Report of Assignment 1

120090702 Li Yiqian

1. Design of program

   a) Program1

   First, use fork() to create the child process, which will return the pid. After creating, the child and parent process will continue to execute the same code. The pid can be used to distinguish two processes. In child process, pid is 0 while in parent process, the pid is a number >0 to denote its pid compare to the whole system. Therefore, if pid=0, ask it to execute execve() funtion to call another file. If pid >0, ask it to execute waitpid() to wait for the signal from child process. In the externel file, it usually will raise some signal (except the normal.c,which will finish executing the whole externel file and raise SIGKILL in program1.c), then the parent process will receive it as well as the status of child process and terminate the child process. Then, the only running parent process will recognize the type of termination according to the output status. WTERMSIG(status) can return a int denoting type of signal. A switch is needed to turn them into char[] for printf.

   b) Program2

   After changing the kernel file and recompling the kernel (details in part 2), the static kernel functions can be invoked. To begin with, since the do_wait() requires the argument as a special struct, which is not defined in

head file, the struct wait_opts needs defining. Then, kthread_create(&my_fork,NULL,"start") can be used to create parent process and enter the my_fork(void *argc). It is convenient, specially all the later handling will only happen in parent process. Use wake_up_process(start) to let it run.

In my_fork(void *argc), init the k_sigaction struct for later signal receive and create a kernel_clone_args child_args for forking a child process. The flag and exit_signal should be set as SIGCHLD for later termination after parent process receving signal, .stack should be set as my_exec's address to push the function into stack and execute and wo_flags should be set as WUNTRACED | WEXITED to receive the stop signal correctly. After that, use kernel_clone(&child_args) to fork the child process. It actually clones the address space of parent process, which can be seen as create a new process.

Then, similarly as program1, use pid to distinguish two processes. If pid=0, execute my_exec(), which use do_execve(getname_kernel("/tmp/test "),NULL,NULL) to execute a external file "test". getname_kernel can turn the char[] filename into the required struct for do_execve. Later, a signal either from external file or do_exit(1) in program2 to kill child process will be sent to parent process, which is waiting through my_wait(pid_t pid).

To invoke the kernel function do_wait, the struct wait_opts wo is set. The do_wait will enqueue the wo and wake it when receiving signal from child process. wo.wo_pid is get from find_get_pid(pid), which will check the

pid hash table and return the actual pid* according to the pid number of parent. wo.wo_stat will receive the child termination status. There are two signals are special. One is 0, which means child processes killed itself through do_exit(1) and needs printk “normal termination”. Lastly, Use put_pid to free memory.

c) Bonus

The information of process is stored in /proc. Every number denotes the pid of the process. Use scandir to get all the dictories under proc. To filter the number-name dictories, define the function int numberOnly(const struct dirent *dir) to work together with scandir. All the process file name will be stored in namelist. Then, open the file /proc/ namelist[i]->d_name/status to get useful information. Search the file one line by one line and use findName, findPid, findPPid, findThreads functions, which are implemented by checking the words to get the name, pid, ppid, number of threads of the process. Here is a trick that the processes with ppid=2 and parent kthreadd are not shown in real “pstree” thus ignoring all of them. Create a struct node

```
{
        string name;
        int pid;
        int ppid;
        vector<TN*> child_threads;
        struct node *child;
        struct node *next;
        struct node *parent;
} NODE;
```

to store the information. If the threads number is larger than 1, meaning the process has child threads and requiring invoke getThreads.

The function getThreads will create the path2 “/proc/new_node->pid/task and read it. It will similarly store the number-name file in namelist. Then, for all files in namelist except the file having same name as new_node->pid, which is itself, getting every line and use

findName and findPid to get information for threads. Store it in struct

threadNode{

```
string name;
int pid;
```

} TN;

Finally, print the tree according to the order. Here, 4 options can be recognized, including no argument, -l, -c, -p. However, I failed to implement the no argument and -l types. And the format of print is not standard.

