NAME: 刘芷萱

STUDENT ID: 118010202

Program1:

1. Design of my program:

In my main function, I use “fork()”, to create a child process from a parent process. Child process and parent process execute the file concurrently.

I use the return value of fork to separate two processes. In the child process, the test file will be execute by using the function “execve()”. The terminal will display the information of the test program and also raise signal. Then the child process is finished.

Parent process is waiting for child process to finish and also receive signal by using function “waitpid()”. After receiving the signal from child process, parent process will display the signal.

2. The environment:

Kernel version: 4.10.14

OS version: Ubuntu 16.04

VM version: VirtualBox 6.0.18

3. Step to execute

1) first is to compile: In the 'program1' directory, type 'make' command and enter.

2) execute: In the 'program1' directory, type './program1 $TEST_CASE’

4. Screen shot of my program output:

1) abort.c

```
[10/08/20]seed@VM:~/.../program1$ ./program1 ./abort
Process start to fork
I am the parent process, my pid = 3516
I am the child process, my pid = 3517
Child process start to execute the program
------------CHILD PROCESS START------------
This is the SIGABRT program

Parent process receiving the SIGCHLD signal
child process get SIGABRT signal
child process is abort by abort signal
CHILD EXECUTION FAILED!!
[10/08/20]seed@VM:~/.../program1$
```

2)alarm.c

```
[10/08/20]seed@VM:~/.../program1$ ./program1 ./alarm
Process start to fork
I am the parent process, my pid = 3521
I am the child process, my pid = 3522
Child process start to execute the program
------------CHILD PROCESS START------------
This is the SIGALRM program

Parent process receiving the SIGCHLD signal
child process get SIGALRM signal
in child process the time limit to alarm function elapse
CHILD EXECUTION FAILED!!
[10/08/20]seed@VM:~/.../program1$
```

3)bus.c

```
[10/08/20]seed@VM:~/.../program1$ ./program1 ./bus
Process start to fork
I am the parent process, my pid = 3529
I am the child process, my pid = 3530
Child process start to execute the program
------------CHILD PROCESS START------------
This is the SIGBUS program

Parent process receiving the SIGCHLD signal
child process get SIGBUS signal
child process access to an undefined portion of a memory object
CHILD EXECUTION FAILED!!
[10/08/20]seed@VM:~/.../program1$
```

4) floating.c

```
[10/08/20]seed@VM:~/.../program1$ ./program1 ./floating
Process start to fork
I am the parent process, my pid = 3532
I am the child process, my pid = 3533
Child process start to execute the program
------------CHILD PROCESS START------------
This is the SIGFPE program

Parent process receiving the SIGCHLD signal
child process get SIGFPE signal
child process encounter erroneous arithmetic operation
CHILD EXECUTION FAILED!!
[10/08/20]seed@VM:~/.../program1$
```

5) hangup.c

```
[10/08/20]seed@VM:~/.../program1$ ./program1 ./hangup
Process start to fork
I am the parent process, my pid = 3574
I am the child process, my pid = 3575
Child process start to execute the program
------------CHILD PROCESS START------------
This is the SIGHUP program

Parent process receiving the SIGCHLD signal
child process get SIGHUP signal
child process controlling terminal is closed
CHILD EXECUTION FAILED!!
[10/08/20]seed@VM:~/.../program1$
```

6) illegal_instr.c

```
[10/08/20]seed@VM:~/.../program1$ ./program1 ./illegal_instr
Process start to fork
I am the parent process, my pid = 3579
I am the child process, my pid = 3580
Child process start to execute the program
------------CHILD PROCESS START------------
This is the SIGILL program

Parent process receiving the SIGCHLD signal
child process get SIGILL signal
child process execute illegal instruction
CHILD EXECUTION FAILED!!
[10/08/20]seed@VM:~/.../program1$
```

