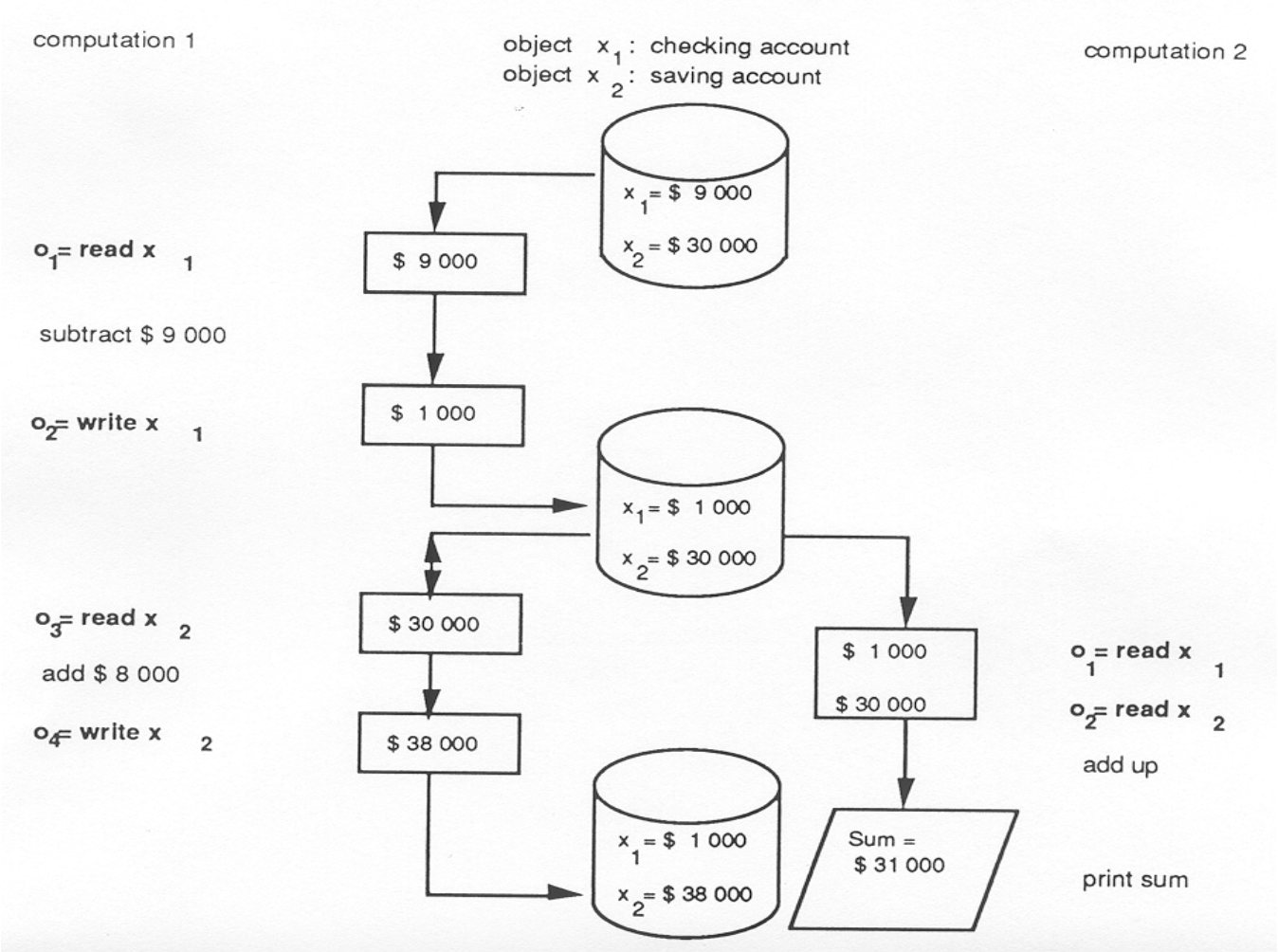
Real-Time (Paradigms) (47)

