CSC3150 Project 1 by 121090697 Jiayan Yang

1. Setup of the environment

1.1 Basic Environment.

I am using VMware Workstation 17 and my version of ubuntu is `20.04`. My kernel's version is `5.10.197`.

1.2 Modification to the Kernel

As program2 asks us to do a little bit modification to the original kernel, I have done the following changes:

(1). In kernel/fork.c:

Add `EXPORT_SYMBOL(kernel_clone)` after the definition of function `kernel_clone()`.

(2). In kernel/ exit.c:

Add `EXPORT_SYMBOL(do_wait)` after the definition of function `do_wait()`.

Delete the `static` announcement of `do_wait()`.

(3). In fs/exec.c:

Add `EXPORT_SYMBOL(do_execve)` after the definition of function `do_execve()`.

Delete the `static` announcement of `do_execve()`.

(4). In fs/namei.c:

Add `EXPORT_SYMBOL(getname_kernel)` after the definition of function `getname_kernel()`.

2. Task 1

2.1 Program Design

The task requires us ot run a process under user mode. So my program will do it in following procedures:

(1). Use `fork()` to fork a child process in user mode.

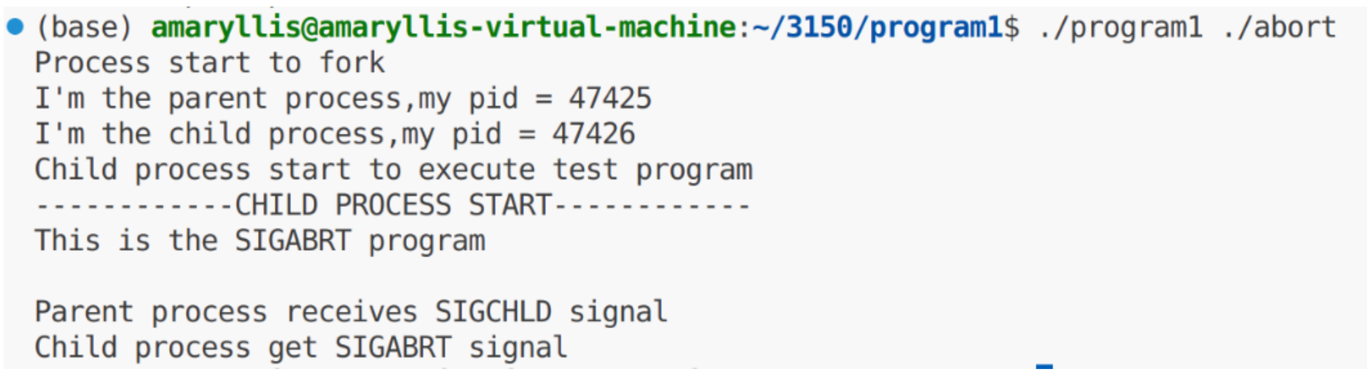
(2). Then use `execl()` to execute the test program in child process

(3). Use `waitpid()` in the main process to to wait the child process and fetch signal.

(4). Process the signal.

2.2 Program Output

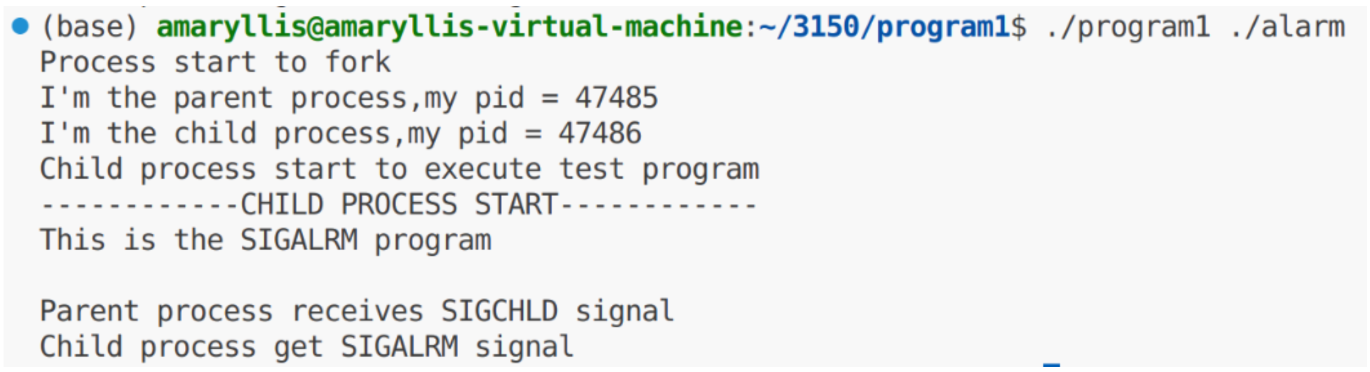
1.abort:



```
● (base) amaryllis@amaryllis-virtual-machine:~/3150/program1$ ./program1 ./abort
  Process start to fork
  I'm the parent process,my pid = 47425
  I'm the child process,my pid = 47426
  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
```

