CSC 112: Computer Operating Systems
Lecture 4

Deadlocks Exercises Solution

Department of Computer Science,
Hofstra University

Acknowledgement: Lecture slides based on UC Berkeley CS 162: Operating Systems and System Programming

Quiz: Deadlocks

- Is there a possible deadlock?
- Yes, there is a deadlock. Consider the following interleaving:
  - thread 1 executes L1.wait(); no blocking
  - thread 2 executes L3.wait(); no blocking
  - thread 3 executes L2.wait(); no blocking
- Now there is a circular wait condition:
- thread 1 waiting for L2 (held by thr. 3) → thr. 2 waiting for L1 (held by thr. 1) → thr. 3 waiting for L3 (held by thr. 2).
- Solution: each thread should acquire locks in the same order, say, L1, L2, L3.

```
1   Semaphore L1=1, L2=1, L3=1;
2
3   // Thread 1:
4   L1.wait();
5   L2.wait();
6   // critical section requiring L1 and L2 locked.
7   L2.post();
8   L1.post();
9
10  // Thread 2:
11  L3.wait();
12  L1.wait();
13  // critical section requiring L3 and L1 locked.
14  L1.post();
15  L3.post();
16
17  // Thread 3:
18  L2.wait();
19  L3.wait();
20  // critical section requiring L2 and L3 locked.
21  L3.post();
22  L2.post();
```

Quiz: Banker's Algorithm I

- 4 processes P1 through P5; 3 resource types R1, R2, R3 with 7, 3, 6 instances each.
- Run Banker's algorithm to check if the current state is safe. If yes, give a safe sequence of process completions and fill in the table with the sequence of process completions without deadlock, and available resources after the completion of each process.
- (You will be graded on "Need matrix", and "Available resources after completion of each process".)

Max

$$R = \begin{bmatrix} 7 & 5 & 3 \\ 3 & 2 & 2 \\ 9 & 0 & 2 \\ 2 & 2 & 2 \\ 4 & 3 & 3 \end{bmatrix}$$

Allocation

$$C = \begin{bmatrix} 0 & 1 & 0 \\ 2 & 0 & 0 \\ 3 & 0 & 2 \\ 2 & 1 & 1 \\ 0 & 0 & 2 \end{bmatrix}$$

Total

$$E = [7 \quad 3 \quad 6]$$

Available resources after completion of each process

	R1	R2	R3
Init			

Quiz Solution: Banker's Algorithm I

- First compute the Need matrix as *Max – Allocation,* and Available vector *A.*
- The state is not safe, the execution sequence P4, P2 leads to a deadlock state, where none of the remaining processes P1, P3, P5 can finish.

Max

$$R = \begin{bmatrix} 7 & 5 & 3 \\ 3 & 2 & 2 \\ 9 & 0 & 2 \\ 2 & 2 & 2 \\ 4 & 3 & 3 \end{bmatrix}$$

Allocation

$$C = \begin{bmatrix} 0 & 1 & 0 \\ 2 & 0 & 0 \\ 3 & 0 & 2 \\ 2 & 1 & 1 \\ 0 & 0 & 2 \end{bmatrix}$$

Need

$$R - C = \begin{bmatrix} 7 & 4 & 3 \\ 1 & 2 & 2 \\ 6 & 0 & 0 \\ 0 & 1 & 1 \\ 4 & 3 & 1 \end{bmatrix}$$

Total

$$E = [7 \quad 3 \quad 6]$$

Available

$$A = [0 \quad 1 \quad 1]$$

Available resources after completion of each process

	R1	R2	R3
Init	0	1	1
P4	2	2	2
P2	4	2	2
	Deadlock		

Your task

Quiz: Banker's algorithm II

- 4 processes P1, P2, P3; 3 resource types R1, R2, R3 with 8, 6, 4 instances each.
- 1) Run Banker's algorithm to check if the current state is safe. If yes, give a safe sequence of process completions and fill in the table with the sequence of process completions without deadlock, and available resources after the completion of each process.
- 2) Starting from the initial state, if P1 makes request for 2 more instances of resource 3, should we grant it?
- 3) Starting from the initial state, if P2 makes request for 2 more instances of resource 1, should we grant it?

