CSC3150 Operating Systems

Assignment Report #1

Name: 徐驰
Student ID: 120090595

The Chinese University of Hong Kong, Shenzhen

1. Design

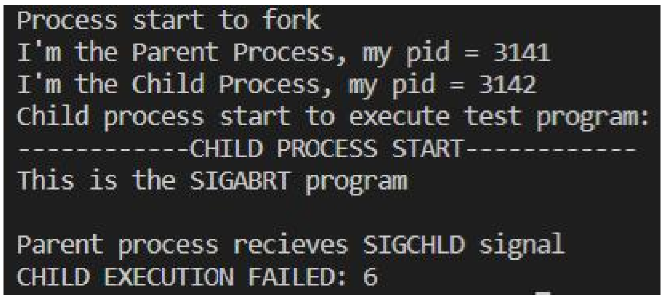
In task 1, I use the fork function to create a new process to execute the test program. After fork, the first thing is to determine the return value of the function. If return value is -1, then the creation of child process is unsuccessful and the program will exit. If return value is 0, the newly created process will firstly print out its pid by getpid function, an then it will execute the test program by execve function. If return value is a positive number, the parent process will print out its pid by getpid function, and then wait for child process until the child process finish execution by waitpid function. Finally printing out the termination information of child process. If the child process terminates normally, the normal termination and exit status will be printed out. If not, the information of how did the child process terminates and what signal was raised in child process will be printed out.

In task 2, I first declare the function I will use ahead, and export them in kernel source code. Including do_wait, kernel_clone, do_execve and getname_kernel. After that I recompile the kernel and intall it. And I define the structure wait_opts, which will be used in do_wait. Then I design my_wait, my_exec and my_fork. In my_wait, I first declare an int a to receive the return value of do_wait. Then I set the members of structure wait_opts wo. And according to wo.wo_stat, I can decide the return signal of child process and print the signal out. A little problem here is that some return signal is the sum of itself and 128, so I use the wo.wo_stat & 0x7F operation to get the correct return signal. Finally I use put_pid function to decrease the count and free memory. In my_exec, I use getname_kernel function to get one of the parameter filename of do_execve function. I set the other two parameters of do_execve function as NULL. Then I use the do_execve function to execute test program. In my_fork, I use kernel_clone to fork a new process to execute test program. Then I use my_wait to wait for child process' termination status and receive return signal. Finally I print out relevant message. In program2_init, I use kthread_create to create a new process.

2. Setup of the Development Environment

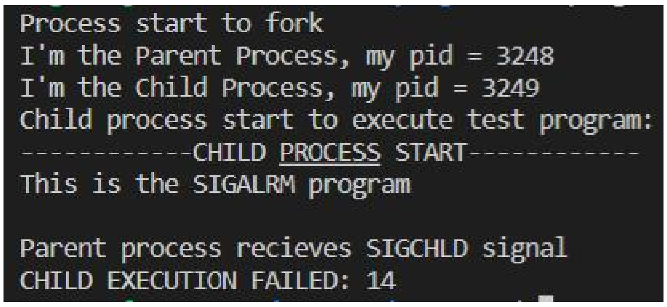
I follow steps in tutorial 1 to set up my virtual machine and remote SSH plugin in VS Code. Firstly I install VM and vagrant. Secondly I use commands from tutorial 1 to initial my VM. Thus I can connect to VM through VS Code. To compile kernel, I first use "wget" command to download kernel source code from tsinghua's mirror site. After decompressing it, I copy the config file to the path of kernel file. After that, I enter commands from tutorial 2 and reboot the VM. Finally I use "uname -r" to exam the version of kernel. Luckily, everything goes right.

3. Result



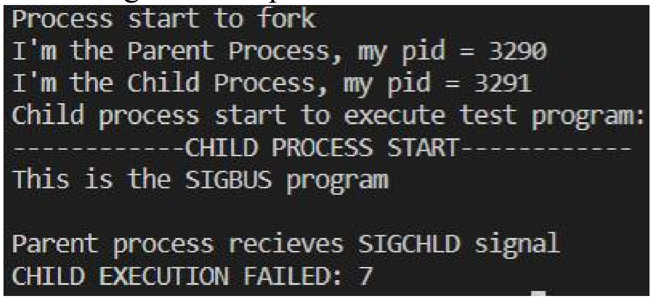
```
Process start to fork
I'm the Parent Process, my pid = 3141
I'm the Child Process, my pid = 3142
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGABRT program

Parent process recieves SIGCHLD signal
CHILD EXECUTION FAILED: 6
```

Figure 1. Output of Abort in Task 1



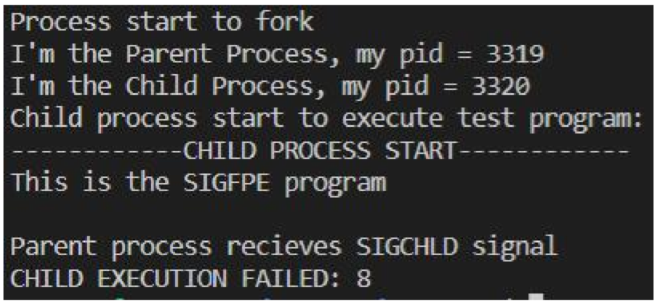
```
Process start to fork
I'm the Parent Process, my pid = 3248
I'm the Child Process, my pid = 3249
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGALRM program

Parent process recieves SIGCHLD signal
CHILD EXECUTION FAILED: 14
```

