CSC3150 Assignment 1 (120090460, 朱骆锴)

1. Program 1

a) Basic idea of the program 1

This program is asked to fork a new process whose parent process is just the running program 1. In this process, the program runs under user mode. Then the program will call some system calls and access into the kernel mode. After the execution which is in the kernel mode has finished, the computer system sends the signals back to the user mode. The main program forks a test child process and wait for the return signal from the child program. Eventually, the parent program prints out the execution information.

b) Implementation

The main program uses a function **fork()** to create a child process which starts processing from the next line of the **fork()** function. The return value of the function is 0 for child process while the return value of the function is larger than 0 for parent process. Thus, it is easy to distinguish the two processes. In the child branch, the program execute an executable file by function **execve()** while in the parent branch, the program waits for the signal of the child process by function **waitpid()**.

c) Development Environment

i. Operating System: Ubuntu 16.04.7 LTS

ii. Linux Kernel: 5.10.5

iii. Gcc: Ubuntu 5.4.0-6ubuntu1~16.04.12

d) How to execute the program

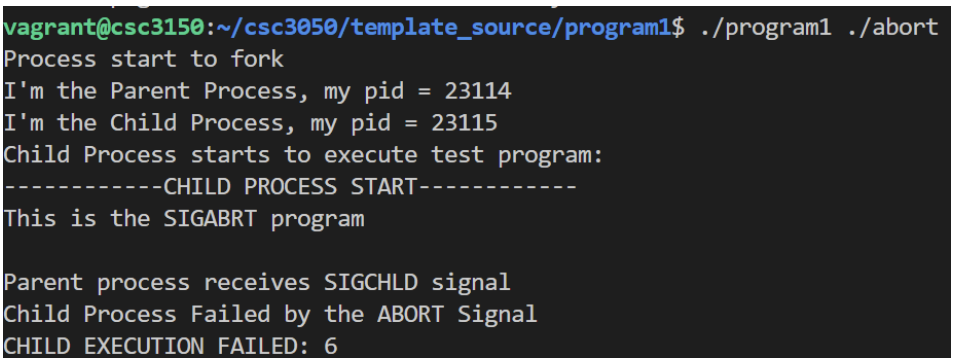
i. Change directory to the folder of the program 1.

ii. Type “sudo make” in the terminal to make c files into executable files.

iii. In the terminal, type “./program1 ./testProgramName”. For example, if you want to test how the child is aborted. The “./testProgramName” above is “./abort”

e) Program Output

i. ./program1 ./abort



ii. ./program1 ./alarm

```
vagrant@csc3150:~/csc3050/template_source/program1$ ./program1 ./alarm
Process start to fork
I'm the Parent Process, my pid = 23169
I'm the Child Process, my pid = 23170
Child Process starts to execute test program:
------------CHILD PROCESS START------------
This is the SIGALRM program

Parent process receives SIGCHLD signal
Child Process Failed by the ALARM Signal
CHILD EXECUTION FAILED: 14
```

iii. ./program1 ./bus

```
vagrant@csc3150:~/csc3050/template_source/program1$ ./program1 ./bus
Process start to fork
I'm the Child Process, my pid = 23212
I'm the Parent Process, my pid = 23211
Child Process starts to execute test program:
------------CHILD PROCESS START------------
This is the SIGBUS program

Parent process receives SIGCHLD signal
Child Process Failed by the SIGBUS Signal
CHILD EXECUTION FAILED: 7
```

iv. ./program1 ./floating

```
vagrant@csc3150:~/csc3050/template_source/program1$ ./program1 ./floating
Process start to fork
I'm the Parent Process, my pid = 23252
I'm the Child Process, my pid = 23253
Child Process starts to execute test program:
------------CHILD PROCESS START------------
This is the SIGFPE program

Parent process receives SIGCHLD signal
Child Process Failed by the SIGFPE Signal
CHILD EXECUTION FAILED: 8
```

v. ./program1 ./hangup

```
vagrant@csc3150:~/csc3050/template_source/program1$ ./program1 ./hangup
Process start to fork
I'm the Parent Process, my pid = 23373
I'm the Child Process, my pid = 23374
Child Process starts to execute test program:
------------CHILD PROCESS START------------
This is the SIGHUP program

Parent process receives SIGCHLD signal
Child Process Failed by the SIGHUP Signal
CHILD EXECUTION FAILED: 1
```

