Shared Resources and Blocking in Real-Time Systems

Giuseppe Lipari
`http://feanor.sssup.it/~lipari`

Scuola Superiore Sant'Anna – Pisa

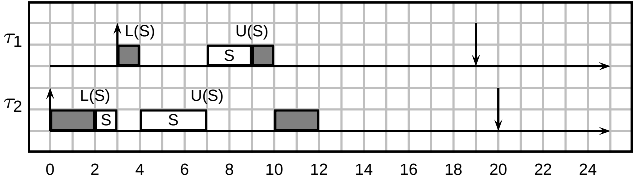
October 10, 2008

Outline

1. Priority inversion
2. Priority Inheritance Protocol
3. Priority Ceiling
4. Stack Resouce Policy

Blocking and priority inversion

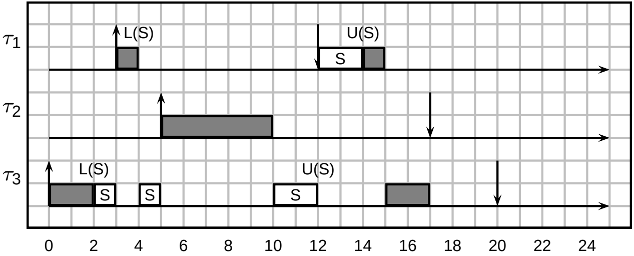
- A blocking condition happens when a high priority tasks wants to access a resource that is held by a lower priority task.
- Consider the following example, where $p_1 > p_2$.



- From time 4 to 7, task τ_1 is blocked by a lower priority taskτ_2; this is a *priority inversion*.
- Priority inversion is not avoidable; in fact, τ_1 must wait for τ_2 to leave the critical section.
- However, in some cases, the priority inversion could be too large.

Example of priority inversion

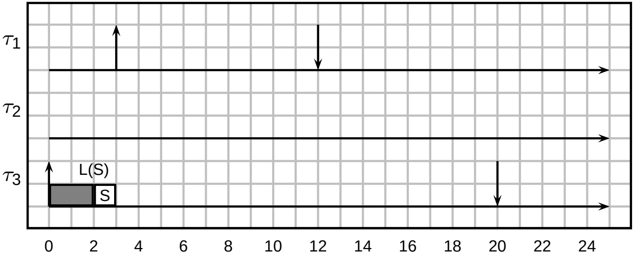
- Consider the following example, with $p_1 > p_2 > p_3$.



- This time the priority inversion is very large: from 4 to 12.
- The problem is that, while τ_1 is blocked, τ_2 arrives and preempt τ_3 before it can leave the critical section.
- If there are other medium priority tasks, they could preempt τ_3 as well.
- Potentially, the priority inversion could be unbounded!

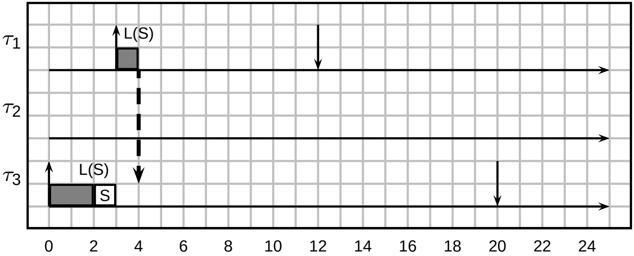
Example

- In the previous example:



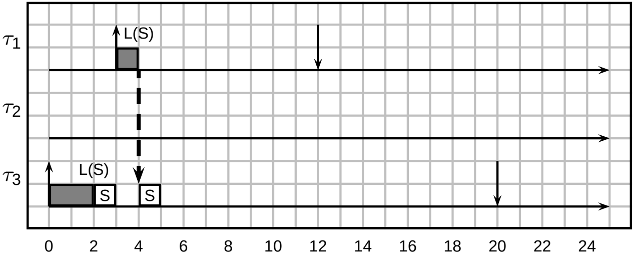
Example

- In the previous example:



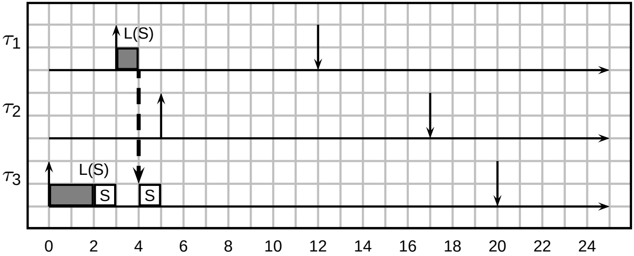
Example

- In the previous example:



Example

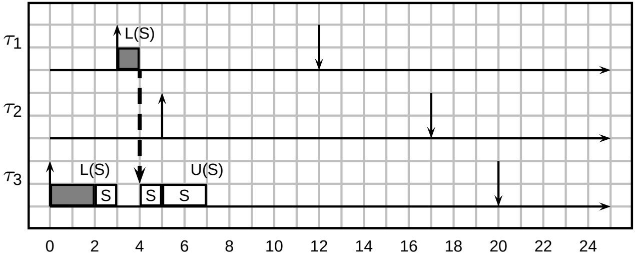
- In the previous example:



- Task τ_3 inherits the priority of τ_1

Example

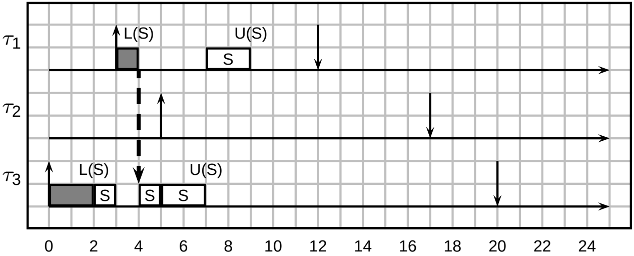
- In the previous example:



