Windows NT and Real-Time?

- Reading: “Inside Microsoft Windows 2000”, (Solomon, Russinovich, Microsoft Programming Series)
- “Real-Time Systems and Microsoft Windows NT” (MSDN Library)
- “Windows XP with RTX - The off-the-shelf platform for Integrated Communication Equipment” (www.venturcom.com)

Priorities in Windows NT/2000
IO System Components (Windows 2k)

- Applications
- Win32 services
- WMI service
- user-mode PnP manager
- setup components
- .inf files
- .cat files
- registry

I/O system
- WDM WMI routines
- PnP manager
- Power manager
- I/O manager

Device Driver Layering

- File system driver
- Disk driver
- System services
- User mode
- Kernel mode
Device Driver Layering (2)

- **Initialization routine**: This routine initializes hardware and sets up data structures used by the driver at startup time.
- **Interrupt service routine (ISR)**: This routine handles an interrupt on the device that the device driver controls.
- **Deferred processing call (DPC)**: One or more DPCs handle non-time-critical processing for the driver.
- **System thread**: Some, but not all, drivers will have a system thread for very low-priority work.

Primary Device Driver Routines

- NT/2000 device drivers run entirely within the system process and have access to all hardware through the HAL. A typical device driver will have several components:

  - **Dispatch routines**
  - **Start I/O routine**
  - **Add-device routine**
  - **Interrupt service routine**
  - **Initialization routine**
  - **DPC routine**
Control Flow for an IO Operation

Whether to wait depends on overlapped flag

Queueing and Completing a Synchronous Request

1. I/O request passes through subsystem DLL
2. NtWriteFile华人 handle...
3. Create IRP and send it to device driver
4. Handle interrupt and return success or error status
5. Transfer data specified in IRP
6. Perform I/O and interrupt
7. Complete IRP and return success or error status
8. User mode
9. Kernel mode
Servicing a Device Interrupt (only Phase I)

1. The device interrupt 1/0 service.
2. The kernel's interrupt dispatcher transfers control to the device's service routine.
3. The ISR stops the device interrupt and queues a DPC.
4. DPC queue

Servicing a Device Interrupt (Phase II)

5. The DPC routine starts the next I/O request in the device (queue) and then compiles interrupt servicing.
6. Device queue
7. The interrupt dispatcher transfers control to the (user's) DPC routine.
Completing an I/O Request (Phase I)

1. The DPC routine calls the I/O manager to complete the original I/O request.
2. The I/O manager queues an APC to complete the I/O request in the caller's context.

Completing an I/O Request (Phase II)

1. The kernel-mode APC routine wakes the caller's thread, sets the original file handle to the signaled state, queues any user-mode APCs for execution, and disposes of the IRP.
2. The interrupt dispatcher transfers control to the I/O manager's APC routine.
### Priority Levels vs. Interrupt Levels

- The HAL maps hardware-interrupt numbers to IRQLs.
- IRQLs are not the same as IRQs in x86.
- Scheduling priority is attribute of thread, while IRQL is attribute of an interrupt source.
- Lazy IRQL management for slow PICs.
- Code running at DPC/dispatch level or above can’t wait on object if so would necessitate scheduler to invoke another thread.

#### Thread Priorities 0-31
- 31: High
- 30: Power Fail
- 29: Inter-Processor Interrupt
- 28: Clock
- 27: Profile
- 26: Device n
- 3: Device 1
- 2 DPC/dispatch
- 1: APC
- 0: Passive

### Memory Management

- Paging I/O occurs at a lower priority level than the real-time priority process levels. Paging within the real-time process is still free to occur, but this really ensures that background virtual memory management won’t interfere with processing at real-time priorities.
- Windows NT permits an application to lock itself into memory so that it is not affected by paging within its own process. This allows even very large processes (such as raster image processing, where some processes are over 100MB) to lock all their memory down into physical memory and avoid the overhead of paging, while allowing the rest of the system to function normally.
- Windows NT memory management allows for memory mapping, which permits multiple processes—even device drivers and user applications—to share the same physical memory. This results in very fast data transfers between cooperating processes or between a driver and an application. Memory mapping can be used to dramatically enhance real-time performance.
Windows 2000/NT and Real-Time Processing

- Windows 2000/NT does not prioritize device IRQs in controllable way.
- User-level applications execute only when a processor’s IRQL is at passive level.
- System’s devices and device drivers – not the OS – ultimately determine the worst-case delay.
- This is a problem with off-the-shelf hardware and drivers.
- System designer must bound the length of device’s ISR and DPC in the worst case.
- Embedded versions of Windows NT/2000 provide control over memory footprint etc, but are not real-time capable.
- Extensions of real-time kernels can be provided through custom extensions of the HAL.

VenturCom RTX Architecture

Figure 1: RTX Architecture