ECE437/CS481 Operating Systems, Fall 2017

Programming Assignment #03 (3%)

Due: Thursday, 09/28/2017

You are encouraged to work on this assignment in a group of two.

1. **(70%) Fibonacci using Fork and message queues**

Let's use the ***fork*** system call to create Linux processes to compute the Fibonacci sequence of a given number. Write a program that takes two parameters: "-F ***n*** –S ***m***", where ***n*** is an input number, to be used to computer the **n**th number of the Fibonacci sequence and **m** specifies a computing threshold.

- In general, fib(x)=fib(x-1)+fib(x-2), where fib(0) = 0 and fib(1) = 1
- if (x-1) > m (or/and (x-2)>m)), your program must use the ***fork*** system call to recursively spawn child processes to compute the next Fibonacci numbers.
  - Your need to create a POSIX message queue for Read and Write before your child processes are created. Thus, the parent process and child processes will communicate via the message queue.
  - When the child process complete its Fibonacci number computation, it will send the result back to its parent process via the message queue.
  - The parent process will receive the results by receive messages from its child processes.
  - The parent process should wait for all its child processes terminated and then close the message queue, if existing.
- if (x-1) <= m (or/and (x-2)<=m)), your program recursively call fib_seq(x-1) to compute the next Fibonacci numbers

```
int fib_seq(int x) { /* slow/recursive implementation of Fib */
    int i, rint = (rand()%30);  double dummy;
    for (i=0; i<rint*100; i++) {dummy=2.345*i*8.765/1.234;}
    if (x==0) return(0); else if (x==1) return(1); else return(fib_seq(x-1)+fib_seq(x-2));
}
```

Your program should follow the given input/output format: "myfib –F ***n*** –S ***m***". It should output/print ONLY the ***n***th Fibonacci number. See the example below.

```
$ myfib –F 6 –S 6
8
$ myfib –F 6 –S 3
8
```

The above two cases will output the same result "8". The former one will do every computation in sequential while the later one will create several child processes to do the computation. In a multicore system, the later one should take less time to complete since multiple processes can utilize multiple cores.

Hint:

- You can use C-library call "getopt" to handle your commend line input
- Use POSIX message queue to do interprocess communication.
- POSIX queue is named by a string "/xxyy", you can use your PID as a part of string to make your queue unique.
- If you left too many queues in system, go to "/dev/mqueue" to manually remove left over queues.
- Be careful about how many child processes to be created.

Name your source code for Q1 as "PA03fib_XXWW.c" , or "PA03fib_XXWW_ZZYY.c" if you are in a group, then submit it as a single source file onto PA03. (W or Y will be your first name initial (2-char) and X or Z will be your last name initial (2-char), such as, "PA03fib-SHWE for Wennie Shu".) We'll compile and test your program from there.

2. **(30%) Process communication using signals**

   Write a program, signals.c that will utilize signal handlers to intercept keyboard interrupts to manage control of a child process. Your main program should fork and exec a new child process to run the "yes" command (use man yes to learn more). You should then register signal handlers for both ctrl-c and ctrl-z in your main program. The ctrl-z signal handler should toggle the child yes stop the child process if it is running, or resume the child process if it is stopped, all while keeping the main program running. The ctrl-c signal should kill the child process, and then exit the main process. In both signal handlers, add print statements indicating what the program is doing – i.e. "ctrl-z caught, stopping child process". Your main program should sit in an infinite loop once the setup has been completed. It should only exit when the user hits ctrl-c, after cleaning up the child process.

Name your source code for Q2 as "PA03sig_XW.c" , or "PA03sig_XW_ZY.c" if you are in a group, then submit into PA03b . We'll compile and test your program from there.