Report

120090738 Bao Rui

[program 1]

The main task of program1 is to fork a child process to execute the test program, and display the return signal the parent process received from the child process.

1.1 Design of program1

Firstly, function “fork()” is used to generate a new process, which is an exact copy of the original parent process, except for it has its own pid.

Then we use the value of pid to separately deal with parent process and child process. The pid = 0, indicates it is the child process that is under execution, while if the pid is a positive value, it indicates the pid of the child process, which will send to the parent process.

In the child process, “execve()” is used to run an executable file in the context of an already existing process, replacing the previous execution. The child process, thus, can do different tasks than the parent process. Unless the replacement failed, the execve() will not return any value.

In the parent process, “wait_pid(pid_t pid, int * status_ptr, nt option)” is the used to get output data from the child process. By using this system call, the parent process is able to decode the information of child process execution. The option parameter is set to “WUNTRACED”, which reports on stopped child process as well as terminated ones. Finally, the parent process can print out the information about the child process.

2.1 Development environment

Linux Distribution: 16.04.7

Linux Kernel Version: 5.10.146

GCC Version: 5.4.0

2.2 How to Run program1

A. Go to “/csc3150/source/program1”

B. Type “make”

C. There are two approaches:

c.1 type “./progam1 ./test_file_name” in the terminal. This way can test the program one by one. And “test_file_name” here should be changed to the target file name, such as “quit” or “stop”

c.2 type “python3 autotester" which is an automatic testing program to execute all 15 test programs once. You will tab “newline” button in the keyboard to continue different executions.

2.3 Sample outputs:

SIGINT

```
cc -o trap trap.c
root@csc3150:/home/vagrant/csc3150/source/program1# python3 autotester.py
['kill.c', 'interrupt.c', 'terminate.c', 'interrupt', 'pipe', 'trap.c', 'abort.c', 'kill', 'illegal_instr.c', 'stop', 'stop.c', 'trap', 'normal.c', 'bus.c', 'abor
t', 'floating', 'segment_fault', 'program1.c', 'quit.c', 'pipe.c', 'program1', 'segment_fault.c', 'alarm.c', 'normal', 'illegal_instr', 'program1_old.c', 'quit',
'autotester.py', 'hangup', 'Makefile', 'floating.c', 'alarm', 'bus', 'hangup.c', 'program1_old', 'terminate']
['interrupt', 'pipe', 'kill', 'stop', 'trap', 'abort', 'floating', 'segment_fault', 'normal', 'illegal_instr', 'quit', 'hangup', 'alarm', 'bus', 'terminate']
15
=====1/15 starts =====
Process start to fork
I'm the Parent Process, my pid = 8628
I'm the Child Process, my pid = 8629
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGINT program

Parent process receives SIGCHILD signal
child process get SIGINT signal.
=====1/15 finished =====
```

SIGPIPE

```
======2/15 starts ======
Process start to fork
I'm the Parent Process, my pid = 8679
I'm the Child Process, my pid = 8680
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGPIPE program

------------CHILD PROCESS END------------
Parent process receives SIGCHILD signal
Normal termination with EXIT STATUS = 0
======2/15 finished ======
```

SIGKILL

```
======3/15 starts ======
Process start to fork
I'm the Parent Process, my pid = 8763
I'm the Child Process, my pid = 8764
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGKILL program

Parent process receives SIGCHILD signal
child process get SIGKILL signal.
======3/15 finished ======
```

SIGSTOP

```
======4/15 starts ======
Process start to fork
I'm the Parent Process, my pid = 8790
I'm the Child Process, my pid = 8791
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

Parent process receives SIGCHILD signal
child process get stop signal.
======4/15 finished ======
```

SIGTRAP

```
======5/15 starts ======
Process start to fork
I'm the Parent Process, my pid = 8817
I'm the Child Process, my pid = 8818
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTRAP program

Parent process receives SIGCHILD signal
child process get SIGTRAP signal.
======5/15 finished ======
```

