Jiale Zhong

121040084

Report

Program 1

Design

The program will fork a child process to execute the test program. As a result, I use fork() function to create a child process at first.

Then the fork() function will return the value to the variable “pid”.

If fork() return 0 (stored in “pid”) to the parent process, it means it successfully creates a child process. If fork() return -1 to the parent process, it means no child process is created. If fork() return the process ID of the child process to the parent process (neither 0 nor 1), it means it is the parent process.

If pid equals neither 0 nor 1, I use waitpid() function to wait for child process to finish execution and then print out the child process termination information including how did the child process terminates and what signal was raised in child process.

Notice: I use WUNTRACED as the third parameter of waitpid() so that it can wait for child process to stop or terminate.

Environment Set Up and Program Compilation

Linux Distribution: Ubuntu 16.04.12

Linux Kernel Version: 5.10.145

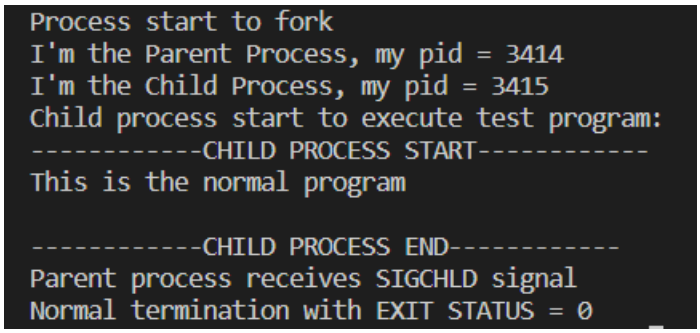
gcc version 5.4.0

1. Go to the source folder
2. Go to the program1 folder

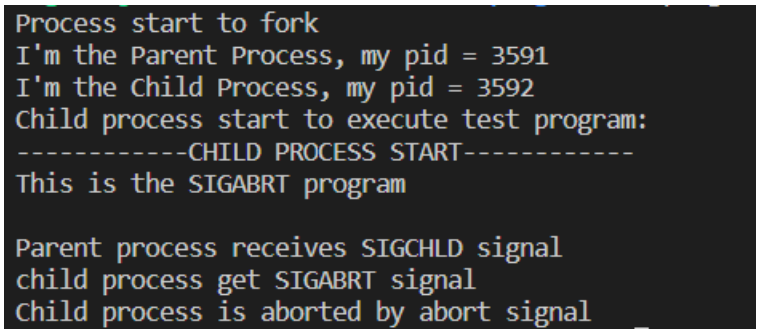
3. Use command gcc <program_name>.c -o <program_name> to compile program1.c and other test files.
4. Use "./program1 ./<test_file_name>" to run the program.

Screenshot

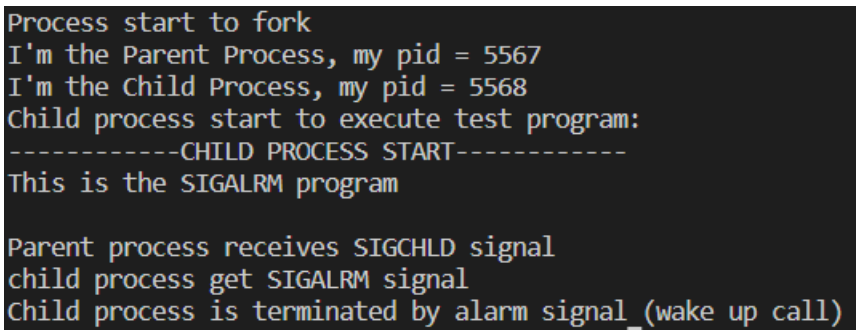
normal



abort



alarm



bus

```
Process start to fork
I'm the Parent Process, my pid = 3935
I'm the Child Process, my pid = 3936
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
Child process is terminated because it has bus error
```

floating

```
Process start to fork
I'm the Parent Process, my pid = 4007
I'm the Child Process, my pid = 4008
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
Child process is terminated because it has floating-point exception
```

