CSC3150 Assignment1 Report

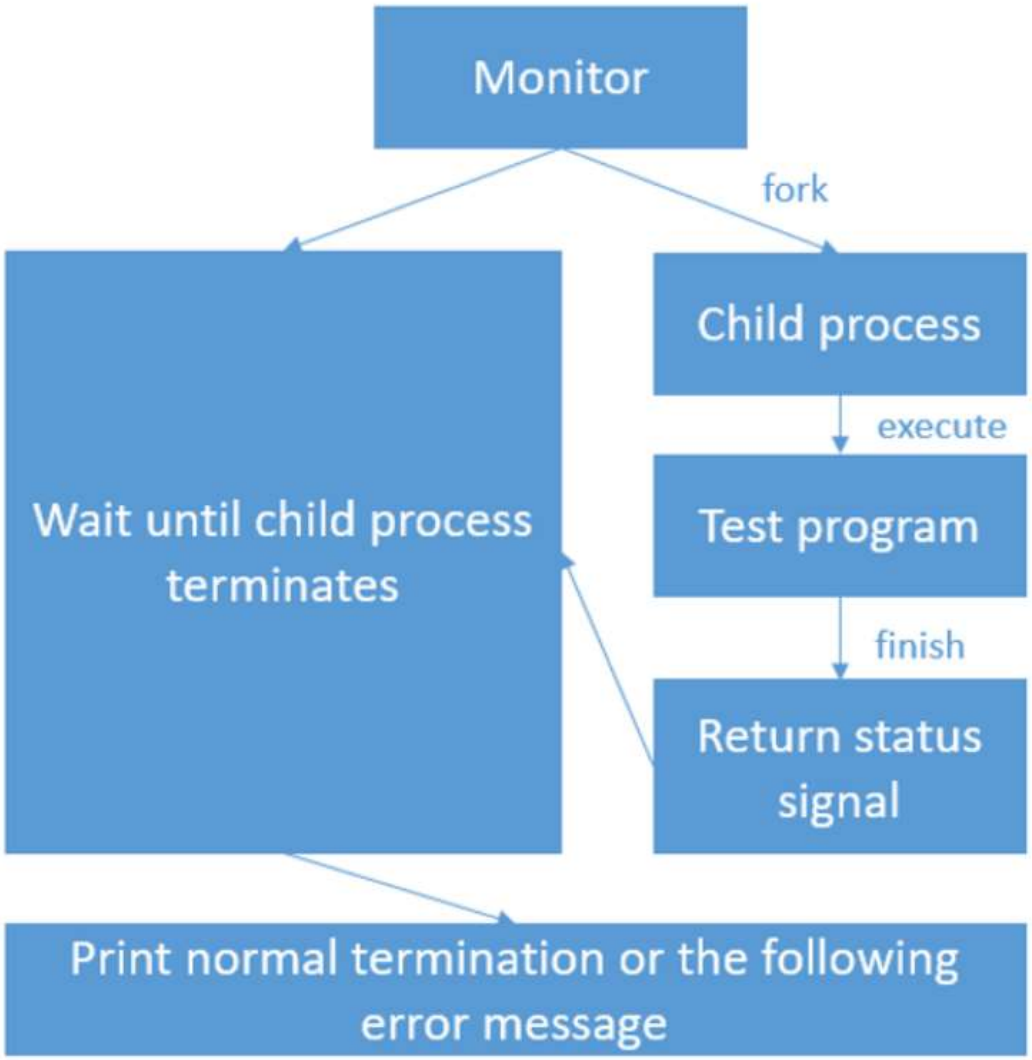
Name: Yan Tong ID: 120090454

Program Design

Task1

The goal of task1 is to fork a process in user mode and properly handle the signal raised by the child process.

The following flow chart is provided in the assignment document which clearly describe the program design.



First, ***fork*** system call is used for createing a new process, which is called child process. The **child process** runs concurrently with the **parent process**(the process which makes the fork syscall). Logically a child process is a exact copy of the parent process, but there're some differences which will not be discussed here. It should be noted that the child process and the parent process run in different memory space. However, their memory space have the same content. After a child process is created, both parent and child process will execute the next instruction following the ***fork()*** system call. ***fork()*** syscall takse no input. It will return twice. In parent process, it will return the child process's pid. In child process, it will return 0.

