CSC3150 Assignment Report 1

CSC3100 Assignment1 Report Author: Zhenkun Huo
Student number: 120090822
Version of OS: Ubuntu 16.04.7
Version of the kernel: 5.10.146
GCC version:5.4.0

1、Task 1

(1) Design ideas

In "program1", a child process is generated to execute test programs. At the same time, the parent process waits until child process terminates. Therefore, we use function fork() to achieve the goal.

After using fork(), the parent process and the child process will have different pids. Then we will use (if,else) to distinguish them.

For the child process, it needs to execute other test programs. So the function execve() is needed. For the parent process, it will wait until getting the signal from the child process. Here the function waitpid() is needed. At the same time, the second parameter of waited() &status will get the signal from the child process. Therefore, we can handle different signals.

There are 15 signals from the child process. One is process normal terminated, the other are exception cases. For the exception cases we use switch to print different signals.

(2) The execute procedure.

First of all, enter the folder Program1 in the Assignemnt1 folder

```
vagrant@csc3150:~/csc3150/Assignment1/Program1$
```

Then, type make in the terminal to compile all the test files and program1

```
vagrant@csc3150:~/csc3150/Assignment1/Program1$ make
cc -o abort abort.c
cc -o alarm alarm.c
cc -o bus bus.c
cc -o floating floating.c
cc -o hangup hangup.c
cc -o illegal_instr illegal_instr.c
cc -o interrupt interrupt.c
cc -o kill kill.c
cc -o normal normal.c
cc -o pipe pipe.c
cc -o program1 program1.c
cc -o quit quit.c
cc -o segment_fault segment_fault.c
cc -o stop stop.c
cc -o terminate terminate.c
cc -o trap trap.c
```

Finally, we type ./program1 ./test files names to execute all the test files. Followings are the output screeshot.

(3) Output

Normal:

```
vagrant@csc3150:~/csc3150/Assignment1/Program1$ ./program1 ./normal
Process start to fork
I'm the Parent Process, my pid = 3068
I'm the Child Process, my pid = 3069
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the normal program

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

Abort:

```
vagrant@csc3150:~/csc3150/Assignment1/Program1$ ./program1 ./abort
Process start to fork
I'm the Parent Process, my pid = 3022
I'm the Child Process, my pid = 3023
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGABRT program

Parent process receiving the SIGCHLD signal
child process get SIGABRT signal
```

Alarm:

```
vagrant@csc3150:~/csc3150/Assignment1/Program1$ ./program1 ./alarm
Process start to fork
I'm the Parent Process, my pid = 3214
I'm the Child Process, my pid = 3215
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGALRM program

Parent process receiving the SIGCHLD signal
child process get SIGALRM signal
```

Bus:

```
vagrant@csc3150:~/csc3150/Assignment1/Program1$ ./program1 ./bus
Process start to fork
I'm the Parent Process, my pid = 3268
I'm the Child Process, my pid = 3269
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGBUS program

Parent process receiving the SIGCHLD signal
child process get SIGBUS signal
```

Floating:

```
vagrant@csc3150:~/csc3150/Assignment1/Program1$ ./program1 ./floating
Process start to fork
I'm the Parent Process, my pid = 3295
I'm the Child Process, my pid = 3296
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGFPE program

Parent process receiving the SIGCHLD signal
child process get SIGFPE signal
```

Hangup:

```
vagrant@csc3150:~/csc3150/Assignment1/Program1$ ./program1 ./hangup
Process start to fork
I'm the Parent Process, my pid = 3335
I'm the Child Process, my pid = 3336
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGHUP program

Parent process receiving the SIGCHLD signal
child process get SIGHUP signal
```

Illegal_instr:

```
vagrant@csc3150:~/csc3150/Assignment1/Program1$ ./program1 ./illegal_instr
Process start to fork
I'm the Parent Process, my pid = 3409
I'm the Child Process, my pid = 3410
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGILL program

Parent process receiving the SIGCHLD signal
child process get SIGILL signal
```

Interrupt:

```
vagrant@csc3150:~/csc3150/Assignment1/Program1$ ./program1 ./interrupt
Process start to fork
I'm the Parent Process, my pid = 3436
I'm the Child Process, my pid = 3437
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGINT program

Parent process receiving the SIGCHLD signal
child process get SIGINT signal
```

Kill:

```
vagrant@csc3150:~/csc3150/Assignment1/Program1$ ./program1 ./kill
Process start to fork
I'm the Parent Process, my pid = 3450
I'm the Child Process, my pid = 3451
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGKILL program

Parent process receiving the SIGCHLD signal
child process get SIGKILL signal
```

Pipe:

```
vagrant@csc3150:~/csc3150/Assignment1/Program1$ ./program1 ./pipe
Process start to fork
I'm the Parent Process, my pid = 3476
I'm the Child Process, my pid = 3477
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGPIPE program

