CMPSCI 377 Operating Systems **Fall 2008**

Lecture 13: October 16

Lecturer: Prashant Shenoy *Scribe: Shashi Singh*

13.1 Banker's Algorithm

The Banker's algorithm is a resource allocation and deadlock avoidance algorithm developed by Edsger Dijkstra that tests for safety by simulating the allocation of pre-determined maximum possible amounts of all resources, and then makes a "safe-state" check to test for possible deadlock conditions for all other pending activities, before deciding whether allocation should be allowed to continue. The Banker's algorithm is run by the operating system whenever a process requests resources. The algorithm prevents deadlock by denying or postponing the request if it determines that accepting the request could put the system in an unsafe state (one where deadlock could occur).

13.1.1 Resources

For the Banker's algorithm to work, it needs to know three things:

1. How much of each resource each process could possibly request
2. How much of each resource each process is currently holding
3. How much of each resource the system has available

Some of the resources that are tracked in real systems are memory, semaphores and interface access.

13.1.2 Safe and Unsafe States

A state is considered safe if it is possible for all processes to finish executing (terminate). Since the system cannot know when a process will terminate, or how many resources it will have requested by then, the system assumes that all processes will eventually attempt to acquire their stated maximum resources and terminate soon afterward. This is a reasonable assumption in most cases since the system is not particularly concerned with how long each process runs (at least not from a deadlock avoidance perspective). Also, if a process terminates without acquiring its maximum resources, it only makes it easier on the system. Given that assumption, the algorithm determines if a state is safe by trying to find a hypothetical set of requests by the processes that would allow each to acquire (one-by-one) its maximum resources and then terminate (returning its resources to the system). Any state where no such set exists is an unsafe state. In a safe state, at least one process should be able to acquire its maximum possible set of resources, and proceed to termination.

13.1.3 The simplified algorithm

When the system receives a request for resources, it runs the Banker's algorithm to determine if it is safe to grant the request. The algorithm is fairly straight forward once the distinction between safe and unsafe states is understood.

1. Can the request be granted?
   - If not, the request is impossible and must either be denied or put on a waiting list
2. Assume that the request is granted
3. Is the new state safe?
   - If so grant the request
   - If not, either deny the request or put it on a waiting list

Whether the system denies or postpones an impossible or unsafe request is a decision specific to the operating system.

Note: Look at lecture slide for the *pseudocode* and *examples*.

13.1.4 Trade-offs

Like most algorithms, the Banker's algorithm involves some trade-offs. Specifically, it needs to know how much of each resource a process could possibly request. In most systems, this information is unavailable, making the Banker's algorithm useless. Besides, it is unrealistic to assume that the number of processes is static. In most systems the number of processes varies dynamically. Moreover, the requirement that a process will eventually release all its resources (when the process terminates) is sufficient for the correctness of the algorithm, however it is not sufficient for a practical system. Waiting for hours (or even days) for resources to be released is usually not acceptable.