As ***fork*** returns 0 when it's child process, we can use a ***if statement*** to distinguish whether the following lines of codes belongs to child process or parent process.

If it's child process, use ***execve()*** system call to invoke the test program. If the ***execve()*** system call is properly invoked, the rest of the codes in original child process will not be executed.

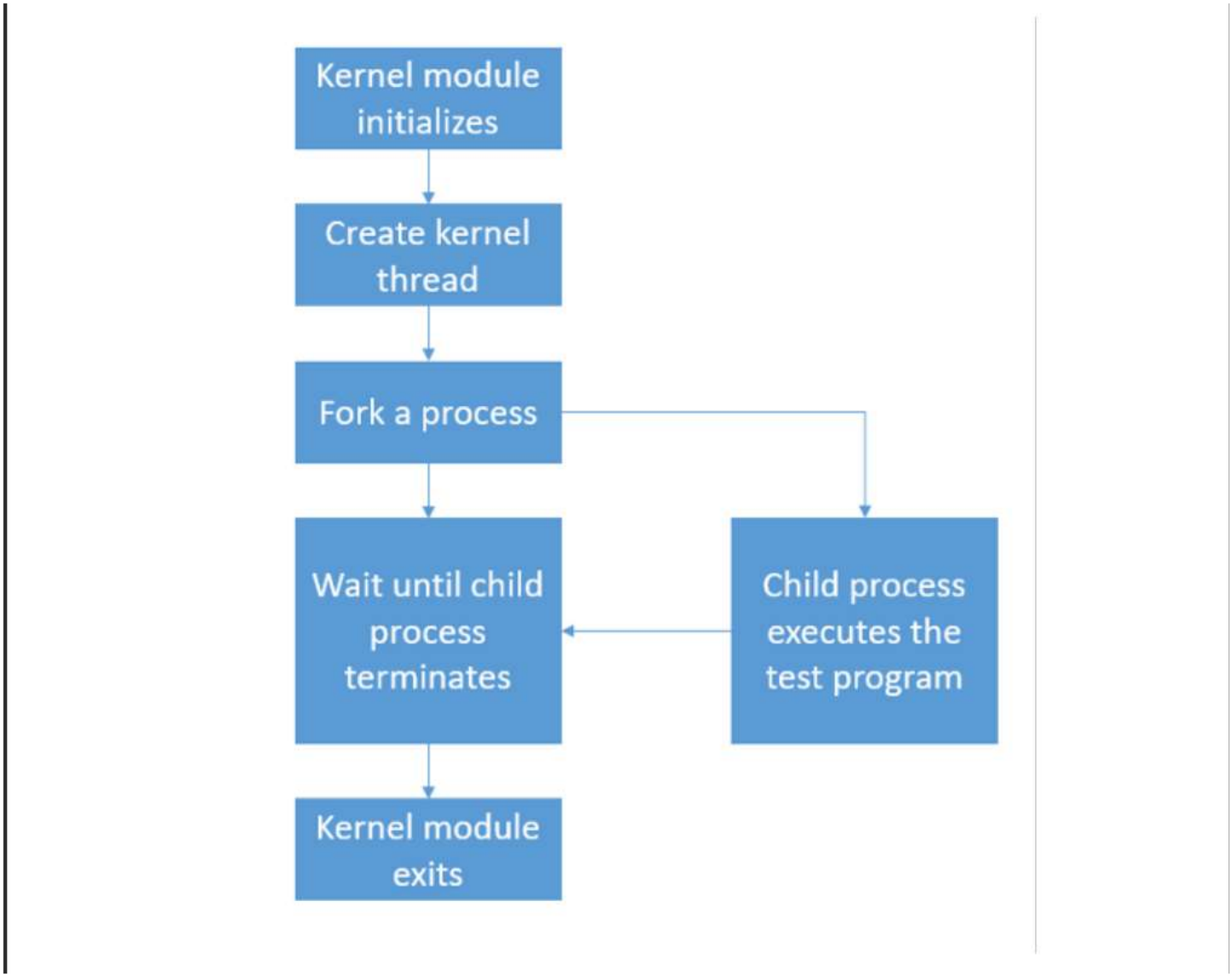
If it's the parent process, use ***waitpid()*** system call to wait for state changes in a child of the calling process, and obtain information about the child whose state has changed. Use WUNTRACED option so that the program will return if the child has stopped.

