Concurrency Conclusion

Sarah Diesburg

Operating Systems

CS 3430

Threads and Synchronization

- Better, cleaner, and simpler abstraction to application programmers

Programming abstraction	Sequential execution, each with its own CPU Semaphores and monitors
Physical hardware	Single CPU Interrupts **`test_and_set`**

Since 1985

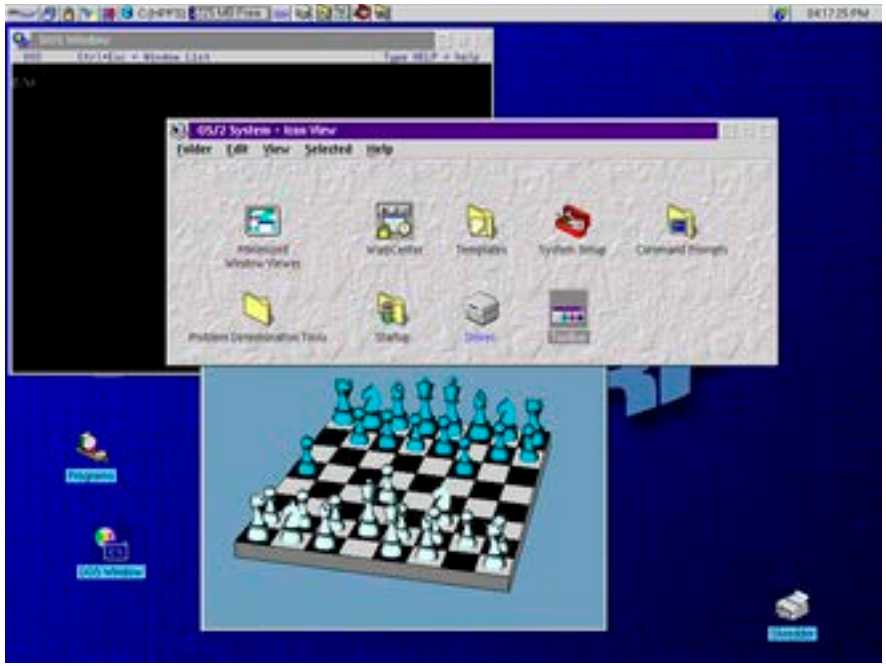
- Every major OS comes with threads
  - OS X
  - OS/2
  - Windows XP, NT, Vista, 7,8,10
  - Linux
  - Solaris

Since 1985

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

A Cautionary Tale

- IBM OS/2
  - https://en.wikipedia.org/wiki/OS/2



A Cautionary Tale

- IBM OS/2
  - 1990
  - 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

New need for threaded programs

- Moore’s Law no longer in effect
  - https://en.wikipedia.org/wiki/Moore's_law
  - Chip performance doubles every 2 years
  - Not true now
- We need to write programs to better take advantage of multiple CPU cores