Max

$$\begin{bmatrix} 8 & 4 & 3 \\ 6 & 2 & 0 \\ 3 & 3 & 3 \end{bmatrix}$$

Allocation

$$\begin{bmatrix} 0 & 0 & 1 \\ 3 & 2 & 0 \\ 2 & 2 & 1 \end{bmatrix}$$

Total

$$E = [8 \quad 6 \quad 4]$$

Available resources after completion of each process

	R1	R2	R3
Init			

Quiz Solution: Banker's algorithm II

1) The initial state is safe, with safe sequences of P2, P3, P1 or P2, P2, P1

Max

$$\begin{bmatrix} 8 & 4 & 3 \\ 6 & 2 & 0 \\ 3 & 3 & 3 \end{bmatrix}$$

Allocation

$$\begin{bmatrix} 0 & 0 & 1 \\ 3 & 2 & 0 \\ 2 & 2 & 1 \end{bmatrix}$$

Need

$$\begin{bmatrix} 8 & 4 & 2 \\ 3 & 0 & 0 \\ 1 & 1 & 2 \end{bmatrix}$$

Total

$E = [8 \quad 6 \quad 4]$

Available

$E = [3 \quad 2 \quad 2]$

Available resources after completion of each process

	R1	R2	R3
Init	3	2	2
P2	6	4	2
P3	8	6	3
P1	8	6	4

Or

Available resources after completion of each process

	R1	R2	R3
Init	3	2	2
P3	5	4	3
P2	8	6	3
P1	8	6	4

Quiz Solution: Banker's algorithm II

2) Starting from the initial state, if P1 makes request for 2 more instances of resource 3, then we calculate the state of the system if this request is fulfilled.

The state is unsafe, so we deny this request.

Max

$$\begin{bmatrix} 8 & 4 & 3 \\ 6 & 2 & 0 \\ 3 & 3 & 3 \end{bmatrix}$$

Allocation

$$\begin{bmatrix} 0 & 0 & 3 \\ 3 & 2 & 0 \\ 2 & 2 & 1 \end{bmatrix}$$

Need

$$\begin{bmatrix} 8 & 4 & 0 \\ 3 & 0 & 0 \\ 1 & 1 & 2 \end{bmatrix}$$

Total

$E = [8 \quad 6 \quad 4]$

Available

$E = [3 \quad 2 \quad 0]$

Available resources after completion of each process

	R1	R2	R3
Init	3	2	0
P2	6	4	0
	Deadlocked		

Quiz Solution: Banker's algorithm II

3) Starting from the initial state, if P2 makes request for 2 more instances of resource 1, then we calculate the state of the system if this request is fulfilled.

The state is safe, with safe sequences of P2, P3, P1 or P2, P2, P1, so we can grant this request.

Max

$$\begin{bmatrix} 8 & 4 & 3 \\ 6 & 2 & 0 \\ 3 & 3 & 3 \end{bmatrix}$$

Allocation

$$\begin{bmatrix} 0 & 0 & 1 \\ 5 & 2 & 0 \\ 2 & 2 & 1 \end{bmatrix}$$

Need

$$\begin{bmatrix} 8 & 4 & 2 \\ 1 & 0 & 0 \\ 1 & 1 & 2 \end{bmatrix}$$

Total

$E = [8 \quad 6 \quad 4]$

Available

$E = [1 \quad 2 \quad 2]$

Available resources after completion of each process

	R1	R2	R3
Init	1	2	2
P2	6	4	2
P3	8	6	3
P1	8	6	4

Or

Available resources after completion of each process

	R1	R2	R3
Init	1	2	2
P3	3	4	3
P2	8	6	3
P1	8	6	4

Banker's Algorithm: 4 philosophers each holding his left fork

Max

$$R = \begin{vmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{vmatrix}$$

Allocation

$$C = \begin{vmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

Need

$$R - C = \begin{vmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{vmatrix}$$

Total

$$E = \begin{vmatrix} 1 & 1 & 1 & 1 & 1 \end{vmatrix}$$

Available

$$A = \begin{vmatrix} 0 & 0 & 0 & 0 & 1 \end{vmatrix}$$

Suppose we have 5 philosophers P1-P5, and 5 forks R1-R5; philosopher Pi has left fork Ri, and right fork R(i+1)%5. Philosophers P1-P4 each is holding his left fork.

Run Banker's algorithm to check if the current state is safe. If yes, give a safe sequence of process completions and fill in the table with the sequence of process completions without deadlock, and available resources after the completion of each process.

