CSC3150_A1_report

HuangPengxiang_119010108

10/5/2021

Contents

Overview

This is the report for CSC3150 project 1. This assignment mainly focuses on revealing the details of processes and multi-processes programming. The content can be divided into two parts, one is user mode multi-process programming in program 1, the other is kernel mode multi-process programming in program 2. And respectively, **program 1** implement the functionality including fork a children process in user mode, execute the test program, parent process wait for the child process terminate and handle and output different signals sended by children process. **program 2** implement the functionality including insert a new model and create a kernel thread, fork a new process in kernel mode, parent process wait until child process and handle some signals. **bonus** this is multi-process programming probelm, and mainly use recursive way to execute the file in terminal and also require us to communicate beteween each process. And here I use MMP to implment the communication among those processes.

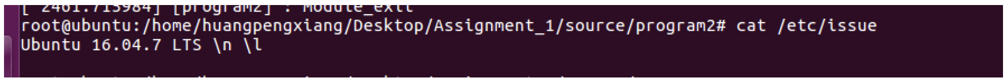
Important Declaration

For Program 2, the path for test in my program is "/opt/test".

Enviroment

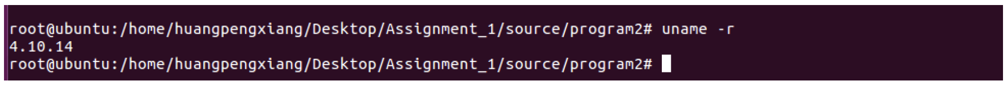
The environment of running my programs is the following:

OS version: Ubuntu 32-bit



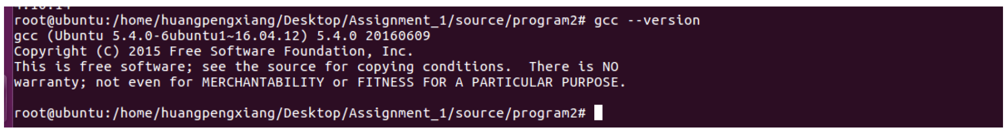
```
[ 2461.713984] [program2] : Module_exit
root@ubuntu:/home/huangpengxiang/Desktop/Assignment_1/source/program2# cat /etc/issue
Ubuntu 16.04.7 LTS \n \l
```

kernel version: Linux-4.10.14



```
root@ubuntu:/home/huangpengxiang/Desktop/Assignment_1/source/program2# uname -r
4.10.14
root@ubuntu:/home/huangpengxiang/Desktop/Assignment_1/source/program2# 
```

gcc –version: gcc (Ubuntu 5.4.0-6ubuntu1~16.04.10) 5.4.0 20160609



```
root@ubuntu:/home/huangpengxiang/Desktop/Assignment_1/source/program2# gcc --version
gcc (Ubuntu 5.4.0-6ubuntu1~16.04.12) 5.4.0 20160609
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

root@ubuntu:/home/huangpengxiang/Desktop/Assignment_1/source/program2# 
```

Program Execution Steps

Program 1

```
In order to execute the program 1, you should follow these steps
$ cd /* directory where the program.c located in */
$ make
$ ./program1 ./test_program_name ## you can only add one test here
then you can see the output
$ make clean # remember to clear
```

Program 2

```
First you should make sure you already install Ubuntu 4.10.14
make sure you already install the modules, export and extern the function including fork, wait, exec,
getname
Then you can do the following steps
$ su root # need you type your password here
$ cd /* directory where your program2.c located in */
$ gcc -o test test.c    # compile the test file
$ make                  # generate the kernel file
$ insmod program2.ko    # insert the kernel module file into kernel
$ rmmmod program2.ko    # remmove the kernel moule file
$ dmesg | tail -n 10    # print the last 10 message in the kernel log to check you successfully finish
the job
$ make clean # remember to clear
$ rm -f test

you may need to change the const char *path = "/home/huangpengxiang/Desktop/Assignment_1/source/program
/test"; to your own path to make it possible to run.
```

bonus

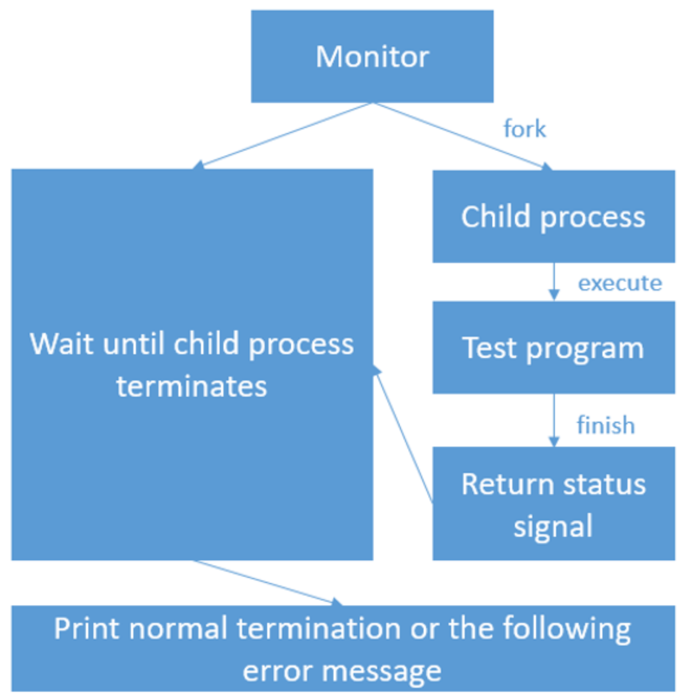
```
In order to execute the bonus, you should follow these steps
$ cd /* directory where the myfork.c located in */
$ make
$ ./myfork /* anyfile you want to add, you can add more than 1 file here */
$ make clean # remember to clear
```

Program Design

Here are the program design, it offer the basic idea for each program and some important detailded code.

