# Real-Time Operating Systems (0_KRI) Course Presentation

Ivan Cibrario Bertolotti

IEIIT-CNR / Politecnico di Torino

Academic Year 2006-2007

# Outline

# About the Teacher


| | | |
|---|---|---|
| Name | : | Ivan Cibrario Bertolotti |
| Address | : | IEIIT-CNR / Politecnico di Torino<br>C.so Duca degli Abruzzi, 24<br>10129 - Torino |
| Phone | : | 011 564-5426 |
| Fax | : | 011 564-5429 |
| Email | : | **ivan.cibrario@polito.it** |
| Study Advice | : | on Tuesday, 14-18 or by appointment<br>“edifici elettrici”, 4$^{th}$ floor,<br>in front of the “M. Boella” E.E. library |

# Course Presentation

## Main goals

This course describes the architecture and the interface of real-time operating systems (RTOS) and introduces the main models and algorithms being used for real-time scheduling and scheduling analysis. A discussion about several RTOS implementation techniques concludes the course.

Moreover, we will look at:

- Real-time concurrent programming techniques.
- IEEE Std 1003.1 (POSIX) international standards.
- Examples and case-studies by means of the RTEMS RTOS.

# Course Presentation

## Main goals

This course describes the architecture and the interface of real-time operating systems (RTOS) and introduces the main models and algorithms being used for real-time scheduling and scheduling analysis. A discussion about several RTOS implementation techniques concludes the course.

Moreover, we will look at:

- Real-time concurrent programming techniques.
- IEEE Std 1003.1 (POSIX) international standards.
- Examples and case-studies by means of the RTEMS RTOS.

# Course Presentation

## Main goals

This course describes the architecture and the interface of real-time operating systems (RTOS) and introduces the main models and algorithms being used for real-time scheduling and scheduling analysis. A discussion about several RTOS implementation techniques concludes the course.

Moreover, we will look at:

- Real-time concurrent programming techniques.
- IEEE Std 1003.1 (POSIX) international standards.
- Examples and case-studies by means of the RTEMS RTOS.

# Course Presentation

## Main goals

This course describes the architecture and the interface of real-time operating systems (RTOS) and introduces the main models and algorithms being used for real-time scheduling and scheduling analysis. A discussion about several RTOS implementation techniques concludes the course.

Moreover, we will look at:

- Real-time concurrent programming techniques.
- IEEE Std 1003.1 (POSIX) international standards.
- Examples and case-studies by means of the RTEMS RTOS.

# Advanced Topics

- Lock and wait-free synchronization techniques. Theory and implementation with and without hardware assistance.
- Microkernel-based operating systems. Scheduling, memory model and IPC in the J. Liedtke's L4 OS.
- Effects of the execution environment on real-time execution characteristics.

# Prerequisites

Main prerequisites:

- Basic programming techniques.
- Good knowledge of the C programming language.
- General notions on computer architecture.
- General-purpose operating systems theory and implementation.

At the beginning of the course, short refresher lessons on:

- Definition, classification and examples of real-time systems.
- Operating systems architecture and structure.
- System calls, trap and interrupt handing mechanisms.
- Memory and I/O device management.

# Prerequisites

Main prerequisites:

- Basic programming techniques.
- Good knowledge of the C programming language.
- General notions on computer architecture.
- General-purpose operating systems theory and implementation.

At the beginning of the course, short refresher lessons on:

- Definition, classification and examples of real-time systems.
- Operating systems architecture and structure.
- System calls, trap and interrupt handing mechanisms.
- Memory and I/O device management.

# Main Learning Materials

## Textbook

A. Burns, A. Wellings,
*Real-Time Systems and Programming Languages*,
3rd edition, 2001, Pearson Education (formerly Addison Wesley),
ISBN: 0-201-72988-1.

## Course Slides

For each lesson, one or more sets of course slides are freely available for download, from the “portale della didattica” web site, at:
`http://didattica.polito.it/`.

# Main Learning Materials

## Textbook

A. Burns, A. Wellings,
*Real-Time Systems and Programming Languages*,
3rd edition, 2001, Pearson Education (formerly Addison Wesley),
ISBN: 0-201-72988-1.

## Course Slides

For each lesson, one or more sets of course slides are freely available for download, from the “portale della didattica” web site, at:
**`http://didattica.polito.it/`**.

