Chapter 3
Input/Output

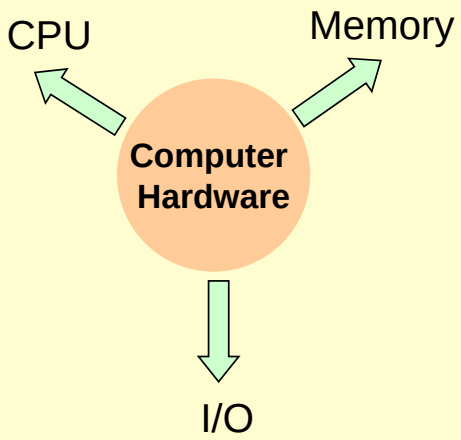
OPERATING SYSTEMS

Design and Implementation

Instructor:
Hadi Salimi
Computer Engineering Department
IRAN University of Science and Technology
hsalimi@iust.ac.ir

Computer Hardware

- The computer hardware has been composed of three main components: CPU, I/O and memory.



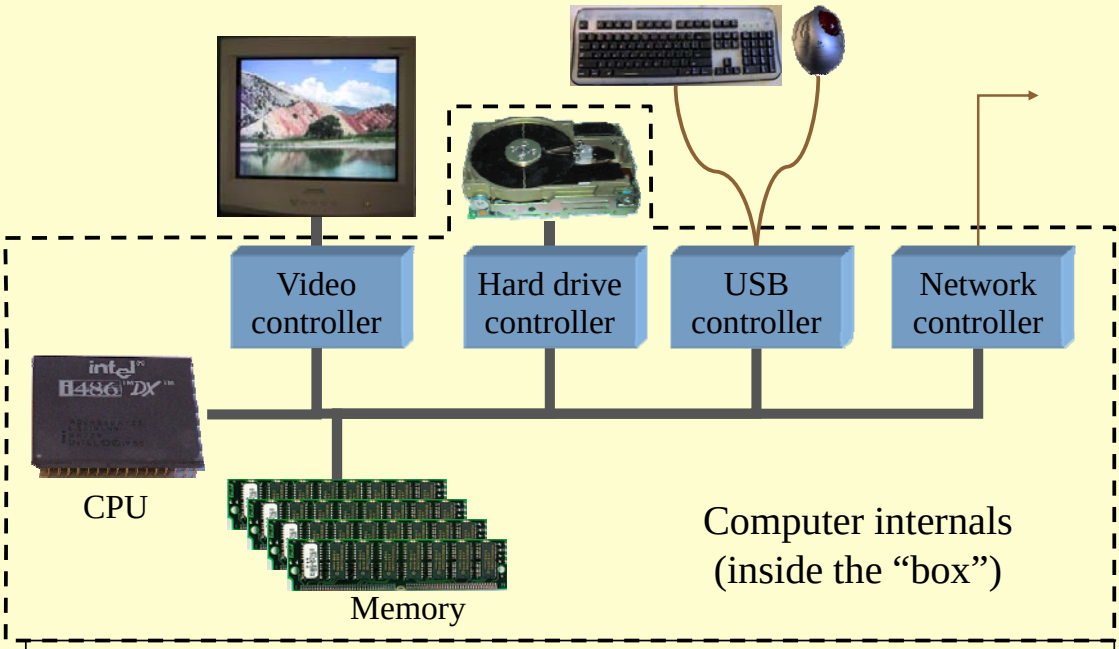
I/O Devices

- From one point of view, I/O devices can be divided into two categories:
  - **Block devices**: A device in which reads and writes data in fixed size blocks, like disks.
  - **Character devices:** A device in which delivers or accepts a stream of characters, regardless of any block structure, like printers or networks.

Device Controllers

- I/O units typically consist of a mechanical component and an electronic component.
- The electronic component is called device controller or adapter.
- Many controllers can handle two or more devices.
- Using the device controller hides all of the complexities of a device.

