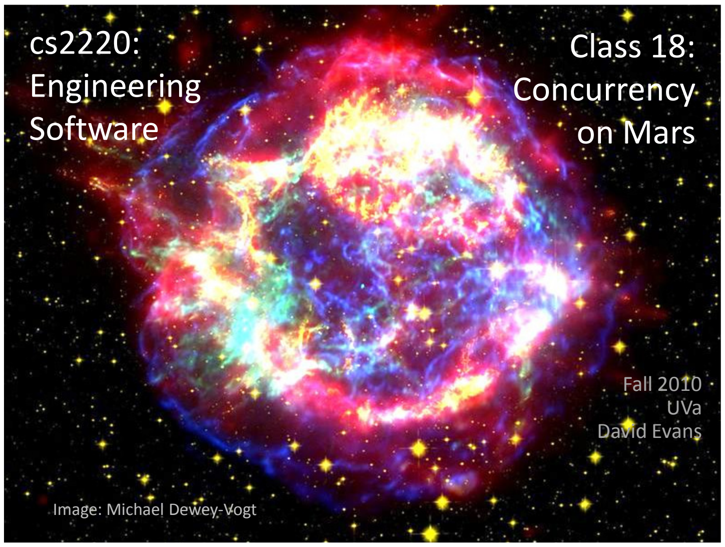


cs2220: Engineering Software

Class 18: Concurrency on Mars

Fall 2010
UVa
David Evans

Image: Michael Dewey-Vogt

Plan for Today

wait and **notify**

Concurrency on Mars!

Project Time

Scheduling Update:

Exam 2 (originally scheduled for Oct 28-Nov 2) will now be: **Nov 18-Nov 23**

Project Team Requests: 11:59pm Friday

in class Tues, Nov 2 $\rightarrow$ **Project Idea Proposals: ~~11:59pm, Wednesday, Nov 3~~**

Project Design Document: Class, Tuesday, Nov 9

Design Reviews: Nov 10-17 (scheduled by team)

Project Progress Reports: Tuesday, Nov 30

Project Demos/Presentations: 7 December (last class)

Synchronizing

synchronized(*obj*) { *code* }

Provides mutual exclusion: code inside synchronized can only run when lock of *obj* is held

***obj*.wait()**

Gives up lock on *obj*; puts current thread in waiting set for *obj*

***obj*.notify(), *obj*.notifyAll()**

Don't give up lock; selects one (notify) or all (notifyAll) threads in waiting set for *obj* and wakes them up (to be scheduled)

Methods inherited from class java.lang.Object
finalize, getClass, notify, notifyAll, wait, wait, wait

Wait, Wait Don't Notify Me!

http://download.oracle.com/javase/6/docs/api/java/lang/Object.html#wait%28%29

public final void wait() throws InterruptedException
Causes the current thread to wait until another thread invokes the notify() method or the notifyAll() method for this object. In other words, this method behaves exactly as if it simply performs the call wait(0).

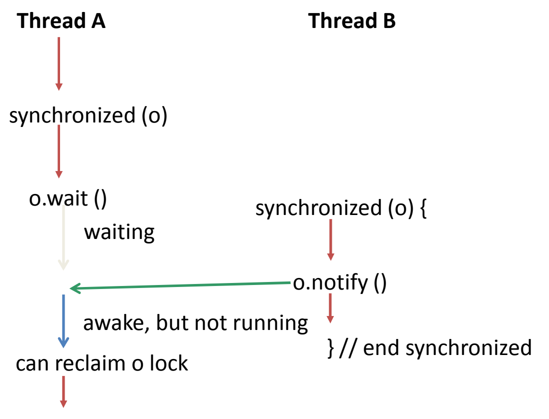
public final void wait(long ms, long nanos)

public final void wait(long timeout) throws InterruptedException
Causes the current thread to wait until either another thread invokes the notify() method or the notifyAll() method for this object, or a specified amount of time has elapsed.

public final void notify()
Wakes up a single thread that is waiting on this object's monitor. If any threads are waiting on this object, one of them is chosen to be awakened. The choice is arbitrary and occurs at the discretion of the implementation. A thread waits on an object's monitor by calling one of the wait methods.