- Task τ_3 inherits the priority of τ_1
- Task τ_2 cannot preempt τ_3 ($p_2 < p_1$)

Example

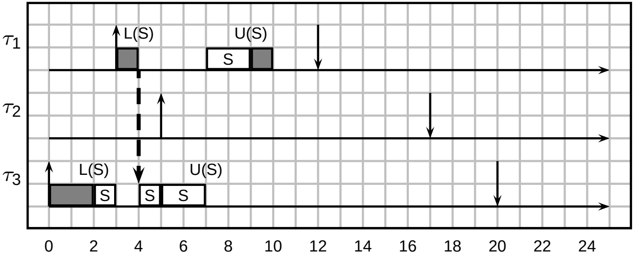
- In the previous example:



- Task τ_3 inherits the priority of τ_1
- Task τ_2 cannot preempt τ_3 ($p_2 < p_1$)

Example

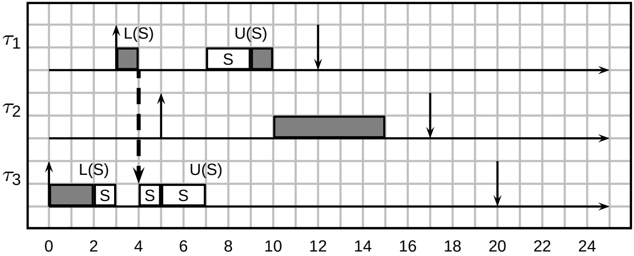
- In the previous example:



- Task τ_3 inherits the priority of τ_1
- Task τ_2 cannot preempt τ_3 ($p_2 < p_1$)

Example

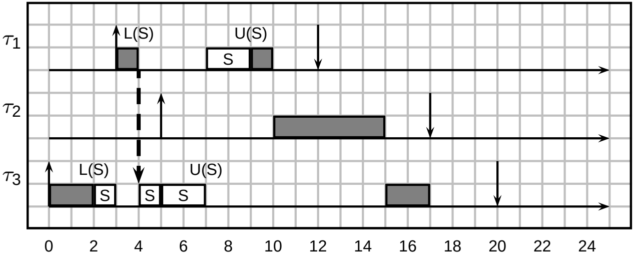
- In the previous example:



- Task τ_3 inherits the priority of τ_1
- Task τ_2 cannot preempt τ_3 ($p_2 < p_1$)

Example

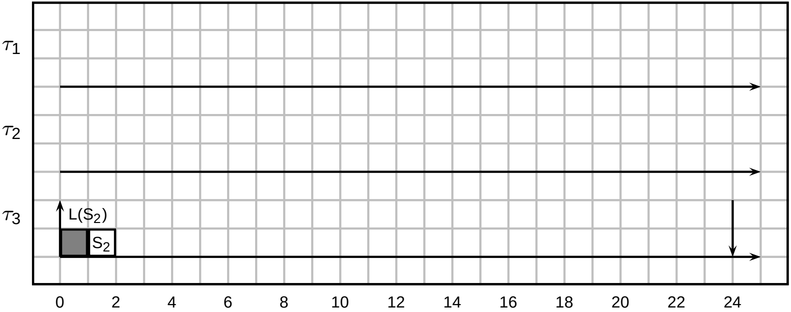
- In the previous example:



- Task τ_3 inherits the priority of τ_1
- Task τ_2 cannot preempt τ_3 ($p_2 < p_1$)

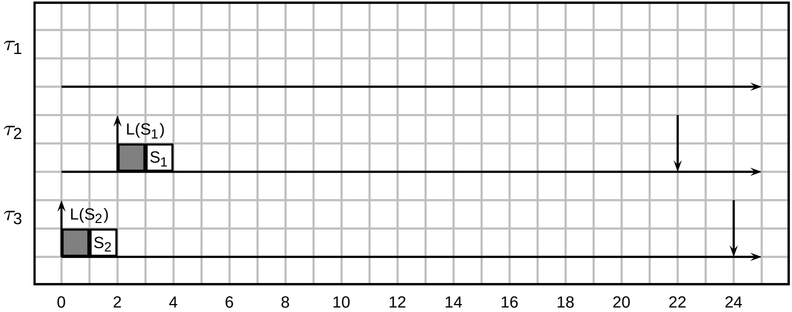
Multiple inheritance

- Task τ_1 uses resource S_1; Task τ_2 uses S_1 and S_2 nested inside S_1; Task τ_3 uses only S_2.
- $p_1 > p_2 > p_3$;



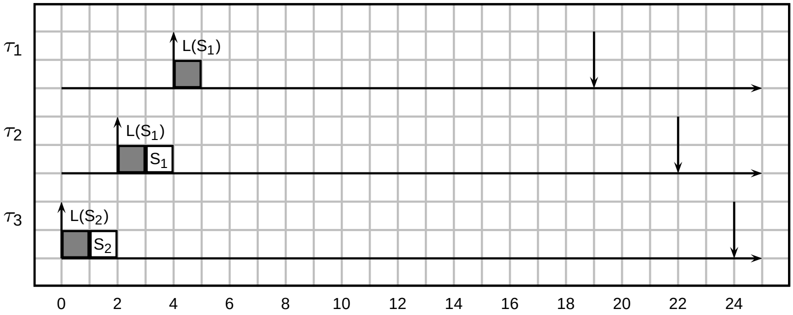
Multiple inheritance

- Task τ_1 uses resource S_1; Task τ_2 uses S_1 and S_2 nested inside S_1; Task τ_3 uses only S_2.
- $p_1 > p_2 > p_3$;



