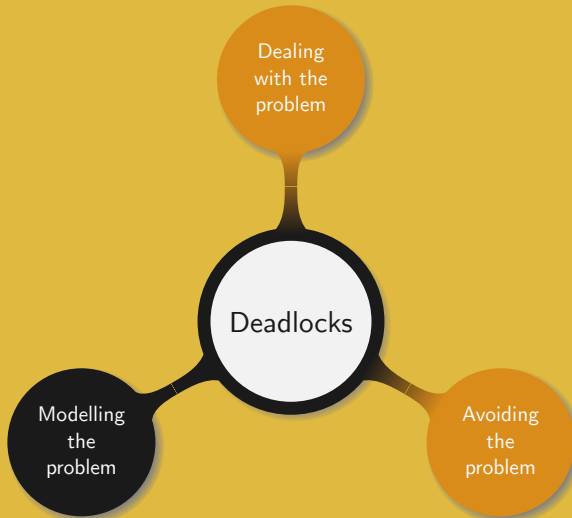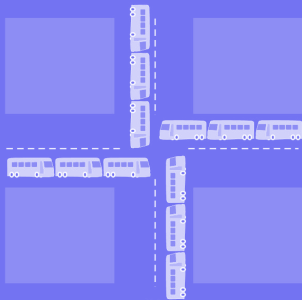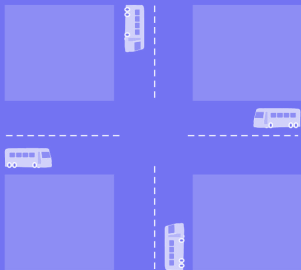# Introduction to Operating Systems

5. Deadlocks

Manuel – Fall 2021

Two main types of resources:

- Preemptable: resources that can be safely taken away

- Non-preemptable: resources cannot be safely taken away

Examples.

- Preemptable:

  - Total RAM is 256MB, A and B are 256 MB each

  - A is loaded in RAM, acquires the printer, but exceeds its quantum

  - B is loaded in RAM but fails to acquire the memory

  - Memory can be taken away of B and given to A, which can complete

- Non-preemptable:

  - A is burning a DVD

  - B wants to access the DVD burner, but the drive is not accessible

  - The resource cannot be taken way from A without any damage
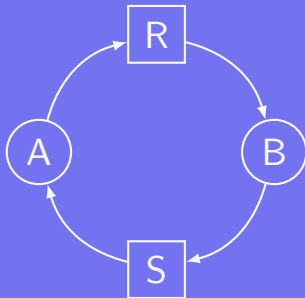
Graphs to represent resource allocation
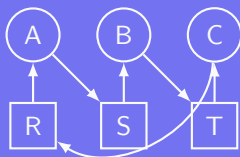
Resource *R* held by *A*   Resource *R* requested by *A*   Deadlock
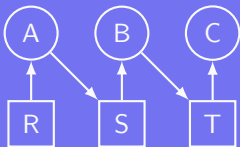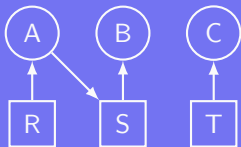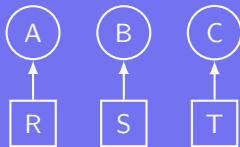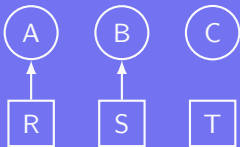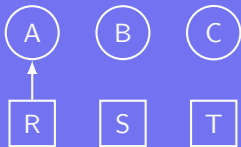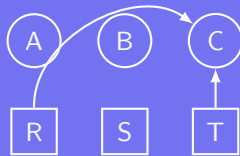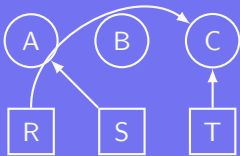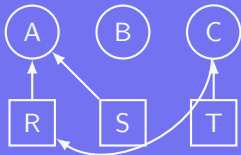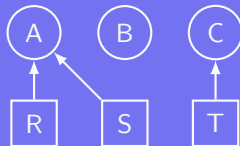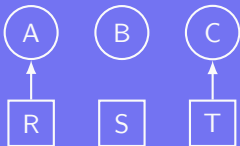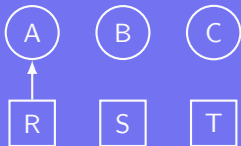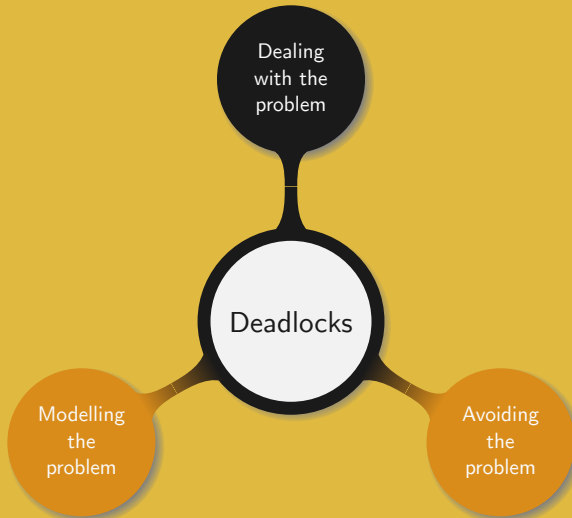
ALERT, ALERT

