
Resource Access Protocols

Prof. Dr. Jian-Jia Chen

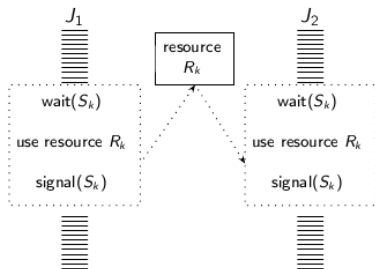
LS 12, TU Dortmund

29, Oct. 2019

Why do We Have to Worry about Resource Sharing?

Shared Resources:

- Data structures, variables, main memory area, file, set of registers, I/O unit, the processor, etc.
- Mutual exclusion, critical section
 - When a job enters the critical section of a shared resource, the accesses to the shared resource from other jobs are *blocked*.

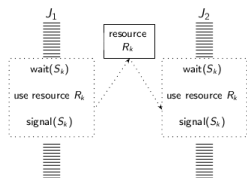




Quiz: Binary Semaphore vs Mutex

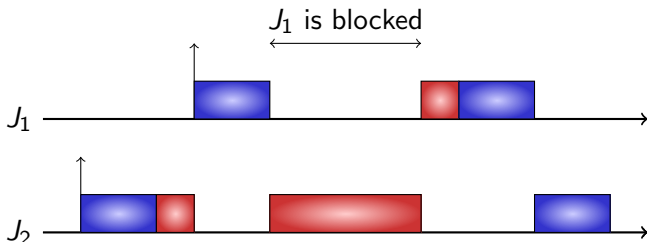
Are they the same or different?

Priority Inversion

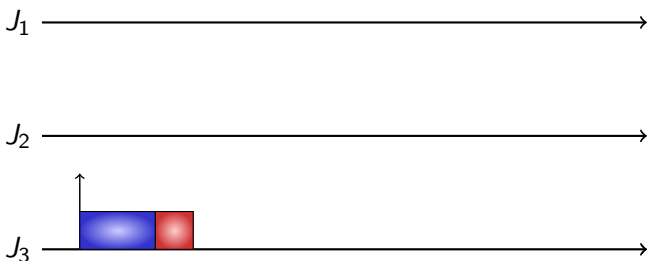
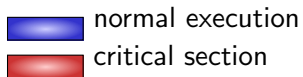
Priority Inversion: A higher priority job is *blocked* by a lower-priority job and *indirectly preempted* by a lower priority task