After that, use marcros defined in "wait.h" to analyze the signals return by wait() function.

Task2

The goal of task 2 is to implement the functionality of fork() function in kernel mode.

The flowchart is provided by assignment document.



- When program2.ko being initialized, create a kernel thread and run my_fork function.
- In my_fork function, fork a processs using kernel_thread function. The kernel_thread will create a child process and go to my_exec function
- In my_exec function, use getname_kernel function to get the test program's name. Then use do_execve function to execute the test program
- Use do_wait function in parent process such that it will wait until the child process terminates.
- Within the test program, it will raise a signal. The signal will be caught and related message will be get the do_wait function.
- When the do_wait function catch the signal, use same macros in task1 to analyze these signals and print the coorespoding information.

How to run the task2 program.

1. Compilie your project using make command:
    1. $make
2. Insert your kernel modules
    1. $sudo insmod program2.ko
3. Remove your kernel modules(important)

    1. $sudo rmmod program2
4. Display the information
    1. $dmesg

Bonus

The goal of bonus task is to implement 5 functionality of pstree command in linux system. I use ***c++14*** to implement this program. This program implement the following options:

- -p: print the pstree with pid.
- -n: sort the pstree by pid
- -c: disable the compact function
- -V: print out the version information
- default: print the pstree in default compacted mode.

The design of my program are as follows:

1. Parse the argument from the command line.
    1. Use getopt() function declared in unistd.h.
2. Get all the process information including pid, ppid, name and store them into a vector.
    1. Get the each process' information from "/proc/{pid of process}/stat". Get the thread information from "/proc/{pid of process}/task/{thread pid}/stat"
3. Build the pstree according to the passed arguments
    1. Build tree without pid:
        1. First determine a root node of the process. Then search the process information vector to find its children process.
        2. After that, recursively call build_tree function for each children of the root.
    2. Build tree with pid: Similar to the build_tree without pid. When assigning name to the Node, add the pid at the end.
4. Merge the tree(merge the brach with same name) in default
    1. Iterate through all children, find out the siblings who have the same name as the current children, and in the meantime do not have children. If that's the case, remove the sibling, and increase the count of the current children by 1.
    2. Recursively call merge_tree function to all children.
5. (optional) sort the pstree by pid.
    1. Use std::sort to sort the children vector of each node by their pid.
6. Print the pstree
    1. Use dfs to iterate through the whole tree. During that process, print out the process information. The details are massive and will not be disscussed here. Please refer to the code.

Environment Setting

The Linux kernel version of this project is 5.10.0.

In order to use certain functions in task2, several files in 5.10.0 kernel are modified. Sepecifically, kernel_thread function in "/kernel/fork.c", do_wait function in "/kernel/exit.c", do_execve function in "/fs/exec.c" and getname_kernel function in "/fs/namei.c" are exported by EXPORT_STMBOL.

After that we need to compile the kernel.

1. First clear the previous setting and start configuration by executing following commands
    1. $make mrproper
    2. $make clean
    3. $make menuconfig
2. Then build the kernel Image and modules
    1. $make - j$(nproc)
3. Install the kernel modules
    1. $make modules_insall
4. Install kernel
    1. $make install
5. Reboot to load new kernel
    1. $reboot

What have I learned

In this project, I learned how to modify and compile a kernel. I learned basic knowledge of process and its underlying working principle. In task2, I learned the underlying mechanism of fork, exec and wait system call. In the meantime, I learned basic knwoledge of kernel modules.

Most importantly, this project train my ability to use GOOGLE.

Program output

task1

The following are the sample output.

