Assignment 1 Report

120090890 Jingyu Huang

Task 1

The Design of the Program:

For the design part of the task 1, first I need to fork a child process to execute the test process. In the user mode, I just use the fork() as the tut slide constructed. Then using the return value of the fork(), I separate the error one and the normal one. Then in the normal ones, using the different return value of fork(), I can separate the child process and the parent process.

Second, I need to let the parent process to receive the signal from the child process when the child process terminated. Here in order to wait for the termination of the child process and terminate the parent process when receiving SIGSTOP signal, I use waitpid() instead of wait() since the options parameter WUNTRACED can report on stopped child processes as well as terminated ones. Then the status value can be gained from the child process.

Third, I need to execute programs using 15 test programs. Here to achieve this, I use the status value sta to make classifications for different situations with the help of the built-in functions WIFEXITED(), WIFSIGNALED() and WIFSTOPPED(). Using the classification, I can print out the termination and exit status whether it is normal or not.

Environment Setup and Kernel Compiling

For the environment setup, I use the virtualbox and vagrant to setup the virtual machine environment. First I download these two softwares. Then I will show how I setup environment.

Install vitrualbox and vagrant. After installation, reboot the machine.

Set up a directory for csc3150 (make sure the full path does not include space or Chinese, e.g. D:\csc3150)

Launch powershell with Administrator previlege (run as administrator) and change current directory to the one you set up (e.g. cd D:\csc3150)

Execute vagrant init cyzhu/csc3150

Then execute vagrant up. It may take a while to download the system image. After that a virtualbox window may pop up. Leave it open but put it aside.

After setting up the environment, I also set up the remote ssh service in the VS Code to help me code. The concrete steps are in below. Just download the extension of Remote SSH and use vagrant ssh-config to get the ssh setup information. Copy everything but the last line (Loglevel) to the ssh config and save the config file. Then just connect to the virtual machine and use the command sudo apt update && sudo apt install -y build-essential. Then the VS Code environment is set up well.

Then I need to compile the kernel to the version 5.10.146. Here using the steps in the tut slide in the following.

Download source code from

- ◦ http://www.kernel.org
- ◦ mirro: https://mirror.tuna.tsinghua.edu.cn/kernel/v5.x/

Install Dependency and development tools

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

Extract the source file to /home/seed/work

- ◦ cp KERNEL_FILE.tar.xz /home/seed/work
- ◦ cd /home/seed/work
- ◦ $sudo tar xvf KERNEL_FILE.tar.xz

Copy config from /boot to /home/seed/work/KERNEL_FILE

Login root account and go to kernel source directory

- ◦ $sudo su
- ◦ $cd /home/seed/work /KERNEL_FILE

Clean previous setting and start configuration

- ◦ $make mrproper
- ◦ $make clean
- ◦ $make menuconfig
- ◦ save the config and exit

Build kernel Image and modules

◦ $make bzImage -j$(nproc)

◦ $make modules -j$(nproc)

◦ ~ 30 mins to finish

◦ $make –j$(nproc)

(you could use this command to replace above two commands)

Install kernel modules

◦ $make modules_install

Install kernel

◦ $make install

Reboot to load new kernel

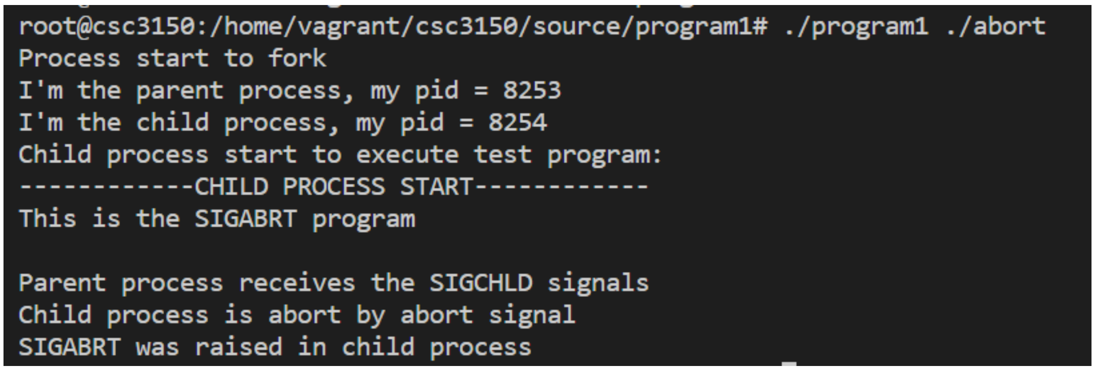
◦ $reboot

(When rebooting, you should select the updated kernel)

After all these steps, the kernel is installed well. Then I can check the version of the kernel using uname -r to verify.

Screenshots:

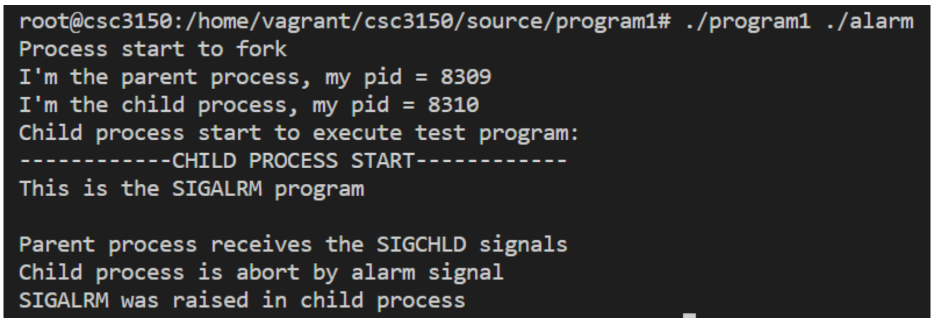
Abort:



```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./abort
Process start to fork
I'm the parent process, my pid = 8253
I'm the child process, my pid = 8254
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGABRT program

Parent process receives the SIGCHLD signals
Child process is abort by abort signal
SIGABRT was raised in child process
```

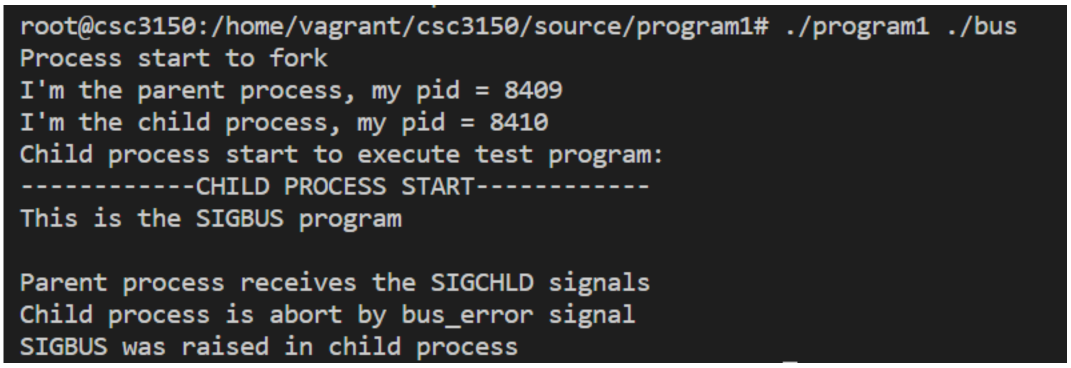
Alarm:



```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./alarm
Process start to fork
I'm the parent process, my pid = 8309
I'm the child process, my pid = 8310
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGALRM program

Parent process receives the SIGCHLD signals
Child process is abort by alarm signal
SIGALRM was raised in child process
```

