Assigment1 Report_120090602_杨楠

1. How to design my program

• For task1, I use fork function to fork a child progress and judge whether it is the child progress or parent progress through the return value of the fork function. If the return value is 0, then it is the child progress and I can execute test program through execve function. If the return value is -1, then the implementation of the fork function failed. When the return value is a positive value, it is the parent process and it will receive the SIGCHLD signal by wait function. And I can use the status the parent process receives and through WIFEXITED, WIFSTOPPED, WIFSIGNALED function to determine whether it is normal termination, stopped or terminated. Then in the terminated condition, I use WTERMSTG function to get the case value and find the specific case through the corresponding relation between them and print out the termination information.

• For task2, I use kernel_clone to fork a new process and create my_execve through getname_kernel to get the name of file and pass it to do_execve toexecute the testn program. Also I use do_wait and determine the signal received through the value it returns. And I find the relationship between the value and the signal number.(some are equivalent, some are with difference value equals 128, and others are specific number).

2. How to set up development environment (including how to compile kernel)

• First, I changed current directory to csc3150 and execute vagrant up to set up VM.

• Second, I scrutinized my gcc version and linux kernel version and found I needed to

update the kernel version, so I download linux-5.10.146.tar.xz from the website in the tutorial slides. Also I made my space large enough to compile the kernel successfully.

• I updated it through the linux command provided on the tutorial slides in the Compile Kernel section. I first created seed directory in the /home path and then created work directory in the seed directory and copied the linux-5.10.146 in the work directory. Then I decompressed the tar file and rooted the kernel source directory to Clean previous setting, start configuration, build kernel Image and modules. Then I installed kernel modules and kernel and rebooted to load new kernel.

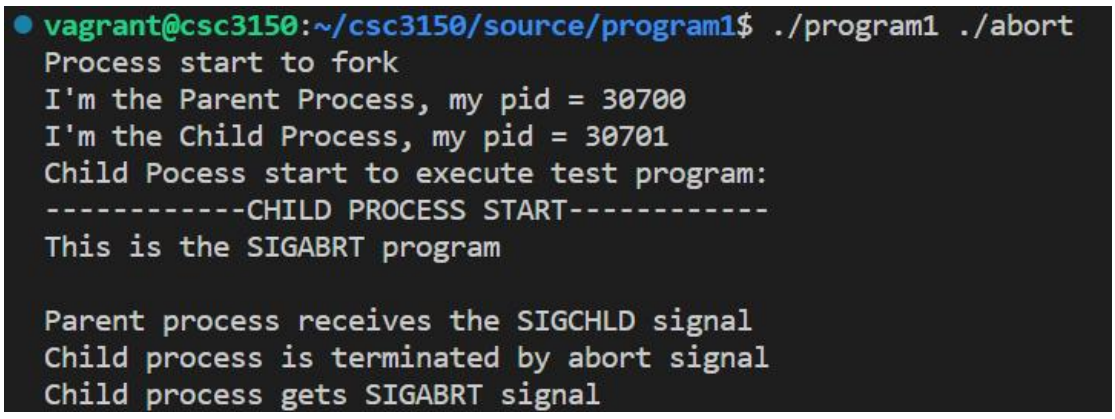
• Before I started my task2, I first exported the 4 symbols, getname_kernel, kernel_clone, do_wait, do_execve by editing the source files adding EXPORT_SYMBOL() after they were defined.

• Then I recompiled the kernel from the step “make bzImage”.