-  normal execution
-  critical section

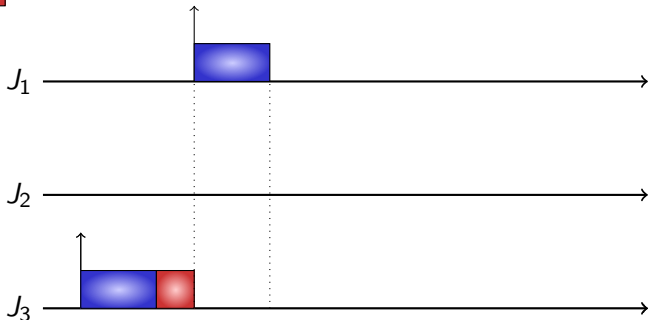


Priority Inversion: An Example

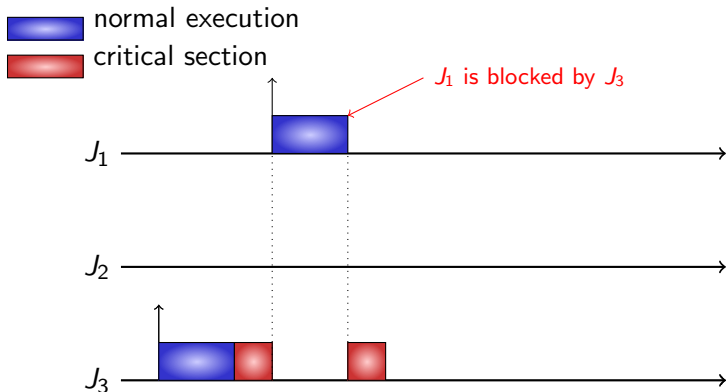


Priority Inversion: An Example

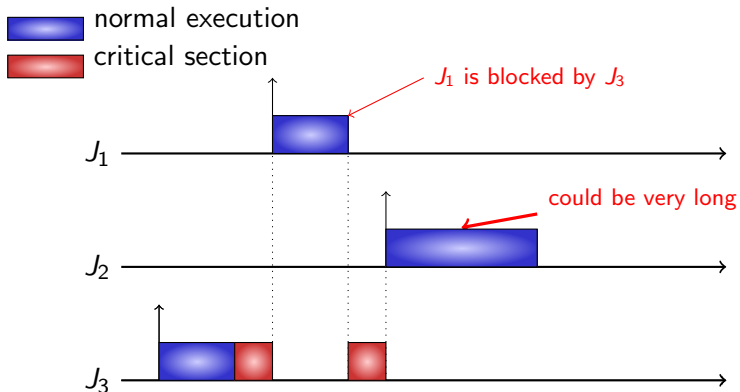
 normal execution
 critical section



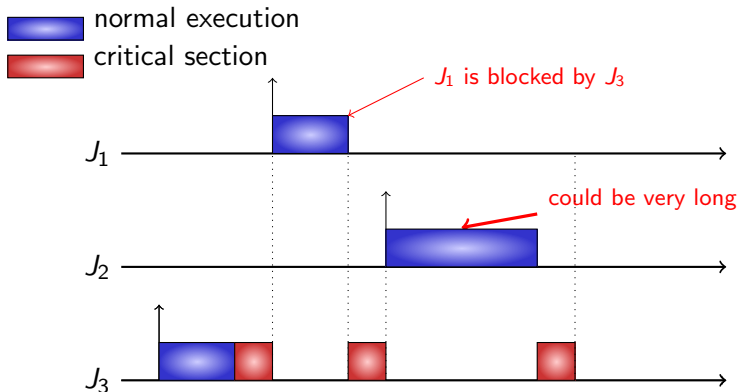
Priority Inversion: An Example



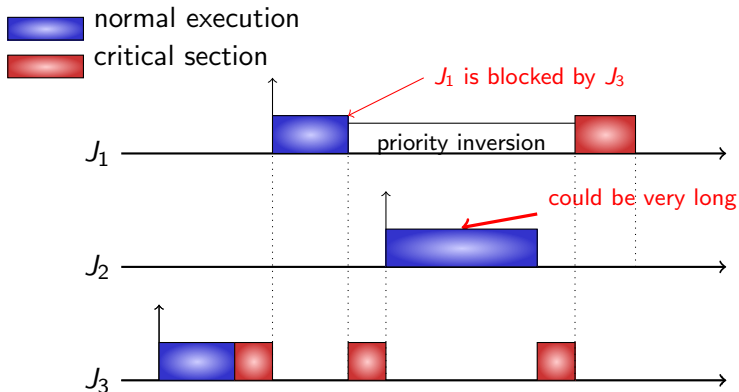
Priority Inversion: An Example



Priority Inversion: An Example





Priority Inversion: An Example

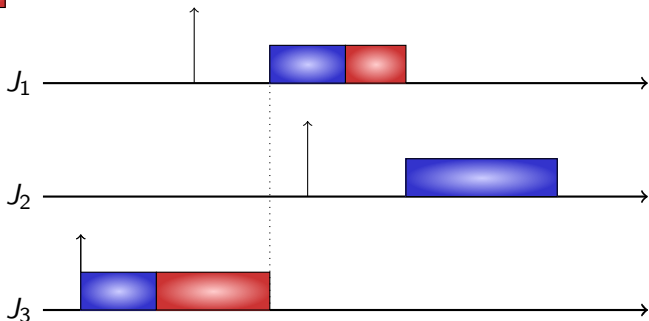


Naïve Solution for Priority Inversion

Disallow preemption during critical sections or set to the highest-priority during priority inversion

- It is simple
- But, it creates unnecessary blocking, as unrelated tasks may be blocked

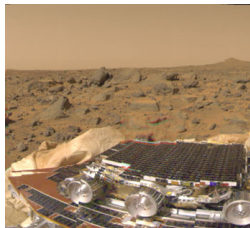
 normal execution
 critical section



Case Study: MARS Pathfinder Problem (1)

A few days into the mission.....

Not long after Pathfinder started gathering meteorological data, the spacecraft began experiencing total system resets, each resulting in losses of data.



Case Study: MARS Pathfinder Problem (2)

“VxWorks provides preemptive priority scheduling of threads. Tasks on the Pathfinder spacecraft were executed as threads with priorities that were assigned in the usual manner reflecting the relative urgency of these tasks.”

“Pathfinder contained an **information bus**, which you can think of as a shared memory area used for passing information between different components of the spacecraft.”

A bus management task ran frequently with high priority to move certain kinds of data in and out of the information bus. Access to the bus was synchronized with mutual exclusion locks (mutexes).

- The meteorological data gathering task ran as an infrequent, low priority thread, When publishing its data, it would acquire a mutex, do writes to the bus, and release the mutex.
- It also had a communications task that ran with medium priority.