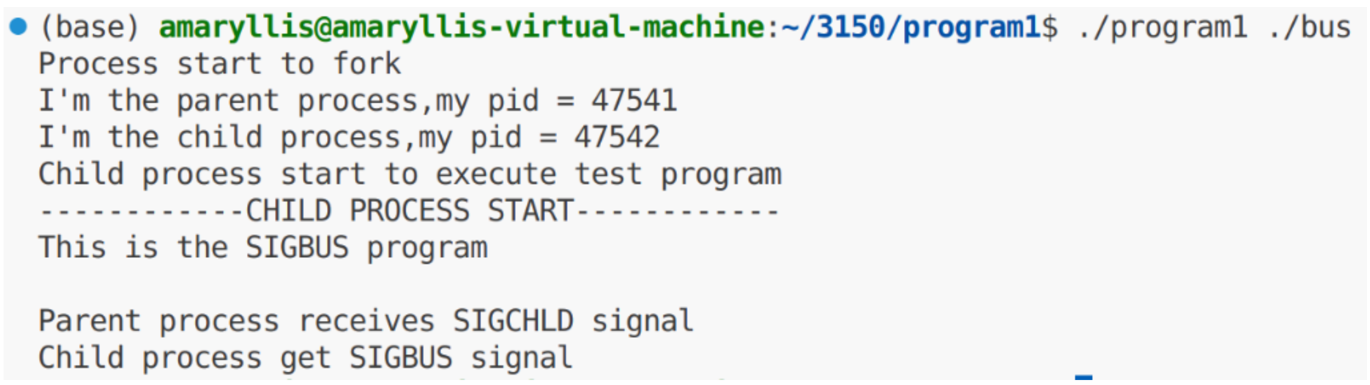
2.alarm:



```
● (base) amaryllis@amaryllis-virtual-machine:~/3150/program1$ ./program1 ./alarm
  Process start to fork
  I'm the parent process,my pid = 47485
  I'm the child process,my pid = 47486
  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
```

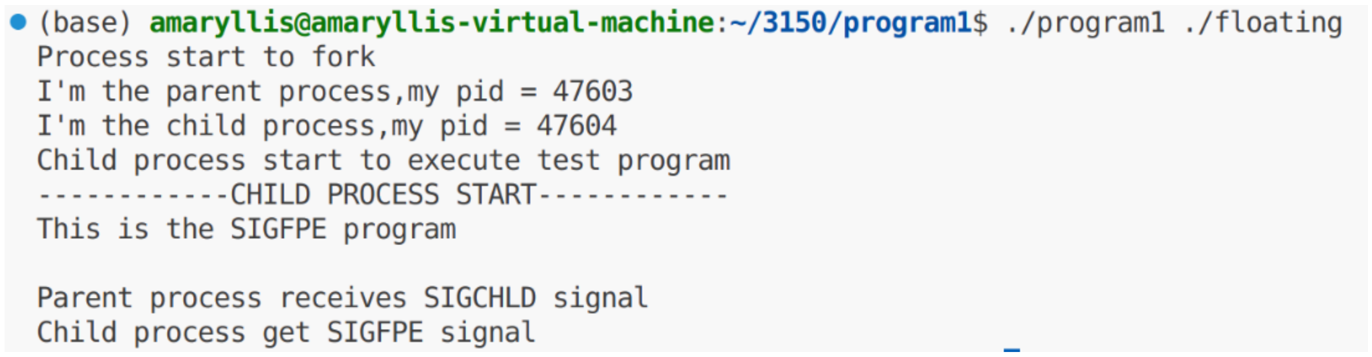
3.bus:



```
● (base) amaryllis@amaryllis-virtual-machine:~/3150/program1$ ./program1 ./bus
  Process start to fork
  I'm the parent process,my pid = 47541
  I'm the child process,my pid = 47542
  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
```

