TECHNISCHE UNIVERSITÄT DRESDEN

Informatik Systemarchitektur, Operating Systems

Real-Time Operating Systems

Dresden, 22.01.2008

TECHNISCHE UNIVERSITÄT DRESDEN

Outline

- Real-Time OS Introduction
  - Basic Variants of RTOSes
  - Real-Time Paradigms
  - Non-Real-Time on RTOS
  - Requirements for RTOSes
- Scheduling
- Modern Hardware
- Memory Management
- Real-Time OS Examples
  - POSIX and Real-Time
  - DROPS: Dresden Real – Time OS
  - ARINC 653-1 Standard for Avionics

TECHNISCHE UNIVERSITÄT DRESDEN

Basic Variants of RTOSes (Simplest)

Cyclic Executive
- Only one *task* as infinite loop
- Time driven, polling for external events

Set of Interrupt Handlers
- Event driven
- Handlers usually have priorities

TECHNISCHE UNIVERSITÄT DRESDEN

Basic Variants of RTOSes II

Thread Packages (iRMX, FreeRTOS, eCos, ...)
- Use a form of *scheduling*
  - *Preemptive* or *cooperative*
  - *Priorities*
- Provide synchronization primitives (e.g., semaphores)
  - Some with priority inheritance/ceiling
- No address-space protection, no virtual memory

Basic Variants of RTOSes III

Microkernels (QNX, VxWorks, Fiasco ...)
- Memory protection (address spaces)
  - With or Without virtual memory
  - ➢ More robustness (fault isolation)
- Extensive Functionality provided by services atop

Basic Variants of RTOSes IV

Monolithic RTOS
(LynxOS, MontaVista Linux ...)
- Monolithic kernel with RT API (POSIX RT ...)
- Often non-real-time APIs as well (e.g., Linux compatibility)
- Device drivers etc. usually run in privileged mode
- Real-time applications usually run in user mode

Monolithic with RT Executive underneath
(RTLinux, RTAI, XtratuM ...)
- Combination of a legacy OS with some form of an RT thread package
- Real-time applications run in kernel mode

Real-Time Paradigms (I)

Time Driven
- Static partitioning in time slots
- Scheduler dispatches time slots in a fixed fashion (e.g., fixed cyclic scheduler)

Event Driven
- Events: Messages, Signals, Interrupts...
- Priorities

Real-Time Paradigms (II)

Partitioned System (Time Driven)
- All resources are statically allocated to the partitions (CPU, Memory, Devices...)
- ➔ Space and time isolation
- Multiple threads/processes in a partition
- Scheduling:
  - Partitions: Time driven (see Slide 5)
  - Threads/Processes: any local scheduling scheme possible
- Arinc 653-1 standard for avionics

Non-Real-Time on RTOS

Non-RT API on RT Kernel:
- Unix emulation on QNX
- Linux emulation on LynxOS

Run Non-RT OS on RT Kernel:
- Xtratum, Windows-NT, RT-MACH, L4Linux on DROPS

Additional Resources

Resources besides CPU and Memory
- Disk, Video...
- ➔ Addressed for example in Linux/RK, DROPS

Requirements for RT OSes

Time as "First Class Citizen"
- Periodic processes or absolute timeouts
- Interface: `clock_gettime, clock_getres`
- High clock resolution necessary
  - Special CPU event counters
  - Non-periodic timers (dynamic ticks in Linux)
- Time synchronization

Real-Time Scheduling (I)

Fixed Priorities
- Sufficient priority levels (e.g., RMS 256 prios [1])
- Protocols to avoid **Priority Inversion**
- Events/Messages with priorities
  - Higher priority events arrive first
  - On some systems priority is donated to the receiver
- Signals are queued (predictability)

Real-Time Scheduling (II)

Dynamic Priorities

- Application based: `set_priority(p)`
  - Good for mode changes
  - Not suitable for EDF
- OS driven EDF scheduling

What if processes abuse their priorities?
Overload situations?

