

CS 385: HW3–Due 31 Oct

Jon A. Solworth

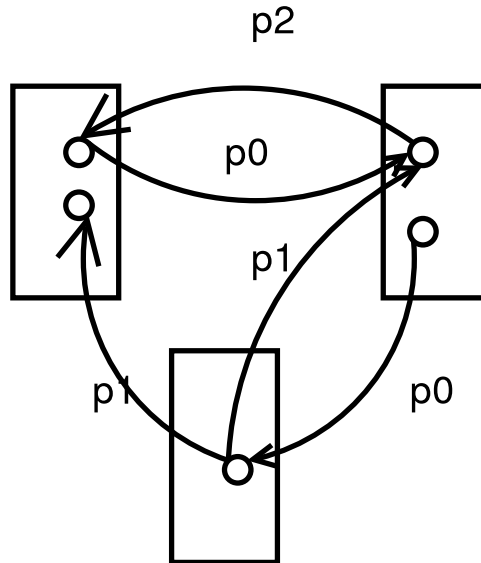
October 26, 2003

1. In 2phase commit, if the deadlock occurs the transaction must abort. This abort is part of the atomic operation and is performed by waiting a fixed amount of time for a lock (say 1 second) and if it is not obtained, all locks are released all locations updated by the transaction are restored, and the program branches back to the start of the transaction.

Implement both get read lock and get write lock code implementing abort.

2. Can the following rules lead to deadlock? If so, give an example or if not explain why not.
 - (a) Each process runs at a different (fixed) priority. If a higher priority process would block on a resource held by a lower priority process, it kills the low priority process.
 - (b) Resources are numbered by integers. All odd numbered resources are requested before any even even ones.
 - (c) Resources are requested in any order, but it is possible to time-share the resource by 1) saving the state associated with one processes use of the resource and 2) restoring the state associated with another process.
 - (d) Resources are numbered 0 through 5. A process requests either resource 0 or 5, then 3 or 4, and then may request 1 and then may request 2.
 - (e) If all the resources requested are available they are allocated otherwise the requesting process releases all the resources it holds.
3. Consider a system with 3 resources A , B , and C with respective quantities N_A , N_B , N_C . Write a set of resource allocation/release routines (with error checking) for each of Dijkstra's 4 conditions which avoids deadlock by denying that condition (that is, you should have 4 sets of routines).

4. In the following wait-for graph, is the system in deadlock?



5. Given the following banker's algorithm configuration, answer the below questions using a banker's algorithm simulation,:

- Is the below configuration safe?
- Is the allocation request of 1 unit of R_0 to p_1 safe?

Variable	R_0	R_1	R_2
total resources	3	4	2
allocated/need p_0	0/3	2/2	1/0
allocated/need p_1	1/2	1/0	0/1
allocated/need p_2	0/1	1/2	0/1

6. In the following configuration, use the banker's algorithm to determine whether the system is in deadlock.

Variable	R_0	R_1	R_2
total resources	3	2	1
allocated to p_0	1	0	1
allocated to p_1	1	1	0
allocated to p_2	0	1	0

Assuming that p_1 is waiting for 1 unit of R_2 and p_2 is waiting for 2 units of R_0 .

7. Consider the following cases, and describe whether starvation is possible and if so how and if not, why not.

- Processes have a fixed priority, and higher priority processes are run before lower priority processes.
- Processes have an assigned priority and a current priority initialized to the assigned priority. Each time a process runs for $1/10$ th of a second continuously, its current priority is cut in half. When a process has not run for 5 seconds its current priority is set to its assigned priority. Processes are scheduled every $1/10$ th of a second in order of their current priority.