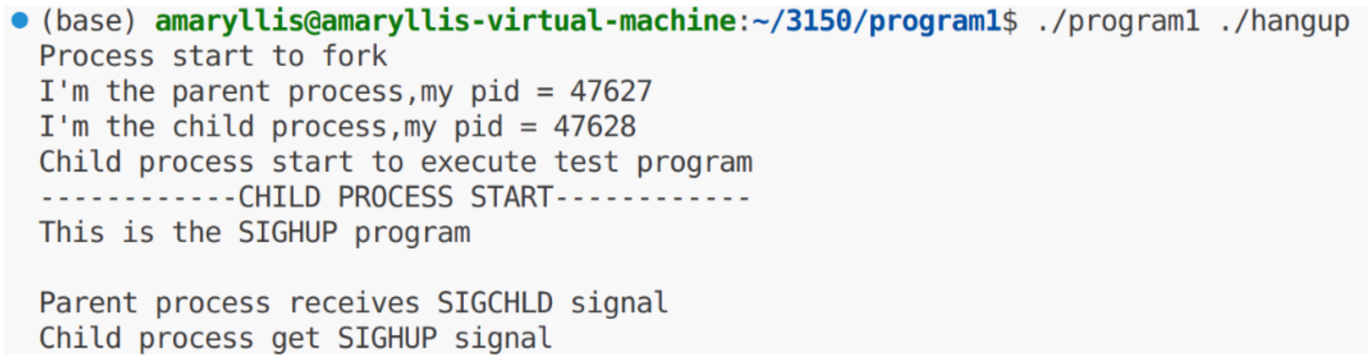
4.floating:



```
● (base) amaryllis@amaryllis-virtual-machine:~/3150/program1$ ./program1 ./floating
  Process start to fork
  I'm the parent process,my pid = 47603
  I'm the child process,my pid = 47604
  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
```

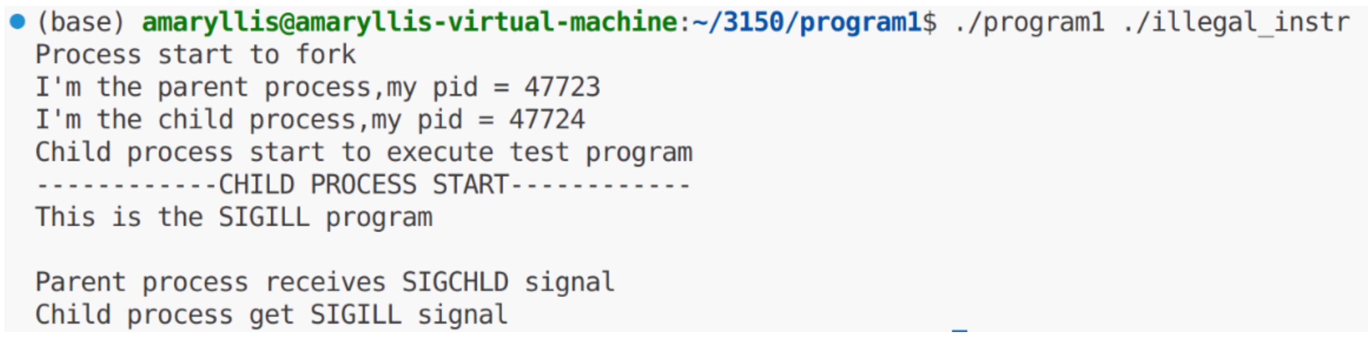
5.hangup:



```
● (base) amaryllis@amaryllis-virtual-machine:~/3150/program1$ ./program1 ./hangup
  Process start to fork
  I'm the parent process,my pid = 47627
  I'm the child process,my pid = 47628
  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
```