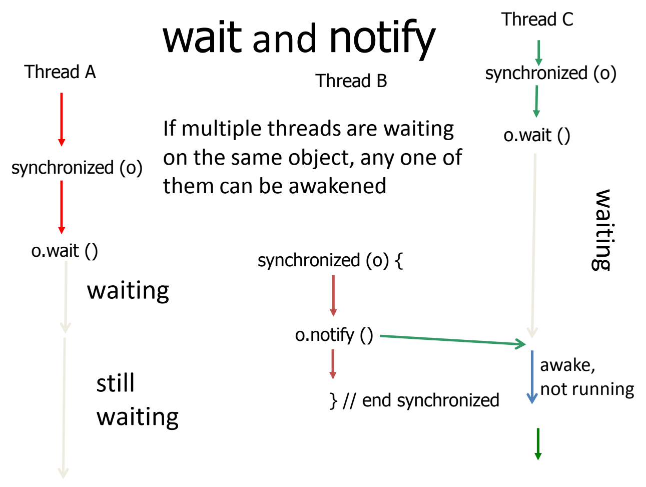
The awakened thread will not be able to proceed until the current thread relinquishes the lock on this object

wait and notify



wait and notify

If multiple threads are waiting on the same object, any one of them can be awakened



```
class IncThread extends Thread {
  private Counter c;                                              while (x>0) {
  public IncThread (Counter p_c) { c = p_c; }
  public void run () {
    while (true) {                                                   :
      synchronized (c) {
        c.increment ();                                            }
        System.err.println ("Running inc thread: " + currentThread () + …);   NOT x>0
        c.notify ();
      } } } }

class DecThread extends Thread {
  …
  public void run () {
     while (true) {
        synchronized (c) {
           while (c.getValue () <= 0) {
             try { c.wait (); } catch (InterruptedException e) { ; }
           }
           c.decrement ();
           System.err.println ("Running dec thread: " + …);
      } } } }
```

```
Counter c = new Counter ();
IncThread ithread = new IncThread (c);
DecThread dthread = new DecThread (c);
ithread.setPriority (Thread.NORM_PRIORITY);
ithread.start ();
dthread.setPriority (Thread.MAX_PRIORITY);
dthread.start ();
```

```
Running inc thread: Thread[Thread-0,5,main] / Value: 1
Running dec thread: Thread[Thread-1,10,main] / Value: 0
Running inc thread: Thread[Thread-0,5,main] / Value: 1
Running dec thread: Thread[Thread-1,10,main] / Value: 0
Running inc thread: Thread[Thread-0,5,main] / Value: 1
Running dec thread: Thread[Thread-1,10,main] / Value: 0
Running inc thread: Thread[Thread-0,5,main] / Value: 1
Running dec thread: Thread[Thread-1,10,main] / Value: 0
```

Priorities

- In general, threads with higher priorities will be scheduled preferentially.
- There are no guarantees: up to Java scheduler

```
Thread class:
   void setPriority (int newPriority)
      // MODIFIES: this
      // EFFECTS: Changes the priority of this
      //     thread to newPriority.
```

Priorities, Priorities

```
ithread.setPriority (Thread.NORM_PRIORITY);
ithread.start ();

dthread.setPriority (Thread.MIN_PRIORITY);
dthread.start ();
```

The ithread should run more than the dthread, ***but*** there is no guarantee.

```
Thread.MIN_PRIORITY
Thread.NORM_PRIORITY
Thread.MAX_PRIORITY
```

Stopping Threads

public class java.lang.Thread {
  public final void **stop**()

Deprecated. *This method is inherently unsafe.*

Forces the thread to stop executing. …The thread represented by this thread is forced to stop whatever it is doing abnormally and to throw a newly created ThreadDeath object as an exception. …

Why deprecate **stop**?

- What should happen to all the locks a thread owns when it is **stop**ped?
- What if an invariant is temporarily broken in a method?

Suspending Threads

public final void **suspend**()

Suspends this thread. If the thread is alive, it is suspended and makes no further progress unless and until it is resumed.

Deprecated. *This method has been deprecated, as it is inherently deadlock-prone. If the target thread holds a lock on the monitor protecting a critical system resource when it is suspended, no thread can access this resource until the target thread is resumed. If the thread that would resume the target thread attempts to lock this monitor prior to calling resume, deadlock results. Such deadlocks typically manifest themselves as "frozen" processes.*

Can't **stop**, can't **suspend**, what can you do?

Thread x = new Thread();

in **java.lang.Thread**

public void **interrupt**()

Interrupts this thread.

x.run();

x = null;

If this thread is blocked in an invocation of the wait(), wait(long), or wait(long, int) methods of the Object class, or of the join(), join(long), join(long, int), sleep(long), or sleep(long, int), methods of this class, then its interrupt status will be cleared and it will receive an InterruptedException.