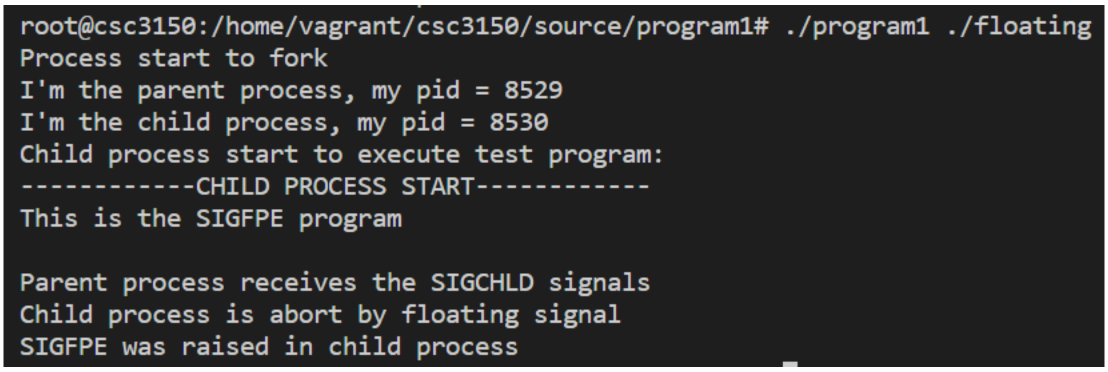
Bus:



```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./bus
Process start to fork
I'm the parent process, my pid = 8409
I'm the child process, my pid = 8410
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGBUS program

Parent process receives the SIGCHLD signals
Child process is abort by bus_error signal
SIGBUS was raised in child process
```

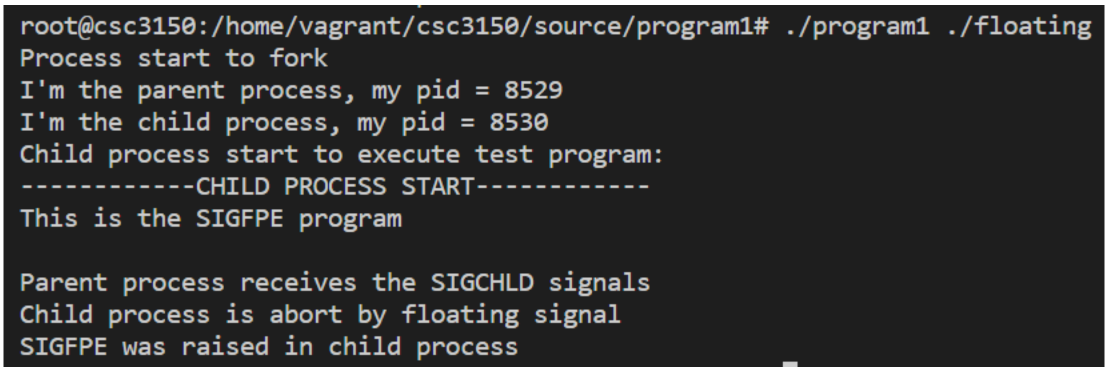
Floating:



```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./floating
Process start to fork
I'm the parent process, my pid = 8529
I'm the child process, my pid = 8530
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGFPE program

Parent process receives the SIGCHLD signals
Child process is abort by floating signal
SIGFPE was raised in child process
```

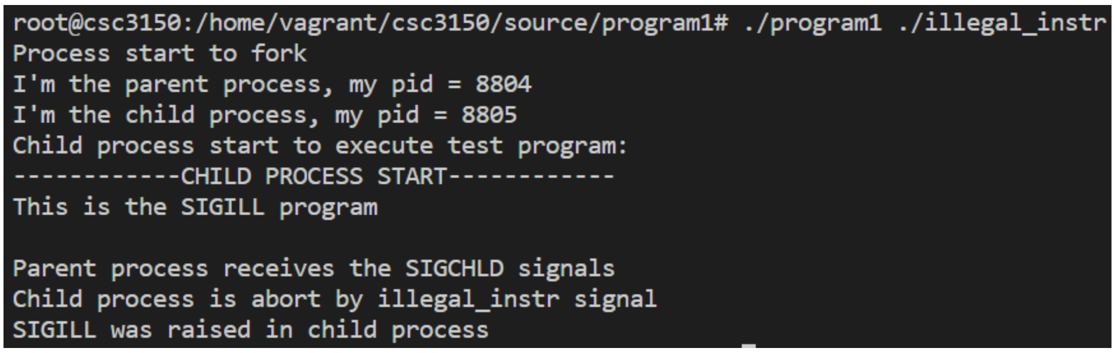
Hangup:



```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./floating
Process start to fork
I'm the parent process, my pid = 8529
I'm the child process, my pid = 8530
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGFPE program

Parent process receives the SIGCHLD signals
Child process is abort by floating signal
SIGFPE was raised in child process
```

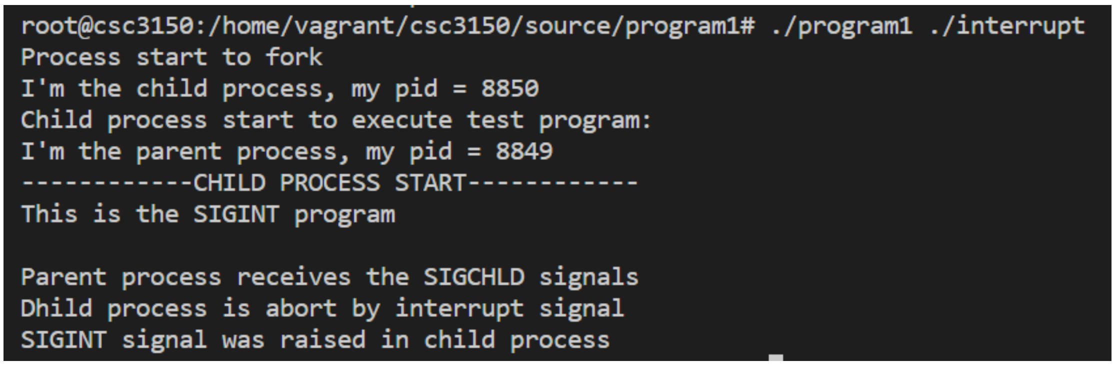
Illegal_instr:



```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./illegal_instr
Process start to fork
I'm the parent process, my pid = 8804
I'm the child process, my pid = 8805
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGILL program

Parent process receives the SIGCHLD signals
Child process is abort by illegal_instr signal
SIGILL was raised in child process
```