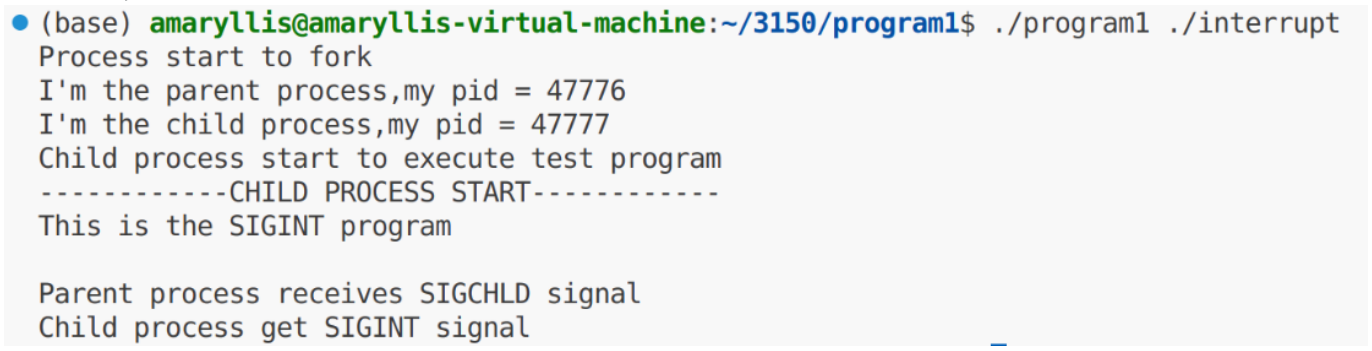
6.illegal_instr:



```
● (base) amaryllis@amaryllis-virtual-machine:~/3150/program1$ ./program1 ./illegal_instr
  Process start to fork
  I'm the parent process,my pid = 47723
  I'm the child process,my pid = 47724
  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
```

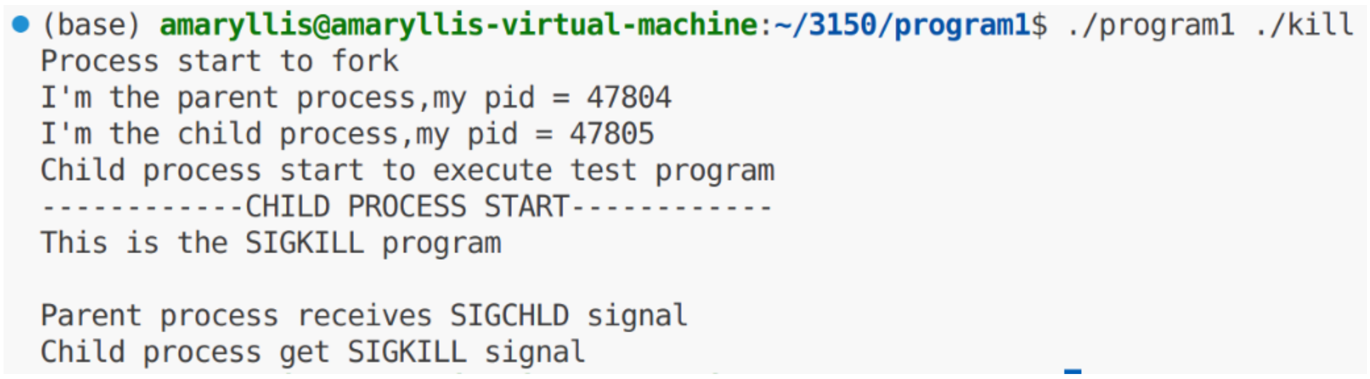
7.interrupt:



```
● (base) amaryllis@amaryllis-virtual-machine:~/3150/program1$ ./program1 ./interrupt
  Process start to fork
  I'm the parent process,my pid = 47776
  I'm the child process,my pid = 47777
  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
```