Figure 2. Output of Alarm in Task 1



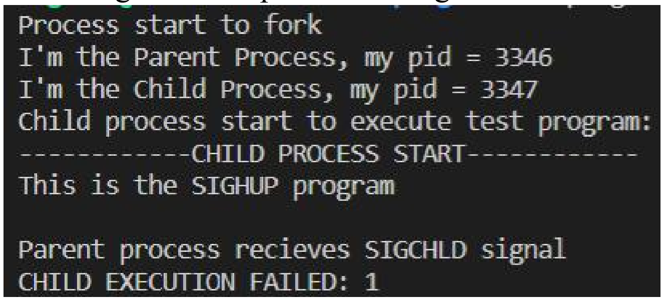
```
Process start to fork
I'm the Parent Process, my pid = 3290
I'm the Child Process, my pid = 3291
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGBUS program

Parent process recieves SIGCHLD signal
CHILD EXECUTION FAILED: 7
```

Figure 3. Output of Bus in Task 1



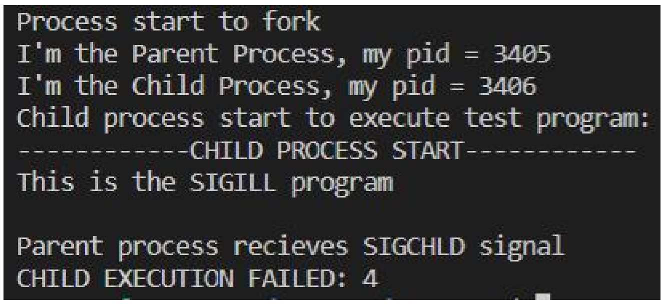
```
Process start to fork
I'm the Parent Process, my pid = 3319
I'm the Child Process, my pid = 3320
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGFPE program

Parent process recieves SIGCHLD signal
CHILD EXECUTION FAILED: 8
```

Figure 4. Output of Floating in Task 1



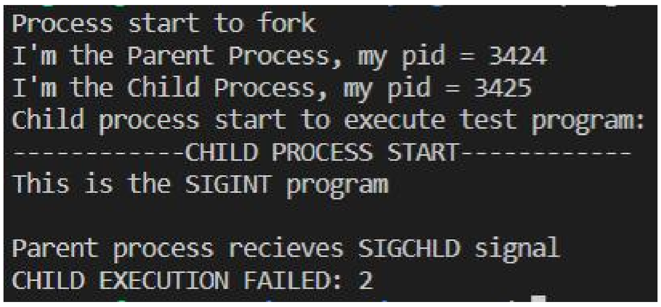
```
Process start to fork
I'm the Parent Process, my pid = 3346
I'm the Child Process, my pid = 3347
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGHUP program

Parent process recieves SIGCHLD signal
CHILD EXECUTION FAILED: 1
```

Figure 5. Output of Hangup in Task 1



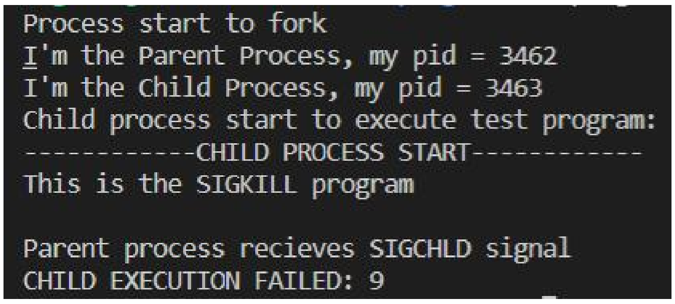
```
Process start to fork
I'm the Parent Process, my pid = 3405
I'm the Child Process, my pid = 3406
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGILL program

Parent process recieves SIGCHLD signal
CHILD EXECUTION FAILED: 4
```

Figure 6. Output of Illegal_instr in Task 1



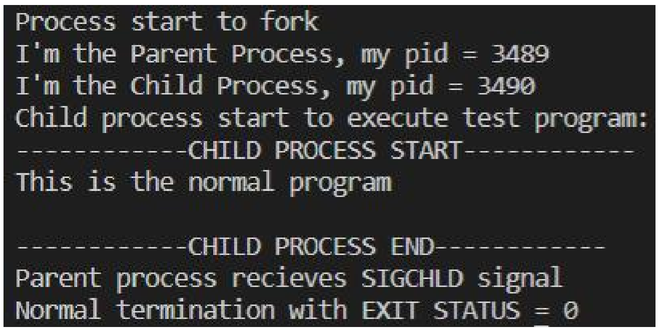
```
Process start to fork
I'm the Parent Process, my pid = 3424
I'm the Child Process, my pid = 3425
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGINT program

Parent process recieves SIGCHLD signal
CHILD EXECUTION FAILED: 2
```