2. Environment set up

a) Download the origin 5.10 kernel

Firstly, download the kernel file. Use $make mrproper and $make clean to clean the enviornment. Use $make menuconfig to create a .config which includes the configuration information. Use $make bzImage -j$(nproc) to create and mirror image document kernel and use $make modules -j$(nproc) and $make modules_install to set and install the modules. Finally, use $make install to install the kernel.

b) Modify the kernel and recompile

Many kernel functions like do_wait, kernel_clone, do_execve are static functions in original kernel. To invoke them in our files, use EXPORT_SYMBOL(function name) to export them. Then, use “extern” in self file to invoke them. It requires to modify the kernel file. First, find which

file in kernel dictory linux-5.10.5 the functions lie in and edit the file, which can only be edited by root for safe, thus need “sudo su”, “vim” the file and “i” to edit it. Add EXPORT_SYMBOL(function name); in the file and use Esc+“:wq” to save the file. After that, from $make bzImage to create mirror image document and set, install the modules, install the kernel same as the first time to compile the kernel. Then, all the symbols will be added into Module.symdvers, from which the external file can invoke the functions.

3. Screenshot of output

a) Program1

```
vagrant@csc3150:~/CSC3150/Assignment_1_120090702/source/program1$ ./program1 ./normal
Process start to fork
I'm the Parent Process, my pid = 2297
I'm the Child Process, my pid = 2298
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
```

```
root@csc3150:/home/vagrant/CSC3150/Assignment_1_120090702/source/program1# ./program1 ./stop
Process start to fork
I'm the Parent Process, my pid = 14725
I'm the Child Process, my pid = 14726
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
```

```
root@csc3150:/home/vagrant/CSC3150/Assignment_1_120090702/source/program1# ./program1 ./abort
Process start to fork
I'm the Parent Process, my pid = 14833
I'm the Child Process, my pid = 14834
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
```

```
root@csc3150:/home/vagrant/CSC3150/Assignment_1_120090702/source/program1# ./program1 ./alarm
Process start to fork
I'm the Parent Process, my pid = 14879
I'm the Child Process, my pid = 14880
Child process start to execute test program
------------CHILD PROCESS START------------
This is the SIGALRM program

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

```
root@csc3150:/home/vagrant/CSC3150/Assignment_1_120090702/source/program1# ./program1 ./bus
Process start to fork
I'm the Parent Process, my pid = 14939
I'm the Child Process, my pid = 14940
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
```

```
root@csc3150:/home/vagrant/CSC3150/Assignment_1_120090702/source/program1# ./program1 ./floating
Process start to fork
I'm the Parent Process, my pid = 15026
I'm the Child Process, my pid = 15027
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
```

```
root@csc3150:/home/vagrant/CSC3150/Assignment_1_120090702/source/program1# ./program1 ./hangup
Process start to fork
I'm the Parent Process, my pid = 15101
I'm the Child Process, my pid = 15102
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
```

```
root@csc3150:/home/vagrant/CSC3150/Assignment_1_120090702/source/program1# ./program1 ./illegal_instr
Process start to fork
I'm the Parent Process, my pid = 15187
I'm the Child Process, my pid = 15188
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
```

```
root@csc3150:/home/vagrant/CSC3150/Assignment_1_120090702/source/program1# ./program1 ./interrupt
Process start to fork
I'm the Parent Process, my pid = 15274
I'm the Child Process, my pid = 15275
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
```

```
root@csc3150:/home/vagrant/CSC3150/Assignment_1_120090702/source/program1# ./program1 ./kill
Process start to fork
I'm the Parent Process, my pid = 15339
I'm the Child Process, my pid = 15340
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

root@csc3150:/home/vagrant/CSC3150/Assignment_1_120090702/source/program1# ./program1 ./pipe
Process start to fork
I'm the Parent Process, my pid = 15426
I'm the Child Process, my pid = 15427
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

root@csc3150:/home/vagrant/CSC3150/Assignment_1_120090702/source/program1# ./program1 ./quit
Process start to fork
I'm the Parent Process, my pid = 15504
I'm the Child Process, my pid = 15505
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

