficulty depends on the use of C language and the familiarity, because before this I haven't systematically exposed to C. Learning some C++ didn't provide me with opportunities to independently write a complete program from scratch, which left mea very big challenge (especially the Bonus for C language understanding and implementation) in this assignment. Although I could not say that I had achieved proficiency in C language and Pointers, and the completion of the program was not perfect, I got the opportunity to practice the use of C language and Pointers in this process.
Another difficulty in this assignment is that we need to build our own environment and compile the kernel. In this process, the system will produce numerous and diverse errors and bugs. Especially for me, a novice who is not familiar with the Linux system, I need to often search for solutions during the system kernel installation, modification, compilation, and use of Linux instructions. After working on this project, I became more proficient with Linux systems and virtual machines. I learned a few things about processes, how to configure the environment, upgrade tools, compile and modify the kernel, how to execute a program in code, etc.