7) interrupt.c

```
[10/08/20]seed@VM:~/.../program1$ ./program1 ./interrupt
Process start to fork
I am the parent process, my pid = 3584
I am the child process, my pid = 3585
Child process start to execute the program
------------CHILD PROCESS START------------
This is the SIGINT program

Parent process receiving the SIGCHLD signal
child process get SIGINT signal
child process terminate by interrupt signal
CHILD EXECUTION FAILED!!
[10/08/20]seed@VM:~/.../program1$
```

8) kill.c

```
[10/08/20]seed@VM:~/.../program1$ ./program1 ./kill
Process start to fork
I am the parent process, my pid = 3587
I am the child process, my pid = 3588
Child process start to execute the program
------------CHILD PROCESS START------------
This is the SIGKILL program

Parent process receiving the SIGCHLD signal
child process get SIGKILL signal
child process terminate immediately
CHILD EXECUTION FAILED!!
[10/08/20]seed@VM:~/.../program1$
```

9) normal.c

```
[10/08/20]seed@VM:~/.../program1$ ./program1 ./normal
Process start to fork
I am the parent process, my pid = 3590
I am the child process, my pid = 3591
Child process start to execute the program
------------CHILD PROCESS START------------
This is the normal program

------------CHILD PROCESS END------------
Parent process receiving the SIGCHLD signal
Normal termination with EXIT STATUS = 0
[10/08/20]seed@VM:~/.../program1$
```

10) pipe.c

```
[10/08/20]seed@VM:~/.../program1$ ./program1 ./pipe
Process start to fork
I am the parent process, my pid = 3595
I am the child process, my pid = 3596
Child process start to execute the program
------------CHILD PROCESS START------------
This is the SIGPIPE program

Parent process receiving the SIGCHLD signal
child process get SIGPIPE signal
child process write on a pipe with no one to read it
CHILD EXECUTION FAILED!!
[10/08/20]seed@VM:~/.../program1$
```

11) quit.c

```
[10/08/20]seed@VM:~/.../program1$ ./program1 ./quit
Process start to fork
I am the parent process, my pid = 3598
I am the child process, my pid = 3599
Child process start to execute the program
------------CHILD PROCESS START------------
This is the SIGQUIT program

Parent process receiving the SIGCHLD signal
child process get SIGQUIT signal
child process quit and perform a core dump
CHILD EXECUTION FAILED!!
[10/08/20]seed@VM:~/.../program1$
```

12) segment_fault.c

```
[10/08/20]seed@VM:~/.../program1$ ./program1 ./segment_fault
Process start to fork
I am the parent process, my pid = 3603
I am the child process, my pid = 3604
Child process start to execute the program
------------CHILD PROCESS START------------
This is the SIGSEGV program

Parent process receiving the SIGCHLD signal
child process get SIGSEGV signal
child process goes to invalid memory reference
CHILD EXECUTION FAILED!!
[10/08/20]seed@VM:~/.../program1$
```

13) stop.c

```
[10/08/20]seed@VM:~/.../program1$ ./program1 ./stop
Process start to fork
I am the parent process, my pid = 3606
I am the child process, my pid = 3607
Child process start to execute the program
------------CHILD PROCESS START------------
This is the SIGSTOP program

Parent process receiving the SIGCHLD signal
Child process get SIGSTOP signal
child process stopped
CHILD PROCESS STOPPED
[10/08/20]seed@VM:~/.../program1$
```

14) terminate.c

```
[10/08/20]seed@VM:~/.../program1$ ./program1 ./terminate
Process start to fork
I am the parent process, my pid = 3611
I am the child process, my pid = 3612
Child process start to execute the program
------------CHILD PROCESS START------------
This is the SIGTERM program

Parent process receiving the SIGCHLD signal
child process get SIGTERM signal
child process is required to terminate
CHILD EXECUTION FAILED!!
[10/08/20]seed@VM:~/.../program1$
```

15) trap.c

```
[10/08/20]seed@VM:~/.../program1$ ./program1 ./trap
Process start to fork
I am the parent process, my pid = 3614
I am the child process, my pid = 3615
Child process start to execute the program
------------CHILD PROCESS START------------
This is the SIGTRAP program

Parent process receiving the SIGCHLD signal
child process get SIGTRAP signal
child process is trapped by exception
CHILD EXECUTION FAILED!!
[10/08/20]seed@VM:~/.../program1$
```

5. What I learned from the task:

From this task, I know how the fork, the execve, and the wait function do their jobs. I learned how to process the signal raised by the child process. I learned some linux standard signals, and so on.