3. **Screenshot of my program output**

TASK1

• Demo output for signaled abort:



• Demo output for signaled alarm:

```
vagrant@csc3150:~/csc3150/source/program1$ ./program1 ./alarm
Process start to fork
I'm the Parent Process, my pid = 31166
I'm the Child Process, my pid = 31167
Child Pocess start to execute test program:
------------CHILD PROCESS START------------
This is the SIGALRM program

Parent process receives the SIGCHLD signal
Child process is terminated by alarm signal
Child process gets SIGALRM signal
```

- Demo output for signaled bus:

```
vagrant@csc3150:~/csc3150/source/program1$ ./program1 ./bus
Process start to fork
I'm the Parent Process, my pid = 31241
I'm the Child Process, my pid = 31242
Child Pocess start to execute test program:
------------CHILD PROCESS START------------
This is the SIGBUS program

Parent process receives the SIGCHLD signal
Child process is terminated by bus signal
Child process gets SIGBUS signal
```

- Demo output for signaled floating:

```
vagrant@csc3150:~/csc3150/source/program1$ ./program1 ./floating
Process start to fork
I'm the Parent Process, my pid = 31260
I'm the Child Process, my pid = 31261
Child Pocess start to execute test program:
------------CHILD PROCESS START------------
This is the SIGFPE program

Parent process receives the SIGCHLD signal
Child process is terminated by floating signal
Child process gets SIGFPE signal
```

- Demo output for signaled hangup:

```
vagrant@csc3150:~/csc3150/source/program1$ ./program1 ./hangup
Process start to fork
I'm the Parent Process, my pid = 31343
I'm the Child Process, my pid = 31344
Child Pocess start to execute test program:
------------CHILD PROCESS START------------
This is the SIGHUP program

Parent process receives the SIGCHLD signal
Child process is terminated by hangup signal
Child process gets SIGUP signal
```

- Demo output for signaled illegal_instr:

```
vagrant@csc3150:~/csc3150/source/program1$ ./program1 ./illegal_instr
Process start to fork
I'm the Parent Process, my pid = 31369
I'm the Child Process, my pid = 31370
Child Pocess start to execute test program:
------------CHILD PROCESS START------------
This is the SIGILL program

Parent process receives the SIGCHLD signal
Child process is terminated by illegal_instr signal
Child process gets SIGILL signal
```

- Demo output for signaled interrupt:

```
vagrant@csc3150:~/csc3150/source/program1$ ./program1 ./interrupt
Process start to fork
I'm the Parent Process, my pid = 31453
I'm the Child Process, my pid = 31454
Child Pocess start to execute test program:
------------CHILD PROCESS START------------
This is the SIGINT program

Parent process receives the SIGCHLD signal
Child process is terminated by interrupt signal
Child process gets SIGINT signal
```

- Demo output for signaled kill:

```
vagrant@csc3150:~/csc3150/source/program1$ ./program1 ./kill
Process start to fork
I'm the Parent Process, my pid = 31514
I'm the Child Process, my pid = 31515
Child Pocess start to execute test program:
------------CHILD PROCESS START------------
This is the SIGKILL program

Parent process receives the SIGCHLD signal
Child process is terminated by kill signal
Child process gets SIGKILL signal
```

- Demo output for normal termination:

```
vagrant@csc3150:~/csc3150/source/program1$ ./program1 ./normal
Process start to fork
I'm the Parent Process, my pid = 31542
I'm the Child Process, my pid = 31543
Child Pocess start to execute test program:
------------CHILD PROCESS START------------
This is the normal program

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

- Demo output for signaled pipe:

```
vagrant@csc3150:~/csc3150/source/program1$ ./program1 ./pipe
Process start to fork
I'm the Parent Process, my pid = 31728
I'm the Child Process, my pid = 31729
Child Pocess start to execute test program:
------------CHILD PROCESS START------------
This is the SIGPIPE program

Parent process receives the SIGCHLD signal
Child process is terminated by pipe signal
Child process gets SIGPIPE signal
```

- Demo output for signaled quit:

```
vagrant@csc3150:~/csc3150/source/program1$ ./program1 ./quit
Process start to fork
I'm the Parent Process, my pid = 31893
I'm the Child Process, my pid = 31894
Child Pocess start to execute test program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