…

If none of the previous conditions hold then this thread's interrupt status will be set.

Being Interrupted

public boolean **isInterrupted**()

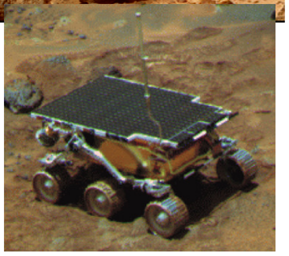
MODIFIES: nothing

EFFECTS: Returns true iff this thread has been interrupted.

```
Counter c = new Counter ();
IncThread ithread = new IncThread (c);
DecThread dthread = new DecThread (c);
ithread.setPriority (Thread.NORM_PRIORITY);
ithread.start ();
dthread.setPriority (Thread.MAX_PRIORITY);
dthread.start ();
dthread.interrupt ();
```

Interrupts are just "polite" requests! The thread can ignore it and keep going…

```
Running inc thread: Thread[Thread-0,5,main] / Value: 1
Running dec thread: Thread[Thread-1,10,main] / Value: 0
Running inc thread: Thread[Thread-0,5,main] / Value: 1
Running dec thread: Thread[Thread-1,10,main] / Value: 0
Running inc thread: Thread[Thread-0,5,main] / Value: 1
Running dec thread: Thread[Thread-1,10,main] / Value: 0
Running inc thread: Thread[Thread-0,5,main] / Value: 1
Running dec thread: Thread[Thread-1,10,main] / Value: 0
```

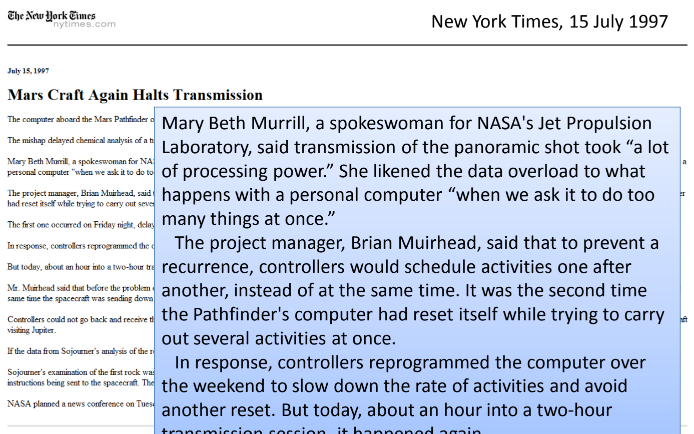


Mars Pathfinder

Landed on Mars

July 4, 1997

Sojourner Rover



The New York Times nytimes.com

New York Times, 15 July 1997

July 15, 1997

Mars Craft Again Halts Transmission

The computer aboard the Mars Pathfinder o

The mishap delayed chemical analysis of a t

Mary Beth Murrill, a spokeswoman for NA personal computer "when we ask it to do to

The project manager, Brian Muirhead, said had reset itself while trying to carry out sever

The first one occurred on Friday night, delay

In response, controllers reprogrammed the c

But today, about an hour into a two-hour tra

Mr. Muirhead said that before the problem same time the spacecraft was sending down

Controllers could not go back and receive th visiting Jupiter.

If the data from Sojourner's analysis of the r

Sojourner's examination of the first rock was instructions being sent to the spacecraft. The

NASA planned a news conference on Tues

Mary Beth Murrill, a spokeswoman for NASA's Jet Propulsion Laboratory, said transmission of the panoramic shot took "a lot of processing power." She likened the data overload to what happens with a personal computer "when we ask it to do too many things at once."

The project manager, Brian Muirhead, said that to prevent a recurrence, controllers would schedule activities one after another, instead of at the same time. It was the second time the Pathfinder's computer had reset itself while trying to carry out several activities at once.

In response, controllers reprogrammed the computer over the weekend to slow down the rate of activities and avoid another reset. But today, about an hour into a two-hour transmission session, it happened again.

