Homework 3

Computer Systems Organization II, V22.0202.003 Spring 2005, Professor Yap

Due: Thu Feb 22.

INSTRUCTIONS:

- This homework in Interprocess Communication is worth 35 points.
- Please read questions carefully. All handed in work must your own (even if you have discussed with other students)
- Please start your programming part immediately! Feel free to send me and the grader emails when you run into problems.

- In this programming homework, you are to simulate the concurrent solution of some computational task using processes. In a future project, you will solve the same problem using threads.
- The task is to to compute the GCD of various pairs of numbers. Given two numbers m and n where $m \geq n > 0$ are two integers, their $GCD(m, n)$ is defined to be the largest number that divides both m and n.

  For instance, $GCD(15, 9) = 3$. There is the well-known Euclidean algorithm for computing GCD. Starting from the numbers

  $$m_0 = m, m_1 = n$$

  we compute the sequence of integers,

  $$(m_0, m_1, m_2, \ldots, m_k, 0)$$

  where $m_{i+1} = m_{i-1} \mod m_i$ for all $i = 2, 3, \ldots$. Thus $0 \leq m_{i+1} < m_i$ for $i \geq 2$. Hence the sequence must eventually reach 0. If $m_{k+1} = 0$, then it is easy to show that m_k is the GCD.

  E.g., for $(m, n) = (15, 9)$, we get the sequence $(15, 9, 6, 3, 0)$ and so $k = 3$ and $m_3 = 3 = \texttt{GCD}(15, 9)$.
- OVERVIEW. Your main program is called `gcd.c`. The input to `gcd.c` is a set of pairs of numbers. If there are $k \geq 1$ pairs, then the main process will spawn k children processes. Each child process will compute the GCD of one pair of numbers. However, the child process does not know how to compute the modulo operation. Only the parent process knows how. Hence the child must submit pairs (a, b) of integers to the parent who will compute and return the modulus $a \mod b$. When a child process has computed the GCD of its pair, it exits. When all the children has exited, the parent exit.

  For instance, if we type

  ```
  > gcc -o GCD GCD.c
  > GCD 24 17 18 10 987654321 123456
  ```

  then GCD will spawn 3 processes which eventually prints ”1”, ”2” and ”3” (as the GCD’s of the three pairs).

  We want the parent process to implement the $a \mod b$ operation by repeated subtractions. Moreover, the parent will use round robin to service the children.
- COMMUNICATION. The parent and each child communicates through a two-way **pipe**. When a child has placed a pair (a, b) in the pipe, it **signals** the parent and then attempts to read the pipe. Of course, it will block until the parent has written the answer back to the pipe.

  The parent responds to the signal by searching through the pipes from each of the children to find a pair (a, b) to work on. In order to avoid busy waiting, the parent will only do this search in response to a signal. There is another trick: if the parent tries to read a pipe that has nothing to be read, it will block. To avoid this, you must use NON-BLOCKING reads for this loop.

- Finally, you are to create several runs using various sets of input pairs, and give the timing for these runs.
- Thus, you must know how to:
  - set up pipes
  - how to read/write from and to pipes
  - how to do non-blocking reads
  - to use the signaling mechanism,
  - time the running time of your program.

  We will give you all the hints necessary to do the programming part.
- WHAT TO HAND IN: a single tar file containing a Makefile file, README file, and all necessary programs. You must give your timings and explain your experiments in the README file.

  I should be able to duplicate your experiments by typing ”make time”.