Figure 7. Output of Interrupt in Task 1



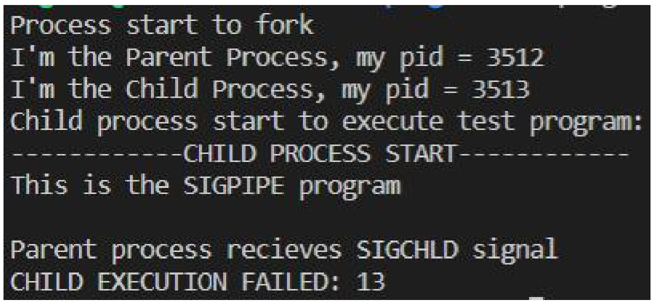
```
Process start to fork
I'm the Parent Process, my pid = 3462
I'm the Child Process, my pid = 3463
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGKILL program

Parent process recieves SIGCHLD signal
CHILD EXECUTION FAILED: 9
```

Figure 8. Output of Kill in Task 1



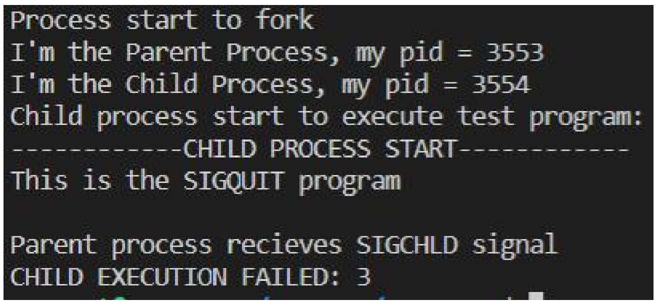
```
Process start to fork
I'm the Parent Process, my pid = 3489
I'm the Child Process, my pid = 3490
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the normal program

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

Figure 9. Output of Normal in Task 1



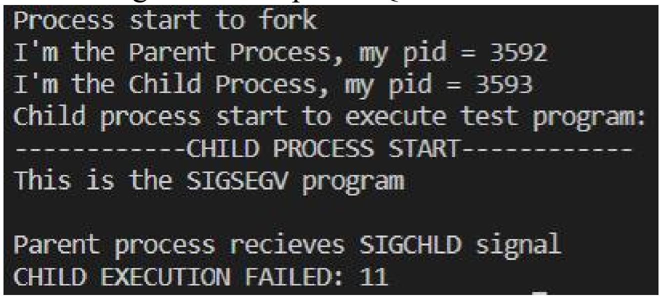
```
Process start to fork
I'm the Parent Process, my pid = 3512
I'm the Child Process, my pid = 3513
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGPIPE program

Parent process recieves SIGCHLD signal
CHILD EXECUTION FAILED: 13
```

Figure 10. Output of Pipe in Task 1



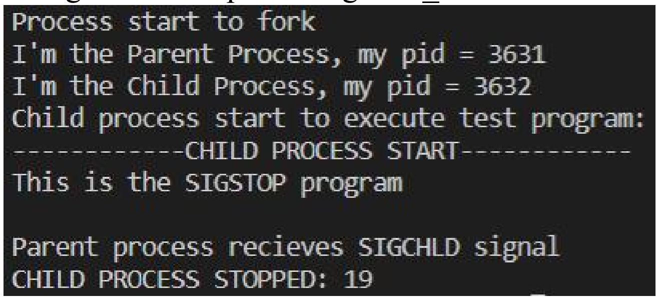
```
Process start to fork
I'm the Parent Process, my pid = 3553
I'm the Child Process, my pid = 3554
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

Parent process recieves SIGCHLD signal
CHILD EXECUTION FAILED: 3
```

Figure 11. Output of Quit in Task 1



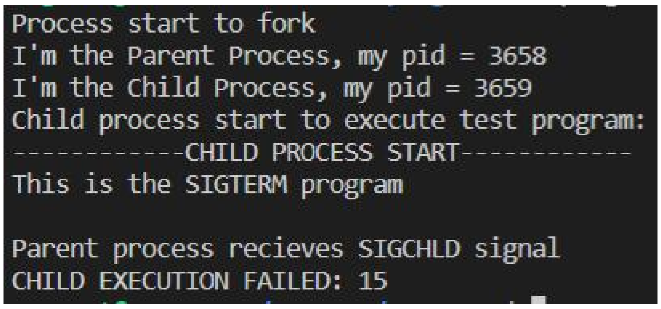
```
Process start to fork
I'm the Parent Process, my pid = 3592
I'm the Child Process, my pid = 3593
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

