Scheduling (Win 2K)

CSE 410, Spring 2004
Computer Systems

http://www.cs.washington.edu/education/courses/410/04sp/
Readings and References

• Reading
  » Chapter 6, Section 6.7.2, *Operating System Concepts*, Silberschatz, Galvin, and Gagne

• Other References
  » Chapter 6, Performance Monitoring, *Windows 2000 Professional Resource Kit*, Microsoft
Dispatcher “database”

- Ready queues
- Process
- Thread
- Process
- Thread
- Thread

- Default base priority
- Default processor affinity
- Default quantum

- Base priority
- Current priority
- Processor affinity
- Quantum

- Ready summary
  - 31 ... 0

- Idle summary
  - 31 ... 0

which ready queues have threads?
which processors are idle?

clockres.exe
Thread State Transitions

- **Initialized**
- **Waiting**
- **Ready**
- **Terminated**
- **Running**
- **Standby**

Transition details:
- **Create thread**
- **Place in ready queue**
- **Wait complete**
- **Select**
- **Preempt**
- **Dispatched (context switch and run)**
- **Reinitialize**

States and transitions:
- From **Initialized** to **Waiting**
- From **Waiting** to **Ready**
- From **Running** to **Standby**
- From **Standby** to **Ready**
- From **Ready** to **Running**
- From **Running** to **Terminated**
- From **Terminated** to **Waiting**
- From **Waiting** to **Terminated**

Transition events:
- **Preempt**
- **Select**
- **Wait complete**
- **Dispatched**
- **Reinitialize**

Graphical representation showing state transitions and events.
Ready, Running, Waiting

• Ready
  » ready to run if there is a processor available
  » there is a ready queue for each priority level

• Running
  » has been switched to and is running

• Waiting
  » waiting on an event (synchronize, I/O, etc)
Other States

• Initialized
  » On its way in the door

• Terminated
  » On its way out the door to history or recycle

• Standby
  » Ready and selected to run next

• Transition
  » Ready, but important parts are paged out
Windows 2000 Thread States

7 - Unknown
6 - Transition
5 - Wait (for something to complete)
4 - Terminated
3 - Standby (on-deck circle)
2 - Running (at bat)
1 - Ready (eligible to be selected)
0 - Initialized
cpustres.exe

perfmon4.exe
Thread Priorities

Win2K  Win32 API

Real-time

- Critical: 31
- Real-time: 24
- High: 16
- Above Normal: 15
- Normal: 13
- Below Normal: 10
- Idle: 8
- Below Normal: 6
- Idle: 4

Dynamic

- Critical: 15
- Normal: 10
- Below Normal: 8
- Normal: 6
- Idle: 4

Process Priority Class

- Highest
- Above Normal
- Normal
- Below Normal
- Lowest

Relative Thread Priority

Used for zero page thread, not available for applications
Setting Thread Priorities

• Base priority
  » normally inherited from process default
  » can be explicitly set

• Current priority
  » starts out same as base
  » real time never changes
  » dynamic is boosted when appropriate for responsiveness
Priority boosting

• After I/O completion or event wait
  » you’ve waited for this data, now use it quick

• User response
  » Foreground thread after a wait or window thread wakeup for window event

• CPU starvation
  » found an aging thread on the ready queues

• The boost decays quickly over time
Quantum

- Thread Quantum is
  - indicator of the amount of time a thread can run before W2K checks whether another thread at the same priority should get to run

- Each thread has a current quantum value
  - a small integer that is decremented under various circumstances
  - not an actual length of time, just a number
Quantum value

• Thread quantum is initialized when thread is put on the ready queue
  » initial value of 6 on Windows 2K Professional
  » initial value of 36 on Windows 2K Server

• Quantum of running thread is decremented by 3 after system clock interrupt
  » so a W2K Pro thread can run for 2 clock intervals
  » a W2K Server thread can run for 12 clock intervals
Quantum is reset to initial value

• a thread moves to ready queue after quantum end
  » in other words, a thread is given another chunk of
time to use after it has exhausted the first chunk

• a real-time thread is preempted and moves from
running to ready or it moves from running to wait
  » the presumption is that you are doing a good job of
explicitly managing priorities and access to the CPU
when you are running real-time threads
Quantum changes

• Quantum is decremented
  » reduced quantum => less time remaining before thread has exhausted its time slice
  » reduced by 3 when the clock ticks
  » by 1 when dynamic thread executes a wait

• Quantum initial value may be boosted
  » “Optimize performance for applications”
  => boost initial quantum for foreground threads
Scheduling Scenarios

- Voluntary switch
  - thread calls a wait function of some sort
- Preemption
  - higher priority thread is ready to run
- Quantum end
  - the running thread exhausts its quantum
Voluntary Switch

- Wait Queues
- Running
  - A-14
- Priority Level
  - 14
  - 13
  - 12
  - 11
  - 10
  - 9
- Ready Queues
  - B
  - C
  - D
  - E
  - F

Dynamic - quantum is decremented by 1
RT - quantum is reset to initial value
Preemption

Wait Queues
A
B-12

Running
A-14

Priority Level
... 14 13 12 11 10 9 ...

Ready Queues
... B C D

Dynamic - quantum unchanged
RT - quantum is reset to initial value
Quantum End

Wait Queues

Running

Priority Level
... 14 13 12 11 10 9 ...

Ready Queues

quantum is reset to initial value