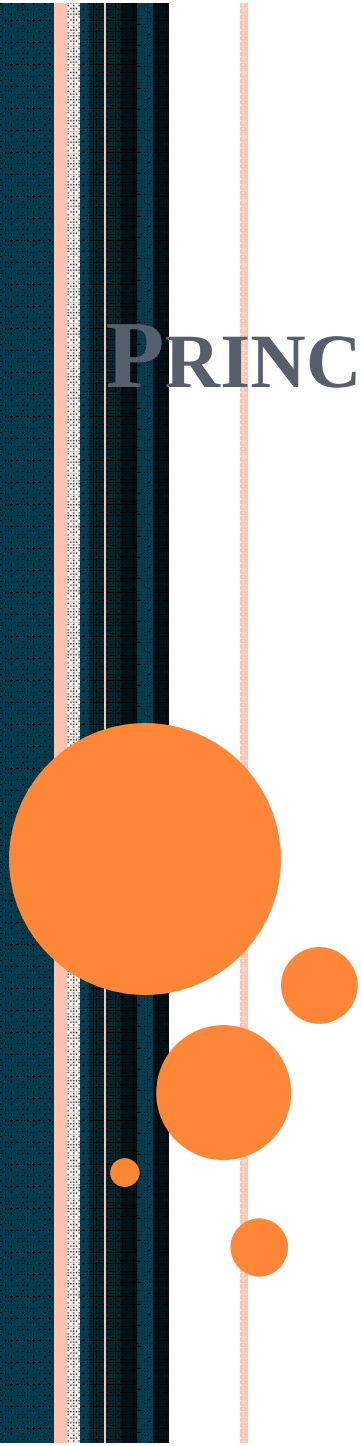
PRINCIPLES OF OPERATING SYSTEMS



LECTURE 25
Deadlock Prevention

The Deadlock problem

- In a computer system deadlocks arise when members of a group of processes which hold resources are blocked indefinitely from access to resources held by other processes within the group.

Deadlock example

- P_i requests one I/O controller and the system allocates one.
- P_j requests one I/O controller and again the system allocates one.
- P_i wants another I/O controller but has to wait since the system ran out of I/O controllers.
- P_j wants another I/O controller and waits.

Strategies for handling deadlocks

- Deadlock prevention. Prevents deadlocks by restraining requests made to ensure that at least one of the four deadlock conditions cannot occur.
- Deadlock avoidance. Dynamically grants a resource to a process if the resulting state is safe. A state is safe if there is at least one execution sequence that allows all processes to run to completion.
- Deadlock detection and recovery. Allows deadlocks to form; then finds and breaks them.

Deadlock Prevention

- 1. A process acquires all the needed resources simultaneously before it begins its execution, therefore breaking the hold and wait condition.
- E.g. In the dining philosophers' problem, each philosopher is required to pick up both forks at the same time. If he fails, he has to release the fork(s) (if any) he has acquired.
- Drawback: over-cautious.

Cont..

- 2. All resources are assigned unique numbers. A process may request a resource with a unique number I only if it is not holding a resource with a number less than or equal to I and therefore breaking the circular wait condition.
- E.g. In the dining philosophers problem, each philosopher is required to pick a fork that has a larger id than the one he currently holds. That is, philosopher P5 needs to pick up fork F5 and then F1; the other philosopher Pi should pick up fork Fi followed by Fi-1.
- Drawback: over-cautions.

Cont…

- 3. Each process is assigned a unique priority number. The priority numbers decide whether process Pi should wait for process Pj and therefore break the non-preemption condition.
- E.g. Assume that the philosophers' priorities are based on their ids, i.e., Pi has a higher priority than Pj if i <j. In this case Pi is allowed to wait for Pi+1 for I=1,2,3,4. P5 is not allowed to wait for P1. If this case happens, P5 has to abort by releasing its acquired fork(s) (if any).
- Drawback: starvation. The lower priority one may always be rolled back. Solution is to raise the priority every time it is victimized.

Cont..

- 4. Practically it is impossible to provide a method to break the mutual exclusion condition since most resources are intrinsically non-sharable, e.g., two philosophers cannot use the same fork at the same time.