Name: 邓毅轩 Deng Yixuan

ID: 120090661

Report

Program Design:

About program 1, first I used fork to create a child process. Then determine the status of the current process based on the pid value returned. If the pid is smaller than 0, there must be some error. So output error message fork. If the pid equals to 0, it meansthat it is now located in the child process. At this point the argv of the original process is deleted from the first compiled code belonging to the parent process and a new argv is created containing only the compiled code of the child process. Pass a new argv to execve to execute on the child process. If pid is larger than 0, it means that it is now located in the parent process. In this case, waitpid is used to wait for the return signal from the child process, and then the return signal is simply output.

About program 2, first create the kernel with kthread_create, then use IS_ERR to determine if the creation was successful. If successful, wake up the process with wake_up_process. Use kernel_clone in my_fork to run the child process, then use do_wait in my_wait to wait for the child process to finish execution and determine the return signal by the returned pid. Use getname_kernel in my_exec to get the filename and then use do_execve to execute the child process. If the return value is 0, it means that the program is normal, otherwise it means that there is an error and do_exit should be called to exit the program.

Environment and compile kernel:

Since I already had ubuntu installed on my computer, I didn't use a virtual box to configure the environment. I downloaded the corresponding version of the kernel from the PowerPoint given by the TA and compiled the kernel according to the below instructions:

sudo apt-get install libncurses-dev gawk flex bison openssl libssl-dev dkms libelf-devlibudev-dev libpci-dev libiberty-dev

cp KERNEL_FILE.tar.xz /home/seed/work

cd /home/seed/work

sudo tar xvf KERNEL_FILE.tar.xz

Copy config from /boot to /home/seed/work/KERNEL_FILE

sudo su

cd /home/seed/work /KERNEL_FILE

make mrproper

make clean

make menuconfig

make bzImage -j$(nproc)

make modules -j$(nproc)

make –j$(nproc)

make modules_install

make install

reboot

At this point the kernel version has been successfully replaced with 5.10.5. Finally, the

internal source code is called via EXPORT_SYMBOL() and the kernel is recompiled once more to start testing the program.

Program Output:

program1:

```
wek_deng@Zero-Inspiron-7500:~/Assignment_1_120090661/source/program1$ ./program1
 abort
Process start to fork
I'm the Parent Process, my pid = 4619
I'm the Child Process, my pid = 4620
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
```

```
wek_deng@Zero-Inspiron-7500:~/Assignment_1_120090661/source/program1$ ./program1
 ./floating
Process start to fork
I'm the Parent Process, my pid = 4694
I'm the Child Process, my pid = 4695
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
```

```
wek_deng@Zero-Inspiron-7500:~/Assignment_1_120090661/source/program1$ ./program1
 ./interrupt
Process start to fork
I'm the Parent Process, my pid = 4787
I'm the Child Process, my pid = 4788
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
```

```
wek_deng@Zero-Inspiron-7500:~/Assignment_1_120090661/source/program1$ ./program1
 ./pipe
Process start to fork
I'm the Parent Process, my pid = 4814
I'm the Child Process, my pid = 4815
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
```

```
wek_deng@Zero-Inspiron-7500:~/Assignment_1_120090661/source/program1$ ./program1
 ./segment_fault
Process start to fork
I'm the Parent Process, my pid = 4951
I'm the Child Process, my pid = 4952
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
```

```
wek_deng@Zero-Inspiron-7500:~/Assignment_1_120090661/source/program1$ ./program1
 ./trap
Process start to fork
I'm the Parent Process, my pid = 5007
I'm the Child Process, my pid = 5008
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
```

```
wek_deng@Zero-Inspiron-7500:~/Assignment_1_120090661/source/program1$ ./program1
 ./alarm
Process start to fork
I'm the Parent Process, my pid = 5057
I'm the Child Process, my pid = 5058
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
```

```
wek_deng@Zero-Inspiron-7500:~/Assignment_1_120090661/source/program1$ ./program1
 ./hangup
Process start to fork
I'm the Parent Process, my pid = 5103
I'm the Child Process, my pid = 5104
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
```

```
wek_deng@Zero-Inspiron-7500:~/Assignment_1_120090661/source/program1$ ./program1
 ./kill
Process start to fork
I'm the Parent Process, my pid = 5155
I'm the Child Process, my pid = 5156
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
```

```
wek_deng@Zero-Inspiron-7500:~/Assignment_1_120090661/source/program1$ ./program1
 ./stop
Process start to fork
I'm the Parent Process, my pid = 5204
I'm the Child Process, my pid = 5205
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
```

```
wek_deng@Zero-Inspiron-7500:~/Assignment_1_120090661/source/program1$ ./program1
 ./bus
Process start to fork
I'm the Parent Process, my pid = 5253
I'm the Child Process, my pid = 5254
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
```

```
wek_deng@Zero-Inspiron-7500:~/Assignment_1_120090661/source/program1$ ./program1
 ./illegal_instr
Process start to fork
I'm the Parent Process, my pid = 5302
I'm the Child Process, my pid = 5303
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
```

```
wek_deng@Zero-Inspiron-7500:~/Assignment_1_120090661/source/program1$ ./program1
 ./normal
Process start to fork
I'm the Parent Process, my pid = 5349
I'm the Child Process, my pid = 5350
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
```

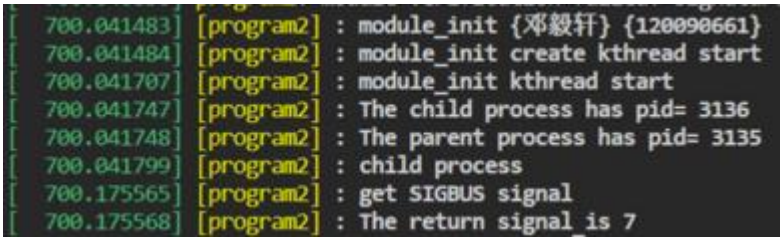
```
wek_deng@Zero-Inspiron-7500:~/Assignment_1_120090661/source/program1$ ./program1
 ./quit
Process start to fork
I'm the Parent Process, my pid = 5399
I'm the Child Process, my pid = 5400
Child process start to execute test program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

Parent process receives SIGCHLD signal
child process get SIGQUIT signal
```

```
wek_deng@Zero-Inspiron-7500:~/Assignment_1_120090661/source/program1$ ./program1
 ./terminate
Process start to fork
I'm the Parent Process, my pid = 5452
I'm the Child Process, my pid = 5453
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
```

program2:



Learn from Program:

Through this program, I learned how to create child processes and receive signals from them. I also learned how to compile a kernel, create child processes and receive signals in the environment in which the kernel was created.