Case Study: MARS Pathfinder Problem (3)

high priority	medium priority	low priority
data retrieval from memory	communication task	data collection

“Most of the time this combination worked fine. However, very infrequently it was possible for an interrupt to occur that caused the (medium priority) communications task to be scheduled during the short interval while the (high priority) information bus thread was blocked waiting for the (low priority) meteorological data thread. In this case, the long-running communications task, having higher priority than the meteorological task, would prevent it from running, consequently preventing the blocked information bus task from running. After some time had passed, a watchdog timer would go off, notice that the data bus task had not been executed for some time, conclude that something had gone drastically wrong, and initiate a total system reset. **This scenario is a classic case of priority inversion.**”

Resource Access Protocols

Priority Inheritance and Priority Ceiling Protocols

Some Crazy Ideas (Only for Reference)

Resource Access Protocols

- Spirit
 - Modify (increase) the priority of those tasks/jobs that cause blocking.
 - When a job J_j blocks one or more higher-priority jobs, it temporarily assumes a higher priority.
- Methods (**The details about fixed-priority and dynamic-priority scheduling will be given later in the lecture.**)
 - Priority Inheritance Protocol (PIP), for fixed-priority scheduling
 - Priority Ceiling Protocol (PCP), for fixed-priority scheduling
 - Stack Resource Policy (SRP), for both fixed- and dynamic-priority scheduling
 - others.....

Priority Inheritance Protocol (PIP)

When a lower-priority job J_j blocks a higher-priority job, the priority of job J_j is *promoted* to the priority level of highest-priority job that job J_j blocks.

Priority Inheritance Protocol (PIP)

When a lower-priority job J_j blocks a higher-priority job, the priority of job J_j is *promoted* to the priority level of highest-priority job that job J_j blocks.

For example, if the priority order is $J_1 > J_2 > J_3 > J_4 > J_5$,

- When job J_4 blocks jobs J_2 and J_3 , the priority of J_4 is promoted to the priority level of J_2 .
- When job J_5 blocks jobs J_1 and J_3 , the priority of J_5 is promoted to the priority level of J_1 .

Priority Inheritance Protocol (PIP)

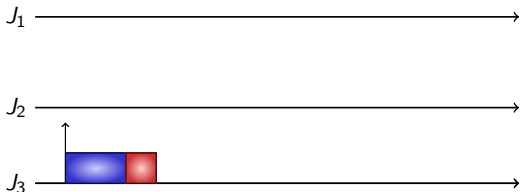
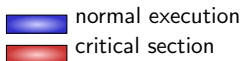
When a lower-priority job J_j blocks a higher-priority job, the priority of job J_j is *promoted* to the priority level of highest-priority job that job J_j blocks.

For example, if the priority order is $J_1 > J_2 > J_3 > J_4 > J_5$,

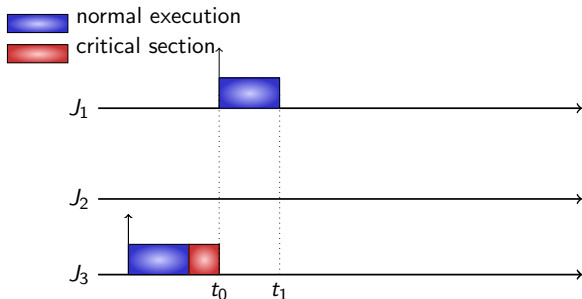
- When job J_4 blocks jobs J_2 and J_3 , the priority of J_4 is promoted to the priority level of J_2 .
- When job J_5 blocks jobs J_1 and J_3 , the priority of J_5 is promoted to the priority level of J_1 .

Priority inheritance solved the Mars Pathfinder problem: the VxWorks operating system used in the pathfinder implements a flag for the calls to mutex primitives. This flag allows priority inheritance to be set to **on**. When the software was shipped, it was set to **off**.

Example of PIP

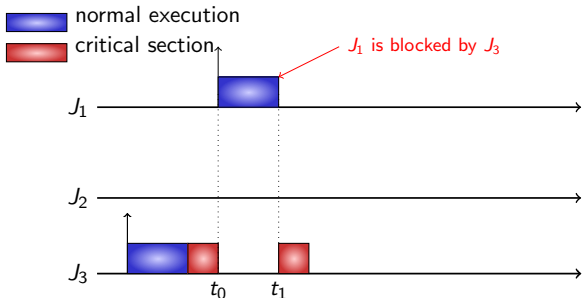


Example of PIP



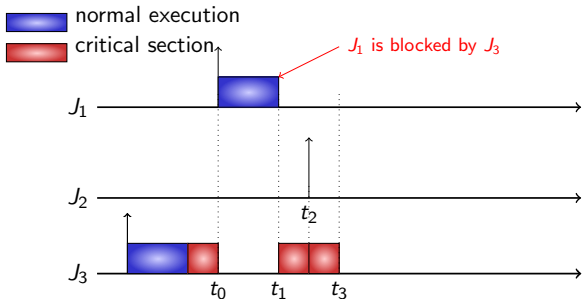
- t_0 : J_1 arrives and preempts J_3 , since J_1 does not want to enter the critical section

Example of PIP



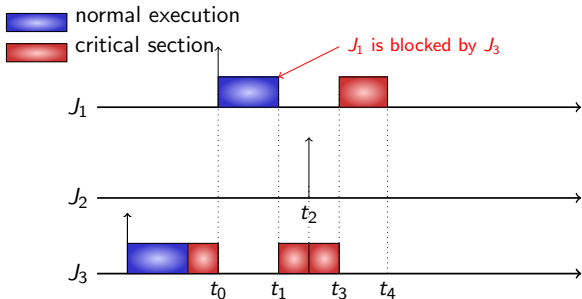
- t_0 : J_1 arrives and preempts J_3 , since J_1 does not want to enter the critical section
- t_1 : J_1 locks the semaphore and tries to enter the critical section. J_1 is blocked by J_3 , and J_3 inherits J_1 's priority

