CSC3150 Operating Systems

Assignment#1 Report: Kernel-Mode Multi-Process Programming

Name: XU,Zijian
Student ID: 120090620
E-mail: 120090620@link.cuhk.edu.cn
Date: October 10, 2022

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

1 The program design

1.1 Task 1

The programmer designed a check function to tell the signal. It can tell 15 situations that the test file offered.

This program fork a child process to execute the test program. This program chooses wait_pid() function to deal with the STOP signal. This program will print the signal received.

1.2 Task 2

This program defines structure wait_opts for my_wait function. The programmer designed a check function to tell the signal. It can tell 15 situations that the test file offered. This program defines my_execve function to execute the test file. This program defines my_wait function to wait. This program defines my_fork function to create a new thread. This program defines module_init and module_exit functions to initial and exit the module.

1.3 bonus

Too difficult for me; I don't design this.

2 Set up the development and Compile kernel

The programmer downloads the Linux-5.10.104 from the website and compiled it.

This program changes the Linux source code (Export_Symbol) and compiles the kernel.

The 'sudo insmod program2.ko' can use to change the model.

3 Program Output

This part will only show some important output.

3.1 Task 1 Output

```
vagrant@csc3150:~/csc3150/program1$ ./program1 ./abort
Process start to fork
I'm the Parent Process, my pid = 12713
I'm the Child Process, my pid = 12714
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGABRT program

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

Figure 1: SIGABRT signal.

```
vagrant@csc3150:~/csc3150/program1$ ./program1 ./normal
Process start to fork
I'm the Parent Process, my pid = 12672
I'm the Child Process, my pid = 12673
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
```

Figure 2: NORMAL signal.

```
vagrant@csc3150:~/csc3150/program1$ ./program1 ./stop
Process start to fork
I'm the Parent Process, my pid = 12781
I'm the Child Process, my pid = 12782
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

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

Figure 3: Stop signal.

3.2 Task 2 Output

```
vagrant@csc3150:~/csc3150/program2$ sudo insmod program2.ko
vagrant@csc3150:~/csc3150/program2$ sudo rmmod program2
vagrant@csc3150:~/csc3150/program2$ dmesg
[19543.583130] [program2] : module_init {Xu Zijian} {120090620}
[19543.583639] [program2] : module_init create kthread start
[19543.584999] [program2] : module_init kthread start
[19543.586399] [program2] : The child process pid = 10975
[19543.586400] [program2] : This is the parent process, pid = 10973
[19543.586400] [program2] : child process
[19543.586401] [program2] : get SIGINT signal
[19543.586401] [program2] : child process terminated
[19543.586402] [program2] : The return signal is 2
[19550.108626] [program2] : module_exit
```

Figure 4: SIGINT signal.

4 Study Result

The programmer learned how the kernel works. The programmer knows about Multi-programming. The programmer improves coding skills. The programmer learns that clang-format.