Program 1

The program 1 chart flow is:

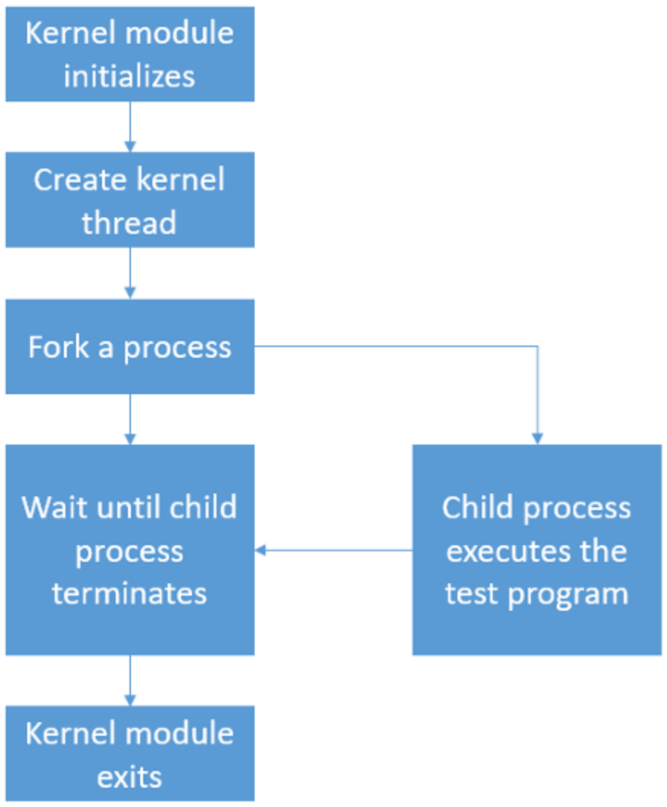

Monitor
fork
Child process
execute
Test program
finish
Return status signal
Wait until child process terminates
Print normal termination or the following error message

For This problem, we need to fork the process in user mode. we first use `fork()` function to fork a identical children process but with different logical memory. after that, we should use `execve()` function to execute the test program. At the same time, parent process should wait the process until it terminate and send some siganls back, we use `wait_pid()` here, also can use `wait()` function, which have no difference here. we also handle some different cases for different send back signal. In summary, 1. Fork a child process to execute test programs (15 of them) 2. Use wait() to let the parent process receives the SIGCHLD signal 3. Print out the termination information of child process (normal or abnormal)

some notice here: you can only test one test_name.c each time, and if you have the correct enviroment but still see some erro, it may have the full process in your computer, try to delete some and you are supposed to see some output.

program 2

The program 2 chart flow is:

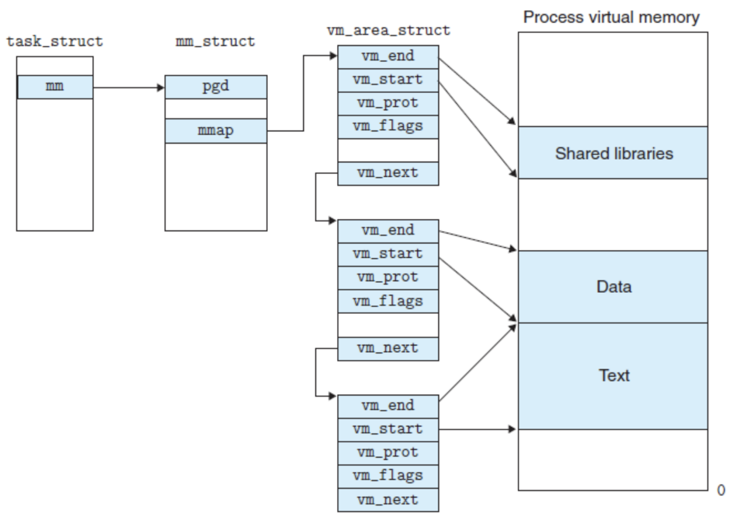

Kernel module initializes
Create kernel thread
Fork a process
Wait until child process terminates
Child process executes the test program
Kernel module exits

For this problem, we need to fork the process in the kernel mode. first we shall made the kernel and kernel modules in the computer before we type `make` command. `program_init()` function is used to initialize the kernel and create the kernel thread. `my_exec` is used to test the kernel. `my_fork()` is used to fork the process and get the pid for parent and child process. the `my_wait()`function is used to wait for the child process terminate and send some signal to parent. moreover, do fork is used to allocate a new process resources area in my fork with given function names and other parametes. my exec, the function called by do fork, will then comes in and start the execution of another test file. my exec takes care of locating the test file, passing in the arguments and start the execution. my wait do its job in the parent process. Within the my wait function, struct wait opts is constructed and paased to do wait as a parameter. Therefore, the parent process can check whether the child process is finished through the given child PID. In summary, 1. Create a kernel thread and run my_fork function 2. Fork a process to execute test.o 3. Use do_wait() to let the parent process wait for the child process 4. Print out pid of both parent and child processes 5. Catch the signal raised by the child process and print out related log 6. Recompile the Linux kernel source code to use its functions

some notice here: you may need to change the `const char *path = "/home/huangpengxiang/Desktop/Assignment_1/so` `// you may need to change this path to you own in order to test;` to your own path to make it possible to run.

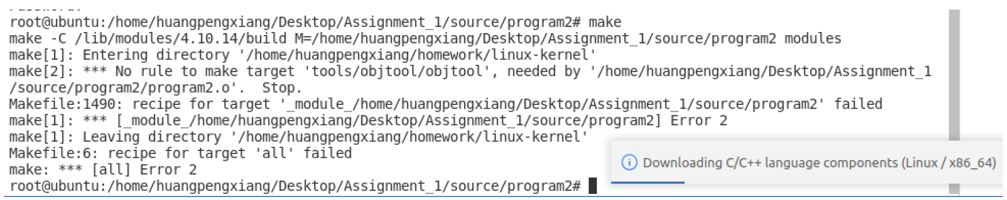
bonus

For this problem, this is a typical muti-process problem in user mode. The difficulty for this problem is how to create a process tree and how to print out the final information for those tree. For first one, I use `fork_tree()` to encounter it. it is a recursive function, it will create a fork process tree, and for each child process, it will send the signal to the parent signal, and every parent signal is terminated until receive the signal sended by child process. For the second one, i use the function called `mmp`. Mmap is a method of memory-mapping files. A file or other object is mapped to the address space of a process to achieve the mapping between the file disk address and a segment of virtual address in the process virtual address space. After such mapping is achieved, the process can use Pointers to read and write the memory, and the system will automatically write back dirty pages to the corresponding file disk, that is, the operation on the file is completed without calling system call functions such as read and write. Conversely, changes made by the kernel space to this area directly reflect user space, allowing file sharing between different processes.



