Combined Approach to Deadlock Handling

**Batch:** B.Tech III year

**Instructor:** Rahul Muthu

DA-IICT
None of the three methods of dealing with the deadlock problem (prevention, avoidance or detection & recovery) is universally the best.
None of the three methods of dealing with the deadlock problem (prevention, avoidance or detection & recovery) is universally the best.

Systems can benefit by using a combination of these schemes for different classes of resources.
None of the three methods of dealing with the deadlock problem (prevention, avoidance or detection & recovery) is universally the best.

Systems can benefit by using a combination of these schemes for different classes of resources.

The resources are partitioned into a hierarchical order and each level uses a scheme best suited to it.
**Internal resources:** These are resources used by the system, like PCBs.
An example system

- **Internal resources:** These are resources used by the system, like PCBs.
- **Central memory:** This is the main memory space used by user programs.
An example system

- **Internal resources:** These are resources used by the system, like PCBs.
- **Central memory:** This is the main memory space used by user programs.
- **Job resources:** Assignable devices (tape drives, printers) and files (in edit mode).
An example system

- **Internal resources:** These are resources used by the system, like PCBs.
- **Central memory:** This is the main memory space used by user programs.
- **Job resources:** Assignable devices (tape drives, printers) and files (in edit mode).
- **Swappable space:** Space used for each user job on the backing store.
A combined solution for the example

- For the internal resources *prevention* through *resource ordering* can be used, since run-time choices between pending requests are unnecessary.
A combined solution for the example

- For the internal resources *prevention* through *resource ordering* can be used, since run-time choices between pending requests are unnecessary.

- For the central memory problem, *prevention* through *preemption* can be used since a job can be swapped out from main memory and reloaded later.
A combined solution for the example

- For the internal resources *prevention* through *resource ordering* can be used, since run-time choices between pending requests are unnecessary.

- For the central memory problem, *prevention* through *preemption* can be used since a job can be swapped out from main memory and reloaded later.

- For job resources an *avoidance scheme* can be used, since the maximum resource requirement of this type can be easily determined and provided to the system.
For the internal resources *prevention* through *resource ordering* can be used, since run-time choices between pending requests are unnecessary.

For the central memory problem, *prevention* through *preemption* can be used since a job can be swapped out from main memory and reloaded later.

For job resources an *avoidance scheme* can be used, since the maximum resource requirement of this type can be easily determined and provided to the system.

Swappable space can be handled by preallocation, since the maximum storage requirements are usually known. This corresponds to *prevention* by no *hold-and-wait*. 