Windows NT 4
from a real-time perspective

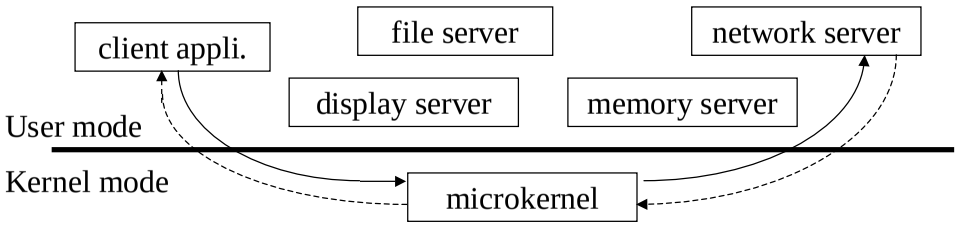
Prof. J.-D. Decotignie
CSEM Centre Suisse d'Electronique et de Microtechnique SA
Jaquet-Droz 1, 2007 Neuchâtel
jean-dominique.decotignie@csem.ch

Outline

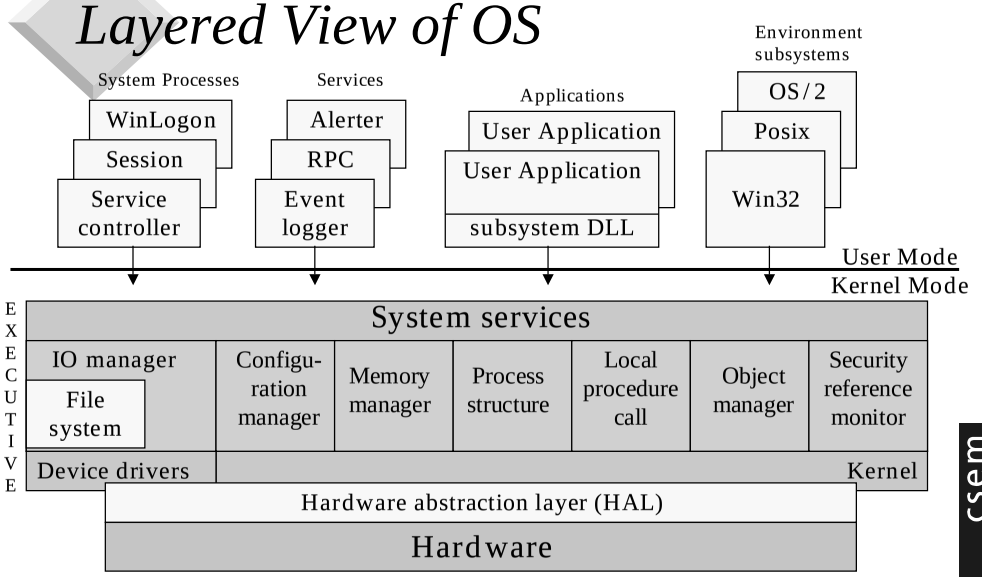
- Architecture
- Tasks and threads
- Memory management
- Interrupt management
- Synchronization and exclusion
- Limitations for real-time applications

Architecture

- Based on client-server model
- Each service implemented as a separate server
- Servers do not inherit client priorities



Layered View of OS



Hardware Abstraction Layer HAL

- Low level interface to hardware
- Hides all HW dependant details
  - IO interfaces
  - Interrupt controllers
  - Multiprocessor configurations
- HAL routines used by Windows NT and by device drivers

Kernel and Device Drivers

- Kernel
  - process and thread scheduling
  - trap handling / exception dispatching
  - interrupt handling and dispatching
  - multiprocessor sync.
  - define base kernel objects used by executive
  - MMU management

Kernel and Device Drivers (2)

- Device Drivers
  - loadable kernel mode modules
  - do not always use HAL
  - 4 types
    - hardware device drivers
    - file system drivers
    - filter drivers
    - network redirectors and servers