Program2

1. design of the program

1) In the module_init function, I create a kernel thread, and let the process execute the function called “my_fork”.

2) In my_fork function, I will use _do_fork() function to generate a child process, and let it execute “my_exec” function. Parent process is waiting for the signal passed by child process by the function “my_wait”.

3) In “my_exec” function, I use “do_exec()” function to execute the file.

4) In “my_wait” function, I use “do_wait()” function and “getname()” function to implement. After child process finish, parent process will receive the signal from child process.

2. The environment:

Kernel version: 4.10.14

OS version: Ubuntu 16.04

VM version: VirtualBox 6.0.18

3. Step to execute:

1) Compile: type “make” to compile the “program2.c”, and type “gcc -o test test.c” to compile the test program

2) Execute: First, type 'sudo insmod program2.ko' under 'program2' directory and enter. Next, type 'sudo rmmod program2' and enter to remove the program2 module. Finally, type “dmesg” to display the output.

4. Screen shot:

1) SIGBUS

```
[   74.216151] [program2] : module_init
[   74.216151] [program2] : module_init create kthread start
[   74.221441] [program2] : module_init kthread start
[   74.225089] [program2] : The child process has pid = 2592
[   74.225153] [program2] : This is the parent process, pid = 2590
[   74.225153] [program2] : child process
[   74.229659] [program2] : get SIGBUS signal
[   74.229660] [program2] : child process has an bus error
[   74.229660] [program2] : The return signal is 7
[  204.171549] [program2] : module_exit
```

2)SIGALRM

```
[  208.880808] [program2] : module_init
[  208.880808] [program2] : module_init create kthread start
[  208.885824] [program2] : module_init kthread start
[  208.888791] [program2] : The child process has pid = 2689
[  208.888791] [program2] : This is the parent process, pid = 2687
[  208.888791] [program2] : child process
[  210.893475] [program2] : get SIGALRM signal
[  210.893476] [program2] : child process has an alarm error
[  210.893476] [program2] : The return signal is 14
[  224.617423] [program2] : module_exit
```

3)SIGABRT

```
[  265.365336] [program2] : module_init
[  265.365337] [program2] : module_init create kthread start
[  265.368985] [program2] : module_init kthread start
[  265.372055] [program2] : The child process has pid = 2723
[  265.372056] [program2] : This is the parent process, pid = 2721
[  265.372056] [program2] : child process
[  265.372599] [program2] : get SIGABRT signal
[  265.372599] [program2] : child process has an abort error
[  265.372600] [program2] : The return signal is 6
[  273.618259] [program2] : module_exit
```

4)SIGFPE

```
[  316.494536] [program2] : module_init
[  316.494537] [program2] : module_init create kthread start
[  316.494551] [program2] : module_init kthread start
[  316.497477] [program2] : The child process has pid = 2747
[  316.497477] [program2] : This is the parent process, pid = 2745
[  316.497477] [program2] : child process
[  316.497932] [program2] : get SIGFPE signal
[  316.497933] [program2] : child process encounter erroneous arithmetic operation
[  316.497933] [program2] : The return signal is 8
[  334.238671] [program2] : module_exit
```

5)SIGUP

```
[  369.495918] [program2] : module_init
[  369.495919] [program2] : module_init create kthread start
[  369.500106] [program2] : module_init kthread start
[  369.502927] [program2] : The child process has pid = 2826
[  369.502928] [program2] : This is the parent process, pid = 2824
[  369.502928] [program2] : child process
[  369.505212] [program2] : get SIGUP signal
[  369.505212] [program2] : child process controlling terminal is closed
[  369.505212] [program2] : The return signal is 1
[  380.148739] [program2] : module_exit
```

