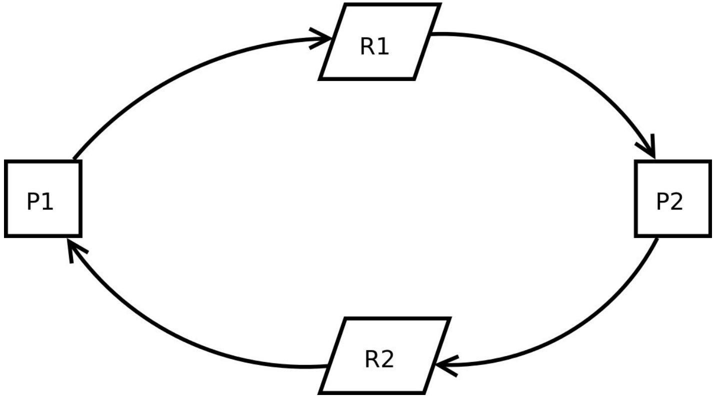
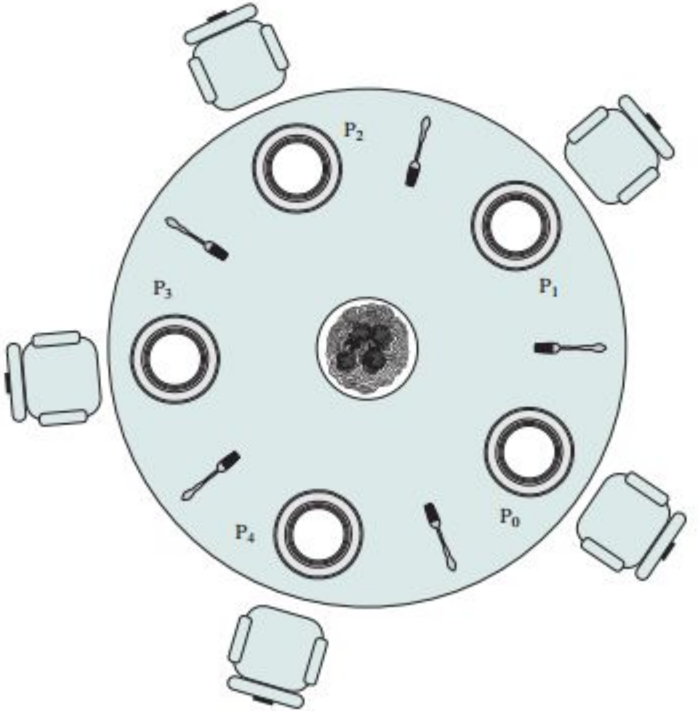
Discussion 3/15/19

Deadlocks

- A situation, typically involving opposing parties, in which no progress can be made.
- Necessary conditions (Coffman conditions):
  - Mutual Exclusion
  - Resource holding
  - No preemption
  - Circular wait
- Deal with them:
  - Ignore
  - Detect
  - Avoid
  - Prevent



Dining Philosophers



- Deadlock
- Livelock
- Resource Hierarchy
- Semaphores

Bankers Algorithm

- Tests for safety by simulating the allocation of predetermined maximum possible amounts of all resources, and then makes an "s-state" check to test for possible deadlock conditions for all other pending activities
- Need to know
    - How much of each resource each process could possibly request[MAX]
    - How much of each resource each process is currently holding[ALLOCATED]
    - How much of each resource the system currently has available[AVAILABLE]

Bankers Algorithm

- Let **n** be the number of processes in the system and **m** be the number of resource types. Then we need the following data structures:
  - **Available**: A vector of length m indicates the number of available resources of each type. If Available[j] = k, there are k instances of resource type R_j available.
  - **Max**: An $n \times m$ matrix defines the maximum demand of each process. If Max[i,j] = k, then P_i may request at most k instances of resource type R_j.
  - **Allocation**: An $n \times m$ matrix defines the number of resources of each type currently allocated to each process. If Allocation[i,j] = k, then process P_i is currently allocated k instances of resource type R_j.
  - **Need**: An $n \times m$ matrix indicates the remaining resource need of each process. If Need[i,j] = k, then P_i may need k more instances of resource type R_j to complete the task.