Parent process recieves SIGCHLD signal
CHILD EXECUTION FAILED: 11
```

Figure 12. Output of Segment_fault in Task 1



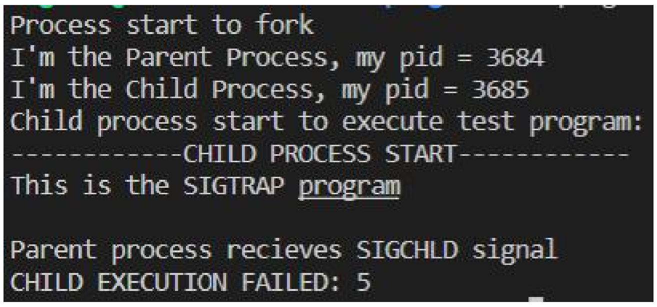
```
Process start to fork
I'm the Parent Process, my pid = 3631
I'm the Child Process, my pid = 3632
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

Parent process recieves SIGCHLD signal
CHILD PROCESS STOPPED: 19
```

Figure 13. Output of Stop in Task 1



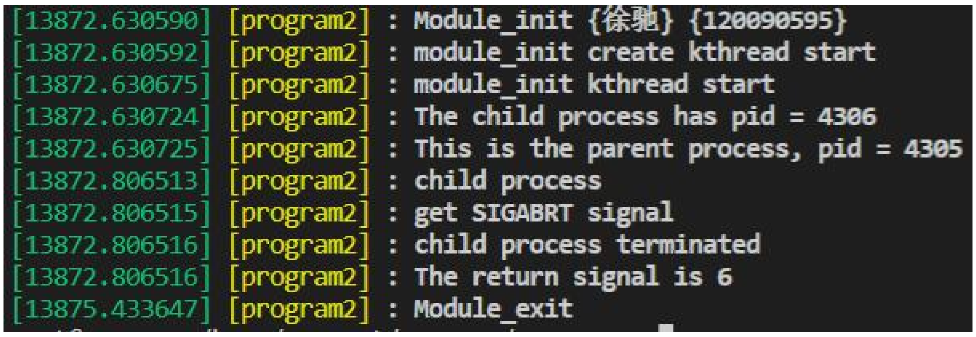
```
Process start to fork
I'm the Parent Process, my pid = 3658
I'm the Child Process, my pid = 3659
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTERM program

Parent process recieves SIGCHLD signal
CHILD EXECUTION FAILED: 15
```

Figure 14. Output of Terminate in Task 1



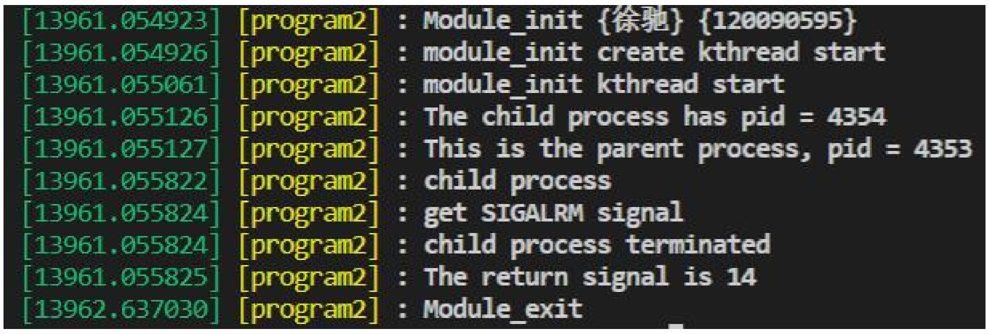
```
Process start to fork
I'm the Parent Process, my pid = 3684
I'm the Child Process, my pid = 3685
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTRAP program