HIDE!!!!

Example. Is there any deadlock in the following system with seven processes (A–G) and six resources (R–W)?

- Process A holds R and wants S

- Process B wants T

- Process C want S

- Process D holds U and wants both S and T

- Process E holds T and wants V

- Process F holds W and wants S

- Process G holds V and wants U
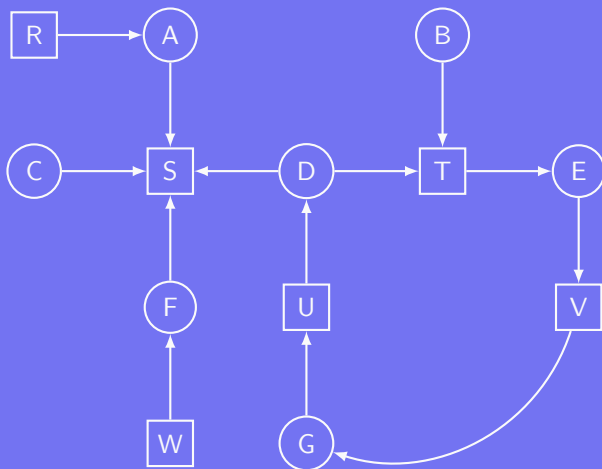
Let E and A be two vectors representing the existing and the available resources respectively. C represents the current allocation matrix and R the request matrix.

Four resource types: Printer, Scanner, DVD burner and Plotter

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix} \qquad\qquad A = \begin{pmatrix} 2 & 1 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \qquad\qquad R = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{pmatrix}$$

Let E and A be two vectors representing the existing and the available resources respectively. C represents the current allocation matrix and R the request matrix.

Four resource types: Printer, Scanner, DVD burner and Plotter

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix} \qquad A = \begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 1 \\ 2 & 2 & 1 & 0 \end{pmatrix} \qquad R = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Let E and A be two vectors representing the existing and the available resources respectively. C represents the current allocation matrix and R the request matrix.

Four resource types: Printer, Scanner, DVD burner and Plotter

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix} \qquad\qquad A = \begin{pmatrix} 2 & 2 & 1 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 1 \\ . & . & . & . \end{pmatrix} \qquad\qquad R = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ . & . & . & . \end{pmatrix}$$

Let E and A be two vectors representing the existing and the available resources respectively. C represents the current allocation matrix and R the request matrix.

Four resource types: Printer, Scanner, DVD burner and Plotter

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix} \qquad\qquad A = \begin{pmatrix} 1 & 2 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 3 & 0 & 2 & 1 \\ . & . & . & . \end{pmatrix} \qquad\qquad R = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ . & . & . & . \end{pmatrix}$$

Let E and A be two vectors representing the existing and the available resources respectively. C represents the current allocation matrix and R the request matrix.

Four resource types: Printer, Scanner, DVD burner and Plotter

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix} \qquad\qquad A = \begin{pmatrix} 4 & 2 & 2 & 1 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix} \qquad\qquad R = \begin{pmatrix} 2 & 0 & 0 & 1 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

Let E and A be two vectors representing the existing and the available resources respectively. C represents the current allocation matrix and R the request matrix.

Four resource types: Printer, Scanner, DVD burner and Plotter

$$E = \begin{pmatrix} 4 & 2 & 3 & 1 \end{pmatrix} \qquad\qquad A = \begin{pmatrix} 2 & 1 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \qquad\qquad R = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 2 & 0 \\ 2 & 1 & 0 & 0 \end{pmatrix}$$

What if the second process requests two DVD burners?