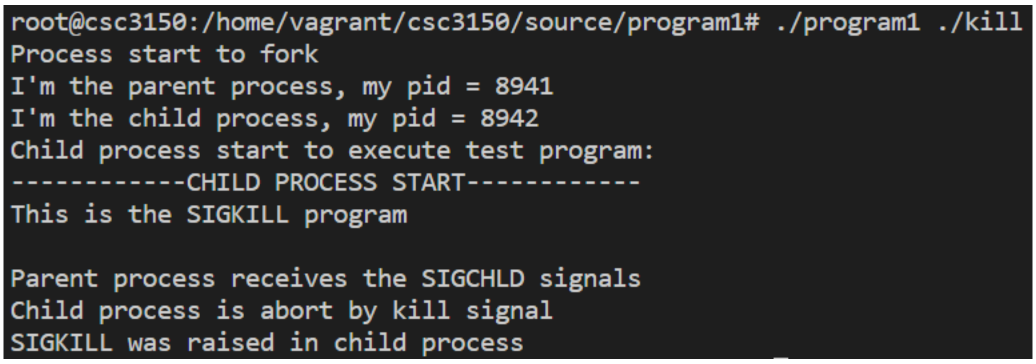
Interrupt:



```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./interrupt
Process start to fork
I'm the child process, my pid = 8850
Child process start to execute test program:
I'm the parent process, my pid = 8849
------------CHILD PROCESS START------------
This is the SIGINT program

Parent process receives the SIGCHLD signals
Dhild process is abort by interrupt signal
SIGINT signal was raised in child process
```

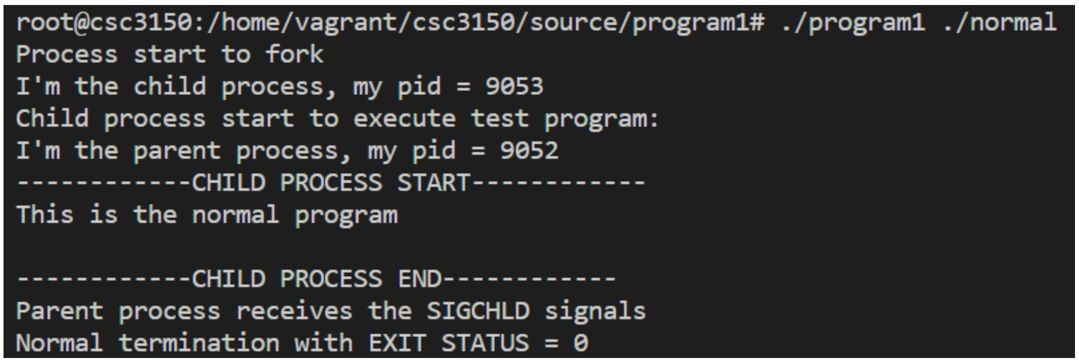
Kill:



```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./kill
Process start to fork
I'm the parent process, my pid = 8941
I'm the child process, my pid = 8942
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGKILL program

Parent process receives the SIGCHLD signals
Child process is abort by kill signal
SIGKILL was raised in child process
```

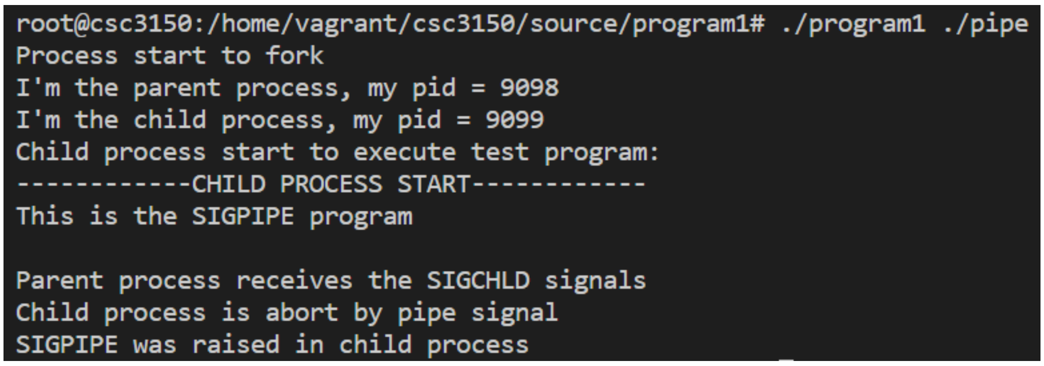
Normal:



```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./normal
Process start to fork
I'm the child process, my pid = 9053
Child process start to execute test program:
I'm the parent process, my pid = 9052
------------CHILD PROCESS START------------
This is the normal program

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

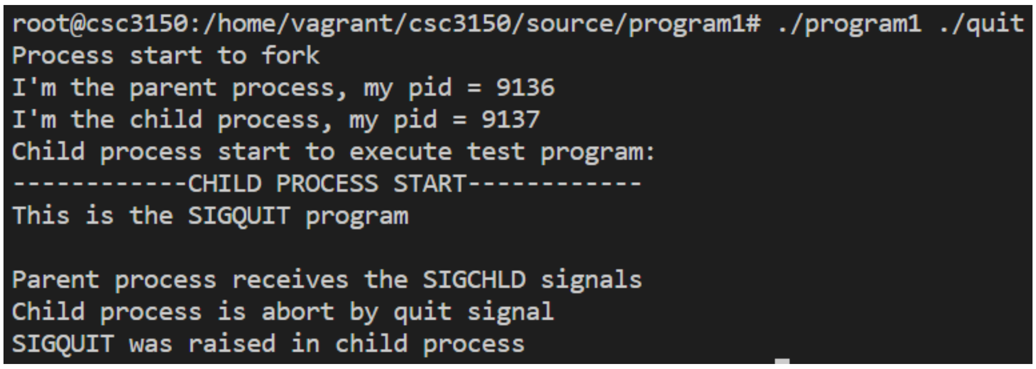
Pipe:



```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./pipe
Process start to fork
I'm the parent process, my pid = 9098
I'm the child process, my pid = 9099
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGPIPE program

Parent process receives the SIGCHLD signals
Child process is abort by pipe signal
SIGPIPE was raised in child process
```

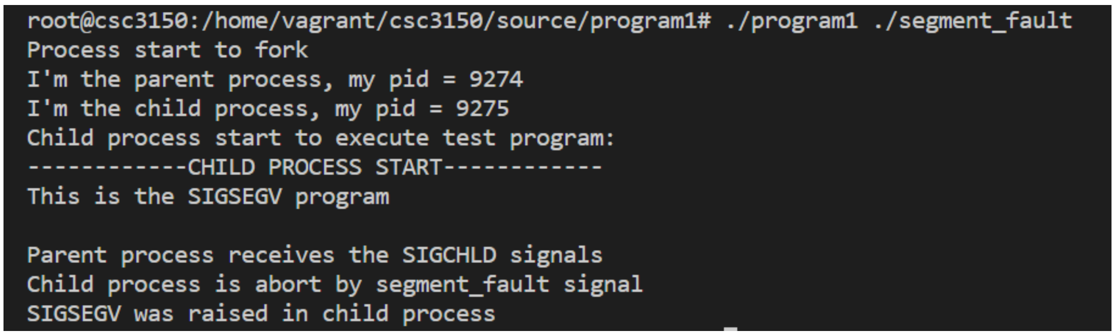
Quit:



```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./quit
Process start to fork
I'm the parent process, my pid = 9136
I'm the child process, my pid = 9137
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

Parent process receives the SIGCHLD signals
Child process is abort by quit signal
SIGQUIT was raised in child process
```

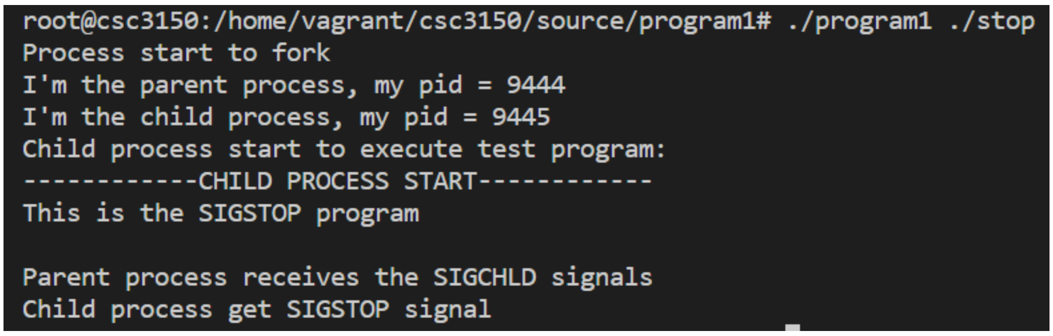
Segment_fault:



```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./segment_fault
Process start to fork
I'm the parent process, my pid = 9274
I'm the child process, my pid = 9275
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

Parent process receives the SIGCHLD signals
Child process is abort by segment_fault signal
SIGSEGV was raised in child process
```

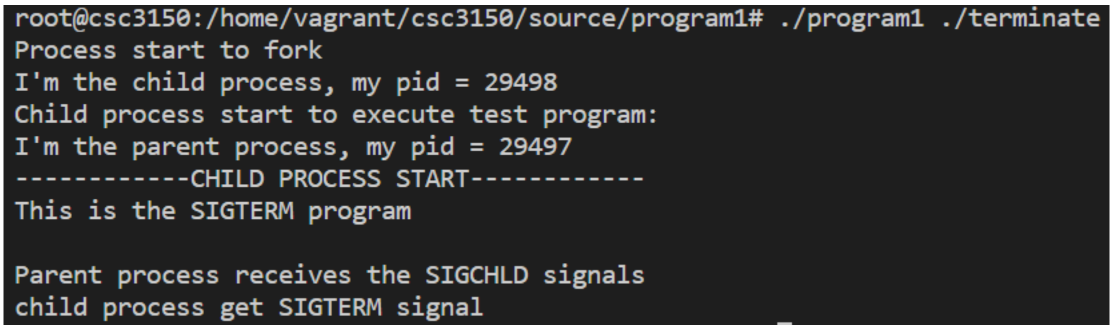
Stop:



```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./stop
Process start to fork
I'm the parent process, my pid = 9444
I'm the child process, my pid = 9445
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

Parent process receives the SIGCHLD signals
Child process get SIGSTOP signal
```

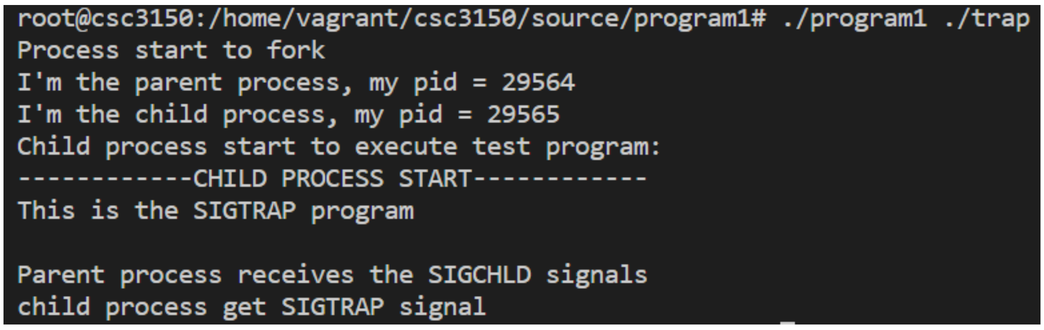
Terminate:



```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./terminate
Process start to fork
I'm the child process, my pid = 29498
Child process start to execute test program:
I'm the parent process, my pid = 29497
------------CHILD PROCESS START------------
This is the SIGTERM program

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

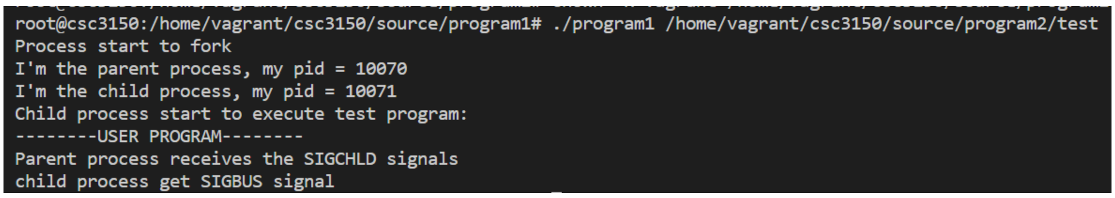
Trap:



```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 ./trap
Process start to fork
I'm the parent process, my pid = 29564
I'm the child process, my pid = 29565
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTRAP program

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

Test (In the program2):



