CS69003: Computing Systems Lab I
Autumn 2008

Lab Test 2

Time: 2:00pm – 4:00pm, October 31, 2008 (Friday)

In this test, you are asked to print all positive divisors d of a positive integer n with $1 \leq d \leq b$, where b is a bound supplied by the user. Multiple processes are used in order to search for potential divisors. These processes write to a shared array and synchronize by using a semaphore.

Part I (75)

Your program first reads two positive integers from the user—the integer n whose factors are desired and the bound b. The process then creates a shared memory segment capable of storing an array of b integers. In addition, a shared integer count (initialized to 0) stores the actual number of divisors in the divisor array.

The program then forks two child processes C_1 and C_2. The process C_1 searches for potential divisors d_1 of n in the range $1 \leq d_1 \leq b/2$. The other process C_2 handles the interval $b/2 < d_2 \leq b$ in search of divisors d_2 of n. The processes C_1 and C_2 search for the divisors in parallel.

When a child process discovers a divisor of n, it appends the new divisor to the shared array and also increases the count of the elements in the array. After both C_1 and C_2 terminate, the parent process sorts the list of divisors in the shared memory and prints the sorted array.

As an example, let $n = 60$ and $b = 15$. Then, C_1 searches for divisors in the range $1 \leq d_1 \leq 7$, and C_2 searches for divisors in the range $8 \leq d_2 \leq 15$. C_1 discovers the divisors $d_1 = 1, 2, 3, 4, 5, 6$, whereas C_2 discovers the divisors $d_2 = 10, 12, 15$. The parent prints the sorted list $1, 2, 3, 4, 5, 6, 10, 12, 15$. Note that since C_1 and C_2 run in parallel, the divisors found by C_1 may be mixed with the divisors found by C_2 in the shared memory. For example, one possible order of insertions in the shared array can be $1, 2, 10, 3, 4, 5, 12, 6, 15$.

Part II (25)

Write a second program that, in addition to n and b, reads a third positive integer k from the user. The process then forks k child processes. Each child process handles a range of size $\approx b/k$ for potential divisors. Also, the ranges of different processes are disjoint. The child processes should run in parallel. After all child processes terminate, the parent process sorts and prints the list of divisors.

Submit two C programs `part1.c` and `part2.c`. Write your name and roll number (as comment) at the beginning of your C programs. Send the files by e-mail with the subject specifying your name and roll number.