Three main recovery strategies:

- Preemption:

    - Take a resource from another process

    - Might require manual intervention (e.g. collect papers from printer, pile them up and resume printing later)

- Rollback:

    - Set periodical checkpoints on processes

    - Save process state at the checkpoints

    - Restart process at a checkpoint (from before the deadlock)

- Killing:

    - Simplest strategy

    - Kill a process that uses resources related to the deadlock

    - Pick a process that can be re-run from the beginning
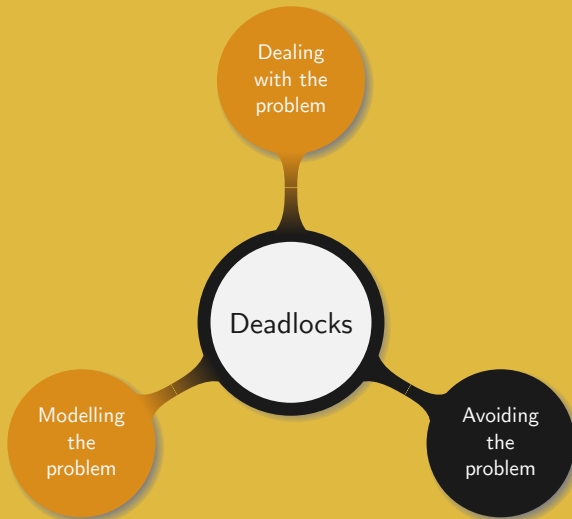
Avoiding deadlocks:

- Resources could be assigned and released one at a time

- Problems to solve

  - How to implement this idea?

  - How efficient would such a strategy be?

Preventing deadlocks:

- Model the circumstances under which a deadlock occurs

- Problems:

  - How would such a model be?

  - How much would this strategy slow down the system?
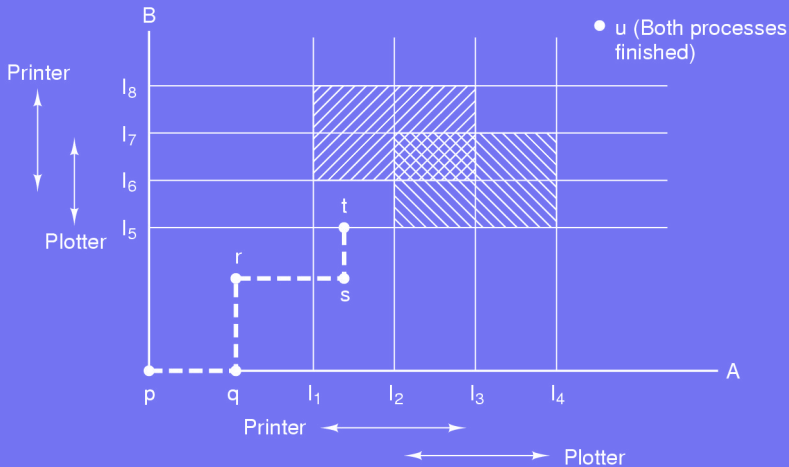
Using the matrices E, A, C and R, define:

- Safe state: there exists an order allowing all processes to complete, even if they request their maximum number of resources when scheduled. It can guarantee that all processes can finish.

- Unsafe state: the ability of the system not to deadlock depends on the order the resources are allocated and deallocated. There is no way to predict whether or not all the processes will finish.

Using the matrices E, A, C and R, define:

- Safe state: there exists an order allowing all processes to complete, even if they request their maximum number of resources when scheduled. It can guarantee that all processes can finish.

- Unsafe state: the ability of the system not to deadlock depends on the order the resources are allocated and deallocated. There is no way to predict whether or not all the processes will finish.

Remarks.

- An unsafe state does not necessarily imply a deadlock; the system can still run for a while, or even complete all processes if some release their resources before requesting some more.

- A process running alone will never encounter a deadlock. If it needs more resources than available it is assumed that they will not be all requested at once.