# Additional Materials & Further Readings (I)

The following books, journal and conference papers will be used to complement the textbook; they are useful as further readings, too:

G. C. Buttazzo,

*Hard real-time computing systems. Predictable scheduling algorithms and applications*,

2nd edition, 2005, Springer, ISBN: 0-387-23137-4.

C. L. Liu and J. W. Layland,

Scheduling algorithms for multiprogramming in a hard-real-time environment,

*Journal of the ACM*, 20(1):46–61, 1973.

# Additional Materials & Further Readings (II)


L. Sha, R. Rajkumar, and J. P. Lehoczky,

Priority inheritance protocols: an approach to real-time synchronization,

*IEEE Transactions on Computers*, 39(9):1175–1185, 1990.

L. Sha, R. Rajkumar, and S. S. Sathaye,

Generalized rate-monotonic scheduling theory: a framework for developing real-time systems,

*Proceedings of the IEEE*, 82(1):68–82, 1994.

J. Liedtke,

On microkernel construction,

In *Proceedings of the 15th ACM Symposium on Operating System Principles (SOSP-15)*, Copper Mountain Resort, CO, 1995.

# Introductory Materials

The following textbook will be used for the refresher lessons:

A. S. Tanenbaum,

*Modern Operating Systems*,

2nd edition, 2001, Prentice Hall, ISBN: 0-13-092641-8.

For a short overview of real-time embedded operating systems and of the POSIX standards, look for example at:



I. Cibrario Bertolotti,

Internal Real-Time Embedded Operating Systems: Standards and Perspectives,

In *The Embedded Systems Handbook*, 2005, CRC Press, ISBN: 0-8493-2824-1.

# Introductory Materials

The following textbook will be used for the refresher lessons:

A. S. Tanenbaum,

*Modern Operating Systems*,

2nd edition, 2001, Prentice Hall, ISBN: 0-13-092641-8.

For a short overview of real-time embedded operating systems and of the POSIX standards, look for example at:

I. Cibrario Bertolotti,

Internal Real-Time Embedded Operating Systems: Standards and Perspectives,

In *The Embedded Systems Handbook*, 2005, CRC Press, ISBN: 0-8493-2824-1.

# Exam Rules

- The full exam is made up of two parts: a written exam, followed by an oral exam.
- The written exam is mandatory, lasts 2 hours, and typically consists of:
  - two practical exercises, one of them often related to real-time concurrent programming;
  - two open-answer, theoretic questions.
- Since the written exam is closed book, you cannot bring with you and peruse textbooks, course notes, and any other materials while taking it.

You pass the written exam if your score $W$ is $W \geq 15/30$.

# Exam Rules

- The full exam is made up of two parts: a written exam, followed by an oral exam.
- The written exam is mandatory, lasts 2 hours, and typically consists of:
  - two practical exercises, one of them often related to real-time concurrent programming;
  - two open-answer, theoretic questions.
- Since the written exam is closed book, you cannot bring with you and peruse textbooks, course notes, and any other materials while taking it.

You pass the written exam if your score $W$ is $W \geq 15/30$.

# Exam Rules

- The full exam is made up of two parts: a written exam, followed by an oral exam.
- The written exam is mandatory, lasts 2 hours, and typically consists of:
  - two practical exercises, one of them often related to real-time concurrent programming;
  - two open-answer, theoretic questions.
- Since the written exam is closed book, you cannot bring with you and peruse textbooks, course notes, and any other materials while taking it.

You pass the written exam if your score $W$ is $W \geq 15/30$.

# Exam Rules

- The full exam is made up of two parts: a written exam, followed by an oral exam.
- The written exam is mandatory, lasts 2 hours, and typically consists of:
  - two practical exercises, one of them often related to real-time concurrent programming;
  - two open-answer, theoretic questions.
- Since the written exam is closed book, you cannot bring with you and peruse textbooks, course notes, and any other materials while taking it.

You pass the written exam if your score $W$ is $W \geq 15/30$.

# What's Next?

- During the oral examination session, you can look at your written examination paper and ask questions about my corrections, if any.
- The oral exam is optional and is taken when either the student or the teacher asks for it, provided you passed the written exam.
- You are encouraged to take the oral exam if your written exam score was mildly unsatisfactory, that is, $15 \leq W < 18$.

If you do not take the oral exam, you pass the full exam with the same score $W$ you obtained in the written exam.

# What's Next?

- During the oral examination session, you can look at your written examination paper and ask questions about my corrections, if any.
- The oral exam is optional and is taken when either the student or the teacher asks for it, provided you passed the written exam.
- You are encouraged to take the oral exam if your written exam score was mildly unsatisfactory, that is, $15 \leq W < 18$.