Parent process recieves SIGCHLD signal
CHILD EXECUTION FAILED: 5
```

Figure 15. Output of Trap in Task 1



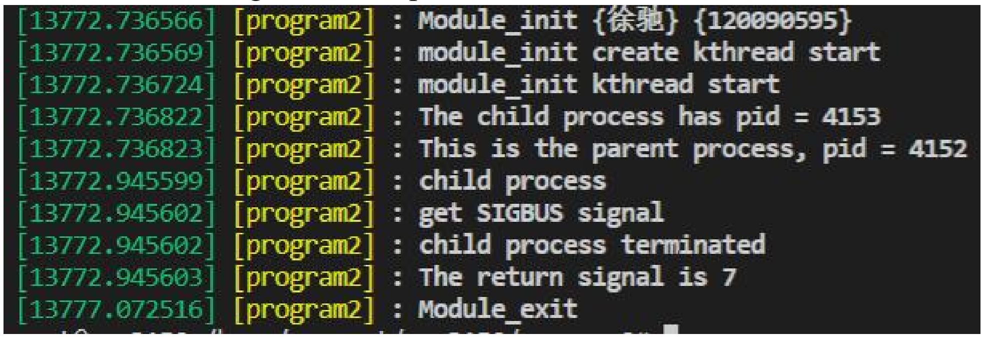
```
[13872.630590] [program2] : Module_init {徐驰} {120090595}
[13872.630592] [program2] : module_init create kthread start
[13872.630675] [program2] : module_init kthread start
[13872.630724] [program2] : The child process has pid = 4306
[13872.630725] [program2] : This is the parent process, pid = 4305
[13872.806513] [program2] : child process
[13872.806515] [program2] : get SIGABRT signal
[13872.806516] [program2] : child process terminated
[13872.806516] [program2] : The return signal is 6
[13875.433647] [program2] : Module_exit
```

Figure 16. Output of SIGABRT in Task 2



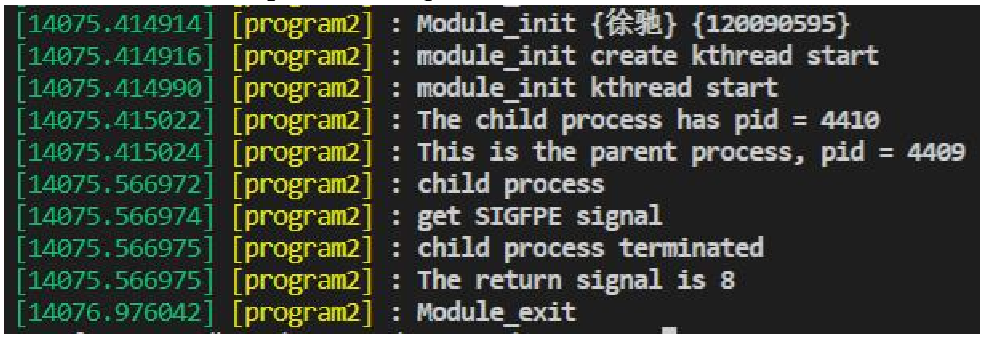
```
[13961.054923] [program2] : Module_init {徐驰} {120090595}
[13961.054926] [program2] : module_init create kthread start
[13961.055061] [program2] : module_init kthread start
[13961.055126] [program2] : The child process has pid = 4354
[13961.055127] [program2] : This is the parent process, pid = 4353
[13961.055822] [program2] : child process
[13961.055824] [program2] : get SIGALRM signal
[13961.055824] [program2] : child process terminated
[13961.055825] [program2] : The return signal is 14
[13962.637030] [program2] : Module_exit
```

Figure 17. Output of SIGALRM in Task 2



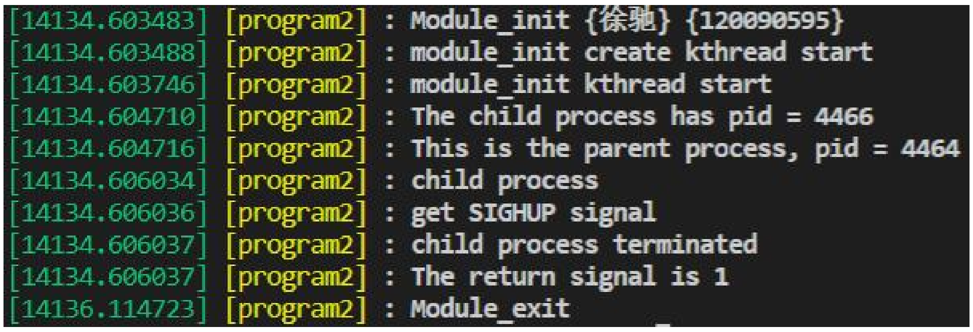
```
[13772.736566] [program2] : Module_init {徐驰} {120090595}
[13772.736569] [program2] : module_init create kthread start
[13772.736724] [program2] : module_init kthread start
[13772.736822] [program2] : The child process has pid = 4153
[13772.736823] [program2] : This is the parent process, pid = 4152
[13772.945599] [program2] : child process
[13772.945602] [program2] : get SIGBUS signal
[13772.945602] [program2] : child process terminated
[13772.945603] [program2] : The return signal is 7
[13777.072516] [program2] : Module_exit
```

Figure 18. Output of SIGBUS in Task 2



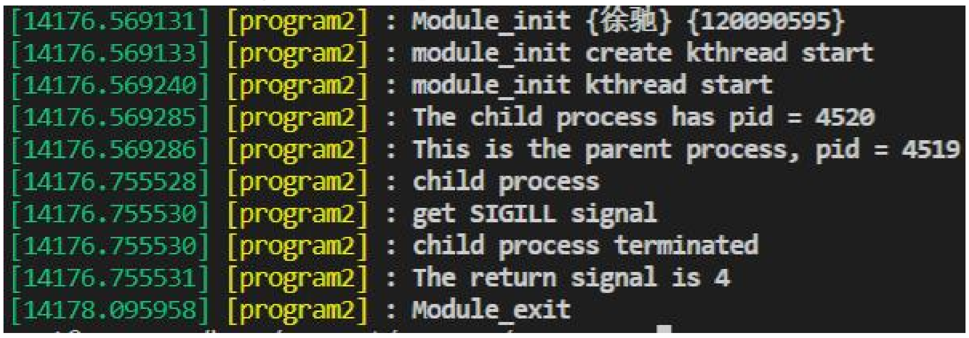
```
[14075.414914] [program2] : Module_init {徐驰} {120090595}
[14075.414916] [program2] : module_init create kthread start
[14075.414990] [program2] : module_init kthread start
[14075.415022] [program2] : The child process has pid = 4410
[14075.415024] [program2] : This is the parent process, pid = 4409
[14075.566972] [program2] : child process
[14075.566974] [program2] : get SIGFPE signal
[14075.566975] [program2] : child process terminated
[14075.566975] [program2] : The return signal is 8
