Homework 1, due 1/28/2016 (Thu)

Hau Tao

1. (10 points)
   How many processes does the following piece of code create? Why?

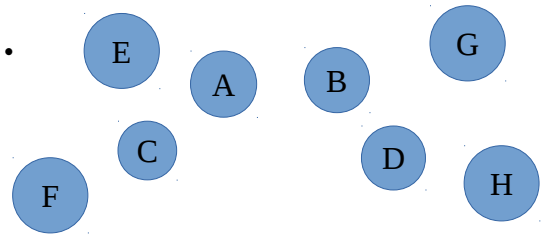
```
int main()
{
  fork();
  fork();
  fork();
  return 0;
}
```

The first fork creates 2 processes A, and B

The second fork creates 2 more processes from 2 previous processes (A,B,C,D)

The third fork creates 4 more processes from their ancestors (A,B,C,D,E,F,G,H)

The total process is : 8

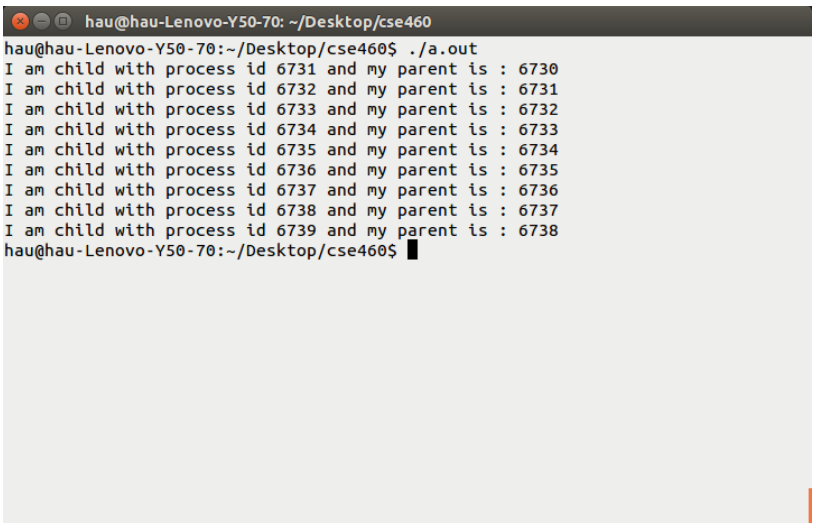
- 

2. (20 points)

a) Write a C-program that creates a chain of 10 processes and prints out their process ids and relationships. For example, process 1 is the parent of process 2, process 2 is the parent of process 3, process 3 is the parent of 4 and so on. Each child has to print out all her ancestors identified by the process ids.

Source code

```cpp
//chain of 10 processes.cpp
#include <iostream>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>

using namespace std;

int main()
{
  pid_t pid;          //process id
  for( int i = 0; i < 9; i++){
        pid = fork();         //create another process
        if ( pid < 0 ) {      //fail
                cout << "\nFork failed" << endl;
                exit ( -1 );
        } else if ( pid == 0 ) {      //child

                cout << "I am child with process id " << getpid()
                << " and my parent is : " << getppid() << endl;

        } else {                      //parent
                wait ( NULL );              //wait for child
                exit ( 0 );

        }

  }
}
```



b) Write a C-program that creates a fan of 10 processes. That is, process 1 is the parent of prcocesses 2, 3, 4, 5, 6 and so on.

Source code:

```
//chain of 10 processes.cpp
#include <iostream>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>

using namespace std;

int main()
{
  pid_t pid;           //process id
  for( int i = 0; i < 10; i++){
        pid = fork();          //create another process


        if ( pid < 0 ) {       //fail
               cout << "\nFork failed" << endl;
               exit ( -1 );
        } else if ( pid == 0 ) {      //child

               cout << "I am child with process id " << getpid()
               << " and my parent is : " << getppid() << endl;
               exit ( 0 );

        } else {                      //parent
               wait ( NULL );               //wait for child


        }


  }

}
```

```
hau@hau-Lenovo-Y50-70: ~/Desktop/cse460
hau@hau-Lenovo-Y50-70:~/Desktop/cse460$ g++ fan_process.cpp
hau@hau-Lenovo-Y50-70:~/Desktop/cse460$ ./a.out
I am child with process id 7695 and my parent is : 7694
I am child with process id 7696 and my parent is : 7694
I am child with process id 7697 and my parent is : 7694
I am child with process id 7698 and my parent is : 7694
I am child with process id 7699 and my parent is : 7694
I am child with process id 7700 and my parent is : 7694
I am child with process id 7701 and my parent is : 7694
I am child with process id 7702 and my parent is : 7694
I am child with process id 7703 and my parent is : 7694
I am child with process id 7704 and my parent is : 7694
hau@hau-Lenovo-Y50-70:~/Desktop/cse460$
```

3. 10 points)

a) Write a simple program named **test1.cpp**, which contains an infinite **while** loop. Compile the program to an executable named **test1** and run it in the background.

