CSC 224 Principles of OS

5. -Inter-process Communication

-Deadlocks

Inter-process Communication

- Since processes frequently need to communicate with other processes therefore, there is a need for a well-structured communication without using interrupts, among processes.
- This can be affected by:
  - Race Condition
  - Mutual exclusion

Race condition

- In operating systems, processes that are working together share some common storage (main memory, file etc.) that each process can read and write.
- When two or more processes are reading or writing some shared data and the final result depends on who runs precisely when, they are called race conditions.

- Concurrently executing threads that share data need to synchronize their operations and processing in order to avoid race condition on shared data. Only one ‘customer’ thread at a time should be allowed to examine and update the shared variable.
- Concurrent processes come into conflict with each other when they are competing for the use of the same resource.

- Here the important point is that when one process is executing shared modifiable data in its critical section, *(part of prog where shared memory is accessed)* no other process is to be allowed to execute in its critical section.
- Thus, the execution of critical sections by the processes is mutually exclusive in time.

Mutual Exclusion

- Is a way of making sure that if one process is using a shared modifiable data, the other processes will be excluded from doing the same thing.
- While one process executes the shared variable, all other processes desiring to do so at the same time moment should be kept waiting; when that process has finished executing the shared variable, one of the processes waiting to do so should be allowed to proceed.

- This way, each process executing the shared data (variables) excludes all others from doing so simultaneously.
- This is called Mutual Exclusion.
- Note that mutual exclusion needs to be enforced only when processes access shared modifiable data - when processes are performing operations that do not conflict with one another they should be allowed to proceed concurrently.

Requirements necessary for Mutual Exclusion

- 1. Mutual exclusion must be enforced: Only one process at a time is allowed into its critical section, among all processes that have critical sections for the same resource or shared object.
- 2. A process that halts in its non critical section must do so without interfering with other processes.

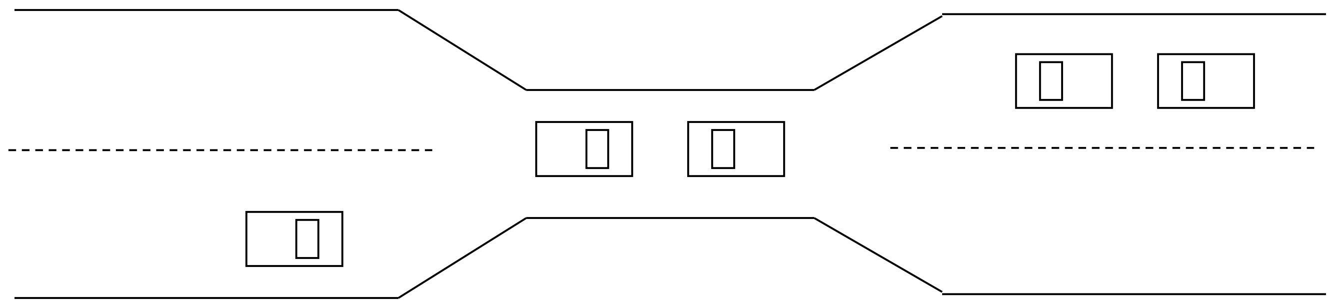
- 3. It must not be possible for a process requiring access to a critical section to be delayed indefinitely: no deadlock or starvation.
- 4. When no process is in a critical section, any process that requests entry to its critical section must be permitted to enter without delay.

- 5. No assumptions are made about relative process speeds or number of processors.
- 6. A process remains inside its critical section for a finite time only.

Deadlocks

- A set of blocked processes each holding a resource and waiting to acquire a resource held by another process in the set.
- Example
  - A System has 2 tape drives.
  - P_1 and P_2 each hold one tape drive and each needs another one.