Priority-Based Scheduling

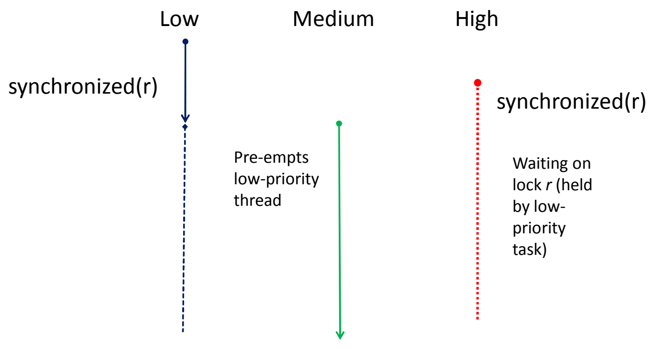
Scheduler ensures that the highest priority task that can run is always running

Lower priority tasks run only when no higher priority task can run

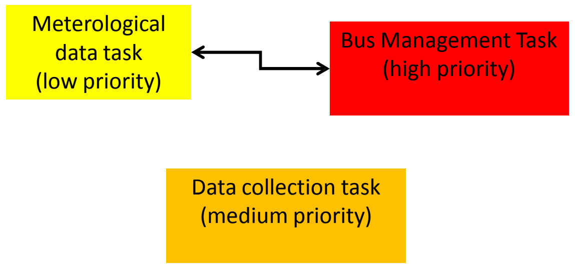
Standard JavaVM scheduler does not do this, but many operating Systems for embedded systems do including the vxWorks used on the PathFinder.

What could go wrong with priority-based scheduling?

Priority Inversion


Low
Medium
High
synchronized(r)
Pre-empts low-priority thread
synchronized(r)
Waiting on lock *r* (held by low-priority task)

Priority Inversion on Mars


Meterological data task (low priority)
Bus Management Task (high priority)
Data collection task (medium priority)

For details, see Glenn Reeves account:
http://research.microsoft.com/~mbj/Mars_Pathfinder/Authoritative_Account.html

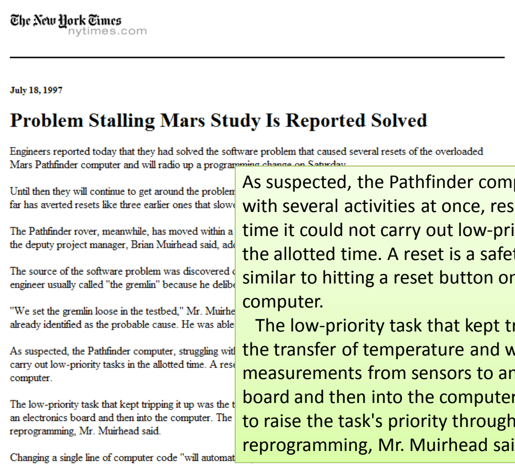
Solutions?

Priority Inheritance

If a low priority task holds a resource needed by a high priority task, the low priority task temporarily inherits the high task's priority

Priority Ceilings

Associate minimum priorities with resources: only a high priority task can acquire the lock on an important resource



The New York Times
nytimes.com

July 18, 1997

Problem Stalling Mars Study Is Reported Solved

Engineers reported today that they had solved the software problem that caused several resets of the overloaded Mars Pathfinder computer and will radio up a programming change on Saturday.

Until then they will continue to get around the problem
far has averted resets like three earlier ones that slow

The Pathfinder rover, meanwhile, has moved within a
the deputy project manager, Brian Muirhead said, ad

The source of the software problem was discovered
engineer usually called "the gremlin" because he delib

"We set the gremlin loose in the testbed," Mr. Muirhe
already identified as the probable cause. He was able

As suspected, the Pathfinder computer, struggling wit
carry out low-priority tasks in the allotted time. A res
computer.

The low-priority task that kept tripping it up was the t
an electronics board and then into the computer. The
reprogramming, Mr. Muirhead said.

Changing a single line of computer code "will automat

As suspected, the Pathfinder computer, struggling with several activities at once, reset itself each time it could not carry out low-priority tasks in the allotted time. A reset is a safety feature similar to hitting a reset button on a home computer.
 The low-priority task that kept tripping it up was the transfer of temperature and wind measurements from sensors to an electronics board and then into the computer. The solution is to raise the task's priority through some reprogramming, Mr. Muirhead said.

Charge

- Computers are single-threaded (or 2/4/8+-threaded) machines that provide their owner the illusion of infinite threads.
- Brains are massively multi-threaded machines that provide their owner with the illusion of a single thread.

```
Thread work = new Thread (project);
work.setPriority (Thread.MAX_PRIORITY);
work.start ();
```