Assignment1 Report

Zhang Qiyue 120090391

- Program1
  - Q1 How did you design your program?

    A:

    This question asked to fork a child process in user mode. Parent process should wait until receive status signal from child process and then print what kind of the signal raised in child process as well as the exit status.

    In order to fork a child process, we can use fork() function to realized, then use getpid() and getppid() to get the PID of the calling system and its parent system respectively.

    In order to execute a file during child process, we can use execve() function to realize.

    Due to the SIGSTOP signal, we should use waitpid() with WUNTRACED as option instead of wait() to prevent the process from actually stopping.

    In order to know which signal was raised in child process, we should read their status. First, we can use WIFEXITED() to judge whether the process terminated normally, WIFSIGNALED() to know whether the process terminated because of signals, and WIFSTOPPED() to tell whether the process was in STOP status. Depend on the condition, we can use WEXITSTATUS(), WTERMSIG() and WSTOPSIG() to get the status value respectively.

  - Q2 How to set up your development environment, including how to compile kernel?

    A:

    In order to protect our physical computer, we should modify the kernel on the virtual machine. So, we download the VM and can remote connect to the VM through Remote-SSH of VS Code. After finishing it, we create a folder called csc3150 as the root of this course and the whole path of it is /home/vagrant/csc3150.

    Besides, due to the different kernel version, we also need to update the kernel. We should download the suitable version of kernel and unzip it within the root directory of the virtual machine. After we cleaning up the previous setting, we begin building and installing the new kernel and modules. When we reboot the virtual machine, if the kernel version changed to the new one, we set up the development environment successfully.

  - Q3 Screenshot of your program output.

A:

```
vagrant@csc3150:~/csc3150/Assignment_1_120090391/source/program1$ ./program1 ./normal
Process start to fork
I'm the Parent Process, my pid = 2408
I'm the Child Process, my pid = 2409
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
```

```
vagrant@csc3150:~/csc3150/Assignment_1_120090391/source/program1$ ./program1 ./abort
Process start to fork
I'm the Parent Process, my pid = 2032
I'm the Child Process, my pid = 2033
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGABRT program

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

```
vagrant@csc3150:~/csc3150/Assignment_1_120090391/source/program1$ ./program1 ./alarm
Process start to fork
I'm the Parent Process, my pid = 2102
I'm the Child Process, my pid = 2103
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGALRM program

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

```
vagrant@csc3150:~/csc3150/Assignment_1_120090391/source/program1$ ./program1 ./bus
Process start to fork
I'm the Parent Process, my pid = 2172
I'm the Child Process, my pid = 2173
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGBUS program

Parent process receives SIGCHLD signal
child process raise SIGBUS signal
```

```
vagrant@csc3150:~/csc3150/Assignment_1_120090391/source/program1$ ./program1 ./floating
Process start to fork
I'm the Parent Process, my pid = 2235
I'm the Child Process, my pid = 2236
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGFPE program

Parent process receives SIGCHLD signal
child process raise SIGFPE signal
```

```
vagrant@csc3150:~/csc3150/Assignment_1_120090391/source/program1$ ./program1 ./hangup
Process start to fork
I'm the Parent Process, my pid = 2277
I'm the Child Process, my pid = 2278
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGHUP program

Parent process receives SIGCHLD signal
child process raise SIGHUP signal
```

```
vagrant@csc3150:~/csc3150/Assignment_1_120090391/source/program1$ ./program1 ./illegal_instr
Process start to fork
I'm the Parent Process, my pid = 2316
I'm the Child Process, my pid = 2317
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGILL program

Parent process receives SIGCHLD signal
child process raise SIGILL signal
```

```
vagrant@csc3150:~/csc3150/Assignment_1_120090391/source/program1$ ./program1 ./interrupt
Process start to fork
I'm the Child Process, my pid = 2356
Child process start to execute test program:
I'm the Parent Process, my pid = 2355
------------CHILD PROCESS START------------
This is the SIGINT program

Parent process receives SIGCHLD signal
child process raise SIGINT signal
```

```
vagrant@csc3150:~/csc3150/Assignment_1_120090391/source/program1$ ./program1 ./kill
Process start to fork
I'm the Parent Process, my pid = 2381
I'm the Child Process, my pid = 2382
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGKILL program

Parent process receives SIGCHLD signal
child process raise SIGKILL signal
```

```
vagrant@csc3150:~/csc3150/Assignment_1_120090391/source/program1$ ./program1 ./pipe
Process start to fork
I'm the Parent Process, my pid = 2448
I'm the Child Process, my pid = 2449
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGPIPE program

Parent process receives SIGCHLD signal
child process raise SIGPIPE signal
```

```
nvagrant@csc3150:~/csc3150/Assignment_1_120090391/source/program1$ ./program1 ./quit
Process start to fork
I'm the Parent Process, my pid = 2510
I'm the Child Process, my pid = 2511
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

Parent process receives SIGCHLD signal
child process raise SIGQUIT signal
```

```
vagrant@csc3150:~/csc3150/Assignment_1_120090391/source/program1$ ./program1 ./segment_fault
Process start to fork
I'm the Parent Process, my pid = 2561
I'm the Child Process, my pid = 2562
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

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