Bridge Crossing Example



- Traffic only in one direction.
- Each section of a bridge can be viewed as a resource.
- If a deadlock occurs, it can be resolved if one car backs up (preempt resources and rollback).
- Several cars may have to be backed up if a deadlock occurs.
- Starvation is possible. *(Same process killed repeatedly*

Example

- process 1 has been allocated non-shareable resources *A, say, a tape drive, and process 2 has be allocated non-sharable resource B, say, a printer.*
- *Now, if* it turns out that process 1 needs resource *B (printer) to proceed and process 2 needs resource A* (the tape drive) to proceed and these are the only two processes in the system, each has blocked the other and all useful work in the system stops.
- This situation is termed deadlock

- The system is in deadlock state because each process holds a resource being requested by the other process neither process is willing to release the resource it holds.

Resource types

- Resources come in two flavors: pre-emptable and non-pre-emptable.
- A pre-emptable resource is one that can be taken away from the process with no ill effects.
- Memory is an example of a pre-emptable resource.

- On the other hand, a non-pre-emptable resource is one that cannot be taken away from process (without causing ill effect).
- For example, *CD resources are not pre emptable* at an arbitrary moment.
- Reallocating resources can resolve deadlocks that involve preem-ptable resources.
- Deadlocks that involve non pre-emptable resources are difficult to deal with.

