CSC3150 Operating System
Assignment1 Report

Yuhang Wang 王禹杭

120090246

1. Design

1.1 Task 1

1. Using fork() to fork a child process. If it fails to fork, it will exit and signal an error. And then child process and parent process will proceed concurrently.
2. In child process, use execve() to execute the test program. The parameter of this function is that you inputted in the command line.
3. In parent process, using waitpid() to get the signal.
   Here is the usage of waitpid():
   pid_t wait_child = waitpid(pid, &status, WUNTRACED);
   the option WUNTRACED traces the status of stopped children.
4. Using WIFEXITED, WEXITSTATUS, WIFSIGNALED, WTERMSIG, and WIFSTOPPED to receive and parse the signal and check the child process' termination status.
5. Identify the signal: using switch case expression to match the exit code of the signal to their corresponding name. Then print it.

1.2 Task 2

1. In linux system file, I have add lines of code In the corresponding file:

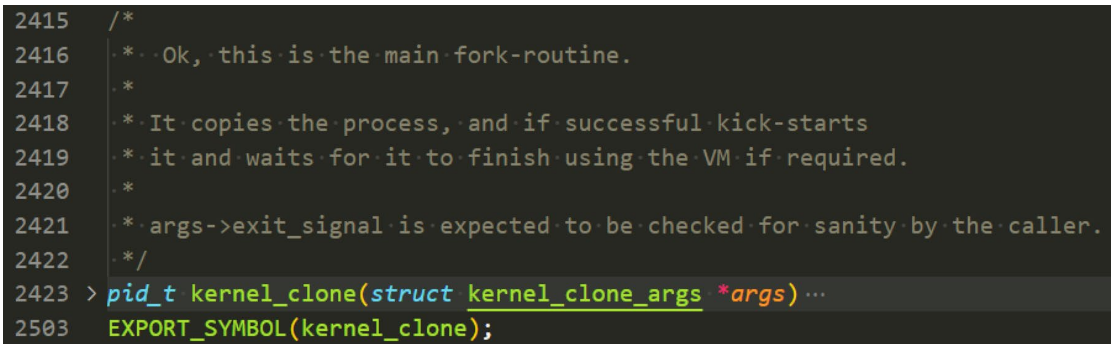
   EXPORT_SYMBOL(kernel_clone)

   EXPORT_SYMBOL(do_wait)

   EXPORT_SYMBOL(do_execve)

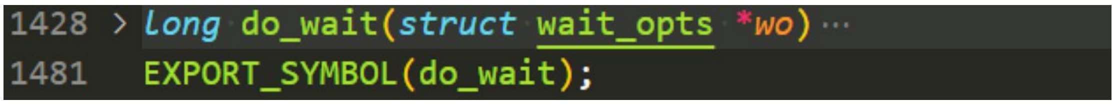
   EXPORT_SYMBOL(getname_kernel)

Linux-5.10.99/kernel/fork.c:



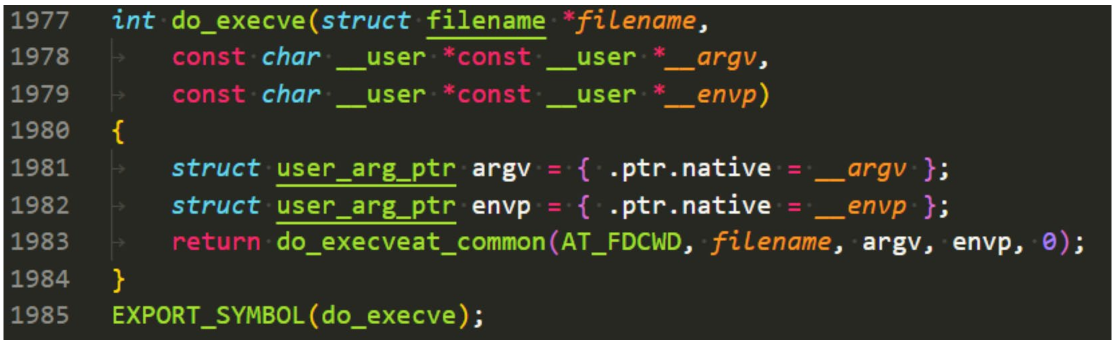
```c
/*
 *  Ok, this is the main fork-routine.
 *
 * It copies the process, and if successful kick-starts
 * it and waits for it to finish using the VM if required.
 *
 * args->exit_signal is expected to be checked for sanity by the caller.
 */
pid_t kernel_clone(struct kernel_clone_args *args) ···
EXPORT_SYMBOL(kernel_clone);
```

Linux-5.10.99/kernel/exit.c:



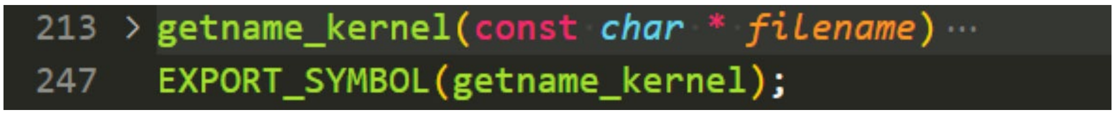
```c
long do_wait(struct wait_opts *wo) ···
EXPORT_SYMBOL(do_wait);
```

Linux-5.10.99/fs/exit.c:



```c
int do_execve(struct filename *filename,
	const char __user *const __user *__argv,
	const char __user *const __user *__envp)
{
	struct user_arg_ptr argv = { .ptr.native = __argv };
	struct user_arg_ptr envp = { .ptr.native = __envp };
	return do_execveat_common(AT_FDCWD, filename, argv, envp, 0);
}
EXPORT_SYMBOL(do_execve);
```

Linux-5.10.99/fs/namei.c:



```c
getname_kernel(const char * filename) ···
EXPORT_SYMBOL(getname_kernel);
```

2. I also extern the symbol in program2.c:

extern int do_execve(struct filename *filename,

const char __user *const __user *__argv,

const char __user *const __user *__envp);

extern struct filename *getname_kernel(const char *filename);

extern long do_wait(struct wait_opts *wo);

extern pid_t kernel_clone(struct kernel_clone_args *kargs);

3. I also constructed a type to better use the **do_wait()** function:

```
struct wait_opts {
        enum pid_type wo_type;
        int wo_flags;
        struct pid *wo_pid;
        struct waitid_info *wo_info;
        int wo_stat;
        struct rusage *wo_rusage;
        wait_queue_entry_t child_wait;
        int notask_error;
};
```

4. Create kernel thread: In kernel mode, using kthread_create() function to create a new thread to do my_fork() function.
5. In my_fork() function, using kernel_clone() function to fork a process linked to my_exec() function.
6. In my_exec(), using do_execve() to execute the test program. The path should be **"/tmp/test"**, where the test is an executable file and act as a child thread.

   And I set the argv[] and envp[] argument in do_execve() function, and it is as shown below:

   ```
   result = do_execve(getname_kernel(path), NULL, NULL);
   ```

7. In my_wait() : using do_wait() to wait the child process and get the exit signal code in wo.wo_stat. Then using switch case expression to identify the signal received and print the corresponding signal raised.

   And I set:

   ```
   wo.wo_flags = WEXITED | WUNTRACED
   ```

   Besides, I add a command:

   ```
   return_signal &= 0x7f;
   ```

   to the returned signal, which converts the signal greater than 128 to the signal from range 0-19.