A Popular Architecture



Device Controllers

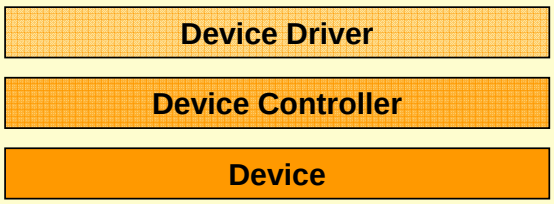
- The interface between the controller and the device is often very low level.
- As an example consider a disk controller
  - The controller should convert the serial bit stream into a block.
  - Consider the preamble and other additional data.
  - Error check the read data.

Device Controllers

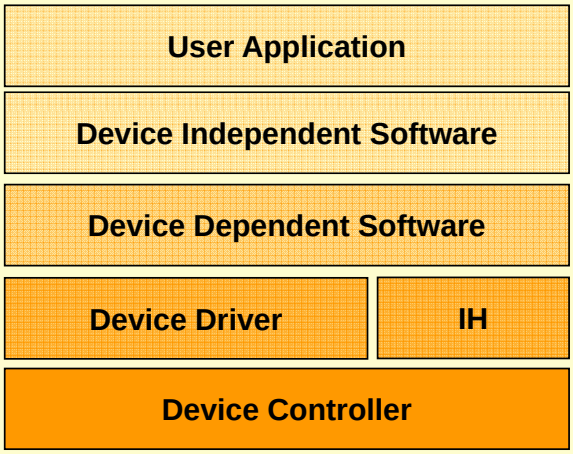
- Each controller has a few registers that are used for communicating with the CPU.
- On some computers these registers are part of the regular memory address space.
- This scheme is called memory-mapped I/O.

Device Driver

- The job of a device driver is to accept abstract requests from a high-level software and talk to the device controller.



Software Layers

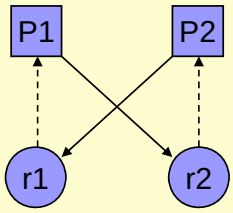


Deadlocks

- Some resources in a computer system can be shared, but others cannot.
- As an example many processes can read from a disk simultaneously.
- As another example, a printer can just be assigned to one process.

Deadlocks (cont.)

- Suppose that two processes have gained access to two resources.
- If they request for the other resource and the resources could not be shared, then the deadlock occurs.
- This situation can also be occurred in software. (DB)



Resources

- A resource is anything that can be used by a process, like disks, printers, files and so on.
- Resources can be preemptable or non-preemptable.
- Memory is an example of a preemptable resource, but printer is a non-preemptable one.

Resources

- The sequence of events required to use a resource is given below in an abstract form.
  - Request the resource.
  - Use the resource.
  - Release the resource.

Deadlock

- Deadlock can be defined formally as follows:
  - A set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause.
- For this model, we assume that processes have only a single thread and that there are no interrupts possible to wake up a blocked process.

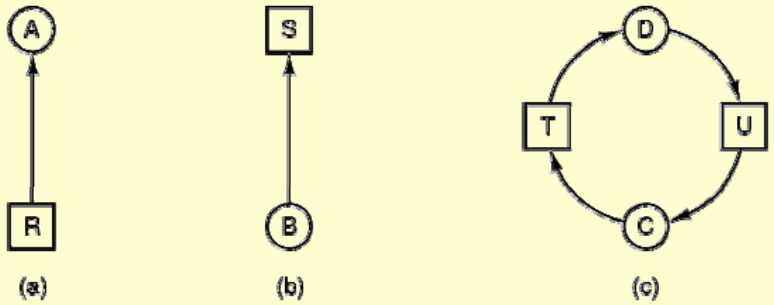
Deadlock Conditions

- Mutual exclusion condition. Each resource is either currently assigned to exactly one process or is available.
- Hold and wait condition. Processes currently holding resources that were granted earlier can request new resources.
- No preemption condition. Resources previously granted cannot be forcibly taken away from a process. They must be explicitly released by the process holding them.
- Circular wait condition. There must be a circular chain of two or more processes, each of which is waiting for a resource held by the next member of the chain.

