Operating Systems
(1DT020 & 1TT802)

Lecture 16
Review

May 26, 2009

Léon Mugwaneza

http://www.it.uu.se/edu/course/homepage/os/vt09
OS 1DT020&1TT802 - Review

- What is an Operating System?
- Processes, Threads
- Scheduling
- Synchronization
- Virtual memory, Paging
- Disks, File systems
- I/O Systems
- Protection and Security
What is an Operating System?

• No universally accepted definition
  – “Everything a vendor ships when you order an operating system” is good approximation
  – “The one program running at all times on the computer” is the kernel.

An OS is responsible of 2 main tasks:

– Provide a virtual machine abstraction
  » Turn hardware/software peculiarities into what programmers want/need
    ⚡ application program view: an OS extends the processor’s instruction set with new (complex) instructions accessible via system calls.

– Resources (Hardware and Software) management, sharing and protection
  » Optimize for convenience, utilization, security, reliability, etc.

💰 The 2 tasks are not separate
Processes

• What are they?
• How are they made?
• What is a process CPU context?
• How are they represented in an OS?  
  – ie what data structures are used?
• How many processes can run on a CPU at once?
• What are process states?
Threads

• What are they?
• How are they made?
• What state do they maintain?
• How many can run on a CPU at once?
• How do they differ from processes?
• Why separate threads from processes?
Threads (cnt’d)

• What is different between a thread and a process context switch?
• How does a thread go from inactive to running?
  – What needs to happen?
• Can we use both threads and processes?
  – Why would we do this?
Dual mode operation, Kernel and user Address spaces

- What is dual mode operation?
- Why do we need dual mode operation?
- What is kernel mode? What is user mode?
  - How do we switch from user mode to kernel mode?
  - How do we switch from kernel mode to user mode?
  - How can a user program access OS services
- What is a privileged instruction?
- What is the address space?
  - What is kernel address space?
  - What is user address space?
Exceptions

• What is an interrupt?
  – What happens when one occurs?
• What is a system call instruction (sometimes called a trap)? What is an error exception?
  – What happens when one occurs?
OS kernel: an exception/interrupt/syscall handler

- Save a minimal context
- Change to kernel mode, IT masked
- Load kernel context

OS kernel (Except/IT handler)

Save full CPU context of p

code depending on the except/IT/syscall. Determine process q to run next (not necessary p)

Load context of q into CPU registers (restore user mode and IT mask)

- A timer interrupt forces pre-emptive process switching
Scheduling

• What is a dispatcher/scheduler?
• What are its responsibilities?
• When does it run?
• How does it maintain control of the CPU?
• How does it put a process on the CPU?
Scheduling (cnt’d)

• What are scheduling goals?
• What is fairness
• What is throughput?
• What is response time?
• What is completion time?
• What is scheduling overhead?
Scheduling (cnt’d)

• How are processes scheduled on the CPU? (scheduling policies)
  – FIFO (First in first out)
  – RR (Round robin)
  – SJF (Shortest job first)
  – SRTF (Shortest remaining time first)
  – MLFQ (Multi-level feedback queues)
  – Lottery scheduling
• How do job characteristics influence scheduling choice?
• What is the difference between a preemptive and non-preemptive scheduling policy?
• How can one estimate the CPU burst of a process?
Cooperating Threads

• What is the difference between multiprocessing and multiprogramming?
• What is the definition of ‘run concurrently’?
• What are independent threads?
• What are cooperating threads?
Atomic operations

• What is an atomic operation?
• Why are they important?
• How can you ensure an operation is atomic?
Synchronization

• What is a race condition?
• What is a critical section?
• What is mutual exclusion?
• What is synchronization?
• What is busy-waiting? (or ‘spin-locking’)
  – Why should we avoid busy waiting?
  – When should we use it?
• What is a lock?
  – What are its properties?
Synchronization (cnt’d)

• What is a semaphore?
  – What are its properties?
• What is a condition variable?
  – What are its properties?
  – How are these different from semaphores?
• What is a monitor?
  – What is a Mesa-style monitor?
  – What is a Hoare-style monitor?
Address translation

• What is address binding?
  – When are instructions and data bound to memory?
• What is base + limit (base and bound) relocation?
  – What are the advantages?
  – What are the disadvantages?
  – What does base-and-bounds relocation add to context switches?
  – What does it add per memory access?
• Why do we need address translation?
• What is a Virtual Address? What is a Physical Address?
• What is internal fragmentation? What is external fragmentation?
Address translation: Base+limit relocation

Diagram:

CPU → Virtual Address → Base + Limit → Physical Address → DRAM

- If Limit < Base, physical address is valid.
- If Limit ≥ Base, error.