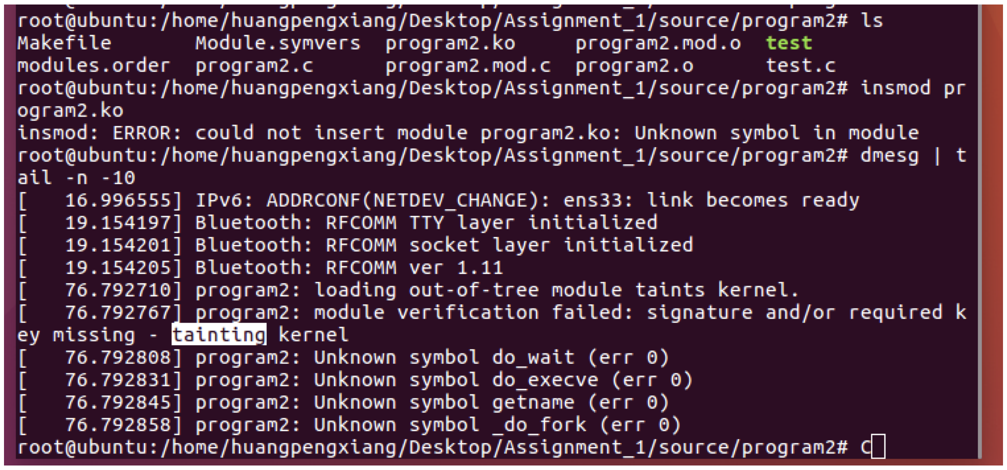
Some problems I have encountered

The environment is the most difficult part i have dealed with. for this one, i found that i have lost a objtool but i already found the objtool.o in my files. so i have to rebuild the kernel to fix it.



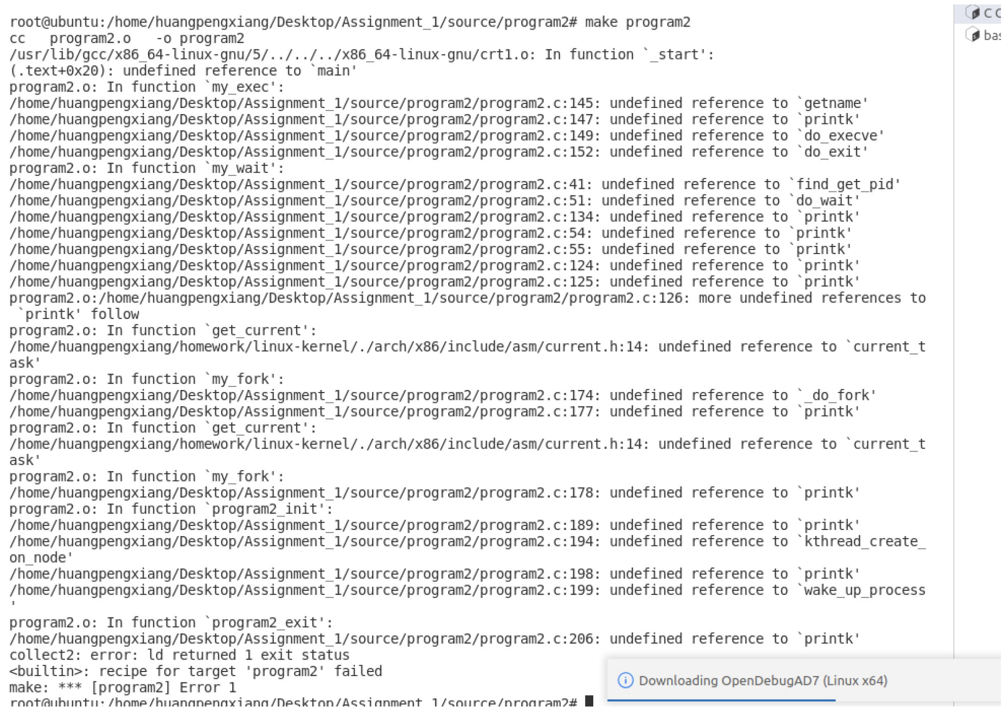
```
root@ubuntu:/home/huangpengxiang/Desktop/Assignment_1/source/program2# make
make -C /lib/modules/4.10.14/build M=/home/huangpengxiang/Desktop/Assignment_1/source/program2 modules
make[1]: Entering directory '/home/huangpengxiang/homework/linux-kernel'
make[2]: *** No rule to make target 'tools/objtool/objtool', needed by '/home/huangpengxiang/Desktop/Assignment_1
/source/program2/program2.o'.  Stop.
Makefile:1490: recipe for target '_module_/home/huangpengxiang/Desktop/Assignment_1/source/program2' failed
make[1]: *** [_module_/home/huangpengxiang/Desktop/Assignment_1/source/program2] Error 2
make[1]: Leaving directory '/home/huangpengxiang/homework/linux-kernel'
Makefile:6: recipe for target 'all' failed
make: *** [all] Error 2
root@ubuntu:/home/huangpengxiang/Desktop/Assignment_1/source/program2#
```

