# REPORT FOR ASSIGNMENT I

*The Chinese University of Hong Kong, Shenzhen*

*CSC3150 OPERATING SYSTEM*

王子忱　120090907

2022/10/10

## 1.Program Design

### 1.1Program1

The execution logic of program 1 is simple. Firstly, fork a child process. In the child process, print logs and then call execv(argv[1], argv + 1) to replace itself with designated program and forward arguments. In the parent process, print logs, wait the child process with WUNTRACED option, and then print termination information according to the exit status.

### 1.2Program2

The first is the init function and the exit function. They will print logs into kernel log. Within the init function, a kernel thread is created and runs the my_fork function. First, use _do_fork() to create a new process. Then in the child process, use do_execve() to replace the current process with test program. In the parent process, use do_wait() to wait the child process and print termination information according to the exit status. Since struct wait_opts and long do_wait(struct wait_opts *); has not been defined in any Linux kernel headers, we must declare them manually in our source codes of our own module. Note that this does not pose any problems, as C language allows multiple function declarations, as well as multiple definitions of structs.

## 2. Environment

OS: Ubuntu 20.04.5 LTS

Kernel: Linux:5.15

## 3.3 Execution Steps

$ cd program1

$ make

$ ./program1 $TEST_CASE $ARG1 $ARG2 ..

Program 2:

$make

$sudo su

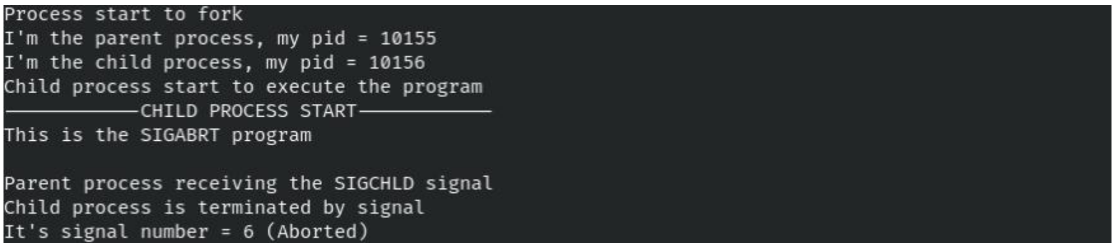
$insmod program2.ko

$rmmod program2.ko

$dmesg | tail -10

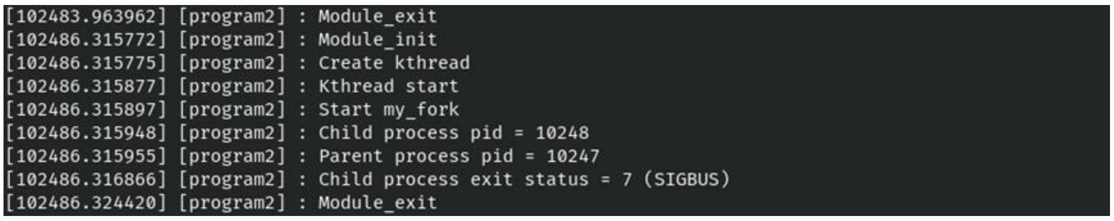
## 4.Output Demo

### Program1



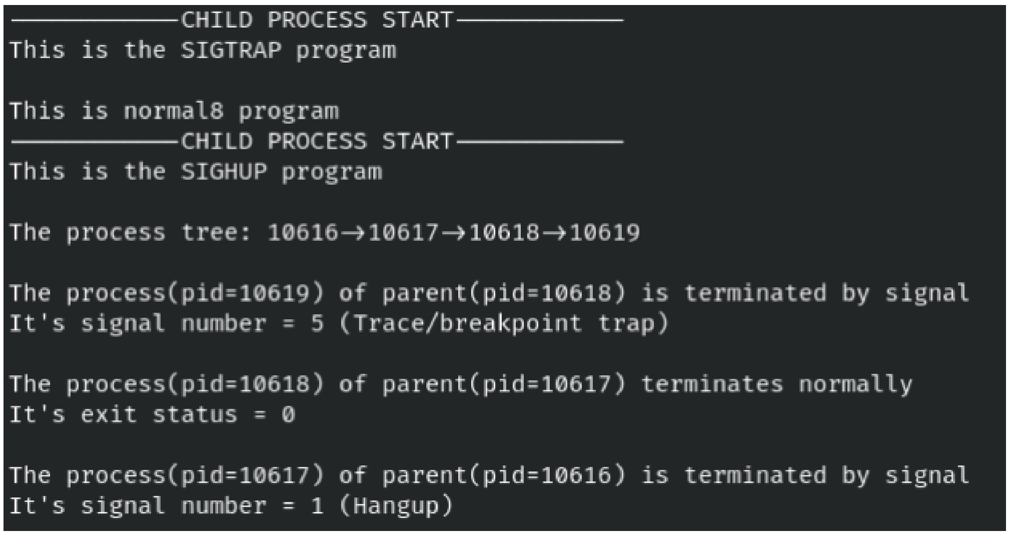
```
Process start to fork
I'm the parent process, my pid = 10155
I'm the child process, my pid = 10156
Child process start to execute the program
------------CHILD PROCESS START------------
This is the SIGABRT program

Parent process receiving the SIGCHLD signal
Child process is terminated by signal
It's signal number = 6 (Aborted)
```

### Program2



```
[102483.963962] [program2] : Module_exit
[102486.315772] [program2] : Module_init
[102486.315775] [program2] : Create kthread
[102486.315877] [program2] : Kthread start
[102486.315897] [program2] : Start my_fork
[102486.315948] [program2] : Child process pid = 10248
[102486.315955] [program2] : Parent process pid = 10247
[102486.316866] [program2] : Child process exit status = 7 (SIGBUS)
[102486.324420] [program2] : Module_exit
```

## Bonus



```
————————————CHILD PROCESS START————————————
This is the SIGTRAP program

This is normal8 program
————————————CHILD PROCESS START————————————
This is the SIGHUP program

The process tree: 10616->10617->10618->10619

The process(pid=10619) of parent(pid=10618) is terminated by signal
It's signal number = 5 (Trace/breakpoint trap)

The process(pid=10618) of parent(pid=10617) terminates normally
It's exit status = 0

The process(pid=10617) of parent(pid=10616) is terminated by signal
It's signal number = 1 (Hangup)
```