technische universität dortmund

fakultät für informatik
informatik 12

# Resource Access Protocols

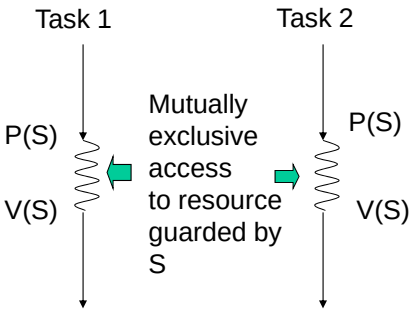
Peter Marwedel
Informatik 12
TU Dortmund
Germany

Graphics: © Alexandra Nolte, Gesine Marwedel, 2003

**Subset of slides selected for EECS 222C.**

## Resource access protocols

**Critical sections:** sections of code at which exclusive access to some resource must be guaranteed. Can be guaranteed with semaphores S or “mutexes”.

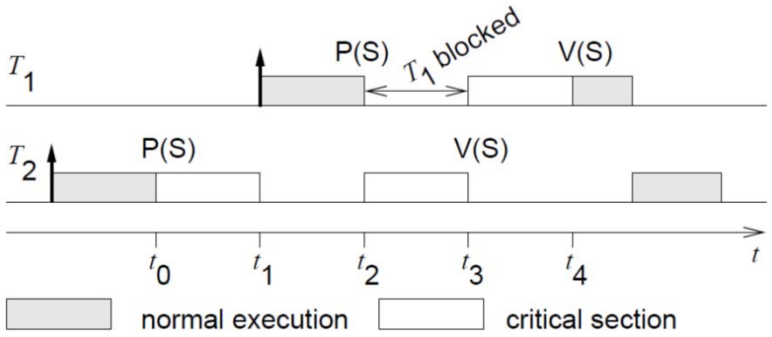



P(S) checks semaphore to see if resource is available and if yes, sets S to “used“. Uninterruptible operations! If no, calling task has to wait.

V(S): sets S to “unused“ and starts sleeping task (if any).

## Blocking due to mutual exclusion

Priority $T_1$ assumed to be > than priority of $T_2$.
If $T_2$ requests exclusive access first (at $t_0$), $T_1$ has to wait until $T_2$ releases the resource (time $t_3$), thus inverting the priority:

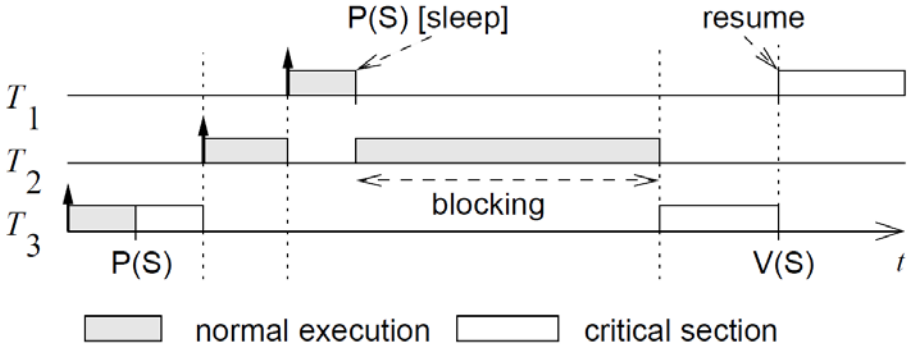



In this example:
blocking is bounded by length of critical section of $T_2$.



## Blocking with >2 tasks can exceed the length of any critical section

Priority of $T_1$ > priority of $T_2$ > priority of $T_3$.
$T_2$ preempts $T_3$:
$T_2$ can prevent $T_3$ from releasing the resource.




**Priority inversion!**

## The MARS Pathfinder problem (1)

"But 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. The press reported these failures in terms such as "software glitches" and "the computer was trying to do too many things at once"." …

http://research.microsoft.com/~mbj/Mars_Pathfinder/Mars_Pathfinder.html



## The 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)."

http://research.microsoft.com/~mbj/Mars_Pathfinder/Mars_Pathfinder.html

## The MARS Pathfinder problem (3)

- 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. ..
- The spacecraft also contained a communications task that ran with medium priority."

☞

High priority: retrieval of data from shared memory
Medium priority: communications task
Low priority: thread collecting meteorological data

http://research.microsoft.com/~mbj/Mars_Pathfinder/Mars_Pathfinder.html



## The MARS Pathfinder problem (4)

"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."

http://research.microsoft.com/~mbj/Mars_Pathfinder/Mars_Pathfinder.html

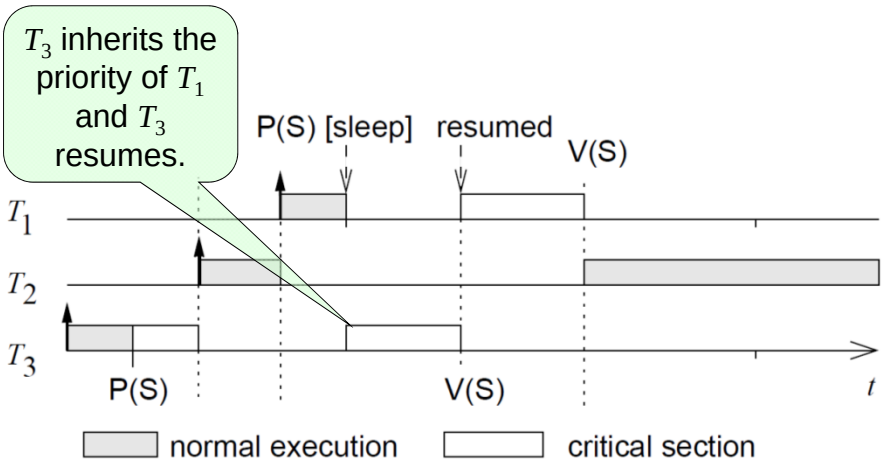
## Coping with priority inversion: the priority inheritance protocol

- Tasks are scheduled according to their active priorities. Tasks with the same priorities are scheduled FCFS.
- If task $T_1$ executes P(S) & exclusive access granted to $T_3$: $T_1$ will become blocked.
  If priority($T_3$) < priority($T_1$): $T_3$ inherits the priority of $T_1$.
  ☞ $T_3$ resumes.
  Rule: tasks inherit the highest priority of tasks blocked by it.
- When $T_3$ executes V(S), its priority is decreased to the highest priority of the tasks blocked by it.
  If no other task blocked by $T_3$: priority($T_3$):= original value.
  Highest priority task so far blocked on S is resumed.
- Transitive: if $T_3$ blocks $T_2$ and $T_2$ blocks $T_1$, then $T_3$ inherits the priority of $T_1$.



## Example

How would priority inheritance affect our example with 3 tasks?



## Priority inversion on Mars

Priority inheritance also 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”.

The problem on Mars was corrected by using the debugging facilities of VxWorks to change the flag to “on”, while the Pathfinder was already on the Mars [Jones, 1997].



## Remarks on priority inheritance protocol

Possible large number of tasks with high priority.

Possible deadlocks.

Ongoing debate about problems with the protocol:

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

Finds application in ADA: During *rendez-vous*, task priority is set to the maximum.

Protocol for fixed set of tasks: priority ceiling protocol.