for this one, in program 2, it told me i have missed some symbols, then i find those kernel file and export the function, then i modify the kernel.



```
root@ubuntu:/home/huangpengxiang/Desktop/Assignment_1/source/program2# ls
Makefile       Module.symvers   program2.ko      program2.mod.o   test
modules.order  program2.c       program2.mod.c   program2.o       test.c
root@ubuntu:/home/huangpengxiang/Desktop/Assignment_1/source/program2# insmod pr
ogram2.ko
insmod: ERROR: could not insert module program2.ko: Unknown symbol in module
root@ubuntu:/home/huangpengxiang/Desktop/Assignment_1/source/program2# dmesg | t
ail -n -10
[   16.996555] IPv6: ADDRCONF(NETDEV_CHANGE): ens33: link becomes ready
[   19.154197] Bluetooth: RFCOMM TTY layer initialized
[   19.154201] Bluetooth: RFCOMM socket layer initialized
[   19.154205] Bluetooth: RFCOMM ver 1.11
[   76.792710] program2: loading out-of-tree module taints kernel.
[   76.792767] program2: module verification failed: signature and/or required k
ey missing - tainting kernel
[   76.792808] program2: Unknown symbol do_wait (err 0)
[   76.792831] program2: Unknown symbol do_execve (err 0)
[   76.792845] program2: Unknown symbol getname (err 0)
[   76.792858] program2: Unknown symbol _do_fork (err 0)
root@ubuntu:/home/huangpengxiang/Desktop/Assignment_1/source/program2# c
```

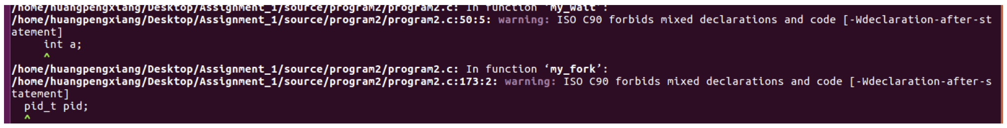
for this one, it is also the undefined symbol problem, i found that i miss extern the symbol in source file.



```
root@ubuntu:/home/huangpengxiang/Desktop/Assignment_1/source/program2# make program2
cc   program2.o   -o program2
/usr/lib/gcc/x86_64-linux-gnu/5/../../../x86_64-linux-gnu/crt1.o: In function `_start':
(.text+0x20): undefined reference to `main'
program2.o: In function `my_exec':
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:145: undefined reference to `getname'
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:147: undefined reference to `printk'
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:149: undefined reference to `do_execve'
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:152: undefined reference to `do_exit'
program2.o: In function `my_wait':
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:41: undefined reference to `find_get_pid'
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:51: undefined reference to `do_wait'
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:134: undefined reference to `printk'
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:54: undefined reference to `printk'
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:55: undefined reference to `printk'
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:124: undefined reference to `printk'
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:125: undefined reference to `printk'
program2.o:/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:126: more undefined references to
 `printk' follow
program2.o: In function `get_current':
/home/huangpengxiang/homework/linux-kernel/./arch/x86/include/asm/current.h:14: undefined reference to `current_t
ask'
program2.o: In function `my_fork':
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:174: undefined reference to `_do_fork'
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:177: undefined reference to `printk'
program2.o: In function `get_current':
/home/huangpengxiang/homework/linux-kernel/./arch/x86/include/asm/current.h:14: undefined reference to `current_t
ask'
program2.o: In function `my_fork':
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:178: undefined reference to `printk'
program2.o: In function `program2_init':
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:189: undefined reference to `printk'
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:194: undefined reference to `kthread_create_
on_node'
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:198: undefined reference to `printk'
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:199: undefined reference to `wake_up_process
'

program2.o: In function `program2_exit':
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:206: undefined reference to `printk'
collect2: error: ld returned 1 exit status
<builtin>: recipe for target 'program2' failed
make: *** [program2] Error 1
root@ubuntu:/home/huangpengxiang/Desktop/Assignment_1/source/program2#
```

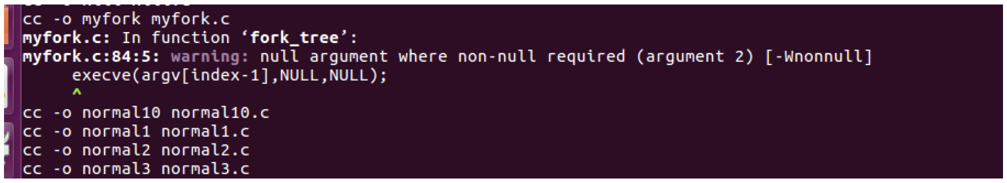
Limitations: some warnings I missed in my program

In program 2, it warns me that ISO C90 have some mixed decalrations in my function. Then I google it, and found that i can ignore the warnings, it won't affect my result.



```
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c: In function 'my_wait':
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:50:5: warning: ISO C90 forbids mixed declarations and code [-Wdeclaration-after-st
atement]
     int a;
     ^
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c: In function 'my_fork':
/home/huangpengxiang/Desktop/Assignment_1/source/program2/program2.c:173:2: warning: ISO C90 forbids mixed declarations and code [-Wdeclaration-after-s
tatement]
  pid_t pid;
  ^
```

In bonus, for `execev()` function, it warns me that I change one pointer to `NULL` , but i only need one argument in this function, so i choose to ignore it and set this argument to `NULL`.