root@csc3150:/home/vagrant/CSC3150/Assignment_1_120090702/source/program1# ./program1 ./segment_fault
Process start to fork
I'm the Parent Process, my pid = 15579
I'm the Child Process, my pid = 15580
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
root@csc3150:/home/vagrant/CSC3150/Assignment_1_120090702/source/program1# ./program1 ./terminate
Process start to fork
I'm the Parent Process, my pid = 15645
I'm the Child Process, my pid = 15646
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

root@csc3150:/home/vagrant/CSC3150/Assignment_1_120090702/source/program1# ./program1 ./trap
Process start to fork
I'm the Parent Process, my pid = 15689
I'm the Child Process, my pid = 15690
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
```

b) Program2

```
[ 1377.113606] [program2] : module_init {Li Yiqian} {120090702}
[ 1377.113759] [program2] : module_init create kthread start
[ 1377.113818] [program2] : module_init kthread start
[ 1377.113894] [program2] : The child process has pid = 5685
[ 1377.113896] [program2] : This is the parent process, pid = 5684
[ 1377.113901] [program2] : child process
[ 1377.114718] [program2] : get SIGTERM signal
[ 1377.114720] [program2] : child process terminated
[ 1377.114722] [program2] : Normal termination with EXIT STATUS = 0
[ 1380.150997] [program2] : module_exit./my
```

```
[ 1777.938405] [program2] : module_init {Li Yiqian} {120090702}
[ 1777.938562] [program2] : module_init create kthread start
[ 1777.938917] [program2] : module_init kthread start
[ 1777.938949] [program2] : The child process has pid = 6190
[ 1777.938951] [program2] : This is the parent process, pid = 6189
[ 1777.938954] [program2] : child process
[ 1777.939742] [program2] : get SIGTERM signal
[ 1777.939745] [program2] : child process terminated
[ 1777.939746] [program2] : CHILD PROCESS STOPPED
[ 1783.423491] [program2] : module_exit./my
```

```
[14553.202907] [program2] : module_init {Li Yiqian} {120090702}
[14553.203008] [program2] : module_init create kthread start
[14553.203034] [program2] : module_init kthread start
[14553.203076] [program2] : The child process has pid = 9874
[14553.203077] [program2] : This is the parent process, pid = 9873
[14553.203080] [program2] : child process
[14553.343764] [program2] : get SIGTERM signal
[14553.343767] [program2] : child process terminated
[14553.343768] [program2] : The return signal is 6
[14556.488652] [program2] : module_exit./my
```

```
[14595.374707] [program2] : module_init {Li Yiqian} {120090702}
[14595.374812] [program2] : module_init create kthread start
[14595.374833] [program2] : module_init kthread start
[14595.374861] [program2] : The child process has pid = 10282
[14595.374861] [program2] : This is the parent process, pid = 10281
[14595.374872] [program2] : child process
[14597.375923] [program2] : get SIGTERM signal
[14597.375926] [program2] : child process terminated
[14597.375927] [program2] : The return signal is 14
[14598.636649] [program2] : module_exit./my
```

```
[14470.095454] [program2] : module_init {Li Yiqian} {120090702}
[14470.095605] [program2] : module_init create kthread start
[14470.095664] [program2] : module_init kthread start
[14470.095738] [program2] : The child process has pid = 9376
[14470.095754] [program2] : This is the parent process, pid = 9375
[14470.095767] [program2] : child process
[14470.257878] [program2] : get SIGTERM signal
[14470.257880] [program2] : child process terminated
[14470.257881] [program2] : The return signal is 7
[14473.471088] [program2] : module_exit./my
```

```
[14693.241389] [program2] : module_init {Li Yiqian} {120090702}
[14693.241546] [program2] : module_init create kthread start
[14693.241587] [program2] : module_init kthread start
[14693.241656] [program2] : The child process has pid = 10694
[14693.241658] [program2] : This is the parent process, pid = 10693
[14693.241684] [program2] : child process
[14693.413835] [program2] : get SIGTERM signal
[14693.413838] [program2] : child process terminated
[14693.413839] [program2] : The return signal is 8
[14696.739118] [program2] : module_exit./my
```