Real-Time Scheduling (III)

Periodic Threads and Time Quanta

- Scheduling
  - Assign budgets per period to threads:
    Thread -> (period, priority, budget)
- Control overuse of budgets
  - Periodic threads as first class object (Fiasco)
  - Watchdog timers to signal budget overruns

Interrupt Latencies

One key property of RTOSes:

- **Predicatble and low interrupts latency**

Interrupt latency reduction:

- No interrupt blocking for synchronization (preemptivity)
- Short interrupt service routines (“top halves”)
- Schedule more complex interrupt handling in a thread like fashion
- Partition data and instruction caches
- ...

Avoid Priority Inversion

Priority Ceiling

- Set priority of lock
- Critical sections as parameter for process creation

Priority Inheritance

- Borrowing of CPU time (priority)

Non-preemptive critical sections

Real-Time on Modern Hardware

Increasing unpredictability through
- TLBs (MMU Caches)
- Caches
- Pipelining (write buffers)
- Multi-Master Busses
- "Intelligent" Devices

Real-Time and Memory Management

Avoid demand paging/swaping
(disk access is orders of magnitude slower than main memory)

However:
- Address space isolation useful for robustness/debugging
- Some scenarios need paging

➢ **Interface**
- `mlock(...)` lock pages in memory (prevent swaping)
- `munlock(...)` allow demand paging again

Real-Time and Memory Management

Static Memory Allocation
- Always good for real time
- Inflexible

Dynamic Memory Management
- Use real-time capable memory allocator (e.g., [3] TLSF)

Long I/O Operations

Use asynchronous I/O
Example: POSIX (IEEE Std 1003.1)
- Initiate I/O
  - `aio_read(struct aiocb *aiocbp)`
  - `aio_write(struct aiocb *aiocbp)`
- POSIX Signals for completion
- `aio_suspend(...)` to wait for completion

Long I/O Operations (II)

```
struct aiocbp {
    int    aio_filedes;  /* file descriptor */
    off_t  aio_offset;   /* absolute file offset */
    void   *aio_buf;     /* pointer to memory buffer */
    size_t aio_nbytes;   /* number of bytes to I/O */
    int    aio_reqprio;  /* prio of request */
    struct sigevent aio_sigevent; /* signal */
    int    aio_lio_opcode; /* opcode for lio_listio */
};
```


POSIX and Real-Time

- POSIX (Portable OS Interface): IEEE 1003.1 REALTIME extensions
  - Semaphores
  - Process Memory Locking
  - Priority Scheduling
  - Realtime Signal Extension
  - Clocks/Timers
  - Interpreocess Communication
  - Syschronized I/O
  - Asynchronous I/O

POSIX: Memory Locking

- Memory ranges can be locked (excluded from swaping)
- Provide latency guarantees for memory accesses

POSIX: Realtime Scheduling

- Multiple scheduling policies
  - SCHED_FIFO (non-preemptive FIFO)
  - SCHED_RR (preemptive/time-sliced FIFO)
  - SCHED_SPORADIC (2 prio levels, replenishment interval, and budget, FIFO on active priority level)
  - SCHED_OTHER (threads without RT policy)
- At least 32 RT priorities

POSIX: Realtime Signals

Difference to non-realtime signals:
- Queued (for the same number)
- Carry user data
- Ordered delivery

Specific Properties
- RT Signals are in the range SIGRTMIN to SIGRTMAX
- Handler gets `siginfo_t` with additional data
- Lowest pending signal is delivered first

POSIX: Realtime Timers/Clocks (I)

Clocks
- Min. resolution of 20ms (clock_getres())
- Multiple clocks
  - CLOCK_REALTIME (wall clock time)
  - CLOCK_MONOTONIC (system-wide monotonic clock)
  - CLOCK_PROCESS_CPUTIME_ID
  - CLOCK_THREAD_CPUTIME_ID
- Per process timers (generate RT signals)
- Periodic timers supported (struct itimerspec)

POSIX: Realtime Timers/Clocks (II)