2. Development Environment

2.1 Version of Linux Distribution:

Distributor ID:	Ubuntu
Description:	Ubuntu 16.04.7 LTS
Release:	16.04
Codename:	xenial

2.2 Version of kernel: 5.10.99

2.3 Version of gcc: 5.4.0 20160609

3. The steps to compile the kernel and execute my program

3.1 Compile the kernel:

1. Download source code from

   http://www.kernel.org

   mirro: https://mirror.tuna.tsinghua.edu.cn/kernel/v5.x/
2. Install Dependency and development tools:

   sudo apt-get install bc libncurses-dev gawk flex bison openssl libssl-dev dkms libelf-dev libudev-dev libpci-dev libiberty-dev autoconf llvm dwarves
3. 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
4. Copy config from /boot to /home/seed/work/KERNEL_FILE
5. Login root account and go to kernel source directory

   sudo su

   cd /home/seed/work /KERNEL_FILE
6. Clean previous setting and start configuration

make mrproper

make clean

make menuconfig

save the config and exit

7. kernel Image and modules(start from here when recompile the kernel)

make bzImage -j$(nproc)

make modules -j$(nproc)

8. Install kernel modules

make modules_install

9. Install kernel

make install

10. Reboot to load new kernel

reboot

(When rebooting, you should select the updated kernel)

3.2 Program1 Compile:

How to compile:

In the 'program1' directory, type 'make' command and enter.

How to clear:

In the 'program1' directory, type 'make clean' command and enter.

3.3 Program1 execution:

After changing into **Assignment_1_120090246/source/program1** directory, then type:

./program1 ./$TEST_CASE

where $TEST_CASE is the name of test program, for example:

./program1 ./abort

3.4 Program2 Complie

How to compile:

In the 'program1' directory, type 'make' command and enter.

How to clear:

In the 'program1' directory, type 'make clean' command and enter.

3.5 Program2 execution

In the 'program1' directory, first type “make” to run Makefile, then type:

sudo insmod program2.ko

sudo rmmod program2

dmesg

Then , you could see messages appearing. The messages are between the messages 'Module init' and 'module exit'. You could replace “dmesg” command by

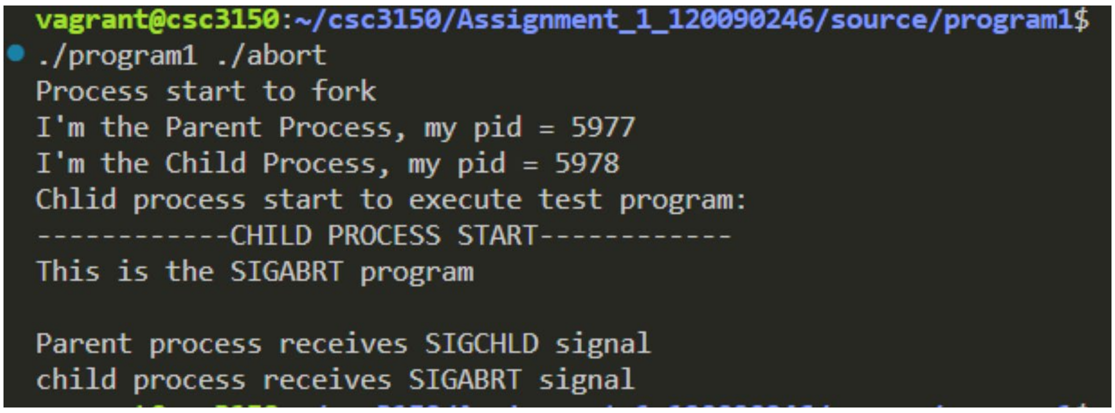
dmesg | tail -n 20

if you want to show last 20 lines of messages.

4. Output Demonstration

4.1 Task1

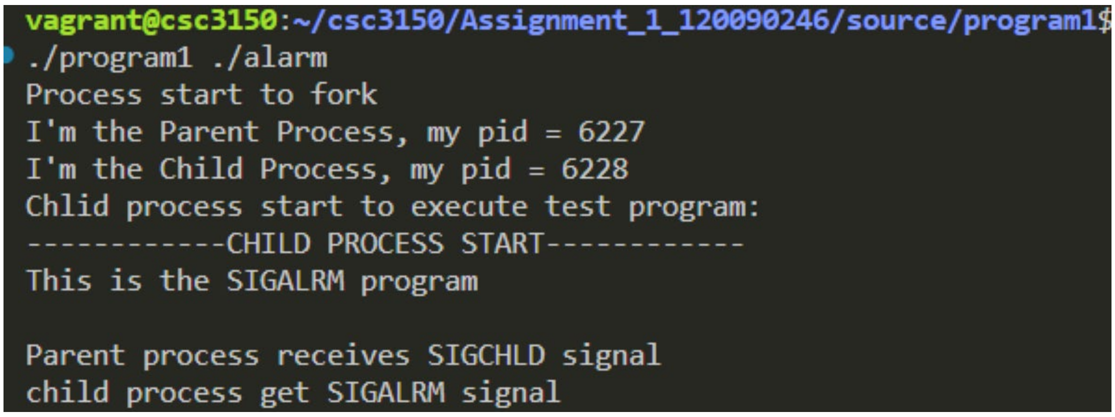
Abort



```
vagrant@csc3150:~/csc3150/Assignment_1_120090246/source/program1$
./program1 ./abort
Process start to fork
I'm the Parent Process, my pid = 5977
I'm the Child Process, my pid = 5978
Chlid process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGABRT program

Parent process receives SIGCHLD signal
child process receives SIGABRT signal
```

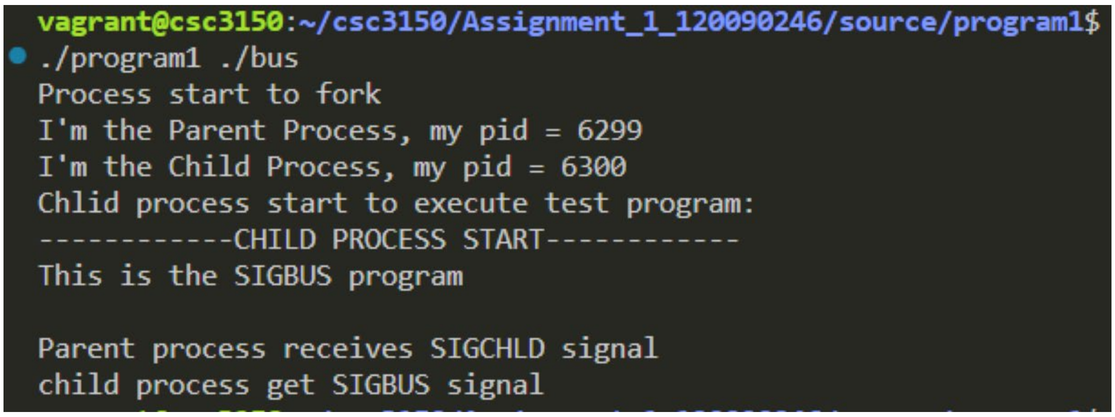
Alarm



```
vagrant@csc3150:~/csc3150/Assignment_1_120090246/source/program1$
./program1 ./alarm
Process start to fork
I'm the Parent Process, my pid = 6227
I'm the Child Process, my pid = 6228
Chlid process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGALRM program

Parent process receives SIGCHLD signal
child process get SIGALRM signal
```