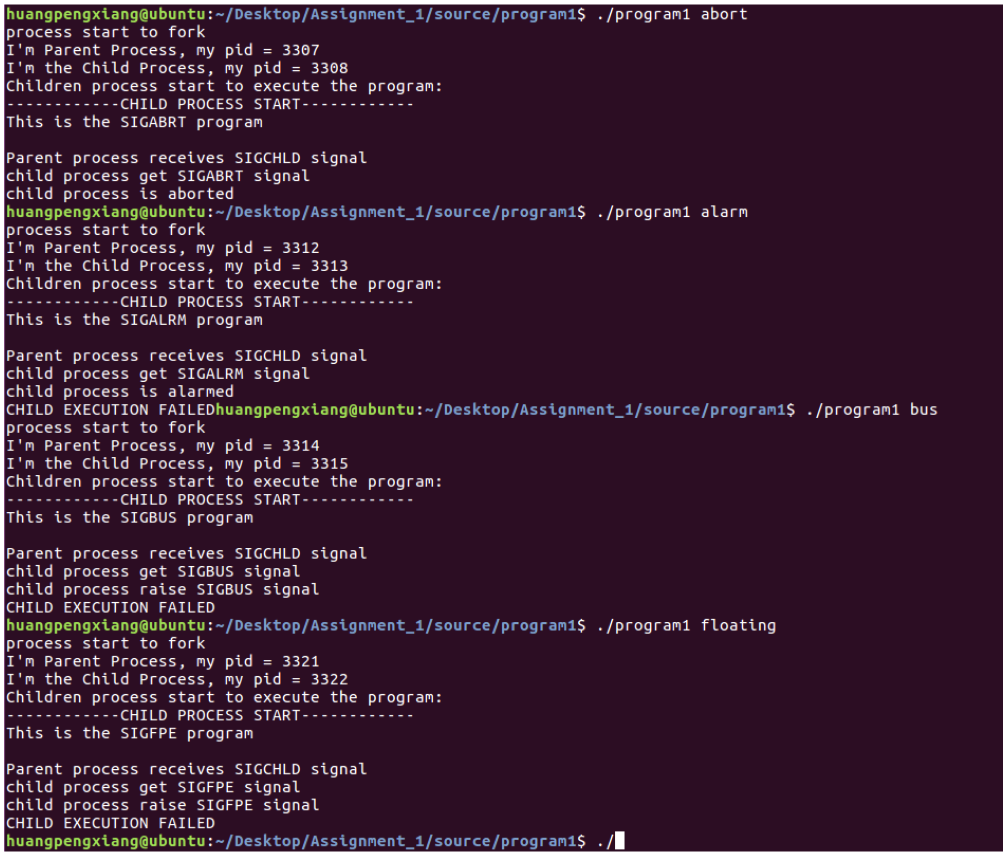


```
cc -o myfork myfork.c
myfork.c: In function 'fork_tree':
myfork.c:84:5: warning: null argument where non-null required (argument 2) [-Wnonnull]
     execve(argv[index-1],NULL,NULL);
     ^
cc -o normal10 normal10.c
cc -o normal1 normal1.c
cc -o normal2 normal2.c
cc -o normal3 normal3.c
```

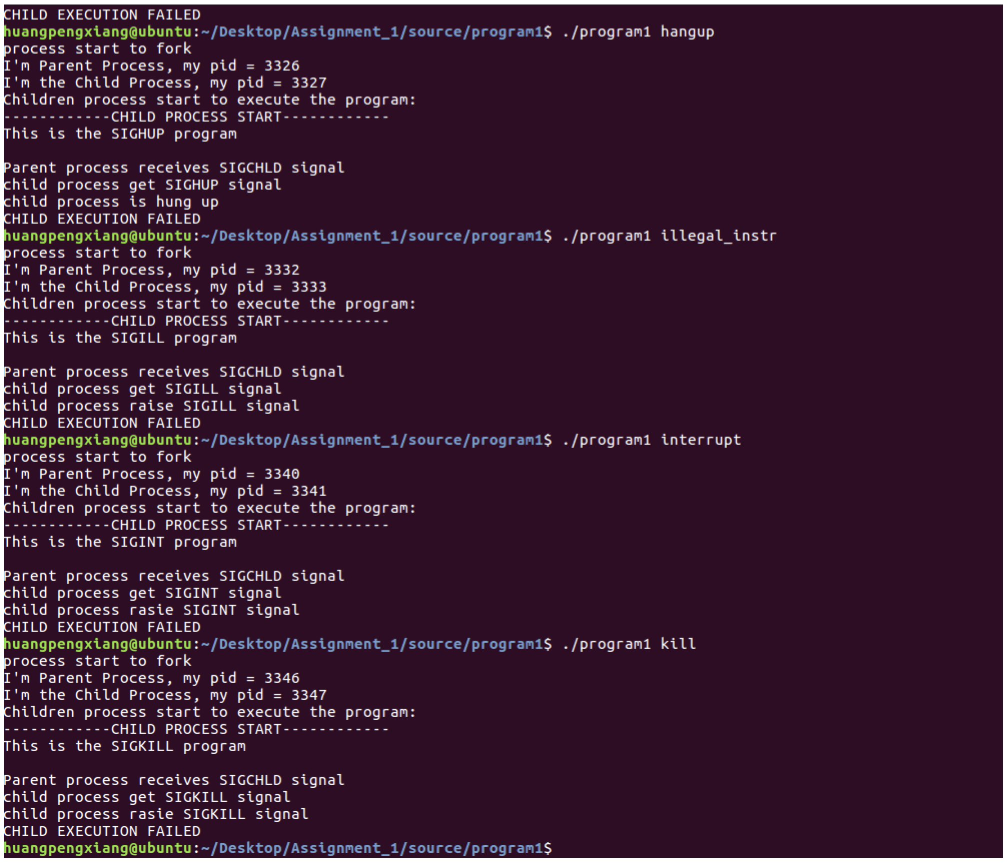
Output Screenshots Demo

Here are some output Screenshots Demo:

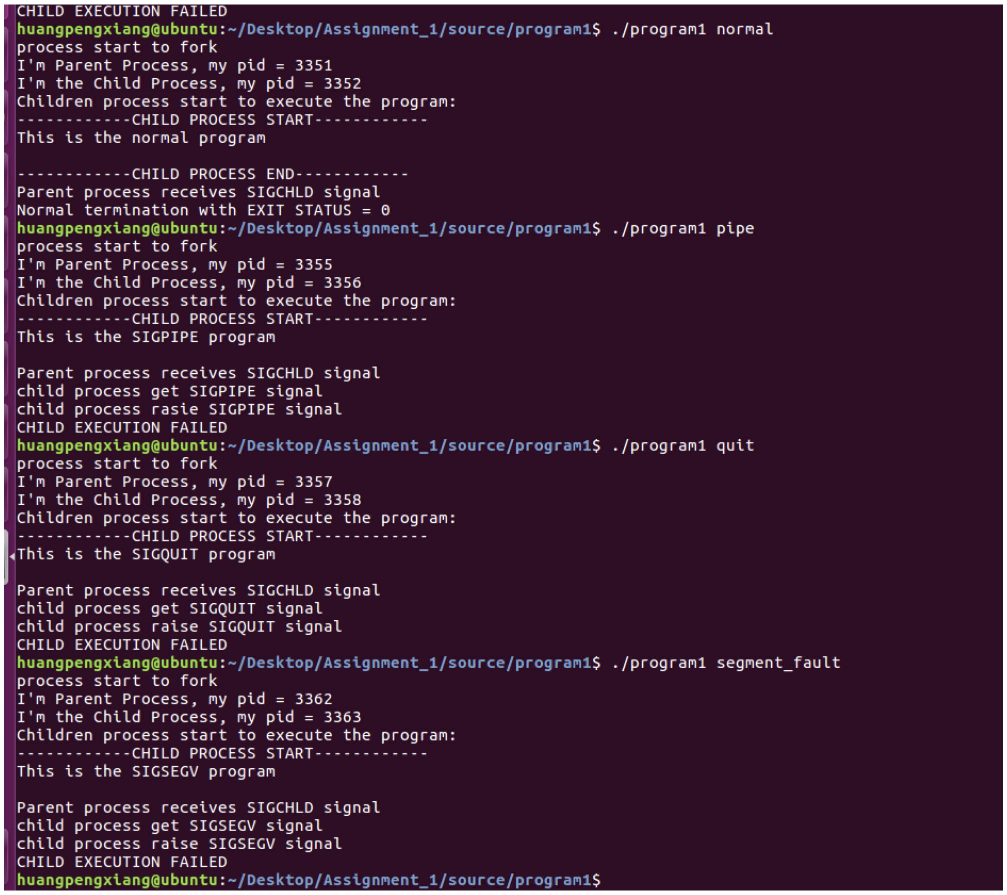
program 1



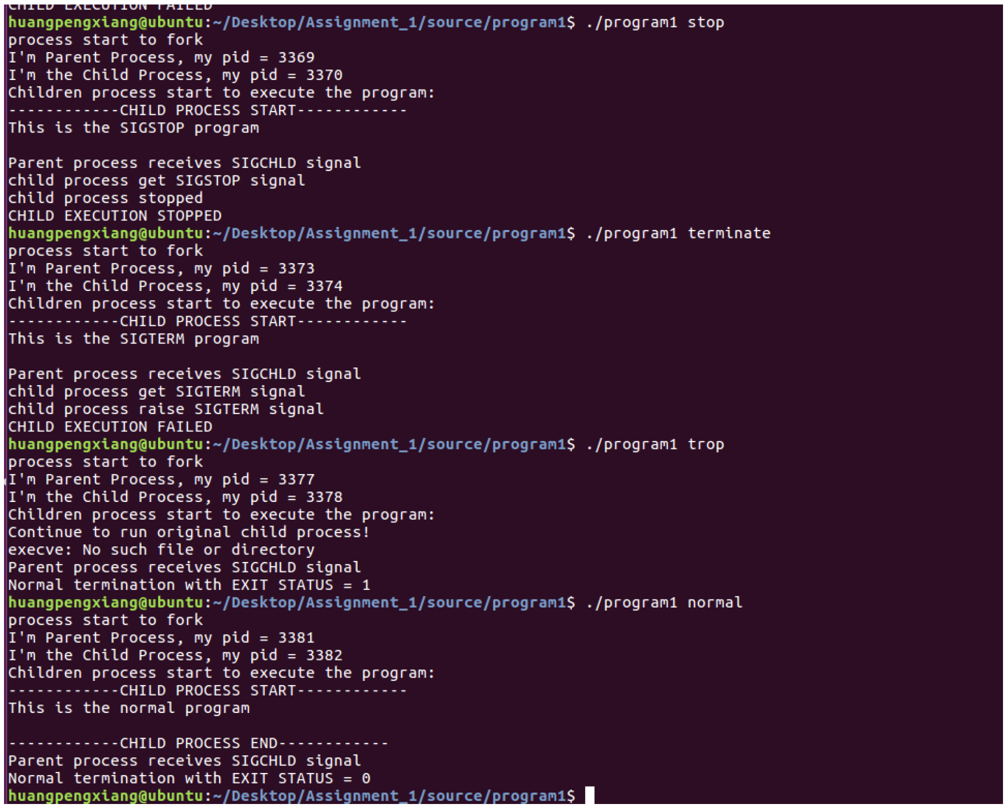
```
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/program1$ ./program1 abort
process start to fork
I'm Parent Process, my pid = 3307
I'm the Child Process, my pid = 3308
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGABRT program

Parent process receives SIGCHLD signal
child process get SIGABRT signal
child process is aborted
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/program1$ ./program1 alarm
process start to fork
I'm Parent Process, my pid = 3312
I'm the Child Process, my pid = 3313
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGALRM program

Parent process receives SIGCHLD signal
child process get SIGALRM signal
child process is alarmed
CHILD EXECUTION FAILEDhuangpengxiang@ubuntu:~/Desktop/Assignment_1/source/program1$ ./program1 bus
process start to fork
I'm Parent Process, my pid = 3314
I'm the Child Process, my pid = 3315
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGBUS program

Parent process receives SIGCHLD signal
child process get SIGBUS signal
child process raise SIGBUS signal
CHILD EXECUTION FAILED
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/program1$ ./program1 floating
process start to fork
I'm Parent Process, my pid = 3321
I'm the Child Process, my pid = 3322
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGFPE program

Parent process receives SIGCHLD signal
child process get SIGFPE signal
child process raise SIGFPE signal
CHILD EXECUTION FAILED
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/program1$ ./
```



```
CHILD EXECUTION FAILED
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/program1$ ./program1 hangup
process start to fork
I'm Parent Process, my pid = 3326
I'm the Child Process, my pid = 3327
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGHUP program

Parent process receives SIGCHLD signal
child process get SIGHUP signal
child process is hung up
CHILD EXECUTION FAILED
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/program1$ ./program1 illegal_instr
process start to fork
I'm Parent Process, my pid = 3332
I'm the Child Process, my pid = 3333
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGILL program

Parent process receives SIGCHLD signal
child process get SIGILL signal
child process raise SIGILL signal
CHILD EXECUTION FAILED
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/program1$ ./program1 interrupt
process start to fork
I'm Parent Process, my pid = 3340
I'm the Child Process, my pid = 3341
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGINT program

Parent process receives SIGCHLD signal
child process get SIGINT signal
child process rasie SIGINT signal
CHILD EXECUTION FAILED
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/program1$ ./program1 kill
process start to fork
I'm Parent Process, my pid = 3346
I'm the Child Process, my pid = 3347
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGKILL program

Parent process receives SIGCHLD signal
child process get SIGKILL signal
child process rasie SIGKILL signal
CHILD EXECUTION FAILED
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/program1$
```