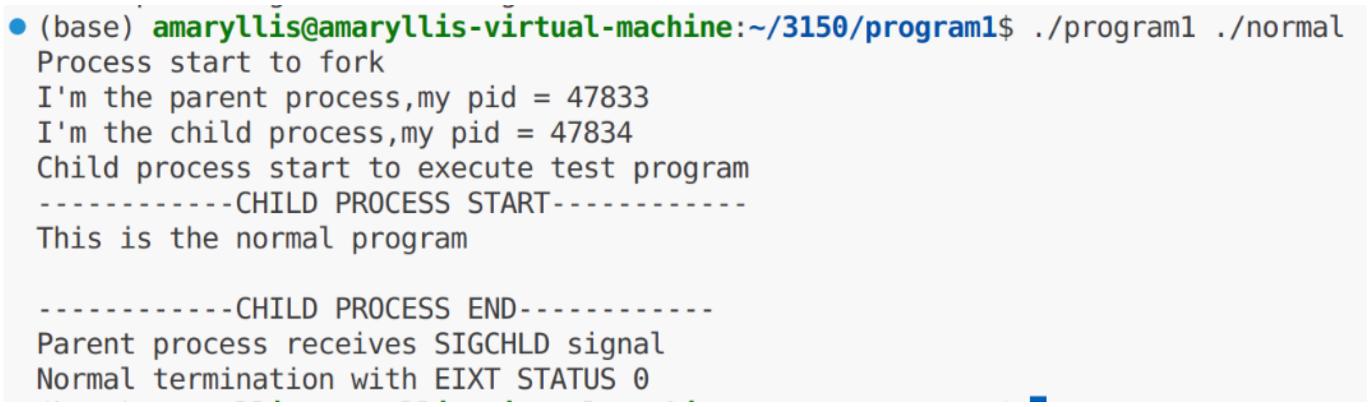
8.kill:



```
(base) amaryllis@amaryllis-virtual-machine:~/3150/program1$ ./program1 ./kill
Process start to fork
I'm the parent process,my pid = 47804
I'm the child process,my pid = 47805
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
```

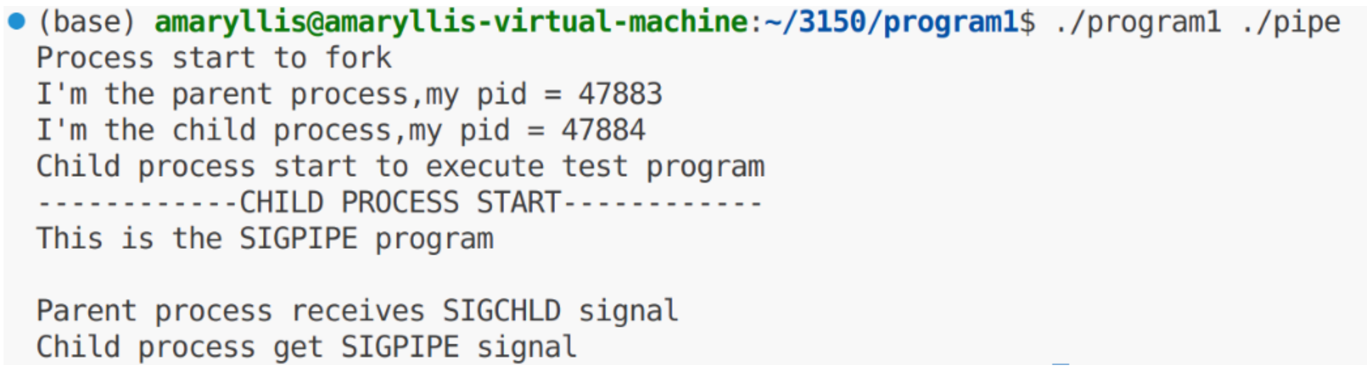
9.normal:



```
(base) amaryllis@amaryllis-virtual-machine:~/3150/program1$ ./program1 ./normal
Process start to fork
I'm the parent process,my pid = 47833
I'm the child process,my pid = 47834
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 EIXT STATUS 0
```