Deadlock Conditions

- All four of these conditions must be present for a deadlock.
- If one or more of these conditions is absent, no deadlock is possible.

Deadlock Modeling

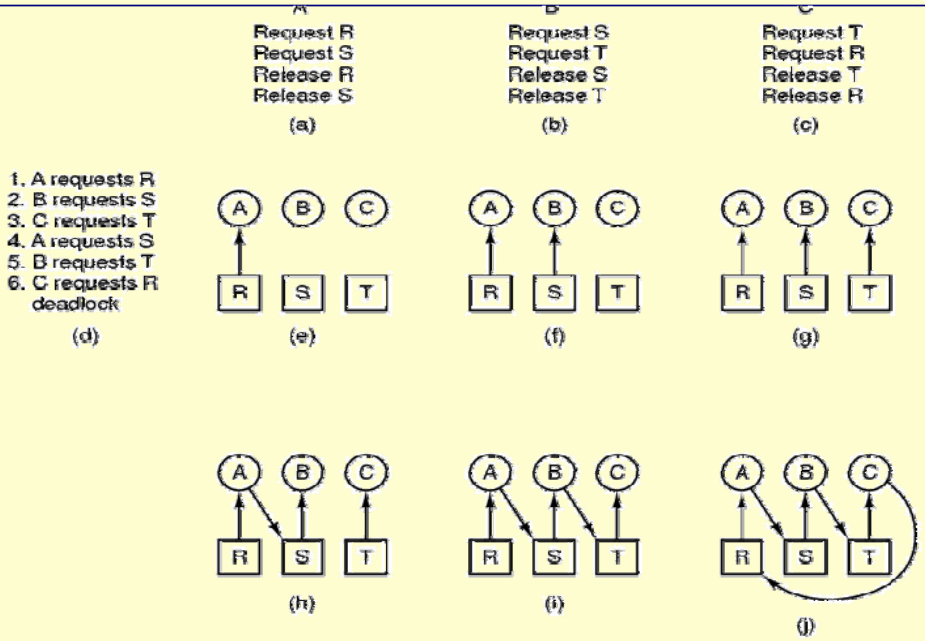


- Resource R assigned to process A
- Process B is requesting/waiting for resource S
- Process C and D are in deadlock over resources T and U

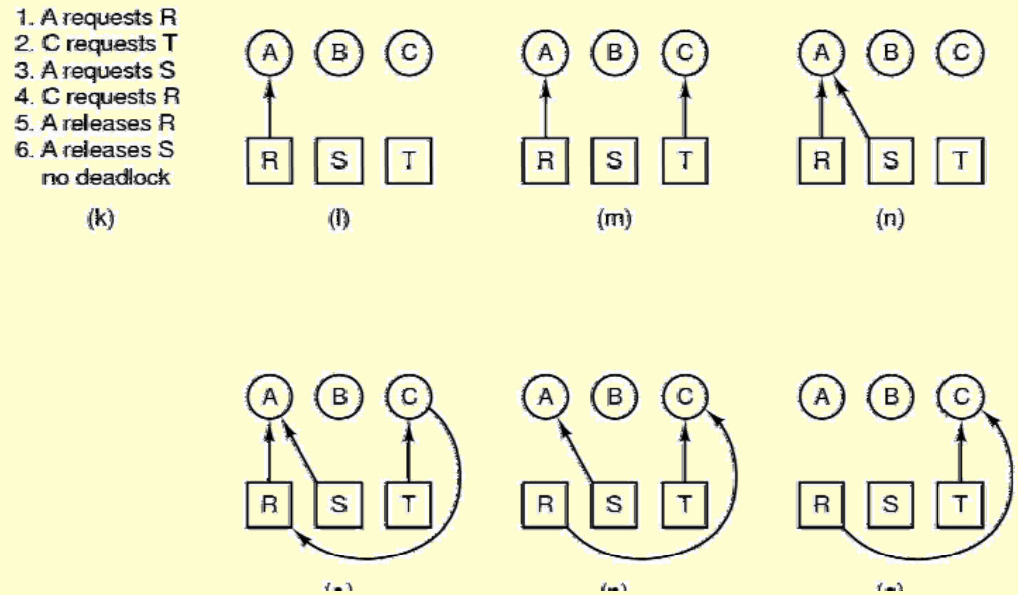
Some Questions

- Is it possible for a batch system to go to any deadlocks?
  - What if all schedulers act like this?
  - What if non of the processes do any I/O at all?