hangup

```
Process start to fork
I'm the Parent Process, my pid = 4172
I'm the Child Process, my pid = 4173
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
Child process is hung up by hangup signal
```

illegal_inst

```
Process start to fork
I'm the Parent Process, my pid = 4251
I'm the Child Process, my pid = 4252
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
Child process is terminated because it has illegal instruction
```

interrupt

```
Process start to fork
I'm the Parent Process, my pid = 4295
I'm the Child Process, my pid = 4296
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
Child process is interrupted by interrupt signal
```

kill

```
Process start to fork
I'm the Parent Process, my pid = 4335
I'm the Child Process, my pid = 4336
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
Child process is killed by kill signal
```

pipe

```
Process start to fork
I'm the Parent Process, my pid = 4392
I'm the Child Process, my pid = 4393
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
Child process is terminated by pipe signal
```

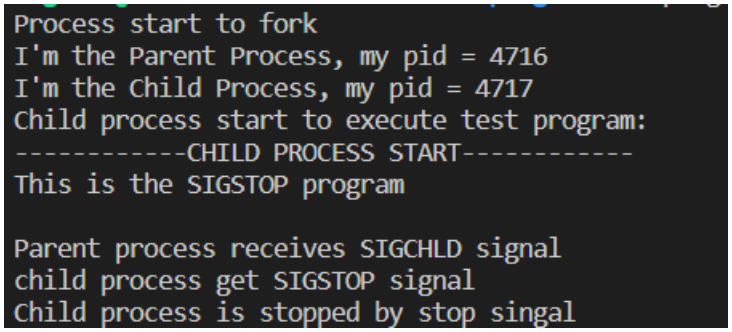
quit

```
Process start to fork
I'm the Parent Process, my pid = 4446
I'm the Child Process, my pid = 4447
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
Child process is quitted by quit signal
```

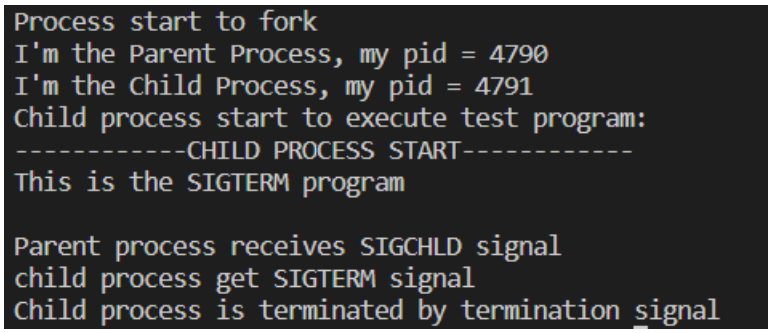
segment_fault

```
Process start to fork
I'm the Parent Process, my pid = 4504
I'm the Child Process, my pid = 4505
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

Parent process receives SIGCHLD signal
child process get SIGASEGV signal
Child process is terminated because it has segment fault
```

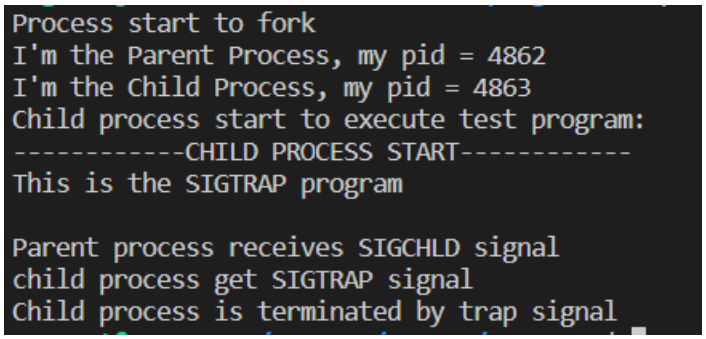
stop



termination



trap



What I learn from the task

I learn the mechanism of creating a child process and the way user mode communicates with kernel mode through the fork() function. Also I learn the meaning of different signals raised from the child process. Other than that, I learn different macros such as

WIFSIGNALED and WIFSTOPPED to analyze the status referenced by the argument in order to properly receive signal.