Executive

- Contains a number of managers (process /thread, virtual memory, security reference, IO, cache)
- Adds semantics to kernel objects
- 4 main groups of support functions
  - object manager
  - LPC facility
  - set of run time library functions
  - executive support functions

Environment Subsystems

- API seen by application
- 3 different subsystems
  - ✓ WIN32 (richest)
  - ✓ POSIX (first version IEEE 1003.1-1990)
  - ✓ OS2
- A program may only access to a single subsystem

Processes and Threads

- Process
  - ✓ executable unit with its own protected memory space & resources, scheduled based on priority
  - ✓ can create new processes
    - ➢ no parent relationship, no inheritance by default
- Thread
  - ✓ unit of execution within a process
  - ✓ share same code and data space (separate stack)

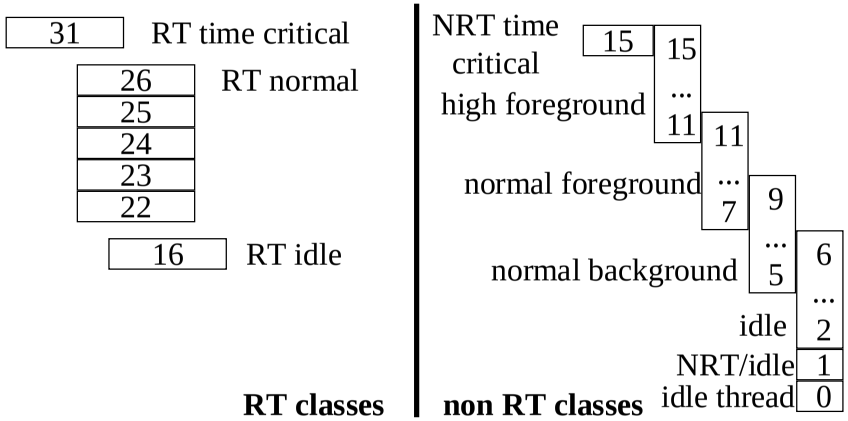
Processes and Threads (2)

- Fibers
  - ✓ “lightweight threads”
  - ✓ cooperative multitasking (coroutines)
  - ✓ created from threads
  - ✓ not scheduled by the system

Processes and Threads Priority

- The higher the value, the higher the priority
- For each thread, priority is based on
  - ✓ process priority class (Idle, Normal, High, RT)
  - ✓ thread priority level within the process (lowest, below_normal, normal, above_normal, highest)
  - ✓ a dynamic boost by the system (except for RT)
- System may change priority (except for RT)

Priority Span



Thread Priority

- Based on process priority class
  - ✓ Idle = 4, Normal = 7 (background) or 9 (foreground), High = 13, Real-time = 24
- With an offset
  - ✓ lowest = -2, below_normal = -1, normal = 0, above_normal = +1, highest = +2
- Or absolute setting
  - ✓ time_critical = 15 or 31, idle = 1 or 16

Priority Mechanism

- Scheduling based on
  - ✓ priority for tasks with different priorities
  - ✓ round robin (time slice) for tasks with same priority
- RT class tasks never get their priority changed
- Priority inversion is dealt by boosting priority of NRT tasks that have not run for long
- Some system calls are handled in lower priority than the calling task

Memory Management

- Paged memory protection with virtual memory
- Pages (4KB) may be locked into memory (seems not working on code pages)
- Drivers may be locked in memory (code+data)
- 2GB of virtual memory / process
- Memory mapped files
- Heaps can be used

Interrupt Management

- Interrupts always gets higher priority than tasks
- Handled in 2 stages
  - ✓ driver level (ISR) doing the necessary minimum
  - ✓ rest of handling done by requesting a DPC (Deferred Procedure Call)
- DPCs are handled in FIFO order
- IRQ are prioritized and may be nested

Deferred Procedure Call (DPC)

- Handled in FIFO order
- Have a higher priority than other tasks
- Some may take up to a few milliseconds (hard disk and network)
- Interrupts have higher priority
- Only a single instance of a DPC may be queued (even if more than one IRQ has occured)

