Assignment 1 Project Report

Li Jiaqi 120090727

Program Design

Task1: User mode linux programming

In this task, we're required to understand and use some linux api for forking a process, executing some pre-written programs, and receiving some unix-standard signals. This task is relatively easy once we understand some linux system call apis and they're provided with countless documentation and examples.

The `main()` function is the entry point of the main process, where we can use `fork` system call to spawn a child process. We can also use `execve` system call to execute a program. We use `waitpid` to let the main process wait for its child. My design strategy is illustrated in the figure below:

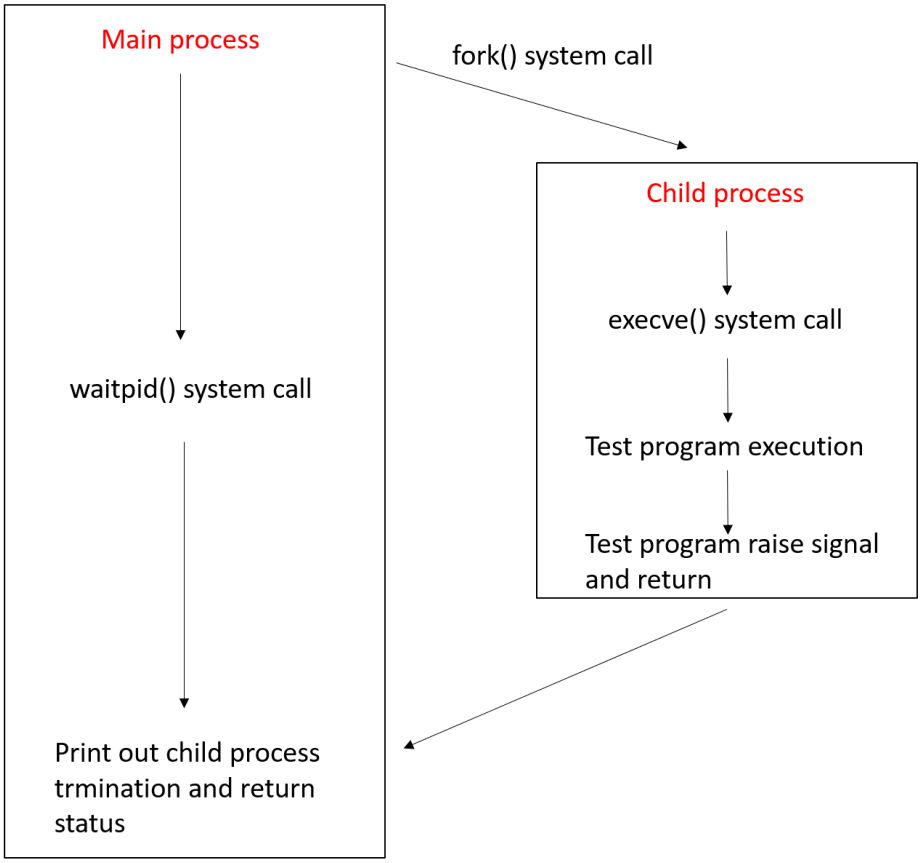

Figure 1. The program design of task 1

Task 2: Kernel mode linux programming

In task 2, we're required to implement programs which directly utilizes linux kernel functions and executes in kernel mode. Although the desired effect is similar to task 1, the implementation is rather difficult primariliy because of less documented kernel codes, and the difficulty in setting up the kernel environment.