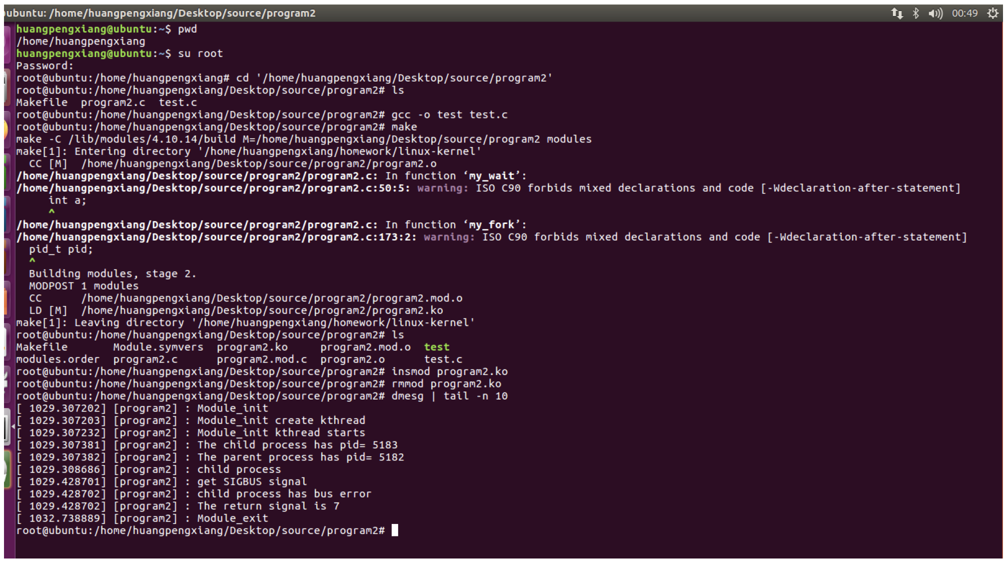
```
CHILD EXECUTION FAILED
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/program1$ ./program1 normal
process start to fork
I'm Parent Process, my pid = 3351
I'm the Child Process, my pid = 3352
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the normal program

------------CHILD PROCESS END------------
Parent process receives SIGCHLD signal
Normal termination with EXIT STATUS = 0
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/program1$ ./program1 pipe
process start to fork
I'm Parent Process, my pid = 3355
I'm the Child Process, my pid = 3356
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGPIPE program

Parent process receives SIGCHLD signal
child process get SIGPIPE signal
child process rasie SIGPIPE signal
CHILD EXECUTION FAILED
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/program1$ ./program1 quit
process start to fork
I'm Parent Process, my pid = 3357
I'm the Child Process, my pid = 3358
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGQUIT program

Parent process receives SIGCHLD signal
child process get SIGQUIT signal
child process raise SIGQUIT signal
CHILD EXECUTION FAILED
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/program1$ ./program1 segment_fault
process start to fork
I'm Parent Process, my pid = 3362
I'm the Child Process, my pid = 3363
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGSEGV program

Parent process receives SIGCHLD signal
child process get SIGSEGV signal
child process raise SIGSEGV signal
CHILD EXECUTION FAILED
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/program1$
```



```
CHILD EXECUTION FAILED
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/program1$ ./program1 stop
process start to fork
I'm Parent Process, my pid = 3369
I'm the Child Process, my pid = 3370
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGSTOP program

Parent process receives SIGCHLD signal
child process get SIGSTOP signal
child process stopped
CHILD EXECUTION STOPPED
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/program1$ ./program1 terminate
process start to fork
I'm Parent Process, my pid = 3373
I'm the Child Process, my pid = 3374
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the SIGTERM program

Parent process receives SIGCHLD signal
child process get SIGTERM signal
child process raise SIGTERM signal
CHILD EXECUTION FAILED
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/program1$ ./program1 trop
process start to fork
I'm Parent Process, my pid = 3377
I'm the Child Process, my pid = 3378
Children process start to execute the program:
Continue to run original child process!
execve: No such file or directory
Parent process receives SIGCHLD signal
Normal termination with EXIT STATUS = 1
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/program1$ ./program1 normal
process start to fork
I'm Parent Process, my pid = 3381
I'm the Child Process, my pid = 3382
Children process start to execute the program:
------------CHILD PROCESS START------------
This is the normal program

------------CHILD PROCESS END------------
Parent process receives SIGCHLD signal
Normal termination with EXIT STATUS = 0
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/program1$
```