Parent process receives the SIGCHLD signal
Child process is terminated by quit signal
Child process gets SIGQUIT signal
```

- Demo output for signaled segment_fault:

```
vagrant@csc3150:~/csc3150/source/program1$ ./program1 ./segment_fault
Process start to fork
I'm the Parent Process, my pid = 32014
I'm the Child Process, my pid = 32015
Child Pocess start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

Parent process receives the SIGCHLD signal
Child process is terminated by segment_fault signal
Child process gets SIGSEGV signal
```

- Demo output for stopped:

```
vagrant@csc3150:~/csc3150/source/program1$ ./program1 ./stop
Process start to fork
I'm the Parent Process, my pid = 32086
I'm the Child Process, my pid = 32087
Child Pocess start to execute test program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

Parent process receives the SIGCHLD signal
Child process is stopped by stop signal
Child process gets SIGSTOP signal
```

- Demo output for signaled terminate:

```
vagrant@csc3150:~/csc3150/source/program1$ ./program1 ./terminate
Process start to fork
I'm the Parent Process, my pid = 32169
I'm the Child Process, my pid = 32170
Child Pocess start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTERM program

Parent process receives the SIGCHLD signal
Child process is terminated by terminate signal
Child process gets SIGTERM signal
```

- Demo output for signaled trap:

```
vagrant@csc3150:~/csc3150/source/program1$ ./program1 ./trap
Process start to fork
I'm the Parent Process, my pid = 32232
I'm the Child Process, my pid = 32233
Child Pocess start to execute test program:
------------CHILD PROCESS START------------
This is the SIGTRAP program

Parent process receives the SIGCHLD signal
Child process is terminated by trap signal
Child process gets SIGTRAP signal
```

TASK2

- Demo output for signaled abort:

```
[ 3283.301798] [program2] : Module_init {Yang Nan} {120090602}
[ 3283.302045] [program2] : Module_init create kthread start
[ 3283.302215] [program2] : Module_init kthread start
[ 3283.302369] [program2] : The child process has pid = 4670
[ 3283.302370] [program2] : This is the parent process, pid = 4668
[ 3283.407717] [program2] : child process
[ 3283.407718] [program2] : get SIGABRT signal
[ 3283.407719] [program2] : child process is aborted
[ 3283.407720] [program2] : the return signal is 6
[ 3295.250027] [program2] : Module_exit
```

- Demo output for signaled alarm:

```
[ 3740.816062] [program2] : Module_init {Yang Nan} {120090602}
[ 3740.816320] [program2] : Module_init create kthread start
[ 3740.816558] [program2] : Module_init kthread start
[ 3740.816568] [program2] : The child process has pid = 5091
[ 3740.816569] [program2] : This is the parent process, pid = 5090
[ 3740.817189] [program2] : child process
[ 3740.817190] [program2] : get SIGALRM signal
[ 3740.817190] [program2] : child process has expired alarm clock
[ 3740.817191] [program2] : the return signal is 14
[ 3746.080820] [program2] : Module_exit
```

- Demo output for signaled bus:

```
[ 4256.307863] [program2] : Module_init {Yang Nan} {120090602}
[ 4256.308108] [program2] : Module_init create kthread start
[ 4256.308272] [program2] : Module_init kthread start
[ 4256.308283] [program2] : The child process has pid = 6239
[ 4256.308283] [program2] : This is the parent process, pid = 6238
[ 4256.413282] [program2] : child process
[ 4256.413320] [program2] : get SIGBUS signal
[ 4256.413321] [program2] : child process causes bus error
[ 4256.413322] [program2] : the return signal is 7
[ 4262.216764] [program2] : Module_exit
```

- Demo output for signaled floating:

```
[ 3361.784271] [program2] : Module_init {Yang Nan} {120090602}
[ 3361.784549] [program2] : Module_init create kthread start
[ 3361.784659] [program2] : Module_init kthread start
[ 3361.784673] [program2] : The child process has pid = 4763
[ 3361.784673] [program2] : This is the parent process, pid = 4762
[ 3361.895491] [program2] : child process
[ 3361.895492] [program2] : get SIGFPE signal
[ 3361.895492] [program2] : child process has floating point exception
[ 3361.895493] [program2] : the return signal is 8
[ 3375.446544] [program2] : Module_exit
```

- Demo output for signaled hangup:

```
[ 2656.507408] [program2] : Module_init {Yang Nan} {120090602}
[ 2656.507649] [program2] : Module_init create kthread start
[ 2656.508327] [program2] : Module_init kthread start
[ 2656.508336] [program2] : The child process has pid = 3933
[ 2656.508336] [program2] : This is the parent process, pid = 3931
[ 2656.508844] [program2] : child process
[ 2656.508845] [program2] : get SIGHUP signal
[ 2656.508845] [program2] : child process is hung up
[ 2656.508846] [program2] : the return signal is 1
[ 2664.496820] [program2] : Module_exit
```

- Demo output for signaled illegal_instr:

```
[ 2884.662853] [program2] : Module_init {Yang Nan} {120090602}
[ 2884.663350] [program2] : Module_init create kthread start
[ 2884.663556] [program2] : Module_init kthread start
[ 2884.663574] [program2] : The child process has pid = 4077
[ 2884.663575] [program2] : This is the parent process, pid = 4076
[ 2884.768990] [program2] : child process
[ 2884.768991] [program2] : get SIGILL signal
[ 2884.768992] [program2] : child process has illegal instruction
[ 2884.768992] [program2] : the return signal is 4
[ 2891.873155] [program2] : Module_exit
```

- Demo output for signaled interrupt:

```
[ 2735.983891] [program2] : Module_init {Yang Nan} {120090602}
[ 2735.984470] [program2] : Module_init create kthread start
[ 2735.985362] [program2] : Module_init kthread start
[ 2735.985370] [program2] : The child process has pid = 3998
[ 2735.985371] [program2] : This is the parent process, pid = 3996
[ 2735.986215] [program2] : child process
[ 2735.986215] [program2] : get SIGINT signal
[ 2735.986216] [program2] : child process is interrupted
[ 2735.986216] [program2] : the return signal is 2
[ 2746.655209] [program2] : Module_exit
```

- Demo output for signaled kill:

```
[ 3540.471574] [program2] : Module_init {Yang Nan} {120090602}
[ 3540.471900] [program2] : Module_init create kthread start
[ 3540.472166] [program2] : Module_init kthread start
[ 3540.472188] [program2] : The child process has pid = 4880
[ 3540.472189] [program2] : This is the parent process, pid = 4879
[ 3540.472940] [program2] : child process
[ 3540.472941] [program2] : get SIGKILL signal
[ 3540.472941] [program2] : child process is killed
[ 3540.472942] [program2] : the return signal is 9
[ 3545.658249] [program2] : Module_exit
```

- Demo output for normal termination:

```
[ 4174.799199] [program2] : Module_init {Yang Nan} {120090602}
[ 4174.799482] [program2] : Module_init create kthread start
[ 4174.799735] [program2] : Module_init kthread start
[ 4174.799746] [program2] : The child process has pid = 6145
[ 4174.799746] [program2] : This is the parent process, pid = 6144
[ 4179.824154] [program2] : child process
[ 4179.824155] [program2] : get SIGCHLD signal
[ 4179.824156] [program2] : Normal termination with return signal 17
[ 4182.274196] [program2] : Module_exit
```

- Demo output for signaled pipe:

```
[ 3677.307302] [program2] : Module_init {Yang Nan} {120090602}
[ 3677.307582] [program2] : Module_init create kthread start
[ 3677.308380] [program2] : Module_init kthread start
[ 3677.308393] [program2] : The child process has pid = 5015
[ 3677.308393] [program2] : This is the parent process, pid = 5013
[ 3677.308908] [program2] : child process
[ 3677.308909] [program2] : get SIGPIPE signal
[ 3677.308909] [program2] : child process has broken pipe
[ 3677.308910] [program2] : the return signal is 13
[ 3684.168007] [program2] : Module_exit
```

- Demo output for signaled quit:

```
vagrant@csc3150:~/csc3150/source/program2$ dmesg
[ 1728.532684] [program2] : Module_init {Yang Nan} {120090602}
[ 1728.533016] [program2] : Module_init create kthread start
[ 1728.533186] [program2] : Module_init kthread start
[ 1728.533204] [program2] : The child process has pid = 3152
[ 1728.533205] [program2] : This is the parent process, pid = 3151
[ 1728.666542] [program2] : child process
[ 1728.666544] [program2] : get SIGQUIT signal
[ 1728.666545] [program2] : child process quit
[ 1728.666545] [program2] : the return signal is 3
[ 1735.753383] [program2] : Module_exit
```

- Demo output for signaled segment_fault:

```
[ 3615.684324] [program2] : Module_init {Yang Nan} {120090602}
[ 3615.684753] [program2] : Module_init create kthread start
[ 3615.684910] [program2] : Module_init kthread start
[ 3615.684934] [program2] : The child process has pid = 4945
[ 3615.684935] [program2] : This is the parent process, pid = 4944
[ 3615.820775] [program2] : child process
[ 3615.820776] [program2] : get SIGSEGV signal
[ 3615.820777] [program2] : child process has invalid memory segment access
[ 3615.820778] [program2] : the return signal is 11
[ 3621.417638] [program2] : Module_exit
```

- Demo output for stopped:

```
[ 4069.105837] [program2] : Module_init {Yang Nan} {120090602}
[ 4069.106362] [program2] : Module_init create kthread start
[ 4069.107459] [program2] : Module_init kthread start
[ 4069.107472] [program2] : The child process has pid = 6024
[ 4069.107473] [program2] : This is the parent process, pid = 6022
[ 4069.108281] [program2] : child process
[ 4069.108282] [program2] : get SIGSTOP signal
[ 4069.108282] [program2] : child process stopped
[ 4069.108283] [program2] : the return signal is 19
[ 4074.814291] [program2] : Module_exit
```

- Demo output for signaled terminate:

```
[ 3809.609810] [program2] : Module_init {Yang Nan} {120090602}
[ 3809.610056] [program2] : Module_init create kthread start
[ 3809.610265] [program2] : Module_init kthread start
[ 3809.610274] [program2] : The child process has pid = 5155
[ 3809.610274] [program2] : This is the parent process, pid = 5154
[ 3809.611159] [program2] : child process
[ 3809.611159] [program2] : get SIGTERM signal
[ 3809.611160] [program2] : child process terminated
[ 3809.611160] [program2] : the return signal is 15
[ 3814.076152] [program2] : Module_exit
```

• Demo output for signaled trap:

```
[ 2967.436044] [program2] : Module_init {Yang Nan} {120090602}
[ 2967.436331] [program2] : Module_init create kthread start
[ 2967.437241] [program2] : Module_init kthread start
[ 2967.437251] [program2] : The child process has pid = 4181
[ 2967.437252] [program2] : This is the parent process, pid = 4179
[ 2967.542344] [program2] : child process
[ 2967.542346] [program2] : get SIGTRAP signal
[ 2967.542346] [program2] : child process causes trace trap
[ 2967.542347] [program2] : the return signal is 5
[ 2972.997475] [program2] : Module_exit
```

4. What I learn from the tasks

From these tasks, I had a better understanding of the multiple processes and the terminology such as process state and signal. And I learned about how to create process from both user mode and kernel mode and knew more about some specific functions in the linux like the fork function and something else on a practical level. Besides, I learned the concrete measures to compile kernel and to modify the source code to use the function. The tasks make me have a practical experience The compilation process.