[14076.976042] [program2] : Module_exit
```

Figure 19. Output of SIGFPE in Task 2



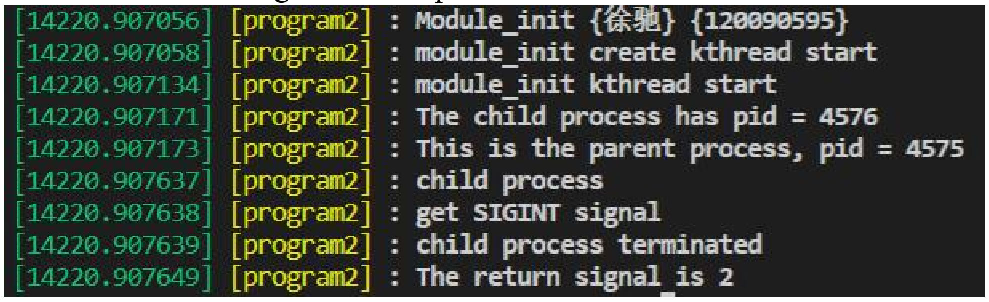
```
[14134.603483] [program2] : Module_init {徐驰} {120090595}
[14134.603488] [program2] : module_init create kthread start
[14134.603746] [program2] : module_init kthread start
[14134.604710] [program2] : The child process has pid = 4466
[14134.604716] [program2] : This is the parent process, pid = 4464
[14134.606034] [program2] : child process
[14134.606036] [program2] : get SIGHUP signal
[14134.606037] [program2] : child process terminated
[14134.606037] [program2] : The return signal is 1
[14136.114723] [program2] : Module_exit
```

Figure 20. Output of SIGHUP in Task 2



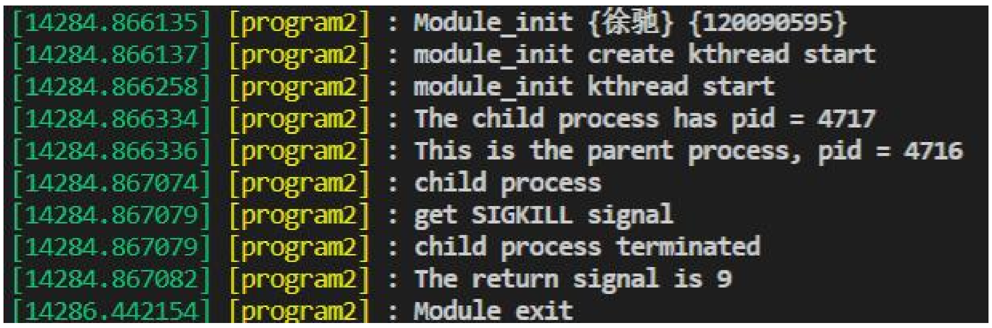
```
[14176.569131] [program2] : Module_init {徐驰} {120090595}
[14176.569133] [program2] : module_init create kthread start
[14176.569240] [program2] : module_init kthread start
[14176.569285] [program2] : The child process has pid = 4520
[14176.569286] [program2] : This is the parent process, pid = 4519
[14176.755528] [program2] : child process
[14176.755530] [program2] : get SIGILL signal
[14176.755530] [program2] : child process terminated
[14176.755531] [program2] : The return signal is 4
[14178.095958] [program2] : Module_exit
```

Figure 21. Output of SIGILL in Task 2



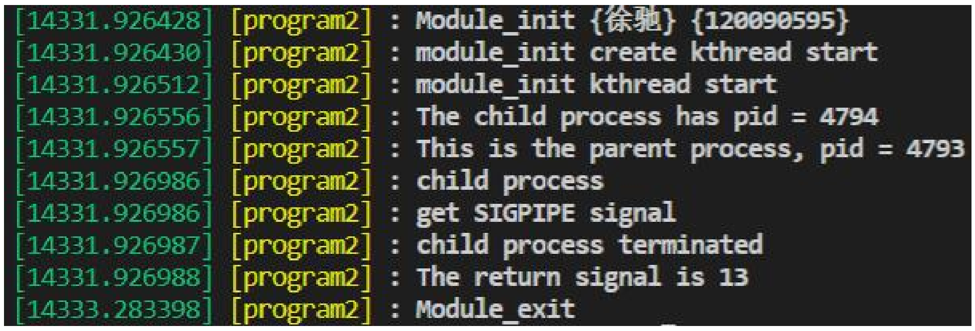
```
[14220.907056] [program2] : Module_init {徐驰} {120090595}
[14220.907058] [program2] : module_init create kthread start
[14220.907134] [program2] : module_init kthread start
[14220.907171] [program2] : The child process has pid = 4576
[14220.907173] [program2] : This is the parent process, pid = 4575
[14220.907637] [program2] : child process
[14220.907638] [program2] : get SIGINT signal
[14220.907639] [program2] : child process terminated
[14220.907649] [program2] : The return signal is 2
```

Figure 22. Output of SIGINT in Task 2



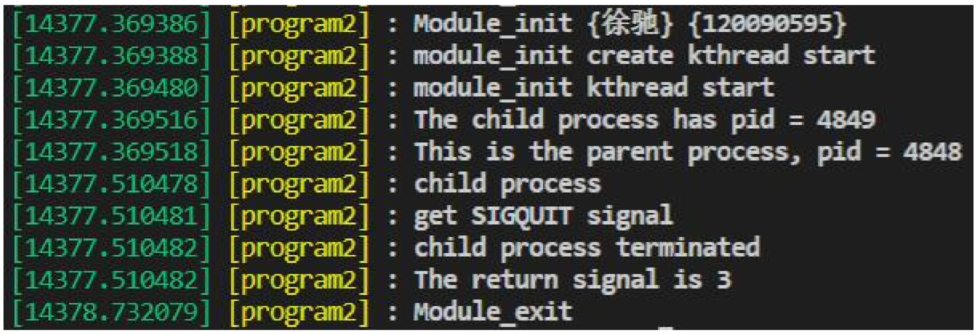
```
[14284.866135] [program2] : Module_init {徐驰} {120090595}
[14284.866137] [program2] : module_init create kthread start
[14284.866258] [program2] : module_init kthread start
[14284.866334] [program2] : The child process has pid = 4717
[14284.866336] [program2] : This is the parent process, pid = 4716
[14284.867074] [program2] : child process
[14284.867079] [program2] : get SIGKILL signal
[14284.867079] [program2] : child process terminated
[14284.867082] [program2] : The return signal is 9
[14286.442154] [program2] : Module_exit