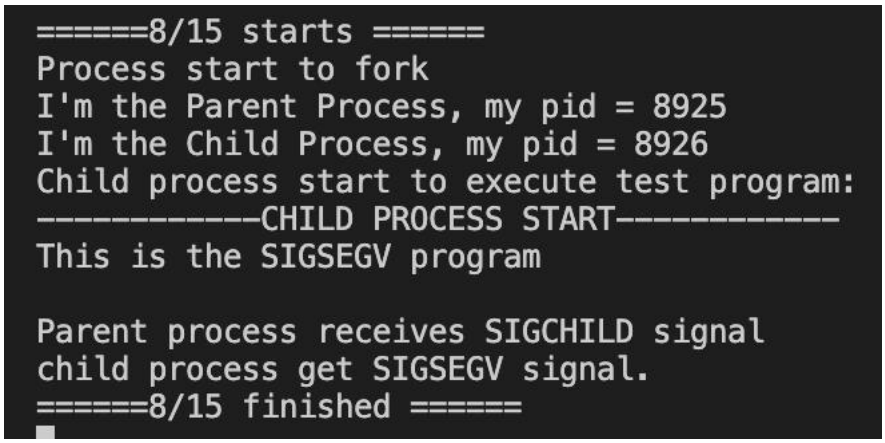
SIGABRT

```
======6/15 starts ======
Process start to fork
I'm the Parent Process, my pid = 8869
I'm the Child Process, my pid = 8870
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGABRT program

Parent process receives SIGCHILD signal
child process get SIGABRT signal.
======6/15 finished ======
```

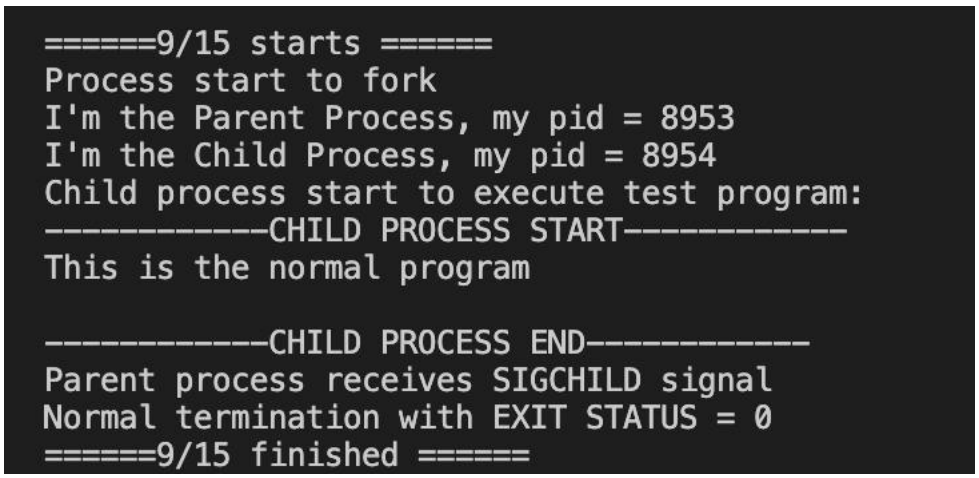
SIGFPE

```
======7/15 starts ======
Process start to fork
I'm the Parent Process, my pid = 8897
I'm the Child Process, my pid = 8898
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGFPE program

Parent process receives SIGCHILD signal
child process get SIGFPE signal.
======7/15 finished ======
```

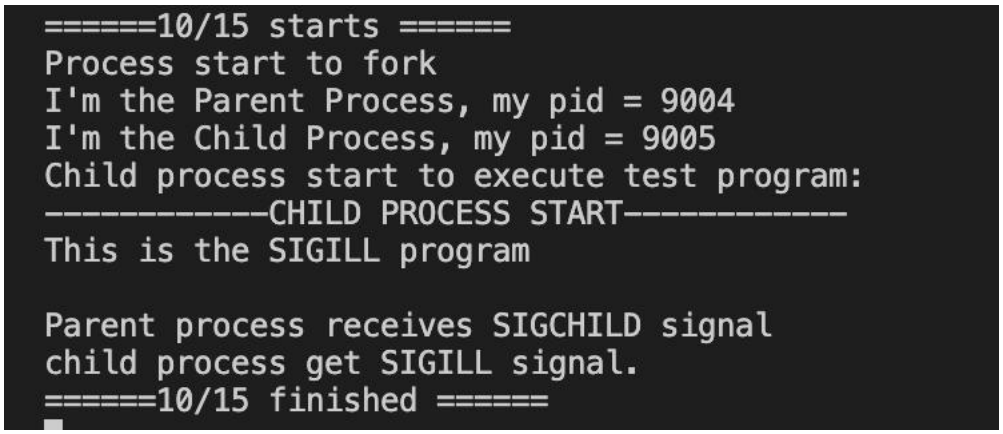
SIGSEGV



NORMAL



SIGILL



SIGQUIT

```
======11/15 starts ======
Process start to fork
I'm the Parent Process, my pid = 9056
I'm the Child Process, my pid = 9057
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

Parent process receives SIGCHILD signal
child process get SIGQUIT signal.
======11/15 finished ======
```