Parent process receiving the SIGCHLD signal
child process get SIGPIPE signal
```

Quit:

```
vagrant@csc3150:~/csc3150/Assignment1/Program1$ ./program1 ./quit
Process start to fork
I'm the Parent Process, my pid = 3502
I'm the Child Process, my pid = 3503
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

Parent process receiving the SIGCHLD signal
child process get SIGQUIT signal
```

Segment_fault:

```
vagrant@csc3150:~/csc3150/Assignment1/Program1$ ./program1 ./segment_fault
Process start to fork
I'm the Parent Process, my pid = 3653
I'm the Child Process, my pid = 3654
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

Parent process receiving the SIGCHLD signal
child process get SIGSEGV signal
```

Stop:

```
vagrant@csc3150:~/csc3150/Assignment1/Program1$ ./program1 ./stop
Process start to fork
I'm the Parent Process, my pid = 3535
I'm the Child Process, my pid = 3536
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

Parent process receiving the SIGCHLD signal
child process get SIGSTOP signal
```

Terminate:

```
vagrant@csc3150:~/csc3150/Assignment1/Program1$ ./program1 ./terminate
Process start to fork
I'm the Parent Process, my pid = 3561
I'm the Child Process, my pid = 3562
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTERM program

Parent process receiving the SIGCHLD signal
child process get SIGTERM signal
```

Trap:

```
vagrant@csc3150:~/csc3150/Assignment1/Program1$ ./program1 ./trap
Process start to fork
I'm the Parent Process, my pid = 3626
I'm the Child Process, my pid = 3627
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTRAP program

Parent process receiving the SIGCHLD signal
child process get SIGTRAP signal
```

2、Task2:

(1) Design ideas:
The task 2 is based on task 1 to create a kernel thread and run my_fork function when program2.ko is initialized. Just like task 1, it will generate a child process to execute the test programs. So we need to write three functions my_exec(), my_wait() and my_fork() corresponding to execute(), waitpid() and fork() in program1.
For my_exec(), its function is to find the path of the target file and let the child process execute it.
For my_wait(), its function is to let the parent process wait until get the child process signal. Then print the signal information and its return value. For different signals, the method switch() is used here.
For my_fork(), we use kernel_clone() to fork the child process .
In program2 init(), we use kthread_create() to create a kernel thread and call my_fork() function. However, the thread will not start running immediately and wake_up_process() is used here. When returned task_struct is passed to wake_up_process(), the thread will start to execute.

(2) Procedure of executing program2
First of all, we need to enter the fork.c and exit.c file in the kernel folder of the linux-5.10.146 to export the kernel_clone fuction in the fork.c and the do_wait function in the exit.c. Then we need to export the do_execu function in the exec.c and the getname function in the namei.c in the fs folder of the linux5.10.146.
Then we enter the folder of the linux5.10.146 and use sudo su to log in root account.
After that we use the make bziimage make ,make modules install and reboot the virtual machine.
Finally we enter the folder of the Program2 in the assignemnt1. Type make to complie all the files, than using the gcc test.c -o test to compile the test files.After the program2.ko file is created, we use insmod program2.ko to insert program2 module then using rmmod program2.ko to remove it. Then using the dmesg | tail -10 to detect the result.
Followings are the screenshot of output.
3、Output
(1) Abort

```
root@csc3150:/home/vagrant/csc3150/Assignment1/Program2# insmod program2.k
root@csc3150:/home/vagrant/csc3150/Assignment1/Program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/Assignment1/Program2# dmesg | tail -10
[ 9770.784306] [program2] : Module_init Zhenkun Huo 120090822
[ 9770.812335] [program2] : Module_init create kernel start
[ 9770.849801] [program2] : Kthread starts
[ 9770.861300] [program2] : The child process has pid = 5501
[ 9770.861351] [program2] : Child process
[ 9770.871925] [program2] : The parent process has pid = 5500
[ 9770.989434] [program2] : get SIGABRT signal
[ 9770.989437] [program2] : child process is aborted
[ 9771.014616] [program2] : The return signal is 6
[ 9776.471612] [program2] : Module_exit
```

(2) Alarm

```
root@csc3150:/home/vagrant/csc3150/Assignment1/Program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/Assignment1/Program2# dmesg | tail -10
[10236.856538] [program2] : Module_init Zhenkun Huo 120090822
[10236.857658] [program2] : Module_init create kernel start
[10236.894652] [program2] : Kthread starts
[10236.914921] [program2] : The child process has pid = 7551
[10236.914956] [program2] : Child process
[10236.916385] [program2] : The parent process has pid = 7550
[10236.958279] [program2] : get SIGALRM signal
[10236.958280] [program2] : child process alarmed
[10236.984434] [program2] : The return signal is 14
[10239.959868] [program2] : Module_exit
```

(3) Bus