Example of PIP



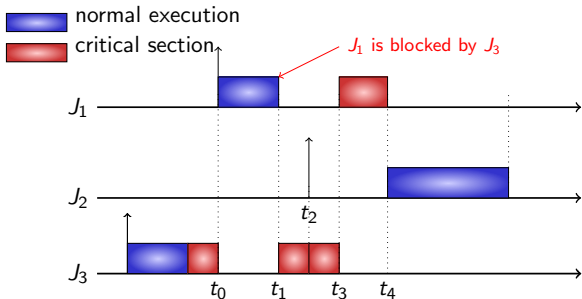
- t_0 : J_1 arrives and preempts J_3 , since J_1 does not want to enter the critical section
- t_1 : J_1 locks the semaphore and tries to enter the critical section. J_1 is blocked by J_3 , and J_3 inherits J_1 's priority
- t_2 : J_2 arrives and has a lower priority than J_3 , since J_3 inherited J_1 's priority.
- t_3 : J_3 leaves its critical section, and J_1 now preempts J_3 .

Example of PIP



- t_0 : J_1 arrives and preempts J_3 , since J_1 does not want to enter the critical section
- t_1 : J_1 locks the semaphore and tries to enter the critical section. J_1 is blocked by J_3 , and J_3 inherits J_1 's priority
- t_2 : J_2 arrives and has a lower priority than J_3 , since J_3 inherited J_1 's priority.
- t_3 : J_3 leaves its critical section, and J_1 now preempts J_3 .
- t_4 : J_1 finishes, and J_2 is the highest-priority task.

Example of PIP

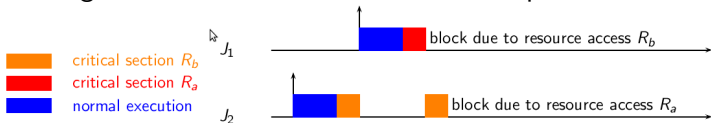


- t_0 : J_1 arrives and preempts J_3 , since J_1 does not want to enter the critical section
- t_1 : J_1 locks the semaphore and tries to enter the critical section. J_1 is blocked by J_3 , and J_3 inherits J_1 's priority
- t_2 : J_2 arrives and has a lower priority than J_3 , since J_3 inherited J_1 's priority.
- t_3 : J_3 leaves its critical section, and J_1 now preempts J_3 .
- t_4 : J_1 finishes, and J_2 is the highest-priority task.

Problem of PIP

Problems of PIP

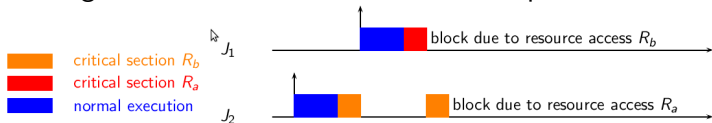
- PIP might cause *deadlock* if there are multiple resources



Problem of PIP

Problems of PIP

- PIP might cause *deadlock* if there are multiple resources



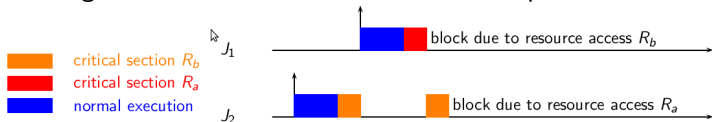
However, if the resource accesses for a task are **properly nested**, then some analysis is still possible.

- This will not be covered in the lecture of Embedded Systems.

Problem of PIP

Problems of PIP

- PIP might cause *deadlock* if there are multiple resources



However, if the resource accesses for a task are **properly nested**, then some analysis is still possible.

- This will not be covered in the lecture of Embedded Systems.

Ongoing debate about problems with the protocol:

- Victor Yodaiken: Against Priority Inheritance, Sept. 2004, http://www.fsmlabs.com/resources/white_papers/priority-inheritance

Resource Access Protocols

Priority Inheritance and Priority Ceiling Protocols

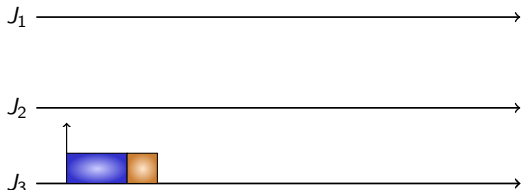
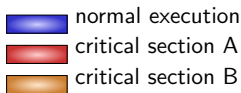
Some Crazy Ideas (Only for Reference)

Priority Ceiling Protocol (PCP)

- Two key assumptions:
 - The assigned priorities of all jobs are fixed.
 - The resources required by all jobs are known a priori before the execution of any job begins.
- Definition: The *priority ceiling* of a resource R is the highest priority of all the jobs that require R , and is denoted $\Pi(R)$.
- Definition: The *current priority* of a job J at time t is denoted $\pi(t, J)$, initialized to the job priority level when J is released. (smaller means higher priority)
- Definition: The *current priority ceiling* $\Pi'(t)$ of the system is equal to the highest priority ceiling of the resources currently in use at time t , or Ω if no resources are currently in use (Ω is a priority lower than any real priority).
- Use the priority ceiling to decide whether a higher priority can allocate a resource or not.