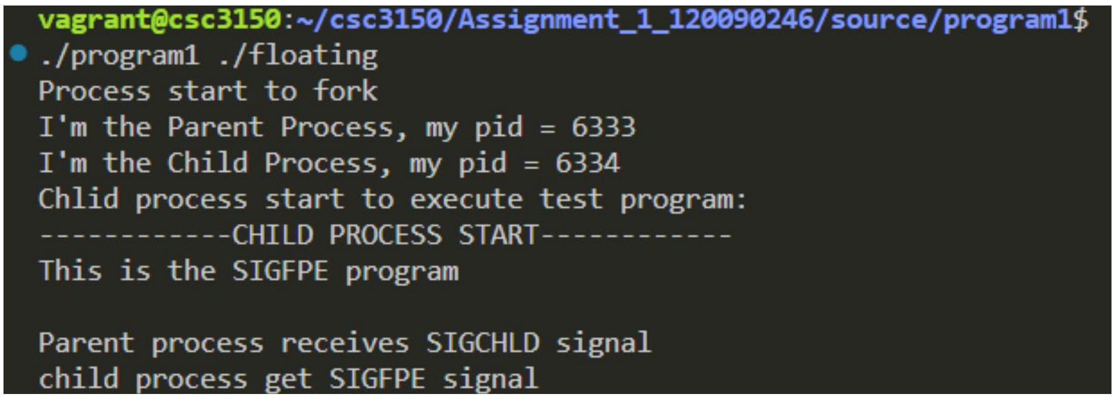
Bus



```
vagrant@csc3150:~/csc3150/Assignment_1_120090246/source/program1$
./program1 ./bus
Process start to fork
I'm the Parent Process, my pid = 6299
I'm the Child Process, my pid = 6300
Chlid process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGBUS program

Parent process receives SIGCHLD signal
child process get SIGBUS signal
```

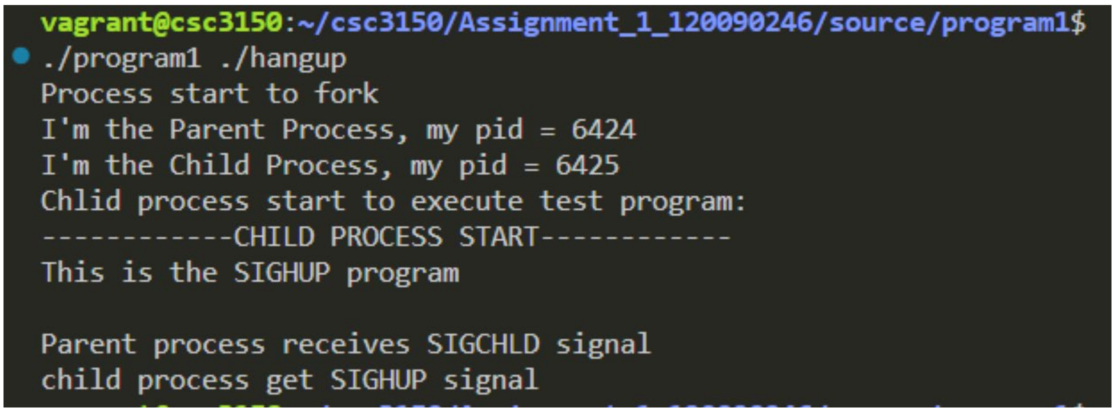
Floating



```
vagrant@csc3150:~/csc3150/Assignment_1_120090246/source/program1$
./program1 ./floating
Process start to fork
I'm the Parent Process, my pid = 6333
I'm the Child Process, my pid = 6334
Chlid process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGFPE program

Parent process receives SIGCHLD signal
child process get SIGFPE signal
```

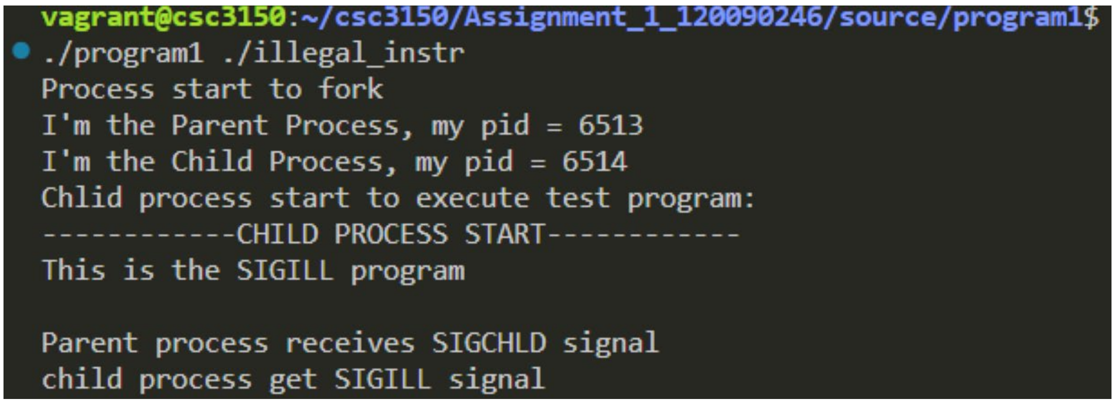
Hangup



```
vagrant@csc3150:~/csc3150/Assignment_1_120090246/source/program1$
./program1 ./hangup
Process start to fork
I'm the Parent Process, my pid = 6424
I'm the Child Process, my pid = 6425
Chlid process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGHUP program

Parent process receives SIGCHLD signal
child process get SIGHUP signal
```

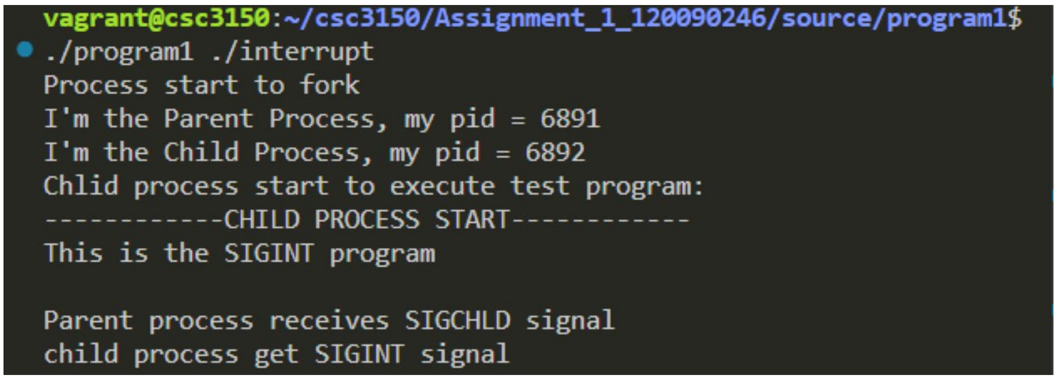
Illegal_instr



```
vagrant@csc3150:~/csc3150/Assignment_1_120090246/source/program1$
./program1 ./illegal_instr
Process start to fork
I'm the Parent Process, my pid = 6513
I'm the Child Process, my pid = 6514
Chlid process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGILL program

Parent process receives SIGCHLD signal
child process get SIGILL signal
```

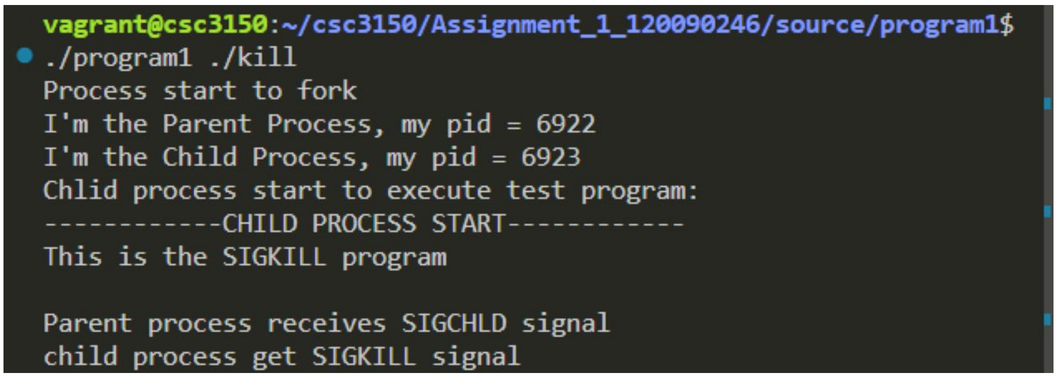
Interrupt



```
vagrant@csc3150:~/csc3150/Assignment_1_120090246/source/program1$
./program1 ./interrupt
Process start to fork
I'm the Parent Process, my pid = 6891
I'm the Child Process, my pid = 6892
Chlid process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGINT program

Parent process receives SIGCHLD signal
child process get SIGINT signal
```

