5. Handling Deadlocks

- Prevention (prevent 1 of the 4 conditions from happening)

– Mutual exclusion. Ex. a unit of CPU time can't be shared.

– Resource holding. Ex. Try to satisfy a job's request completely.

– No preemption. Ex. Allow OS to deallocate resources from jobs.

– Circular waiting. Ex. Try to force the graph to be without cycles, e.g., numbering the same resources as #1, #2, #3, ⋯.

- Avoidance (Banker's Algorithm)
  - 1. No customer will be granted a loan exceeding the bank's total capital.
  - 2. A customer will be given a maximum credit limit.
  - 3. No customer will be allowed to borrow over the limit.
  - 4. Sum of all loans ≤ bank's total capital.

Ex.

Job #	Devices allocated		Max limit	(Remaining)	
1	0	2	4	4	2
2	2	3	5	3	2
3	4	4	8	4	4

system has 10 devices

$10-(0+2+4)=4$ system has 4 left

Safe state ↳ there is one way out.

system has only $10-(2+3+4) = 1$ left, unsafe state

CS418: Exercise on processor management-II

Consider a computing system with 13 tape drives. All jobs running on this system require a maximum of 5 tape drives to complete. Assume all of the jobs run for long periods of time with just 4 drives and request the 5th one only at the very end of the run. The job request stream is endless.

a. If your OS supports a very conservative device allocation policy that no job will be started unless all tapes required have been allocated to it for the duration of its run:

(2.1) What is the maximum number of jobs that can be active at once?

2 $(2 \times 5 = 10$ devices$)$

(2.2) What are the minimum and maximum number of tape drives that may be idle as a result of the policy? Why?

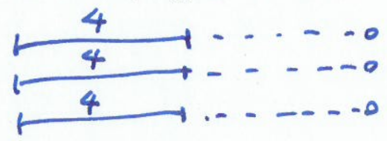
min: 3

max: 5

b. If your OS supports the Banker's algorithm:

(2.3) What is the maximum number of jobs that can be in progress at once?

3



(2.4) What are the minimum and maximum number of tape drives that may be idle as a result of the policy? Why?

min: 0

max: 1

- Detection (CES Algorithm, by Coffman, et al, 1971)
  - 1. Mark each process that has a row of 0's in the allocation matrix.
  - 2. $W \leftarrow$ available vector.
  - 3. Find i such that process i is currently unmarked and row$-i$ of Q is $\leq W$. If no such row exists, exit.
  - 4. If such a row is found, mark process i and $W \leftarrow W + A_i$, where A is the allocation matrix. Repeat Step 3.
  - **A deadlock exists if and only if there are unmarked processes at the end of the algorithm).**

- Recovery

① stop all jobs.

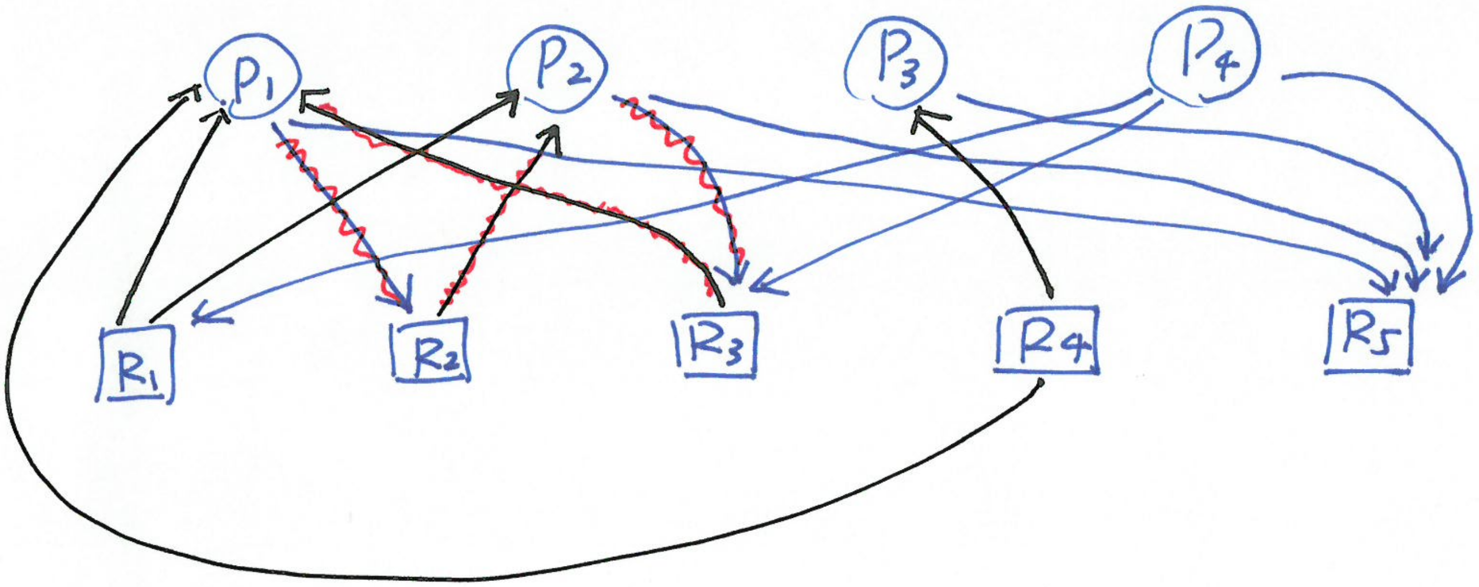
② Stop any of the jobs involved in the deadlock.