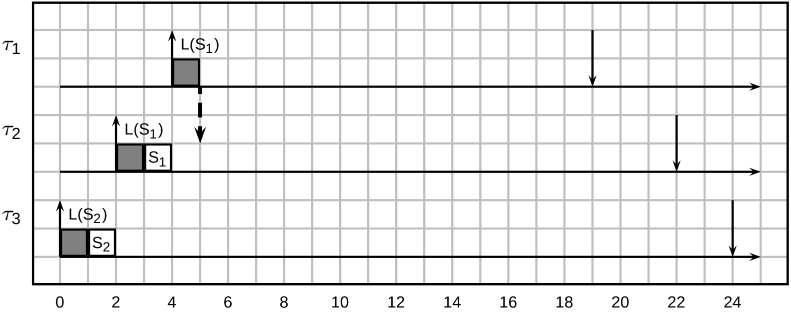
Multiple inheritance

- Task τ_1 uses resource S_1; Task τ_2 uses S_1 and S_2 nested inside S_1; Task τ_3 uses only S_2.
- $p_1 > p_2 > p_3$;



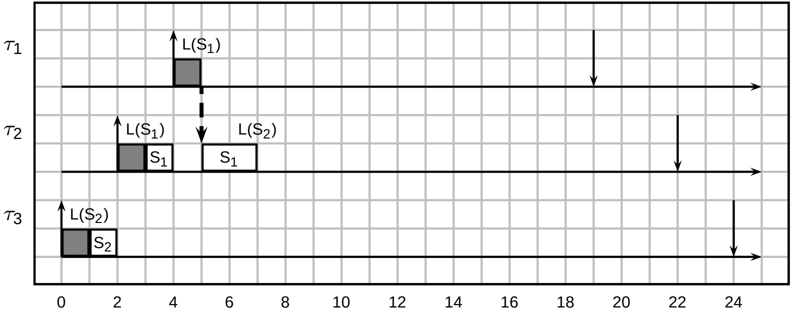
Multiple inheritance

- Task τ_1 uses resource S_1; Task τ_2 uses S_1 and S_2 nested inside S_1; Task τ_3 uses only S_2.
- $p_1 > p_2 > p_3$;



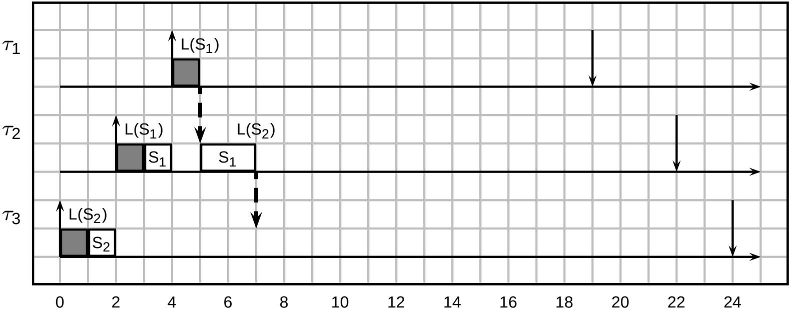
Multiple inheritance

- Task τ_1 uses resource S_1; Task τ_2 uses S_1 and S_2 nested inside S_1; Task τ_3 uses only S_2.
- $p_1 > p_2 > p_3$;



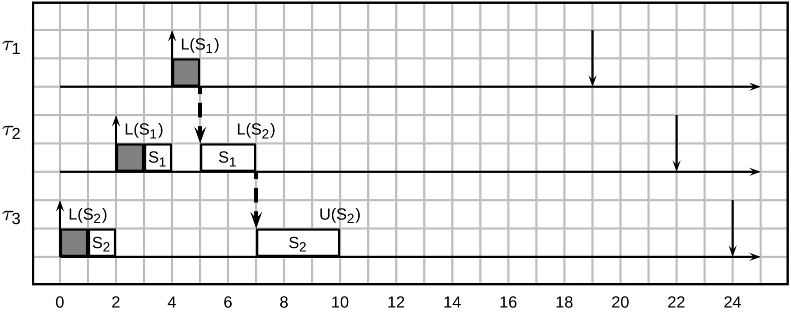
Multiple inheritance

- Task τ_1 uses resource S_1; Task τ_2 uses S_1 and S_2 nested inside S_1; Task τ_3 uses only S_2.
- $p_1 > p_2 > p_3$;



Multiple inheritance

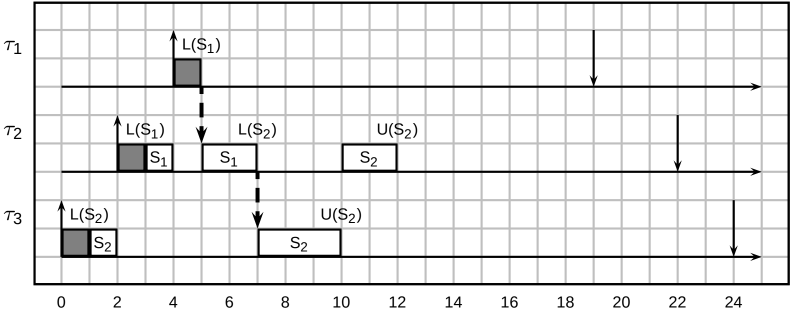
- Task τ_1 uses resource S_1; Task τ_2 uses S_1 and S_2 nested inside S_1; Task τ_3 uses only S_2.
- $p_1 > p_2 > p_3$;



- At time $t = 7$ task τ_3 inherits the priority of τ_2, which at time 5 had inherited the priority of τ_1. Hence, the priority of τ_3 is p_1.