Memory: Memory Access Protocols

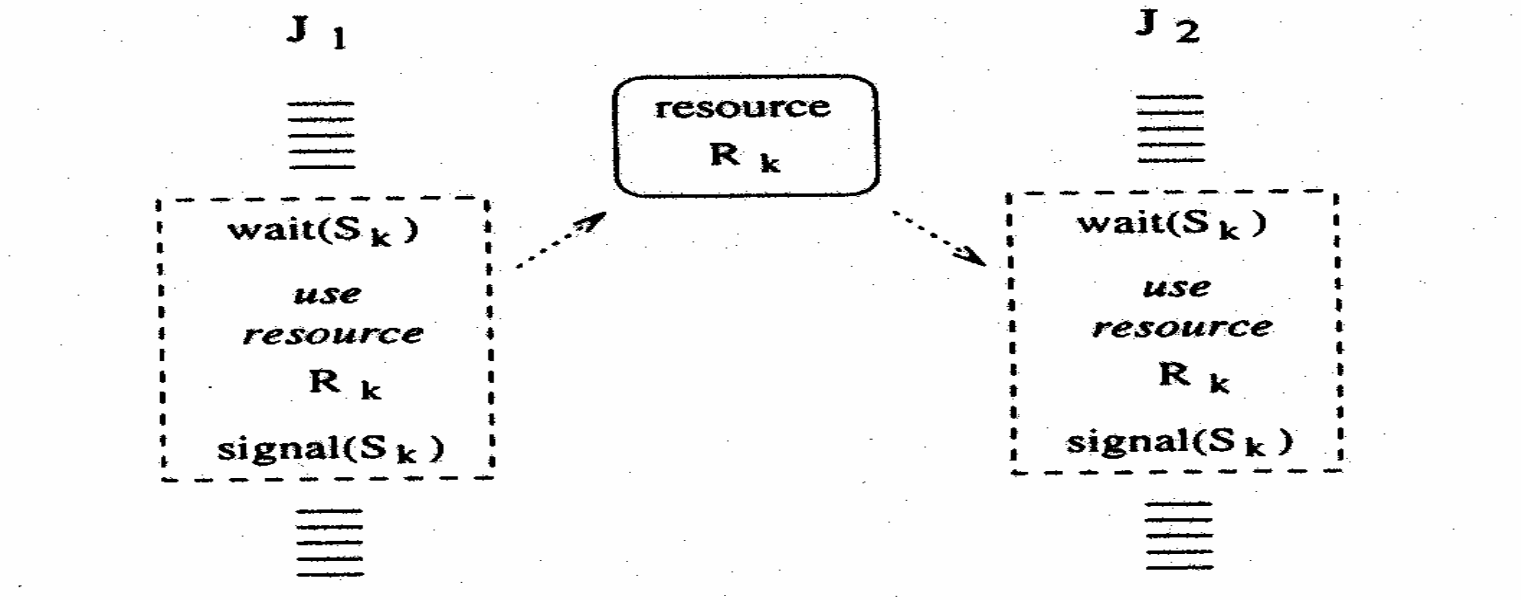
Tasks competing for memory access (become interdependent, a common phenomenon especially in distributed systems