Available resources after completion of each process

	R1	R2	R3	R4	R5
	0	0	0	0	1

Banker's Algorithm: 4 philosophers each holding his left fork ANS

- Yes, current state is safe, and the only safe sequence is P4, P3, P2, P1, P5

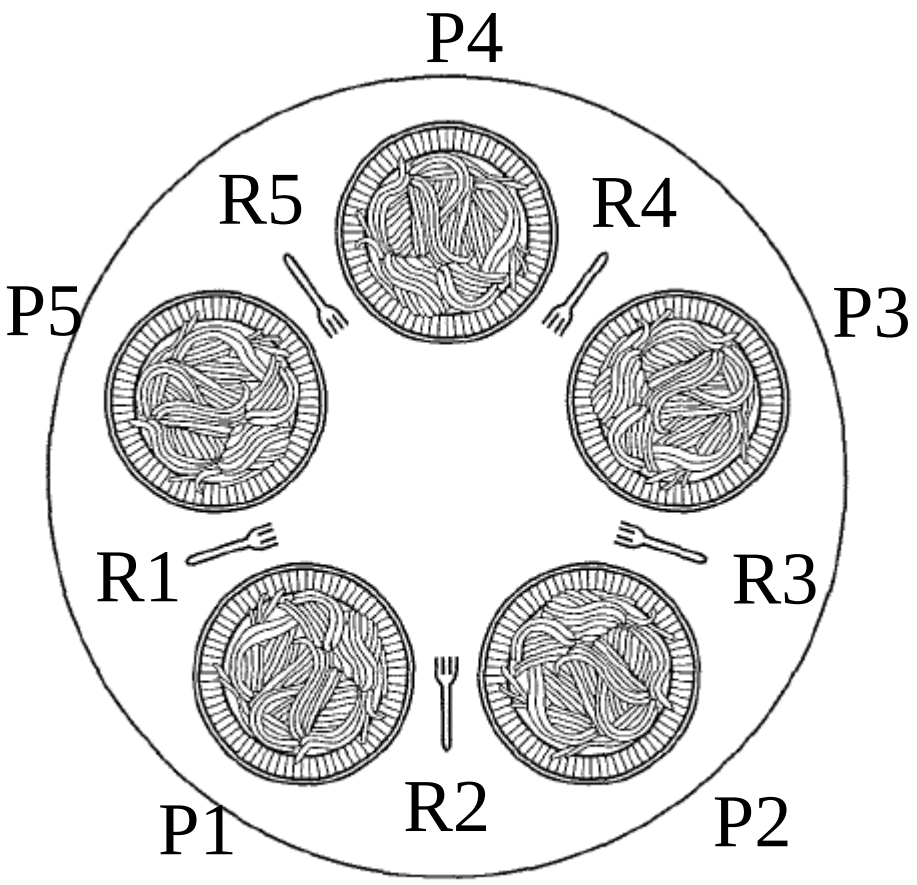


Figure 2-44. Lunch time in the Philosophy Department.

$$R - C = \text{Need} = \begin{vmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{vmatrix}$$

Available resources after completion of each process

	R1	R2	R3	R4	R5
	0	0	0	0	1
P4	0	0	0	1	1
P3	0	0	1	1	1
P2	0	1	1	1	1
P1	1	1	1	1	1
P5	1	1	1	1	1

Banker's Algorithm: 5 philosophers each holding his left fork

Max

$$R = \begin{vmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{vmatrix}$$

Allocation

$$C = \begin{vmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \boxed{1} \end{vmatrix}$$

Need

$$C = \begin{vmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{vmatrix}$$

Total

$$E = \begin{vmatrix} 1 & 1 & 1 & 1 & 1 \end{vmatrix}$$

Available

$$A = \begin{vmatrix} 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

Run Banker's algorithm to check if the current state is safe.

ANS: It is not safe, as no process can run to completion based on Need matrix and Available vector.

Multi-Armed Lawyers

- Consider a large table with identical multi-armed alien lawyers. There is a pile of chopsticks at the center of the table. In order to eat, a lawyer must have one chopstick in each hand. Assume total number of chopsticks >= number of hands of each lawyer, so at least one lawyer can eat.
- It is not a generalization of the 2-armed Dining Philosophers problem. Since the chopsticks are in a pile at center of the table, we should model them as a single resource with multiple instances, instead of multiple resources for the Dining Philosophers, where each fork (chopstick) has a fixed position in-between two philosophers. Hence the R and C matrices have a single column.