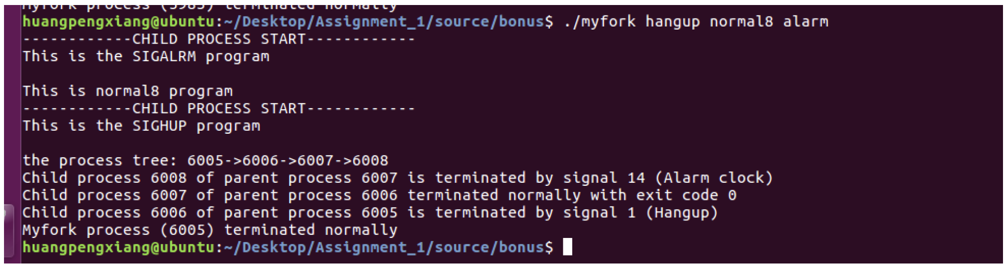
program 2

the output is showing below:

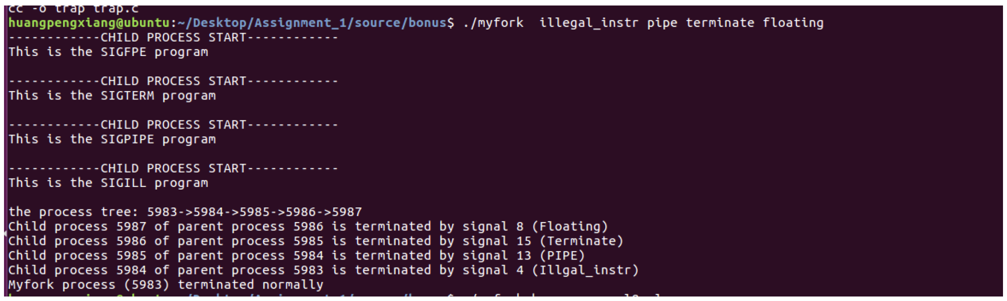


```
ubuntu: /home/huangpengxiang/Desktop/source/program2                00:49
huangpengxiang@ubuntu:~$ pwd
/home/huangpengxiang
huangpengxiang@ubuntu:~$ su root
Password:
root@ubuntu:/home/huangpengxiang# cd '/home/huangpengxiang/Desktop/source/program2'
root@ubuntu:/home/huangpengxiang/Desktop/source/program2# ls
Makefile  program2.c  test.c
root@ubuntu:/home/huangpengxiang/Desktop/source/program2# gcc -o test test.c
root@ubuntu:/home/huangpengxiang/Desktop/source/program2# make
make -C /lib/modules/4.10.14/build M=/home/huangpengxiang/Desktop/source/program2 modules
make[1]: Entering directory '/home/huangpengxiang/homework/linux-kernel'
  CC [M]  /home/huangpengxiang/Desktop/source/program2/program2.o
/home/huangpengxiang/Desktop/source/program2/program2.c: In function ‘my_wait’:
/home/huangpengxiang/Desktop/source/program2/program2.c:50:5: warning: ISO C90 forbids mixed declarations and code [-Wdeclaration-after-statement]
     int a;
     ^
/home/huangpengxiang/Desktop/source/program2/program2.c: In function ‘my_fork’:
/home/huangpengxiang/Desktop/source/program2/program2.c:173:2: warning: ISO C90 forbids mixed declarations and code [-Wdeclaration-after-statement]
  pid_t pid;
  ^
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/huangpengxiang/Desktop/source/program2/program2.mod.o
  LD [M]  /home/huangpengxiang/Desktop/source/program2/program2.ko
make[1]: Leaving directory '/home/huangpengxiang/homework/linux-kernel'
root@ubuntu:/home/huangpengxiang/Desktop/source/program2# ls
Makefile       Module.symvers  program2.ko     program2.mod.o  test
modules.order  program2.c      program2.mod.c  program2.o      test.c
root@ubuntu:/home/huangpengxiang/Desktop/source/program2# insmod program2.ko
root@ubuntu:/home/huangpengxiang/Desktop/source/program2# rmmod program2.ko
root@ubuntu:/home/huangpengxiang/Desktop/source/program2# dmesg | tail -n 10
[ 1029.307202] [program2] : Module_init
[ 1029.307203] [program2] : Module_init create kthread
[ 1029.307232] [program2] : Module_init kthread starts
[ 1029.307381] [program2] : The child process has pid= 5183
[ 1029.307382] [program2] : The parent process has pid= 5182
[ 1029.308686] [program2] : child process
[ 1029.428701] [program2] : get SIGBUS signal
[ 1029.428702] [program2] : child process has bus error
[ 1029.428702] [program2] : The return signal is 7
[ 1032.738889] [program2] : Module_exit
root@ubuntu:/home/huangpengxiang/Desktop/source/program2# 
```

bonus



```
Myfork process (5983) terminated normally
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/bonus$ ./myfork hangup normal8 alarm
------------CHILD PROCESS START------------
This is the SIGALRM program

This is normal8 program
------------CHILD PROCESS START------------
This is the SIGHUP program

the process tree: 6005->6006->6007->6008
Child process 6008 of parent process 6007 is terminated by signal 14 (Alarm clock)
Child process 6007 of parent process 6006 terminated normally with exit code 0
Child process 6006 of parent process 6005 is terminated by signal 1 (Hangup)
Myfork process (6005) terminated normally
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/bonus$ 
```



```
cc -o trap trap.c
huangpengxiang@ubuntu:~/Desktop/Assignment_1/source/bonus$ ./myfork  illegal_instr pipe terminate floating
------------CHILD PROCESS START------------
This is the SIGFPE program

------------CHILD PROCESS START------------
This is the SIGTERM program

------------CHILD PROCESS START------------
This is the SIGPIPE program

------------CHILD PROCESS START------------
This is the SIGILL program

the process tree: 5983->5984->5985->5986->5987
Child process 5987 of parent process 5986 is terminated by signal 8 (Floating)
Child process 5986 of parent process 5985 is terminated by signal 15 (Terminate)
Child process 5985 of parent process 5984 is terminated by signal 13 (PIPE)
Child process 5984 of parent process 5983 is terminated by signal 4 (Illgal_instr)
Myfork process (5983) terminated normally
```