```
root@csc3150:/home/vagrant/csc3150/Assignment1/Program2# dmesg | tail -10
[10362.544295] [program2] : Module_init Zhenkun Huo 120090822
[10362.565710] [program2] : Module_init create kernel start
[10362.592566] [program2] : Kthread starts
[10362.609125] [program2] : The child process has pid = 8511
[10362.609596] [program2] : Child process
[10362.634580] [program2] : The parent process has pid = 8510
[10362.798874] [program2] : get SIGBUS signal
[10362.798875] [program2] : child process got memory error
[10362.814013] [program2] : The return signal is 7
[10365.977178] [program2] : Module_exit
```

(4) Floating

```
root@csc3150:/home/vagrant/csc3150/Assignment1/Program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/Assignment1/Program2# dmesg | tail -10
[10468.128588] [program2] : Module_init Zhenkun Huo 120090822
[10468.129492] [program2] : Module_init create kernel start
[10468.130387] [program2] : Kthread starts
[10468.133174] [program2] : The child process has pid = 8922
[10468.133330] [program2] : Child process
[10468.156029] [program2] : The parent process has pid = 8921
[10468.308572] [program2] : get SIGFPE signal
[10468.308574] [program2] : child process got arthemetic error
[10468.337427] [program2] : The return signal is 8
[10472.584854] [program2] : Module_exit
```

(5) Hang up

```
root@csc3150:/home/vagrant/csc3150/Assignment1/Program2# dmesg | tail -10
[10560.378602] [program2] : Module_init Zhenkun Huo 120090822
[10560.412157] [program2] : Module_init create kernel start
[10560.440331] [program2] : Kthread starts
[10560.455019] [program2] : The child process has pid = 9530
[10560.455146] [program2] : Child process
[10560.455965] [program2] : The parent process has pid = 9529
[10560.485912] [program2] : get SIGHUP signal
[10560.485915] [program2] : child process is hung up
[10560.488224] [program2] : The return signal is 1
[10564.744325] [program2] : Module_exit
```

(6) illegal_instr

```
root@csc3150:/home/vagrant/csc3150/Assignment1/Program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/Assignment1/Program2# dmesg | tail -10
[10717.186033] [program2] : Module_init Zhenkun Huo 120090822
[10717.222454] [program2] : Module_init create kernel start
[10717.261884] [program2] : Kthread starts
[10717.286440] [program2] : The child process has pid = 10099
[10717.286485] [program2] : Child process
[10717.289211] [program2] : The parent process has pid = 10098
[10717.412504] [program2] : get SIGILL signal
[10717.412507] [program2] : child process got illegal instructions
[10717.440854] [program2] : The return signal is 4
[10721.280347] [program2] : Module_exit
```

(7) interrupt

```
root@csc3150:/home/vagrant/csc3150/Assignment1/Program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/Assignment1/Program2# dmesg | tail -10
[10779.768191] [program2] : Module_init Zhenkun Huo 120090822
[10779.786852] [program2] : Module_init create kernel start
[10779.816603] [program2] : Kthread starts
[10779.835295] [program2] : The child process has pid = 10560
[10779.835316] [program2] : Child process
[10779.836883] [program2] : The parent process has pid = 10559
[10779.839040] [program2] : get SIGINT signal
[10779.839041] [program2] : child process is interrupted
[10779.840243] [program2] : The return signal is 2
[10783.191555] [program2] : Module_exit
```

(8) kill

```
root@csc3150:/home/vagrant/csc3150/Assignment1/Program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/Assignment1/Program2# dmesg | tail -10
[10848.312343] [program2] : Module_init Zhenkun Huo 120090822
[10848.331628] [program2] : Module_init create kernel start
[10848.358917] [program2] : Kthread starts
[10848.376244] [program2] : The child process has pid = 11308
[10848.376278] [program2] : Child process
[10848.377350] [program2] : The parent process has pid = 11307
[10848.399509] [program2] : get SIGKILL signal
[10848.399509] [program2] : child process is killed
[10848.407376] [program2] : The return signal is 9
[10851.794944] [program2] : Module_exit
```

(9) normal

```
root@csc3150:/home/vagrant/csc3150/Assignment1/Program2# dmesg | tail -10
[11008.332447] CR2: ffffffffc07150f5 CR3: 0000000166dfe000 CR4: 00000000000506e0
[11036.368746] [program2] : Module_init Zhenkun Huo 120090822
[11036.387044] [program2] : Module_init create kernel start
[11036.415433] [program2] : Kthread starts
[11036.434129] [program2] : The child process has pid = 12715
[11036.434141] [program2] : Child process
[11036.435136] [program2] : The parent process has pid = 12714
[11041.482708] [program2] : get  signal
[11041.482712] [program2] : The return signal is 0
[11044.871775] [program2] : Module_exit
root@csc3150:/home/vagrant/csc3150/Assignment1/Program2# make
```

(10) pipe