Multiple inheritance

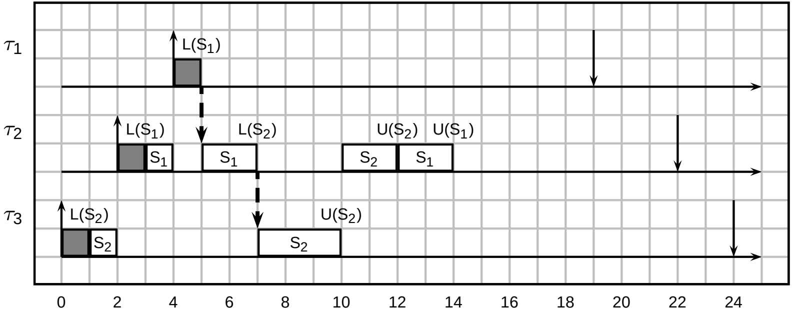
- Task τ_1 uses resource S_1; Task τ_2 uses S_1 and S_2 nested inside S_1; Task τ_3 uses only S_2.
- $p_1 > p_2 > p_3$;



- At time $t = 7$ task τ_3 inherits the priority of τ_2, which at time 5 had inherited the priority of τ_1. Hence, the priority of τ_3 is p_1.

Multiple inheritance

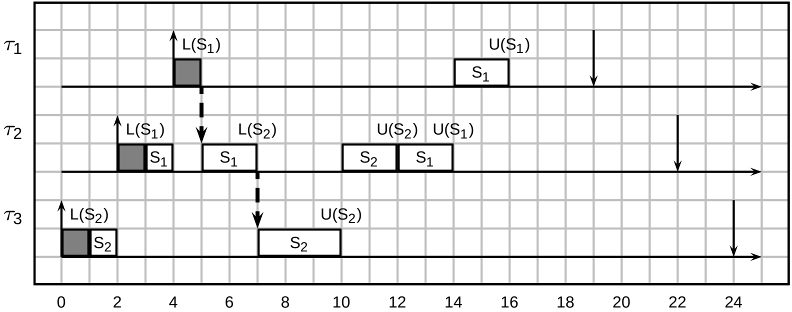
- Task τ_1 uses resource S_1; Task τ_2 uses S_1 and S_2 nested inside S_1; Task τ_3 uses only S_2.
- $p_1 > p_2 > p_3$;



- At time $t = 7$ task τ_3 inherits the priority of τ_2, which at time 5 had inherited the priority of τ_1. Hence, the priority of τ_3 is p_1.

Multiple inheritance

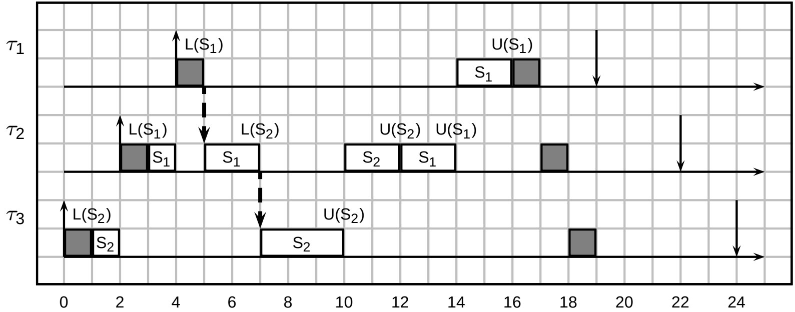
- Task τ_1 uses resource S_1; Task τ_2 uses S_1 and S_2 nested inside S_1; Task τ_3 uses only S_2.
- $p_1 > p_2 > p_3$;



- At time $t = 7$ task τ_3 inherits the priority of τ_2, which at time 5 had inherited the priority of τ_1. Hence, the priority of τ_3 is p_1.

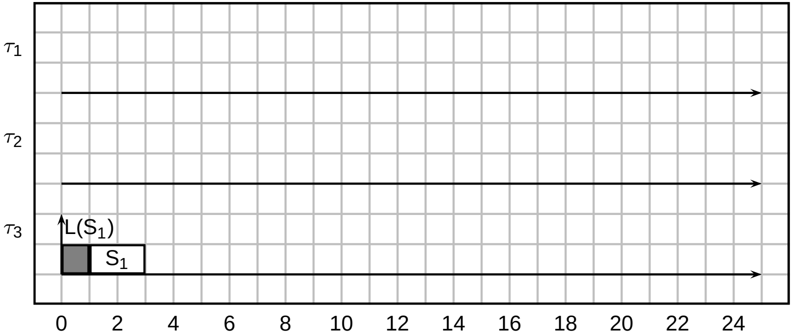
Multiple inheritance

- Task τ_1 uses resource S_1; Task τ_2 uses S_1 and S_2 nested inside S_1; Task τ_3 uses only S_2.
- $p_1 > p_2 > p_3$;

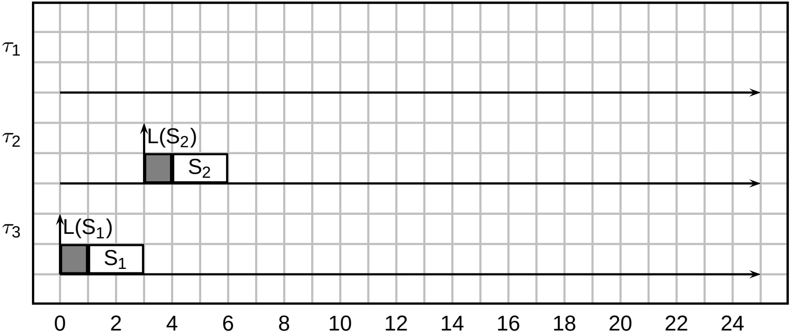


- At time $t = 7$ task τ_3 inherits the priority of τ_2, which at time 5 had inherited the priority of τ_1. Hence, the priority of τ_3 is p_1.