③ Terminate the jobs involved in the deadlock one by one, and check whether the problem is resolved.

④ Terminate a job not involved in the deadlock but with sufficient amount of resources, deallocate them to the jobs involved in the deadlock.

Example. (On Deadlock Detection, e.g., CES algorithm)

The set up is in the handout (4.pdf).



$\longrightarrow$: from the Allocate Matrix

$\longrightarrow$: from the Request Matrix

$$\begin{tikzcd} P_1 \arrow[r] & \boxed{R_2} \arrow[r] & P_2 \arrow[r] & \boxed{R_3} \arrow[lll, bend left] \end{tikzcd}$$

forms a cycle, hence P_1, P_2 are deadlocked.

6. Starvation

- The dining philosophers problem (Dijsktra, 1968)

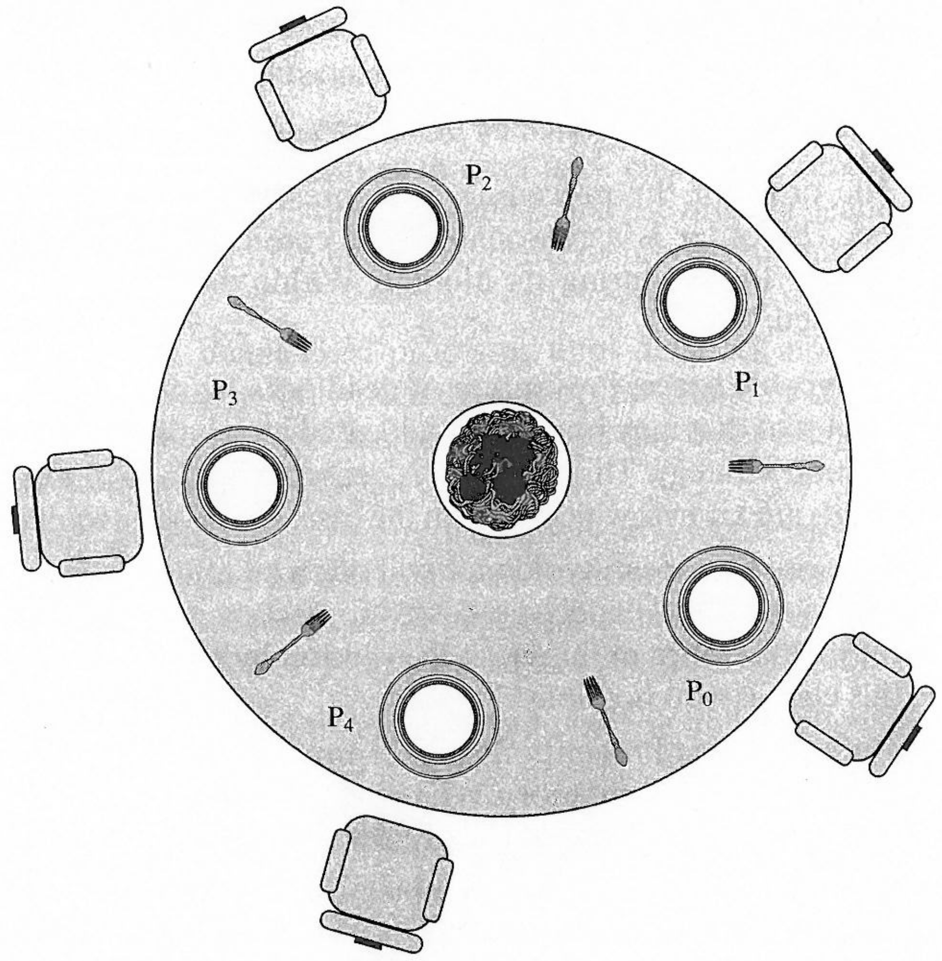


Figure 6.11 Dining Arrangement for Philosophers