Ack: this example is taken from UC Berkeley CS162 course.

Quiz: Dining Lawyers I

- If each lawyer has 2 arms, and there is a pile of 5 chopsticks at the center of the table. Each lawyer follows the following steps:
  - (1) Pick up a chopstick
  - (2) Pick up another chopstick
  - (3) Eat
  - (4) Return both chopsticks to the pile
- Q0: Can the system be deadlocked?
- Q1: Two lawyers each grab two chopsticks and start eating. One lawyer grabs one chopstick. Is the current state safe? Check it using Banker's algorithm.
- Q2: Each lawyer grabs 1 chopstick. Is the current state safe? Check it using Banker's algorithm. Check it using Banker's algorithm.

Example: 5 Lawyers, each with 2 arms, 5 chopsticks

Initially, all chopsticks are free.

Max	Allocation
2	0
2	0
2	0
2	0
2	0

Total	Available
5	5

Q1: Two lawyers each grab two chopsticks and start eating. Is the current state safe? Check it using Banker's algorithm.

Max	Allocation	Need
2	2	0
2	2	0
2	1	1
2	0	2
2	0	2

Total	Available
5	0

Q1: ANS: Yes, current state is safe. You can run P1 first, then the remaining lawyers in any order.

Available resources after completion of each process

	R1
Init	0
P1	2
P2	4
P3	5
P4	5
P5	5

Example: 5 Lawyers, each with 2 arms, 5 chopsticks

Initially, all chopsticks are free.

Max	Allocation
2	0
2	0
2	0
2	0
2	0

Total	Available
5	5

Q2: Each lawyer grabs 1 chopstick. Is the current state safe? Check it using Banker's algorithm.

Max	Allocation	Need
2	1	1
2	1	1
2	1	1
2	1	1
2	1	1

Total	Available
5	0

Q2 ANS: No, current state is a deadlock.

Available resources after completion of each process

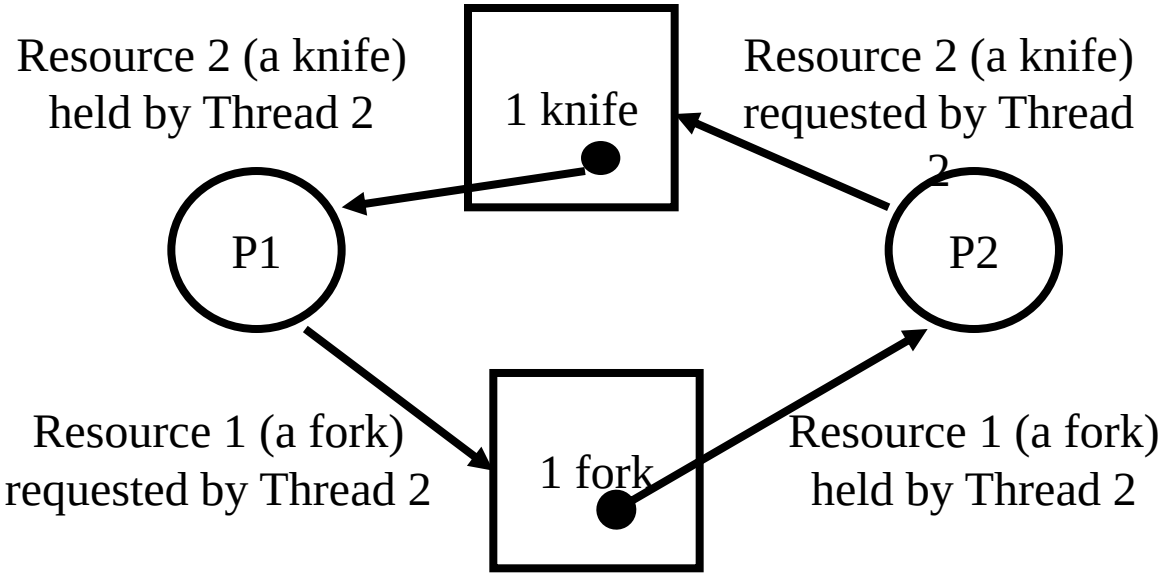
	R1
Init	0
Deadlock	

Quiz: Dining Lawyers II

- If each lawyer has 2 arms, and there is a pile of knives and forks at center of the table. Assume there are at least 1 knife and 1 fork, so at least one lawyer can eat. Each lawyer follows the following steps:
  - (1) Pick up a knife
  - (2) Pick up a fork
  - (3) Eat
  - (4) Return the knife and fork to the pile
- Q: Can the system be deadlocked?

