# Deadlocks

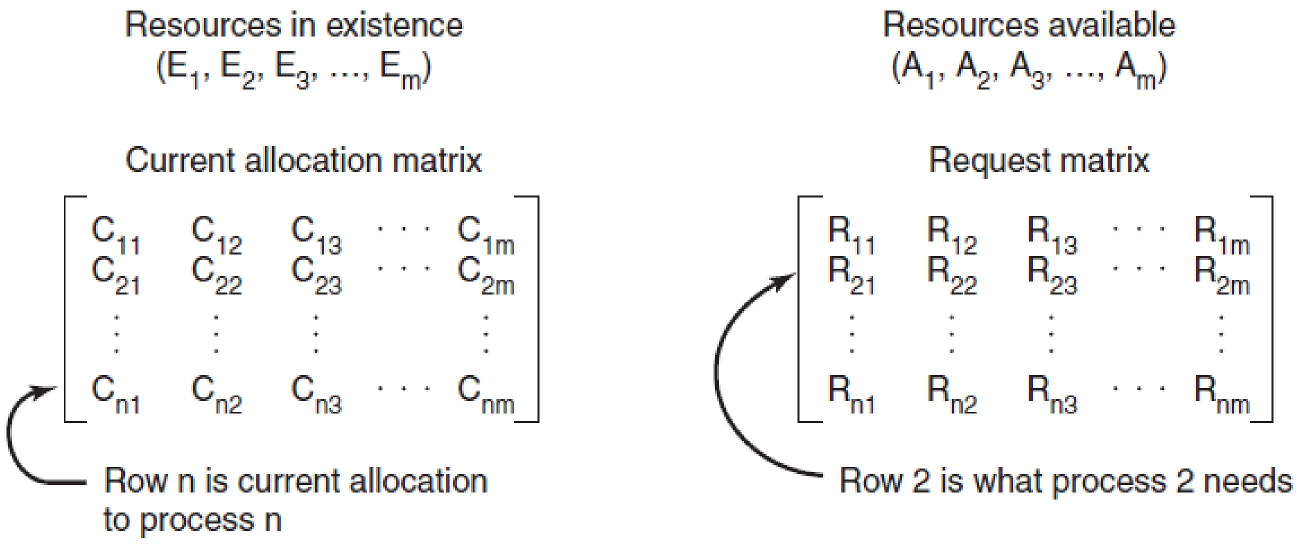
Week 13 – Lab

## Deadlock Detection

- Let the number of resource classes be *m*, with $E_1$ resources of class 1, $E_2$ resources of class 2, and generally, $E_i$ resources of class i ($1 \leq i \leq m$).
- E is the existing resource vector. It gives the total number of instances of each resource in existence
- Let A be the available resource vector, with $A_i$ equals the number of instances of resource *i* that are currently available (i.e., unassigned)

# Deadlock Detection

- We have two arrays:
  - C, the current allocation matrix
  - R, the request matrix


Resources in existence
$(E_1, E_2, E_3, \ldots, E_m)$
Current allocation matrix
$\begin{bmatrix} C_{11} & C_{12} & C_{13} & \cdots & C_{1m} \\ C_{21} & C_{22} & C_{23} & \cdots & C_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ C_{n1} & C_{n2} & C_{n3} & \cdots & C_{nm} \end{bmatrix}$
Row n is current allocation to process n
Resources available
$(A_1, A_2, A_3, \ldots, A_m)$
Request matrix
$\begin{bmatrix} R_{11} & R_{12} & R_{13} & \cdots & R_{1m} \\ R_{21} & R_{22} & R_{23} & \cdots & R_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ R_{n1} & R_{n2} & R_{n3} & \cdots & R_{nm} \end{bmatrix}$
Row 2 is what process 2 needs