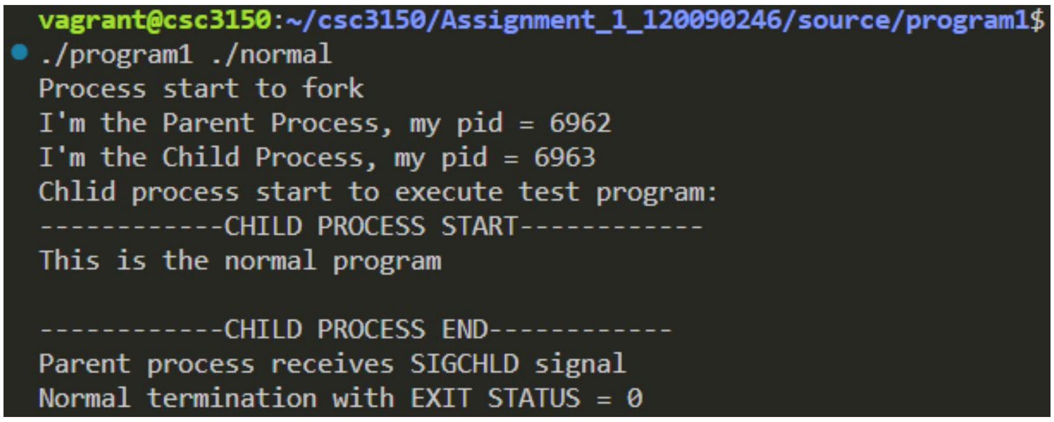
Kill



```
vagrant@csc3150:~/csc3150/Assignment_1_120090246/source/program1$
./program1 ./kill
Process start to fork
I'm the Parent Process, my pid = 6922
I'm the Child Process, my pid = 6923
Chlid process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGKILL program

Parent process receives SIGCHLD signal
child process get SIGKILL signal
```

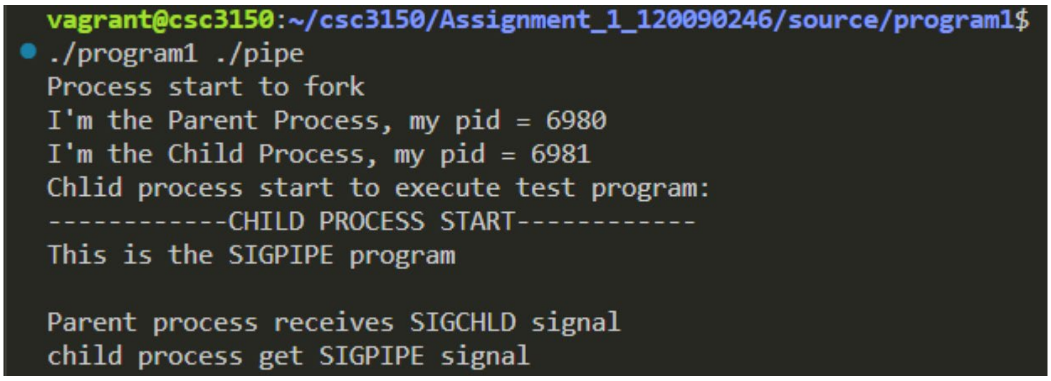
Normal



```
vagrant@csc3150:~/csc3150/Assignment_1_120090246/source/program1$
./program1 ./normal
Process start to fork
I'm the Parent Process, my pid = 6962
I'm the Child Process, my pid = 6963
Chlid 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
```

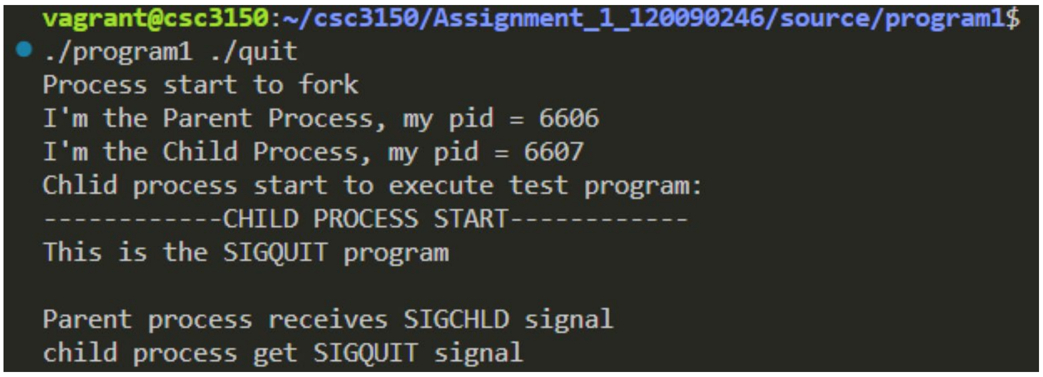
Pipe



```
vagrant@csc3150:~/csc3150/Assignment_1_120090246/source/program1$
./program1 ./pipe
Process start to fork
I'm the Parent Process, my pid = 6980
I'm the Child Process, my pid = 6981
Chlid process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGPIPE program

Parent process receives SIGCHLD signal
child process get SIGPIPE signal
```

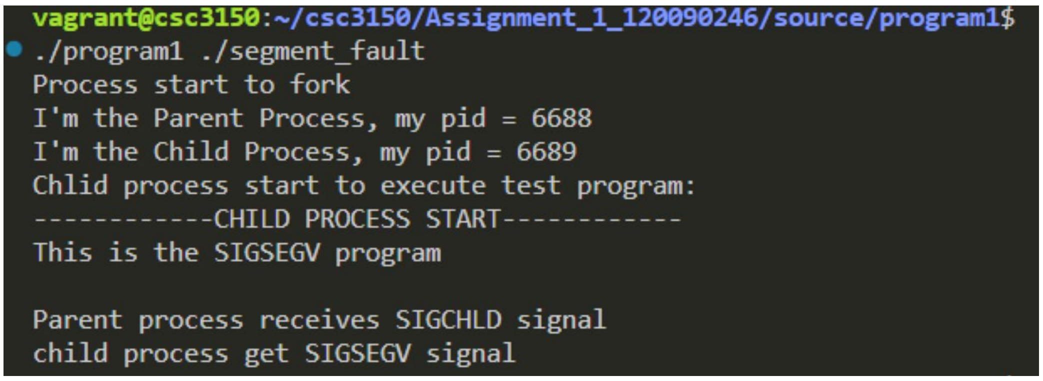
Quit



```
vagrant@csc3150:~/csc3150/Assignment_1_120090246/source/program1$
./program1 ./quit
Process start to fork
I'm the Parent Process, my pid = 6606
I'm the Child Process, my pid = 6607
Chlid process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

Parent process receives SIGCHLD signal
child process get SIGQUIT signal
```