- ① Scheduling Rule
 - Every job J is scheduled based on the current priority $\pi(t, J)$.
- ② Allocation Rule: Whenever a job J requests a resource R at time t , one of the following two conditions occurs:
 - R is held by another job and J becomes blocked.
 - R is free:
 - If J 's priority $\pi(t, J)$ is higher than the current priority ceiling $\Pi'(t)$, R is allocated to J .
 - Otherwise, only if J is the job holding the resource(s) whose priority ceiling equals $\Pi'(t)$, R is allocated to J
 - Otherwise, J becomes blocked.
- ③ Priority-inheritance Rule: When J becomes blocked, the job J_I that blocks J inherits the current priority $\pi(t, J)$ of J . J_I executes at its inherited priority until it releases every resource whose priority ceiling is $\geq \pi(t, J)$ (or until it inherits an even higher priority); at that time, the priority of J_I returns to its priority $\pi(t', J_I)$ at the time t' when it was granted the resources.

Example of PCP



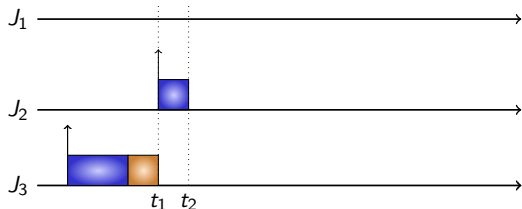
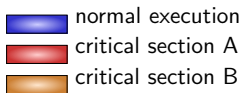
Three jobs with two semaphores A and B.

- J_1 : normal execution, wait A, signal A, normal execution
- J_2 : normal execution, wait A, wait B, signal B, signal A, normal execution
- J_3 : normal execution, wait B, signal B, normal execution

priority ceilings:

- A: priority level 1
- B: priority level 2

Example of PCP



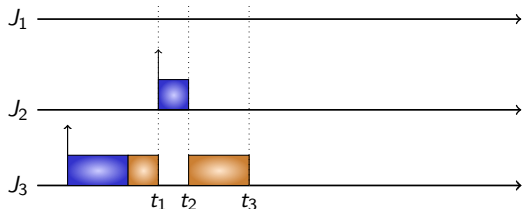
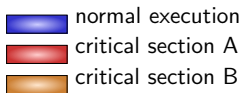
priority ceilings:

- A: priority level 1
- B: priority level 2

Three jobs with two semaphores A and B.

- J_1 : normal execution, wait A, signal A, normal execution
- J_2 : normal execution, wait A, wait B, signal B, signal A, normal execution
- J_3 : normal execution, wait B, signal B, normal execution

Example of PCP



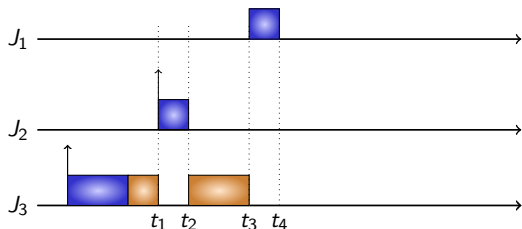
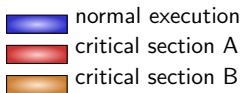
Three jobs with two semaphores A and B.

- J_1 : normal execution, wait A, signal A, normal execution
- J_2 : normal execution, wait A, wait B, signal B, signal A, normal execution
- J_3 : normal execution, wait B, signal B, normal execution

priority ceilings:

- A: priority level 1
- B: priority level 2

Example of PCP



priority ceilings:

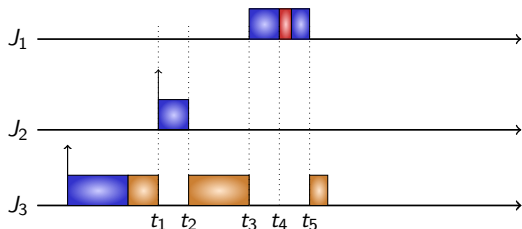
- A: priority level 1
- B: priority level 2

Three jobs with two semaphores A and B.

- J_1 : normal execution, wait A, signal A, normal execution
- J_2 : normal execution, wait A, wait B, signal B, signal A, normal execution
- J_3 : normal execution, wait B, signal B, normal execution

Example of PCP

- normal execution
- critical section A
- critical section B



priority ceilings:

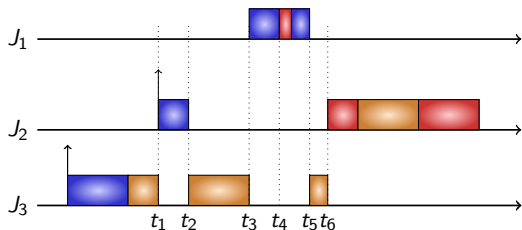
- A: priority level 1
- B: priority level 2

Three jobs with two semaphores A and B.

