# PRINCIPLES OF OPERATING SYSTEMS

# LECTURE – 29
# Deadlock Avoidance

# *Introduction*

- Possible side effects of preventing deadlock are low device utilization and reduce system throughput

- An alternative method for avoiding deadlocks is to require additional information about how resources are to be required.
- With this knowledge of the complete sequence of requests and releases for each process the system can decide for each request whether or not the process should wait in order to avoid possible future deadlock.
- A deadlock avoidance algorithm dynamically examines the resource-allocation state to ensure that a **circular wait** condition can never happen

# Safe State

- A state is **safe** if the system can allocate resources to each processes in some order **(safe sequence)** and still avoid a deadlock.
- If no such sequence exists, then the system state is said to be **unsafe.**
- If we have prior knowledge of how resources will be requested, it's possible to determine if we are entering an "unsafe" state.

# Cont...

Possible states are:

**Deadlock** No forward progress can be made.

**Unsafe state** A state that **may** allow deadlock.

**Safe state** A state is safe if a sequence of processes exist such that there are enough resources for the first to finish, and as each finishes and releases its resources there are enough for the next to finish.

**The rule is simple:** If a request allocation would cause an unsafe state, do not honor that request.

# Safe State Example

- Let's assume a very simple model: each process declares its maximum needs. In this case, algorithms exist that will ensure that no unsafe state is reached. *Maximum needs* does NOT mean it *must* use that many resources – simply that it *might* do so under some circumstances.
- There exists a total of 12 resources. Each resource is used exclusively by a process. The current state looks like this:

| | Max needs | Current needs |
|---|---|---|
| P0 | 10 | 5 |
| P1 | 4 | 2 |
| P2 | 9 | 2 |

At T0, the system is in safe state, since <P1,P0,P2> satisfied safe state condition

What if P2 currently ask for one more tape and has that one?

# Safe, Unsafe, Deadlock

***Note: All deadlocks are unsafe, but all unsafes are NOT deadlocks.***