CSC3150 Assignment1 Report

@October 7, 2023 @Yuzhe YANG 121090684

Environment

Basic Information

OS: Debian GNU/Linux 11

Kernel Version: 5.10.191

Architecture: aarch64

Compile Kernel

In order to set up my Linux environment, I need to compile the Linux kernel, here is how did I do.

1. Kernel Version: As I have already download the UTM virtual machine with kernel version 5.10.191, it is already satisfied the assignment's requirements.
2. Compile:

```
sudo su
cd /home/seed/work/linux-source-5.10

# Clean previous setting and start configuration
make mrproper
make clean
make menuconfign # save config and exit

# Build kernel Image and modules
make

# Install kernel modules
make modules_install

# Install kernel
make install

# Reboot to load new kernel
reboot
```

Task 1

Description

In task1, we need to call some basic Linux function to implement the main flow:

1. Create a Child Process: Use `fork()` to create a new process, which becomes the child process.
2. Raise a Signal in the Child Process: Use `raise()` to generate a signal within the child process.
3. Execute a Test Program in the Child Process: Use `execve()` to replace the child process's image with the specified test program.
4. Wait for Child Process to Terminate in the Parent Process: Use `waitpid()` in the parent process to wait for the child process to terminate.
5. Handle Signals in the Parent Process: Upon receiving a signal from the child process, use signal handlers or other mechanisms to determine the type of signal received.
6. Print Information Based on the Received Signal: Depending on the type of signal received, print relevant information in the parent process.

Implementation

In order to test all the file in the program1 folder, I write a shell file.

```
make all
cd /home/seed/csc3150_toby/CSC3150_P1/program1
./program1 ./abort
./program1 ./alarm
./program1 ./bus
./program1 ./floating
./program1 ./hangup
./program1 ./illegal_instr
./program1 ./interrupt
./program1 ./kill
./program1 ./normal
./program1 ./pipe
./program1 ./quit
./program1 ./segment_fault
./program1 ./stop
./program1 ./terminate
./program1 ./trap
```

How to run this:

1. Navigate into program1 folder
2. `./test_output.sh`

Output

This is the screenshot of a part of my output:

And the full output:

```
seed □ □ main - □ bash "/home/seed/csc3150_toby/CSC3150_P1/program1/test_output.sh"
cc -o abort abort.c
cc -o alarm alarm.c
cc -o bus bus.c
cc -o floating floating.c
cc -o hangup hangup.c
cc -o illegal_instr illegal_instr.c
cc -o interrupt interrupt.c
cc -o kill kill.c
cc -o normal normal.c
cc -o pipe pipe.c
cc -o program1 program1.c
cc -o quit quit.c
cc -o segment_fault segment_fault.c
cc -o stop stop.c
cc -o terminate terminate.c
cc -o trap trap.c
Process start to fork
I'm the Parent Process, my pid = 5685
I'm the Child Process, my pid = 5686
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
Process start to fork
I'm the Parent Process, my pid = 5687
I'm the Child Process, my pid = 5688
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
Process start to fork
I'm the Parent Process, my pid = 5716
I'm the Child Process, my pid = 5717
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
Process start to fork
I'm the Parent Process, my pid = 5718
I'm the Child Process, my pid = 5719
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
Process start to fork
I'm the Parent Process, my pid = 5720
I'm the Child Process, my pid = 5721
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
Process start to fork
I'm the Parent Process, my pid = 5722
I'm the Child Process, my pid = 5723
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
Process start to fork
I'm the Parent Process, my pid = 5724
I'm the Child Process, my pid = 5725
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
Process start to fork
I'm the Parent Process, my pid = 5726
I'm the Child Process, my pid = 5727
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
Process start to fork
I'm the Parent Process, my pid = 5728
I'm the Child Process, my pid = 5729
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

Process start to fork
I'm the Parent Process, my pid = 5730
I'm the Child Process, my pid = 5731
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
Process start to fork
I'm the Parent Process, my pid = 5732
I'm the Child Process, my pid = 5733
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
Process start to fork
I'm the Parent Process, my pid = 5734
I'm the Child Process, my pid = 5735
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
Process start to fork
I'm the Parent Process, my pid = 5736
I'm the Child Process, my pid = 5737
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
Process start to fork
I'm the Parent Process, my pid = 5738
I'm the Child Process, my pid = 5739
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
Process start to fork
I'm the Parent Process, my pid = 5740
I'm the Child Process, my pid = 5741
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

 [~/csc3150_toby/CSC3150_P1/program1]
 seed □ □ main - □
```

Task 2

Description

The procedure of task 2 is similar like task 1, but all the work flow is under the kernel mode, which is different from task 1.

