CSC3150 Project1 Report

Liang MingRui 120090723

2022.10.5

My way to design the program:

Task1:

For task1, the mode is user mode. I can use those user mode functions to create a child process to execute those test files. First, I need to use fork() to create a child process. Then I use waitpid() to allow the parent process to wait until the child process terminates. In the child process, I use execve() to execute those test files. After that, the parent process will receive different signals according to different test files. Lastly, I print out the signal that parent process receives in the terminal.

Task2:

For task2, the mode is kernel mode. I can only use those functions in the kernel mode instead of user mode functions. In the code, I should firstly use kthread_create() and wake_up_process() to create a kernel thread and to wake up my_fork(). In function my_fork, I fork a child process using kernel_clone(). Then in the child process, I create a function called new_create_exec(). It is more like a wrapper function of do_execve(). In new_create_exec(), I use getname_kernel() to get the content of test file and put it into the do_execve as a parameter. In the parent process, I use new_create_wait() to ask parent process to wait until the child process terminates. In new_create_wait(), I use do_wait() to help me finish this job, and also I create three int function(is_fnormal, is_stop, is_failed) to help me judge what the return signal is. According to different signals, I print out different messages(all 15 signals showing in the task1 are considered). Lastly, the kernel module exits. With the help of insmod and rmmod, I can insert my module in the kernel module and remove it. After that, the messages will be printed out to show the correctness of my project.

Development environment:

Task1:

In task1, the environment is not important. I just make sure the gcc version is correct.

Task2:

In task2, the environment is very important. Originally, my kernel version was 4.4. In order to satisfy the version 5.10, I need to download the version 5.10 from the internet. Before compiling it, I also need to do the process-export symbol. The functions getname_kernel(), do_wait(), do_execve() and kernel_clone() are all needed to export

symbol. Also. I need to declare them in my code first. After that, I compile the kernel. First step is to clear the previous settings and start configuration. Then, I need to build kernel image and modules with the commands `make bzImage -j$(nproc)` and `make modules -j$(nproc)`. Lastly, I install kernel modules and kernel with the commands `make modules_install` and `make install`. Rebooting the VM to get the version 5.10.10.

Screenshot:

In task1, the output is the same as demo.

In task2, the output is a little different than demo.

Task1:(15 cases)

Normal:

```
vagrant@csc3150:~/csc3150/Assignment_1_120090723/source/program1$ ./program1 ./normal
Process start to fork
I'm the Child Process, my pid = 10046
I'm the Parent Process, my pid = 10045
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
```

Stop:

```
vagrant@csc3150:~/csc3150/Assignment_1_120090723/source/program1$ ./program1 ./stop
Process start to fork
I'm the Parent Process, my pid = 2944
I'm the Child Process, my pid = 2945
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
```

Abort:

```
vagrant@csc3150:~/csc3150/Assignment_1_120090723/source/program1$ ./program1 ./abort
Process start to fork
I'm the Child Process, my pid = 3007
Child process start to execute test program:
I'm the Parent Process, my pid = 3006
------------CHILD PROCESS START------------
This is the SIGABRT program

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

Bus:

```
vagrant@csc3150:~/csc3150/Assignment_1_120090723/source/program1$ ./program1 ./bus
Process start to fork
I'm the Child Process, my pid = 3055
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGBUS program

I'm the Parent Process, my pid = 3054
Parent process receives SIGCHLD signal
child process get SIGBUS signal
```

Floating:

```
vagrant@csc3150:~/csc3150/Assignment_1_120090723/source/program1$ ./program1 ./floating
Process start to fork
I'm the Parent Process, my pid = 3096
I'm the Child Process, my pid = 3097
Child 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_120090723/source/program1$ ./program1 ./hangup
Process start to fork
I'm the Parent Process, my pid = 3112
I'm the Child Process, my pid = 3113
Child 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_120090723/source/program1$ ./program1 ./illegal_instr
Process start to fork
I'm the Parent Process, my pid = 3187
I'm the Child Process, my pid = 3188
Child 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_120090723/source/program1$ ./program1 ./interrupt
Process start to fork
I'm the Parent Process, my pid = 3226
I'm the Child Process, my pid = 3227
Child 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_120090723/source/program1$ ./program1 ./kill
Process start to fork
I'm the Child Process, my pid = 3254
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGKILL program

