CSC3150 Assignment 1 Report

Junyu Jin

120090477

2022.10.10

Brief Introduction:

The project consists of 2 parts. In part 1, we need to write program1.c to complete the task 1. In part 2, we need to write program2.c to complete the task 2.

Task 1 includes:

1. Fork a child process
2. Let the parent process wait for the child process
3. Print the termination information of the child process

Task 2 includes:

1. Revise and recompile the Linux kernel source code
2. Create a kernel thread and run my_fork function
3. Fork a process to execute test.o
4. Let the parent process wait for the child process
5. Print pids of parent and child processes
6. Catch the signal raised by the child process and print infos

Program environment:

Linux Distribution: Ubuntu 16.04.12

Linux Kernel Version: 5.10.5

GCC Version: 5.4.0

How to Run the Code:

Program 1:

```
cd program1

make

./program1 filename
```

Program 2:

```
cd program2

        sudo su

gcc -o test test.c

make

sudo insmod program2.ko

sudo rmmod program2.ko

    dmesg | tail -n 10
```

Function Explanation:

Program1:

Fork the child process

```
pid_t pid;
int status;
printf("Process start to fork\n");
pid = fork();
```

Child process execute

```
if (pid == 0) {
    printf("I'm the Child Process, my pid = %d\n", getpid());
    printf("Child process start to execute test program:\n");
```

```
  int i;
  char *arg[argc];
  for (i = 0; i < argc - 1; i++) {
    arg[i] = argv[i + 1];
  }
  arg[argc - 1] = NULL;
  execve(arg[0], arg, NULL);
  perror("execve");
  exit(EXIT_FAILURE);
}
```

Exit status check

```
if (WIFEXITED(status)) {
    printf("Normal termination with EXIT STATUS = %d\n",
           WEXITSTATUS(status));
  } else if (WIFSIGNALED(status)) {
    int s = WTERMSIG(status);
    switch (s) {
    case 6:
      printf("Child process get SIGABRT signal\n");
      break;
    case 14:
      printf("Child process get SIGALRM signal\n");
      break;
    case 7:
      printf("Child process get SIGBUS signal\n");
      break;
   ...
    }
    // printf("CHILD EXECUTION FAILED: %d\n", WTERMSIG(status));
  } else if (WIFSTOPPED(status)) {
    printf("Child process get SIGSTOP signal\n");
  } else {
    printf("Child process get SIGCONT signal\n");
  }
```

Parent wait

```
waitpid(-1, &status, WUNTRACED);
 printf("Parent process receives SIGCHLD signal\n");
```

Program2:

Recompile functions (Export symbol in sourse)

```
extern pid_t kernel_clone(struct kernel_clone_args *kargs);
extern int do_execve(struct filename *filename,
             const char __user *const __user *__argv,
             const char __user *const __user *__envp);
extern long do_wait(struct wait_opts *wo);
extern struct filename *getname_kernel(const char *filename);
```

Kernel thread create

```
printk("[program2] : Module_init {Junyu Jin} {120090477}\n");
printk("[program2] : Module_init create kthread start\n");
task = kthread_create(&my_fork, NULL, "Mythread");


if (!IS_ERR(task)) {
    printk("[program2] : module_init kthread starts\n");
    wake_up_process(task);
}
```

Process fork

```
pid_t pid;
pid = kernel_clone(&args);
printk("[program2] : The child process has pid= %d\n", pid);
printk("[program2] : The parent process has pid= %d\n",
       (int)current->pid);
```

Program execute

```
int my_exec(void)
{
    int result;
    const char path[] = "/home/vagrant/csc3150/program2/test";
    const char *const argv[] = { path, NULL, NULL };
    const char *const envp[] = { "HOME=/",
                     "PATH=/sbin:/user/sbin:/bin:/usr/bin",
                     NULL };


    struct filename *my_filename = getname_kernel(path);


    printk("[program2] : child process");
    result = do_execve(my_filename, NULL, NULL);
    // printk("result: %d\n", result);
    // printk("!result: %d\n", !result);
    if (!result) {
        // printk("abnormal");
        return 0;
    }
    // printk("normal");
    do_exit(result);
}
```