```
vagrant@csc3150:~/csc3150/Assignment_1_120090391/source/program1$ ./program1 ./stop
Process start to fork
I'm the Parent Process, my pid = 2778
I'm the Child Process, my pid = 2779
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

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

```
vagrant@csc3150:~/csc3150/Assignment_1_120090391/source/program1$ ./program1 ./terminate
Process start to fork
I'm the Parent Process, my pid = 2707
I'm the Child Process, my pid = 2708
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTERM program

Parent process receives SIGCHLD signal
child process raise SIGTERM signal
```

```
vagrant@csc3150:~/csc3150/Assignment_1_120090391/source/program1$ ./program1 ./trap
Process start to fork
I'm the Child Process, my pid = 2750
Child process start to execute test program:
I'm the Parent Process, my pid = 2749
------------CHILD PROCESS START------------
This is the SIGTRAP program

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

- Q4 What did you learn from this task?

  A:

  First of all, I learnt how to compile a new version kernel in my virtual machine. In addition, I get more familiar with Linux operation. What's more, through adjusting the order of the code and using the sleep() function, I get better understanding about the output order of parent process and child process. Through using some WIF...() and W...() functions, I learnt how to check the reason why a child process terminated and how to get its exit status value which will be helpful for solving task2.

Program2

- Q1 How did you design your program?

  A:

The question required to initialize a kernel modules and create a kernel thread to fork a child process. The latter part is quiet similar to task1, but we should do it in kernel mode instead of the user mode.

Above all, using kthread_create() function to create the kernel thread and designed my_fork() function to fork the child process. Also, the address of my_fork() should be set as the parameter of kthread_create().

Within my_fork() function, using kernel_clone() function to fork the child process. The argument of the kernel_clone() is a structure. Its flag and exit_signal should be SIGCHLD to let parent process know when the child process terminated and its stack should be my_exec() to execute the test.c file. Within my_exec() function, using getname_kernel() to get the file name in the specified path and do_execve() to execute the specified file. Finally, designing my_wait() function to make parent process wait until child process terminated.

Within my_wait() function, initialize the element in structure wait_opts. The special note is that the flag should be WEXITED | WUNTRACED which can consider both SIGSTOP and other signals. Next, use do_wait() to wait and the value of wait_opts will be changed. According to the value of status, we can know which signal was raised in child process and output the corresponding information.

- Q2 How to set up your development environment, including how to compile kernel?

  A:

  Since we use some function in linux kernel, so will should tag these four functions as extern and also add "EXPORT_SYMBOL" in the corresponding file of the source code of linux. After finishing it, we should recompile the kernel as we had done before (but do not do "make clean").

- Q3 Screenshot of your program output.

  A:

```
[ 4146.337904] [program2] : Module_init {Zhang Qiyue} {120090391}
[ 4146.337906] [program2] : Module_init create kthread start
[ 4146.338404] [program2] : Module_init kthread start
[ 4146.338662] [program2] : The child process has pid = 6646
[ 4146.338663] [program2] : This is the parent process, pid = 6645
[ 4146.338800] [program2] : child process
[ 4146.339737] [program2] : child process exit normally
[ 4146.339739] [program2] : The return signal is 0
[ 4158.473908] [program2] : Module_exit
```