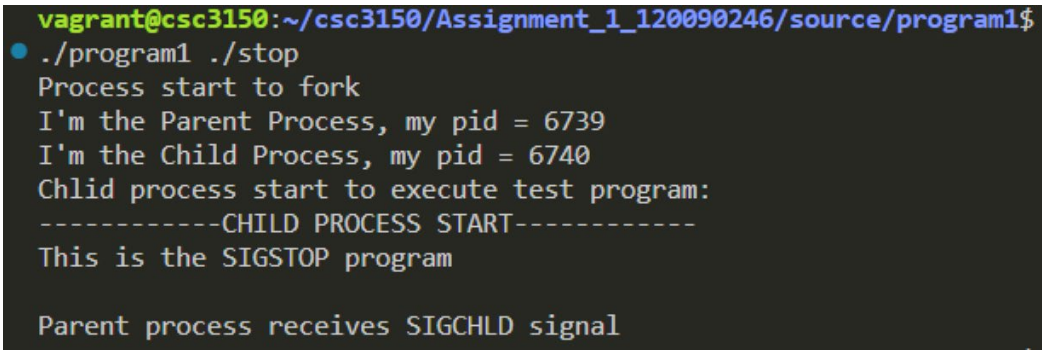
Segment_fault



```
vagrant@csc3150:~/csc3150/Assignment_1_120090246/source/program1$
./program1 ./segment_fault
Process start to fork
I'm the Parent Process, my pid = 6688
I'm the Child Process, my pid = 6689
Chlid process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

Parent process receives SIGCHLD signal
child process get SIGSEGV signal
```

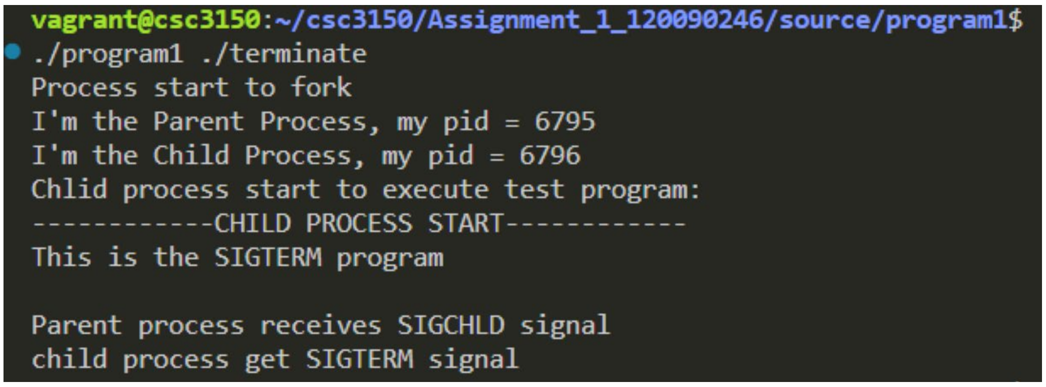
Stop



```
vagrant@csc3150:~/csc3150/Assignment_1_120090246/source/program1$
./program1 ./stop
Process start to fork
I'm the Parent Process, my pid = 6739
I'm the Child Process, my pid = 6740
Chlid process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

Parent process receives SIGCHLD signal
```

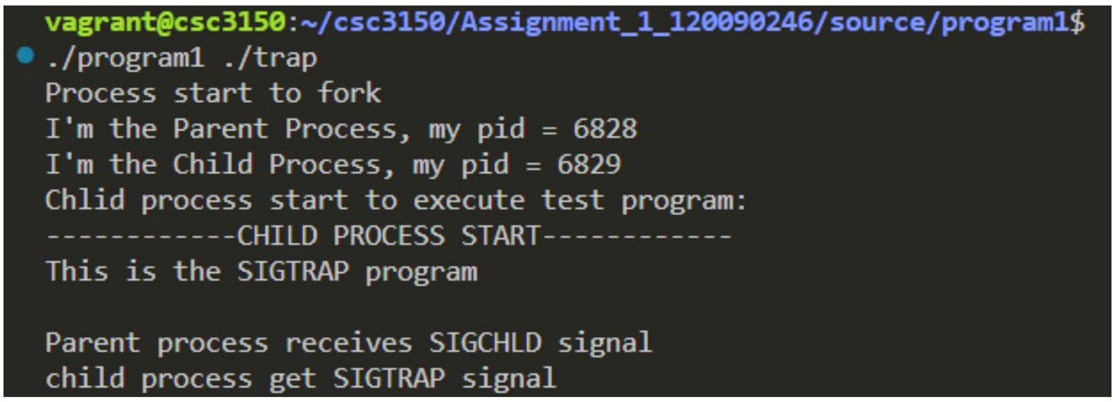
Terminate



```
vagrant@csc3150:~/csc3150/Assignment_1_120090246/source/program1$
./program1 ./terminate
Process start to fork
I'm the Parent Process, my pid = 6795
I'm the Child Process, my pid = 6796
Chlid process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTERM program

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

Trap

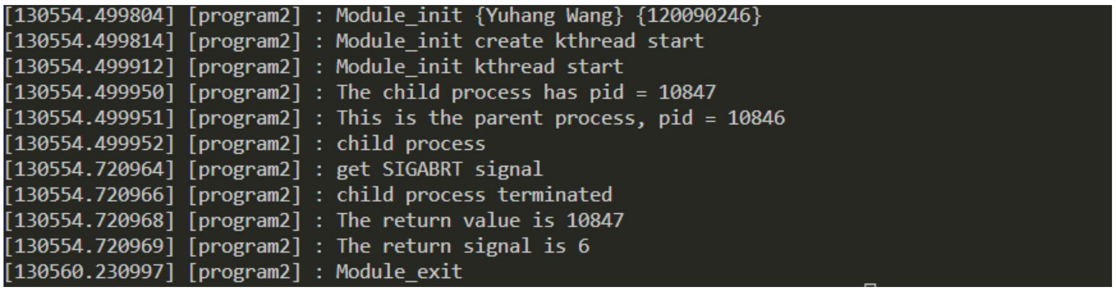


```
vagrant@csc3150:~/csc3150/Assignment_1_120090246/source/program1$
./program1 ./trap
Process start to fork
I'm the Parent Process, my pid = 6828
I'm the Child Process, my pid = 6829
Chlid process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTRAP program

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

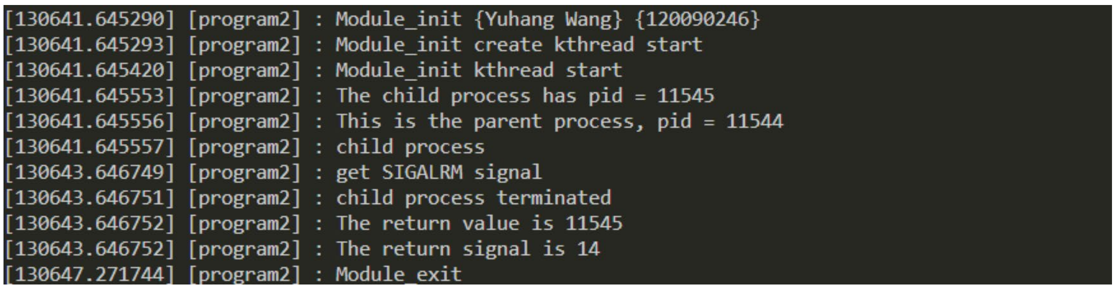
4.2 task2

Abort



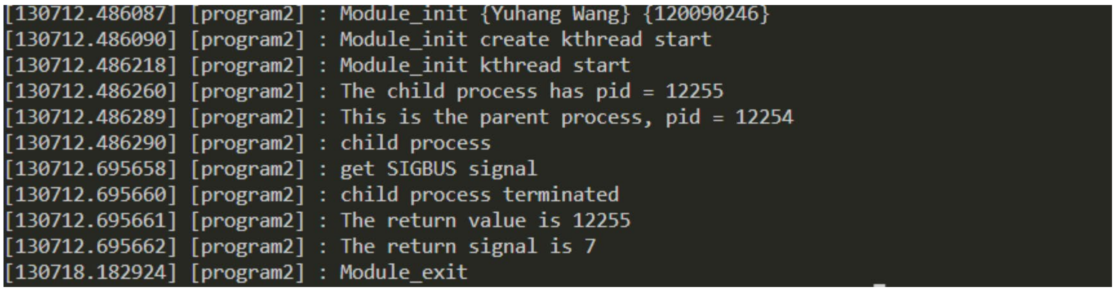
```
[130554.499804] [program2] : Module_init {Yuhang Wang} {120090246}
[130554.499814] [program2] : Module_init create kthread start
[130554.499912] [program2] : Module_init kthread start
[130554.499950] [program2] : The child process has pid = 10847
[130554.499951] [program2] : This is the parent process, pid = 10846
[130554.499952] [program2] : child process
[130554.720964] [program2] : get SIGABRT signal
[130554.720966] [program2] : child process terminated
[130554.720968] [program2] : The return value is 10847
[130554.720969] [program2] : The return signal is 6
[130560.230997] [program2] : Module_exit
```