1. ABORT
    1.

```
vagrant@csc3150:~/csc3150/Assignment_1_120090454/source/program1$ ./program1 ./abort
Process start to fork
I'm the Parent Process, my pid = 28401
I'm the Child Process, my pid = 28402
Child Process start to execute a test program:
------------CHILD PROCESS START------------
This is the SIGABRT program

Parent process receives SIGCHLD signal
status: 134Child process is killed by SIGNAL: SIGABRT
Child process is killed by SIGNAL: 6
```

2. ALARM
    1.
```
vagrant@csc3150:~/csc3150/Assignment_1_120090454/source/program1$ ./program1 ./alarm
Process start to fork
I'm the Parent Process, my pid = 28511
I'm the Child Process, my pid = 28512
Child Process start to execute a test program:
------------CHILD PROCESS START------------
This is the SIGALRM program

Parent process receives SIGCHLD signal
status: 14Child process is killed by SIGNAL: SIGALRM
Child process is killed by SIGNAL: 14
```

3. BUS
    1.
```
vagrant@csc3150:~/csc3150/Assignment_1_120090454/source/program1$ ./program1 ./bus
Process start to fork
I'm the Parent Process, my pid = 28584
I'm the Child Process, my pid = 28585
Child Process start to execute a test program:
------------CHILD PROCESS START------------
This is the SIGBUS program

Parent process receives SIGCHLD signal
status: 135Child process is killed by SIGNAL: SIGBUS
Child process is killed by SIGNAL: 7
```

4. FLOATING
    1.
```
vagrant@csc3150:~/csc3150/Assignment_1_120090454/source/program1$ ./program1 ./floating
Process start to fork
I'm the Parent Process, my pid = 28644
I'm the Child Process, my pid = 28645
Child Process start to execute a test program:
------------CHILD PROCESS START------------
This is the SIGFPE program

Parent process receives SIGCHLD signal
status: 136Child process is killed by SIGNAL: SIGFPE
Child process is killed by SIGNAL: 8
```

5. HANGUP
    1.
```
vagrant@csc3150:~/csc3150/Assignment_1_120090454/source/program1$ ./program1 ./hangup
Process start to fork
I'm the Parent Process, my pid = 28715
I'm the Child Process, my pid = 28716
Child Process start to execute a test program:
------------CHILD PROCESS START------------
This is the SIGHUP program

Parent process receives SIGCHLD signal
status: 1Child process is killed by SIGNAL: SIGHUP
Child process is killed by SIGNAL: 1
```

6. INTERRUPT

1.
```
vagrant@csc3150:~/csc3150/Assignment_1_120090454/source/program1$ ./program1 ./interrupt
Process start to fork
I'm the Parent Process, my pid = 28864
I'm the Child Process, my pid = 28865
Child Process start to execute a test program:
------------CHILD PROCESS START------------
This is the SIGINT program

Parent process receives SIGCHLD signal
status: 2Child process is killed by SIGNAL: SIGINT
Child process is killed by SIGNAL: 2
```

7. KILL
    1.
```
vagrant@csc3150:~/csc3150/Assignment_1_120090454/source/program1$ ./program1 ./kill
Process start to fork
I'm the Parent Process, my pid = 28898
I'm the Child Process, my pid = 28899
Child Process start to execute a test program:
------------CHILD PROCESS START------------
This is the SIGKILL program

Parent process receives SIGCHLD signal
status: 9Child process is killed by SIGNAL: SIGKILL
Child process is killed by SIGNAL: 9
```

8. NORMAL
    1.
```
vagrant@csc3150:~/csc3150/Assignment_1_120090454/source/program1$ ./program1 ./normal
Process start to fork
I'm the Parent Process, my pid = 28961
I'm the Child Process, my pid = 28962
Child Process start to execute a test program:
------------CHILD PROCESS START------------
This is the normal program

------------CHILD PROCESS END------------
Parent process receives SIGCHLD signal
Normal termination with EXIT STATUS = 0
```