I'm the Parent Process, my pid = 3253
Parent process receives SIGCHLD signal
child process get SIGKILL signal
```

Pipe:

```
vagrant@csc3150:~/csc3150/Assignment_1_120090723/source/program1$ ./program1 ./pipe
Process start to fork
I'm the Parent Process, my pid = 3288
I'm the Child Process, my pid = 3289
Child 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_120090723/source/program1$ ./program1 ./quit
Process start to fork
I'm the Parent Process, my pid = 3317
I'm the Child Process, my pid = 3318
Child 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_120090723/source/program1$ ./program1 ./segment_fault
Process start to fork
I'm the Parent Process, my pid = 3357
I'm the Child Process, my pid = 3358
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

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

Terminate:

```
vagrant@csc3150:~/csc3150/Assignment_1_120090723/source/program1$ ./program1 ./terminate
Process start to fork
I'm the Parent Process, my pid = 3385
I'm the Child Process, my pid = 3386
Child 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_120090723/source/program1$ ./program1 ./trap
Process start to fork
I'm the Parent Process, my pid = 3425
I'm the Child Process, my pid = 3426
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTRAP program

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

Alarm:

```
vagrant@csc3150:~/csc3150/Assignment_1_120090723/source/program1$ ./program1 ./alarm
Process start to fork
I'm the Parent Process, my pid = 3452
I'm the Child Process, my pid = 3453
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGALRM program

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

Task2:(15cases)

Normal:

```
[  723.570697] [program2] : Module_init {Liang MingRui} {120090723}
[  723.570698] [program2] : module_init create kthread start
[  723.571548] [program2] : Module_init kthread start
[  723.572010] [program2] : The child process has pid = 2719
[  723.572011] [program2] : This is the parent process, pid = 2718
[  723.572012] [program2] : child process
[  723.574636] [program2] : get normal signal
[  723.574636] [program2] : child process normally terminated
[  723.574637] [program2] : The return signal is 25600
[  732.329856] [program2] : Module_exit./my
```

Stop:

```
[  822.297544] [program2] : Module_init {Liang MingRui} {120090723}
[  822.297569] [program2] : module_init create kthread start
[  822.297858] [program2] : Module_init kthread start
[  822.298044] [program2] : The child process has pid = 3180
[  822.298045] [program2] : This is the parent process, pid = 3179
[  822.298045] [program2] : child process
[  822.299733] [program2] : get SIGSTOP signal
[  822.299734] [program2] : child process terminated
[  822.299735] [program2] : The return signal is 19
[  826.991615] [program2] : Module_exit./my
```

Bus:

```
[  883.425315] [program2] : Module_init {Liang MingRui} {120090723}
[  883.425316] [program2] : module_init create kthread start
[  883.425788] [program2] : Module_init kthread start
[  883.428967] [program2] : The child process has pid = 3642
[  883.428969] [program2] : This is the parent process, pid = 3641
[  883.428970] [program2] : child process
[  883.548393] [program2] : get SIGBUS signal
[  883.548394] [program2] : child process terminated
[  883.548395] [program2] : The return signal is 7
[  887.055428] [program2] : Module_exit./my
```

Interrupt:

```
[  936.581622] [program2] : Module_init {Liang MingRui} {120090723}
[  936.581623] [program2] : module_init create kthread start
[  936.582177] [program2] : Module_init kthread start
[  936.582524] [program2] : The child process has pid = 4098
[  936.582525] [program2] : This is the parent process, pid = 4097
[  936.582526] [program2] : child process
[  936.583625] [program2] : get SIGINT signal
[  936.583626] [program2] : child process terminated
[  936.583627] [program2] : The return signal is 2
[  940.942379] [program2] : Module_exit./my
```

Quit:

```
[ 1059.031922] [program2] : Module_init {Liang MingRui} {120090723}
[ 1059.031923] [program2] : module_init create kthread start
[ 1059.032425] [program2] : Module_init kthread start
[ 1059.034635] [program2] : The child process has pid = 4542
[ 1059.034666] [program2] : This is the parent process, pid = 4541
[ 1059.034667] [program2] : child process
[ 1059.137395] [program2] : get SIGQUIT signal
[ 1059.137397] [program2] : child process terminated
[ 1059.137398] [program2] : The return signal is 3
[ 1065.535024] [program2] : Module_exit./my
```