Wait for child

```
void my_wait(pid_t pid)
{
    int status = 0;
    int value;
    struct wait_opts wo;
    struct pid *wo_pid = NULL;
    enum pid_type type;
    type = PIDTYPE_PID;
    wo_pid = find_get_pid(pid);
    wo.wo_type = type;
    wo.wo_pid = wo_pid;
    wo.wo_flags = WEXITED;
    wo.wo_info = NULL;
    wo.wo_stat = status;
    wo.wo_rusage = NULL;
    value = do_wait(&wo);
    int s = wo.wo_stat;
      switch (s) {
    case 134:
        printk("[program2] :get SIGABRT signal\n");
        printk("[program2] : Child process terminated\n");
        printk("[program2] :The return signal is 6\n");
        break;
    case 14:
        printk("[program2] :get SIGALRM signal\n");
        printk("[program2] : Child process terminated\n");
        printk("[program2] :The return signal is 14\n");
        break;
    ...
    }
    put_pid(wo_pid);
    return;
}
```

Environment Set Up and Kernel Compile:

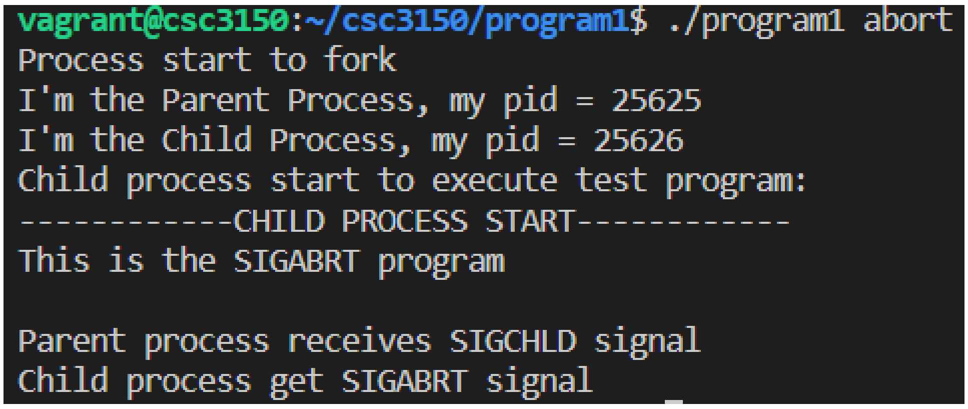
I followed the instructions in the tut2, downloaded linux-5.10.5 and copied the config. Then start configuration and built modules. I used EXPORT_SYMBOL() function to use the code out of sourse. To achieve that I recompile the kernel and reboot it. Finally the work is completed.

What I learn:

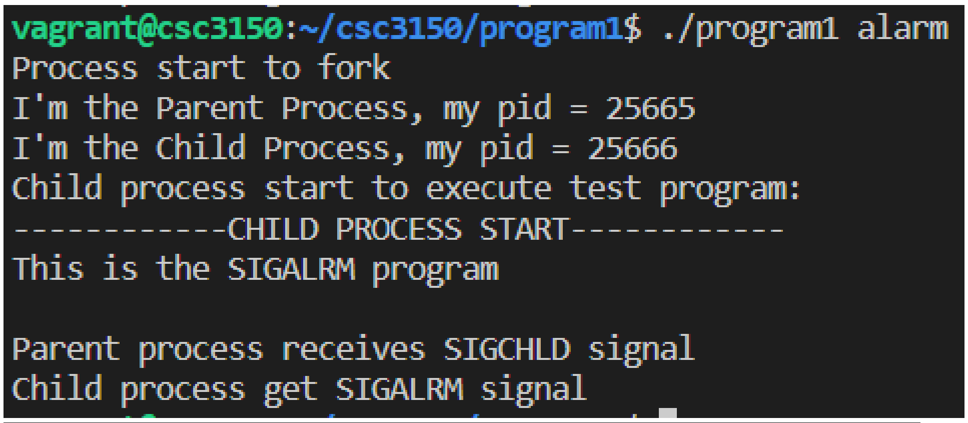
I get to know clearer about the process. I mastered linux environment set up, the usage of clang format. These tasks enable me to program in C although I didn't learnt that before. I use much more linux API than before, these tasks give me experience.