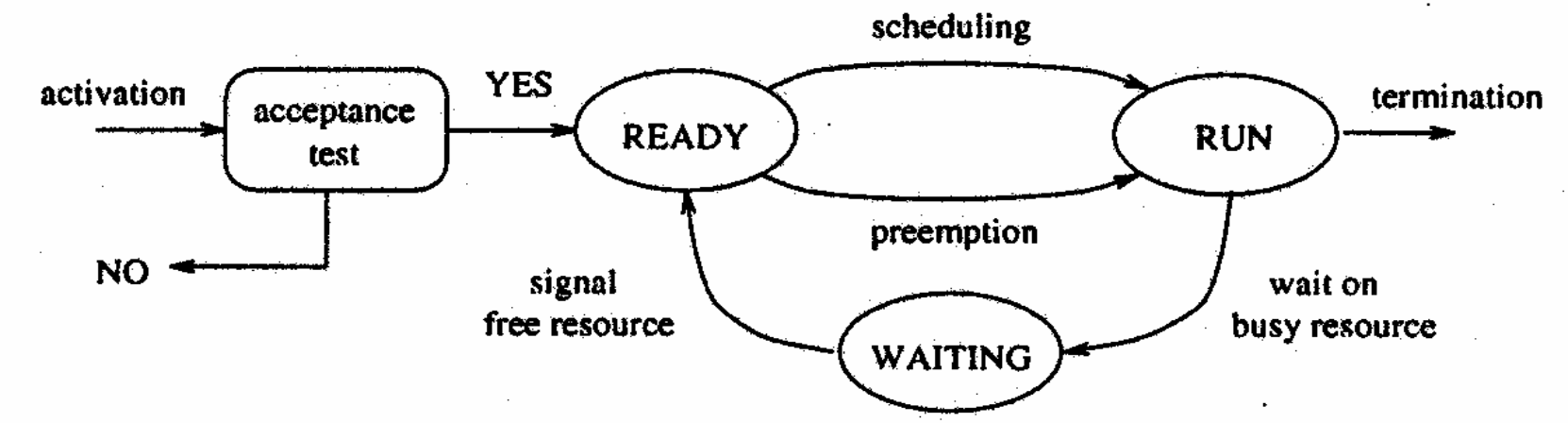
Example:



Real-Time (Paradigms) (47a)

Realizing mutual exclusion by semaphores when accessing an exclusive resource:



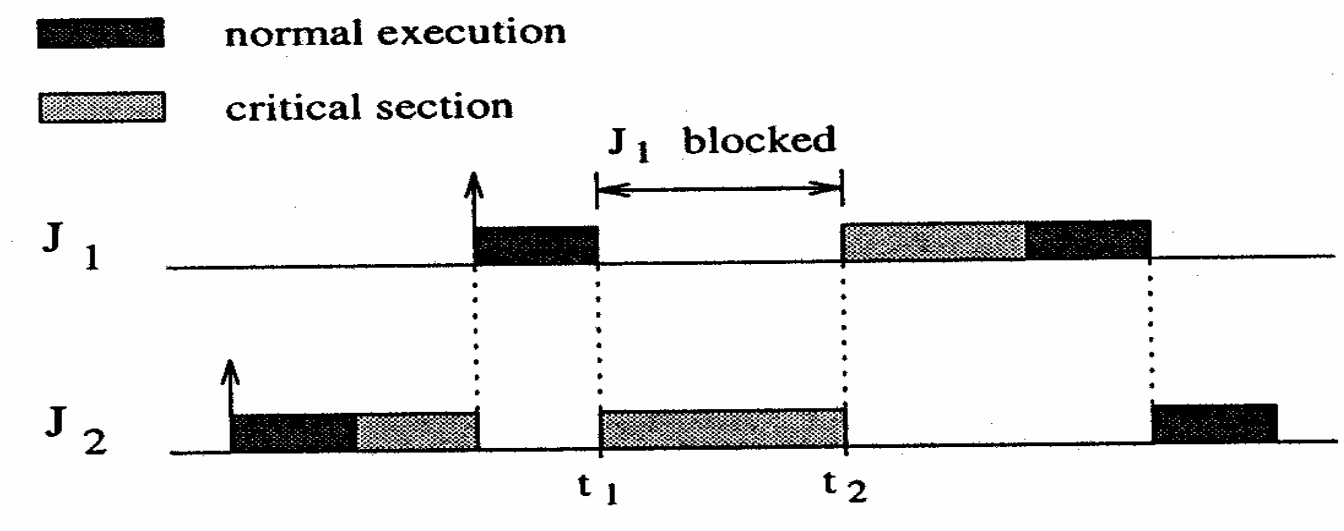


Real-Time (Paradigms) (47a)

---> determining memory access times as part of the overall execution time becomes extremely difficult

Why?

