COMP 2560 Fall 2020

Assignment 5

Due: Dec. 1, 11:59 P.M.

1. Write a C program simulating alarm clock. In the main function, you fork a child process, and the child process sleeps for 5 seconds (the number of seconds is an command line argument, see below for a sample run) , after which the child process sends the signal (SIGALRM) to its parent process. The parent process, after forking the child process, pause, upon receiving the SIGALRM signal, and prints out a message "Ding!". The following is a sample run

   ```
   $ ./alarm 5
   alarm application starting
   waiting for alarm to go off
   <5 second pause>
   Ding!
   done
   ```

   Submit both your source code and the script files.

2. Write a C program where two child processes are created using fork(). The parent process and the two child processes communicate with each other via a pipe. More specifically, The first child process writes 5 random number in the range of 0-99 to the pipe, and the second child process also writes another 5 random number in the same range to the pipe. The order of which child process writes to the pipe does not matter and the 10 random numbers could be interleaved between the two child processes.

   The parent process reads out the 10 random numbers from the pipe and print them out on the display (terminal).

   Submit both your source code and the script files.

3. Write a C program with the parent process and a child process communicating with each other using a pipe. The child process will execute the shell command provided by the user via command line arguments. The result of executing this shell command is passed to the parent process using a pipe. The parent process will write the result into a file called

result.txt and acknowledge the user on the screen with the shell command and the total number of bytes in the result.

For simplicity, the shell command contains only the command name, no argument.

Sample run:

```
>>>>> ls
pipeexec        pipeexec.c
>>>>> pipeexec ls
The result of ls is written into result.txt with total 20 bytes.
>>>>> more result.txt
pipeexec
pipeexec.c
>>>>> pipeexec uname
The result of uname is written into result.txt with total 7 bytes.
>>>>> more result.txt
Darwin
>>>>> od -c result.txt
0000000    D   a   r   w   i   n  \n
0000007
>>>>>
```