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
- System Processes
  - WinLogon
  - Session
  - Service controller
  - RPC
  - Event logger
- Services
  - Alerter
  - User Application
  - User Application subsystem DLL
- Applications
  - User Application
  - Win32
  - Posix
  - OS/2
- Environment subsystems
- User Mode
- Kernel Mode
- System services
  - IO manager
  - Configuration manager
  - Memory manager
  - Process structure
  - Local procedure call
  - Object manager
  - Security reference monitor
- Device drivers
  - Hardware abstraction layer (HAL)
- Hardware
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**

<table>
<thead>
<tr>
<th>RT Classes</th>
<th>non RT classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT time critical</td>
<td>31</td>
</tr>
<tr>
<td>RT normal</td>
<td>26</td>
</tr>
<tr>
<td>RT high foreground</td>
<td>25</td>
</tr>
<tr>
<td>RT normal foreground</td>
<td>24</td>
</tr>
<tr>
<td>RT normal background</td>
<td>23</td>
</tr>
<tr>
<td>RT idle</td>
<td>22</td>
</tr>
<tr>
<td>RT idle thread</td>
<td>16</td>
</tr>
</tbody>
</table>

**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 occurred)

---

**Interrupt Priorities vs Thread**

<table>
<thead>
<tr>
<th>Priority</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>High</td>
</tr>
<tr>
<td>30</td>
<td>Power fail</td>
</tr>
<tr>
<td>29</td>
<td>Interprocessor</td>
</tr>
<tr>
<td>28</td>
<td>Clock</td>
</tr>
<tr>
<td></td>
<td>Device n</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>Device 1</td>
</tr>
</tbody>
</table>

**Software interrupts**

- Dispatch / DPC
- APC
- 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)

**Effect when synchronization object is signaled**

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

**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, executive 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

References