In this task we implement a kernel module, whose entry point is a designated function declared by `module_init(program2_init);` Inside this function we first use `kthread_create` to create a kernel thread, passing a function named `my_fork`, and then use `wake_up_process` to wake it up. Note that the child process runs in kernel space, therefore is very delicate and could cause system failure if not well implemented. The `my_fork` function will execute now, inside which we first set up the default sigaction using the `k_sigaction` struct. Then we use `kernel_thread` to fork a process, passing another function named `my_exec` and `SIGCHLD` as its signal argument, which uses `getname_kernel` to get the target address of a program and use `do_execve` to execute it. The `kernel_thread` function is an abstraction of the `kernel_clone` function. After this we call the `my_wait` function to wait for the child process. This function is an abstraction of the kernel `do_wait` function. My design strategy is illustrated in the figure below:

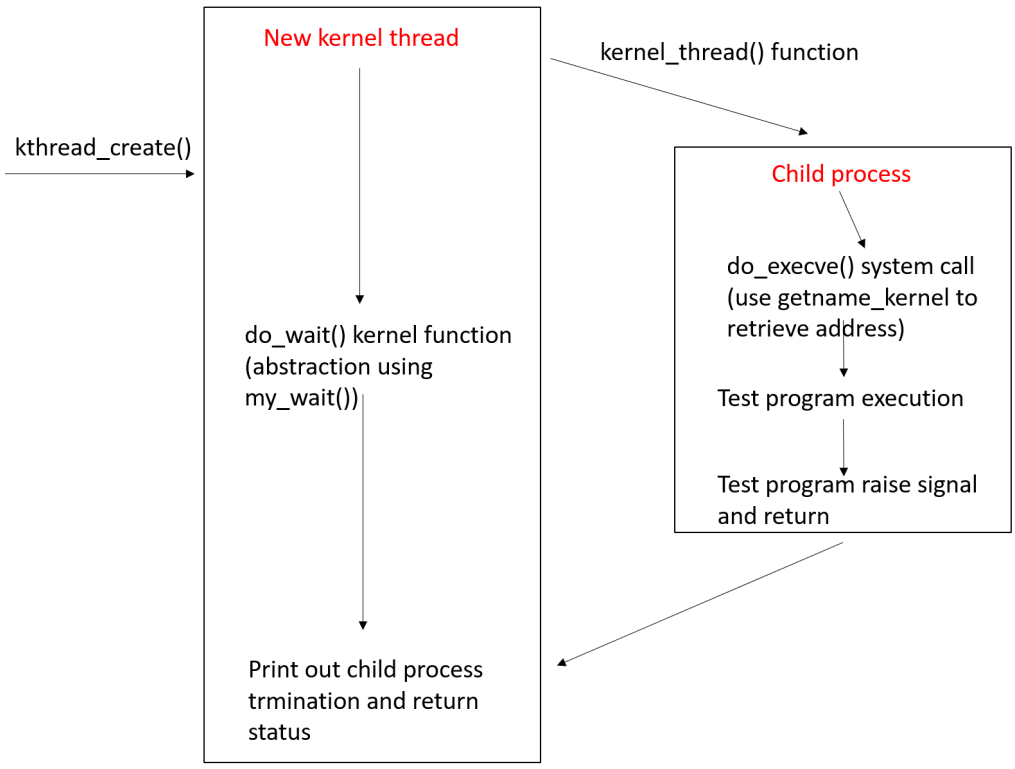

Figure2. The program design of task 2

Environment setup and kernel compilation

First, I installed a virtual machine of Ubuntu on VirtualBox Windows, whose kernel version is behind 5.10, which is the version requireed in our project. Therefore I need to compile the linux kernel 5.10 and install it in my virtual machine.

First, I download the 5.10.146 kernel source code from https://mirror.tuna.tsinghua.edu.cn/kernel/v5.x/ using `wget`. Then extract using `tar xvf $zip_file`. Then I installed the build tools on my VM using the command provided by TAs:

```
sudo apt-get install libncurses-dev gawk flex bison openssl libssl-dev dkms
libelf-dev libudev- dev libpci-dev libiberty-dev autoconf llvm
```

After this, I entered the root account and compile the kernel with the following commands provided by TAs:

```
# In root user mode, in the linux source root path
$make mrproper
$make clean
$make menuconfig
$make bzImage -j$(num_procs)
$make modules -j$(num_procs)
$make modules_install
$make install
$reboot
```

After rebooting, I checked that my kernel was successfully updated with `uname -r`

There are several symbols that I needed to manually export in the source code. To do this, I add `EXPORT_SYMBOL($target_function)` in the source code, then recompile and reboot. During the programming, I have used VSCode SSH remote to speed up.