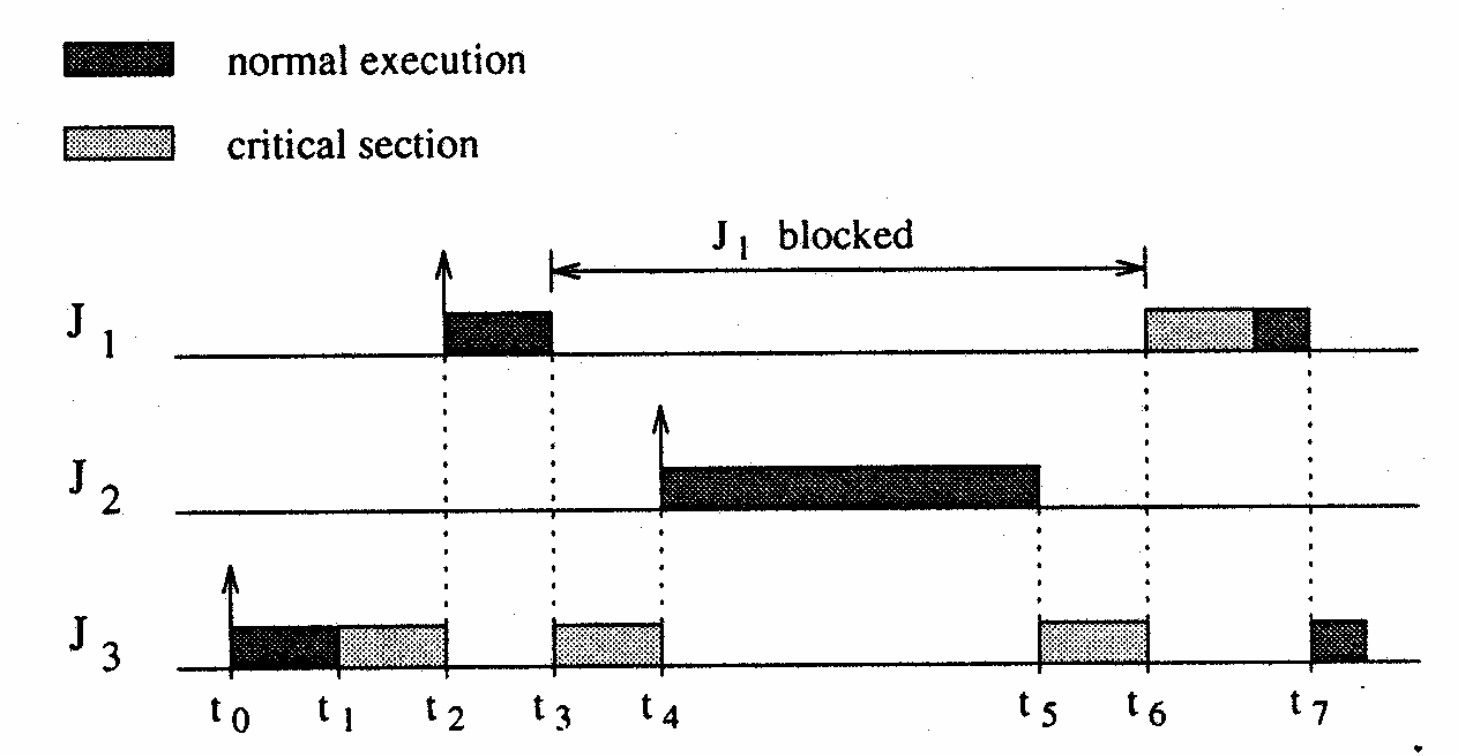
Example of blocking:



——> We still can compute an upper bound on the execution time

Real-Time (Paradigms) (48)