```
root@csc3150:/home/vagrant/csc3150/source/program1# ./program1 /home/vagrant/csc3150/source/program2/test
Process start to fork
I'm the parent process, my pid = 10070
I'm the child process, my pid = 10071
Child process start to execute test program:
--------USER PROGRAM--------
Parent process receives the SIGCHLD signals
child process get SIGBUS signal
```

What I learn:

The major part of the task 1 is very basic. Here I learn the usage of the fork() function, the child process's relationship with the parent's process and the importance of signal processing. Only with proper signal processing can the process management works well.

Task 2

The Design of the Program:

For the design of the program, first following the tut slides, I first use extern and struct to list the function and struct I need to use like kernel_clone(), do_wait(), do_execve(), getname_kernel(), and the struct wait_opts.

Secondly, in order to fork a process, I write the my_fork() function which is implemented by kernel_clone(), do_execve(), getname_kernel() and do_wait(). Here kernel_clone() is actually used to clone the information from the test program to the child process. And it's return value is the pid value which can be printed out. And to execute the test program, the parameter stack of kernel_clone()'s argument is needed. With the parameter stack, it can connect to the function my_exec() in my code which is used to execute the test program with the help of do_execve(). Then the do_execve() just use getname_kernel() to get the test program file. Using these steps, a child process can be created. Then in order to let the parent process wait for the termination of the child process, using do_wait() to let the parameter wo receives the status of the child process. Then for the signal processing, just use wo_stat from wo to classify different signal and return the signal name and value.

Thirdly, it's the design for the initiation and exit function. For the exit one, just use one printk sentence to indicate the exit status in the kernel mode. For the initiation one, create a variable task, using kthread_create to create a kernel thread and run my_fork() to start to fork the process.

Environment Setup and Kernel Compiling

For the task 2, since do_execve(), do_wait(), getname_kernel() and kernel_clone() are non-static functions, I need to add export_symbol to the linux kernel code in fork.c, namei.c and exit.c. After the revising the kernel code, I recompile the kernel. The concrete steps are in the followings.

Build kernel Image and modules

◦ $make bzImage -j$(nproc)

◦ $make modules -j$(nproc)

◦ ~ 30 mins to finish

◦ $make –j$(nproc)

(you could use this command to replace above two commands)

Install kernel modules

◦ $make modules_install

Install kernel

◦ $make install

Reboot to load new kernel

◦ $reboot

(When rebooting, you should select the updated kernel)

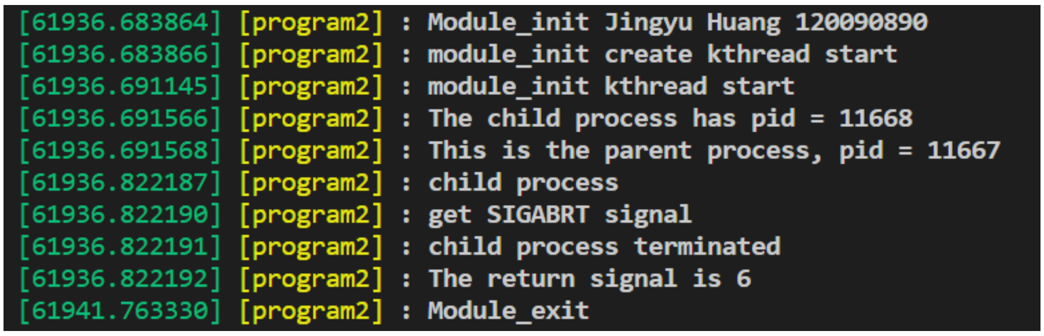
After these steps, the kernel can be recompiled well. Then if is needed to use the functions above, just add extern to these function in the code.

Screenshots

For the output, I need to notice that since I use the kthread_create in the tut slide, when first time execute the program it will show "process 'MyThread' launched '/home/vagrant/csc3150/source/program2/test' with NULL argv: empty string added".

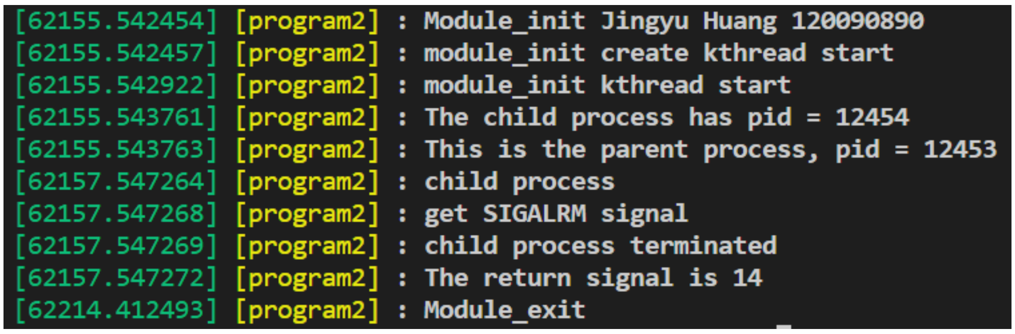
But the notice will disappear when executing the program for more times. And actually it has no effect to the output but it will show one notice in the terminal.

Abort:



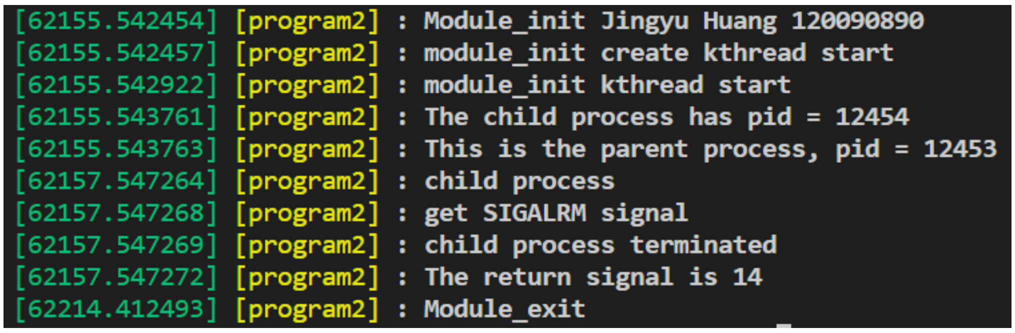
```
[61936.683864] [program2] : Module_init Jingyu Huang 120090890
[61936.683866] [program2] : module_init create kthread start
[61936.691145] [program2] : module_init kthread start
[61936.691566] [program2] : The child process has pid = 11668
[61936.691568] [program2] : This is the parent process, pid = 11667
[61936.822187] [program2] : child process