Program 2

Design

The program will export some functions including kernel_clone(), do_wait(), do_execve(), and getname_kernel() from the kernel source code using “EXPORT SYMBOL <symbol_name>” and implement a kernel module to insert into the kernel.

I use kthread_create() to create a kernel thread and execute my_fork() function.

Within my_fork() function, assign necessary arguments into the kernel_clone_args named args. Then pass args into kernel_clone() function to create a child process to execute my_exec() function.

Within my_exec() function, call getname_kernel() function to get the absolute path of the test file then pass it into do_exec() function to execute the test program.

Within my_fork() function, call my_wait() function to wait for the child process to terminate or stop.

Within my_wait() function, first struct wait_opts named wo and assign necessary value to the variables in “wo”. Then pass “wo” to the do_wait() function to wait for the child process to terminate. Then print out the type of signal and its related messages.

Environment Set Up and Kernel Compilation

Linux Distribution: Ubuntu 16.04.12

Linux Kernel Version: 5.10.145

gcc version 5.4.0

1. Go to the kernel source code (located in the folder of linux-5.10.145 in my case)

2. Find the location of the 4 functions within the folder: kernel_clone(), do_wait(), do_execve(), and getname_kernel()
3. Export the functions using “EXPORT_SYMBOL(function_name)” after the functions in the source code.
4. Start to compile the kernel:
    1. Login to root account so that I have the permission using “sudo su” command
    2. Use command “make bzImage -j$(nproc)” to build the changes
    3. Use command “make modules -j$(nproc)”
    4. Use command “make modules_install” to install kernel modules then “make install” to install kernel.
    5. Reboot the virtual machine
    6. Login for another time with command “sudo su”
    7. Go to the folder “program2”
    8. Use the command “make”
    9. Use the command “insmod program2.ko” to insert the module
    10. Use the command “rmmod program2.ko” to remove the module
    11. Finally, use the command “dmesg” or “dmesg |grep program2”to view the message.

    Notice: we need to first compile all the test files as the way as mentioned above.

test

```
[21006.013152] [program2] : module_init {Zhong Jiale} {121040084}
[21006.013153] [program2] : module_init create kthread start
[21006.013256] [program2] : module_init kthread start
[21006.013315] [program2] : The child process has pid = 8303
[21006.013316] [program2] : This is the parent process, pid = 8302
[21006.013327] [program2] : child process
[21006.143492] [program2] : get SIGBUS signal
[21006.143494] [program2] : child process terminated because of bus error
[21006.143494] [program2] : The return signal is 7
[21010.824685] [program2] : module_exit
```

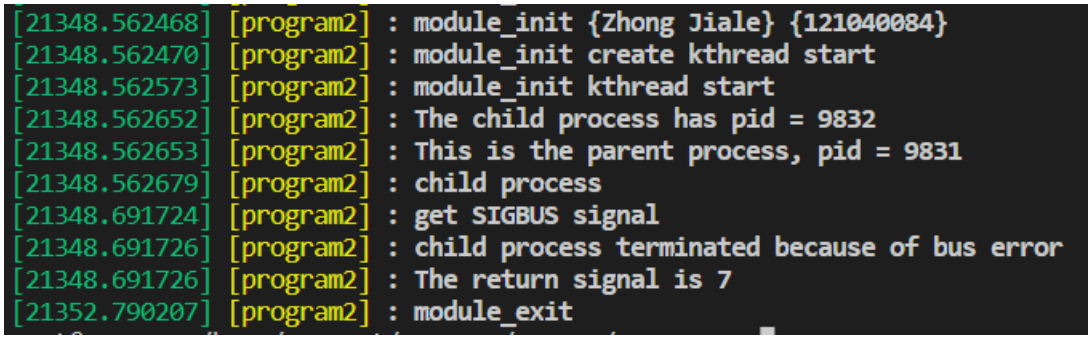
abort

```
[21098.433016] [program2] : module_init {Zhong Jiale} {121040084}
[21098.433017] [program2] : module_init create kthread start
[21098.433136] [program2] : module_init kthread start
[21098.433203] [program2] : The child process has pid = 8773
[21098.433204] [program2] : This is the parent process, pid = 8772
[21098.433218] [program2] : child process
[21098.562106] [program2] : get SIGABRT signal
[21098.562108] [program2] : child process aborted
[21098.562108] [program2] : The return signal is 6
[21104.788827] [program2] : module_exit
```