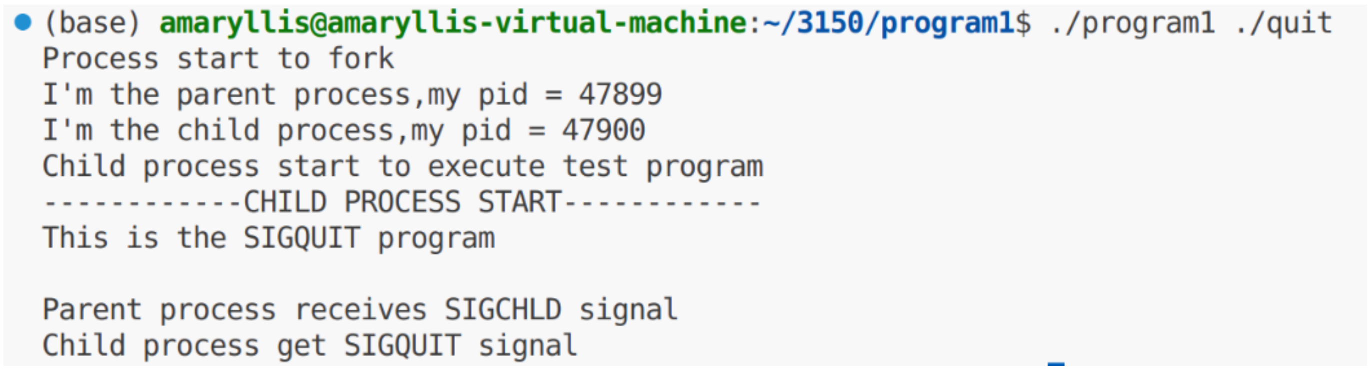
10.pipe:



```
(base) amaryllis@amaryllis-virtual-machine:~/3150/program1$ ./program1 ./pipe
Process start to fork
I'm the parent process,my pid = 47883
I'm the child process,my pid = 47884
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
```

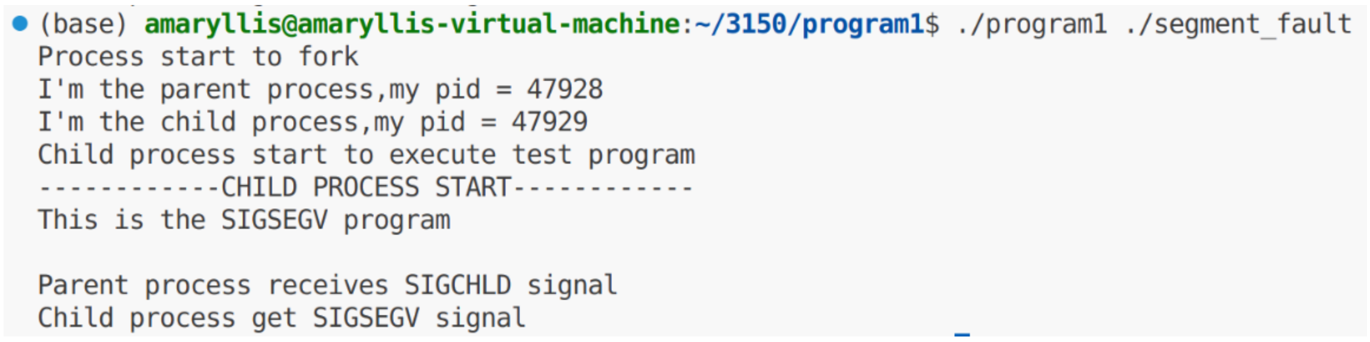
11.quit:



```
(base) amaryllis@amaryllis-virtual-machine:~/3150/program1$ ./program1 ./quit
Process start to fork
I'm the parent process,my pid = 47899
I'm the child process,my pid = 47900
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
```