6)SIGILL

```
[  409.006614] [program2] : module_init
[  409.006614] [program2] : module_init create kthread start
[  409.011227] [program2] : module_init kthread start
[  409.014115] [program2] : The child process has pid = 2852
[  409.014116] [program2] : This is the parent process, pid = 2850
[  409.014116] [program2] : child process
[  409.017272] [program2] : get SIGILL signal
[  409.017272] [program2] : child process execute illegal instruction
[  409.017273] [program2] : The return signal is 4
[  415.330030] [program2] : module_exit
```

7)SIGINT

```
[  433.032729] [program2] : module_init
[  433.032730] [program2] : module_init create kthread start
[  433.032743] [program2] : module_init kthread start
[  433.045498] [program2] : The child process has pid = 2873
[  433.045499] [program2] : This is the parent process, pid = 2871
[  433.045499] [program2] : child process
[  433.046164] [program2] : get SIGINT signal
[  433.046165] [program2] : child process terminate by interrupt signal
[  433.046165] [program2] : The return signal is 2
[  440.377898] [program2] : module_exit
```

8)SIGKILL

```
[  459.181818] [program2] : module_init
[  459.181819] [program2] : module_init create kthread start
[  459.186035] [program2] : module_init kthread start
[  459.188716] [program2] : The child process has pid = 2896
[  459.188717] [program2] : This is the parent process, pid = 2894
[  459.188717] [program2] : child process
[  459.193411] [program2] : get SIGKILL signal
[  459.193412] [program2] : child process terminate immediately
[  459.193412] [program2] : The return signal is 9
[  466.077926] [program2] : module_exit
```

9)SIGCHLD

```
[  485.892860] [program2] : module_init
[  485.892861] [program2] : module_init create kthread start
[  485.892880] [program2] : module_init kthread start
[  485.895649] [program2] : The child process has pid = 2918
[  485.895649] [program2] : This is the parent process, pid = 2916
[  485.895650] [program2] : child process
[  485.896054] [program2] : get SIGCHLD signal
[  485.896054] [program2] : Normal termination
[  491.011148] [program2] : module_exit
```

10)SIGPIPE

```
[  513.038929] [program2] : module_init
[  513.038930] [program2] : module_init create kthread start
[  513.043486] [program2] : module_init kthread start
[  513.045861] [program2] : The child process has pid = 2941
[  513.045862] [program2] : This is the parent process, pid = 2939
[  513.045862] [program2] : child process
[  513.049503] [program2] : get SIGPIPE signal
[  513.049503] [program2] : child process write on a pipe with no one to read it
[  513.049504] [program2] : The return signal is 13
[  522.749799] [program2] : module_exit
```

11)SIGQUIT

```
[  541.853182] [program2] : module_init
[  541.853183] [program2] : module_init create kthread start
[  541.856865] [program2] : module_init kthread start
[  541.860324] [program2] : The child process has pid = 2962
[  541.860325] [program2] : This is the parent process, pid = 2960
[  541.860325] [program2] : child process
[  541.862886] [program2] : get SIGQUIT signal
[  541.862887] [program2] : child process quit and perform a core dump
[  541.862887] [program2] : The return signal is 3
[  546.038259] [program2] : module_exit
```

12)SIGSEGV

```
[  569.476584] [program2] : module_init
[  569.476584] [program2] : module_init create kthread start
[  569.476599] [program2] : module_init kthread start
[  569.479472] [program2] : The child process has pid = 2985
[  569.479472] [program2] : This is the parent process, pid = 2983
[  569.479473] [program2] : child process
[  569.479898] [program2] : get SIGSEGV signal
[  569.479898] [program2] : child process goes to invalid memory reference
[  569.479899] [program2] : The return signal is 11
```