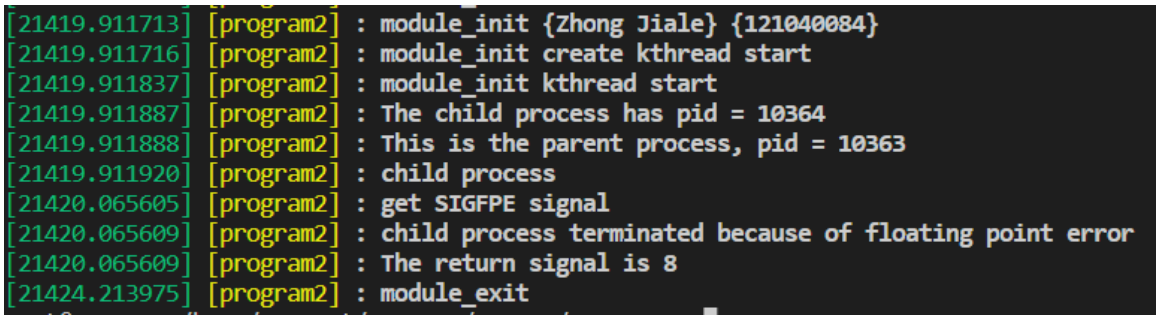
alarm

```
[21268.814719] [program2] : module_init {Zhong Jiale} {121040084}
[21268.814738] [program2] : module_init create kthread start
[21268.814905] [program2] : module_init kthread start
[21268.814992] [program2] : The child process has pid = 9328
[21268.814993] [program2] : This is the parent process, pid = 9327
[21268.815011] [program2] : child process
[21270.815939] [program2] : get SIGALRM signal
[21270.815943] [program2] : child process terminated by alarm signal (wake up call)
[21270.815944] [program2] : The return signal is 14
[21272.944017] [program2] : module_exit
```