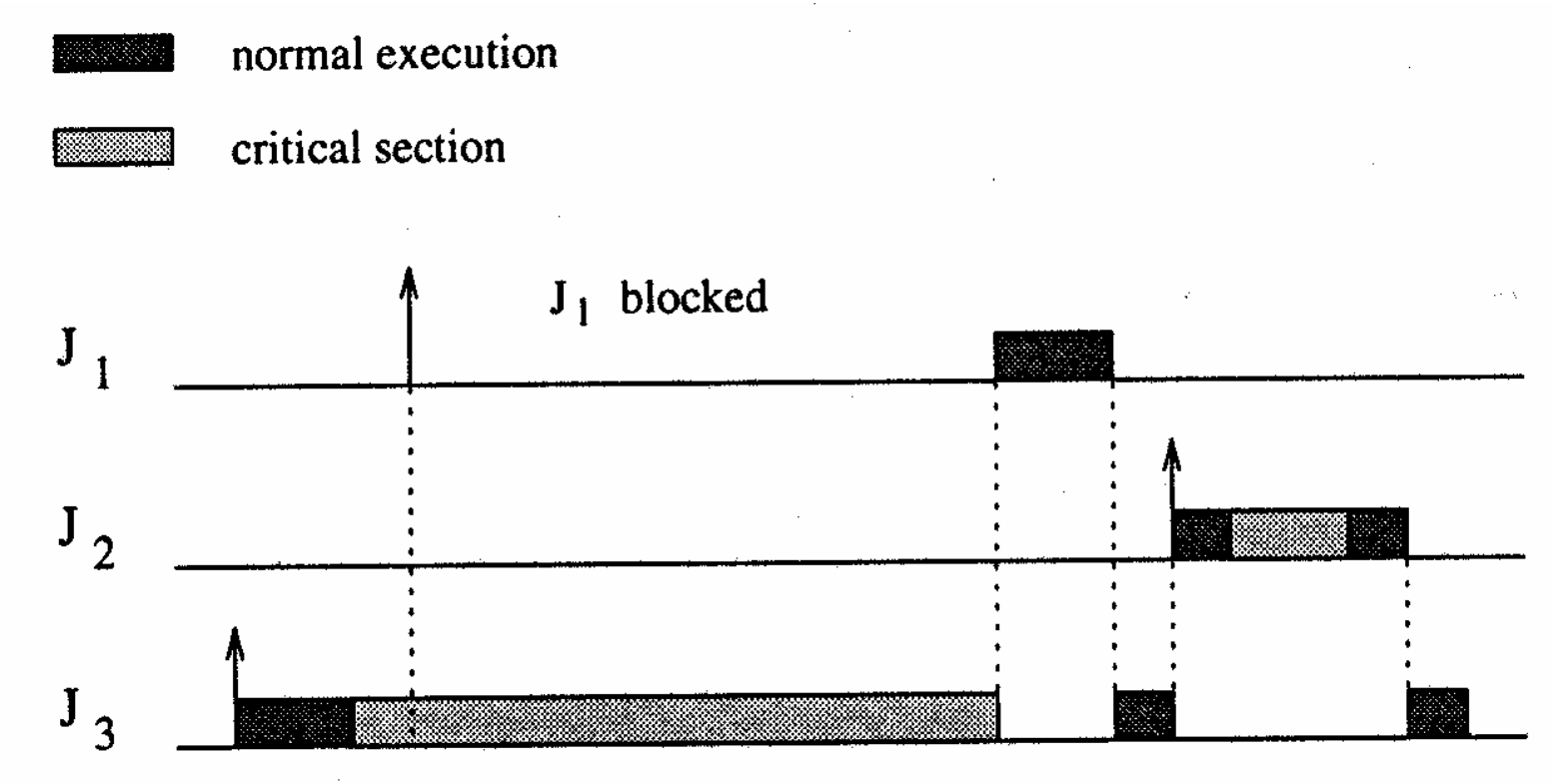
Example of priority inversion:



Effects of priority inversion do affect predictability!