Interrupt Priorities vs Thread

Thread Priorities 0-31			
	31	High	Hardware interrupts
	30	Power fail	
	29	Interprocessor	
	28	Clock	
		Device n	
		...	
		...	
		...	
		Device 1	
	2	Dispatch / DPC	Software interrupts
	1	APC	
	0	Passive	

Timers

- Sleep (duration in ms)
- Waitable Timers
  - ✓ dispatch policy
    - ➢ on shot
    - ➢ periodic
  - ✓ may queue an APC on completion
  - ✓ wait one or all (manual reset) waiting threads
  - ✓ can be shared between processes if named

Synchronization and Exclusion

- Synchronization
  - Shared Variable
    - Mutex (recursive take)
    - Semaphore
    - Critical region (thread in same process)
  - Message based
    - Event
  - State
    - Termination
    - Idle
- Exclusion
  - Busy waiting
  - Mutexes
  - Semaphores
  - Disabling interrupts
  - Tasks (Asynchronous Procedure Call)

Synchronization

- Effect when synchronization object is signaled

Object type	Signaled when	Effect on waiting threads
Process	Last thread terminates	All released
Thread	Thread terminates	All released
File	I/O completed	All released
Event (notification type)	Set by thread	All released
Event (synchronization type)	Set by thread	One released, event reset
Semaphore	Count not equal to 0	All released
Timer	Time arrives or delay expires	All released
Mutex	Released by thread	One released

Communication

- Internal or with Windows machines
  - Pipes
    - named (duplex capable, may be over network)
    - anonymous (simplex, no network)
  - Message queues (only to threads that control a window)
  - Mailslots (unidirectional, not reliable, may be over network, broadcast capability)
  - Shared memory and files

Communication (2)

- Internal or with Windows machines (cont.)
  - Asynchronous Procedure Call (APC)
    - only take place when a thread is waiting
    - is used as call back mechanism for async IO
    - can be invoked manually (single 32 bit data)
  - RPC (internal map to LPC)
- With other machines (windows or not)
  - Sockets
  - RPCs

Synchronization & comm. between drivers & applications

- DPC
- APC
- Shared memory
- Events

Synchronization Objects at kernel level

- Events
- Timers
- Threads
- Mutexes
- Semaphores
- Fast mutexes (not recursive,exec. level only)
- Resources (executive level only)

Temporal Behavior

- Threads at same priority in FIFO order
- Threads waiting on Mutex / semaphores queued in FIFO order
- APC: one FIFO per invoked thread (lower priority than DPCs and IRQ, higher than threads)
- Synchronization event in FIFO
- Threads waiting to enter critical sections in FIFO order

Limitations for Real-Time

- Limited number of priority levels
- Non prioritized interrupt handling (DPC in FIFO & can be interrupted)
- No priority inversion prevention
- Non deterministic behavior
- Servers do not inherit priority from clients

What to Do ?

- Will be discussed in hands on

csem

References

- [Sol98] D. Solomon, “Inside Windows NT” 2nd edition, Microsoft Press, Redmond, 1998, ISBN 1-57231-677-2
- [Bra96] M. Brain, “Win 32 System Services”, Prentice Hall, Upper Saddle River, 1996, ISBN 0-13-324732-5
- [Low97] A. Lowe, “Porting UNIX Applications to Windows NT, Macmillan, Indianapolis, 1997, ISBN 1-57870-004-3
- [Bak97] A. Baker, “The Windows NT Device Driver Book”, Prentice Hall, Upper Saddle River, 1997
- [Ram98] K. Ramamritham et al., “Using Windows NT for Real-Time Applications”, Proc. 4th RTAS, Denver, June 3-5, 1998, pp.102-11
- [Dek99] E. Dekker et al., “Developing Windows NT Device Drivers”, Addison Wesley, Reading, 1999, ISBN 0-201-69590-1

csem