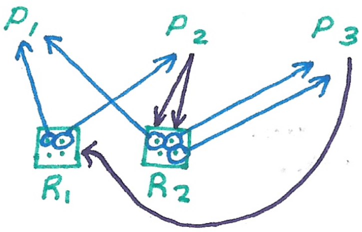
CSC 130 Operating System Principles March 24, 2020

Review:



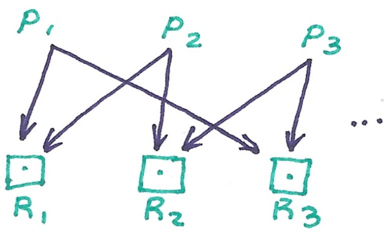
... three processes...

... two resources with 4 instances.

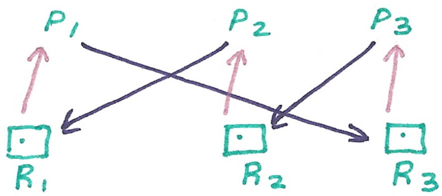
P_1... is not blocked.

P_2... is blocked, only one request for resources can be approved.

P_3... is not blocked.



...



approved edge

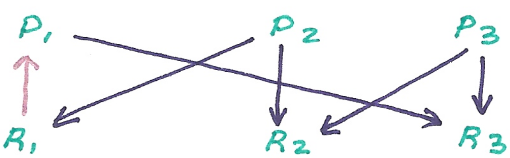
request edge

DEADLOCK!

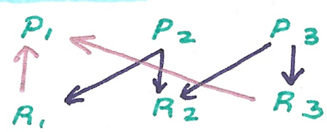
What if we change order of requests?

There can be two possiblities?

P_1 wins

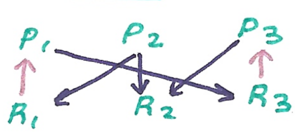


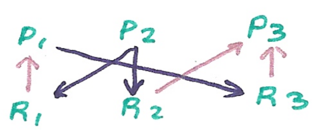
P_1 wins



No deadlock!

P_3 wins





No deadlock!

Deadlock prevention

Mutal-Exclusion & No premption not much we can do to affect this, based off the nature of the operating system.

But Hold-&-wait... yes we can influence!

$R_1 \longrightarrow P_1 \longrightarrow R_2$

A system can guaretee that you can only have a request or allocate edge, a deadlock is structurally impossible.

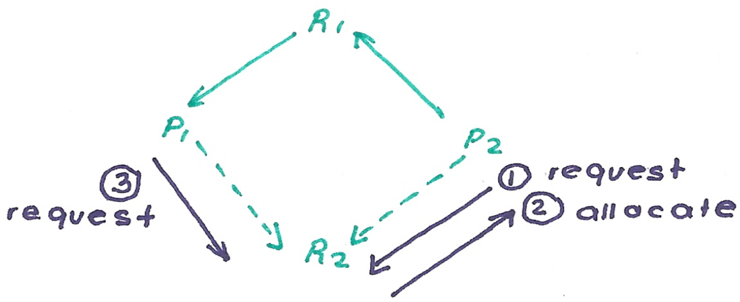
A process can only ever request a resource or be allocated all resources.

Deadlock Avoidance

a system trys to define a safe condition before it makes any action.

a safe state exists if all processes in system can have satisfied request.

Resource - Allocation Graph Scheme



This series of steps will result in a deadlock.

Banker's Algorithm

The banker's algorithm resource allocation & deadlock avoidance algorithm.

Is the current system in a safe state?

'n' = # of process

'm' = # of resource types

available = indicates the # of available resources of each type

max = maximum demand of each process

allocation = # of resources of each type currently allocated to each process

need = remaining resource needed of each process

need [i, j] = max [i, j] - allocation [i, j]

	Need		
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Safety algorithm:

1. Initialize: work = Available

   work = Available =

A	B	C
3	3	2

   Finish[i] = false

Finish =	0	1	2	3	4
	F	F	F	F	F

2. For i = 0

   Need 0 = 7 4 3 < work = 3 3 2 ✗

   For i = 1

   Need 1 = 1 2 2 < work = 3 3 2 ✓

   P_1 can be kept in safety sequence

   For i = 2

   Need 2 = 6 0 0 < work = 332 + 200 = 532 ✗

   For i = 3

   Need 3 = 0 1 1 < work = 5 3 2 ✓

   P_3 can stay in sequence!

   For i = 4

   Need 4 = 4 3 2 < work 532 + 211 = 743 ✓

   P_4 can stay in safety sequence?

Need i ≤ work must be met to stay in safety sequence.