Program Output:

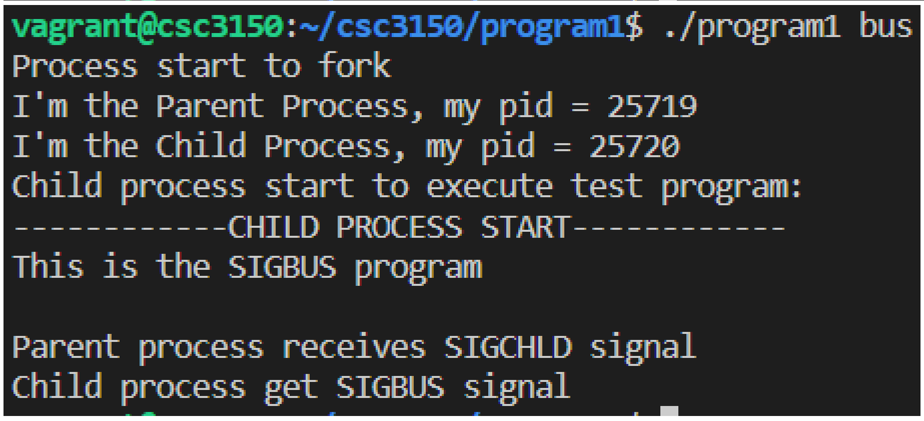
Program1:



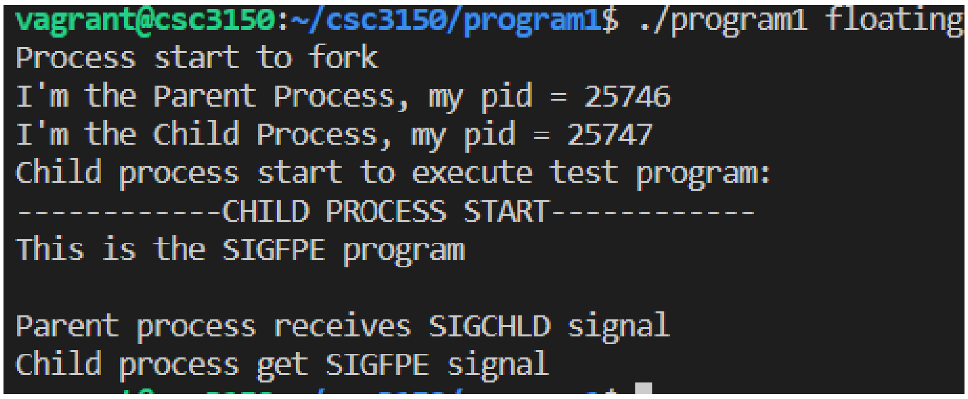
```
vagrant@csc3150:~/csc3150/program1$ ./program1 abort
Process start to fork
I'm the Parent Process, my pid = 25625
I'm the Child Process, my pid = 25626
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGABRT program

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



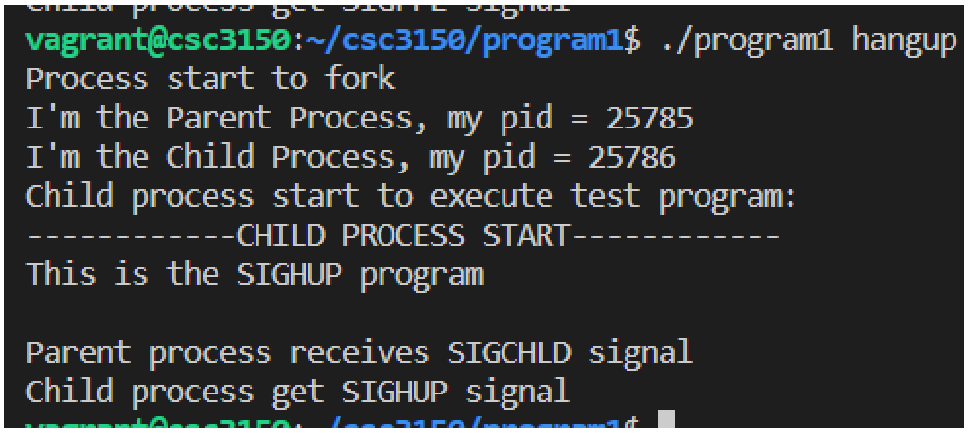
```
vagrant@csc3150:~/csc3150/program1$ ./program1 alarm
Process start to fork
I'm the Parent Process, my pid = 25665
I'm the Child Process, my pid = 25666
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGALRM program