13) STOP

```
[ 1027.867048] [program2] : module_init
[ 1027.867048] [program2] : module_init create kthread start
[ 1027.867061] [program2] : module_init kthread start
[ 1027.869862] [program2] : The child process has pid = 3066
[ 1027.869863] [program2] : This is the parent process, pid = 3064
[ 1027.869863] [program2] : child process
[ 1027.870240] [program2] : get STOP signal
[ 1027.870241] [program2] : child process stopped!
[ 1027.870241] [program2] : The return signal is 19
[ 1034.352131] [program2] : module_exit
```

14)SIGTERM

```
[ 1068.046426] [program2] : module_init
[ 1068.046427] [program2] : module_init create kthread start
[ 1068.046440] [program2] : module_init kthread start
[ 1068.048928] [program2] : The child process has pid = 3090
[ 1068.048929] [program2] : This is the parent process, pid = 3088
[ 1068.048930] [program2] : child process
[ 1068.049833] [program2] : get SIGTERM signal
[ 1068.049833] [program2] : child process is required to terminate
[ 1068.049834] [program2] : The return signal is 15
[ 1071.326682] [program2] : module_exit
```

15)SIGTRAP

```
[ 1089.519185] [program2] : module_init
[ 1089.519186] [program2] : module_init create kthread start
[ 1089.519200] [program2] : module_init kthread start
[ 1089.521942] [program2] : The child process has pid = 3109
[ 1089.521942] [program2] : This is the parent process, pid = 3107
[ 1089.521943] [program2] : child process
[ 1089.522365] [program2] : get SIGTRAP signal
[ 1089.522366] [program2] : child process is trapped by exception
[ 1089.522366] [program2] : The return signal is 5
[ 1116.732182] [program2] : module_exit
```

16)NORMAL

```
[  281.133208] [program2] : module_init
[  281.133209] [program2] : module_init create kthread start
[  281.140442] [program2] : module_init kthread start
[  281.143186] [program2] : The child process has pid = 3441
[  281.143187] [program2] : This is the parent process, pid = 3439
[  281.143187] [program2] : child process
[  281.143760] [program2] : get SIGCHLD signal
[  281.143760] [program2] : Normal termination
[  284.059979] [program2] : module_exit
```

5. What I learned:

1) I learned how to EXPORT_SYMBOL, how to compile a kernel

2) I learned how to use _do_fork(), do_execve(), do_wait() functions in kernel mode.

3) I learned the difference between user mode and kernel mode.

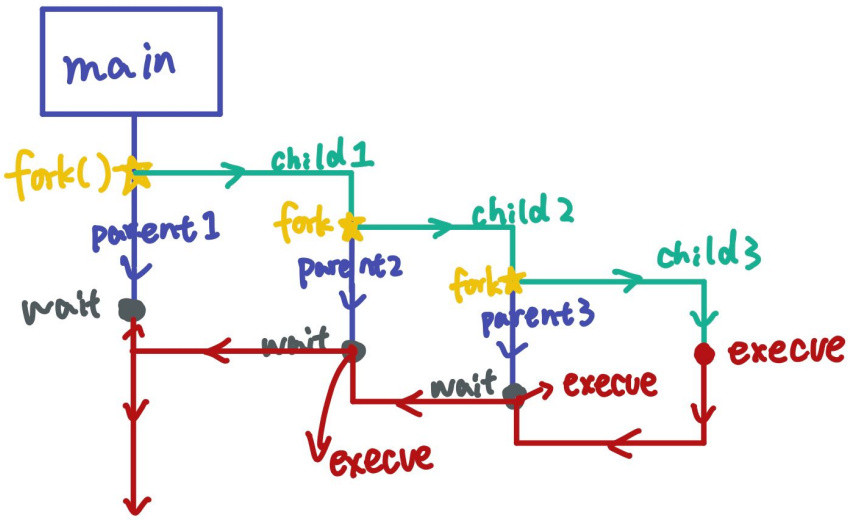
4) I learned how to process STOP signals and so on.