9. PIPE
    1.
```
vagrant@csc3150:~/csc3150/Assignment_1_120090454/source/program1$ ./program1 ./pipe
Process start to fork
I'm the Parent Process, my pid = 29015
I'm the Child Process, my pid = 29016
Child Process start to execute a test program:
------------CHILD PROCESS START------------
This is the SIGPIPE program

Parent process receives SIGCHLD signal
status: 13Child process is killed by SIGNAL: SIGPIPE
Child process is killed by SIGNAL: 13
```

    =

10. SEGMENTFAULT
    1.
```
vagrant@csc3150:~/csc3150/Assignment_1_120090454/source/program1$ ./program1 ./segment_fault
Process start to fork
I'm the Parent Process, my pid = 29118
I'm the Child Process, my pid = 29119
Child Process start to execute a test program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

Parent process receives SIGCHLD signal
status: 139Child process is killed by SIGNAL: SIGSEGV
Child process is killed by SIGNAL: 11
```

11. QUIT

1.
```
vagrant@csc3150:~/csc3150/Assignment_1_120090454/source/program1$ ./program1 ./quit
Process start to fork
I'm the Parent Process, my pid = 29066
I'm the Child Process, my pid = 29067
Child Process start to execute a test program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

Parent process receives SIGCHLD signal
status: 131Child process is killed by SIGNAL: SIGQUIT
Child process is killed by SIGNAL: 3
```

12. STOP

1.
```
vagrant@csc3150:~/csc3150/Assignment_1_120090454/source/program1$ ./program1 ./stop
Process start to fork
I'm the Parent Process, my pid = 29138
I'm the Child Process, my pid = 29139
Child Process start to execute a test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

Parent process receives SIGCHLD signal
Child process is stopped with SIGNAL: SIGSTOP
CHILD PROCESS STOPPED
```

13. TERMINATE

1.
```
vagrant@csc3150:~/csc3150/Assignment_1_120090454/source/program1$ ./program1 ./terminate
Process start to fork
I'm the Parent Process, my pid = 29188
I'm the Child Process, my pid = 29189
Child Process start to execute a test program:
------------CHILD PROCESS START------------
This is the SIGTERM program

Parent process receives SIGCHLD signal
status: 15Child process is killed by SIGNAL: SIGTERM
Child process is killed by SIGNAL: 15
```

14. TRAP

1.
```
vagrant@csc3150:~/csc3150/Assignment_1_120090454/source/program1$ ./program1 ./trap
Process start to fork
I'm the Parent Process, my pid = 29237
I'm the Child Process, my pid = 29238
Child Process start to execute a test program:
------------CHILD PROCESS START------------
This is the SIGTRAP program

Parent process receives SIGCHLD signal
status: 133Child process is killed by SIGNAL: SIGTRAP
Child process is killed by SIGNAL: 5
```

15. ILLEFAL INSTRUCTION

1.
```
vagrant@csc3150:~/csc3150/Assignment_1_120090454/source/program1$ ./program1 ./illegal_instr
Process start to fork
I'm the Parent Process, my pid = 29273
I'm the Child Process, my pid = 29274
Child Process start to execute a test program:
------------CHILD PROCESS START------------
This is the SIGILL program

Parent process receives SIGCHLD signal
status: 132Child process is killed by SIGNAL: SIGILL
Child process is killed by SIGNAL: 4
```

task2

The following are the sample output.

The following are the sample output.

