Luo Hao 120090712

Part I. How did I design the program.

For program1: In this task, I need to fork a child process to execute the test program and meanwhile, the parent process should wait until the child process terminates. As the test program will send terminate signal to the child process to terminate the process using various signals, I need to print out the signal raised in the test program. So, I can conquer the task int some small tasks.

First, I need to fork a child process. I used “fork()” function to create a new process. According to the definition of the fork function, both parent and child process will execve from the next line of this function. So, for each process, I used their different return valve of fork function to distinguish the two different process and used “if” statement to let the two processes do different execution. Both of them will print out their pids on the screen by using “getpid()” function. For the child process, I used “execve()” function to execute the test file. If the test file were executed, the process will terminate and parent process will get the signal. Then, to make sure the test file can be executed correctly, I write some function which can report error after the “execve()” function. If the test file were not been executed, I can get the error message. For the parent process, it should wait for the termination of child process, so I used “waitpid()” function which can pause the child process until the child process terminates and store the status of the child process. Now we can get the information of child process using the status. To get the signal, I used some functions related to the signal processing to detect what number the signal is. At last, the program needs to print out the message according the signal of the child process. I used a switch statement to print out the specified content. This is how I design the program1.

For program2: In this task, I need to do similar thing to task1 but in the kernel mode. According to the requirement. First, I need to create a kernel thread to run the program. I used “kthread_create()” function to create a thread and use “wake_up_process()” to wake up the thread. In the initialization of the thread, the “my_fork()” function will be executed after the thread is waked up. To the design the “my_fork” function, I need to implement the “kernel_clone()” function. So, I set up the arguments of the “kernel_clone_args” structure and fork a child process. But different with the task1, the child process will execute the function I wrote in the “kernel_clone_args” structure, which is “my_exec()” and the parent process will keep on running the rest of the program. Both of the parent and child processes will print out their pids. Now, I need to design two functions, one the “my_exec()”, and another is the “my_wait()” function because the parent process should wait the child process. For the “my_exec()” function, I used “do_execve()” function to execute the test file. As this function need special arguments, I used “getname_kernel()” to get the required argument. If the “do_execve()” returns 0, the test file is executed correctly. Furthermore, the parent process needs to wait the child process and get the signal of it. So, I design a function “my_wait()” for the parent process to wait the child process. In the “my_wait()” I need to call the “do_wait()” function. This function requires some settings, so I initialized the arguments in the function and change some of them to my sure the parent process can receive the signal. After that, the parent process executes this function, it will pause until the child process terminates. Similar to the task1, when the parent process gets the signal, it should be able to figure out what signal it is. So, inside this function, the parent process will use the wo_stat variable to get the signal of the child process and print out which signal the child process sent. This is how I design the program2.

Part II. How I set up the development environment

After installed the virtual machine, I need to set up the development environment to run the program. First, I changed the default settings of the virtual machine such as the size of the memory and the number of the processor to prepare for the compilation of the Linux kernel. I enlarge the memory for the virtual machine and increase the number of the processors because the compilation requires a lot of resources. Then I started to compile kernel. First, I need to download the required version of the kernel source file and decompress it. Then, I used “make mrproper” and “make clean” to clean the previous setting. I copied the config file from the old version of the kernel. After that, I used the command ”make menuconfig” to load and save the configuration. Later, I used the command”make bzIamge -j$(nproc)” and “make modules -j$(nproc)” to compile the kernel. At last, I could use the commands”make modules_install”, “make install”, “reboot” to finish the initial installation of the Linux kernel. Because in the Linux kernel, some functions are not static, which means I cannot directly use some functions in my program. If I want to run the program2, I first need to use the function “EXPORT_SYMBOL()” to make the functions be achievable. I found four functions in the Linux source code and change the source code by adding this function. These four functions are “getname_kernel()”, “kernel_clone()”, “do_wait()”, “do_execve()”. After changing the source code, I need to recompile the kernel. But this time, I don’t need to start from the scratch. I can start from the command of “make bzIamge -j$(nproc)”. Then after redo the previous step, I can prepare the new kernel environment for the program to be executed. What’s more, at first, my visual code cannot open the file in the virtual machine. So, I changed the folder permission using command: “chmod”. This is how I set up the development environment.

