Yang Yahe
119010384
Assignment 1
School of Science and Engineer

Report

Design of Program

Design of Program 1

In the program 1, I initially fork a child process by **fork()** function. Then I get the pid value of the process, and I use **if()** to judge the return value. If the return value is -1, it means that the fork fails. If the pid value is 0, it means that the Child Process is successfully created. If the pid value is positive, it meas that it implements the parent process.

In the Child Process, I use the code below to execute test program.

```
    for(int i = 0; i < argc - 1; i++){
      arg[i] = argv[i+1];
    }
    arg[argc - 1] = NULL;
    execve(arg[0], arg, NULL);
}
```

In the Parent Process, I use waitpid(pid, &status, WUNTRACED). This function let parent process wait for termination of child process and transfer the return signal of child process to parent process.

The use the signal the child process gives out, I use **switch()** function to printf the signal that the child process get.

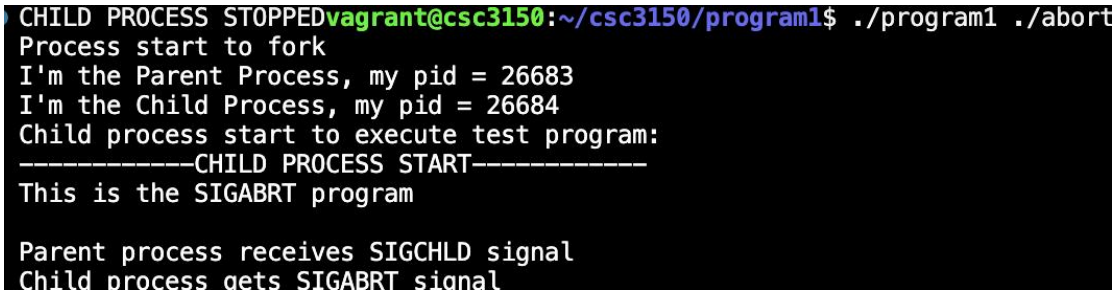
Output Sample:

Normal termination:

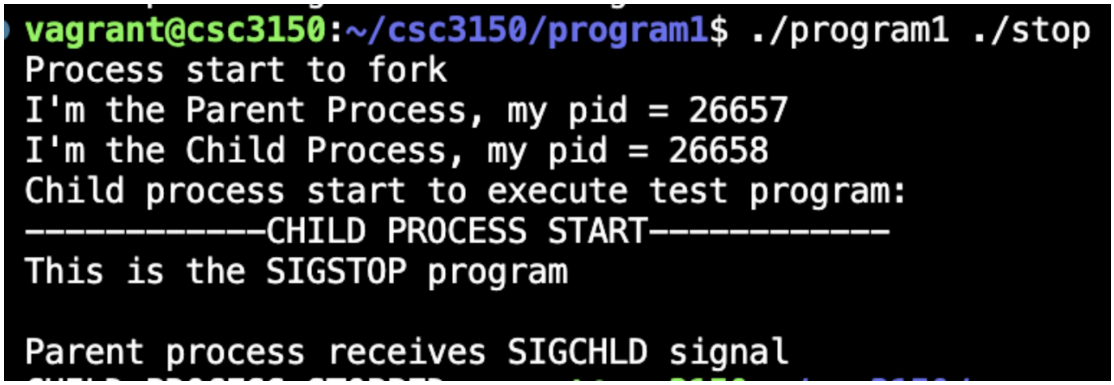
```
vagrant@csc3150:~/csc3150/program1$ ./program1 ./abort
Process start to fork
I'm the Parent Process, my pid = 26630
I'm the Child Process, my pid = 26631
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGABRT program

Parent process receives SIGCHLD signal
Child process gets SIGABRT signal
```

Abort:



Stopped:



Design of Program 2

In __initial program2_init(void), I create a kernel thread by calling function kthread-create(). This function connects the function my_fork(). In my my_fork(), I fork a process by _do_fork(SIGHLD,(long unsigned)&my_exec,0,NULL,NULL,0). The child process will implement the program of my_exec. My_exec points to path of the implemented test program. My_exec() actually is a wrapped function, it calls kernel module function do_execve(). Do_execve is exported by kernel module. In the parent process, I also call my_wait(pid), which let parent process wait for termination of child process. My_wait(pid) is also a wrapped function, do_wait() kernel module function is inside. Do_wait() is also exported by kernel module. After the termination of child process. *wo.wo_stat sores the signal value. Then according to signal value. We printf the error reason or normal termination. Module_exit is called finally, which means kernel module exiting.

Environment Set up and Kernel Compile

Environment:

ubuntu 18.04.6

Kernel linux-5.15.72

Kernel Compile:

Download source code and extract it into /home/seed/work
Sudo tar xvf linux-5.15.72.tar.xz
cp /boot/config ~/home/seed/work/linux-5.15.72

sudo su
cd /home/seed/work/linux-5.15.72
make mrproper
make clean

sudo apt-get install libncurses-dev flex bison openssl libssl-dev dkms libelf-dev libudev-dev libpci-dev libiberty-dev autoconf

make menuconfig
make -j$(nproc)
meke modules_install
make install
reboot

Learn from the Project

Through the project, I learned how to compile the linux Kernel and how to use makefile to compile a group of C file. Besides, I have a deep understand of the state of process. Such as how to fork a child process, how to execute the test program, or how to judge the termination status.
I also learned how to rewrite the kernel, it is a very useful homework.