1. ABORT
    1.
```
[178465.065121] [program2] : Module_init YANTONG 120090454
[178465.068227] [program2] : module_init create kthread start
[178465.071658] [program2] : module_init kthread starts
[178465.074353] [program2] : This is the Child process, pid = 32120
[178465.078194] [program2] : This is the parent process, pid = 32119
[178465.085032] [program2] : child process
[178465.228525] [program2] : Child process is killed by SIGNAL: SIGABRT
[178465.236348] [program2] : children process terminated
[178465.236349] [program2] : The return signal is: 6
[178467.164403] [program2] : Module_exit
```
2. ALARM
    1.
```
[178372.056912] [program2] : Module_init YANTONG 120090454
[178372.060553] [program2] : module_init create kthread start
[178372.064637] [program2] : module_init kthread starts
[178372.068613] [program2] : This is the Child process, pid = 31705
[178372.072652] [program2] : This is the parent process, pid = 31704
[178372.079517] [program2] : child process
[178374.081182] [program2] : Child process is killed by SIGNAL: SIGALRM
[178374.088312] [program2] : children process terminated
[178374.088312] [program2] : The return signal is: 14
[178374.921707] [program2] : Module_exit
```
3. BUS
    1.
```
[178504.898689] [program2] : Module_init YANTONG 120090454
[178504.902062] [program2] : module_init create kthread start
[178504.905902] [program2] : module_init kthread starts
[178504.908907] [program2] : This is the Child process, pid = 32536
[178504.912576] [program2] : This is the parent process, pid = 32535
[178504.919720] [program2] : child process
[178505.015623] [program2] : Child process is killed by SIGNAL: SIGBUS
[178505.022943] [program2] : children process terminated
[178505.022944] [program2] : The return signal is: 7
[178505.863596] [program2] : Module_exit
```
4. FLOATING
    1.
```
[178545.793775] [program2] : Module_init YANTONG 120090454
[178545.796874] [program2] : module_init create kthread start
[178545.804335] [program2] : module_init kthread starts
[178545.807376] [program2] : This is the Child process, pid = 473
[178545.810578] [program2] : This is the parent process, pid = 472
[178545.820755] [program2] : child process
[178545.902802] [program2] : Child process is killed by SIGNAL: SIGFPE
[178545.909462] [program2] : children process terminated
[178545.909463] [program2] : The return signal is: 8
[178547.488163] [program2] : Module_exit
```
5. HANGUP
    1.

```
[178686.066774] [program2] : Module_init YANTONG 120090454
[178686.070040] [program2] : module_init create kthread start
[178686.073650] [program2] : module_init kthread starts
[178686.076803] [program2] : This is the Child process, pid = 1608
[178686.080083] [program2] : This is the parent process, pid = 1607
[178686.086262] [program2] : child process
[178686.087322] [program2] : Child process is killed by SIGNAL: SIGINT
[178686.093296] [program2] : children process terminated
[178686.093297] [program2] : The return signal is: 2
[178687.025118] [program2] : Module_exit
```

6. INTERRUPT
    1.
```
[178579.544997] [program2] : Module_init YANTONG 120090454
[178579.548007] [program2] : module_init create kthread start
[178579.551196] [program2] : module_init kthread starts
[178579.554290] [program2] : This is the Child process, pid = 855
[178579.557686] [program2] : This is the parent process, pid = 854
[178579.563513] [program2] : child process
[178579.564001] [program2] : Child process is killed by SIGNAL: SIGHUP
[178579.570045] [program2] : children process terminated
[178579.570046] [program2] : The return signal is: 1
[178580.443406] [program2] : Module_exit
```

7. KILL
    1.
```
[178714.394843] [program2] : Module_init YANTONG 120090454
[178714.398270] [program2] : module_init create kthread start
[178714.401535] [program2] : module_init kthread starts
[178714.404485] [program2] : This is the Child process, pid = 2043
[178714.407669] [program2] : This is the parent process, pid = 2042
[178714.413266] [program2] : child process
[178714.414927] [program2] : Child process is killed by SIGNAL: SIGKILL
[178714.421018] [program2] : children process terminated
[178714.421019] [program2] : The return signal is: 9
[178715.362258] [program2] : Module_exit
```

8. NORMAL
    1.
```
[178787.752868] [program2] : Module_init YANTONG 120090454
[178787.756008] [program2] : module_init create kthread start
[178787.759769] [program2] : module_init kthread starts
[178787.762697] [program2] : This is the Child process, pid = 2786
[178787.765765] [program2] : This is the parent process, pid = 2785
[178787.771699] [program2] : child process
[178787.773478] [program2] : Normal termination with EXIT STATUS = 0
[178788.694777] [program2] : Module_exit
```