Part III. Screenshot of my program output

Task1:

Abort:

```
Process strat to fork
I'm the Parent Process, my pid = 2461
I'm the Child Process, my pid = 2462
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGABRT program

Parent process receives SIGCHLD signal
Child process get SIGABRT siganl
```

Alarm:

```
Process strat to fork
I'm the Parent Process, my pid = 2571
I'm the Child Process, my pid = 2572
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGALRM program

Parent process receives SIGCHLD signal
Child process get SIGALRM siganl
```

Bus:

```
Process strat to fork
I'm the Parent Process, my pid = 2654
I'm the Child Process, my pid = 2655
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGBUS program

Parent process receives SIGCHLD signal
Child process get SIGBUS siganl
```

Floating:

```
Process strat to fork
I'm the Parent Process, my pid = 2752
I'm the Child Process, my pid = 2753
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGFPE program

Parent process receives SIGCHLD signal
Child process get SIGFPE siganl
```

Hangup:

```
Process strat to fork
I'm the Parent Process, my pid = 2800
I'm the Child Process, my pid = 2801
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGHUP program

Parent process receives SIGCHLD signal
child process get SIGHUP siganl
```

Illegal_instr:

```
Process strat to fork
I'm the Parent Process, my pid = 2961
I'm the Child Process, my pid = 2962
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGILL program

Parent process receives SIGCHLD signal
Child process get SIGILL siganl
```

Interrupt:

```
Process strat to fork
I'm the Parent Process, my pid = 3007
I'm the Child Process, my pid = 3009
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGINT program

Parent process receives SIGCHLD signal
Child process get SIGINT sigal
```

Kill:

```
Process strat to fork
I'm the Parent Process, my pid = 3080
I'm the Child Process, my pid = 3081
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGKILL program

Parent process receives SIGCHLD signal
Child process get SIGKILL siganl
```

Normal:

```
Process strat to fork
I'm the Parent Process, my pid = 3147
I'm the Child Process, my pid = 3148
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
```

Pipe:

```
Process strat to fork
I'm the Parent Process, my pid = 3170
I'm the Child Process, my pid = 3171
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGPIPE program

Parent process receives SIGCHLD signal
Child process get SIGPIPE siganl
```

Quit:

```
Process strat 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 SIGQUIT program

Parent process receives SIGCHLD signal
Child process get SIGQUIT siganl
```

Segment_fault:

```
Process strat to fork
I'm the Parent Process, my pid = 3395
I'm the Child Process, my pid = 3396
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

Parent process receives SIGCHLD signal
Child process get SIGSEGV siganl
```

Stop:

```
Process strat to fork
I'm the Parent Process, my pid = 5071
I'm the Child Process, my pid = 5072
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

Parent process receives SIGCHLD signal
child process get SIGSTOP siganl
```

Terminate:

```
Process strat to fork
I'm the Parent Process, my pid = 3485
I'm the Child Process, my pid = 3486
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTERM program

Parent process receives SIGCHLD signal
Child process get SIGTERM siganl
```

Trap:

```
Process strat to fork
I'm the Child Process, my pid = 5139
Child process start to execute test program:
I'm the Parent Process, my pid = 5138
------------CHILD PROCESS START------------
This is the SIGTRAP program

Parent process receives SIGCHLD signal
Child process get SIGTRAP siganl
```

Task2:

Abort:

```
[40272.189003] [program2] : module_init {Luohao} {120090712}
[40272.189004] [program2] : module_init create kthread start
[40272.189100] [program2] : module_init kthread start
[40272.189237] [program2] : The Child pid has pid = 7012
[40272.189238] [program2] : This is the parent process, pid = 7011
[40272.189293] [program2] : child process
[40272.284938] [program2] : get SIGABRT signal
[40272.284940] [program2] : child process terminated
[40272.284941] [program2] : The return signal is 6
[40279.782626] [program2] : Module_exit
```

Alarm:

```
[40353.217038] [program2] : module_init {Luohao} {120090712}
[40353.217039] [program2] : module_init create kthread start
[40353.217573] [program2] : module_init kthread start
[40353.217688] [program2] : The Child pid has pid = 7748
[40353.217690] [program2] : This is the parent process, pid = 7747
[40353.217715] [program2] : child process
[40355.250754] [program2] : get SIGALRM signal
[40355.250757] [program2] : child process terminated
[40355.250759] [program2] : The return signal is 14
[40387.350784] [program2] : Module_exit
```

