120090516 YangYin
Assignment 1 Report

Program1:

Design

1. Use fork function to fork a child process.
2. Check whether the process is parent process or child process
   (1) Pid=0: Child process. Execve the test program and raise SIGHLD signal.
   (2) Pid!=0: Parent process. Wait for child process terminates and output how did the child process terminates and what signal was raised in child process.

Set Up Environment

I used a VM environment (with Linux Kernel Version 5.10.146, GCC version 5.4.0).

Makefile is used to compiler the program.

Output

abort

```
vagrant@csc3150:~/csc3150/Assignment1/program1$ ./program1 ./abort
Process start to fork
I'm the Parent Process, my pid = 6224
I'm the Child Process, my pid = 6225
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
```

Alarm

```
vagrant@csc3150:~/csc3150/Assignment1/program1$ ./program1 ./alarm
Process start to fork
I'm the Parent Process, my pid = 6194
I'm the Child Process, my pid = 6195
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
```

bus

```
vagrant@csc3150:~/csc3150/Assignment1/program1$ ./program1 ./bus
Process start to fork
I'm the Parent Process, my pid = 6167
I'm the Child Process, my pid = 6168
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
```

Floating

```
vagrant@csc3150:~/csc3150/Assignment1/program1$ ./program1 ./floating
Process start to fork
I'm the Parent Process, my pid = 6140
I'm the Child Process, my pid = 6141
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
```

Hangup

```
vagrant@csc3150:~/csc3150/Assignment1/program1$ ./program1 ./hangup
Process start to fork
I'm the Parent Process, my pid = 6114
I'm the Child Process, my pid = 6115
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
```

Inllegal_instr

```
vagrant@csc3150:~/csc3150/Assignment1/program1$ ./program1 ./illegal_instr
Process start to fork
I'm the Parent Process, my pid = 6086
I'm the Child Process, my pid = 6087
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
```

Interrupt

```
vagrant@csc3150:~/csc3150/Assignment1/program1$ ./program1 ./interrupt
Process start to fork
I'm the Parent Process, my pid = 6048
I'm the Child Process, my pid = 6049
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
```

Kill

```
vagrant@csc3150:~/csc3150/Assignment1/program1$ ./program1 ./kill
Process start to fork
I'm the Parent Process, my pid = 6021
I'm the Child Process, my pid = 6022
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
```

Normal

```
vagrant@csc3150:~/csc3150/Assignment1/program1$ ./program1 ./normal
Process start to fork
I'm the Parent Process, my pid = 5396
I'm the Child Process, my pid = 5397
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
```

Pipe

```
vagrant@csc3150:~/csc3150/Assignment1/program1$ ./program1 ./pipe
Process start to fork
I'm the Parent Process, my pid = 5995
I'm the Child Process, my pid = 5996
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
```

Quit

```
vagrant@csc3150:~/csc3150/Assignment1/program1$ ./program1 ./quit
Process start to fork
I'm the Parent Process, my pid = 5968
I'm the Child Process, my pid = 5969
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
```

Segment_fault

```
vagrant@csc3150:~/csc3150/Assignment1/program1$ ./program1 ./segment_fault
Process start to fork
I'm the Parent Process, my pid = 5941
I'm the Child Process, my pid = 5942
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

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

Stop

```
vagrant@csc3150:~/csc3150/Assignment1/program1$ ./program1 ./stop
Process start to fork
I'm the Parent Process, my pid = 5915
I'm the Child Process, my pid = 5916
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

Parent process receives SIGCHLD signal
CHILD PROCESS STOPPED
```

Terminate

```
Process start to fork
I'm the Parent Process, my pid = 5900
I'm the Child Process, my pid = 5901
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
```

Trap

```
vagrant@csc3150:~/csc3150/Assignment1/program1$ ./program1 ./trap
Process start to fork
I'm the Parent Process, my pid = 5846
I'm the Child Process, my pid = 5848
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:

Design

1. Initialize a program2 module and when it is initializing, use kthread_create function to create a thread run my_fork function.
2. Within my_fork, use kernel_clone to fork a new process which is used to run my_exec function.
    (1) Output the pid of both child and parent process in kernel.
    (2) Use parent_wait function to wait for the child process terminate.
3. Within my_exce function,
    (1) Use getfilename_kernel to get the test file name.
    (2) Use do_exceve to exceve the test program.
4. Within parent_wait function, use do_wait to wait child process terminate. After waiting, output the signal was raised in child process in kernel.

Set Up Environment

1. Use chmod 777 to edit the kernel code.
2. Export kernel_clone, getfilename_kernel, do_execve, do_wait function.
3. Build kernel Image and modules, install kernel modules and kernel.
4. Reboot the VM.

Output

NOTES