- J_1 : normal execution, wait A, signal A, normal execution
- J_2 : normal execution, wait A, wait B, signal B, signal A, normal execution
- J_3 : normal execution, wait B, signal B, normal execution

Example of PCP

- normal execution
- critical section A
- critical section B



priority ceilings:

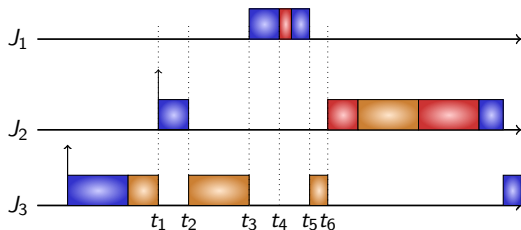
- A: priority level 1
- B: priority level 2

Three jobs with two semaphores A and B.

- J_1 : normal execution, wait A, signal A, normal execution
- J_2 : normal execution, wait A, wait B, signal B, signal A, normal execution
- J_3 : normal execution, wait B, signal B, normal execution

Example of PCP

- normal execution
- critical section A
- critical section B





priority ceilings:



- A: priority level 1
- B: priority level 2

Three jobs with two semaphores A and B.

- J_1 : normal execution, wait A, signal A, normal execution
- J_2 : normal execution, wait A, wait B, signal B, signal A, normal execution
- J_3 : normal execution, wait B, signal B, normal execution

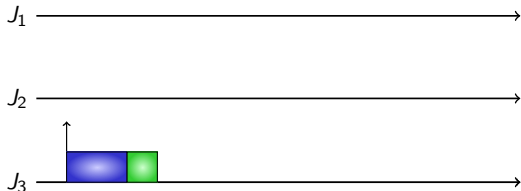
Example of PCP (2nd)

 normal execution
 critical section B

 critical section A
 critical section C

priority ceilings:

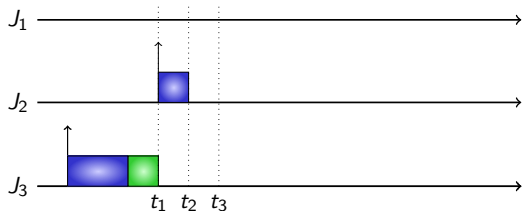
- A: priority level 1
- B: priority level 1
- C: priority level 2



Three jobs with three semaphores A, B, and C.

- J_1 : normal execution, wait A, signal A, normal execution, wait B, signal B, normal execution
- J_2 : normal execution, wait C, signal C, normal execution
- J_3 : normal execution, wait C, wait B, signal B, signal C, normal execution

Example of PCP (2nd)



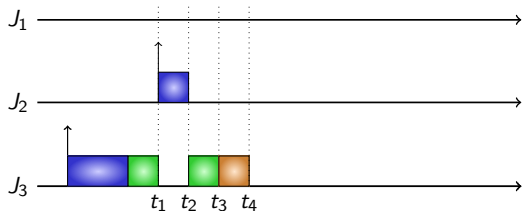
priority ceilings:

- A: priority level 1
- B: priority level 1
- C: priority level 2

Three jobs with three semaphores A, B, and C.

- J_1 : normal execution, wait A, signal A, normal execution, wait B, signal B, normal execution
- J_2 : normal execution, wait C, signal C, normal execution
- J_3 : normal execution, wait C, wait B, signal B, signal C, normal execution

Example of PCP (2nd)



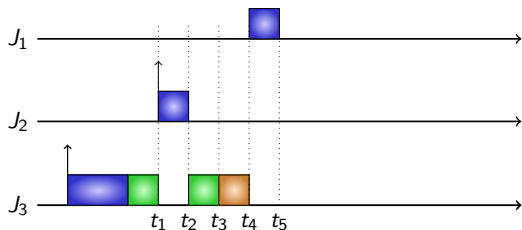
priority ceilings:

- A: priority level 1
- B: priority level 1
- C: priority level 2

Three jobs with three semaphores A, B, and C.

- J_1 : normal execution, wait A, signal A, normal execution, wait B, signal B, normal execution
- J_2 : normal execution, wait C, signal C, normal execution
- J_3 : normal execution, wait C, wait B, signal B, signal C, normal execution

Example of PCP (2nd)



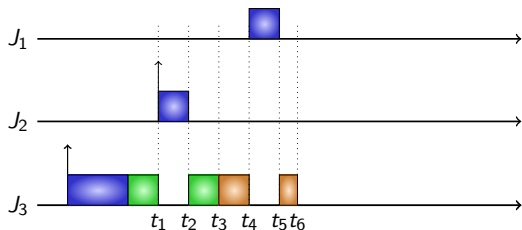
priority ceilings:

- A: priority level 1
- B: priority level 1
- C: priority level 2

Three jobs with three semaphores A, B, and C.