Task 2 involves creating a kernel module that operates in kernel mode and performs basic process management tasks such as forking, executing a test program, and waiting for the child process to terminate. The implementation includes creating a kernel thread that runs the `my_fork()` function, which is responsible for forking a child process `my_exec()` to execute a test program.

For parent process, I use `my_wait()` function to wait the end signal from child process. This function is implemented by `do_wait()` function, and the child process's status is stored in `*wo.wo_stat`.

At last, the program will output the information to kernel log according to the what signal the parent process gets. This procedure is as same as task 1.

Implementation

1. Modifying Kernel Files

   I have used four extern functions to implement task 2.

```
extern pid_t kernel_clone(struct kernel_clone_args *args);
extern struct filename *getname_kernel(const char *filename);
extern long do_wait(struct wait_opts *wo);
extern int do_execve(struct filename *filename, const char __user *const __user *__argv, const char __user *const __user *__envp);
```

Use `kernel_clone()` to fork a new process, and the corresponding kernel file path is `/kernel/fork.c`

Use `do_execve()` to execute the test program, and the corresponding kernel file path is `/fs/exec.c`

Use `getname_kernel()` to get filename, and the corresponding kernel file path is `/fs/namei.c`

Use `do_wait()` to wait for child process' termination status, and the corresponding kernel file pathe is `/kernel/exit.c`

I need to find all the function and its location in the corresponding kernel file, then I use `sudo vim <filepath>` to modify the original kernel file: add `EXPORT_SYMBOL()` at the end of the function. It allows me to call these funcitons properly in my program

After modify all the kernel file, we need to re-compile the kernel and reboot.

2. Normal signal and SIGSTOP signal

   In Task 1, I use `WIFEXITED()` and `WIFSTOPPED()` to judge the normal and stop signal. However these two function in located in the `signal.h` ,which is not supported in the kernel mode. In order to detect the signal properly, I wrote two function to implement `WIFEXITED()` and `WIFSTOPPED()` .

```
int my_WIFEXITED(int status)
{
  return (status & 0xff) == 0;
}

int my_WIFSTOPPED(int status)
{
  return ((status) & 0xff) == 0x7f;
}
```

3. Test program

   I write a shell file to test the output:

```
# cd /home/seed/csc3150_toby/CSC3150_P1/program2
gcc test.c -o test
make clean
make
sudo insmod program2.ko
sudo rmmod program2.ko
sudo dmesg -c
```

   How to run this:

   1. Navigate into program 2 folder
   2. run `./test_output.sh`

Output

```
[ 2372.387647] [program2] : module_init
[ 2372.387648] [program2] : module_init create kthread start
[ 2372.387685] [program2] : module_init kthread start
[ 2372.387706] [program2] : The child process has pid = 26659
[ 2372.387706] [program2] : This is the parent process, pid = 26658
[ 2372.387982] [program2] : child process
[ 2372.390241] [program2] : get SIGABRT signal
[ 2372.390242] [program2] : The return signal is 6
[ 2372.390960] [program2] : module_exit
```

```
[ 2394.844943] [program2] : module_init
[ 2394.844944] [program2] : module_init create kthread start
[ 2394.844997] [program2] : module_init kthread start
[ 2394.845194] [program2] : The child process has pid = 27265
[ 2394.846710] [program2] : This is the parent process, pid = 27263
[ 2394.847761] [program2] : child process
[ 2394.847762] [program2] : get SIGALRM signal
[ 2394.847762] [program2] : The return signal is 14
[ 2394.848754] [program2] : module_exit
```

```
[ 2427.910679] [program2] : module_init
[ 2427.911124] [program2] : module_init create kthread start
[ 2427.911811] [program2] : module_init kthread start
[ 2427.911830] [program2] : The child process has pid = 27904
[ 2427.911831] [program2] : This is the parent process, pid = 27903
[ 2427.912005] [program2] : child process
[ 2427.912005] [program2] : get SIGBUS signal
[ 2427.912006] [program2] : The return signal is 7
[ 2427.914676] [program2] : module_exit
```

```
[ 2465.671103] [program2] : module_init
[ 2465.671567] [program2] : module_init create kthread start
[ 2465.672207] [program2] : module_init kthread start
[ 2465.672702] [program2] : The child process has pid = 28539
[ 2465.673136] [program2] : This is the parent process, pid = 28538
[ 2465.673569] [program2] : child process
[ 2465.673570] [program2] : get SIGFPE signal
[ 2465.674401] [program2] : The return signal is 8
[ 2465.678458] [program2] : module_exit
```

```
[ 2488.239272] [program2] : module_init
[ 2488.239702] [program2] : module_init create kthread start
[ 2488.240394] [program2] : module_init kthread start
[ 2488.240847] [program2] : The child process has pid = 29146
[ 2488.241273] [program2] : This is the parent process, pid = 29145
[ 2488.241733] [program2] : child process
[ 2488.241733] [program2] : get SIGILL signal
[ 2488.242631] [program2] : The return signal is 4
[ 2488.247019] [program2] : module_exit
```

```
[ 2504.872990] [program2] : module_init
[ 2504.873392] [program2] : module_init create kthread start
[ 2504.874064] [program2] : module_init kthread start
[ 2504.874502] [program2] : The child process has pid = 29742
[ 2504.874945] [program2] : This is the parent process, pid = 29741
[ 2504.875351] [program2] : child process
[ 2504.875351] [program2] : get SIGKILL signal
[ 2504.876101] [program2] : The return signal is 9
[ 2504.880181] [program2] : module_exit
```

```
[ 2540.975539] [program2] : module_init
[ 2540.975986] [program2] : module_init create kthread start
[ 2540.976820] [program2] : module_init kthread start
[ 2540.977345] [program2] : The child process has pid = 30403
[ 2540.977839] [program2] : This is the parent process, pid = 30402
[ 2540.978280] [program2] : child process
[ 2540.978281] [program2] : get SIGPIPE signal
[ 2540.979094] [program2] : The return signal is 13
[ 2540.983022] [program2] : module_exit
```

```
[ 2564.870183] [program2] : module_init
[ 2564.870586] [program2] : module_init create kthread start
[ 2564.871163] [program2] : module_init kthread start
[ 2564.871580] [program2] : The child process has pid = 31010
[ 2564.871979] [program2] : This is the parent process, pid = 31009
[ 2564.872394] [program2] : child process
[ 2564.872395] [program2] : get SIGQUIT signal
[ 2564.873263] [program2] : The return signal is 3
[ 2564.877337] [program2] : module_exit
```

```
[ 2580.989454] [program2] : module_init
[ 2580.989847] [program2] : module_init create kthread start
[ 2580.990445] [program2] : module_init kthread start
[ 2580.990900] [program2] : The child process has pid = 31607
[ 2580.991299] [program2] : This is the parent process, pid = 31606
[ 2580.991766] [program2] : child process
[ 2580.991767] [program2] : get SIGSEGV signal
[ 2580.992540] [program2] : The return signal is 11
[ 2581.002934] [program2] : module_exit
```

```
[ 2624.869455] [program2] : module_init
[ 2624.869891] [program2] : module_init create kthread start
[ 2624.870554] [program2] : module_init kthread start
[ 2624.870988] [program2] : The child process has pid = 32822
[ 2624.871466] [program2] : This is the parent process, pid = 32821
[ 2624.871897] [program2] : child process
[ 2624.871898] [program2] : get SIGTERM signal
[ 2624.872715] [program2] : The return signal is 15
[ 2624.876352] [program2] : module_exit
```

```
[ 2649.461734] [program2] : module_init
[ 2649.462139] [program2] : module_init create kthread start
[ 2649.462762] [program2] : module_init kthread start
[ 2649.463250] [program2] : The child process has pid = 33432
[ 2649.463684] [program2] : This is the parent process, pid = 33431
[ 2649.464121] [program2] : child process
[ 2649.464121] [program2] : get SIGTRAP signal
[ 2649.464889] [program2] : The return signal is 5
[ 2649.468805] [program2] : module_exit
```

SIGSTOP signal

```
[ 5263.456409] [program2] : module_init
[ 5263.457059] [program2] : module_init create kthread start
[ 5263.457686] [program2] : module_init kthread start
[ 5263.458188] [program2] : The child process has pid = 67240
[ 5263.458833] [program2] : This is the parent process, pid = 67239
[ 5263.459362] [program2] : child process
[ 5263.459363] [program2] : get SIGSTOP signal
[ 5263.460225] [program2] : The return signal is 4991
[ 5263.462238] [program2] : module_exit
```

Normal termination

```
[ 4973.915580] [program2] : module_init
[ 4973.915581] [program2] : module_init create kthread start
[ 4973.915628] [program2] : module_init kthread start
[ 4973.915832] [program2] : The child process has pid = 60987
[ 4973.917032] [program2] : This is the parent process, pid = 60986
[ 4973.917175] [program2] : child process
[ 4973.917386] [program2] : Normal termination
[ 4973.917386] [program2] : The return signal is 25600
[ 4973.918240] [program2] : module_exit
```

Bonus

Description