If you do not take the oral exam, you pass the full exam with the same score $W$ you obtained in the written exam.

# What's Next?

- During the oral examination session, you can look at your written examination paper and ask questions about my corrections, if any.
- The oral exam is optional and is taken when either the student or the teacher asks for it, provided you passed the written exam.
- You are encouraged to take the oral exam if your written exam score was mildly unsatisfactory, that is, $15 \leq W < 18$.

If you do not take the oral exam, you pass the full exam with the same score $W$ you obtained in the written exam.

# What's Next?

- During the oral examination session, you can look at your written examination paper and ask questions about my corrections, if any.
- The oral exam is optional and is taken when either the student or the teacher asks for it, provided you passed the written exam.
- You are encouraged to take the oral exam if your written exam score was mildly unsatisfactory, that is, $15 \leq W < 18$.

If you do not take the oral exam, you pass the full exam with the same score $W$ you obtained in the written exam.

# Important Remarks

- Exam results are always recorded, regardless of the score.
- You can resign from either the written or the oral exam only before it ends and its score is expressed.
- The oral exam does not necessarily add up to the score of the written exam.
- If you don't show up for the oral examination session either without informing the teacher beforehand, or without a serious and documentable reason, you implicitly waive your right to take the oral exam. The score $W$ you obtained in the written exam will be recorded anyway.

If your final exam score $S$ is $15 \leq S < 18$, the exam may be considered "incomplete". Be sure to fully understand what this means, by carefully reading the Student's Guide.

# Important Remarks

- Exam results are always recorded, regardless of the score.
- You can resign from either the written or the oral exam only before it ends and its score is expressed.
- The oral exam does not necessarily add up to the score of the written exam.
- If you don't show up for the oral examination session either without informing the teacher beforehand, or without a serious and documentable reason, you implicitly waive your right to take the oral exam. The score $W$ you obtained in the written exam will be recorded anyway.

If your final exam score $S$ is $15 \leq S < 18$, the exam may be considered "incomplete". Be sure to fully understand what this means, by carefully reading the Student's Guide.

# Important Remarks

- Exam results are always recorded, regardless of the score.
- You can resign from either the written or the oral exam only before it ends and its score is expressed.
- The oral exam does not necessarily add up to the score of the written exam.
- If you don't show up for the oral examination session either without informing the teacher beforehand, or without a serious and documentable reason, you implicitly waive your right to take the oral exam. The score $W$ you obtained in the written exam will be recorded anyway.

If your final exam score $S$ is $15 \leq S < 18$, the exam may be considered "incomplete". Be sure to fully understand what this means, by carefully reading the Student's Guide.

# Important Remarks

- Exam results are always recorded, regardless of the score.
- You can resign from either the written or the oral exam only before it ends and its score is expressed.
- The oral exam does not necessarily add up to the score of the written exam.
- If you don't show up for the oral examination session either without informing the teacher beforehand, or without a serious and documentable reason, you implicitly waive your right to take the oral exam. The score $W$ you obtained in the written exam will be recorded anyway.

If your final exam score $S$ is $15 \leq S < 18$, the exam may be considered "incomplete". Be sure to fully understand what this means, by carefully reading the Student's Guide.

# Important Remarks

- Exam results are always recorded, regardless of the score.
- You can resign from either the written or the oral exam only before it ends and its score is expressed.
- The oral exam does not necessarily add up to the score of the written exam.
- If you don't show up for the oral examination session either without informing the teacher beforehand, or without a serious and documentable reason, you implicitly waive your right to take the oral exam. The score $W$ you obtained in the written exam will be recorded anyway.

If your final exam score $S$ is $15 \leq S < 18$, the exam may be considered "incomplete". Be sure to fully understand what this means, by carefully reading the Student's Guide.

# Real-Time Systems and Operating Systems

## Refresher

- Definition, characteristics, classification and examples of real-time systems.
- Structure of a real-time operating system: monolithic and layered systems, microkernel-based systems, virtual machines.
- System calls, trap and interrupt handling mechanisms.
- Memory and I/O device management.

# Real-Time Systems and Operating Systems

## Refresher

- Definition, characteristics, classification and examples of real-time systems.
- Structure of a real-time operating system: monolithic and layered systems, microkernel-based systems, virtual machines.
- System calls, trap and interrupt handling mechanisms.
- Memory and I/O device management.