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



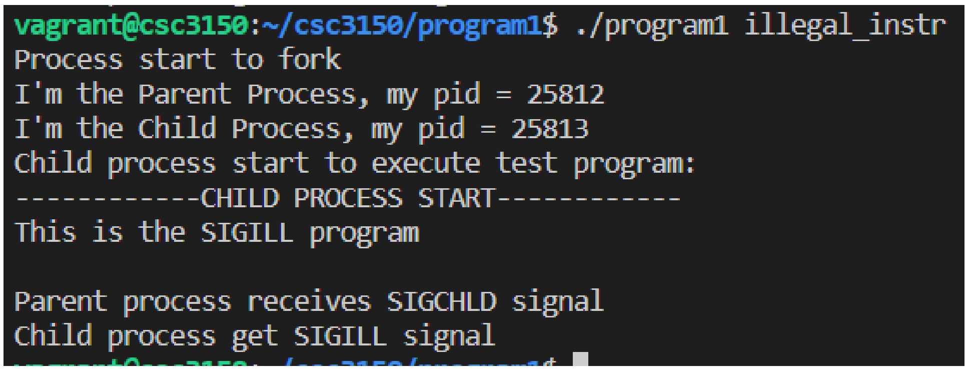
```
vagrant@csc3150:~/csc3150/program1$ ./program1 bus
Process start to fork
I'm the Parent Process, my pid = 25719
I'm the Child Process, my pid = 25720
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGBUS program

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



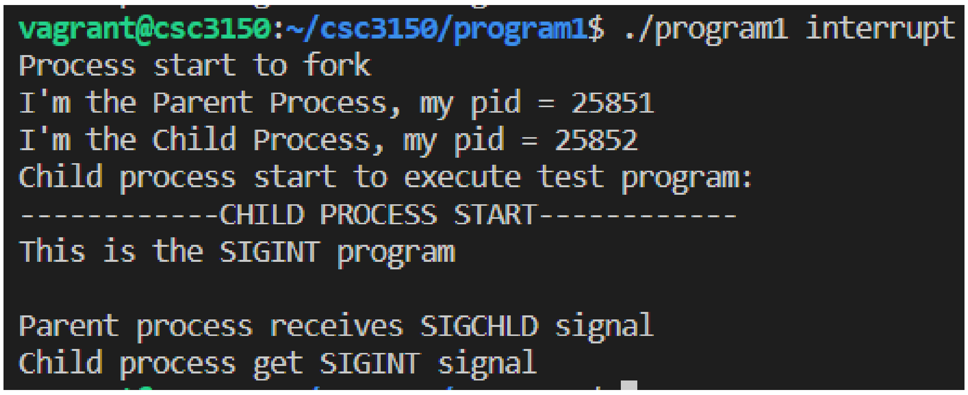
```
vagrant@csc3150:~/csc3150/program1$ ./program1 floating
Process start to fork
I'm the Parent Process, my pid = 25746
I'm the Child Process, my pid = 25747
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGFPE program