```
[ 4502.897637] [program2] : Module_init {Zhang Qiyue} {120090391}
[ 4502.897641] [program2] : Module_init create kthread start
[ 4502.898308] [program2] : Module_init kthread start
[ 4502.898803] [program2] : The child process has pid = 7095
[ 4502.898805] [program2] : This is the parent process, pid = 7094
[ 4502.899082] [program2] : child process
[ 4503.169429] [program2] : get SIGABRT signal
[ 4503.169431] [program2] : child process abort
[ 4503.169432] [program2] : The return signal is 6
[ 4505.710135] [program2] : Module_exit
```

```
[ 4527.662208] [program2] : Module_init {Zhang Qiyue} {120090391}
[ 4527.662211] [program2] : Module_init create kthread start
[ 4527.662854] [program2] : Module_init kthread start
[ 4527.663103] [program2] : The child process has pid = 7450
[ 4527.663105] [program2] : This is the parent process, pid = 7449
[ 4527.663108] [program2] : child process
[ 4527.666379] [program2] : get SIGALRM signal
[ 4527.666380] [program2] : child process timer error
[ 4527.666381] [program2] : The return signal is 14
[ 4529.448528] [program2] : Module_exit
```

```
[ 4549.611838] [program2] : Module_init {Zhang Qiyue} {120090391}
[ 4549.611840] [program2] : Module_init create kthread start
[ 4549.612184] [program2] : Module_init kthread start
[ 4549.620550] [program2] : The child process has pid = 7792
[ 4549.620553] [program2] : This is the parent process, pid = 7790
[ 4549.621181] [program2] : child process
[ 4549.803199] [program2] : get SIGBUS signal
[ 4549.803201] [program2] : bus error
[ 4549.803202] [program2] : The return signal is 7
[ 4551.750616] [program2] : Module_exit
```

```
[ 4572.161374] [program2] : Module_init {Zhang Qiyue} {120090391}
[ 4572.161377] [program2] : Module_init create kthread start
[ 4572.162466] [program2] : Module_init kthread start
[ 4572.163012] [program2] : The child process has pid = 8145
[ 4572.163014] [program2] : This is the parent process, pid = 8144
[ 4572.163411] [program2] : child process
[ 4572.370067] [program2] : get SIGFPE signal
[ 4572.370069] [program2] : floating point exception
[ 4572.370070] [program2] : The return signal is 8
[ 4574.463059] [program2] : Module_exit
```

```
[ 4706.495599] [program2] : Module_init {Zhang Qiyue} {120090391}
[ 4706.495601] [program2] : Module_init create kthread start
[ 4706.496028] [program2] : Module_init kthread start
[ 4706.496262] [program2] : The child process has pid = 8517
[ 4706.496264] [program2] : This is the parent process, pid = 8516
[ 4706.496267] [program2] : child process
[ 4706.497798] [program2] : get SIGHUP signal
[ 4706.497799] [program2] : child process hung up
[ 4706.497800] [program2] : The return signal is 1
[ 4708.789623] [program2] : Module_exit
```

```
[ 4727.317443] [program2] : Module_init {Zhang Qiyue} {120090391}
[ 4727.317446] [program2] : Module_init create kthread start
[ 4727.317860] [program2] : Module_init kthread start
[ 4727.318693] [program2] : The child process has pid = 8872
[ 4727.318694] [program2] : This is the parent process, pid = 8871
[ 4727.318843] [program2] : child process
[ 4727.477598] [program2] : get SIGILL signal
[ 4727.477601] [program2] : illegal instruction
[ 4727.477602] [program2] : The return signal is 4
[ 4729.151801] [program2] : Module_exit
```

```
[ 4748.207232] [program2] : Module_init {Zhang Qiyue} {120090391}
[ 4748.207234] [program2] : Module_init create kthread start
[ 4748.207758] [program2] : Module_init kthread start
[ 4748.207963] [program2] : The child process has pid = 9215
[ 4748.207964] [program2] : This is the parent process, pid = 9214
[ 4748.208039] [program2] : child process
[ 4748.211670] [program2] : get SIGINT signal
[ 4748.211673] [program2] : terminal interrupt
[ 4748.211674] [program2] : The return signal is 2
[ 4749.960758] [program2] : Module_exit
```