# Real-Time Systems and Operating Systems

## Refresher

- Definition, characteristics, classification and examples of real-time systems.
- Structure of a real-time operating system: monolithic and layered systems, microkernel-based systems, virtual machines.
- System calls, trap and interrupt handling mechanisms.
- Memory and I/O device management.

# Real-Time Systems and Operating Systems

## Refresher

- Definition, characteristics, classification and examples of real-time systems.
- Structure of a real-time operating system: monolithic and layered systems, microkernel-based systems, virtual machines.
- System calls, trap and interrupt handling mechanisms.
- Memory and I/O device management.

# Real-Time Concurrent Programming

## Refresher

- Notion of process and thread.
- Concurrent execution and its issues: race conditions, deadlock, starvation.
- Classical IPC problems and their pitfalls revisited from the real-time point of view.

# Real-Time Concurrent Programming

## Refresher

- Notion of process and thread.
- Concurrent execution and its issues: race conditions, deadlock, starvation.
- Classical IPC problems and their pitfalls revisited from the real-time point of view.

# Real-Time Concurrent Programming

## Refresher

- Notion of process and thread.
- Concurrent execution and its issues: race conditions, deadlock, starvation.
- Classical IPC problems and their pitfalls revisited from the real-time point of view.

# Real-Time Support in IEEE Std 1003.1 (I)

- Processes and threads.
- Shared variable-based inter-process communication:
  - shared memory by memory mapping,
  - semaphores and mutual exclusion devices,
  - condition variables and relationship with monitors.
- Message passing-based inter-process communication:
  - message queues.
- Clocks and timers.

# Real-Time Support in IEEE Std 1003.1 (I)

- Processes and threads.
- Shared variable-based inter-process communication:
  - shared memory by memory mapping,
  - semaphores and mutual exclusion devices,
  - condition variables and relationship with monitors.
- Message passing-based inter-process communication:
  - message queues.
- Clocks and timers.

# Real-Time Support in IEEE Std 1003.1 (I)

- Processes and threads.
- Shared variable-based inter-process communication:
  - shared memory by memory mapping,
  - semaphores and mutual exclusion devices,
  - condition variables and relationship with monitors.
- Message passing-based inter-process communication:
  - message queues.
- Clocks and timers.

# Real-Time Support in IEEE Std 1003.1 (I)

- Processes and threads.
- Shared variable-based inter-process communication:
  - shared memory by memory mapping,
  - semaphores and mutual exclusion devices,
  - condition variables and relationship with monitors.
- Message passing-based inter-process communication:
  - message queues.
- Clocks and timers.

# Real-Time Support in IEEE Std 1003.1 (II)

- Asynchronous notifications and signals.
- Thread-specific data and cleanup handlers.
- Atomic actions (outline).
- Cancellation requests and their effects.
- Programming examples.

# Real-Time Support in IEEE Std 1003.1 (II)

- Asynchronous notifications and signals.
- Thread-specific data and cleanup handlers.
- Atomic actions (outline).
- Cancellation requests and their effects.
- Programming examples.

# Real-Time Support in IEEE Std 1003.1 (II)

- Asynchronous notifications and signals.
- Thread-specific data and cleanup handlers.
- Atomic actions (outline).
- Cancellation requests and their effects.
- Programming examples.

# Real-Time Support in IEEE Std 1003.1 (II)

- Asynchronous notifications and signals.
- Thread-specific data and cleanup handlers.
- Atomic actions (outline).
- Cancellation requests and their effects.
- Programming examples.

# Real-Time Support in IEEE Std 1003.1 (II)

- Asynchronous notifications and signals.
- Thread-specific data and cleanup handlers.
- Atomic actions (outline).
- Cancellation requests and their effects.
- Programming examples.

# Real-Time Scheduling Models and Algorithms

- Scheduling models and their implementation techniques:
  - cyclic executive,
  - rate monotonic (RMS),
  - earliest deadline first (EDF).
- Preemptive versus non-preemptive schemes.

# Real-Time Scheduling Models and Algorithms

- Scheduling models and their implementation techniques:
  - cyclic executive,
  - rate monotonic (RMS),
  - earliest deadline first (EDF).
- Preemptive versus non-preemptive schemes.

# Scheduling Analysis

- Utilization-based schedulability tests for RMS and EDF.
- Response time analysis for fixed priority schedulers (full) and EDF (outline).
- Handling of sporadic and aperiodic processes (outline).