- J_1 : normal execution, wait A, signal A, normal execution, wait B, signal B, normal execution
- J_2 : normal execution, wait C, signal C, normal execution
- J_3 : normal execution, wait C, wait B, signal B, signal C, normal execution

Example of PCP (2nd)



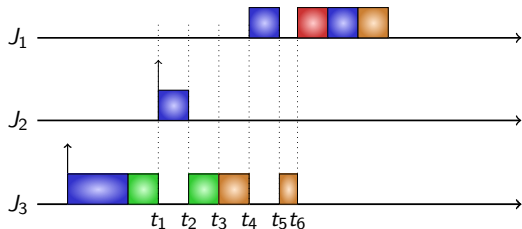
priority ceilings:

- A: priority level 1
- B: priority level 1
- C: priority level 2

Three jobs with three semaphores A, B, and C.

- J_1 : normal execution, wait A, signal A, normal execution, wait B, signal B, normal execution
- J_2 : normal execution, wait C, signal C, normal execution
- J_3 : normal execution, wait C, wait B, signal B, signal C, normal execution

Example of PCP (2nd)



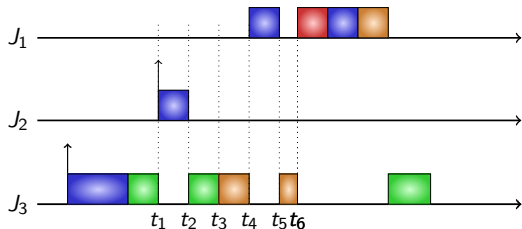
priority ceilings:

- A: priority level 1
- B: priority level 1
- C: priority level 2

Three jobs with three semaphores A, B, and C.

- J_1 : normal execution, wait A, signal A, normal execution, wait B, signal B, normal execution
- J_2 : normal execution, wait C, signal C, normal execution
- J_3 : normal execution, wait C, wait B, signal B, signal C, normal execution

Example of PCP (2nd)



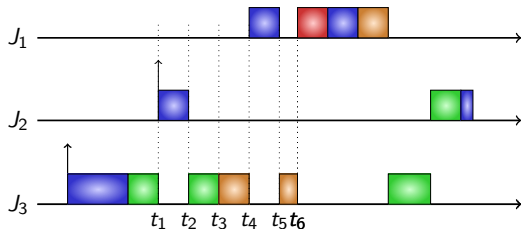
priority ceilings:

- A: priority level 1
- B: priority level 1
- C: priority level 2

Three jobs with three semaphores A, B, and C.

- J_1 : normal execution, wait A, signal A, normal execution, wait B, signal B, normal execution
- J_2 : normal execution, wait C, signal C, normal execution
- J_3 : normal execution, wait C, wait B, signal B, signal C, normal execution

Example of PCP (2nd)



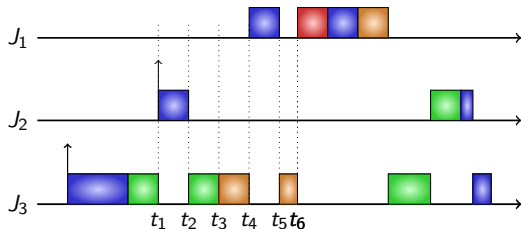
priority ceilings:

- A: priority level 1
- B: priority level 1
- C: priority level 2

Three jobs with three semaphores A, B, and C.

- J_1 : normal execution, wait A, signal A, normal execution, wait B, signal B, normal execution
- J_2 : normal execution, wait C, signal C, normal execution
- J_3 : normal execution, wait C, wait B, signal B, signal C, normal execution

Example of PCP (2nd)



priority ceilings:

- A: priority level 1
- B: priority level 1
- C: priority level 2

Three jobs with three semaphores A, B, and C.

- J_1 : normal execution, wait A, signal A, normal execution, wait B, signal B, normal execution
- J_2 : normal execution, wait C, signal C, normal execution
- J_3 : normal execution, wait C, wait B, signal B, signal C, normal execution

Properties of PCP

Theorem

When the resource accesses of a system of preemptive, priority-driven jobs on one processor are controlled by the PCP, deadlock can never occur.

Theorem

When the resource accesses of a system of preemptive, priority-driven jobs on one processor are controlled by the PCP, a job can be blocked for at most the duration of *one* critical section.

Resource Access Protocols

Priority Inheritance and Priority Ceiling Protocols

Some Crazy Ideas (Only for Reference)

Critical Sections - Some Recalls

- Recall the definition of critical sections in the textbook “Operating System Concepts” by Peter B. Galvin et al. (in fact Peterson’s algorithm invented in 1981)
 - **Progress:** If no process is executing in its critical section and some processes wish to enter their critical sections, then only those processes that are not executing in their remainder section can participate in deciding which will enter its critical section next, and this selection cannot be postponed indefinitely.
 - **Bounded waiting:** There exists a bound, or limit, on the number of times other processes are allowed to enter their critical sections after a process has made request to enter its critical section and before that request is granted.
- In any existing resource access protocols, there is no need to postpone the execution of a critical section to make the processor idle for a while.