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



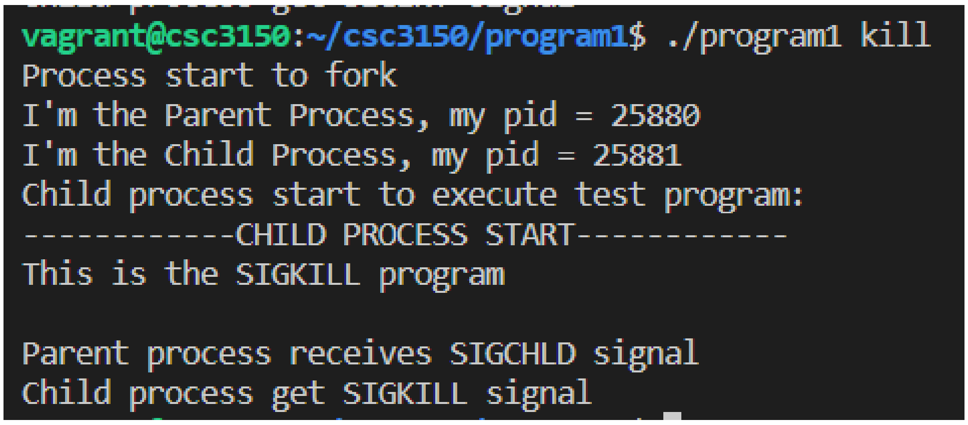
```
vagrant@csc3150:~/csc3150/program1$ ./program1 hangup
Process start to fork
I'm the Parent Process, my pid = 25785
I'm the Child Process, my pid = 25786
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGHUP program

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



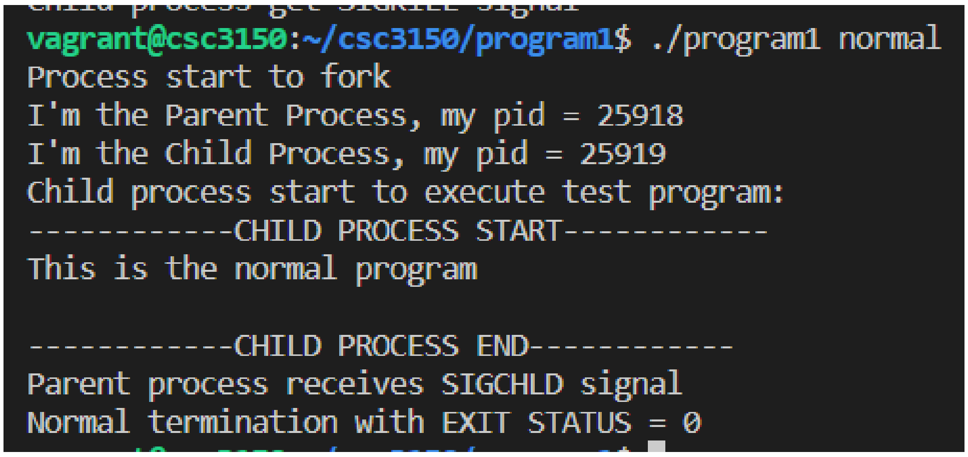
```
vagrant@csc3150:~/csc3150/program1$ ./program1 illegal_instr
Process start to fork
I'm the Parent Process, my pid = 25812
I'm the Child Process, my pid = 25813
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGILL program