```

Figure 23. Output of SIGKILL in Task 2



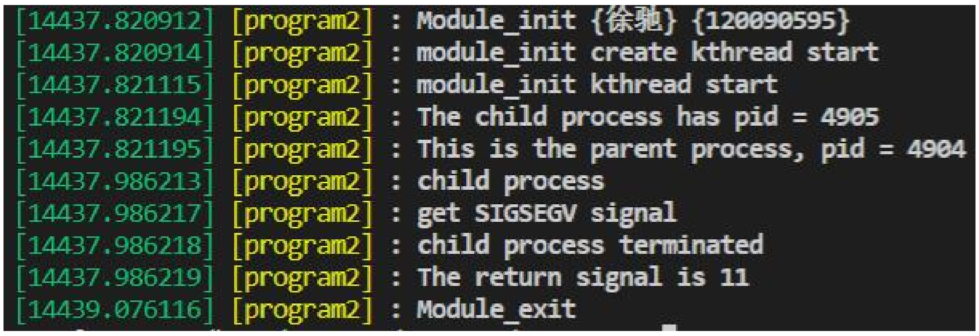
```
[14331.926428] [program2] : Module_init {徐驰} {120090595}
[14331.926430] [program2] : module_init create kthread start
[14331.926512] [program2] : module_init kthread start
[14331.926556] [program2] : The child process has pid = 4794
[14331.926557] [program2] : This is the parent process, pid = 4793
[14331.926986] [program2] : child process
[14331.926986] [program2] : get SIGPIPE signal
[14331.926987] [program2] : child process terminated
[14331.926988] [program2] : The return signal is 13
[14333.283398] [program2] : Module_exit
```

Figure 24. Output of SIGPIPE in Task 2



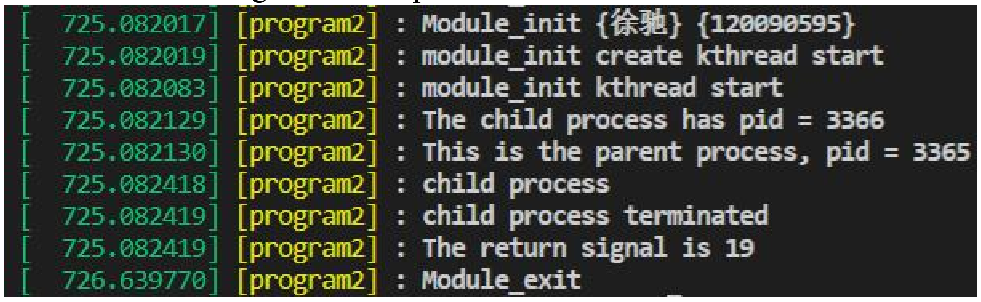
```
[14377.369386] [program2] : Module_init {徐驰} {120090595}
[14377.369388] [program2] : module_init create kthread start
[14377.369480] [program2] : module_init kthread start
[14377.369516] [program2] : The child process has pid = 4849
[14377.369518] [program2] : This is the parent process, pid = 4848
[14377.510478] [program2] : child process
[14377.510481] [program2] : get SIGQUIT signal
[14377.510482] [program2] : child process terminated
[14377.510482] [program2] : The return signal is 3
[14378.732079] [program2] : Module_exit
```

Figure 25. Output of SIGQUIT in Task 2



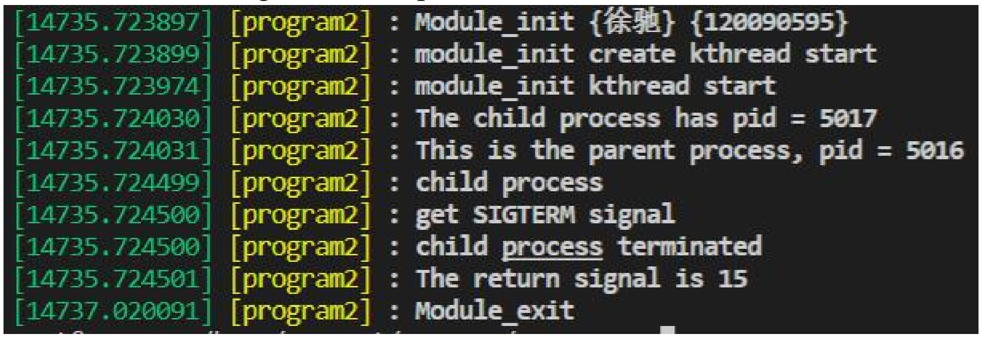
```
[14437.820912] [program2] : Module_init {徐驰} {120090595}
[14437.820914] [program2] : module_init create kthread start
[14437.821115] [program2] : module_init kthread start
[14437.821194] [program2] : The child process has pid = 4905
[14437.821195] [program2] : This is the parent process, pid = 4904
[14437.986213] [program2] : child process
[14437.986217] [program2] : get SIGSEGV signal
[14437.986218] [program2] : child process terminated
[14437.986219] [program2] : The return signal is 11
[14439.076116] [program2] : Module_exit
```

Figure 26. Output of SIGSEGV in Task 2



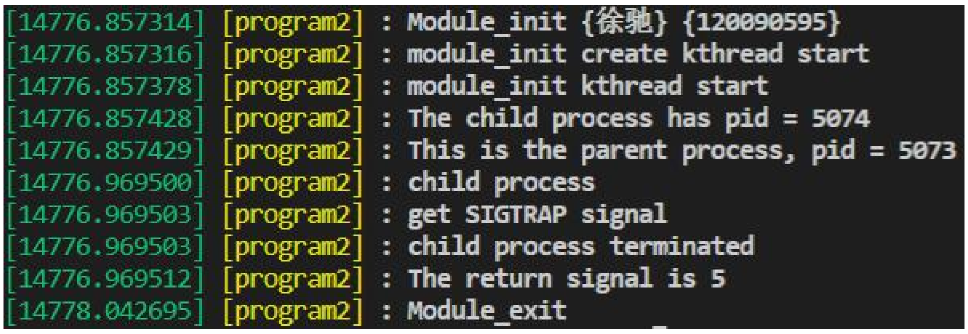
```
[  725.082017] [program2] : Module_init {徐驰} {120090595}
[  725.082019] [program2] : module_init create kthread start
[  725.082083] [program2] : module_init kthread start
[  725.082129] [program2] : The child process has pid = 3366
[  725.082130] [program2] : This is the parent process, pid = 3365
[  725.082418] [program2] : child process
[  725.082419] [program2] : child process terminated
[  725.082419] [program2] : The return signal is 19
[  726.639770] [program2] : Module_exit
```

Figure 27. Output of SIGSTOP in Task 2