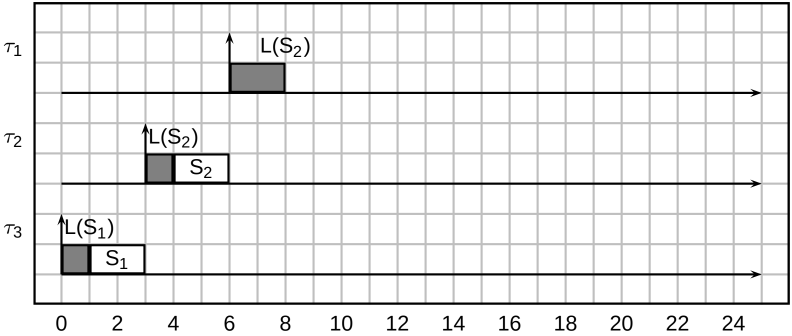
Multiple blocking example



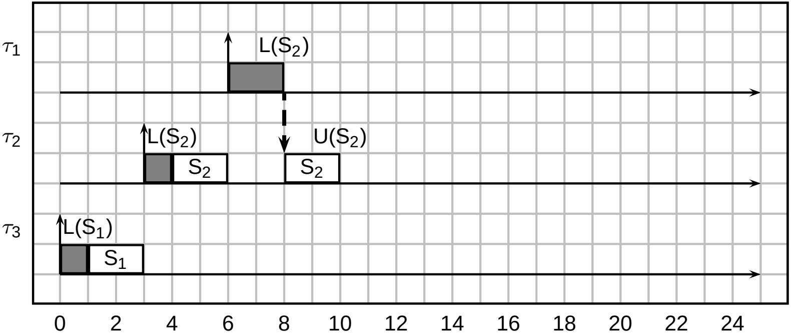
Multiple blocking example



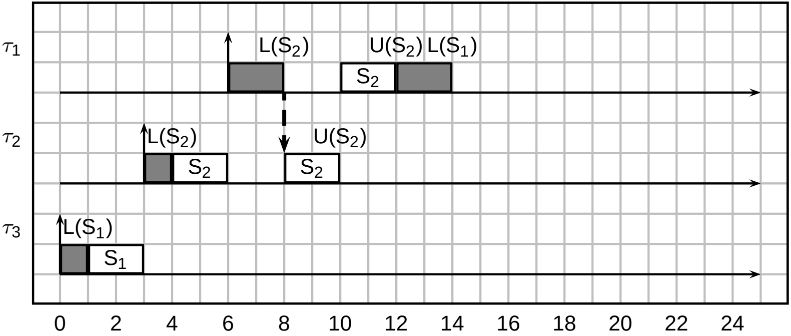
Multiple blocking example



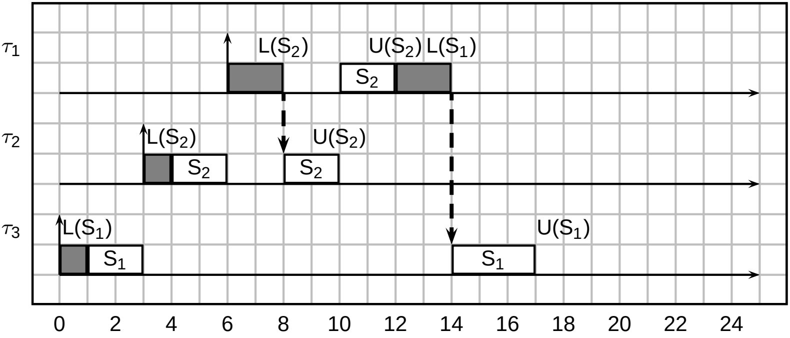
Multiple blocking example



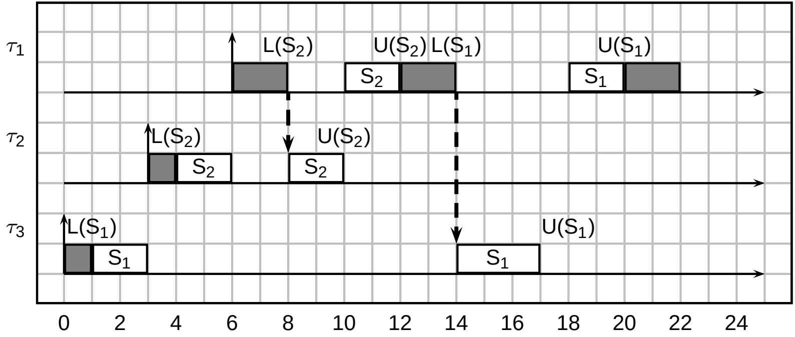
Multiple blocking example



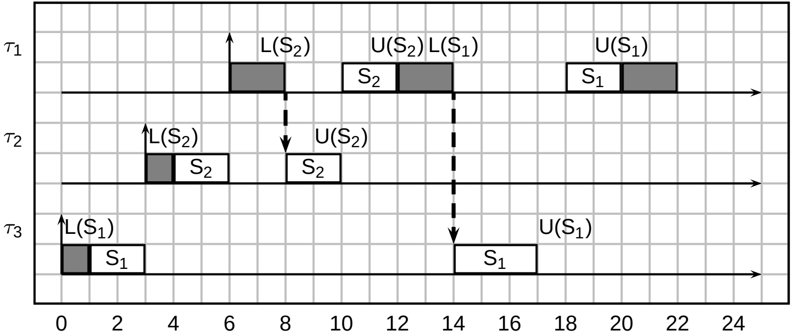
Multiple blocking example



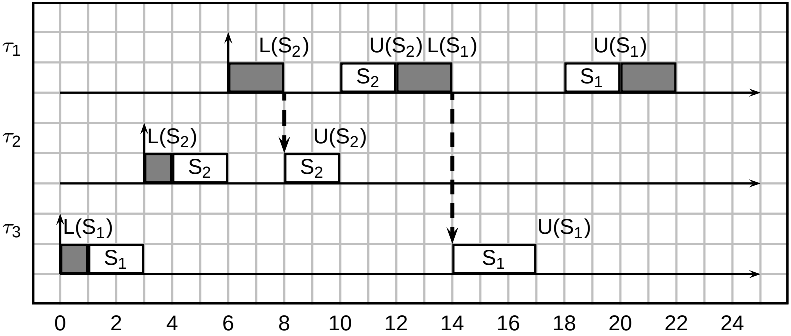
Multiple blocking example



Multiple blocking example



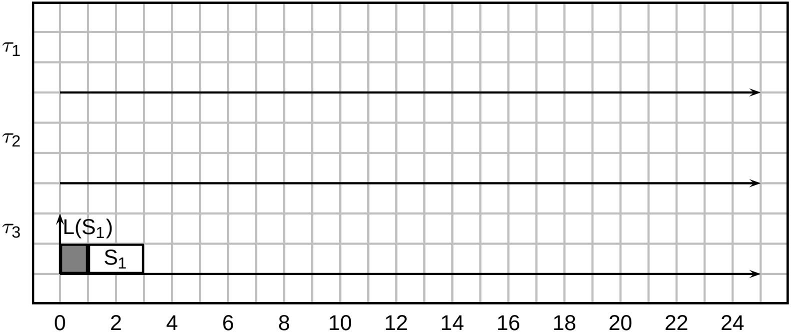
Multiple blocking example



- task τ_1 is blocked twice on two different resources