Bus:

```
[40505.391800] [program2] : module_init {Luohao} {120090712}
[40505.391802] [program2] : module_init create kthread start
[40505.391866] [program2] : module_init kthread start
[40505.392305] [program2] : The Child pid has pid = 8419
[40505.392307] [program2] : This is the parent process, pid = 8417
[40505.392322] [program2] : child process
[40505.502897] [program2] : get SIGBUS signal
[40505.502899] [program2] : child process terminated
[40505.502900] [program2] : The return signal is 7
[40512.367873] [program2] : Module_exit
```

Floating:

```
[40615.480760] [program2] : module_init {Luohao} {120090712}
[40615.480762] [program2] : module_init create kthread start
[40615.480977] [program2] : module_init kthread start
[40615.481320] [program2] : The Child pid has pid = 9772
[40615.481321] [program2] : This is the parent process, pid = 9770
[40615.481344] [program2] : child process
[40615.596038] [program2] : get SIGFPE signal
[40615.596040] [program2] : child process terminated
[40615.596040] [program2] : The return signal is 8
[40620.967034] [program2] : Module_exit
```

Hangup:

```
[40740.961088] [program2] : module_init {Luohao} {120090712}
[40740.961089] [program2] : module_init create kthread start
[40740.961326] [program2] : module_init kthread start
[40740.961443] [program2] : The Child pid has pid = 10440
[40740.961444] [program2] : This is the parent process, pid = 10439
[40740.961445] [program2] : child process
[40740.961687] [program2] : get SIGHUP signal
[40740.961687] [program2] : child process terminated
[40740.961688] [program2] : The return signal is 1
[40746.849652] [program2] : Module_exit
```

illegal_instr:

```
[40788.178602] [program2] : module_init {Luohao} {120090712}
[40788.178604] [program2] : module_init create kthread start
[40788.178792] [program2] : module_init kthread start
[40788.178829] [program2] : The Child pid has pid = 11105
[40788.178830] [program2] : This is the parent process, pid = 11104
[40788.178832] [program2] : child process
[40788.280081] [program2] : get SIGQILL signal
[40788.280083] [program2] : child process terminated
[40788.280084] [program2] : The return signal is 4
[40793.852131] [program2] : Module_exit
```

Interrupt:

```
[40858.152576] [program2] : module_init {Luohao} {120090712}
[40858.152578] [program2] : module_init create kthread start
[40858.153149] [program2] : module_init kthread start
[40858.153617] [program2] : The Child pid has pid = 11774
[40858.153618] [program2] : This is the parent process, pid = 11771
[40858.153631] [program2] : child process
[40858.153964] [program2] : get SIGINT signal
[40858.153965] [program2] : child process terminated
[40858.153965] [program2] : The return signal is 2
[40868.701551] [program2] : Module_exit
```

Kill:

```
[40858.152576] [program2] : module_init {Luohao} {120090712}
[40858.152578] [program2] : module_init create kthread start
[40858.153149] [program2] : module_init kthread start
[40858.153617] [program2] : The Child pid has pid = 11774
[40858.153618] [program2] : This is the parent process, pid = 11771
[40858.153631] [program2] : child process
[40858.153964] [program2] : get SIGINT signal
[40858.153965] [program2] : child process terminated
[40858.153965] [program2] : The return signal is 2
[40868.701551] [program2] : Module_exit
```

Normal:

```
[41276.182776] [program2] : module_init {Luohao} {120090712}
[41276.182778] [program2] : module_init create kthread start
[41276.182939] [program2] : module_init kthread start
[41276.183067] [program2] : The Child pid has pid = 12480
[41276.183068] [program2] : This is the parent process, pid = 12479
[41276.183071] [program2] : child process
[41276.183955] [program2] : get SIGCHLD signal
[41276.183957] [program2] : child process terminated
[41276.183958] [program2] : The return signal is 0
[41280.464779] [program2] : Module_exit
```