Alarm



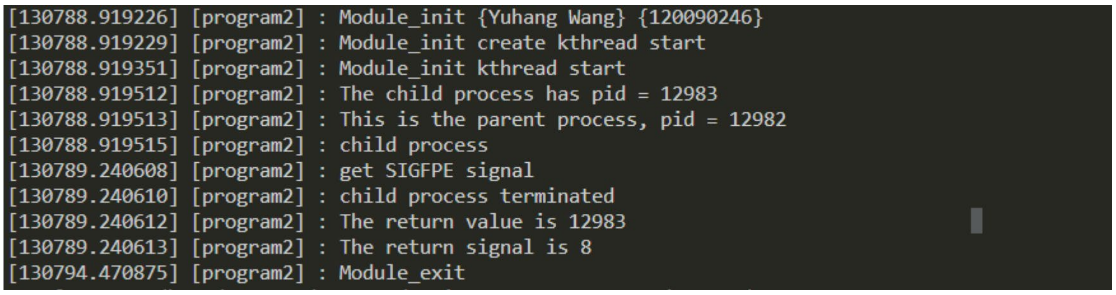
```
[130641.645290] [program2] : Module_init {Yuhang Wang} {120090246}
[130641.645293] [program2] : Module_init create kthread start
[130641.645420] [program2] : Module_init kthread start
[130641.645553] [program2] : The child process has pid = 11545
[130641.645556] [program2] : This is the parent process, pid = 11544
[130641.645557] [program2] : child process
[130643.646749] [program2] : get SIGALRM signal
[130643.646751] [program2] : child process terminated
[130643.646752] [program2] : The return value is 11545
[130643.646752] [program2] : The return signal is 14
[130647.271744] [program2] : Module_exit
```

Bus

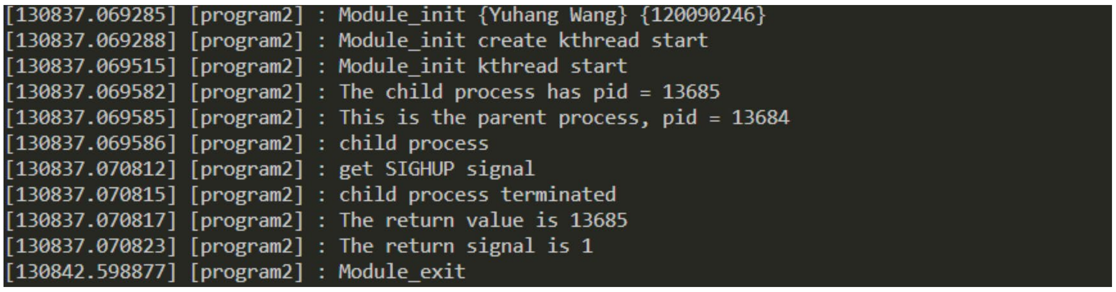


```
[130712.486087] [program2] : Module_init {Yuhang Wang} {120090246}
[130712.486090] [program2] : Module_init create kthread start
[130712.486218] [program2] : Module_init kthread start
[130712.486260] [program2] : The child process has pid = 12255
[130712.486289] [program2] : This is the parent process, pid = 12254
[130712.486290] [program2] : child process
[130712.695658] [program2] : get SIGBUS signal
[130712.695660] [program2] : child process terminated
[130712.695661] [program2] : The return value is 12255
[130712.695662] [program2] : The return signal is 7
[130718.182924] [program2] : Module_exit
```

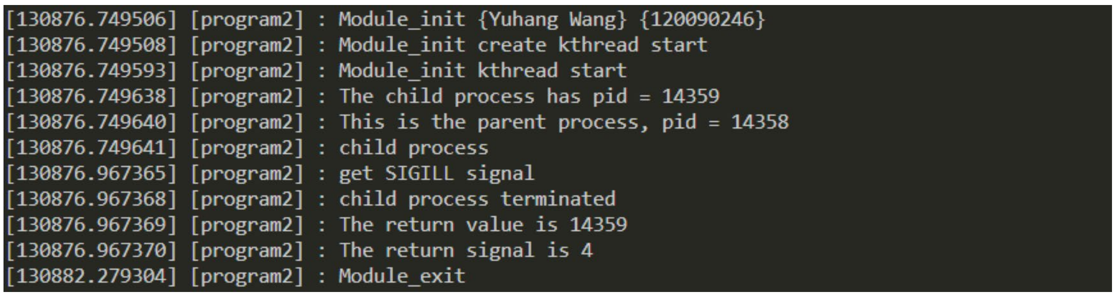
Floating


```
[130788.919226] [program2] : Module_init {Yuhang Wang} {120090246}
[130788.919229] [program2] : Module_init create kthread start
[130788.919351] [program2] : Module_init kthread start
[130788.919512] [program2] : The child process has pid = 12983
[130788.919513] [program2] : This is the parent process, pid = 12982
[130788.919515] [program2] : child process
[130789.240608] [program2] : get SIGFPE signal
[130789.240610] [program2] : child process terminated
[130789.240612] [program2] : The return value is 12983
[130789.240613] [program2] : The return signal is 8
[130794.470875] [program2] : Module_exit
```

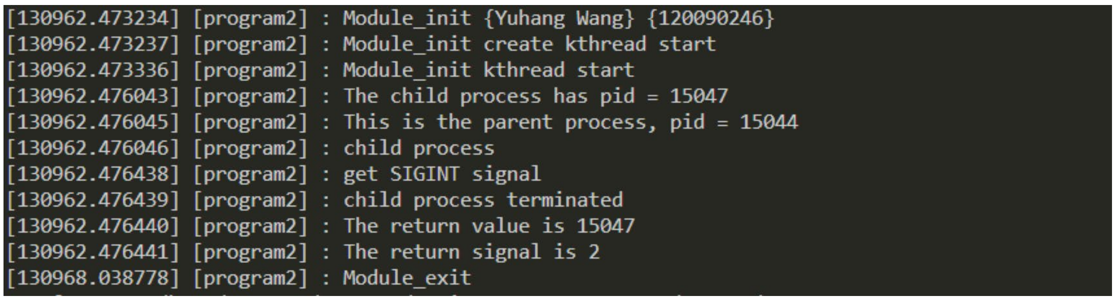
Hangup


```
[130837.069285] [program2] : Module_init {Yuhang Wang} {120090246}
[130837.069288] [program2] : Module_init create kthread start
[130837.069515] [program2] : Module_init kthread start
[130837.069582] [program2] : The child process has pid = 13685
[130837.069585] [program2] : This is the parent process, pid = 13684
[130837.069586] [program2] : child process
[130837.070812] [program2] : get SIGHUP signal
[130837.070815] [program2] : child process terminated
[130837.070817] [program2] : The return value is 13685
[130837.070823] [program2] : The return signal is 1
[130842.598877] [program2] : Module_exit
```