```
root@csc3150:/home/vagrant/csc3150/Assignment1/Program2# dmesg | tail -10
[11222.992186] [program2] : Module_init Zhenkun Huo 120090822
[11222.993153] [program2] : Module_init create kernel start
[11222.995447] [program2] : Kthread starts
[11222.996322] [program2] : The child process has pid = 13571
[11222.996474] [program2] : Child process
[11222.997224] [program2] : The parent process has pid = 13570
[11223.036077] [program2] : get SIGPIPE signal
[11223.036078] [program2] : child process is piped
[11223.054993] [program2] : The return signal is 13
[11226.511494] [program2] : Module_exit
```

(11) quit

```
[11323.962831] [program2] : Module_init Zhenkun Huo 120090822
[11323.993287] [program2] : Module_init create kernel start
[11324.012041] [program2] : Kthread starts
[11324.035363] [program2] : The child process has pid = 14449
[11324.035441] [program2] : Child process
[11324.068465] [program2] : The parent process has pid = 14448
[11324.188842] [program2] : get SIGQUIT signal
[11324.188844] [program2] : child process quitted
[11324.209777] [program2] : The return signal is 3
[11327.648692] [program2] : Module_exit
```

(12) segment_fault

```
[11637.649567] [program2] : Module_init Zhenkun Huo 120090822
[11637.680048] [program2] : Module_init create kernel start
[11637.701206] [program2] : Kthread starts
[11637.715340] [program2] : The child process has pid = 16627
[11637.715365] [program2] : Child process
[11637.733475] [program2] : The parent process has pid = 16626
[11637.853579] [program2] : get SIGSEGV signal
[11637.853582] [program2] : child process got segment faults
[11637.891684] [program2] : The return signal is 11
[11641.961095] [program2] : Module_exit
```

(13) stop

```
root@csc3150:/home/vagrant/csc3150/Assignment1/Program2# rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/Assignment1/Program2# dmesg | tail -10
[11470.612324] [program2] : Child process
[11470.622197] [program2] : The parent process has pid = 15312
[11478.976988] [program2] : Module_exit
[11537.274084] [program2] : Module_init Zhenkun Huo 120090822
[11537.275336] [program2] : Module_init create kernel start
[11537.276479] [program2] : Kthread starts
[11537.277411] [program2] : The child process has pid = 15781
[11537.277438] [program2] : Child process
[11537.290060] [program2] : The parent process has pid = 15780
[11541.424598] [program2] : Module_exit
```

(14) terminate

```
[11710.248558] [program2] : Module_init Zhenkun Huo 120090822
[11710.283292] [program2] : Module_init create kernel start
[11710.310153] [program2] : Kthread starts
[11710.331062] [program2] : The child process has pid = 17087
[11710.331081] [program2] : Child process
[11710.334063] [program2] : The parent process has pid = 17086
[11710.345437] [program2] : get SIGTERM signal
[11710.345438] [program2] : child process terminated
[11710.361119] [program2] : The return signal is 15
[11714.425139] [program2] : Module_exit
```

(15) trap

```
root@csc3150:/home/vagrant/csc3150/Assignment1/Program2# dmesg | tail -10
[11791.801456] [program2] : Module_init Zhenkun Huo 120090822
[11791.839596] [program2] : Module_init create kernel start
[11791.881442] [program2] : Kthread starts
[11791.904256] [program2] : The child process has pid = 17531
[11791.904393] [program2] : Child process
[11791.931499] [program2] : The parent process has pid = 17530
[11792.049383] [program2] : get SIGTRAP signal
[11792.049385] [program2] : child process is trapped
[11792.069891] [program2] : The return signal is 5
[11794.881600] [program2] : Module_exit
```

3、From the task 1, I learned how to fork a child process using fork() and distinguish parent and child process by different return value of fork(). I also have a better understanding of waitpid(), which tells me the wait mechanism of parent and child process. In the child process part, the using of execve() makes me know how it executes an executable file. In parent process part, I knew more about process signals and learned how to use macros to handle different situations. The task 2 is based on the task 1. The difference is a module is inserted into the kernel to complete the task. The task 2 will creat a kernel thread to fork a child process to execute the test program. Here, we need to write three kernel functions my_fork(), my_wait() and my_exec() to finish the task. When the module is inserted, the kernel thread will be created immediately and call my_fork(). From task 2, I have a better understanding of the Linux kernel. I learned how to use some kernel functions, such as kernel_clone() and do_execve(). At the same time, I also knew how a module is inserted, runs and exits. Through this assignment, I

have a better understanding of Linux Kernel-Mode multi-process programming. I learned how parent process and process run concurrently