Quiz: Dining Lawyers II Answer

- ANS: No deadlocks, since it's not possible to have circular waiting.
- All lawyers follow the same resource acquisition order:
  - 1. Knife
  - 2. Fork
- Since lawyers do not wait for resources held by others in a cyclic manner, no circular dependency forms.
- Example: If all knives are taken, lawyers without knives wait for returned knives. Those with knives either acquire forks (if available) or wait for forks to be returned. With at least one fork in the system, progress is guaranteed once utensils are released.



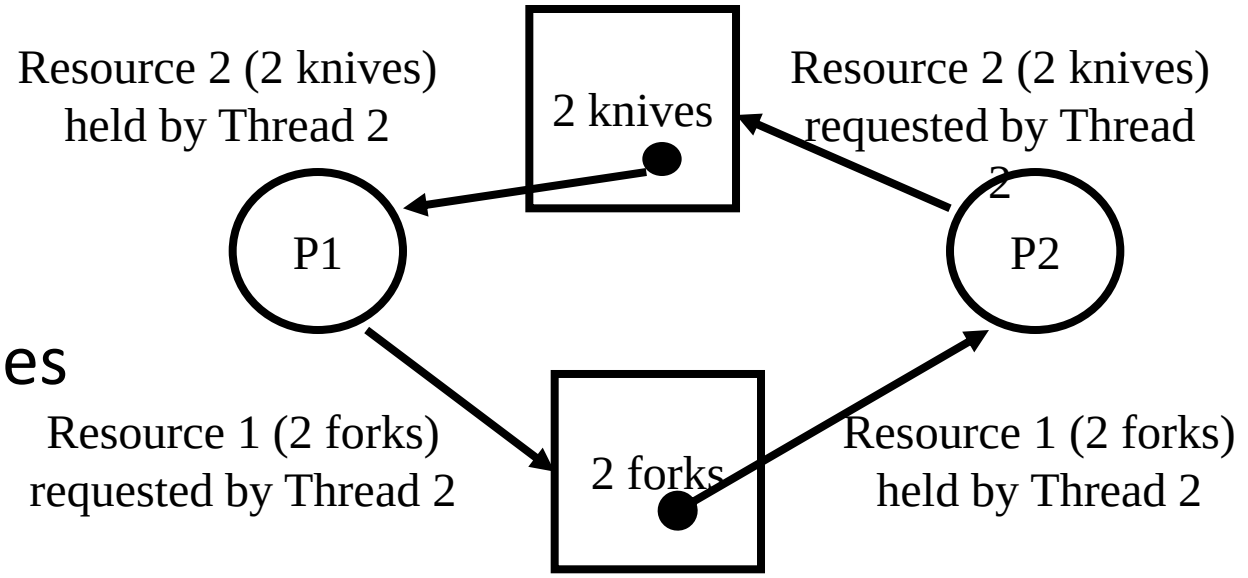
This deadlock scenario is NOT possible under the fixed resource acquisition order: 1. Knife 2. Fork

Quiz: Dining Lawyers III

- If each lawyer has 4 arms, and there is a pile of knives and forks at center of the table. Assume there are at least 2 knives and 2 forks, so at least one lawyer can eat. Each lawyer follows the following steps:
  - (1) Pick up 2 knives atomically
  - (2) Pick up 2 forks atomically
  - (3) Eat
  - (4) Return the knives and forks to the pile
- Q: Can the system be deadlocked?

Quiz: Dining Lawyers III Answer

- ANS: No deadlocks, since it's not possible to have circular waiting.
- All lawyers follow the same resource acquisition order:
  - 1. Two knives
  - 2. Two forks
- Since lawyers do not wait for resources held by others in a cyclic manner, no circular dependency forms.



