Concurrency:
Processes, Threads, and Address Spaces

January 27, 2014
Anthony D. Joseph
http://inst.eecs.berkeley.edu/~cs162
Operating System Roles (Recap)

• OS as a Traffic Cop:
  – Manages all resources
  – Settles conflicting requests for resources
  – Prevent errors and improper use of the computer

• OS as a facilitator (“useful” abstractions):
  – Provides facilities/services that everyone needs
  – Standard libraries, windowing systems
  – Make application programming easier, faster, less error-prone

• OS as government:
  – Provides a share of expensive resources as needed
  – Nudges users toward standard interfaces, allowing improved sharing
Migration of OS Concepts and Features
Definitions

• **Threads:**
  – Unit of execution and scheduling
  – Independent Fetch/Decode/Execute loop
  – Operate in an **address space**
Definitions

• **Threads:**
  – Unit of execution and scheduling
  – Independent Fetch/Decode/Execute loop
  – Operate in an *address space*

• **Uniprogramming:** *one thread at a time*

• **Multiprogramming:** *more than one thread at a time*
Definitions

• **Threads:**
  – Unit of execution and scheduling
  – Independent Fetch/Decode/Execute loop
  – Operate in an **address space**

• **Uniprogramming:** *one thread at a time*

• **Multiprogramming:** *more than one thread at a time*

• **Process:** unit of resource allocation and execution
  – Owns memory (address space), file descriptors, file system context, …
  – Encapsulate one or more threads sharing process resources
Definitions

• **Uniprogramming**: *one thread at a time*
• **Multiprogramming**: *more than one thread at a time*

• Virtual “CPU” structure holds:
  – Program Counter (PC), Stack Pointer (SP)
  – Registers (Integer, Floating point, others…?)

• Virtual “CPU” switch:
  – Save PC, SP, and registers in current state block
  – Load PC, SP, and registers from new state block
Definitions

- **Uniprogramming**: one thread at a time
- **Multiprogramming**: more than one thread at a time

- Virtual “CPU” structure holds:
  - Program Counter (PC), Stack Pointer (SP)
  - Registers (Integer, Floating point, others…?)

- Virtual “CPU” switch:
  - Save PC, SP, and registers in current state block
  - Load PC, SP, and registers from new state block

- **Cooperative multiprogramming**: Switch on voluntary yield, I/O, other things
- **Preemptive multiprogramming**: Switch on timer, voluntary yield, I/O, other things
Providing Illusion of Separate Address Space:
Load new Translation Map on Switch

Translation Map 1
Translation Map 2

Physical Address Space
5min Break
Definitions

- **Traditional UNIX Process**: “HeavyWeight Process”
  - A single, sequential stream of execution (one thread) in its *own* address space

```
process state
process number
program counter
registers
memory limits
list of open files
...
```

Process Control Block
Definitions

• **Traditional UNIX Process**: “HeavyWeight Process”
  – A single, sequential stream of execution (one thread) in its *own* address space

• **Scheduling**: Give out CPU time to different processes
  – Only one process “running” at a time

• **Protection**: Give pieces of resources to different processes
  – Controlled access to non-CPU resources
HeavyWeight Process Scheduling

- PCBs move from queue to queue as they change state
  - Decisions about which order to remove from queues are Scheduling decisions
  - Many algorithms possible (few weeks from now)
Inter-Process Communication

- Two mechanisms:
  - Shared-Memory Mapping
    » Accomplished by mapping addresses to common DRAM
    » Read and Write through memory – *complex but low overhead*
  - Message Passing
    » send() and receive() messages across communication channel
    » *Works across global network*
Single and Multithreaded Processes

- Multithreading: *a single program made up of a number of different concurrent activities*
- “Active” threads encapsulate concurrency (“LightWeight Process”)
- “Passive” address spaces encapsulate protection
- Heavyweight Process ≡ Process with one thread
### Classification

<table>
<thead>
<tr>
<th># threads Per AS:</th>
<th># of addr spaces:</th>
<th>One</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>One</td>
<td></td>
<td>MS/DOS, early Macintosh</td>
<td>Traditional UNIX</td>
</tr>
<tr>
<td>Many</td>
<td></td>
<td>Embedded systems (Geoworks, VxWorks, JavaOS, etc)</td>
<td>Mach, OS/2, HP-UX, Win NT to 8, Solaris, OS X, Android, iOS</td>
</tr>
<tr>
<td></td>
<td>JavaOS, Pilot(PC)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Real operating systems have either
  - One or many address spaces
  - One or many threads per address space