Note: Need[i,j] = Max[i,j] - Allocation[i,j]. n=m-a.

Bankers Algorithm - Safety

1) Let work (length m) = available, finish (length n) = {false}
2) Find and i such that
   a) Finish[i] = false
   b) Need[i] <= work
   c) If none exist, go to step 4
3) Work = Work + allocation[i]
   a) Finish[i] = true
   b) Goto step 2
4) If finish[i] is true for all i, then we are safe

Bankers Algorithm - Requests

1) If request[i] <= Need[i]
    a) Goto step 2, otherwise raise an error (exceeded maximum claim)
2) If request[i] <= Available
    a) Goto step 3, otherwise we must wait
3) Have the system pretend to allocate by doing:
    a) Available = Available - Request[i]
    b) Allocation[i] = Allocation[i] + Request[i]
    c) Need[i] = Need[i] - Request[i]

Bankers Algorithm - Example

Process	Allocation			Max			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	7	5	3	3	3	2
P_1	2	0	0	3	2	2			
P_2	3	0	2	9	0	2			
P_3	2	1	1	2	2	2			
P_4	0	0	2	4	3	3			

- What will be the need matrix?
- Is the system safe? What is the safe sequence?

Bankers Algorithm - Example

m=3, n=5 — Step 1 of Safety Algo

Work = Available

Work =

3	3	2

Finish =

0	1	2	3	4
false	false	false	false	false

For i = 0 — Step 2

$Need_0 = 7, 4, 3$

Finish [0] is false and $\overset{7,4,3}{Need_0} > \overset{3,3,2}{Work}$

So P_0 must wait — But Need $\leq$ Work

For i = 1 — Step 2

$Need_1 = 1, 2, 2$

Finish [1] is false and $\overset{1,2,2}{Need_1} < \overset{3,3,2}{Work}$

So P_1 must be kept in safe sequence

Step 3

$Work = \overset{3,3,2}{Work} + \overset{2,0,0}{Allocation_1}$

Work =

A	B	C
5	3	2

Finish =

0	1	2	3	4
false	true	false	false	false

For i = 2 — Step 2

$Need_2 = 6, 0, 0$

Finish [2] is false and $\overset{6,0,0}{Need_2} > \overset{5,3,2}{Work}$

So P_2 must wait

For i=3 — Step 2

$Need_3 = 0, 1, 1$

Finish [3] = false and $\overset{0,1,1}{Need_3} < \overset{5,3,2}{Work}$

So P_3 must be kept in safe sequence

Step 3

$Work = \overset{5,3,2}{Work} + \overset{2,1,1}{Allocation_3}$

Work =

A	B	C
7	4	3

Finish =

0	1	2	3	4
false	true	false	true	false

For i = 4 — Step 2

$Need_4 = 4, 3, 1$

Finish [4] = false and $\overset{4,3,1}{Need_4} < \overset{7,4,3}{Work}$

So P_4 must be kept in safe sequence

Step 3

$Work = \overset{7,4,3}{Work} + \overset{0,0,2}{Allocation_4}$

Work =

A	B	C
7	4	5

Finish =

0	1	2	3	4
false	true	false	true	true

For i = 0 — Step 2

$Need_0 = 7, 4, 3$

Finish [0] is false and $\overset{7,4,3}{Need} < \overset{7,4,5}{Work}$

So P_0 must be kept in safe sequence

Step 3

$Work = \overset{7,4,5}{Work} + \overset{0,1,0}{Allocation_0}$

Work =

A	B	C
7	5	5

Finish =

0	1	2	3	4
true	true	false	true	true

For i = 2 — Step 2

$Need_2 = 6, 0, 0$

Finish [2] is false and $\overset{6,0,0}{Need_2} < \overset{7,5,5}{Work}$

So P_2 must be kept in safe sequence

Step 3

$Work = \overset{7,5,5}{Work} + \overset{3,0,2}{Allocation_2}$

Work =

A	B	C
10	5	7

Finish =

0	1	2	3	4
true	true	true	true	true

Step 4

Finish [i] = true for $0 \leq i \leq n$

Hence the system is in Safe state

The safe sequence is P_1, P_3, P_4, P_0, P_2

Assignment 3 Questions?