[61936.822190] [program2] : get SIGABRT signal
[61936.822191] [program2] : child process terminated
[61936.822192] [program2] : The return signal is 6
[61941.763330] [program2] : Module_exit
```

Alarm:



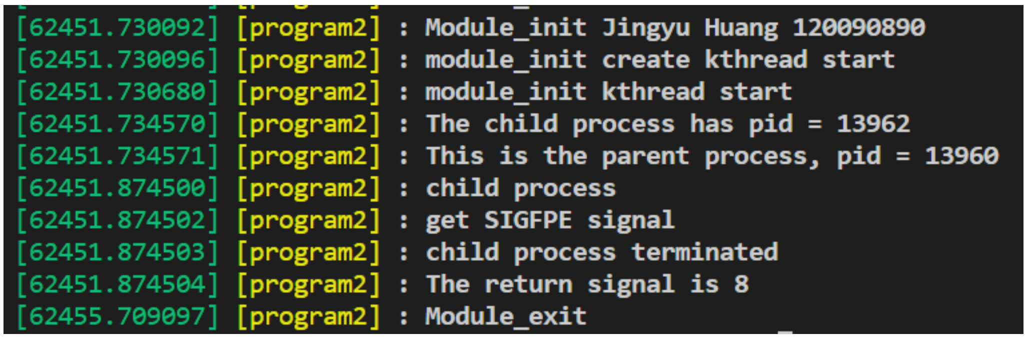
```
[62155.542454] [program2] : Module_init Jingyu Huang 120090890
[62155.542457] [program2] : module_init create kthread start
[62155.542922] [program2] : module_init kthread start
[62155.543761] [program2] : The child process has pid = 12454
[62155.543763] [program2] : This is the parent process, pid = 12453
[62157.547264] [program2] : child process
[62157.547268] [program2] : get SIGALRM signal
[62157.547269] [program2] : child process terminated
[62157.547272] [program2] : The return signal is 14
[62214.412493] [program2] : Module_exit
```

Bus:



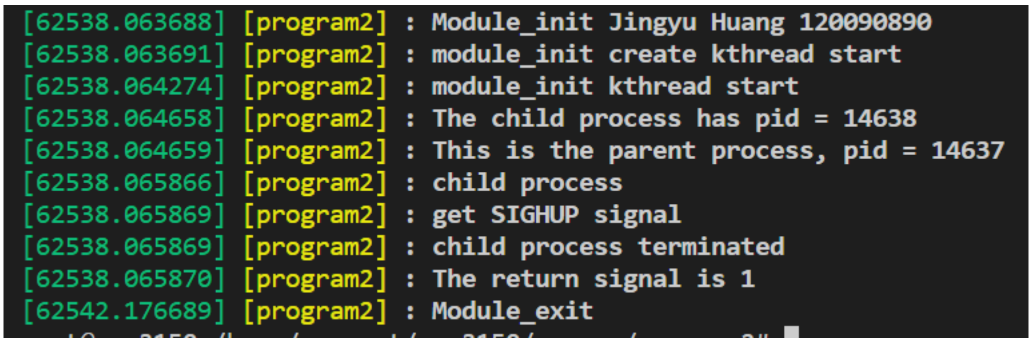
```
[62155.542454] [program2] : Module_init Jingyu Huang 120090890
[62155.542457] [program2] : module_init create kthread start
[62155.542922] [program2] : module_init kthread start
[62155.543761] [program2] : The child process has pid = 12454
[62155.543763] [program2] : This is the parent process, pid = 12453
[62157.547264] [program2] : child process
[62157.547268] [program2] : get SIGALRM signal
[62157.547269] [program2] : child process terminated
[62157.547272] [program2] : The return signal is 14
[62214.412493] [program2] : Module_exit
```

Floating:



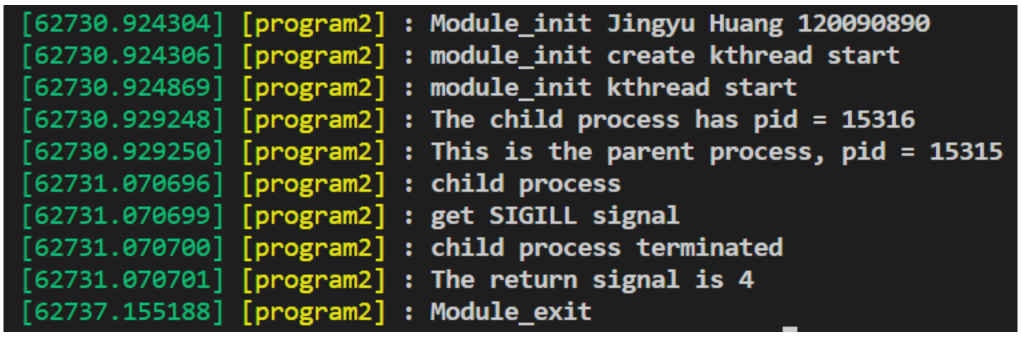
```
[62451.730092] [program2] : Module_init Jingyu Huang 120090890
[62451.730096] [program2] : module_init create kthread start
[62451.730680] [program2] : module_init kthread start
[62451.734570] [program2] : The child process has pid = 13962
[62451.734571] [program2] : This is the parent process, pid = 13960
[62451.874500] [program2] : child process
[62451.874502] [program2] : get SIGFPE signal
[62451.874503] [program2] : child process terminated
[62451.874504] [program2] : The return signal is 8
[62455.709097] [program2] : Module_exit
```

Hangup:



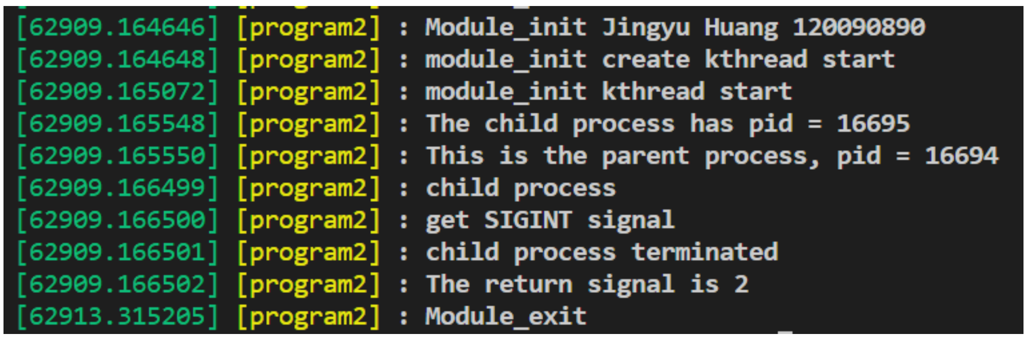
```
[62538.063688] [program2] : Module_init Jingyu Huang 120090890
[62538.063691] [program2] : module_init create kthread start
[62538.064274] [program2] : module_init kthread start
[62538.064658] [program2] : The child process has pid = 14638
[62538.064659] [program2] : This is the parent process, pid = 14637
[62538.065866] [program2] : child process
[62538.065869] [program2] : get SIGHUP signal
[62538.065869] [program2] : child process terminated
[62538.065870] [program2] : The return signal is 1
[62542.176689] [program2] : Module_exit
```

Illegal_instr:



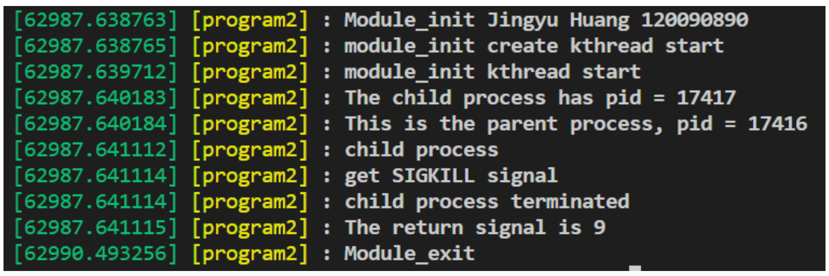
```
[62730.924304] [program2] : Module_init Jingyu Huang 120090890
[62730.924306] [program2] : module_init create kthread start
[62730.924869] [program2] : module_init kthread start
[62730.929248] [program2] : The child process has pid = 15316
[62730.929250] [program2] : This is the parent process, pid = 15315
[62731.070696] [program2] : child process
[62731.070699] [program2] : get SIGILL signal
[62731.070700] [program2] : child process terminated
[62731.070701] [program2] : The return signal is 4
[62737.155188] [program2] : Module_exit
```

Interrupt:



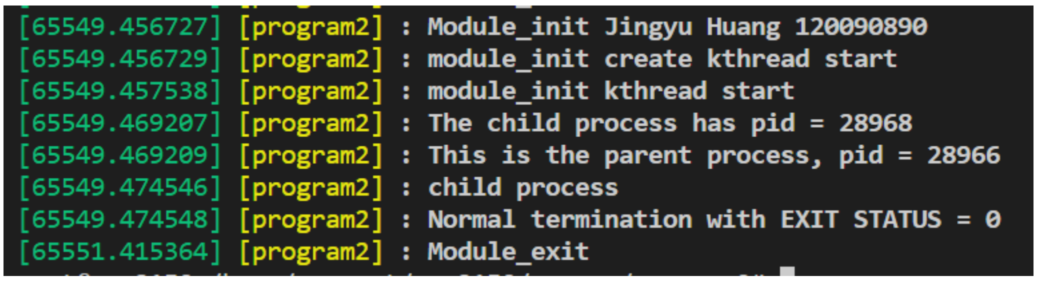
```
[62909.164646] [program2] : Module_init Jingyu Huang 120090890
[62909.164648] [program2] : module_init create kthread start
[62909.165072] [program2] : module_init kthread start
[62909.165548] [program2] : The child process has pid = 16695
[62909.165550] [program2] : This is the parent process, pid = 16694
[62909.166499] [program2] : child process
[62909.166500] [program2] : get SIGINT signal
[62909.166501] [program2] : child process terminated
[62909.166502] [program2] : The return signal is 2
[62913.315205] [program2] : Module_exit
```

Kill:



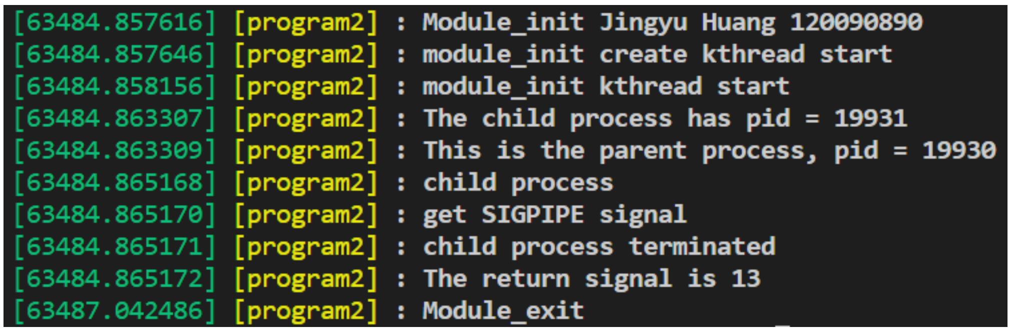
```
[62987.638763] [program2] : Module_init Jingyu Huang 120090890
[62987.638765] [program2] : module_init create kthread start