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



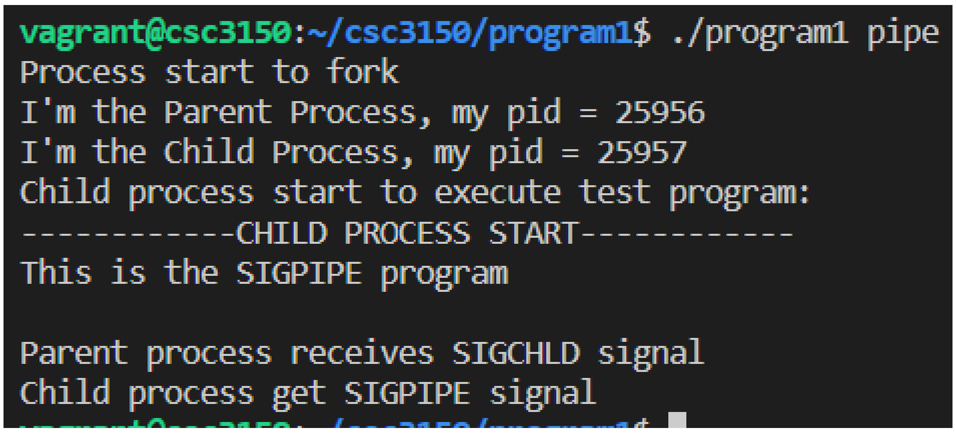
```
vagrant@csc3150:~/csc3150/program1$ ./program1 interrupt
Process start to fork
I'm the Parent Process, my pid = 25851
I'm the Child Process, my pid = 25852
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGINT program

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



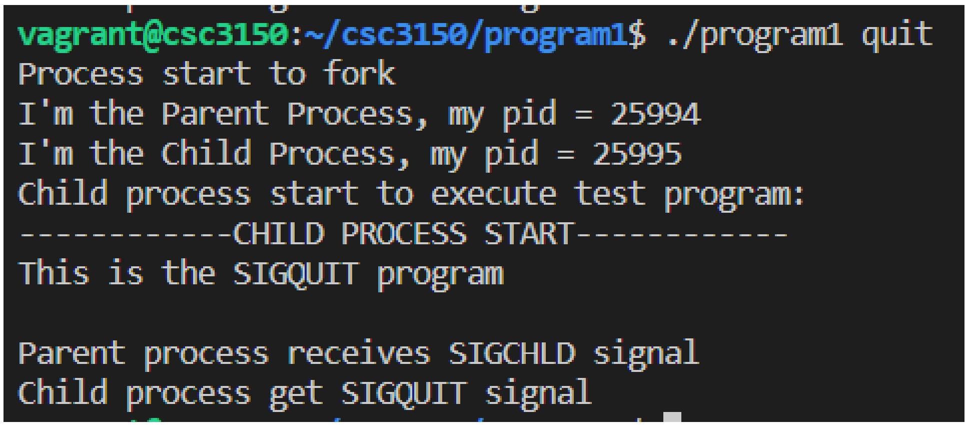
```
vagrant@csc3150:~/csc3150/program1$ ./program1 kill
Process start to fork
I'm the Parent Process, my pid = 25880
I'm the Child Process, my pid = 25881
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGKILL program

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



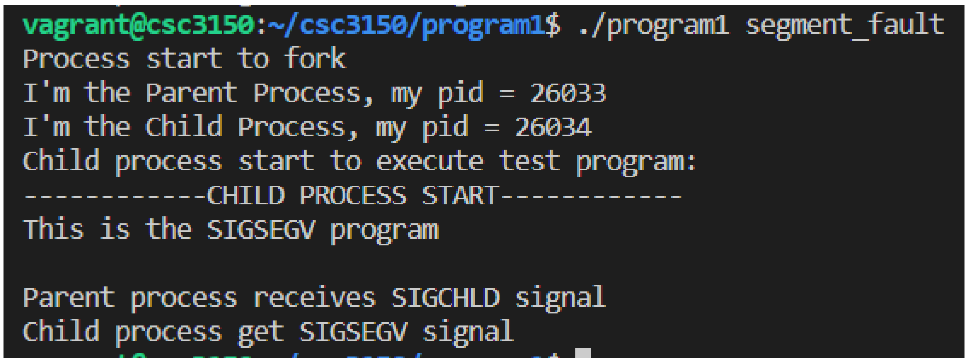
```
vagrant@csc3150:~/csc3150/program1$ ./program1 normal
Process start to fork
I'm the Parent Process, my pid = 25918
I'm the Child Process, my pid = 25919
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the normal program

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



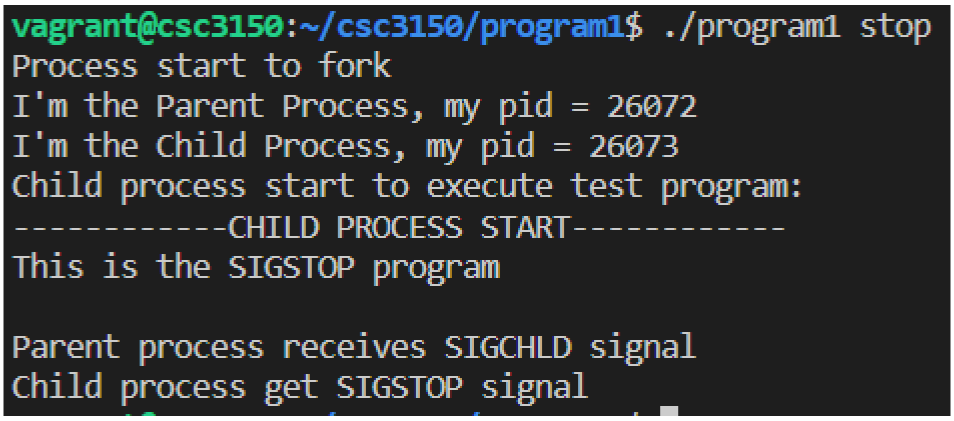
```
vagrant@csc3150:~/csc3150/program1$ ./program1 pipe
Process start to fork
I'm the Parent Process, my pid = 25956
I'm the Child Process, my pid = 25957
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGPIPE program