Real-Time (Paradigms) (48a)

Scheduling with non-preemptive critical sections

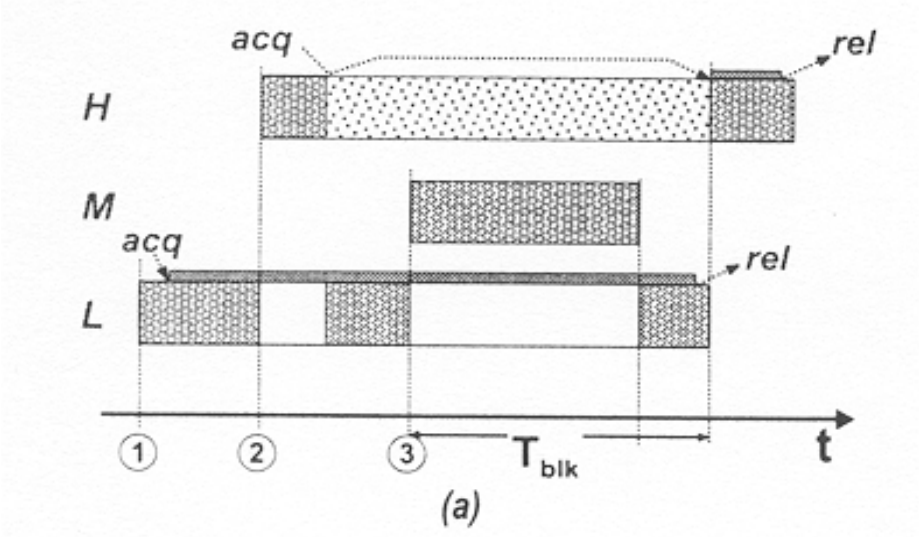


Definition of the priority inheritance protocol:

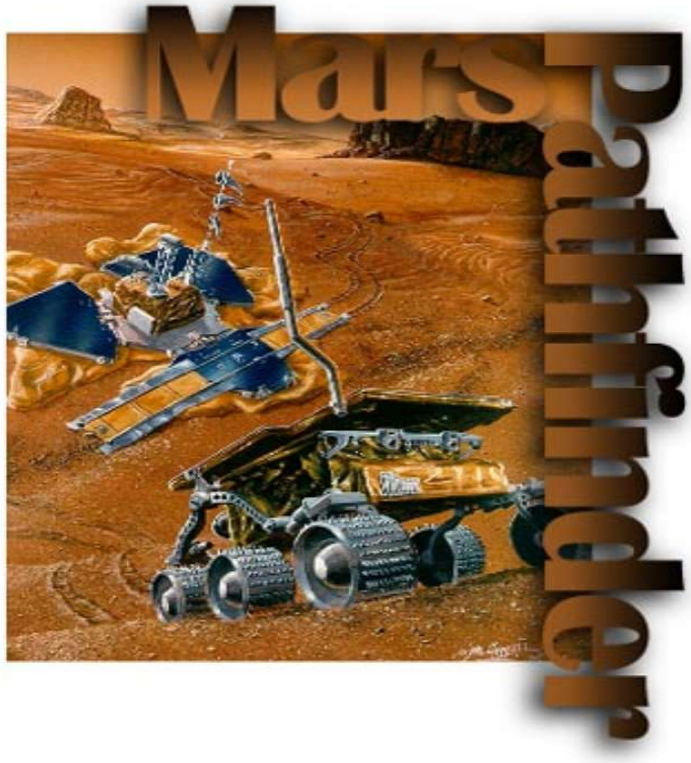
- Jobs are scheduled based on their active priorities. Jobs with the same priority are executed in a First Come First Served discipline.
- When job J_i tries to enter a critical section $z_{i,j}$ and resource $R_{i,j}$ is already held by a lower-priority job, J_i will be blocked. J_i is said to be blocked by the task that holds the resource. Otherwise, J_i enters the critical section $z_{i,j}$.
- When a job J_i is blocked on a semaphore, it transmits its active priority to the job, say J_k, that holds that semaphore. Hence, J_k resumes and executes the rest of its critical section with a priority $p_k = p_i$. J_k is said to *inherit* the priority of J_i. In general, a task inherits the highest priority of the jobs blocked by it.
- When J_k exits a critical section, it unlocks the semaphore, and the highest-priority job, if any, blocked on that semaphore is awakened. Moreover, the active priority of J_k is updated as follows: if no other jobs are blocked by J_k, p_k is set to its nominal priority P_k, otherwise it is set to the highest priority of the jobs blocked by J_k.
- Priority inheritance is transitive; that is, if a job J_3 blocks a job J_2, and J_2 blocks a job J_1, then J_3 inherits the priority of J_1 via J_2.