9. PIPE
    1.

```
[178819.859468] [program2] : Module_init YANTONG 120090454
[178819.862425] [program2] : module_init create kthread start
[178819.866046] [program2] : module_init kthread starts
[178819.869392] [program2] : This is the Child process, pid = 3150
[178819.873372] [program2] : This is the parent process, pid = 3149
[178819.881376] [program2] : child process
[178819.884998] [program2] : Child process is killed by SIGNAL: SIGPIPE
[178819.893053] [program2] : children process terminated
[178819.893054] [program2] : The return signal is: 13
[178820.757974] [program2] : Module_exit
```

10. SEGMENTFAULT
    1.
```
[178877.137336] [program2] : Module_init YANTONG 120090454
[178877.140695] [program2] : module_init create kthread start
[178877.144267] [program2] : module_init kthread starts
[178877.147289] [program2] : This is the Child process, pid = 4230
[178877.150606] [program2] : This is the parent process, pid = 4229
[178877.157153] [program2] : child process
[178877.245050] [program2] : Child process is killed by SIGNAL: SIGSEGV
[178877.251282] [program2] : children process terminated
[178877.251283] [program2] : The return signal is: 11
[178878.440650] [program2] : Module_exit
```

11. QUIT
    1.
```
[178844.428761] [program2] : Module_init YANTONG 120090454
[178844.431625] [program2] : module_init create kthread start
[178844.434593] [program2] : module_init kthread starts
[178844.438256] [program2] : This is the Child process, pid = 3533
[178844.441365] [program2] : This is the parent process, pid = 3532
[178844.446702] [program2] : child process
[178844.532210] [program2] : Child process is killed by SIGNAL: SIGQUIT
[178844.537996] [program2] : children process terminated
[178844.537996] [program2] : The return signal is: 3
[178845.309227] [program2] : Module_exit
```

12. STOP
    1.
```
[178923.539828] [program2] : Module_init YANTONG 120090454
[178923.542846] [program2] : module_init create kthread start
[178923.546029] [program2] : module_init kthread starts
[178923.548990] [program2] : This is the Child process, pid = 4954
[178923.552277] [program2] : This is the parent process, pid = 4953
[178923.558631] [program2] : child process
[178923.558980] [program2] : Child process is stopped with SIGNAL: SIGSTOP
[178923.564638] [program2] : children process stopped
[178923.567315] [program2] : Child process is stopped with SIGNAL: 19
[178924.544031] [program2] : Module_exit
```

13. TERMINATE
    1.

```
[178960.221993] [program2] : Module_init YANTONG 120090454
[178960.224982] [program2] : module_init create kthread start
[178960.228427] [program2] : module_init kthread starts
[178960.231435] [program2] : This is the Child process, pid = 5308
[178960.234683] [program2] : This is the parent process, pid = 5307
[178960.240888] [program2] : child process
[178960.241281] [program2] : Child process is killed by SIGNAL: SIGTERM
[178960.248082] [program2] : children process terminated
[178960.248083] [program2] : The return signal is: 15
[178961.141746] [program2] : Module_exit
```

14. TRAP
    1.
```
[178987.184890] [program2] : Module_init YANTONG 120090454
[178987.188089] [program2] : module_init create kthread start
[178987.191851] [program2] : module_init kthread starts
[178987.195621] [program2] : This is the Child process, pid = 5674
[178987.199033] [program2] : This is the parent process, pid = 5673
[178987.205410] [program2] : child process
[178987.299803] [program2] : Child process is killed by SIGNAL: SIGTRAP
[178987.306332] [program2] : children process terminated
[178987.306333] [program2] : The return signal is: 5
[178988.180333] [program2] : Module_exit
```

15. ILLEFAL INSTRUCTION
    1.
```
[178627.514631] [program2] : Module_init YANTONG 120090454
[178627.517630] [program2] : module_init create kthread start
[178627.521533] [program2] : module_init kthread starts
[178627.524601] [program2] : This is the Child process, pid = 1239
[178627.527778] [program2] : This is the parent process, pid = 1238
[178627.533401] [program2] : child process
[178627.620578] [program2] : Child process is killed by SIGNAL: SIGILL
[178627.626890] [program2] : children process terminated
[178627.626891] [program2] : The return signal is: 4
[178628.442502] [program2] : Module_exit
```