Scheduling (Win 2K)

CSE 410 - Computer Systems
November 21, 2001
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

31

... 

0

Ready summary

Idle summary

which ready queues have threads?

which processors are idle?

clockres.exe
Thread State Transitions

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

Transition details:
- **create thread**
- **place in ready queue**
- **wait complete**
- **select**
- **preempt**
- **dispatched (context switch and run)**
- **reinitialize**
- **done**

States:
- Initialized
- Waiting
- Ready
- Running
- Standby

Transitions:
- From Terminated to Initialized
- From Initialized to Ready
- From Ready to Running
- From Running to Standby
- From Standby to Terminated
- From Terminated to Terminated (loop)
- From Ready to Waiting
- From Waiting to Ready
- From Waiting to Terminated
- From Running to Terminated
- From Terminated to Running
- From Running to Standby
- From Standby to Running
- From Standby to Terminated

Events:
- wait
- wait complete
- select
- preempt
- dispatch (context switch and run)
- create thread
- place in ready queue
- reinitialize
- done

History node

Arrows indicate transitions between states.
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
Thread Priorities

Win2K

Real-time

Critical

Idle

Dynamic

Win32 API

Process Priority Class

Relative Thread Priority

Highest
Above Normal
Normal
Below Normal
Lowest

Real-time

High

Above Normal

Normal

Below Normal

Idle

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 $\Rightarrow$ 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”
  - $\Rightarrow$ 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

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

<table>
<thead>
<tr>
<th>Priority Level</th>
<th>Ready Queues</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>B</td>
</tr>
<tr>
<td>13</td>
<td>C</td>
</tr>
<tr>
<td>12</td>
<td>D</td>
</tr>
<tr>
<td>11</td>
<td>E</td>
</tr>
<tr>
<td>10</td>
<td>F</td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Wait Queues</th>
<th>Running</th>
<th>Priority Level</th>
<th>Ready Queues</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C-12</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B-12</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

quantum is reset to initial value