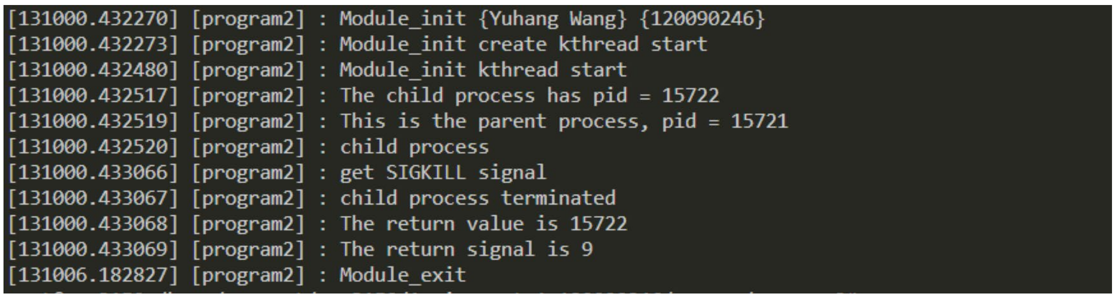
Illegal_instr


```
[130876.749506] [program2] : Module_init {Yuhang Wang} {120090246}
[130876.749508] [program2] : Module_init create kthread start
[130876.749593] [program2] : Module_init kthread start
[130876.749638] [program2] : The child process has pid = 14359
[130876.749640] [program2] : This is the parent process, pid = 14358
[130876.749641] [program2] : child process
[130876.967365] [program2] : get SIGILL signal
[130876.967368] [program2] : child process terminated
[130876.967369] [program2] : The return value is 14359
[130876.967370] [program2] : The return signal is 4
[130882.279304] [program2] : Module_exit
```

Interrupt

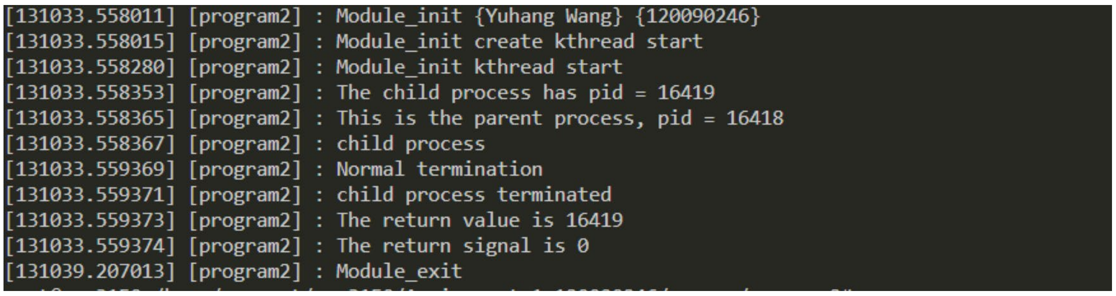

```
[130962.473234] [program2] : Module_init {Yuhang Wang} {120090246}
[130962.473237] [program2] : Module_init create kthread start
[130962.473336] [program2] : Module_init kthread start
[130962.476043] [program2] : The child process has pid = 15047
[130962.476045] [program2] : This is the parent process, pid = 15044
[130962.476046] [program2] : child process
[130962.476438] [program2] : get SIGINT signal
[130962.476439] [program2] : child process terminated
[130962.476440] [program2] : The return value is 15047
[130962.476441] [program2] : The return signal is 2
[130968.038778] [program2] : Module_exit
```

Kill



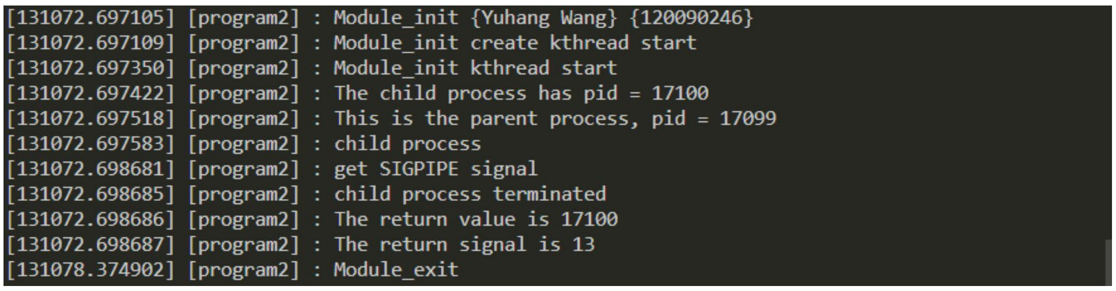
```
[131000.432270] [program2] : Module_init {Yuhang Wang} {120090246}
[131000.432273] [program2] : Module_init create kthread start
[131000.432480] [program2] : Module_init kthread start
[131000.432517] [program2] : The child process has pid = 15722
[131000.432519] [program2] : This is the parent process, pid = 15721
[131000.432520] [program2] : child process
[131000.433066] [program2] : get SIGKILL signal
[131000.433067] [program2] : child process terminated
[131000.433068] [program2] : The return value is 15722
[131000.433069] [program2] : The return signal is 9
[131006.182827] [program2] : Module_exit
```

Normal



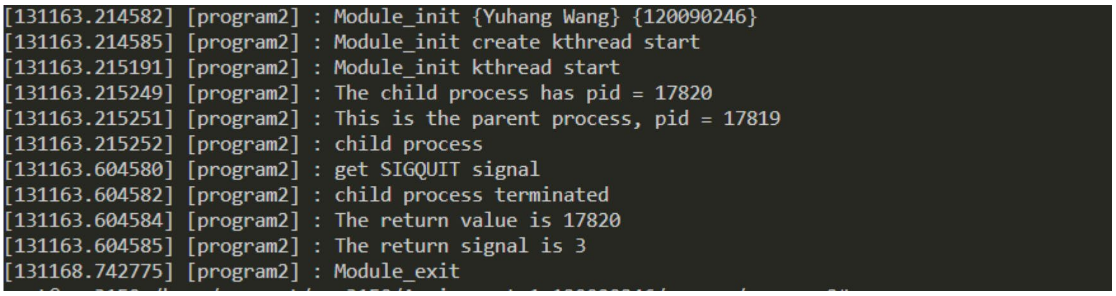
```
[131033.558011] [program2] : Module_init {Yuhang Wang} {120090246}
[131033.558015] [program2] : Module_init create kthread start
[131033.558280] [program2] : Module_init kthread start
[131033.558353] [program2] : The child process has pid = 16419
[131033.558365] [program2] : This is the parent process, pid = 16418
[131033.558367] [program2] : child process
[131033.559369] [program2] : Normal termination
[131033.559371] [program2] : child process terminated
[131033.559373] [program2] : The return value is 16419
[131033.559374] [program2] : The return signal is 0
[131039.207013] [program2] : Module_exit
```

Pipe



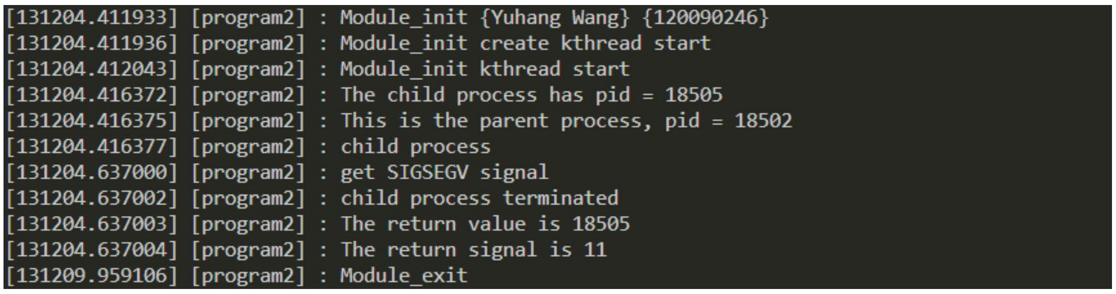
```
[131072.697105] [program2] : Module_init {Yuhang Wang} {120090246}
[131072.697109] [program2] : Module_init create kthread start
[131072.697350] [program2] : Module_init kthread start
[131072.697422] [program2] : The child process has pid = 17100
[131072.697518] [program2] : This is the parent process, pid = 17099
[131072.697583] [program2] : child process
[131072.698681] [program2] : get SIGPIPE signal
[131072.698685] [program2] : child process terminated
[131072.698686] [program2] : The return value is 17100
[131072.698687] [program2] : The return signal is 13
[131078.374902] [program2] : Module_exit
```