Real-Time (Paradigms) (49)

Application Example :



It happened on Mars!

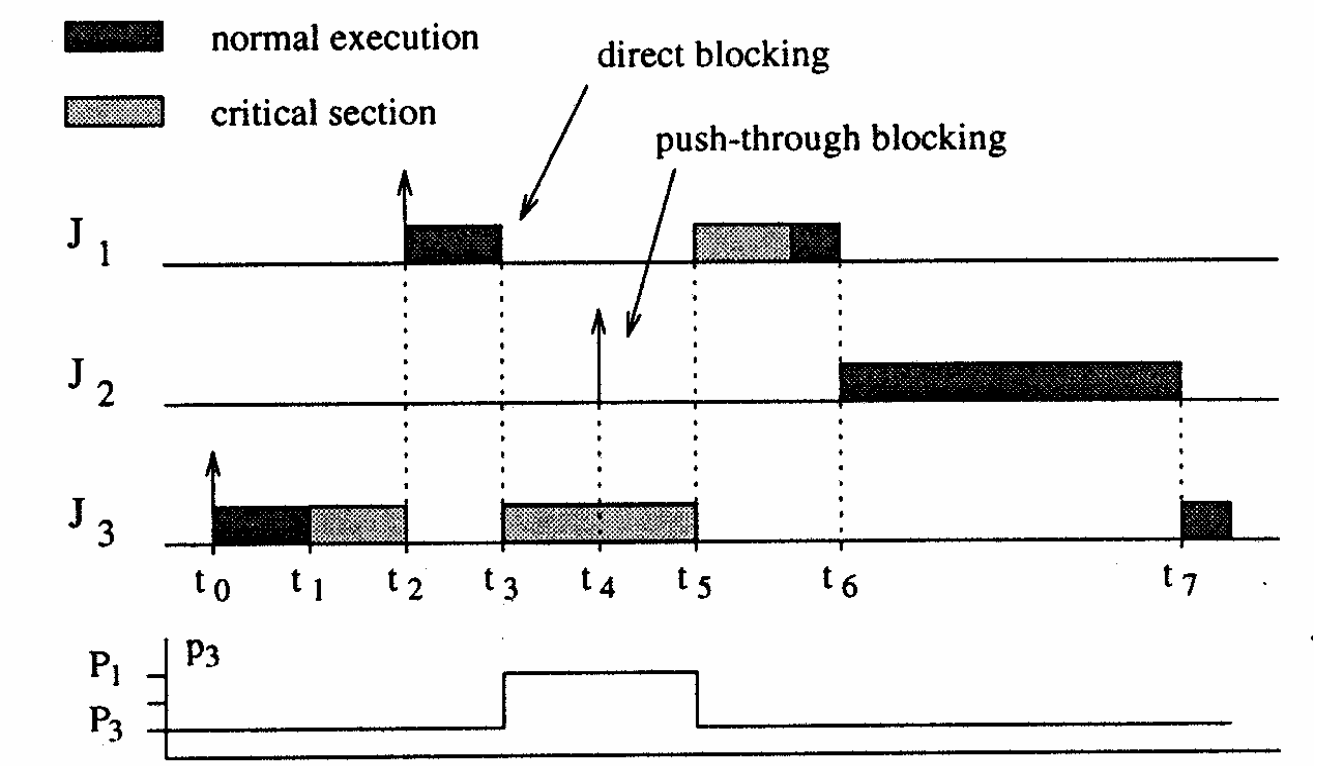


Using *priority inheritance* can prevent priority inversion.