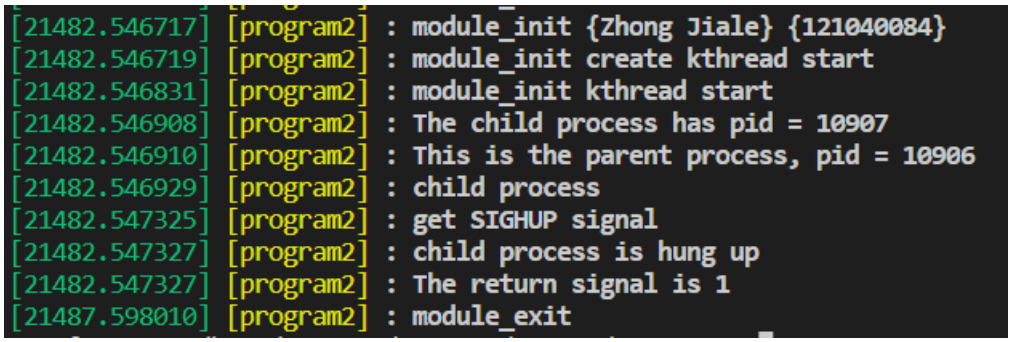
bus



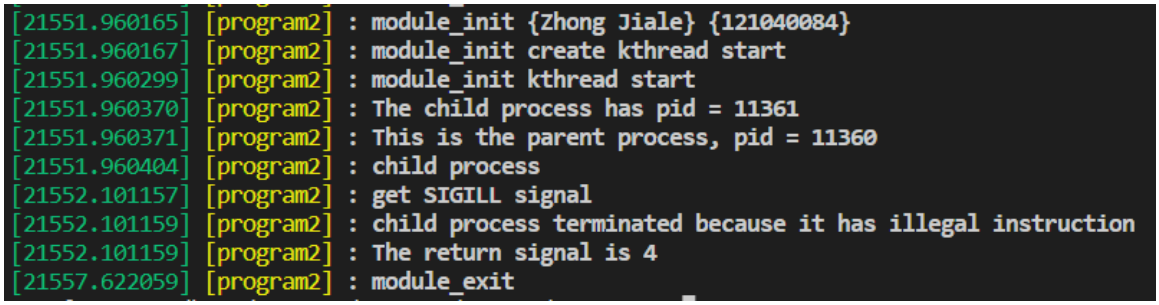
floating



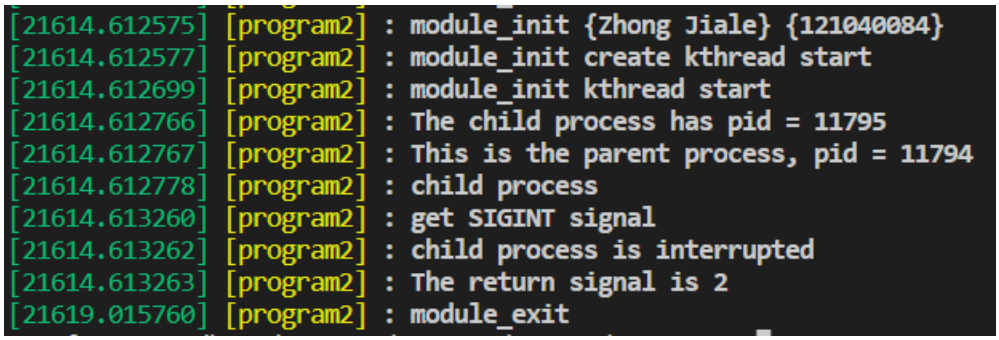
hangup



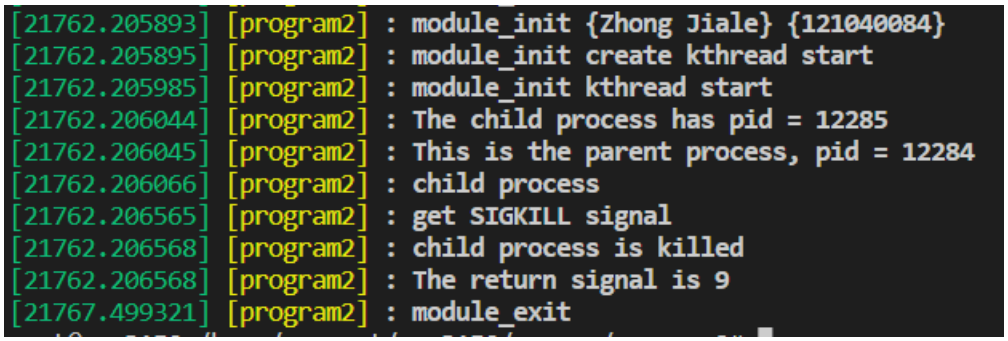
illegal_inst



interrupt



kill



normal

```
[21823.209630] [program2] : module_init {Zhong Jiale} {121040084}
[21823.209632] [program2] : module_init create kthread start
[21823.209756] [program2] : module_init kthread start
[21823.209810] [program2] : The child process has pid = 12751
[21823.209812] [program2] : This is the parent process, pid = 12750
[21823.209843] [program2] : child process
[21823.210315] [program2] : child process terminated normally
[21823.210317] [program2] : The return signal is 0
[21827.743527] [program2] : module_exit
```