Bonus Question:

1. Design of my program:

The general idea of my program is using recursion to solve the question.

1) In my main function, I first initial to array pointer: pid_array and signal_array, which are used to store pid and signal in the child process. Because child process and parent process do not share the same memory, I use mmap function to global pid_array and signal_array. Therefore, those arrays will be passed by reference in my function correctly.
2) my_process function is my recursion function. For the base case, it will just execute the given file and return. For other cases, it will use fork() function to create a child process and a parent process. In the child process, it will go into my_process function again, while in the parent process, it will wait for the last child process to finish, and execute its own test file.
3) Judge_signal function is used to receive the signal raised by child process. display_mesg function will show the prompt according to the signal.
4) The main idea of my program can be expressed in the figure below (an example when argc = 4):



2. The environment:

Kernel version: 4.10.14

OS version: Ubuntu 16.04

VM version: VirtualBox 6.0.18

3. Steps to execute:

1) To compile: In the 'bonus' directory, type 'make' command and enter.

2) execute: In the 'bonus' directory, type './myfork $TEST_PRO1 $TEST_PRO2 $TEST_PRO3 ...', where $TEST_PRO1, $TEST_PRO2,... are names of programs myfork executes.

4. Screen shot:

1) the same file in example pdf:

```
Terminal
[10/11/20]seed@VM:~/.../bonus$ ./myfork hangup normal8 trap
------------CHILD PROCESS START------------
This is the SIGTRAP program

This is normal8 program
------------CHILD PROCESS START------------
This is the SIGHUP program

the process tree : 3898->3899->3900->3901
The child process(pid=3901) of parent process(pid=3900) is terminated by signal
Its signal number = 5
child process get SIGTRAP signal
child process is trapped by exception

The child process(pid=3900) of parent process(pid=3899) has normal execution
Its exit status is 0

The child process(pid=3899) of parent process(pid=3898) is terminated by signal
Its signal number = 1
child process get SIGHUP signal
child process is hung up

fork process(pid=3898) execute normally
[10/11/20]seed@VM:~/.../bonus$
```

2) only one test file

```
[10/11/20]seed@VM:~/.../bonus$ ./myfork hangup
------------CHILD PROCESS START------------
This is the SIGHUP program

the process tree : 3911->3912
The child process(pid=3912) of parent process(pid=3911) is terminated by signal
Its signal number = 1
child process get SIGHUP signal
child process is hung up

fork process(pid=3911) execute normally
[10/11/20]seed@VM:~/.../bonus$
```

3) random test

```
[10/11/20]seed@VM:~/.../bonus$ ./myfork terminate kill normal1 bus
------------CHILD PROCESS START------------
This is the SIGBUS program

This is normal1 program
------------CHILD PROCESS START------------
This is the SIGKILL program

------------CHILD PROCESS START------------
This is the SIGTERM program

the process tree : 3942->3943->3944->3945->3946
The child process(pid=3946) of parent process(pid=3945) is terminated by signal
Its signal number = 7
child process get SIGBUS signal
child process access to an undefined portion of a memory object

The child process(pid=3945) of parent process(pid=3944) has normal execution
Its exit status is 0

The child process(pid=3944) of parent process(pid=3943) is terminated by signal
Its signal number = 9
child process get SIGKILL signal
child process terminate immediately

The child process(pid=3943) of parent process(pid=3942) is terminated by signal
Its signal number = 15
child process get SIGTERM signal
child process reach a break point

fork process(pid=3942) execute normally
[10/11/20]seed@VM:~/.../bonus$
```

5. What I learned:

1) I learned how to use recursion to create parent process and child process.

2) I had a deeper understanding of each parameter in fork() function and exec() function as well as wait() function.

3) I learned that child process and parent process do not share the same memory space. To fix that, we should use mmap function to malloc a space for the global variable we need.