TDDD7 – Real-time Systems Lecture 10: Real-time OS and wrapping up

Simin Nadjm-Tehrani

Real-time Systems Laboratory Department of Computer and information Science



Reading Material

- Baskiyar (2005): main requirements and some examples not so contemporary!
- Parts of Ch. 2 in Carlsson et al. may be useful as complementary text depending on your OS background
- Use the lecture slides as a guide, and for more details use a reference book e.g. chapter 10 in https://link.springer.com/book/10.1007/978-3-031-28701-5)



What have we learnt so far?



Summary of the course

During the course we have ...

- Studied methods for allocation of CPU as a resource
 - Hard real-time systems: Three scheduling algorithms (Cyclic, RMS, EDF)
 - Soft real-time systems: Data centre scheduling (adaptation wrt load and energy optimisation)



Summary of the course

- We further looked at sharing multiple resources:
 - Single CPU case: Potential for deadlocks and starvation, a prevention technique (ICP)
 - Cloud CPUs & energy: Virtual machines
- As well as sharing resources in networked applications
 - Communication bus in hard real-time systems, dedicated applications (CAN vs. TTP scheduling)
 - Bandwidth, buffer space in IP/Multimedia networks



Different requirement types

- From "every deadline met" to QoS expressions
- Who enforces predictability?
 - **RTOS** in single CPU, Bus protocol in dist. system
 - Admission controllers, Packet schedulers, Buffer managers in soft real-time networked systems
- Relation between dependability and predictability
 - Faults models
 - Availability and some threats against it
 - Host crashes, Software process crashes, transients
 - Eliminating faults at design stage



The industry perspective

Ericsson speaker (Blas) talked about:

- Fast vs. Critical, Key Performance Indicators
 - Availability & accessibility
 - Degrees of reliability
- Moving from dedicated HW to COTS and virtualisation
- Replication of microservices (losing capacity but not capability)
- Uniform HW for simplicity (=no diversity!)



The industry perspective

- Move from special purpose hardware to COTS
 - Makes benchmarking software from different vendors possible
 - Hard to make availability claims due to third party software on the same node
- Different performance indicators
- Core isolation for predictability



Real-time operating systems

An overview



This lecture: Real-time OS







Variations remain

• The 20-year perspective:

https://www.embedded.com/how-embedded-softwaredevelopment-has-evolved-over-20-years/

Shows a clear trend from commercial OS to open source OS (see Figure 5)



Sector interoperability

• Keeping competitive advantage but sharing interfaces (Application APIs)

– AUTOSAR: 20 years in 2023

https://www.autosar.org/standards/classic-platform

- Did you look at the Toyoto Oklahoma court ruling from 2013?
 - Mentioned OSEK standard and Misra-C



Linux alternatives

- Co-kernel approach
 - e.g. Xenomai.org
- Running PREEMPT-RT Linux (patch) extensions enabled for support of real-time services
- See the overview (figure 1) in section 2, Reghenzani et al. 2019

https://doi.org/10.1145/3297714



Game changer?

November 2023

- Microsoft makes its RTOS ThreadX open source through eclipse foundation
- https://www.embedded.com/what-open-sourceazure-rtos-means-to-developers/



Let's start with basics



Main functions of an OS

- Task management
- Inter-task communication and synchronisation

From Nanokernels

to Kernels

- Timer services
- Interrupt services
- Memory management (DMA)
- Device I/O management



Task management

May be:

- Time-driven
 - At each tick of a clock the kernel checks if some tasks need to be queued, a task should start to run, or a task should stop running
- Event-driven
 - When an I/O operation is completed, or a task signals completion, the kernel checks ...



Task attributes

- On creation of threads RT kernels allow specification of attributes such as
 - Start time
 - Deadline
 - Priority

. . .

• Used for releasing, aborting, and scheduling



Event-based task switching

Upon arrival of an event:

- Determine whether the current running task should continue (based on scheduling policy)
- If not, determine the next task to be run
- Save the environment of the preempted task
- Prepare the selected task to be run

... in deterministic time!



Basic operations

- Search, insert and delete tasks in ready queue
- Restore the state of the highest priority task

Recall our assumption of zero overhead!



Main functions of an OS

- Task management
- Inter-task communication and synchronisation
- Timer services
- Interrupt services
- Memory management (DMA)
- Device I/O management



Task communication

- Shared memory and semaphores
 - Priority inversion
 - Deadlocks
- Message passing
 - Can above problems still arise?
- Deterministic time:
 - Locking and unlocking latency
 - Message passing delays

Can be made size-independent!



uses inheritence

to avoid it!



Main functions of an OS

- Task management
- Inter-task communication and synchronisation
- Timer services
- Interrupt services
- Memory management (DMA)
- Device I/O management



Time services

- Even event-driven OS need time services to construct timers and detect overruns
- OS may allow defining one or more high resolution clocks with get_value, set_value, get_resolution operations
- Timers can be defined to **signal** an event towards application processes after a particular period
- Real-time POSIX allows queued signals according to priority





Main functions of an OS

- Task management
- Inter-task communication and synchronisation
- Timer services
- Interrupt services
- Memory management (DMA)
- Device I/O management



Interrupt services

When an interrupt event is generated:

- State of the current running task is saved
- Interrupt handler Interrupt Service Routine (ISR)
 is executed
- Next task to be run (application task or the scheduler) is invoked



Deterministic interrupts

- RTOS vendor has to provide data on the timing determinism for the given steps
- Interrupt service routines (ISR) have to be short
- But also predictable!
- If several interrupts are to be serviced, the relative (fixed) priority determines the order



Main functions of an OS

- Task management
- Inter-task communication and synchronisation
- Timer services
- Interrupt services
- Memory management (DMA)
- Device I/O management



Memory allocation

Different views:

- Real-time programmer should have absolute control over resources used by the program
 - Dynamic memory allocation (malloc, free) not supported by RTOS
- Dynamic memory allocation takes non-deterministic time due to fragmentation and should be replaced by other mechanisms
 - Pools instead of heap
 - Control over garbage collection (GC) is an ingredient in real-time Java



Can GC be deferred?

- Swedish company Cinnober specialises Java programs to reduce non-deterministic access time when performing stock market transactions
- Uses replicated servers
- "In most cases garbage collection will not take place in both servers at the same time."

Computer Sweden 5 Dec 2013

https://www.idg.se/2.2683/1.536623/svenskar-bakom-java-genombrott



File system

- Traditional file systems not suitable for real-time OS!
- Those that support filing services also provide a mechanism for efficient locking of task data into main memory storage
- Avoiding unpredictable memory swaps!



Device controllers

- Initialise device interrupt information and disable/enable a device interrupt
- Upon generation of a hardware interrupt identify which device is involved
- Managing interrupt-driven I/O can be difficult unless the number of generated interrupts can be bounded



In addition to OS function and timing...



Other constraints

- So far, we only looked at timing constraints
- A general requirement for RTOS is low overhead, also from a memory *footprint* point of view
- Smallest embedded systems profile of POSIX.13, PSE51 can be written in thousands of lines of code
- Compare to Windows XP: ~30GB!
- Linux first release 10000 loc (2012:15m, 2022: 35m!)



Finally...

- Thank you for your attention!
- Hope you are better performance engineers and resource management experts after this course!





http://www.ida.liu.se/~TDDD07/