```
[14737.246412] [program2] : module_init {Li Yiqian} {120090702}
[14737.246524] [program2] : module_init create kthread start
[14737.246566] [program2] : module_init kthread start
[14737.246613] [program2] : The child process has pid = 11104
[14737.246614] [program2] : This is the parent process, pid = 11103
[14737.246617] [program2] : child process
[14737.258552] [program2] : get SIGTERM signal
[14737.258554] [program2] : child process terminated
[14737.258555] [program2] : The return signal is 1
[14741.171662] [program2] : module_exit./my

[14783.926304] [program2] : module_init {Li Yiqian} {120090702}
[14783.926440] [program2] : module_init create kthread start
[14783.926469] [program2] : module_init kthread start
[14783.926553] [program2] : The child process has pid = 11521
[14783.926554] [program2] : This is the parent process, pid = 11520
[14783.926573] [program2] : child process
[14784.076432] [program2] : get SIGTERM signal
[14784.076434] [program2] : child process terminated
[14784.076435] [program2] : The return signal is 4
[14787.428815] [program2] : module_exit./my

[14827.347152] [program2] : module_init {Li Yiqian} {120090702}
[14827.347252] [program2] : module_init create kthread start
[14827.347281] [program2] : module_init kthread start
[14827.347330] [program2] : The child process has pid = 11930
[14827.347332] [program2] : This is the parent process, pid = 11929
[14827.347351] [program2] : child process
[14827.358330] [program2] : get SIGTERM signal
[14827.358332] [program2] : child process terminated
[14827.358333] [program2] : The return signal is 2
[14830.413029] [program2] : module_exit./my

[14895.595356] [program2] : module_init {Li Yiqian} {120090702}
[14895.595609] [program2] : module_init create kthread start
[14895.595709] [program2] : module_init kthread start
[14895.595758] [program2] : The child process has pid = 12405
[14895.595760] [program2] : This is the parent process, pid = 12404
[14895.595795] [program2] : child process
[14895.606367] [program2] : get SIGTERM signal
[14895.606370] [program2] : child process terminated
[14895.606372] [program2] : The return signal is 9
[14900.455461] [program2] : module_exit./my

[14959.640917] [program2] : module_init {Li Yiqian} {120090702}
[14959.641003] [program2] : module_init create kthread start
[14959.641032] [program2] : module_init kthread start
[14959.641084] [program2] : The child process has pid = 12896
[14959.641086] [program2] : This is the parent process, pid = 12895
[14959.641146] [program2] : child process
[14959.651747] [program2] : get SIGTERM signal
[14959.651749] [program2] : child process terminated
[14959.651750] [program2] : The return signal is 13
[14962.905255] [program2] : module_exit./my
```