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



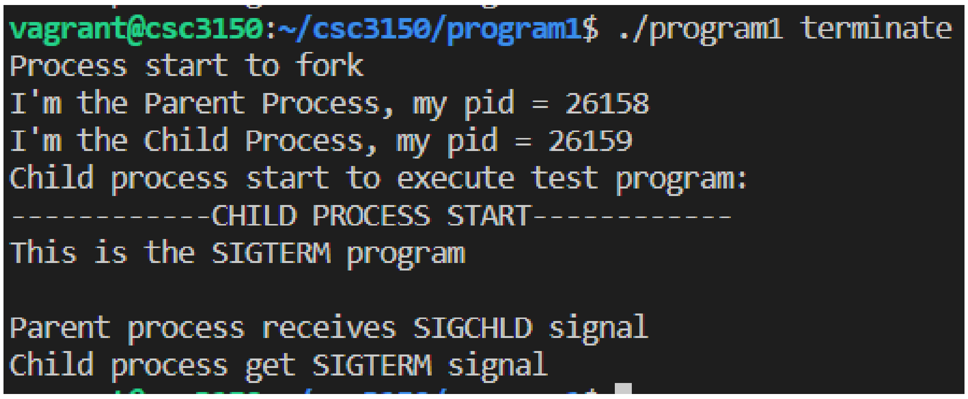
```
vagrant@csc3150:~/csc3150/program1$ ./program1 quit
Process start to fork
I'm the Parent Process, my pid = 25994
I'm the Child Process, my pid = 25995
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

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



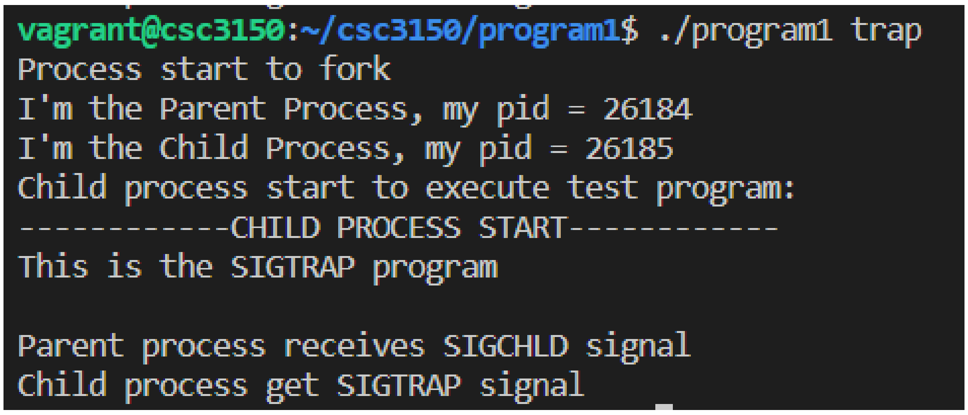
```
vagrant@csc3150:~/csc3150/program1$ ./program1 segment_fault
Process start to fork
I'm the Parent Process, my pid = 26033
I'm the Child Process, my pid = 26034
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