Source code:

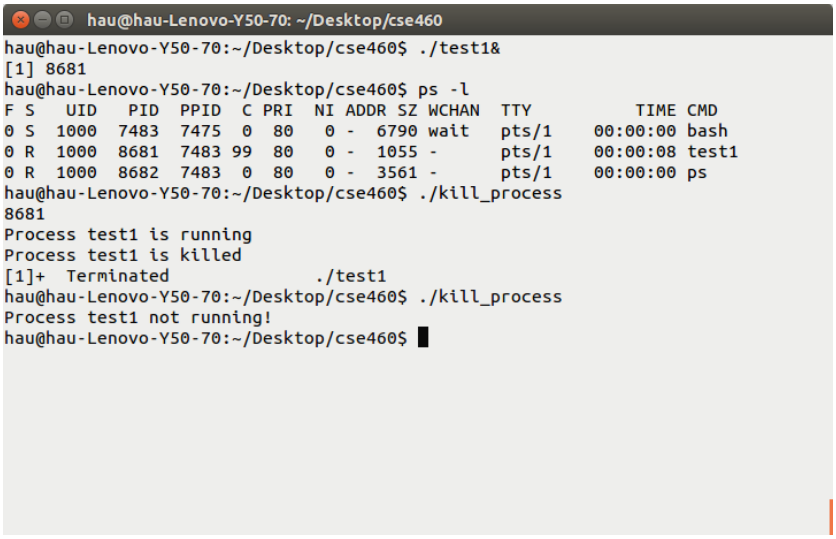
```
/*
 *  test1.cpp
 *  Dummy program running an infite loop used for hw1-part3.
 *  Compile: g++ -o test1 test1.cpp
 *  Run: ./test1 &
 */

int main()
{
  while ( 1 );

  return 0;
}
```

```
hau@hau-Lenovo-Y50-70: ~/Desktop/cse460
hau@hau-Lenovo-Y50-70:~/Desktop/cse460$ ./test1&
[1] 7936
hau@hau-Lenovo-Y50-70:~/Desktop/cse460$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000  7483  7475  0  80   0 -  6790 wait   pts/1    00:00:00 bash
0 R  1000  7936  7483 95  80   0 -  1055 -      pts/1    00:00:02 test1
0 R  1000  7937  7483  0  80   0 -  3561 -      pts/1    00:00:00 ps
hau@hau-Lenovo-Y50-70:~/Desktop/cse460$ ^C
hau@hau-Lenovo-Y50-70:~/Desktop/cse460$
```

b) Write a shell script that searches for whether the process **test1** is in the system. If it is not, your script displays the message 'Process test1 not running!'. If it is running, your script kills the process, and displays the message 'Process test1 killed!'.

```
if pgrep test1
then
        echo "Process test1 is running"
        pkill  test1 # kill all test1 processes are running
        echo "Process test1 is killed"
else
        echo "Process test1 not running!"

fi
```

4.


hau@hau-Lenovo-Y50-70: ~/Desktop/cse460
hau@hau-Lenovo-Y50-70:~/Desktop/cse460$ ./test1&
[1] 8681
hau@hau-Lenovo-Y50-70:~/Desktop/cse460$ ps -l
F S UID PID PPID C PRI NI ADDR SZ WCHAN TTY TIME CMD
0 S 1000 7483 7475 0 80 0 - 6790 wait pts/1 00:00:00 bash
0 R 1000 8681 7483 99 80 0 - 1055 - pts/1 00:00:08 test1
0 R 1000 8682 7483 0 80 0 - 3561 - pts/1 00:00:00 ps
hau@hau-Lenovo-Y50-70:~/Desktop/cse460$ ./kill_process
8681
Process test1 is running
Process test1 is killed
[1]+ Terminated ./test1
hau@hau-Lenovo-Y50-70:~/Desktop/cse460$ ./kill_process
Process test1 not running!
hau@hau-Lenovo-Y50-70:~/Desktop/cse460$