

Robotics

Shared resources

Tullio Facchinetti
<tullio.facchinetti@unipv.it>

Tuesday 1st October, 2019

<http://robot.unipv.it/toolleeo>

What's a shared resource?

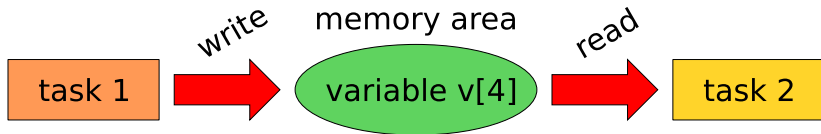
tasks are not independent: they need to share some resource (information)

shared resources:

- registers
- variables
- data structures
- files
- address spaces for peripheral I/O

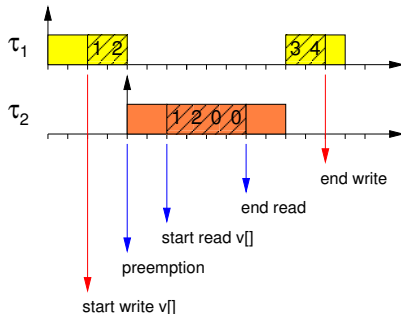
in practice, they all are memory areas

Example



initial condition: $v[4] = [0, 0, 0, 0]$

task 1: $v[4] \leftarrow [1, 2, 3, 4]$



$v[] = [0, 0, 0, 0]$

t1: write $v[0]=1$

t1: write $v[1]=2$

t2: read $v[0]=1$

t2: read $v[1]=2$

t2: read $v[2]=0$

t2: read $v[3]=0$

t1: write $v[2]=3$

t1: write $v[3]=4$

t1 writes

$v[] = [1, 2, 3, 4]$

t2 reads

$v[] = [1, 2, 0, 0]$

when there is a protection of the concurrent access the resource is said mutually exclusive

- protection is managed by semaphores
- not the only solution (e.g., non preemptive sections)

critical section

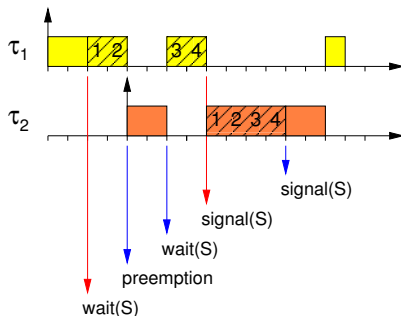
portion of code that accesses a mutually exclusive resource

Critical section

semaphores are used to reserve a resource

initial condition: $v[4] = [0, 0, 0, 0]$

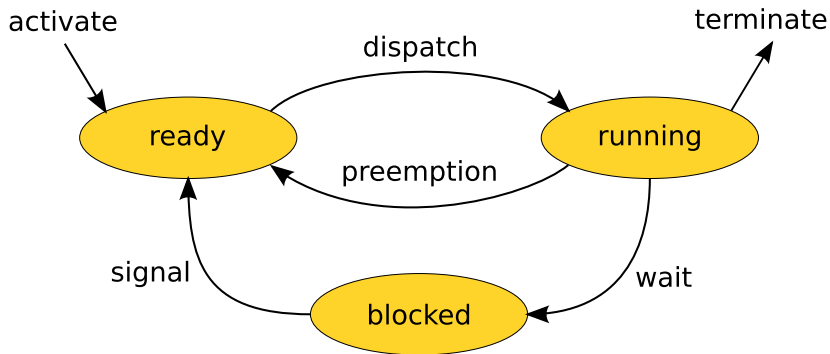
task 1: $v[4] \leftarrow [1, 2, 3, 4]$



```
t1: wait(S)
t1: write v[0]=1
t1: write v[1]=2
t2: wait(S)
t1: write v[2]=3
t1: write v[3]=4
t1: signal(S)
t2: read v[0]=1
t2: read v[1]=2
t2: read v[2]=0
t2: read v[3]=0
t2: signal(S)
```