# Scheduling Analysis

- Utilization-based schedulability tests for RMS and EDF.
- Response time analysis for fixed priority schedulers (full) and EDF (outline).
- Handling of sporadic and aperiodic processes (outline).

# Scheduling Analysis

- Utilization-based schedulability tests for RMS and EDF.
- Response time analysis for fixed priority schedulers (full) and EDF (outline).
- Handling of sporadic and aperiodic processes (outline).

# Real-Time Scheduling and IPC

- Modeling process interactions and blocking.
- The priority inversion problem and its solutions:
  - priority inheritance,
  - priority ceiling,
  - priority ceiling emulation protocols.
- Relationship between priority ceiling protocols and deadlock prevention.
- Extension of scheduling analysis to handle blocking.

# Real-Time Scheduling and IPC

- Modeling process interactions and blocking.
- The priority inversion problem and its solutions:
  - priority inheritance,
  - priority ceiling,
  - priority ceiling emulation protocols.
- Relationship between priority ceiling protocols and deadlock prevention.
- Extension of scheduling analysis to handle blocking.

# Real-Time Scheduling and IPC

- Modeling process interactions and blocking.
- The priority inversion problem and its solutions:
  - priority inheritance,
  - priority ceiling,
  - priority ceiling emulation protocols.
- Relationship between priority ceiling protocols and deadlock prevention.
- Extension of scheduling analysis to handle blocking.

# Real-Time Scheduling and IPC

- Modeling process interactions and blocking.
- The priority inversion problem and its solutions:
  - priority inheritance,
  - priority ceiling,
  - priority ceiling emulation protocols.
- Relationship between priority ceiling protocols and deadlock prevention.
- Extension of scheduling analysis to handle blocking.

# RTOS Implementation Techniques

- The RTEMS hardware abstraction layers.
- RTEMS implementation of:
  - context switch,
  - scheduling algorithms,
  - basic synchronization primitives.

# RTOS Implementation Techniques

- The RTEMS hardware abstraction layers.
- RTEMS implementation of:
  - context switch,
  - scheduling algorithms,
  - basic synchronization primitives.

# I/O Device Interface

- Structure of the I/O software in a real-time operating system.
- Device driver interface
- Anatomy of a RTEMS real-time device driver.

# I/O Device Interface

- Structure of the I/O software in a real-time operating system.
- Device driver interface
- Anatomy of a RTEMS real-time device driver.

# I/O Device Interface

- Structure of the I/O software in a real-time operating system.
- Device driver interface
- Anatomy of a RTEMS real-time device driver.

# Lock and Wait-Free Synchronization

## Advanced

- Comparison between lock and wait-free synchronization, and semaphore-based synchronization techniques.
- Suitability and advantages of lock and wait-free synchronization for real-time.
- Implementation with and without hardware assistance.

# Lock and Wait-Free Synchronization

## Advanced

- Comparison between lock and wait-free synchronization, and semaphore-based synchronization techniques.
- Suitability and advantages of lock and wait-free synchronization for real-time.
- Implementation with and without hardware assistance.

# Lock and Wait-Free Synchronization

## Advanced

- Comparison between lock and wait-free synchronization, and semaphore-based synchronization techniques.
- Suitability and advantages of lock and wait-free synchronization for real-time.
- Implementation with and without hardware assistance.

# Microkernel-Based Operating Systems

## Advanced

- Comparison between layered and microkernel-based operating systems.
- The L4 operating system.
- Scheduling, memory model and IPC in L4.

# Microkernel-Based Operating Systems

## Advanced

- Comparison between layered and microkernel-based operating systems.
- The L4 operating system.
- Scheduling, memory model and IPC in L4.

# Microkernel-Based Operating Systems

## Advanced

- Comparison between layered and microkernel-based operating systems.
- The L4 operating system.
- Scheduling, memory model and IPC in L4.

# Execution Environment and Real-Time

## Advanced

- Scheduling models.
- Modeling non-negligible context switch times, interrupt handling, and the real-time clock handler.
- Effects of pipelines and caches on worst-case execution time analysis.

# Execution Environment and Real-Time

## Advanced

- Scheduling models.
- Modeling non-negligible context switch times, interrupt handling, and the real-time clock handler.
- Effects of pipelines and caches on worst-case execution time analysis.

# Execution Environment and Real-Time

## Advanced

- Scheduling models.
- Modeling non-negligible context switch times, interrupt handling, and the real-time clock handler.
- Effects of pipelines and caches on worst-case execution time analysis.