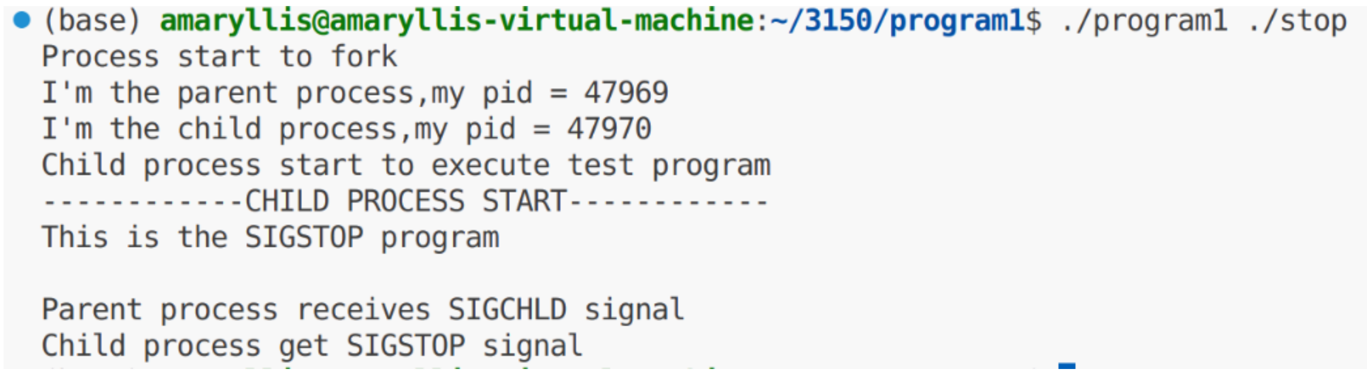
12.segment:



```
● (base) amaryllis@amaryllis-virtual-machine:~/3150/program1$ ./program1 ./segment_fault
  Process start to fork
  I'm the parent process,my pid = 47928
  I'm the child process,my pid = 47929
  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
```

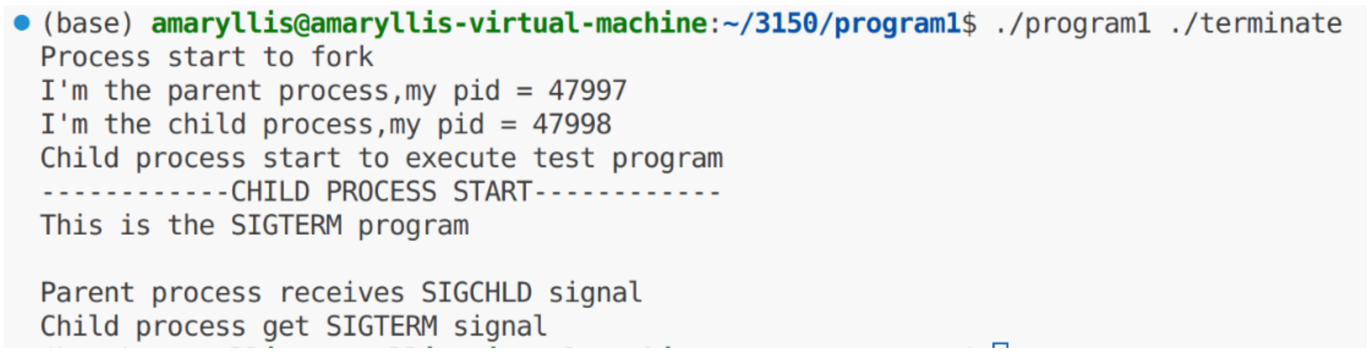
13.stop:



```
● (base) amaryllis@amaryllis-virtual-machine:~/3150/program1$ ./program1 ./stop
  Process start to fork
  I'm the parent process,my pid = 47969
  I'm the child process,my pid = 47970
  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
```

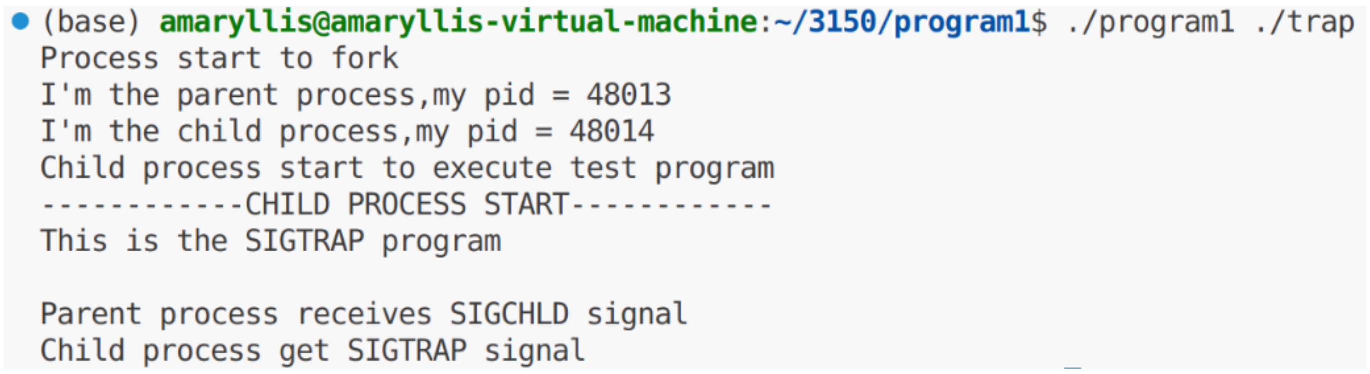
14.terminate:



```
● (base) amaryllis@amaryllis-virtual-machine:~/3150/program1$ ./program1 ./terminate
  Process start to fork
  I'm the parent process,my pid = 47997
  I'm the child process,my pid = 47998
  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
```