Timers
- Associated to a specific clock (see Clocks)
- Per process timers (generate RT signals)
- Periodic timers supported (struct itimerspec)

POSIX: Execution Time Monitoring

Clocks measuring thread/process execution time
- CLOCK_PROCESS_CPUTIME_ID
- CLOCK_THREAD_CPUTIME_ID

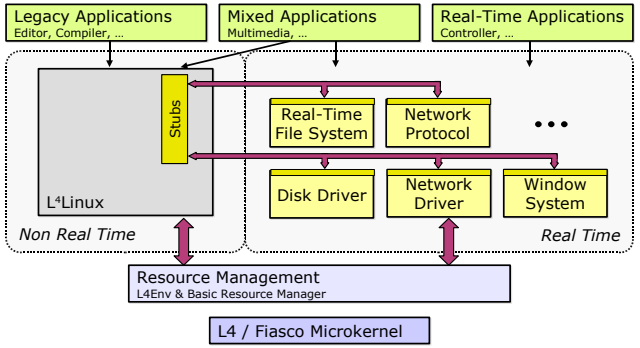
Timers connected to these clocks
- Signal deadline misses

POSIX: Asynchronous I/O

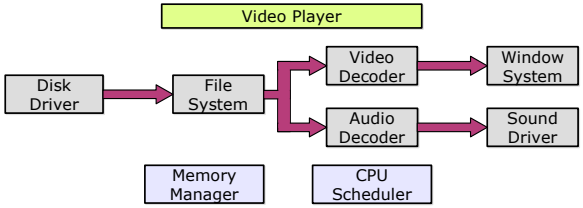
Explicitly overlap I/O operations and processing

- See Asynchronous I/O before

DROPS: Dresden Real – Time OS



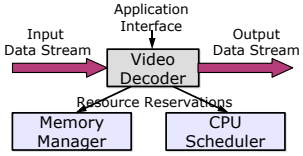
DROPS – Real-Time Application Model (I)



(RT-)Component-based Applications

- Application sets up and controls chains of (RT-)components (comparable to filter chains)
- Stream based data processing in components
- Real-time streaming based on Data Streaming Interface (DSI)

DROPS – Real-Time Application Model (II)



(RT-)Component need guarantees for resources

- Reserve required amount of resources (memory, max. latency, min bandwidth ...)
  - No "out of memory" etc.

L4/Fiasco Real-Time Scheduling (I)

Periodic mode for Real time execution
- Period defines deadline and minimum refresh interval for real-time scheduling contexts

Multiple scheduling contexts per thread
- Scheduling context is tuple (Priority, Timeslice length) = Reservation

Time slice overrun
- Thread exceeded reserved time quantum (reservation time)

Deadline miss
- Thread exceeded its periodic deadline

L4/Fiasco Real-Time Scheduling (II)

Time slice overrun and Deadline miss
- Signaled via IPC to a special preempter thread

Execution models
- Strictly periodic (constant interrelease times)

- Periodic (minimal interrelease times)

- Sporadic (random interrelease time, hard deadline)
- Aperiodic (random interrelease time, no deadline)

ARINC 653-1 Standard for Avionics

Statically partitioned System (time-driven scheduling)

- Execution in one partition must not influence execution in another partition (strict isolation)
  - Strictly time-driven scheduling of partitions
  - No transfer of idle CPU time among partitions
- Additionally defines
  - System Health Monitoring
  - Intra/Inter-Partition Communication
  - Time Management

Bibliography

[1] **Real-Time Systems** (Jane W. S. Liu, University of Ilinois at Urbana-Champaign, Publisher: Prentice Hall)

[2] **Real-Time Systems: Design Principles for Distributed Embedded Applications (**Hermann Kopetz, Publisher: Kluwer Academic Pub)

[3] **TLSF: A New Dynamic Memory Allocator for Real-Time Systems** (M. Masmano, I. Ripoll, A. Crespo and J. Real). 16th Euromicro Conference on Real-Time Systems (ECRTS 2004).