Example:

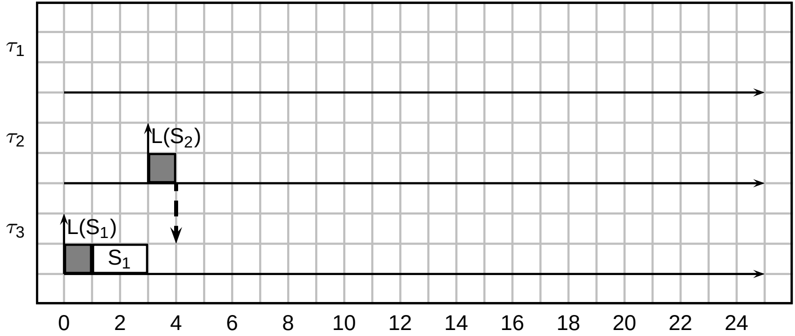
ceil$(S_1) = p_1$ and ceil$(S_2) = p_1$



- task τ_3 acquires the lock

Example:

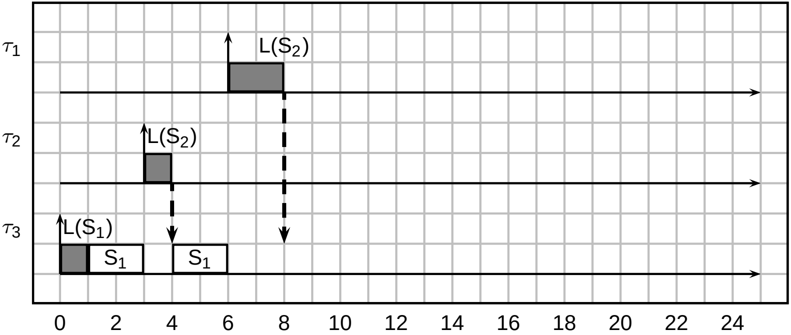
ceil(S_1) $= p_1$ and ceil(S_2) $= p_1$



- task τ_3 acquires the lock
- task τ_2 is blocked because $p_2 <$ maxceil $= p_1$
- task τ_3 inherits τ_2's priority

Example:

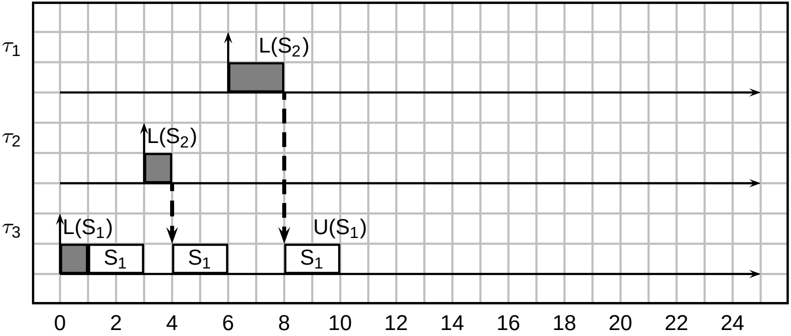
ceil(S_1) $= p_1$ and ceil(S_2) $= p_1$



- task τ_3 acquires the lock
- task τ_2 is blocked because $p_2 <$ maxceil $= p_1$
- task τ_3 inherits τ_2's priority
- task τ_1 is blocked for the same reason
- task τ_3 inherits τ_1's priority

