Concurrenty Conclusion

Sarah Diesburg
Operating Systems
CS 3430
Threads and Synchronization

- Better, cleaner, and simpler abstraction to application programmers

<table>
<thead>
<tr>
<th>Programming abstraction</th>
<th>Sequential execution, each with its own CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Semaphores and monitors</td>
</tr>
<tr>
<td>Physical hardware</td>
<td>Single CPU</td>
</tr>
<tr>
<td></td>
<td>Interrupts</td>
</tr>
<tr>
<td></td>
<td><code>test_and_set</code></td>
</tr>
</tbody>
</table>
Since 1985

- Every major OS comes with threads
  - OS X
  - OS/2
  - Windows XP, NT, Vista, 7
  - Linux
  - Solaris
Since 1985

- Major applications are written in threads
  - Word processing
  - Databases
  - Web servers
  - Embedded systems
A Cautionary Tale

- Microsoft OS/2
  - Spectacular failure (IBM re-wrote the whole OS from scratch)
  - Used threads for everything
    - Window systems
    - Communication among programs
Microsoft OS/2

- Created many threads
  - Few are ready to run
  - Most threads wait around for user typing and network packets
  - Since each thread needs to store its own execution stack (running or waiting), OS/2 required $200 extra memory to store those threads
  - $200 for working while printing?
The Moral of the Story…

- Threads are cheap
  - But they are not free