It introduces dynamic priorities defined as follows:
the dynamic priority of a task at time t is the maximum of its initial fixed priority and the priorities of all tasks blocked on account of it at time t.

Real-Time (Paradigms) (50)

Example of a priority inheritance protocol:

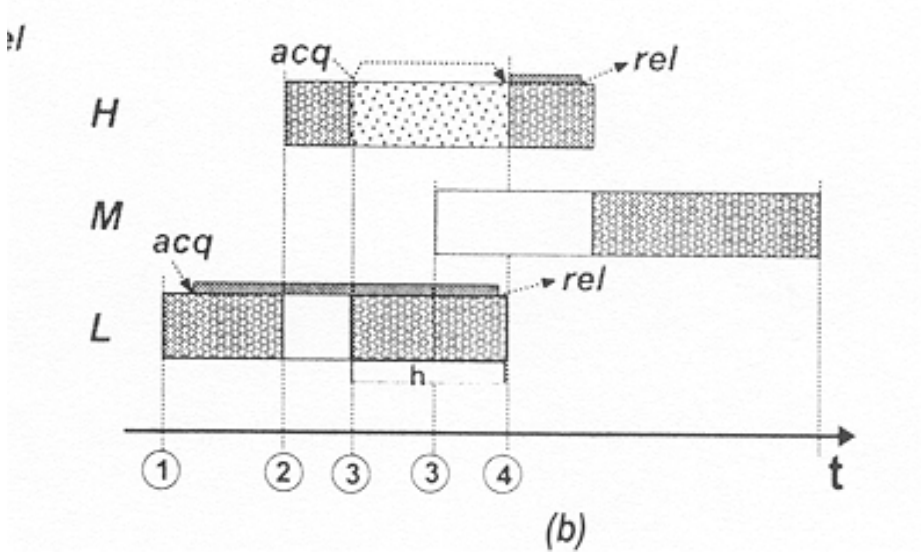


Two kinds of blocking can be distinguished:

- *direct blocking*
- *push-through blocking*

Real-Time (Paradigms) (50a)

The same example using priority inheritance:



- *1. The meteo task (L) runs with priority l, acquires the mutex and publishes*
- *2. The dispatcher task (H) runs with priority h: H preempts L, tries to acquire the mutex and blocks on it, awaiting for the meteo task L, which runs again inheriting H's priority, h.*
- *3. The communications task (M), with priority m, becomes ready for execution: M waits for L, since h (current pri. of L) is higher than m.*
- *4. L finishes and releases the mutex unblocking H, which grabs the processor (h > m), acquires the mutex, and runs to completion. Then, M runs.*
- *Conclusion: H had the minimum blocking possible: waiting for L to finish.*

Resource Access Protocols (6)

Properties of the priority inheritance protocol :

Lemma 7.3 *If there are n lower-priority jobs that can block a job J_i, then J_i can be blocked for at most the duration of n critical sections (one for each of the n lower-priority jobs), regardless of the number of semaphores used by J_i.*

Lemma 7.4 *If there are m distinct semaphores that can block a job J_i, then J_i can be blocked for at most the duration of m critical sections, one for each of the m semaphores.*

Theorem 7.1 (Sha-Rajkumar-Lehoczky) *Under the Priority Inheritance Protocol, a job J can be blocked for at most the duration of* $\min(n, m)$ *critical sections, where n is the number of lower-priority jobs that could block J and m is the number of distinct semaphores that can be used to block J.*

Theorem 7.2 *A set of n periodic tasks using the Priority Inheritance Protocol can be scheduled by the Rate-Monotonic algorithm if*

$$\forall i, \quad 1 \leq i \leq n, \quad \sum_{k=1}^{i} \frac{C_k}{T_k} + \frac{B_i}{T_i} \leq i(2^{1/i} - 1). \qquad (7.2)$$