Pipe:

```
[41330.353401] [program2] : module_init {Luohao} {120090712}
[41330.353403] [program2] : module_init create kthread start
[41330.353457] [program2] : module_init kthread start
[41330.353481] [program2] : The Child pid has pid = 13142
[41330.353483] [program2] : This is the parent process, pid = 13141
[41330.353495] [program2] : child process
[41330.354135] [program2] : get SIGPIPE signal
[41330.354136] [program2] : child process terminated
[41330.354136] [program2] : The return signal is 13
[41334.786758] [program2] : Module_exit
```

Quit:

```
[41391.584102] [program2] : module_init {Luohao} {120090712}
[41391.584103] [program2] : module_init create kthread start
[41391.584249] [program2] : module_init kthread start
[41391.584277] [program2] : The Child pid has pid = 13784
[41391.584278] [program2] : This is the parent process, pid = 13783
[41391.584290] [program2] : child process
[41391.714008] [program2] : get SIGQUIT signal
[41391.714011] [program2] : child process terminated
[41391.714012] [program2] : The return signal is 3
[41408.553566] [program2] : Module_exit
```

Segment_fault:

```
[41445.864937] [program2] : module_init {Luohao} {120090712}
[41445.864940] [program2] : module_init create kthread start
[41445.865094] [program2] : module_init kthread start
[41445.865309] [program2] : The Child pid has pid = 14471
[41445.865310] [program2] : This is the parent process, pid = 14470
[41445.865326] [program2] : child process
[41445.979388] [program2] : get SIGSEGV signal
[41445.979390] [program2] : child process terminated
[41445.979391] [program2] : The return signal is 11
[41451.689438] [program2] : Module_exit
```

Stop:

```
[41451.689438] [program2] : Module_exit
[41561.494374] [program2] : module_init {Luohao} {120090712}
[41561.494376] [program2] : module_init create kthread start
[41561.494537] [program2] : module_init kthread start
[41561.494615] [program2] : The Child pid has pid = 15177
[41561.494616] [program2] : This is the parent process, pid = 15176
[41561.494618] [program2] : child process
[41561.494948] [program2] : get SIGSTOP signal
[41561.494949] [program2] : child process terminated
[41561.494949] [program2] : The return signal is 19
[41566.099772] [program2] : Module_exit
```

terminate:

```
[41605.058685] [program2] : module_init {Luohao} {120090712}
[41605.058687] [program2] : module_init create kthread start
[41605.058765] [program2] : module_init kthread start
[41605.058797] [program2] : The Child pid has pid = 15822
[41605.058798] [program2] : This is the parent process, pid = 15821
[41605.058815] [program2] : child process
[41605.060402] [program2] : get SIGTERM signal
[41605.060405] [program2] : child process terminated
[41605.060406] [program2] : The return signal is 15
[41610.174037] [program2] : Module_exit
```

Trap:

```
[41669.110031] [program2] : module_init {Luohao} {120090712}
[41669.110032] [program2] : module_init create kthread start
[41669.110145] [program2] : module_init kthread start
[41669.110165] [program2] : The Child pid has pid = 16488
[41669.110179] [program2] : This is the parent process, pid = 16487
[41669.110187] [program2] : child process
[41669.226934] [program2] : get SIGTRAP signal
[41669.226935] [program2] : child process terminated
[41669.226936] [program2] : The return signal is 5
[41675.061273] [program2] : Module_exit
```

Part IV. What I learnt from the task

After finishing this task, I really learnt a lot of things.

First, I got some knowledge of how the Linux kernel code was stored and how they were used when they were called. I read a lot of source code these days and I found that after compilation, some source are stored as .o type and so on. It means that when I change the content of the source code (such as do the EXPORT_SYMBOL), if I want my changes to the files work, I have to recompile it. Another thing I learnt is that the functions we use (no matter the user space or kernel space) are actually call some other functions and there are so many variables storing different information. For example, in the source code, I can see that execve() is actually related to do_execve() and do_execve() is related to sys_execve(). Also, when implementing some of the functions, I need to set many structures. Though it may be hard to fully understand all the meanings of the variables, it can help me make sense of how the information is sent and received. Furthermore, I get familiar with the functions I used in this task. The kernel mode and user mode are very different, the functions we used in the user mode is more like the packaging of programs in the kernel mode. Even though they are very similar, they have huge difference and the ways to use them are also different. The last thing I learnt is the importance of the pointer and address. Actually, most of the errors caused by my program are related to the address. If I enter an illegal address accidently, fatal problems will occur. So, I have learnt a lot though this task.