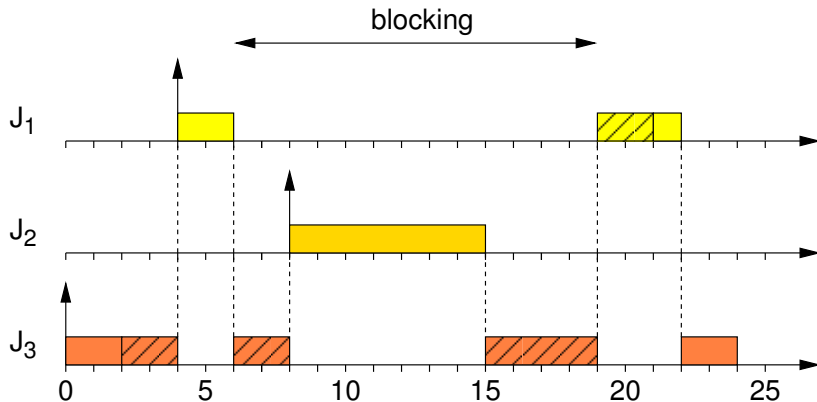
```
t1 writes
v[]=[1,2,3,4]
t2 reads
v[]=[1,2,3,4]
```

Task blocking diagram



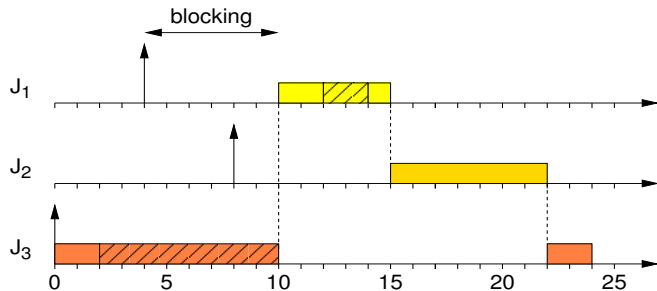
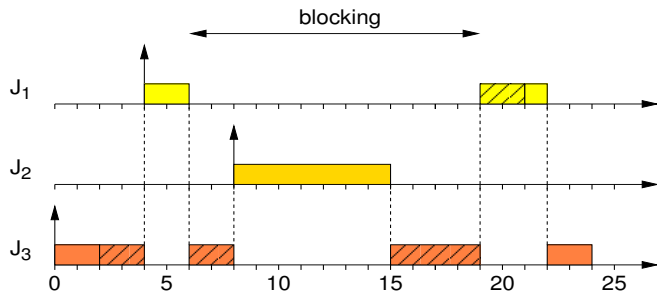
a task waiting for a resource is said **blocked**

Priority inversion



task J_1 is blocked for the whole duration of a medium priority task that does not even require the blocking resource

Non-preemptive critical sections



Priority Inheritance Protocol

IDEA

when a task J_{low} is blocking a higher priority task J_{high} ,
it inherits the priority of J_{high}

in this way

a medium priority task J_{med} can not preempt J_{low} ,
thus can not block J_{high}

therefore

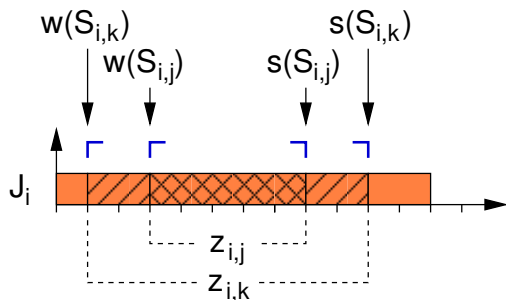
the priority inversion is avoided

PIP: assumptions

- n periodic tasks $\tau_1 \dots \tau_n$ in decreasing priority order
- periods T_i , WCETs C_i , implicit deadlines
- m resources $R_1 \dots R_m$ associated to semaphores $S_1 \dots S_m$
- J_i is a job of τ_i
- each job has a nominal priority P_i and an active priority p_i
- $p_i \geq P_i$
- set $p_i = P_i$ at $t = 0$