```
[14992.708264] [program2] : module_init {Li Yiqian} {120090702}
[14992.708381] [program2] : module_init create kthread start
[14992.708405] [program2] : module_init kthread start
[14992.708446] [program2] : The child process has pid = 13284
[14992.708447] [program2] : This is the parent process, pid = 13283
[14992.708463] [program2] : child process
[14992.850137] [program2] : get SIGTERM signal
[14992.850149] [program2] : child process terminated
[14992.850150] [program2] : The return signal is 3
[15000.671266] [program2] : module_exit./my
[15038.189553] [program2] : module_init {Li Yiqian} {120090702}
[15038.189707] [program2] : module_init create kthread start
[15038.190179] [program2] : module_init kthread start
[15038.190205] [program2] : The child process has pid = 13739
[15038.190207] [program2] : This is the parent process, pid = 13737
[15038.190210] [program2] : child process
[15038.335300] [program2] : get SIGTERM signal
[15038.335302] [program2] : child process terminated
[15038.335303] [program2] : The return signal is 11
[15041.234823] [program2] : module_exit./my
[15075.656977] [program2] : module_init {Li Yiqian} {120090702}
[15075.657094] [program2] : module_init create kthread start
[15075.657133] [program2] : module_init kthread start
[15075.657189] [program2] : The child process has pid = 14153
[15075.657191] [program2] : This is the parent process, pid = 14152
[15075.657199] [program2] : child process
[15075.661869] [program2] : get SIGTERM signal
[15075.661872] [program2] : child process terminated
[15075.661873] [program2] : The return signal is 15
[15078.669375] [program2] : module_exit./my
[15129.476763] [program2] : module_init {Li Yiqian} {120090702}
[15129.477026] [program2] : module_init create kthread start
[15129.477065] [program2] : module_init kthread start
[15129.477105] [program2] : The child process has pid = 14579
[15129.477106] [program2] : This is the parent process, pid = 14578
[15129.477123] [program2] : child process
[15129.607290] [program2] : get SIGTERM signal
[15129.607293] [program2] : child process terminated
[15129.607294] [program2] : The return signal is 5
[15132.507093] [program2] : module_exit./my
```

c) Bonus

```
● vagrant@csc3150:~/CSC3150/Assignment_1_120090702/source/bonus$ pstree
systemd─┬─accounts-daemon─┬─{gdbus}
        │                 └─{gmain}
        ├─acpid
        ├─2*[agetty]
        ├─atd
        ├─cron
        ├─dbus-daemon
        ├─dhclient
        ├─irqbalance
        ├─2*[iscsid]
        ├─lvmetad
        ├─lxcfs───2*[{lxcfs}]
        ├─mdadm
        ├─polkitd─┬─{gdbus}
        │         └─{gmain}
        ├─rsyslogd─┬─{in:imklog}
        │          ├─{in:imuxsock}
        │          └─{rs:main Q:Reg}
        ├─sshd─┬─sshd───sshd───bash─┬─sh───node─┬─node─┬─bash───pstree
        │      │                    │           │      └─11*[{node}]
        │      │                    │           ├─node─┬─node───6*[{node}]
        │      │                    │           │      └─11*[{node}]
        │      │                    │           ├─node───12*[{node}]
        │      │                    │           └─10*[{node}]
        │      │                    └─sleep
        │      └─sshd───sshd───bash───sleep
        ├─4*[stop]
        ├─systemd───(sd-pam)
        ├─systemd-journal
        ├─systemd-logind
        ├─systemd-udevd
        └─unattended-upgr───{gmain}
```

./pstree or ./pstree -l or ./pstree -c:

```
● vagrant@csc3150:~/CSC3150/Assignment_1_120090702/source/bonus$ ./pstree
systemd─┬─acpid
        ├─lxcfs─┬─{lxcfs}
        │       └─{lxcfs}
        ├─sshd─┬─sshd───sshd───bash───sh───node─┬─{node}
        │      │                               ├─{node}
        │      │                               ├─{node}
        │      │                               ├─{node}
        │      │                               ├─{node}
        │      │                               ├─{node}
        │      │                               ├─{node}
        │      │                               ├─{node}
        │      │                               ├─{node}
        │      │                               └─{node}
        │      └─sshd───sshd───bash─┬─node─┬─{node}
        │                           │      ├─{node}
        │                           │      ├─{node}
        │                           │      ├─{node}
        │                           │      ├─{node}
        │                           │      ├─{node}
        │                           │      ├─{node}
        │                           │      ├─{node}
        │                           │      ├─{node}
        │                           │      └─{node}
        │                           └─pstree
        ├─4*[stop]
        ├─systemd───(sd-pam)
        ├─systemd-journal
        ├─systemd-logind
        ├─systemd-udevd
        └─unattended-upgr───{gmain}
```


|—node——{node}
—{node}
—{node}
—{node}
—{node}
—{node}
—{node}
—{node}
—{node}
—{node}
—node——{node}
—{node}
—{node}
—{node}
—{node}
—{node}
|—node——{node}
—{node}
—{node}
—{node}
—{node}
—{node}
—{node}
|—sleep
|—sshd—sshd—bash
|—iscsid
|—iscsid
|—unattended-upgr——{gmain}
|—mdadm
|—polkitd——{gmain}
—{gdbus}
|—agetty
|—agetty
|—irqbalance
|—systemd—(sd-pam)
|—stop
|—systemd-journal
|—lvmetad
|—stop
|—systemd-udevd
|—stop
|—dhclient
|—stop
|—cron
|—dbus-daemon
|—accounts-daemon——{gmain}
—{gdbus}
|—atd
|—systemd-logind
|—rsyslogd——{in:imuxsock}
—{in:imklog}
—{rs:main Q:Reg}

./pstree -p:

```
vagrant@csc3150:~/CSC3150/Assignment_1_120090702/source/bonus$ ./pstree -p
systemd---acpid(1000)
        |-lxcfs(1003)---{lxcfs}(1010)
                       |-{lxcfs}(1012)
        |-sshd(1025)-sshd(1454)-sshd(1493)-bash(1494)-sh(1539)-node(1549)---{node}(1574)
                                                                          |-{node}(1575)
                                                                          |-{node}(1576)
                                                                          |-{node}(1577)
                                                                          |-{node}(1578)
                                                                          |-{node}(1597)
                                                                          |-{node}(1634)
                                                                          |-{node}(1635)
                                                                          |-{node}(1636)
                                                                          |-{node}(1637)
 -node(1653)---{node}(1654)
                                                                          |-{node}(1655)
                                                                          |-{node}(1656)
                                                                          |-{node}(1657)
                                                                          |-{node}(1658)
                                                                          |-{node}(1662)
                                                                          |-{node}(1674)
                                                                          |-{node}(1675)
                                                                          |-{node}(1676)
                                                                          |-{node}(1677)
                                                                          |-{node}(17954)
 -bash(17953)-pstree(18090)
        |-node(1686)---{node}(1687)
                                  |-{node}(1688)
                                  |-{node}(1689)
                                  |-{node}(1690)
                                  |-{node}(1691)
                                  |-{node}(1692)
                                  |-{node}(1693)
                                  |-{node}(1694)
                                  |-{node}(1695)
                                  |-{node}(1696)
                                  |-{node}(1710)
 -node(1765)---{node}(1766)
                                  |-{node}(1767)
                                  |-{node}(1768)
                                  |-{node}(1769)
                                  |-{node}(1770)
                                  |-{node}(1771)
        |-node(1697)---{node}(1698)
                                  |-{node}(1699)
                                  |-{node}(1700)
                                  |-{node}(1701)
                                  |-{node}(1702)
                                  |-{node}(1703)
                                  |-{node}(1704)
                                  |
```

4. Learn from the task

Firstly, I learnt how to create child process and how it work parallelly with parent process. I found that after creating the child process, two processes will continue to execute the same code. To let them execute different task, that is let child process execute the file and let parent wait for the signal from child, I need to use pid to divide them. Pid is like a ID number of process to help programmer distinguish different processes when they are running at the same time. Besides, parent process need to wait for the signal from child process and terminate it when receiving the signal, which means they cannot run independently. I chose to use waitpid() in program1 and design my_wait() in program2 to let parent wait for the signal.

In addition, I learnt how to invoke kernel mode functions to write program2. I met many difficulties, like the meanings of complex arguments and modify the kernel files. I tries to understand the meanings of arguments through reading the official documents and comparing the kernel mode functions with more familiar user mode functions. To invoke the static kernel functions, I learnt how to use EXPORT_SYMBOL and “extern”to let them be able to be invoked by other codes.

Finally, I learnt more about the kernel. Before the assignment, kernel and OS were far off and mysterious for me. However, after downloading a kernel and modifying it by myself, I realized that OS is just a complex program. It is based on codes, which we can modify according to our requirements. I can explore more about it than I thought.