Quit



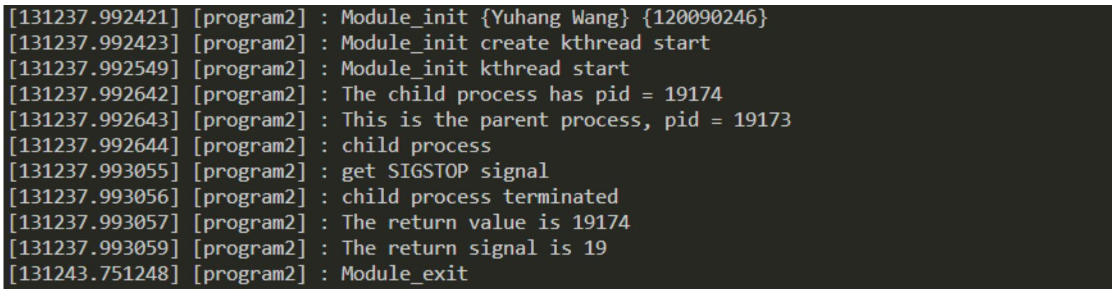
```
[131163.214582] [program2] : Module_init {Yuhang Wang} {120090246}
[131163.214585] [program2] : Module_init create kthread start
[131163.215191] [program2] : Module_init kthread start
[131163.215249] [program2] : The child process has pid = 17820
[131163.215251] [program2] : This is the parent process, pid = 17819
[131163.215252] [program2] : child process
[131163.604580] [program2] : get SIGQUIT signal
[131163.604582] [program2] : child process terminated
[131163.604584] [program2] : The return value is 17820
[131163.604585] [program2] : The return signal is 3
[131168.742775] [program2] : Module_exit
```

Segment_fault



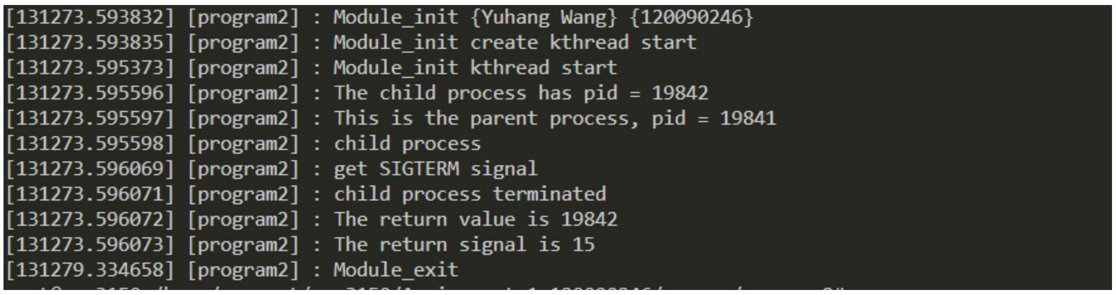
```
[131204.411933] [program2] : Module_init {Yuhang Wang} {120090246}
[131204.411936] [program2] : Module_init create kthread start
[131204.412043] [program2] : Module_init kthread start
[131204.416372] [program2] : The child process has pid = 18505
[131204.416375] [program2] : This is the parent process, pid = 18502
[131204.416377] [program2] : child process
[131204.637000] [program2] : get SIGSEGV signal
[131204.637002] [program2] : child process terminated
[131204.637003] [program2] : The return value is 18505
[131204.637004] [program2] : The return signal is 11
[131209.959106] [program2] : Module_exit
```

Stop



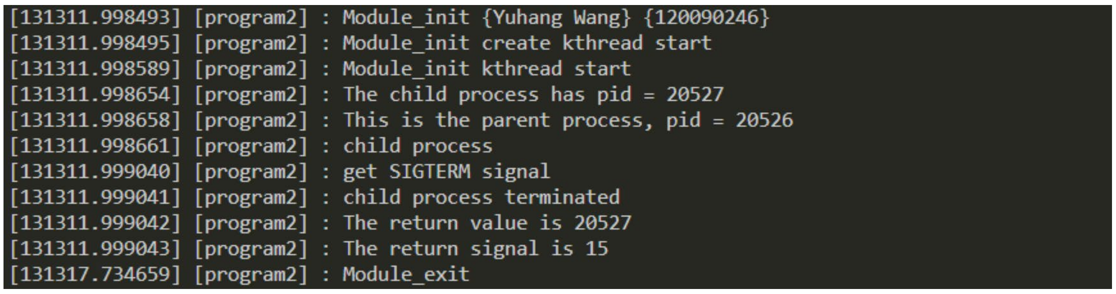
```
[131237.992421] [program2] : Module_init {Yuhang Wang} {120090246}
[131237.992423] [program2] : Module_init create kthread start
[131237.992549] [program2] : Module_init kthread start
[131237.992642] [program2] : The child process has pid = 19174
[131237.992643] [program2] : This is the parent process, pid = 19173
[131237.992644] [program2] : child process
[131237.993055] [program2] : get SIGSTOP signal
[131237.993056] [program2] : child process terminated
[131237.993057] [program2] : The return value is 19174
[131237.993059] [program2] : The return signal is 19
[131243.751248] [program2] : Module_exit
```

Terminate



```
[131273.593832] [program2] : Module_init {Yuhang Wang} {120090246}
[131273.593835] [program2] : Module_init create kthread start
[131273.595373] [program2] : Module_init kthread start
[131273.595596] [program2] : The child process has pid = 19842
[131273.595597] [program2] : This is the parent process, pid = 19841
[131273.595598] [program2] : child process
[131273.596069] [program2] : get SIGTERM signal
[131273.596071] [program2] : child process terminated
[131273.596072] [program2] : The return value is 19842
[131273.596073] [program2] : The return signal is 15
[131279.334658] [program2] : Module_exit
```

Trap



```
[131311.998493] [program2] : Module_init {Yuhang Wang} {120090246}
[131311.998495] [program2] : Module_init create kthread start
[131311.998589] [program2] : Module_init kthread start
[131311.998654] [program2] : The child process has pid = 20527
[131311.998658] [program2] : This is the parent process, pid = 20526
[131311.998661] [program2] : child process
[131311.999040] [program2] : get SIGTERM signal
[131311.999041] [program2] : child process terminated
[131311.999042] [program2] : The return value is 20527
[131311.999043] [program2] : The return signal is 15
[131317.734659] [program2] : Module_exit
```

5. What I Learned

5.1 In user mode, I learned to

fork a children process,

use children process to execute certain file or program,

wait until child process terminates,

receive signal raised from the child process,

process the signal, and finally print the signal.

5.2 In kernel mode, I learned to

initialize the kernel module and create kernel thread,

fork a process, execute certain file or program in child module,

wait the children process to terminate, how to execute file in kernel mode,

generate kernel object, insert kernel object to kernel module,

remove kernel object, and open kernel log to look into the process.

5.3 I also learned how to compile the kernel, how to modify the source code in the kernel source file, how to recompile the kernel, and how to write Makefile code, and the usage of clang-format.