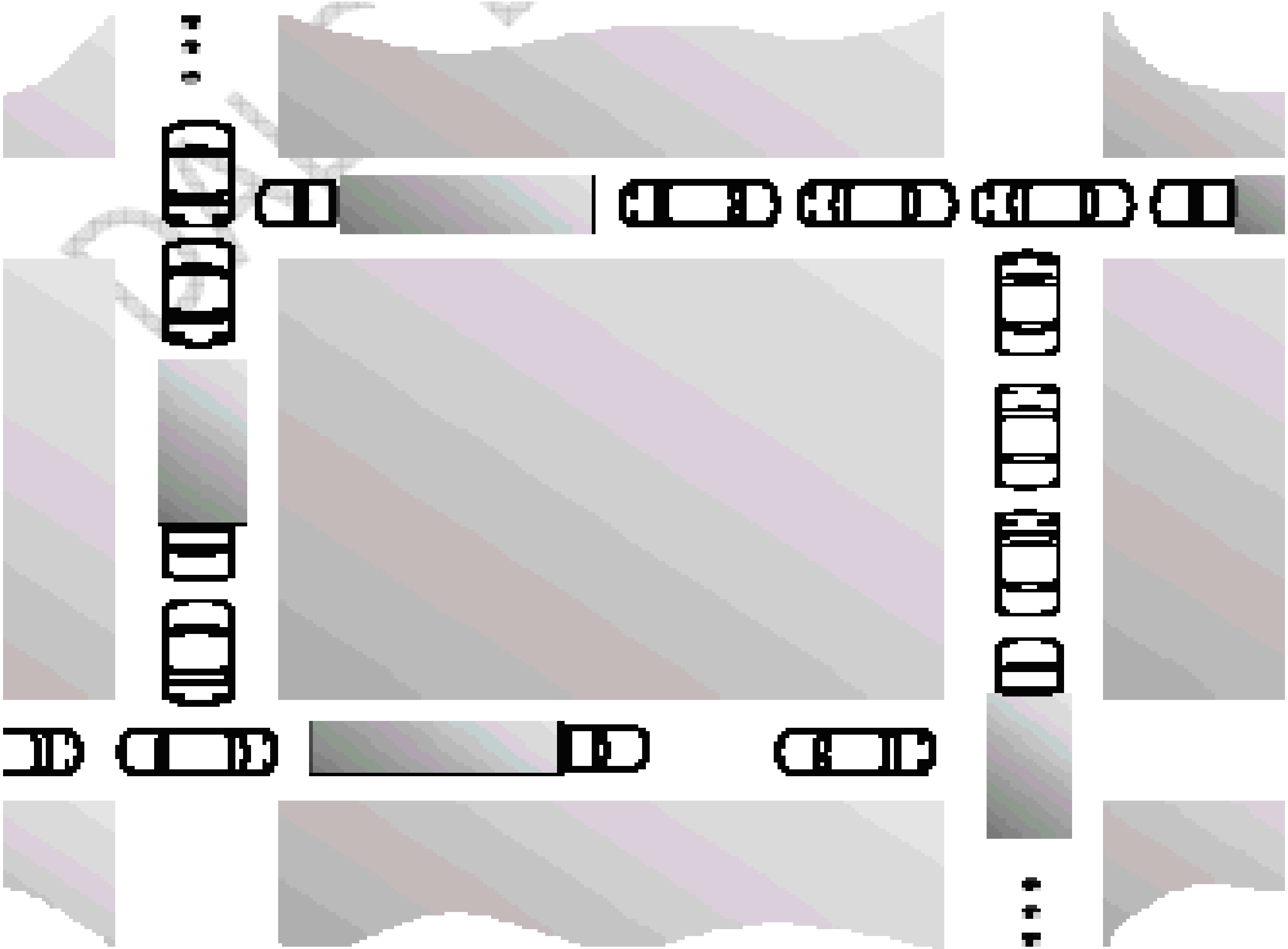
Deadlock Characterization

Deadlock can arise if four conditions hold simultaneously.

- **Mutual exclusion:** only one process at a time can use a resource.
- **Hold and wait:** a process holding at least one resource is waiting to acquire additional resources held by other processes.

- **No preemption:** a resource can be released only voluntarily by the process holding it, after that process has completed its task.
- **Circular wait:** there exists a set $\{P_0, P_1, \ldots, P_0\}$ of waiting processes such that P_0 is waiting for a resource that is held by P_1, P_1 is waiting for a resource that is held by P_2, …, P_{n-1} is waiting for a resource that is held by P_n, and P_0 is waiting for a resource that is held by P_0.

- The processes in the system form a circular list or chain where each process in the list is
- waiting for a resource held by the next process in the list.



- 1. Mutual exclusion condition applies, since only one vehicle can be on a section of the street at a time.
- 2. Hold-and-wait condition applies, since each vehicle is occupying a section of the street, and waiting to move on to the next section of the street.

- 3. No-preemptive condition applies, since a section of the street that is a section of the street that is occupied by a vehicle cannot be taken away from it.
- 4. Circular wait condition applies, since each vehicle is waiting on the next vehicle to move. That is, each vehicle in the traffic is waiting for a section of street held by the next vehicle in the traffic.

- It is not possible to have a deadlock involving only one single process. The deadlock involves a circular “hold-and-wait” condition between two or more processes, so “one” process cannot hold a resource, yet be waiting for another resource that it is holding.

- In addition, deadlock is not possible between two threads in a process, because it is the process that holds resources, not the thread that is, each thread has access to the resources held by the process.

Deadlock detection

- Deadlock detection is the process of actually determining that a deadlock exists and identifying the processes and resources involved in the deadlock.

- The basic idea is to check allocation against resource availability for all possible allocation sequences to determine if the system is in deadlocked state
- Once a deadlock is detected, there needs to be a way to recover.

Several alternatives exists:

- Temporarily prevent resources from deadlocked processes.
- Back off a process to some check point allowing preemption of a needed resource and restarting the process at the checkpoint later.
- Successively kill processes until the system is deadlock free.

- Abort one process at a time until the deadlock cycle is eliminated.
- In which order should we choose to abort?
  - Priority of the process.
  - How long process has computed, and how much longer to completion.
  - Resources the process has used.
  - Resources process needs to complete.
  - How many processes will need to be terminated.
  - Is process interactive or batch?

Revision questions

- Detail three conditions that are necessary for mutual exclusion
- How can deadlocks be avoided?
- For a deadlock to occur, a circular wait condition is necessary. Express this diagrammatically using a resource allocation graph.