Trap:

```
[ 1110.462775] [program2] : Module_init {Liang MingRui} {120090723}
[ 1110.462777] [program2] : module_init create kthread start
[ 1110.463823] [program2] : Module_init kthread start
[ 1110.463926] [program2] : The child process has pid = 5023
[ 1110.463927] [program2] : This is the parent process, pid = 5022
[ 1110.463927] [program2] : child process
[ 1110.548211] [program2] : get SIGTRAP signal
[ 1110.548213] [program2] : child process terminated
[ 1110.548213] [program2] : The return signal is 5
[ 1114.812565] [program2] : Module_exit./my
```

Abort:

```
[ 1205.331605] [program2] : Module_init {Liang MingRui} {120090723}
[ 1205.331607] [program2] : module_init create kthread start
[ 1205.332173] [program2] : Module_init kthread start
[ 1205.333453] [program2] : The child process has pid = 5455
[ 1205.333454] [program2] : This is the parent process, pid = 5454
[ 1205.333455] [program2] : child process
[ 1205.450098] [program2] : get SIGABRT signal
[ 1205.450099] [program2] : child process terminated
[ 1205.450100] [program2] : The return signal is 6
[ 1209.609908] [program2] : Module_exit./my
```

Floating:

```
[ 1244.748925] [program2] : Module_init {Liang MingRui} {120090723}
[ 1244.748927] [program2] : module_init create kthread start
[ 1244.749378] [program2] : Module_init kthread start
[ 1244.749665] [program2] : The child process has pid = 5865
[ 1244.749667] [program2] : This is the parent process, pid = 5864
[ 1244.749667] [program2] : child process
[ 1244.850405] [program2] : get SIGFPE signal
[ 1244.850434] [program2] : child process terminated
[ 1244.850435] [program2] : The return signal is 8
[ 1249.233897] [program2] : Module_exit./my
```

Kill:

```
[ 1307.741238] [program2] : Module_init {Liang MingRui} {120090723}
[ 1307.741239] [program2] : module_init create kthread start
[ 1307.741677] [program2] : Module_init kthread start
[ 1307.741897] [program2] : The child process has pid = 6293
[ 1307.741897] [program2] : This is the parent process, pid = 6292
[ 1307.741898] [program2] : child process
[ 1307.748973] [program2] : get SIGKILL signal
[ 1307.748974] [program2] : child process terminated
[ 1307.748975] [program2] : The return signal is 9
[ 1312.186135] [program2] : Module_exit./my
```

Segment_fault:

```
[ 1350.884649] [program2] : Module_init {Liang MingRui} {120090723}
[ 1350.884651] [program2] : module_init create kthread start
[ 1350.885200] [program2] : Module_init kthread start
[ 1350.885594] [program2] : The child process has pid = 6710
[ 1350.885596] [program2] : This is the parent process, pid = 6709
[ 1350.885596] [program2] : child process
[ 1350.984100] [program2] : get SIGSEGV signal
[ 1350.984102] [program2] : child process terminated
[ 1350.984103] [program2] : The return signal is 11
[ 1354.421444] [program2] : Module_exit./my
```

Pipe:

```
[ 1393.673011] [program2] : Module_init {Liang MingRui} {120090723}
[ 1393.673013] [program2] : module_init create kthread start
[ 1393.673544] [program2] : Module_init kthread start
[ 1393.676489] [program2] : The child process has pid = 7110
[ 1393.676490] [program2] : This is the parent process, pid = 7109
[ 1393.676491] [program2] : child process
[ 1393.686561] [program2] : get SIGPIPE signal
[ 1393.686563] [program2] : child process terminated
[ 1393.686564] [program2] : The return signal is 13
[ 1397.194994] [program2] : Module_exit./my
```