General idea:

- Introduced by Dijkstra in 1965

- Based on the detection algorithm

- Idea: avoid deadlocks by avoiding to run into an unsafe state

- Any request leading to an unsafe state is denied

General idea:

- Introduced by Dijkstra in 1965

- Based on the detection algorithm

- Idea: avoid deadlocks by avoiding to run into an unsafe state

- Any request leading to an unsafe state is denied

Remark. Mostly useless in practice since a process rarely knows the maximum resources it will need and the number of processes keeps varying. It also does not take into account hardware related issues, e.g. crashed printer

Deciding whether a state is safe or not:

1. Select a row in R whose resource request can be met. If no such row exists there is a possibility for a deadlock

2. When the process terminates it releases all its resource, and they can be added to the vector A

3. If all the processes terminate when repeating steps 1. and 2. then the sate is safe. If step 1. fails at any stage (not all the processes being finished) then the state is unsafe and the request should be denied

Example. Consider a system with 6 scanners, 3 plotters, 4 printers and 2 DVD drives:

$$E = \begin{pmatrix} 6 & 3 & 4 & 2 \end{pmatrix} \qquad\qquad A = \begin{pmatrix} 1 & 1 & 2 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 3 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad R = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 3 & 1 & 1 & 0 \end{pmatrix}$$

Example. Consider a system with 6 scanners, 3 plotters, 4 printers and 2 DVD drives:

$$E = \begin{pmatrix} 6 & 3 & 4 & 2 \end{pmatrix} \qquad A = \begin{pmatrix} 2 & 2 & 2 & 1 \end{pmatrix}$$

$$C = \begin{pmatrix} 3 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ . & . & . & . \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad R = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 3 & 1 & 0 & 0 \\ . & . & . & . \\ 3 & 1 & 1 & 0 \end{pmatrix}$$

Example. Consider a system with 6 scanners, 3 plotters, 4 printers and 2 DVD drives:

$$E = \begin{pmatrix} 6 & 3 & 4 & 2 \end{pmatrix} \qquad\qquad A = \begin{pmatrix} 5 & 2 & 3 & 2 \end{pmatrix}$$

$$C = \begin{pmatrix} . & . & . & . \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ . & . & . & . \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad R = \begin{pmatrix} . & . & . & . \\ 0 & 1 & 1 & 1 \\ 3 & 1 & 0 & 0 \\ . & . & . & . \\ 3 & 1 & 1 & 0 \end{pmatrix}$$

Example. Consider a system with 6 scanners, 3 plotters, 4 printers and 2 DVD drives:

$$E = \begin{pmatrix} 6 & 3 & 4 & 2 \end{pmatrix} \qquad\qquad A = \begin{pmatrix} 1 & 1 & 2 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 3 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad R = \begin{pmatrix} 3 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 3 & 1 & 1 & 0 \end{pmatrix}$$

If the first process requests 3 scanners instead of 2, is the sate safe?

Resource deadlocks only occur under the following four conditions:

- Mutual exclusion: a resource can be assigned to at most one process at a time

- Hold and wait: a process currently holding some resources can request some more

- No preemption: resources must be released by the process itself, i.e. they cannot be taken away by another process

- Circular wait: there is a circular chain of processes each of them waiting from some resources held by another process

Preventing deadlocks:

- Not possible to remove it, e.g. two processes cannot print at the same time

- Use daemon that handle specific output, e.g. printing daemon uses SPOOL

- Deadlock can still happen, e.g. two processes fill up the SPOOL disk, without any of them being full

- SPOOL cannot always be applied

  Aside from carefully assigning resources not much can be done

Preventing deadlocks:

- Require processes to claim all the resources at once

- Not realistic, a process does not always know what resources will be necessary

- What if computation last for hours, and then the result is burnt on a DVD?

- Resources are not handle in an optimal way

- Alternative strategy: process has to release its resources before getting new ones

Possible, but far from optimal

Preventing deadlocks:

- Issue inherent to the hardware

- Often impossible to do anything (e.g. stop burning a DVD and resume later)

- Might require human intervention (e.g. stop a printer job, get the already printed pages, print another job and resume the first one)

Not a viable solution

Preventing deadlocks:

- Order the resources

- Processes have to request resources in increasing order

- A process can only request a lower resource if it has released all the larger ones

- Is there an order satisfying everybody?

    Best solution but it not always possible to use it in practice

Deadlocks are not necessarily related to hardware resources:

- Database records: two-phase locking solution; lock all the records in phase one; if one fails release all the locks and retry later, otherwise proceed with phase two and manipulate the records

- Communication deadlocks: mutex can lead to deadlocks; no hardware resource involved; could be due to the loss of a message

- Livelock: lack of resources, process can not keep going so seat in tight loop and keeps trying not knowing it is hopeless

- Starvation: a long process delayed to let shorter ones run, might never run...

- How to detect deadlocks?

- How to fix a deadlock in a clean way?

- What are the four conditions that characterise a deadlock?

- What is a common practice regarding deadlocks?

Thank you!