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



```
vagrant@csc3150:~/csc3150/program1$ ./program1 stop
Process start to fork
I'm the Parent Process, my pid = 26072
I'm the Child Process, my pid = 26073
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
```

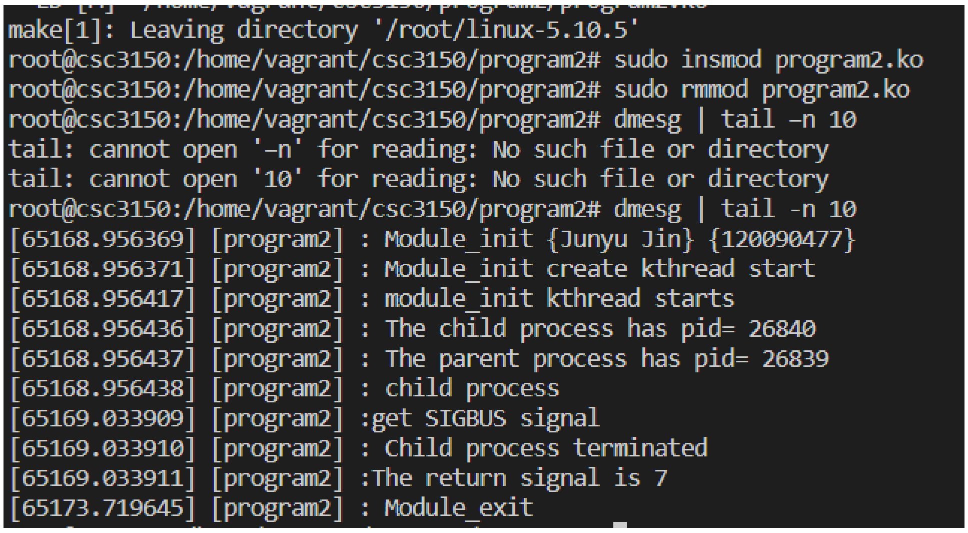


```
vagrant@csc3150:~/csc3150/program1$ ./program1 terminate
Process start to fork
I'm the Parent Process, my pid = 26158
I'm the Child Process, my pid = 26159
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTERM program

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



```
vagrant@csc3150:~/csc3150/program1$ ./program1 trap
Process start to fork
I'm the Parent Process, my pid = 26184
I'm the Child Process, my pid = 26185
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
```

Program2:



```
  LD [M]  /home/vagrant/csc3150/program2/program2.ko
make[1]: Leaving directory '/root/linux-5.10.5'
root@csc3150:/home/vagrant/csc3150/program2# sudo insmod program2.ko
root@csc3150:/home/vagrant/csc3150/program2# sudo rmmod program2.ko
root@csc3150:/home/vagrant/csc3150/program2# dmesg | tail –n 10
tail: cannot open '–n' for reading: No such file or directory
tail: cannot open '10' for reading: No such file or directory
root@csc3150:/home/vagrant/csc3150/program2# dmesg | tail -n 10
[65168.956369] [program2] : Module_init {Junyu Jin} {120090477}
[65168.956371] [program2] : Module_init create kthread start
[65168.956417] [program2] : module_init kthread starts
[65168.956436] [program2] : The child process has pid= 26840
[65168.956437] [program2] : The parent process has pid= 26839
[65168.956438] [program2] : child process
[65169.033909] [program2] :get SIGBUS signal
[65169.033910] [program2] : Child process terminated
[65169.033911] [program2] :The return signal is 7
[65173.719645] [program2] : Module_exit
```