SIGHUP

```
======12/15 starts ======
Process start to fork
I'm the Parent Process, my pid = 9108
I'm the Child Process, my pid = 9109
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGHUP program

Parent process receives SIGCHILD signal
child process get SIGHUP signal.
======12/15 finished ======
```

SIGALRM

```
======13/15 starts ======
Process start to fork
I'm the Parent Process, my pid = 9148
I'm the Child Process, my pid = 9149
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGALRM program

Parent process receives SIGCHILD signal
child process get SIGALRM signal.
======13/15 finished ======
```

SIGBUS

```
======14/15 starts ======
Process start to fork
I'm the Parent Process, my pid = 9205
I'm the Child Process, my pid = 9206
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGBUS program

Parent process receives SIGCHILD signal
child process get SIGBUS signal.
======14/15 finished ======
```

SIGTERM

```
======15/15 starts ======
Process start to fork
I'm the Parent Process, my pid = 9257
I'm the Child Process, my pid = 9258
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTERM program

Parent process receives SIGCHILD signal
child process get SIGTERM signal.
======15/15 finished ======
```

[program2]

This task is to fork a child process in kernel mode. Like program 1, parent process will wait for the child process' termination and display its execution information, only in kernel space.

1.1 Design of program2

[Preparation stage]

since we need to use "kernel_thread()", "do_exeve()", "getname_kernel()" and "do_wait()", first we need to go to their source files to add line "EXPORT_SYMBOL" and recompile the kernel source code.

[Coding logic]

First, "kernel_thread_create()" is used to generate a kernel thread to run "my_fork" function, which is a function used to fork a child process in kernel.

In "my_fork", "kernel_thread()" is invoked to execute a self-defined function "my_work". In the meanwhile, the return value of kernel_thread() is the pid of the child process. There is also a self-define "my_wait()" function, which is similar to the "wait()" function in task 1, for parent process to collect information from the child

process.

"my_work" is for execution of the child process. First, using the "getname_kernel()" function to generate an executable file structure, and pass it to "do_exec()" for reading and executing the file's content. The "do_exec()" function returns 0 if execution is successful.

"my_wait()" invokes "do_wait" system call, and bind child pid with a struct. After collecting the status of the child process, my_wait() returns it to "my_fork()" function.

Back to "my_fork", when the status of the child process is ready, it will be passed to "display" function, which is a function to decode the status value and display the information.

2.1 Development environment

Linux Distribution: 16.04.7
Linux Kernel Version: 5.10.146
GCC Version: 5.4.0

2.3 How to run program2

0. Type in "sudo su"
1. Go to linux source files to modify "EXPORT_SYMBOL()" to export"kernel_thread()", "do_exeve()", "getname_kernel()" and "do_wait()".
2. Type in "make bzImage -j$(nproc)
3. Type in "make modules -j$(nproc)
4. Type in "make modules_install"
5. Type in "make install"
6. Type in "reboot"

A. Go to "csc3150/source/program2"
B. Type in "sudo su"
C. Type in "make" to compile
D. Type in "insmod program2.ko" to insect kernel
E. Type in "lsmod | grep program2" to check
F. Type in "dmesg | tail -n 34" to display the lastest 34 lines
G. Type in "rmmod program2.o" to remove it from kernel

2.4 Sample outputs:

Normal execution:

```
[ 1452.635752] [program2] : Module_init{Bao Rui} {120090738}
[ 1452.635756] [program2] : Module_init create kthread start
[ 1452.635815] [program2] : Module_init kthread start
[ 1452.635817] [program2] : Kthrad starts
[ 1452.639786] [program2] : The child process has pid = 7313
[ 1452.639788] [program2] : The parent process has pid= 7311
[ 1452.639793] [program2] : Child proces
[ 1452.642084] [program2] : child process gets normal termination
[ 1452.642087] [program2] : The return signal is 0
[ 1455.649917] [program2] : Module_exit./my
```

SIGBUS:

```
[ 1313.498424] [program2] : Module_exit
[ 1406.754613] [program2] : Module_init{Bao Rui} {120090738}
[ 1406.754615] [program2] : Module_init create kthread start
[ 1406.754653] [program2] : Module_init kthread start
[ 1406.754653] [program2] : Kthrad starts
[ 1406.756645] [program2] : The child process has pid = 6873
[ 1406.756647] [program2] : The parent process has pid= 6871
[ 1406.756650] [program2] : Child proces
[ 1406.922079] [program2] : child process gets EXECUTION FAILED
[ 1406.922081] [program2] : The return signal is 7
[ 1409.774398] [program2] : Module_exit./my
```