## Exercise 1 (1/2)

- Write a C program for deadlock detection algorithm:
  - Program should read the input from a file (*input.txt*)
  - For testing purposes consider 5 processes and 3 type of resources. However, your program must be able to process as many processes and resource types, as needed (check next slide for input file structure description)
  - Your program should either say that no deadlock is detected or print out the numbers of processes that are deadlocked
  - Save source code to *ex1.c*
  - Save both kinds of input and output (with and without DL). Use consistent naming, e.g. *input_dl.txt*, *input_ok.txt*, *output_dl.txt*, *output_ok.txt*

# Exercise 1 (2/2)

- Data shown on the left will be stored in input file as shown on the right
  - E= (7 2 6)
  - A= (0 0 0)
  - Current allocation matrix: C =

    0 1 0
    2 0 0
    3 0 3
    2 1 1
    0 0 2
  - Request Matrix R =

    0 0 0
    2 0 2
    0 0 0
    1 0 0
    0 0 2

Data

```
7 2 6

0 0 0

0 1 0
2 0 0
3 0 3
2 1 1
0 0 2

0 0 0
2 0 2
0 0 0
1 0 0
0 0 2
```

Input file

## Deadlock Avoidance

- Banker’s Algorithm:
  - The algorithm checks if granting the request leads to an unsafe state
  - In that case, the request is denied

## Deadlock Avoidance

- The algorithm for checking to see if a state is safe can be stated as:
  - Look for a row, R, whose unmet resource needs are all smaller than or equal to A. If no such row exists, the system will eventually deadlock since no process can run to completion
  - Assume the process of the chosen row requests all the resources it needs (which is guaranteed to be possible) and finishes. Mark that process as terminated and add all of its resources to the A vector
  - Repeat steps 1 and 2 until either all processes are marked terminated (in which case the initial state was safe) or no process is left whose resource needs can be met (in which case the system was not safe)

# Exercise 2

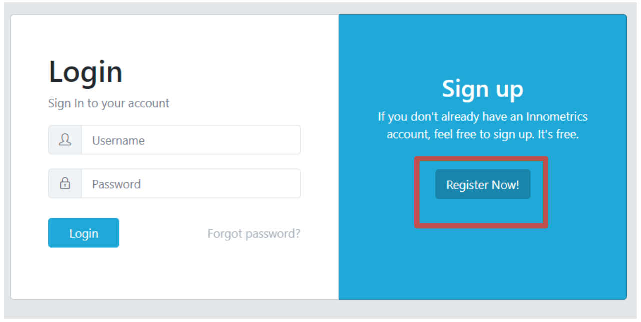
- Theory covered three strategies of dealing with deadlocks: recovery, prevention and avoidance
- Explain step by step in which cases one strategy might be better than others and save your answer to *ex2.txt* file

## Exercise 3 (Optional)

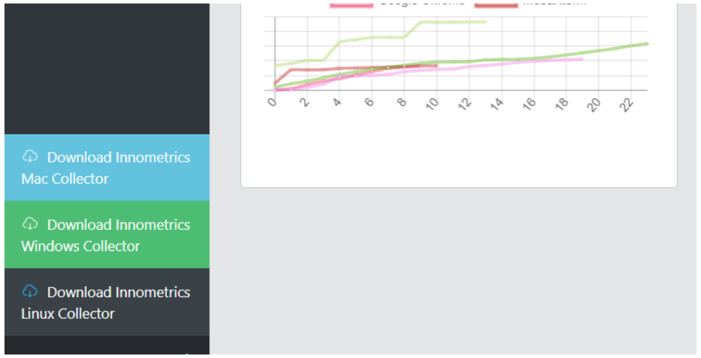
- Create two different solutions for dining philosophers problem

# Innometrics installation(1/2)

- Go to https://innometrics-12856.firebaseapp.com/ and create your account



- Download and Install the data collector based on your preferred operative system.


Download Innometrics Mac Collector
Download Innometrics Windows Collector
Download Innometrics Linux Collector

## Innometrics installation(2/2)

- Run and sign-in on the data collector

- Note: this application will be required to participate in the pre-final and final exam.