```
[14735.723897] [program2] : Module_init {徐驰} {120090595}
[14735.723899] [program2] : module_init create kthread start
[14735.723974] [program2] : module_init kthread start
[14735.724030] [program2] : The child process has pid = 5017
[14735.724031] [program2] : This is the parent process, pid = 5016
[14735.724499] [program2] : child process
[14735.724500] [program2] : get SIGTERM signal
[14735.724500] [program2] : child process terminated
[14735.724501] [program2] : The return signal is 15
[14737.020091] [program2] : Module_exit
```

Figure 28. Output of SIGTERM in Task 2



```
[14776.857314] [program2] : Module_init {徐驰} {120090595}
[14776.857316] [program2] : module_init create kthread start
[14776.857378] [program2] : module_init kthread start
[14776.857428] [program2] : The child process has pid = 5074
[14776.857429] [program2] : This is the parent process, pid = 5073
[14776.969500] [program2] : child process
[14776.969503] [program2] : get SIGTRAP signal
[14776.969503] [program2] : child process terminated
[14776.969512] [program2] : The return signal is 5
[14778.042695] [program2] : Module_exit
```

Figure 29. Output of SIGTRAP in Task 2

4. Conclusion

This assignment let me know how operating system fork a new process and communication between processes. From task 1, I know that how to fork a new process in user mode. From task 2, when I write my own LKM, I know that EXPORT_SYMBOL function can export the function I want in kernel. So I can call the function directly. It is a very convenient function.