SIGSTOP:

```
[ 1590.957365] [program2] : Module_init{Bao Rui} {120090738}
[ 1590.957367] [program2] : Module_init create kthread start
[ 1590.957406] [program2] : Module_init kthread start
[ 1590.957406] [program2] : Kthrad starts
[ 1590.959495] [program2] : The child process has pid = 8135
[ 1590.959497] [program2] : The parent process has pid= 8133
[ 1590.959501] [program2] : Child proces
[ 1590.962850] [program2] : Child process stops
[ 1590.962853] [program2] : child process gets SIGSTOP signal
[ 1590.962854] [program2] : The return signal is 19
[ 1592.838677] [program2] : Module_exit./my
```

ABORT:

```
[ 1154.763106] [program2] : Module_init{Bao Rui} {120090738}
[ 1154.763109] [program2] : Module_init create kthread start
[ 1154.763154] [program2] : Module_init kthread start
[ 1154.763154] [program2] : Kthrad starts
[ 1154.766708] [program2] : The child process has pid = 5754
[ 1154.766711] [program2] : The parent process has pid= 5752
[ 1154.766715] [program2] : Child proces
[ 1155.008935] [program2] : child process gets EXECUTION FAILED
[ 1155.008937] [program2] : The return signal is 6
[ 1156.822694] [program2] : Module_exit./my
```

ALARM:

```
[ 1333.998318] [program2] : Module_init{Bao Rui} {120090738}
[ 1333.998320] [program2] : Module_init create kthread start
[ 1333.998367] [program2] : Module_init kthread start
[ 1333.998368] [program2] : Kthrad starts
[ 1334.001476] [program2] : The child process has pid = 6274
[ 1334.001478] [program2] : The parent process has pid= 6272
[ 1334.001481] [program2] : Child proces
[ 1336.003925] [program2] : child process gets EXECUTION FAILED
[ 1336.003928] [program2] : The return signal is 14
[ 1336.881483] [program2] : Module_exit./my
```

FLOATING:

```
[ 1627.309501] [program2] : Module_init{Bao Rui} {120090738}
[ 1627.309504] [program2] : Module_init create kthread start
[ 1627.309720] [program2] : Module_init kthread start
[ 1627.309721] [program2] : Kthrad starts
[ 1627.311526] [program2] : The child process has pid = 7507
[ 1627.311528] [program2] : The parent process has pid= 7505
[ 1627.311837] [program2] : Child proces
[ 1627.476532] [program2] : child process gets EXECUTION FAILED
[ 1627.476535] [program2] : The return signal is 8
[ 1657.745553] [program2] : Module_exit./my
```

HANGUP:

```
[ 1546.129370] [program2] : Module_init create kthread start
[ 1546.129580] [program2] : Module_init kthread start
[ 1546.129581] [program2] : Kthrad starts
[ 1546.131519] [program2] : The child process has pid = 6935
[ 1546.131521] [program2] : The parent process has pid= 6933
[ 1546.131524] [program2] : Child proces
[ 1546.131996] [program2] : child process gets EXECUTION FAILED
[ 1546.131998] [program2] : The return signal is 1
[ 1573.428525] [program2] : Module_exit./my
```

ILLEGAL_INSTR:

```
[ 1744.081204] [program2] : Module_init{Bao Rui} {120090738}
[ 1744.081207] [program2] : Module_init create kthread start
[ 1744.081261] [program2] : Module_init kthread start
[ 1744.081262] [program2] : Kthrad starts
[ 1744.083440] [program2] : The child process has pid = 8099
[ 1744.083442] [program2] : The parent process has pid= 8097
[ 1744.083445] [program2] : Child proces
[ 1744.237938] [program2] : child process gets EXECUTION FAILED
[ 1744.237940] [program2] : The return signal is 4
[ 1770.343139] [program2] : Module_exit./my
```