vi. ./program1 ./illegal_instr

```
vagrant@csc3150:~/csc3050/template_source/program1$ ./program1 ./illegal_instr
Process start to fork
I'm the Child Process, my pid = 23424
I'm the Parent Process, my pid = 23423
Child Process starts to execute test program:
------------CHILD PROCESS START------------
This is the SIGILL program

Parent process receives SIGCHLD signal
Child Process Failed by the SIGILL Signal
CHILD EXECUTION FAILED: 4
```

vii. ./program1 ./interrupt

```
vagrant@csc3150:~/csc3050/template_source/program1$ ./program1 ./interrupt
Process start to fork
I'm the Parent Process, my pid = 23468
I'm the Child Process, my pid = 23469
Child Process starts to execute test program:
------------CHILD PROCESS START------------
This is the SIGINT program

Parent process receives SIGCHLD signal
Child Process Failed by the SIGHUP Signal
CHILD EXECUTION FAILED: 2
```

viii. ./program1 ./kill

```
vagrant@csc3150:~/csc3050/template_source/program1$ ./program1 ./kill
Process start to fork
I'm the Parent Process, my pid = 23543
I'm the Child Process, my pid = 23544
Child Process starts to execute test program:
------------CHILD PROCESS START------------
This is the SIGKILL program

Parent process receives SIGCHLD signal
Child Process Failed by the SIGKILL Signal
CHILD EXECUTION FAILED: 9
```

ix. ./program1 ./normal

```
vagrant@csc3150:~/csc3050/template_source/program1$ ./program1 ./normal
Process start to fork
I'm the Parent Process, my pid = 23583
I'm the Child Process, my pid = 23584
Child Process starts 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
```

x. ./program1 ./pipe

```
vagrant@csc3150:~/csc3050/template_source/program1$ ./program1 ./pipe
Process start to fork
I'm the Parent Process, my pid = 23622
I'm the Child Process, my pid = 23623
Child Process starts to execute test program:
------------CHILD PROCESS START------------
This is the SIGPIPE program

Parent process receives SIGCHLD signal
Child Process Failed by the SIGPIPE Signal
CHILD EXECUTION FAILED: 13
```

xi. ./program1 ./quit

```
vagrant@csc3150:~/csc3050/template_source/program1$ ./program1 ./quit
Process start to fork
I'm the Parent Process, my pid = 23670
I'm the Child Process, my pid = 23671
Child Process starts to execute test program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

Parent process receives SIGCHLD signal
Child Process Failed by the SIGQUIT Signal
CHILD EXECUTION FAILED: 3
```

xii. ./program1 ./segment_fault

```
vagrant@csc3150:~/csc3050/template_source/program1$ ./program1 ./segment_fault
Process start to fork
I'm the Parent Process, my pid = 23726
I'm the Child Process, my pid = 23727
Child Process starts to execute test program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

Parent process receives SIGCHLD signal
Child Process Failed by the SIGSEGV Signal
CHILD EXECUTION FAILED: 11
```

xiii. ./program1 ./stop

```
vagrant@csc3150:~/csc3050/template_source/program1$ ./program1 ./stop
Process start to fork
I'm the Parent Process, my pid = 23776
I'm the Child Process, my pid = 23777
Child Process starts to execute test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

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

xiv. ./program1 ./terminate

```
vagrant@csc3150:~/csc3050/template_source/program1$ ./program1 ./terminate
Process start to fork
I'm the Parent Process, my pid = 23807
I'm the Child Process, my pid = 23808
Child Process starts to execute test program:
------------CHILD PROCESS START------------
This is the SIGTERM program

Parent process receives SIGCHLD signal
Child Process Failed by the SIGTERM Signal
CHILD EXECUTION FAILED: 15
```

xv. ./program1 ./trap

```
vagrant@csc3150:~/csc3050/template_source/program1$ ./program1 ./trap
Process start to fork
I'm the Parent Process, my pid = 23844
I'm the Child Process, my pid = 23845
Child Process starts to execute test program:
------------CHILD PROCESS START------------
This is the SIGTRAP program