pipe

```
[22035.727049] [program2] : module_init {Zhong Jiale} {121040084}
[22035.727051] [program2] : module_init create kthread start
[22035.727166] [program2] : module_init kthread start
[22035.727224] [program2] : The child process has pid = 13263
[22035.727225] [program2] : This is the parent process, pid = 13262
[22035.727235] [program2] : child process
[22035.727714] [program2] : get SIGPIPE signal
[22035.727716] [program2] : child process terminated by pipe signal
[22035.727717] [program2] : The return signal is 13
[22040.053029] [program2] : module_exit
```

quit

```
[22093.915660] [program2] : module_init {Zhong Jiale} {121040084}
[22093.915662] [program2] : module_init create kthread start
[22093.915783] [program2] : module_init kthread start
[22093.915844] [program2] : The child process has pid = 13710
[22093.915846] [program2] : This is the parent process, pid = 13709
[22093.915861] [program2] : child process
[22094.049566] [program2] : get SIGQUIT signal
[22094.049569] [program2] : child process quited
[22094.049569] [program2] : The return signal is 3
[22098.290919] [program2] : module_exit
```

segment_fault

```
[22144.691415] [program2] : module_init {Zhong Jiale} {121040084}
[22144.691417] [program2] : module_init create kthread start
[22144.691512] [program2] : module_init kthread start
[22144.691579] [program2] : The child process has pid = 14138
[22144.691580] [program2] : This is the parent process, pid = 14137
[22144.691594] [program2] : child process
[22144.828958] [program2] : get SIGSEGV signal
[22144.828960] [program2] : child process terminated because it has segment fault
[22144.828960] [program2] : The return signal is 11
[22148.950963] [program2] : module_exit
```

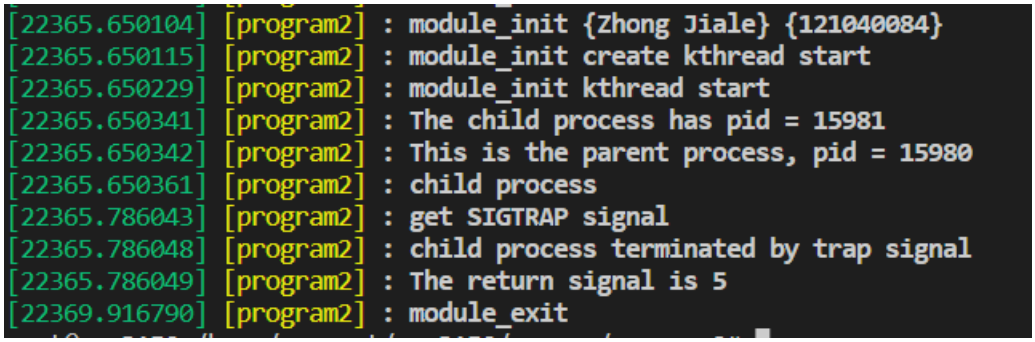
stop

```
[22192.212462] [program2] : module_init {Zhong Jiale} {121040084}
[22192.212464] [program2] : module_init create kthread start
[22192.212604] [program2] : module_init kthread start
[22192.212681] [program2] : The child process has pid = 14591
[22192.212684] [program2] : This is the parent process, pid = 14590
[22192.212696] [program2] : child process
[22192.213084] [program2] : get SIGSTOP signal
[22192.213086] [program2] : child process stopped
[22192.213086] [program2] : The return signal is 19
[22196.984967] [program2] : module_exit
```

termination

```
[22288.293419] [program2] : module_init {Zhong Jiale} {121040084}
[22288.293421] [program2] : module_init create kthread start
[22288.293577] [program2] : module_init kthread start
[22288.293634] [program2] : The child process has pid = 15440
[22288.293636] [program2] : This is the parent process, pid = 15439
[22288.293707] [program2] : child process
[22288.294346] [program2] : get SIGTERM signal
[22288.294348] [program2] : child process terminated
[22288.294348] [program2] : The return signal is 15
[22294.179068] [program2] : module_exit
```

trap



What I learn from the task

I learn how to export function from kernel source code and modification of kernel source code. Also I learn the mechanism of creating child process and waiting for the termination or stop of the child process in the kernel mode through reading the kernel source code.