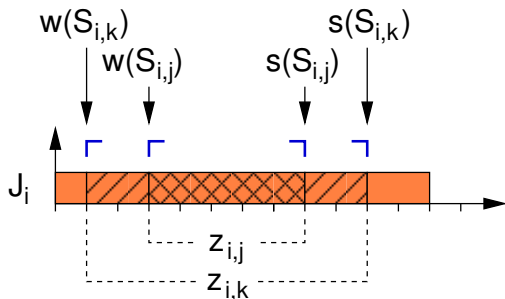
PIP can only used under a static priority assignment

PIP: assumptions



- $z_{i,j}$ is the j -th critical section of job J_i
- $z_{i,j}$ is associated to the semaphore $S_{i,j}$ of the resource $R_{i,j}$
- $z_{i,j} \in z_{i,k}$ means that $z_{i,j}$ is completely contained into $z_{i,k}$

PIP: assumptions



- tasks do not self-suspend
- critical sections are correctly nested
- critical sections are guarded by binary semaphores (only one job can lock a given resource)

PIP: operating rules

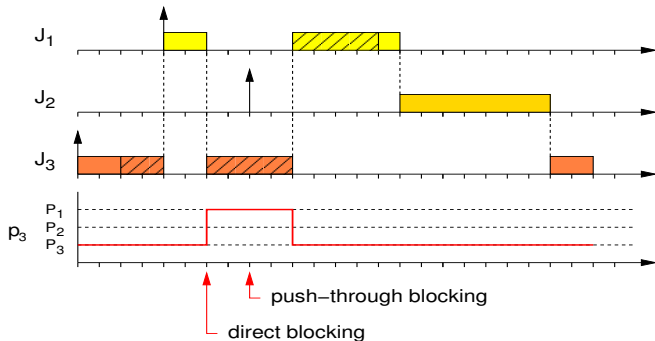
- jobs scheduling is based on the active priority p_i
- tasks having the same priority are scheduled using FIFO
- when J_i tries to enter the critical section $z_{i,j}$
 - if the resource $R_{i,j}$ is locked by a task J_k , J_i is blocked
 - otherwise J_i enters the critical section

PIP: operating rules

- if J_i is blocked by a lower priority task J_k , J_k inherits the priority of J_i , i.e., $p_k = p_i$
- J_k is resumed and scheduled at priority p_i
- when J_k exits the critical section, the higher priority task w.r.t. J_k is activated and...
- if J_k does not block any other task, its priority is set $p_k = P_k$; otherwise J_k inherits the priority of the highest priority task currently blocked

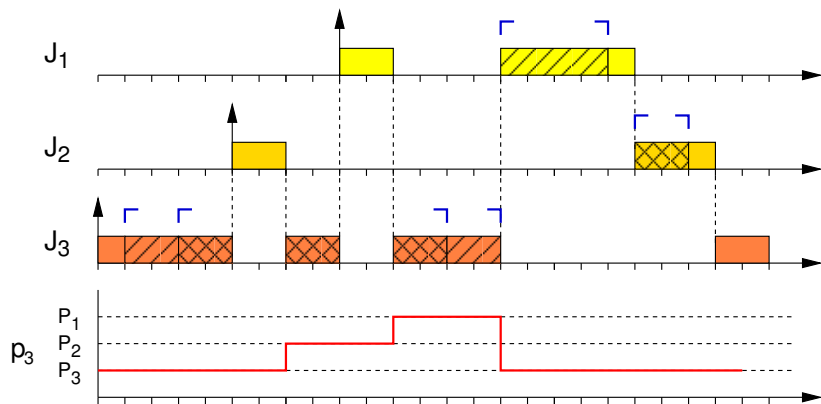
the inheritance is transitive: if J_3 is blocking J_2 and J_2 is blocking J_1 , then J_3 inherits the priority of J_1 through J_2

PIP: example



- direct blocking: it happens when a higher priority task tries to access the blocked resource; it guarantees the consistency of the resource
- push-through blocking: a medium priority task is blocked by a lower priority task; it avoids the priority inversion

nested critical sections



transitive inheritance