Parent process receives SIGCHLD signal
Child Process Failed by the SIGTRAP Signal
CHILD EXECUTION FAILED: 5
```

f) What I have learned

I learned how to fork a child process through the funcrtion fork(), The next one I learned is the execve function which allows child process to execute another binary file. Finally, I learned how the operating system wait the option of the child process to get signal.

2. Program 2

a) Basic idea of program 2

The basic idea of program 2 is to create a kernel thread in the kernel mode. In this thread, it will fork a child process and run a testbench in the child process. Finally, the child process will return a signal of the execution result of the testbench.

b) Implementation

i. The kernel module initializes through the function named **"program2_init"**. In the function, it creates a kernel thread through the Linux Function **"Kthread_create"** to run **"my_fork"** function.

ii. The **"my_fork"** function will fork a child process through the function **"kernel_clone"**. Also it also waits for the return signal from the executed file.

The argument of the kernel_clone is set like this

```
struct kernel_clone_args args = {.flags = SIGCHLD,
                                 .exit_signal = SIGCHLD,
                                 .stack = &my_execve,
                                 .stack_size = 0,
                                 .parent_tid = NULL,
                                 .child_tid = NULL};
```

iii. Function **"kernel_clone"** will call the function **"my_execve"** where the Linux function **"do_execve"** is in. **"do_execve"** will run the testbench by getting the name of running binary

files.

iv. Eventually, in the function “my_wait()”, Linux function “do_wait() is called to get the return signal with arguments listed below.

```
wo.wo_type = type;
wo.wo_pid = wo_pid;
wo.wo_flags = WEXITED | WUNTRACED;
wo.wo_info = NULL;
wo.wo_stat = status;
wo.wo_rusage = NULL;

int waitValue;
waitValue = do_wait(&wo);
```

c) Development Environment

i. Operating System: Ubuntu 16.04.7 LTS

ii. Linux Kernel: 5.10.5

iii. GCC: Ubuntu 5.4.0-6ubuntu1~16.04.12

d) How to execute my program

i. The first step is to add some settings to the kernel file so that the function in the kernel module can be called by the loadable kernel module. We find the corresponding file positions where the five called function “kernel_clone()”, “do_wait()”, “do_execve()”, “kthread_create()”, “getname_kernel()” are in. After that, I add the EXPORT_SYMBOL(function name).

Then we just use keyword **extern** to import these functions.

ii. To compile the kernel, I download the kernel and decompress the file. And then I make a config to let the operating system knows how to make the kernel.

iii. The command to compile the kernel is listed below in sequence.

1. cd ~/home/seed/work/linux-5.10.5
2. sudo su
3. make bzImage
4. make modules
5. make modules_install
6. make install

iv. Change the directory to the folder where the program is located and start to make the program2.c

1. make
2. gcc -o test test.c

v. Using “rmmod program2.ko” to remove the previous module.

vi. Using “insmod program2.ko” to insert the new module that I just compiled

vii. Using “dmesg” to observe the executing result.

e) Program 2 Output

i. test

```
[167352.553962] [program2] : Module_init Zhu_Luokai 120090460
[167352.553964] [program2] : Module_init create kthread start
[167352.554449] [program2] : The child process has pid = 25274
[167352.554450] [program2] : The parent process has pid = 25273
[167352.555061] [program2] : child process
[167352.654243] [Program2] : The return signal is 7
[167352.654244] [program2] : Child Process gets SIGBUS signal
[167352.654245] [program2] : Child Process gets bus error
[167413.210946] [program2] : Module_exit
```

ii. normal

```
[167894.268060] [program2] : Module_init Zhu_Luokai 120090460
[167894.268062] [program2] : Module_init create kthread start
[167894.268402] [program2] : The child process has pid = 26832
[167894.268403] [program2] : The parent process has pid = 26831
[167894.268479] [program2] : child process
[167894.270789] [Program2] : The return signal is 0
[167894.270791] [program2] : Child Process terminated normally
[168282.278117] [program2] : Module_exit
```

iii. stop

```
[167523.562385] [program2] : Module_init Zhu_Luokai 120090460
[167523.562386] [program2] : Module_init create kthread start
[167523.562782] [program2] : The child process has pid = 26320
[167523.562783] [program2] : The parent process has pid = 26319
[167523.562969] [program2] : child process
[167523.566002] [Program2] : The return signal is 19
[167523.566003] [program2] : Child Process STOPS
[167523.566004] [program2] : Child Process gets SIGSTOP signal
[167530.631997] [program2] : Module_exit
```