INTERRUPT:

```
[ 1856.394799] [program2] : Module_init{Bao Rui} {120090738}
[ 1856.394801] [program2] : Module_init create kthread start
[ 1856.394848] [program2] : Module_init kthread start
[ 1856.394849] [program2] : Kthrad starts
[ 1856.397348] [program2] : The child process has pid = 9101
[ 1856.397350] [program2] : The parent process has pid= 9099
[ 1856.397354] [program2] : Child proces
[ 1856.397832] [program2] : child process gets EXECUTION FAILED
[ 1856.397834] [program2] : The return signal is 2
[ 1880.455926] [program2] : Module_exit./my
```

KILL:

```
[ 1984.668331] [program2] : Module_init{Bao Rui} {120090738}
[ 1984.668333] [program2] : Module_init create kthread start
[ 1984.670689] [program2] : Module_init kthread start
[ 1984.670691] [program2] : Kthrad starts
[ 1984.672804] [program2] : The child process has pid = 9655
[ 1984.672806] [program2] : The parent process has pid= 9652
[ 1984.673740] [program2] : Child proces
[ 1984.674195] [program2] : child process gets EXECUTION FAILED
[ 1984.674196] [program2] : The return signal is 9
[ 2006.839138] [program2] : Module_exit./my
```

PIPE:

```
[ 2135.905112] [program2] : Module_init{Bao Rui} {120090738}
[ 2135.905115] [program2] : Module_init create kthread start
[ 2135.905464] [program2] : Module_init kthread start
[ 2135.905467] [program2] : Kthrad starts
[ 2135.909983] [program2] : The child process has pid = 10836
[ 2135.909985] [program2] : The parent process has pid= 10834
[ 2135.911178] [program2] : Child proces
[ 2135.912751] [program2] : child process gets EXECUTION FAILED
[ 2135.912754] [program2] : The return signal is 13
[ 2148.957851] [program2] : Module_exit./my
```

QUIT:

```
[ 2191.113303] [program2] : Module_init{Bao Rui} {120090738}
[ 2191.113306] [program2] : Module_init create kthread start
[ 2191.113347] [program2] : Module_init kthread start
[ 2191.113348] [program2] : Kthrad starts
[ 2191.116500] [program2] : The child process has pid = 11352
[ 2191.116502] [program2] : The parent process has pid= 11350
[ 2191.116505] [program2] : Child proces
[ 2191.427577] [program2] : child process gets EXECUTION FAILED
[ 2191.427580] [program2] : The return signal is 3
[ 2222.306631] [program2] : Module_exit./my
```

Segment_fault:

```
[ 2320.958718] [program2] : Module_init{Bao Rui} {120090738}
[ 2320.958929] [program2] : Module_init create kthread start
[ 2320.959044] [program2] : Module_init kthread start
[ 2320.959045] [program2] : Kthrad starts
[ 2320.963103] [program2] : The child process has pid = 11968
[ 2320.963105] [program2] : The parent process has pid= 11966
[ 2320.963108] [program2] : Child proces
[ 2321.376969] [program2] : child process gets EXECUTION FAILED
[ 2321.376972] [program2] : The return signal is 11
[ 2325.007969] [program2] : Module_exit./my
```

TERMINATE:

```
[ 2388.609966] [program2] : Module_init{Bao Rui} {120090738}
[ 2388.609969] [program2] : Module_init create kthread start
[ 2388.610009] [program2] : Module_init kthread start
[ 2388.610009] [program2] : Kthrad starts
[ 2388.614435] [program2] : The child process has pid = 12533
[ 2388.614437] [program2] : The parent process has pid= 12531
[ 2388.614441] [program2] : Child proces
[ 2388.614952] [program2] : child process gets EXECUTION FAILED
[ 2388.614954] [program2] : The return signal is 15
[ 2401.873863] [program2] : Module_exit./my
```

TRAP:

```
[ 2191.113303] [program2] : Module_init{Bao Rui} {120090738}
[ 2191.113306] [program2] : Module_init create kthread start
[ 2191.113347] [program2] : Module_init kthread start
[ 2191.113348] [program2] : Kthrad starts
[ 2191.116500] [program2] : The child process has pid = 11352
[ 2191.116502] [program2] : The parent process has pid= 11350
[ 2191.116505] [program2] : Child proces
[ 2191.427577] [program2] : child process gets EXECUTION FAILED
[ 2191.427580] [program2] : The return signal is 3
[ 2222.306631] [program2] : Module_exit./my
```

What I have learned in this assignment:

1, Understand programs can be executed in both kernel mode and user mode.

2, Thoroughly understand how process thread works and how parent process and child process communicate with each other.

3, Know how to implement forking a child process in kernel mode and user mode.

4, How to compile kernel.

5, To have strong belief and optimism towards difficult task:)