MANCHESTER 1824

The University of Manchester

A Few Words About Deadlock

Deadlock

- Deadlock occurs when four particular conditions hold simultaneously in a system:
  1. Mutual exclusion: at least one resource must be non-shareable, i.e. only one process can use that resource at a given time.
  2. Hold and wait: a process is currently holding at least one resource, and is requesting at least one more resource that is currently being held by another process (i.e., it is waiting for a process to release something).
  3. No pre-emption: once a process has acquired a resource, nothing (e.g., the operating system or some external factor) can force it to relinquish that resource; it has to do so voluntarily.
  4. Circular wait: a process must be waiting for a resource that is being held by another process, which in turn and possibly indirectly, is waiting for the first process to release a resource.
     - Put another way, processes are waiting for one another in such a way that there is a cycle of dependencies between them. For example, if A waits for B, B waits for C, and C waits for A, we have a cycle of waiting that would fulfil this condition.

An Example

- Consider a situation where two processes are running at the same time; one of them is using a printer, and the other, a scanner. It comes a point in time when:
  - Both processes can no longer continue to do their work until they can get access to the device that is being used by the other process.
  - The processes will not relinquish control of the device they are currently using until they get it.
- The result is: now both processes are in a deadlock state—neither can continue because they are waiting for the other.

Dealing with a Deadlock

- There are three potentially sensible strategies for dealing with a deadlock:
  - Prevent deadlock from occurring in the first place by making sure that it is never possible for all four of the conditions to be true at the same time (you might like to think about what this means in each of the four cases).
  - Avoid deadlock by making sure that, even though it is potentially possible for all four conditions to hold at the same time, that they never do.
  - Detect a deadlock and deal with the consequences (this last approach is very similar to 'avoid' really, in that it requires having some mechanism for detecting whether all four conditions are about to hold (for avoid) or actually do hold (for detect)).
    - Whether you decide to go for a detect or avoid approach really depends on how likely or frequently a deadlock is likely to happen—if some property of your system means that deadlock is possible, but hugely unlikely, it might make sense to allow it to occur once in a while and to then deal with the consequences, rather than to incur the overhead of trying to avoid it from happening in the first place.

Supplement Material – Classical Computer Science Problems

- The Dining Philosophers Problem:

https://www.youtube.com/watch?v=NbwbQQB7xNQ

- The Two Generals' Problem

https://www.youtube.com/watch?v=IP-rGJKSZ3s