Deadlock Modeling



Deadlock Modeling



Strategies

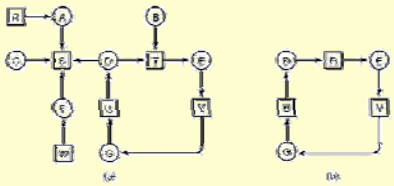
- Just Ignore the problem.
- Detection and recovery
- Prevention, by negating one of the four required conditions.
- Dynamic avoidance by careful resource allocation.

Ostrich Algorithm

- In some point of view, the best solution is to pretend that there is no problem at all.
- How often deadlock occurs? Does the frequency is greater than the frequency of system crash?
- Whenever a deadlock occurs, just reboot the system.
- Linux and MINIX act like this.

Detection and Recovery

- The system monitors the request and the release of the resources and updates its graph.
- In the case of any release or request, the graph is checked to see is there any cycle or not. If yes, kill a process in the cycle.
- Any important point about killing a process?



Detection and Recovery

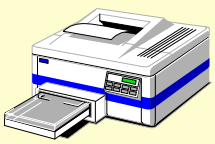
- A somewhat cruder method is not even maintain the resource graph.
- But the operating system should periodically check to see if there are any processes that have been continuously blocked for more than say, 1 hour.
- Such processes are killed.

Deadlock Prevention

- Put suitable restrictions on processes so deadlocks are impossible.
- Four deadlock conditions stated by Coffman, provide a clue for some solutions.
- How to attack these 4 conditions?

Attack to Mutual Exclusion

- If no resource was assigned exclusively to a single process, we would never have deadlocks.
- But how a resource like a printer can be used non-exclusively?
- Answer:
  - By spooling printer output, several processes can generate output at the same time.
  - Only the printer daemon is using the printer.



Attack to Hold and Wait

- One way to achieve this goal is to require all processes to request all their resources before starting execution.
- Defects:
  - Many processes do not know which resources they need.
  - Resources will not be used optimally.

Attack to no preemption

- Attacking to this condition is not easy.
- If a process has been assigned the printer and is in the middle of printing, taking away the printer because of a needed plotter is not acceptable.

Attack to circular wait

- A process is allowed to just have one resource at a time.
  - Is it reasonable?
- Another way is to provide a global numbering for resources and all requests must be made in numerical order.

All Approaches

Condition	Approach
Mutual Exclusion	Spool Everything
Hold and Wait	Request all resources initially
No Preemption	Take resources away
Circular Wait	Order resources numerically

Summary of all approaches to deadlock prevention.

Deadlock Avoidance

- Banker's Algorithm

	Has	Max
A	0	6
B	0	5
C	0	4
D	0	7

Free: 10

	Has	Max
A	1	6
B	1	5
C	2	4
D	4	7

Free: 2

	Has	Max
A	1	6
B	2	5
C	2	4
D	4	7

Free: 1

- Bankers' algorithm: before granting a request, ensure that a sequence exists that will allow all processes to complete.

Banker's Algorithm

Process	Tape drives	Plotters	Scanners	CD ROMs
A	3	0	1	1
B	0	1	0	0
C	1	1	1	0
D	1	1	0	1
E	0	0	0	0

Resources assigned

Process	Tape drives	Plotters	Scanners	CD ROMs
A	1	1	0	0
B	0	1	1	2
C	3	1	0	0
D	0	0	1	0
E	2	1	1	0

Resources still needed

E = (6342)
P = (5322)
A = (1020)

E: Existing resources

P: Possessed resources

A: Available resources

Example of banker's algorithm with multiple resources