```
[ 4767.224065] [program2] : Module_init {Zhang Qiyue} {120090391}
[ 4767.224067] [program2] : Module_init create kthread start
[ 4767.224906] [program2] : Module_init kthread start
[ 4767.225339] [program2] : The child process has pid = 9570
[ 4767.225340] [program2] : This is the parent process, pid = 9568
[ 4767.226201] [program2] : child process
[ 4767.230336] [program2] : get SIGKILL signal
[ 4767.230337] [program2] : kill process
[ 4767.230338] [program2] : The return signal is 9
[ 4769.612000] [program2] : Module_exit
```

```
[ 5245.641595] [program2] : Module_init {Zhang Qiyue} {120090391}
[ 5245.641597] [program2] : Module_init create kthread start
[ 5245.642005] [program2] : Module_init kthread start
[ 5245.642245] [program2] : The child process has pid = 9939
[ 5245.642247] [program2] : This is the parent process, pid = 9938
[ 5245.642492] [program2] : child process
[ 5245.644048] [program2] : get SIGPIPE signal
[ 5245.644051] [program2] : broken pipe
[ 5245.644052] [program2] : The return signal is 13
[ 5247.870548] [program2] : Module_exit
```

```
[ 5266.852540] [program2] : Module_init {Zhang Qiyue} {120090391}
[ 5266.852542] [program2] : Module_init create kthread start
[ 5266.853141] [program2] : Module_init kthread start
[ 5266.853474] [program2] : The child process has pid = 10292
[ 5266.853476] [program2] : This is the parent process, pid = 10291
[ 5266.853866] [program2] : child process
[ 5267.064045] [program2] : get SIGQUIT signal
[ 5267.064048] [program2] : terminal quit
[ 5267.064049] [program2] : The return signal is 3
[ 5268.766870] [program2] : Module_exit
```

```
[ 5296.702086] [program2] : Module_init {Zhang Qiyue} {120090391}
[ 5296.702089] [program2] : Module_init create kthread start
[ 5296.702883] [program2] : Module_init kthread start
[ 5296.703195] [program2] : The child process has pid = 10682
[ 5296.703196] [program2] : This is the parent process, pid = 10681
[ 5296.703348] [program2] : child process
[ 5296.863548] [program2] : get SIGSEGV signal
[ 5296.863550] [program2] : invalid memory reference
[ 5296.863551] [program2] : The return signal is 11
[ 5299.553112] [program2] : Module_exit
```

```
[ 5322.200344] [program2] : Module_init {Zhang Qiyue} {120090391}
[ 5322.200347] [program2] : Module_init create kthread start
[ 5322.200901] [program2] : Module_init kthread start
[ 5322.201159] [program2] : The child process has pid = 11044
[ 5322.201161] [program2] : This is the parent process, pid = 11043
[ 5322.201328] [program2] : child process
[ 5322.202738] [program2] : get SIGSTOP signal
[ 5322.202740] [program2] : child process stopped
[ 5322.202741] [program2] : The return signal is 19
[ 5324.817758] [program2] : Module_exit
```

```
[ 5344.543175] [program2] : Module_init {Zhang Qiyue} {120090391}
[ 5344.543177] [program2] : Module_init create kthread start
[ 5344.543758] [program2] : Module_init kthread start
[ 5344.544277] [program2] : The child process has pid = 11408
[ 5344.544279] [program2] : This is the parent process, pid = 11407
[ 5344.544283] [program2] : child process
[ 5344.545298] [program2] : get SIGTERM signal
[ 5344.545301] [program2] : child process terminated
[ 5344.545302] [program2] : The return signal is 15
[ 5346.954182] [program2] : Module_exit
```

```
[ 5373.085413] [program2] : Module_init {Zhang Qiyue} {120090391}
[ 5373.085416] [program2] : Module_init create kthread start
[ 5373.086506] [program2] : Module_init kthread start
[ 5373.088306] [program2] : The child process has pid = 11775
[ 5373.088308] [program2] : This is the parent process, pid = 11773
[ 5373.088314] [program2] : child process
[ 5373.274151] [program2] : get SIGTRAP signal
[ 5373.274153] [program2] : trap error
[ 5373.274155] [program2] : The return signal is 5
[ 5376.281980] [program2] : Module_exit
```

- Q4 What did you learn from this task?

  A:

  I learned how to use some external functions and how to print something in kernel mode. I also remembered different signals and their corresponding exit value. The most important part that I learned from this task is how to do things in kernel mode. Comparing this task to task1, I can understand the difference and relationship of kernel mode and user mode better.