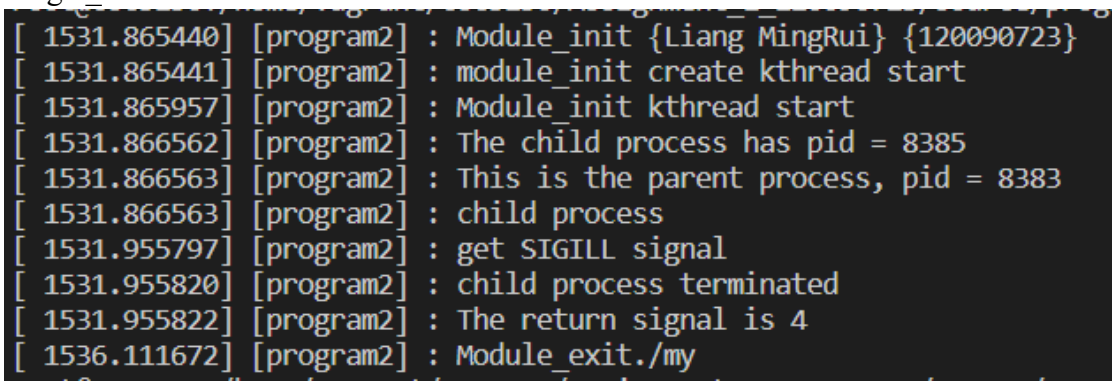
Alarm:

```
[ 1439.845651] [program2] : Module_init {Liang MingRui} {120090723}
[ 1439.845652] [program2] : module_init create kthread start
[ 1439.845920] [program2] : Module_init kthread start
[ 1439.847512] [program2] : The child process has pid = 7527
[ 1439.847513] [program2] : This is the parent process, pid = 7526
[ 1439.847513] [program2] : child process
[ 1439.848752] [program2] : get SIGALRM signal
[ 1439.848754] [program2] : child process terminated
[ 1439.848755] [program2] : The return signal is 14
[ 1444.334475] [program2] : Module_exit./my
```

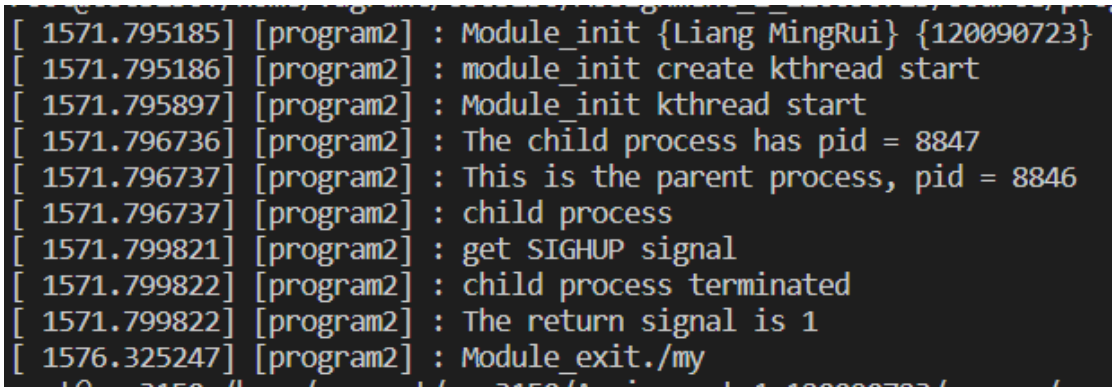
Terminate:

```
[ 1483.241140] [program2] : Module_init {Liang MingRui} {120090723}
[ 1483.241142] [program2] : module_init create kthread start
[ 1483.241586] [program2] : Module_init kthread start
[ 1483.242288] [program2] : The child process has pid = 7946
[ 1483.242289] [program2] : This is the parent process, pid = 7945
[ 1483.242289] [program2] : child process
[ 1483.246970] [program2] : get SIGTERM signal
[ 1483.246971] [program2] : child process terminated
[ 1483.246972] [program2] : The return signal is 15
[ 1487.294120] [program2] : Module_exit./my
```

Illegal_instr:



Hangup:



What did I learn from the tasks:

1. I learned how and why to install and compile the kernel, including EXPORT_STMBOL.
2. I learned how to create a child process in the user mode and the kernel mode.
3. I learned how to manipulate the parent process to wait the child process.
4. I learned how to use some simple Linux commands.
5. I learned some basic c language.
6. I learned how to use clang-format to change my code format.
7. I learned how to read kernel file to get the API I want.