An Example

Task	$C_{i,1}$	A_i	$C_{i,2}$	$T_i = D_i$
τ_1	ϵ	$1 - 2\epsilon$	ϵ	5
τ_2	ϵ	$1 - 2\epsilon$	$3 + \epsilon$	10
τ_3	4	8	6	20

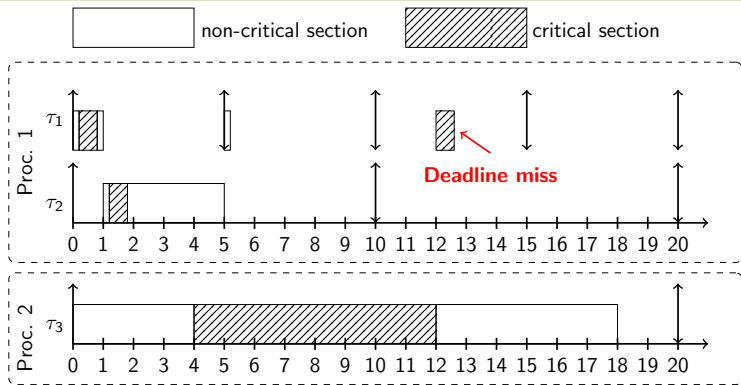
Table: An example task set, where $0 < \epsilon < 0.5$.

We consider three periodic tasks:

- T_i is the period of task τ_i
- D_i is the relative deadline of task τ_i
- $C_{i,1}$ is the execution time of the first non-critical section of task τ_i
- $C_{i,2}$ is the execution time of the second non-critical section of task τ_i
- A_i is the execution time of the **only** critical section of task τ_i

They share a mutex lock and start the first execution at time 0.

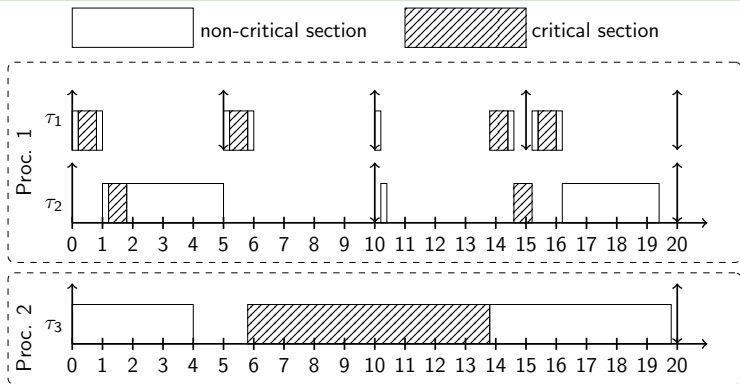
Example Schedule of Locking Protocols



An example schedule on 2 processors, one executes τ_1 and τ_2 and one executes τ_3 .

- The execution of the critical section of task τ_3 , starting at time 4 makes the second job of task τ_1 miss its deadline

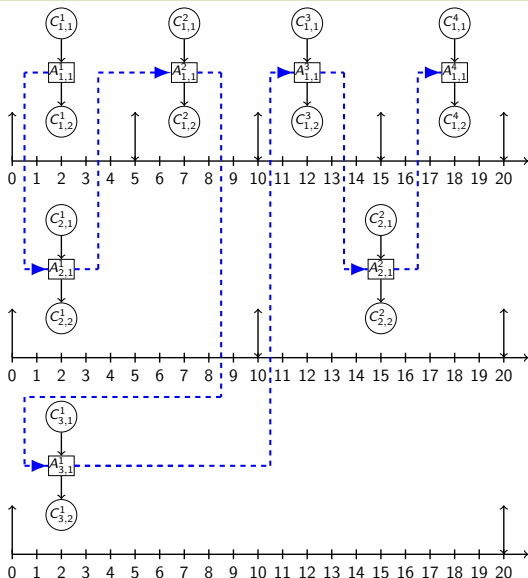
Make Unnecessary Idles



In this schedule, all of them can meet their deadlines.

- Processor 2 idles and postpones the critical section of task τ_3 .
- This seems to violate the **progress** property, but has a better result for this example.

Precedence Constraints Instead of Locking?



Remarks

Pros of using precedence constraints:

- The sequence of the jobs entering the critical sections is defined by the given precedence constraints
- Schedules can be planned offline (*will talk later*)
- The system is determinate!!!!
- Easier to implement by removing the complicated priority inheritance and locking mechanisms

Cons of using precedence constraints:

- Constructing the task graph to represent the precedence constraints can be complicated and the graph can be large
- No dynamics of job arrivals can be allowed
- Limited to scenarios where the critical sections are well-defined and fixed