[62987.639712] [program2] : module_init kthread start
[62987.640183] [program2] : The child process has pid = 17417
[62987.640184] [program2] : This is the parent process, pid = 17416
[62987.641112] [program2] : child process
[62987.641114] [program2] : get SIGKILL signal
[62987.641114] [program2] : child process terminated
[62987.641115] [program2] : The return signal is 9
[62990.493256] [program2] : Module_exit
```

Normal:



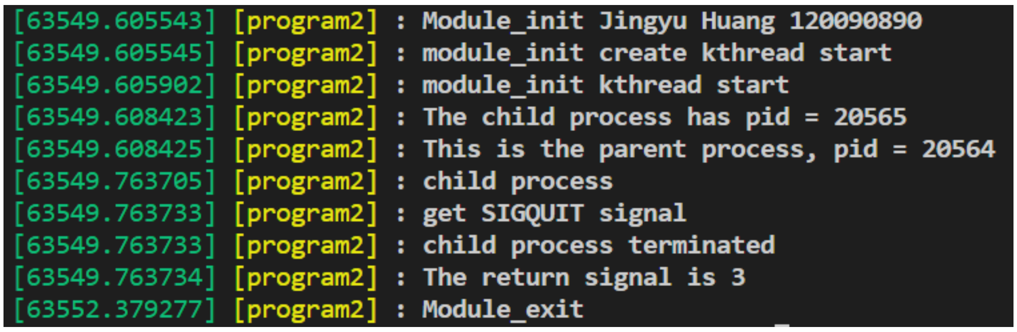
```
[65549.456727] [program2] : Module_init Jingyu Huang 120090890
[65549.456729] [program2] : module_init create kthread start
[65549.457538] [program2] : module_init kthread start
[65549.469207] [program2] : The child process has pid = 28968
[65549.469209] [program2] : This is the parent process, pid = 28966
[65549.474546] [program2] : child process
[65549.474548] [program2] : Normal termination with EXIT STATUS = 0
[65551.415364] [program2] : Module_exit
```

Pipe:



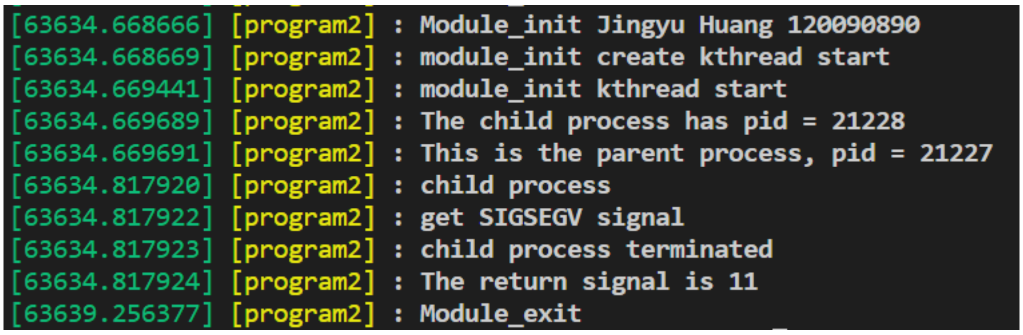
```
[63484.857616] [program2] : Module_init Jingyu Huang 120090890
[63484.857646] [program2] : module_init create kthread start
[63484.858156] [program2] : module_init kthread start
[63484.863307] [program2] : The child process has pid = 19931
[63484.863309] [program2] : This is the parent process, pid = 19930
[63484.865168] [program2] : child process
[63484.865170] [program2] : get SIGPIPE signal
[63484.865171] [program2] : child process terminated
[63484.865172] [program2] : The return signal is 13
[63487.042486] [program2] : Module_exit
```

Quit:



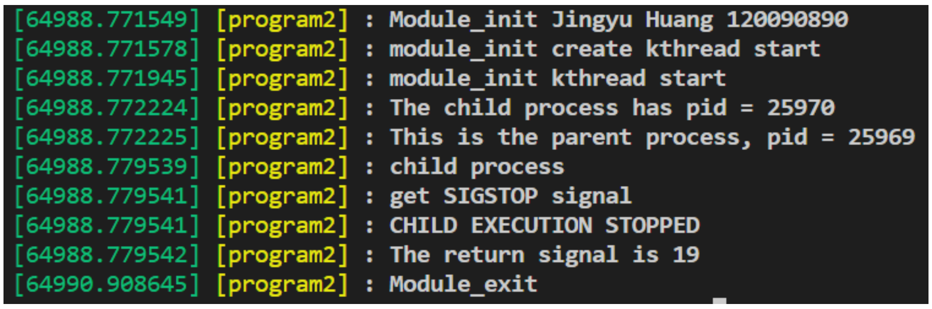
```
[63549.605543] [program2] : Module_init Jingyu Huang 120090890
[63549.605545] [program2] : module_init create kthread start
[63549.605902] [program2] : module_init kthread start
[63549.608423] [program2] : The child process has pid = 20565
[63549.608425] [program2] : This is the parent process, pid = 20564
[63549.763705] [program2] : child process
[63549.763733] [program2] : get SIGQUIT signal
[63549.763733] [program2] : child process terminated
[63549.763734] [program2] : The return signal is 3
[63552.379277] [program2] : Module_exit
```

Segment_fault:



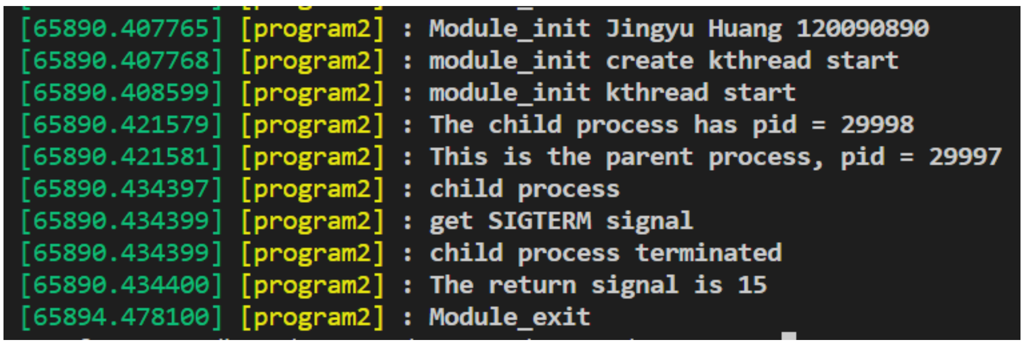
```
[63634.668666] [program2] : Module_init Jingyu Huang 120090890
[63634.668669] [program2] : module_init create kthread start
[63634.669441] [program2] : module_init kthread start
[63634.669689] [program2] : The child process has pid = 21228
[63634.669691] [program2] : This is the parent process, pid = 21227
[63634.817920] [program2] : child process
[63634.817922] [program2] : get SIGSEGV signal
[63634.817923] [program2] : child process terminated
[63634.817924] [program2] : The return signal is 11
[63639.256377] [program2] : Module_exit
```

Stop:



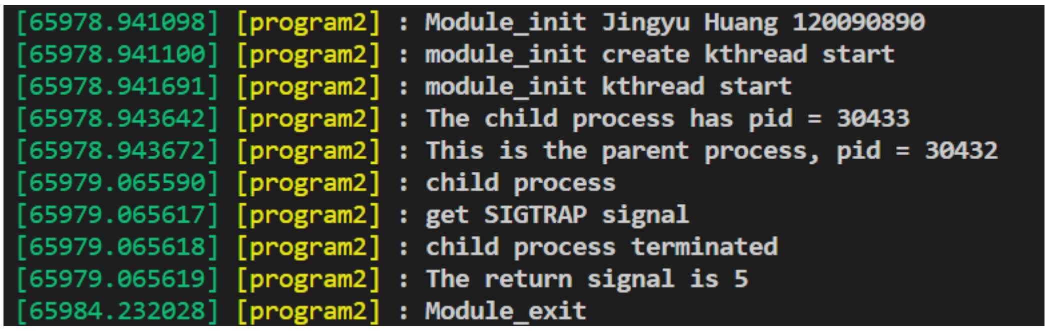
```
[64988.771549] [program2] : Module_init Jingyu Huang 120090890
[64988.771578] [program2] : module_init create kthread start
[64988.771945] [program2] : module_init kthread start
[64988.772224] [program2] : The child process has pid = 25970
[64988.772225] [program2] : This is the parent process, pid = 25969
[64988.779539] [program2] : child process
[64988.779541] [program2] : get SIGSTOP signal
[64988.779541] [program2] : CHILD EXECUTION STOPPED
[64988.779542] [program2] : The return signal is 19
[64990.908645] [program2] : Module_exit
```

Terminate:



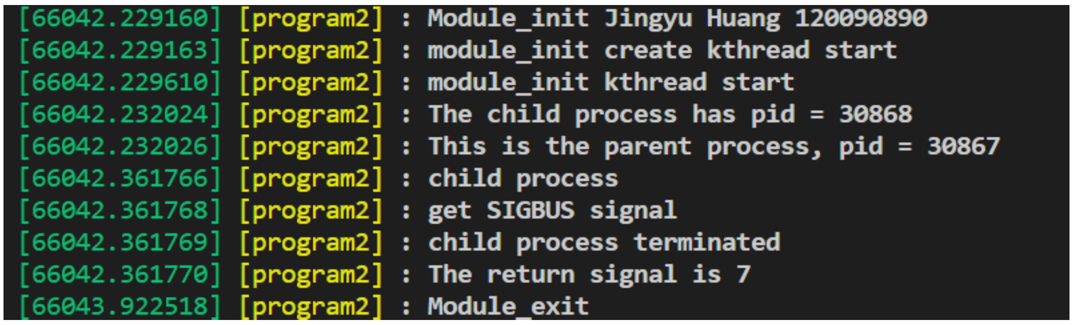
```
[65890.407765] [program2] : Module_init Jingyu Huang 120090890
[65890.407768] [program2] : module_init create kthread start
[65890.408599] [program2] : module_init kthread start
[65890.421579] [program2] : The child process has pid = 29998
[65890.421581] [program2] : This is the parent process, pid = 29997
[65890.434397] [program2] : child process
[65890.434399] [program2] : get SIGTERM signal
[65890.434399] [program2] : child process terminated
[65890.434400] [program2] : The return signal is 15
[65894.478100] [program2] : Module_exit
```

Trap:



```
[65978.941098] [program2] : Module_init Jingyu Huang 120090890
[65978.941100] [program2] : module_init create kthread start
[65978.941691] [program2] : module_init kthread start
[65978.943642] [program2] : The child process has pid = 30433
[65978.943672] [program2] : This is the parent process, pid = 30432
[65979.065590] [program2] : child process
[65979.065617] [program2] : get SIGTRAP signal
[65979.065618] [program2] : child process terminated
[65979.065619] [program2] : The return signal is 5
[65984.232028] [program2] : Module_exit
```

Test:



```
[66042.229160] [program2] : Module_init Jingyu Huang 120090890
[66042.229163] [program2] : module_init create kthread start
[66042.229610] [program2] : module_init kthread start
[66042.232024] [program2] : The child process has pid = 30868
[66042.232026] [program2] : This is the parent process, pid = 30867
[66042.361766] [program2] : child process
[66042.361768] [program2] : get SIGBUS signal
[66042.361769] [program2] : child process terminated
[66042.361770] [program2] : The return signal is 7
[66043.922518] [program2] : Module_exit
```

What I learn

From this task, I learn some more basic implementation of the user mode function like fork(), execve() and wait() using kernel_clone(), do_execve() and do_wait() and get to understand the module notion in an operating system. And I also learn more about the recompiling of the linux kernel which really take me a lot of effort to understand.