15.trap:



```
● (base) amaryllis@amaryllis-virtual-machine:~/3150/program1$ ./program1 ./trap
  Process start to fork
  I'm the parent process,my pid = 48013
  I'm the child process,my pid = 48014
  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
```

2.3 What I learnt from the task

I have learnt how to creat a child process and make use of them, how to execute other programs in the process by using `execve()`, and also some knowledge on how to deal with the signal sent by child process.

3. Program 2

3.1 Program Design

(1). Use `kthread_create()` to create a kernel thread.

(2). Fork the process by using `my_fork()`.

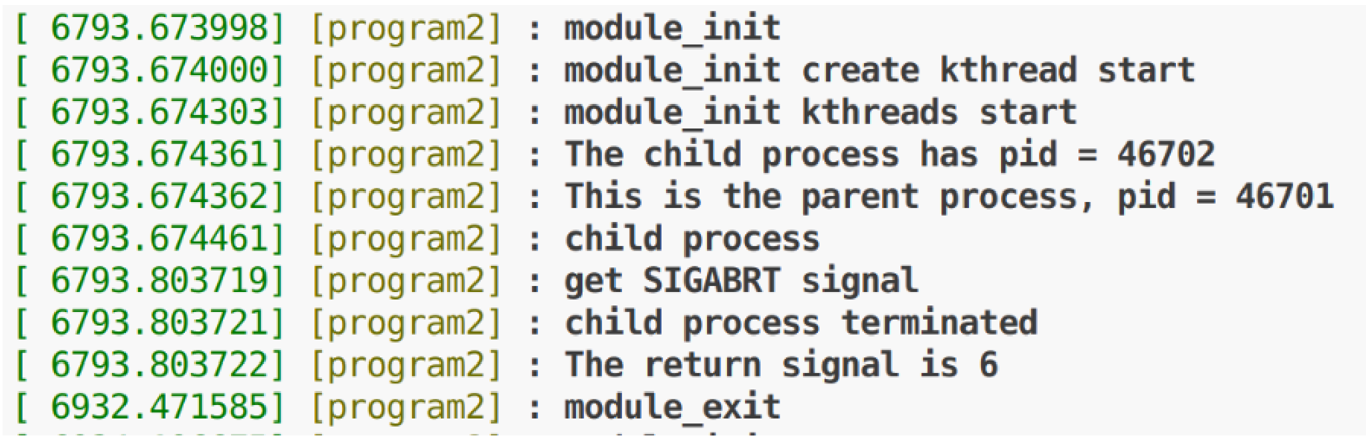
(3). Execute the test program in the child process by using `do_execve()`.

(4). Create a new thread by `kernel_clone()`.

(5). Use `my_wait()` function to wait the signal from the child process, and then Process it.

3.2 Program Output

I use SIGABRT signal to test it, and it works well:



```
[ 6793.673998] [program2] : module_init
[ 6793.674000] [program2] : module_init create kthread start
[ 6793.674303] [program2] : module_init kthreads start
[ 6793.674361] [program2] : The child process has pid = 46702
[ 6793.674362] [program2] : This is the parent process, pid = 46701
[ 6793.674461] [program2] : child process
[ 6793.803719] [program2] : get SIGABRT signal
[ 6793.803721] [program2] : child process terminated
[ 6793.803722] [program2] : The return signal is 6
[ 6932.471585] [program2] : module_exit
```

3.3 What I learnt from the task

(1). How to modify kernel files and install a new kernel.

(2). How to insert/remove modules to kernel.

4. Bonus

Due to limited skills I gave up the bonus part.