Example:

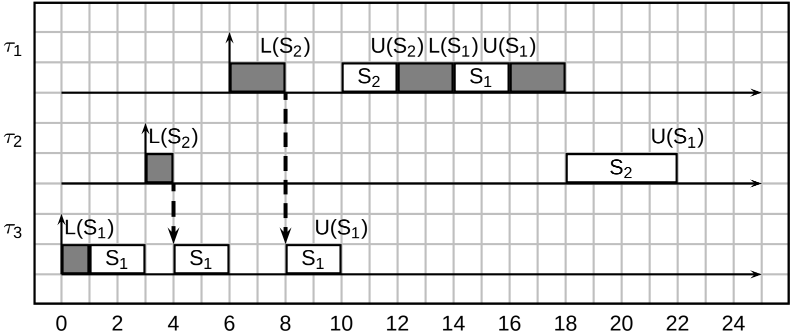
ceil(S_1) $= p_1$ and ceil(S_2) $= p_1$



- task τ_3 acquires the lock
- task τ_2 is blocked because $p_2 <$ maxceil $= p_1$
- task τ_3 inherits τ_2's priority
- task τ_1 is blocked for the same reason
- task τ_3 inherits τ_1's priority
- task τ_3 returns to its original priority, since it is not blocking anyone

Example:

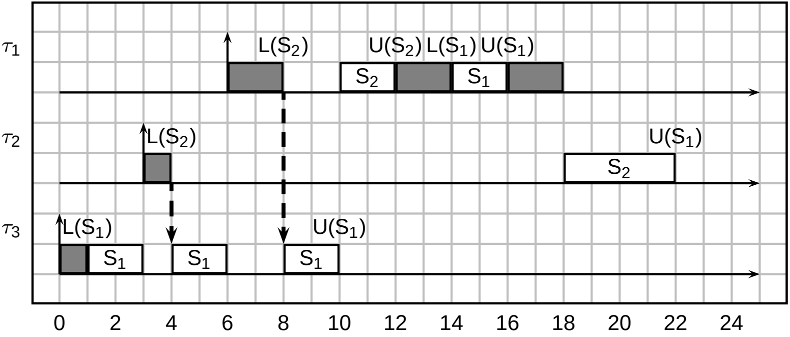
ceil$(S_1) = p_1$ and ceil$(S_2) = p_1$



- task τ_3 acquires the lock
- task τ_2 is blocked because $p_2 <$ maxceil $= p_1$
- task τ_3 inherits τ_2's priority
- task τ_1 is blocked for the same reason
- task τ_3 inherits τ_1's priority
- task τ_3 returns to its original priority, since it is not blocking anyone

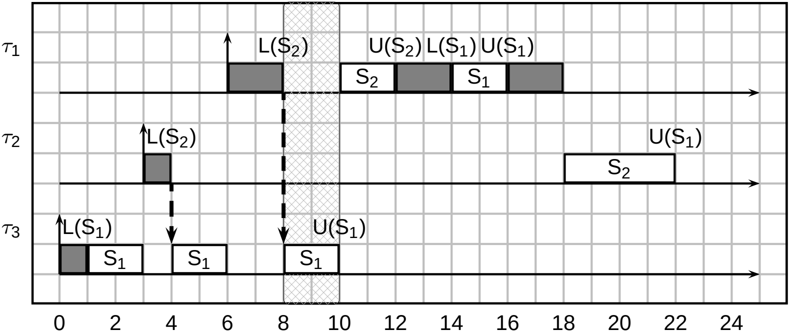
Blocking

In the previous example:



Blocking

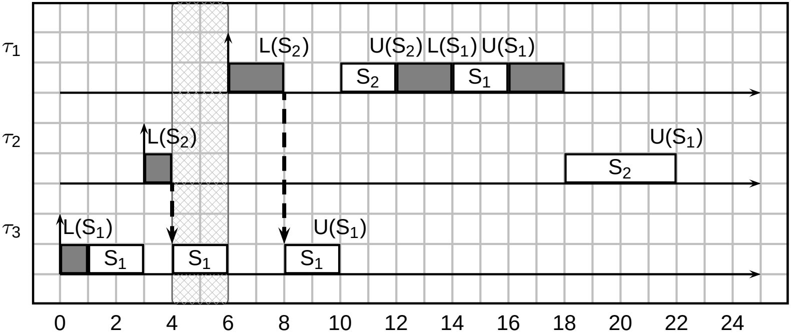
In the previous example:



- Blocking time for τ_1: 2

Blocking

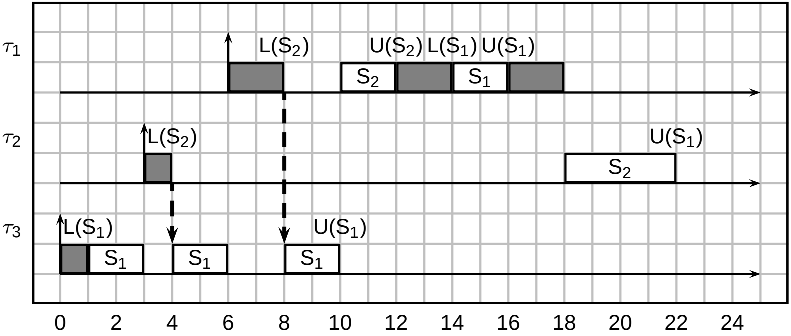
In the previous example:



- Blocking time for τ_1: 2
- Blocking time for τ_2: 2

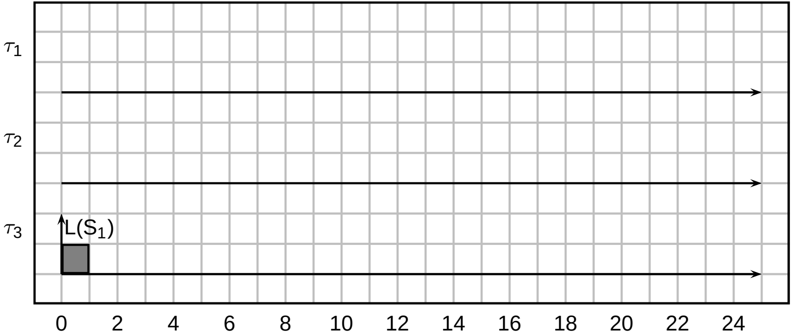
Blocking

In the previous example:



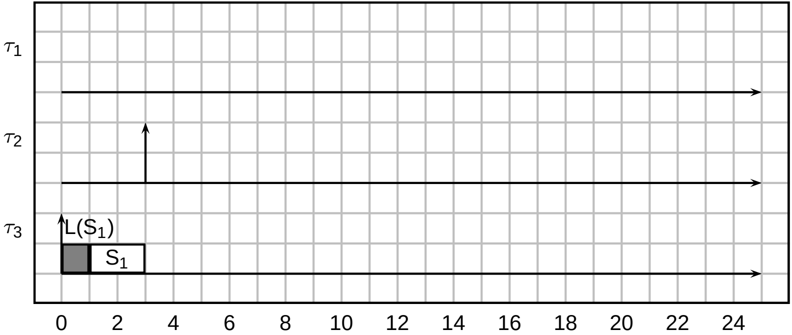
- Blocking time for τ_1: 2
- Blocking time for τ_2: 2
- No multiple blockings!

Example



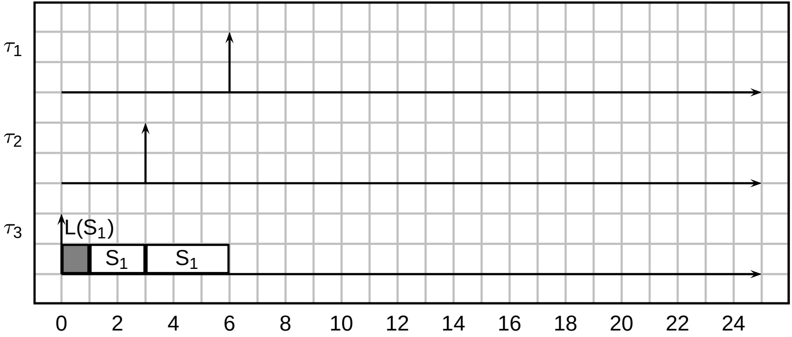
- Task τ_3 raises the sys ceiling to p_1

Example



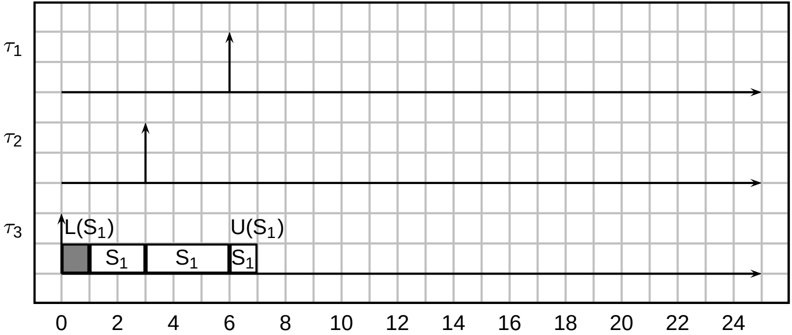
- Task τ_3 raises the sys ceiling to p_1
- Task τ_2 cannot start because $p_2 < \Pi_s = p_1$

Example



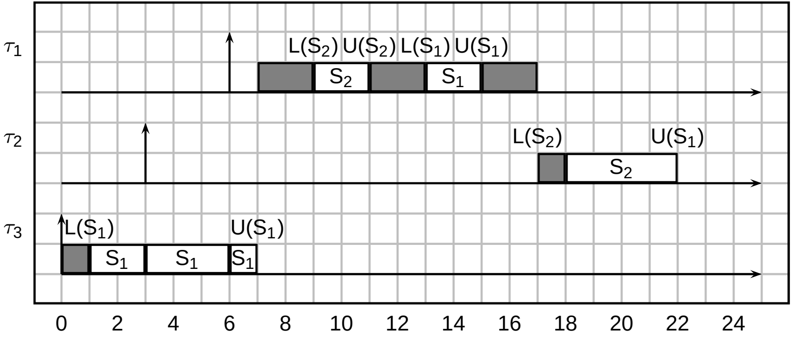
- Task τ_3 raises the sys ceiling to p_1
- Task τ_2 cannot start because $p_2 < \Pi_s = p_1$
- Task τ_1 cannot start because $p_1 = \Pi_s = p_1$

Example



- Task τ_3 raises the sys ceiling to p_1
- Task τ_2 cannot start because $p_2 < \Pi_s = p_1$
- Task τ_1 cannot start because $p_1 = \Pi_s = p_1$
- When task τ_3 unlocks, the sys ceiling goes down, and all other tasks can start executing

Example



- Task τ_3 raises the sys ceiling to p_1
- Task τ_2 cannot start because $p_2 < \Pi_s = p_1$
- Task τ_1 cannot start because $p_1 = \Pi_s = p_1$
- When task τ_3 unlocks, the sys ceiling goes down, and all other tasks can start executing