

CS 385: Prog 2–Due various

Jon A. Solworth

November 4, 2003

Due 7 November These are from the test

- Write two programs using only semaphores for synchronization.
The first write A , synchronizes to ensure the second process has written B and then writes C .
The second synchronizes until the first process writes A and then writes B it then synchronizes until the first process writes C and then writes D
- The following is done using shared memory and semaphores. The procedures allocate resources in shared memory. Resource 1 (resp. Resource 2) is represented N_1 (N_2) bytes in shared memory. The corresponding byte is 1 if that unit of the resource is allocated and 0 otherwise.

You are to write a class call ResourceAllocator with two methods:

- allocate(int n1, int n2): allocated the resources and keep track of which resources were allocated.
- release(): releases the resources which provided by the last allocator.

Your solution should not deadlock.

Due 10 November

- part 1. Implement a parallel histogram counter in sharred memory. The histogram has N counters, each guarded by a semaphore. The operation is increment the counter.
- part 2. Associate with each histogram counter is an array of M elements. Associate a semaphore with each array. Implement each array as a stack.

Due 14 November You are to write a program which will implement a form of two phased locking.

You should write this program in C++. To support 2-phase locking you will need to create a class, twoPhaseLocking with member functions:

- `int read2pl(int i)`: lock and read the *i*th data element in shared memory.
- `void write2pl(int i, int v)`: write lock and write the *i*th data element in shared memory with the value *v*.
- `void commit2pl()`: release all the locks held by the process in shared memory.

To prevent deadlock, you can ensure (and you should check) that the location are read/written in increasing order of index.

You should also support upgrade of locks.

To do this assignment, you will need to allocate and attach to shared memory. Moreover, you will need to use semaphores for synchronization.