Program Output

Task1

To compile: `make`

Running normal program:

```
vagrant@csc3150:~/csc3150as1/program1$ ./pro
gram1 ./normal
Process start to fork
I'm the Parent Process, my pid = 30132
I'm the Child Process, my pid = 30133
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
```

Abort:

```
vagrant@csc3150:~/csc3150as1/program1$ ./program1 ./abo
rt
Process start to fork
I'm the Parent Process, my pid = 30720
I'm the Child Process, my pid = 30721
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGABRT program

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

Bus:

```
vagrant@csc3150:~/csc3150as1/program1$ ./program1 ./bus
Process start to fork
I'm the Parent Process, my pid = 30759
I'm the Child Process, my pid = 30760
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGBUS program

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

Floating:

```
vagrant@csc3150:~/csc3150as1/program1$ ./program1 ./flo
ating
Process start to fork
I'm the Parent Process, my pid = 30788
I'm the Child Process, my pid = 30789
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGFPE program

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

Hangup:

```
vagrant@csc3150:~/csc3150as1/program1$ ./program1 ./han
gup
Process start to fork
I'm the Parent Process, my pid = 30817
I'm the Child Process, my pid = 30818
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGHUP program

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

Illegal_instr:

```
vagrant@csc3150:~/csc3150as1/program1$ ./program1 ./ill
egal_instr
Process start to fork
I'm the Parent Process, my pid = 30845
I'm the Child Process, my pid = 30846
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGILL program

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

Interrupt:

```
vagrant@csc3150:~/csc3150as1/program1$ ./program1 ./int
errupt
Process start to fork
I'm the Parent Process, my pid = 30874
I'm the Child Process, my pid = 30875
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGINT program

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

Kill:

```
vagrant@csc3150:~/csc3150as1/program1$ ./program1 ./kil
l
Process start to fork
I'm the Parent Process, my pid = 30890
I'm the Child Process, my pid = 30891
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGKILL program

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

Pipe:

```
vagrant@csc3150:~/csc3150as1/program1$ ./program1 ./pip
e
Process start to fork
I'm the Parent Process, my pid = 30918
I'm the Child Process, my pid = 30919
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGPIPE program

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

Quit:

```
vagrant@csc3150:~/csc3150as1/program1$ ./program1 ./qui
t
Process start to fork
I'm the Parent Process, my pid = 30946
I'm the Child Process, my pid = 30947
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

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

Segmentation_fault:

```
vagrant@csc3150:~/csc3150as1/program1$ ./program1 ./seg
ment_fault
Process start to fork
I'm the Parent Process, my pid = 30976
I'm the Child Process, my pid = 30977
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

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

Stop:

```
vagrant@csc3150:~/csc3150as1/program1$ ./program1 ./sto
p
Process start to fork
I'm the Parent Process, my pid = 31006
I'm the Child Process, my pid = 31007
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

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

Terminate:

```
vagrant@csc3150:~/csc3150as1/program1$ ./program1 ./ter
minate
Process start to fork
I'm the Parent Process, my pid = 31046
I'm the Child Process, my pid = 31047
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTERM program

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

Trap:

```
vagrant@csc3150:~/csc3150as1/program1$ ./program1 ./tra
p
Process start to fork
I'm the Parent Process, my pid = 31074
I'm the Child Process, my pid = 31075
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTRAP program

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

Task 2

To compile: `make`

Running normal program:

```
[87572.744852] [program2] : module_init {Li Jiaqi} {120090727}
[87572.783427] [program2] : module_init create kthread start
[87572.829353] [program2] : module_init kthread start
[87572.837096] [program2] : The child process has pid = 13004
[87572.837097] [program2] : This is the parent process, pid = 13002
[87572.838491] [program2] : child process
[87577.811128] [program2] : get exit status = 0
[87577.819684] [program2] : child process terminated
[87577.832435] [program2] : The return signal is 0
[87586.983477] [program2] : module_exit
```

Abort:

```
[80779.478001] [program2] : module_init {Li Jiaqi} {120090727}
[80779.488347] [program2] : module_init create kthread start
[80779.497136] [program2] : module_init kthread start
[80779.505138] [program2] : The child process has pid = 3578
[80779.505140] [program2] : This is the parent process, pid = 3577
[80779.505286] [program2] : child process
[80796.218036] [program2] : get SIGABRT signal
[80796.242467] [program2] : child process terminated
[80796.278582] [program2] : The return signal is 6
[80851.455336] [program2] : module_exit
```

Bus:

```
[80854.203217] [program2] : module_init {Li Jiaqi} {120090727}
[80854.218037] [program2] : module_init create kthread start
[80854.247532] [program2] : module_init kthread start
[80854.266916] [program2] : The child process has pid = 3704
[80854.266918] [program2] : This is the parent process, pid = 3703
[80854.266987] [program2] : child process
[80854.566730] [program2] : get SIGBUS signal
[80854.598840] [program2] : child process terminated
[80854.629422] [program2] : The return signal is 7
```

Floating:

```
[80934.531597] [program2] : module_init {Li Jiaqi} {120090727}
[80934.578952] [program2] : module_init create kthread start
[80934.625296] [program2] : module_init kthread start
[80934.664739] [program2] : The child process has pid = 3794
[80934.664741] [program2] : This is the parent process, pid = 3793
[80934.664801] [program2] : child process
[80934.881947] [program2] : get SIGFPE signal
[80934.916138] [program2] : child process terminated
[80934.957203] [program2] : The return signal is 8
```

Hangup:

```
[80964.798364] [program2] : module_init {Li Jiaqi} {120090727}
[80964.844819] [program2] : module_init create kthread start
[80964.891404] [program2] : module_init kthread start
[80964.929579] [program2] : The child process has pid = 3890
[80964.929656] [program2] : This is the parent process, pid = 3889
[80964.929902] [program2] : child process
[80965.045759] [program2] : get SIGHUP signal
[80965.063566] [program2] : child process terminated
[80965.090024] [program2] : The return signal is 1
[81017.305989] [program2] : module_exit
```

Illegal_instr:

```
[81066.147212] [program2] : module_init {Li Jiaqi} {120090727}
[81066.156306] [program2] : module_init create kthread start
[81066.166410] [program2] : module_init kthread start
[81066.173798] [program2] : The child process has pid = 4043
[81066.173799] [program2] : This is the parent process, pid = 4042
[81066.173873] [program2] : child process
[81079.646497] [program2] : get SIGILL signal
[81079.679267] [program2] : child process terminated
[81079.711440] [program2] : The return signal is 4
[81092.851493] [program2] : module_exit
```

Interrupt:

```
[81122.986625] [program2] : module_init {Li Jiaqi} {120090727}
[81123.027532] [program2] : module_init create kthread start
[81123.064796] [program2] : module_init kthread start
[81123.106344] [program2] : The child process has pid = 4178
[81123.106346] [program2] : This is the parent process, pid = 4176
[81123.106441] [program2] : child process
[81123.186313] [program2] : get SIGINT signal
[81123.219341] [program2] : child process terminated
[81123.254770] [program2] : The return signal is 2
[81125.242730] [program2] : module_exit
```

Kill:

```
[81154.870303] [program2] : module_init {Li Jiaqi} {120090727}
[81154.916969] [program2] : module_init create kthread start
[81154.961518] [program2] : module_init kthread start
[81155.001288] [program2] : The child process has pid = 4257
[81155.001289] [program2] : This is the parent process, pid = 4256
[81155.001385] [program2] : child process
[81155.116076] [program2] : get SIGKILL signal
[81155.151324] [program2] : child process terminated
[81155.190452] [program2] : The return signal is 9
[81157.112036] [program2] : module_exit
```

Pipe:

```
[81180.209088] [program2] : module_init {Li Jiaqi} {120090727}
[81180.249253] [program2] : module_init create kthread start
[81180.303392] [program2] : module_init kthread start
[81180.346545] [program2] : The child process has pid = 4324
[81180.346546] [program2] : This is the parent process, pid = 4322
[81180.347034] [program2] : child process
[81180.448587] [program2] : get SIGPIPE signal
[81180.461266] [program2] : child process terminated
[81180.508479] [program2] : The return signal is 13
[81182.182894] [program2] : module_exit
```

Quit:

```
[81211.725859] [program2] : module_init {Li Jiaqi} {120090727}
[81211.773072] [program2] : module_init create kthread start
[81211.826140] [program2] : module_init kthread start
[81211.864242] [program2] : The child process has pid = 4412
[81211.864243] [program2] : This is the parent process, pid = 4411
[81211.864292] [program2] : child process
[81218.763784] [program2] : get SIGQUIT signal
[81218.797565] [program2] : child process terminated
[81218.841846] [program2] : The return signal is 3
[81221.591681] [program2] : module_exit
```

Segmentation_fault:

```
[81246.280683] [program2] : module_init {Li Jiaqi} {120090727}
[81246.311359] [program2] : module_init create kthread start
[81246.363659] [program2] : module_init kthread start
[81246.406092] [program2] : The child process has pid = 4516
[81246.406093] [program2] : This is the parent process, pid = 4515
[81246.406380] [program2] : child process
[81246.770647] [program2] : get SIGSEGV signal
[81246.823629] [program2] : child process terminated
[81246.867294] [program2] : The return signal is 11
[81248.310714] [program2] : module_exit
```

Stop:

```
[87136.436296] [program2] : module_init {Li Jiaqi} {120090727}
[87136.506328] [program2] : module_init create kthread start
[87136.569149] [program2] : module_init kthread start
[87136.607708] [program2] : The child process has pid = 11941
[87136.607710] [program2] : This is the parent process, pid = 11939
[87136.607802] [program2] : child process
[87136.720334] [program2] : get SIGSTOP signal
[87136.762682] [program2] : child process terminated
[87136.815553] [program2] : The return signal is 19
[87142.237163] [program2] : module_exit
```

Terminate:

```
[87221.563461] [program2] : module_init {Li Jiaqi} {120090727}
[87221.627680] [program2] : module_init create kthread start
[87221.691642] [program2] : module_init kthread start
[87221.706155] [program2] : The child process has pid = 12068
[87221.706157] [program2] : This is the parent process, pid = 12067
[87221.706576] [program2] : child process
[87221.823248] [program2] : get SIGTERM signal
[87221.855677] [program2] : child process terminated
[87221.891287] [program2] : The return signal is 15
[87225.806599] [program2] : module_exit
```

Trap:

```
[87253.165187] [program2] : module_init {Li Jiaqi} {120090727}
[87253.226733] [program2] : module_init create kthread start
[87253.265403] [program2] : module_init kthread start
[87253.316077] [program2] : The child process has pid = 12138
[87253.316078] [program2] : This is the parent process, pid = 12136
[87253.316151] [program2] : child process
[87253.516870] [program2] : get SIGTRAP signal
[87253.571495] [program2] : child process terminated
[87253.615718] [program2] : The return signal is 5
[87256.933895] [program2] : module_exit
```

Conclusion

I think this project is a valuable experience particularly in terms of giving me an experience of linux kernel. I once believe that the linux kernel is such an overwhelming project that I'm afraid of. By building the linux kernel from source, injecting a kernel module into it, and browsing the source code for hint of usage, I became more familiar with it. This project also enhances my C programming ability, which is a language I like very much for its simplicity.