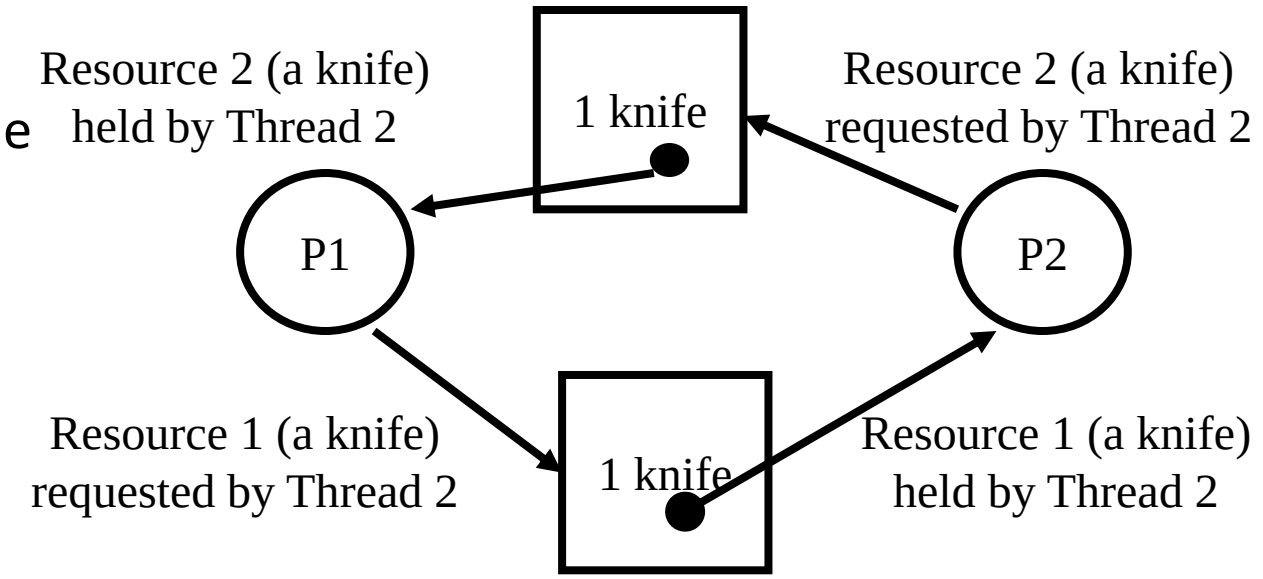
This deadlock scenario is NOT possible under the fixed resource acquisition order: 2 knives, 2 forks

Quiz: Dining Lawyers IV

- If each lawyer has 4 arms, and there is a pile of knives and forks at center of the table. Assume there are at least 2 knives and 2 forks, so at least one lawyer can eat. Each lawyer follows the following steps:
  - (1) Pick up a knife
  - (2) Pick up another knife
  - (3) Pick up a fork
  - (4) Pick up another fork
  - (5) Eat
  - (6) Return the knife and fork to the pile
- Q1: Can the system be deadlocked?
- Q2: What if each lawyer may have a different number of arms, and may request a different ratio of knives vs. forks?

Quiz: Dining Lawyers IV Answer

- Q1 ANS: Yes, since requests for each resource type (knife or fork) are not granted atomically. Need Banker's algorithm to detect (potential) deadlocks.
- Consider 2 lawyers, and a total of 2 knives and 2 forks available. If each lawyer picks up a knife, the system is deadlocked.
- Recall: "Define a total order of resources; If a thread holds certain resources, it can subsequently request only resources that follow the types of held resources in the total order." Since all knives are the same and not numbered, you cannot form a total order like "request knife 1 before knife 2". If a lawyer requests a knife while holding a knife, there may be circular waiting.
- Q2 ANS: The solution is basically the same, except implementation of Banker's algorithm needs to take into account this factor, e.g., have an array of variables NumArms[] instead of a single variable NumArms, and so on.



This deadlock scenario is possible under the resource acquisition order: a knife, a knife, a fork, a fork.

Example: 2 lawyers, each with 4 arms, 2 knives and 2 forks

Initially, all knives and forks are free.

Max	
2	2
2	2

Allocation	
0	0
0	0

Total	
2	2

Available	
2	2

Each lawyer grabs 1 knife.

Max	
2	2
2	2

Allocation	
1	0
1	0

Need	
1	2
1	2

Total	
2	2

Available	
0	2

Available resources after completion of each process

	K	F
Init	0	2
Deadlock		

The result is a deadlock state, as no process can run to completion based on Need matrix and Available vector.