No: Error!
Address translation (cnt’d)

• Segmentation
  – What is a segment table?
  – What does segmenting fix that is broken with base and bounds?
  – What is a segment table entry? What does it look like?
    » What is each field for?
  – What is a segment fault?
  – What exactly happens when a program tries to read from memory?
  – What does segmentation add to context switches?
Address translation: Segmentation

<table>
<thead>
<tr>
<th>Seg #</th>
<th>Offset</th>
<th>Base0</th>
<th>Limit0</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base1</td>
<td>Limit1</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base2</td>
<td>Limit2</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base3</td>
<td>Limit3</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base4</td>
<td>Limit4</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base5</td>
<td>Limit5</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base6</td>
<td>Limit6</td>
<td>N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base7</td>
<td>Limit7</td>
<td>V</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Also compare offset to limit (error if offset $\geq$ limit)
Address Translation (cnt’d)

• Paging
  – What is a page table?
  – What does paging fix that is broken in segmenting?
  – What does a page table entry look like?
    » What is each field for?
  – What is a page fault?
  – What exactly happens when a program tries to read from memory?
Address Translation: Paging

Virtual Address:

PageTablePtr

PageTableSize

> (Access Error)

Virtual Page #

Offset

page #0  V,R
page #1  V,R
page #2  V,R,W
page #3  V,R,W
page #4  N
page #5  V,R,W

PageTablePtr

PageTableSize

> (Access Error)

Physical Page #

Offset

Physical Address

Check Perm

Access Error
Address translation (cnt’d)

• What does paging add to context switches?
• Can you page out anything in memory?
• How big is a page table?
• What is 2-level paging?
  – How does this work?
• What is an inverted page table?
  – How does this work?
  – Why do you need one?
Address translation (cnt’d)

• How do these address translation methods provide protection among processes?
• How do they allow sharing?
• How would you combine segmentation with paging?
  – Why would you want to?
Address translation caching (TLBs)

• What is a TLB?
  – Why do we use TLBs?
• What happens to the TLB on a context switch?
• What does using a TLB add to handling page faults?
• What is the effective access time of a memory reference?
  – how is it calculated?
Demand paging

• What is demand paging?
• What happens when you write/read to a memory location that's swapped out?
  – TLB lookup
  – TLB miss
  – Page table lookup
  – Page fault
  – Context switch
Page fault handling

• What does the OS do to handle a page fault?
• What does the OS do after handling a page fault?
• How many page faults can occur in one instruction?
• Managing pages
  – When to fetch pages into memory
  – Page replacement algorithm: which pages should be replaced
  – Allocation of Page frames: How do we allocate memory among different processes
Page replacement algorithms

- FIFO: Place pages on queue, replace page at end
- MIN: Replace page that will be used farthest in future
- LRU: Replace page used farthest in past
  - Limitations in implement LRU? What do we need?
- Clock Algorithm: Approximation to LRU
  - Arrange all pages in circular list
  - Sweep through them, marking as not “in use”
  - If page not “in use” for one pass, than can replace
- $N$th-chance clock algorithm: Another approx LRU
  - Give pages multiple passes of clock hand before replacing
- List of free page frames makes page fault handling faster
  - Filled in background by pageout demon
Page Table Entry fields

• What is the Valid bit in a PTE for?
• What is the Reference/Used bit for?
• What is the Dirty bit for?
Thrashing and Working Sets

• What is thrashing?
  – How do we fix this problem?

• What is a working set of a process?
  – Be able to define this formally
  – Why is it useful?

• What is the Working Set tracking replacement algorithm?
  – How can it be implemented?
File System Interface

• What is a file system?
• What is a file?
  – Can you name some file attributes? Some operations on files?
  – Who decides on the file structure?
• What is a file header? (the unix file header is called an inode)
• What is directory?
• What is a hard link?
• What is soft (or symbolic) link?
File System Interface (cnt’d)

• File naming:
  – How does the user ask for a particular file?
  – How does the file system translates from user-visible names to system resources?

• Why do file systems use a system-wide open files table and a per-process open files tables?
File System Implementation

• Properties of magnetic disks?
  – What is the smallest addressable element?
    » Typical sizes?
  – What is the unit of transfer between disk and OS?
• Disk Performance?
  – Queuing time + Controller + Seek + Rotational + Transfer
  – Rotational latency: on average \( \frac{1}{2} \) rotation
  – Transfer time: spec of disk depends on rotation speed and bit storage density
• Disk scheduling?
  – FIFO, SSTF, SCAN, C-SCAN
  – Which component of disk performance disk scheduling algorithms try to optimize?
File System Implementation (cnt’d)

• Access patterns
  – Sequential access – bytes read in order (give me the next X bytes, then give me next)
  – Random access – read/write element out of middle of array
  – Content based access – “find me 100 bytes starting with SMITH”

• Usage patterns
  – Most files are small
  – Large files use up most of the disk space
  – Large files account for most of the bytes transferred to/from disk
File System Implementation (cnt’d)

• What is Logical Block Addressing?
  – Why is LBA used?
  – Advantages? Disadvantages?

• How to organize files on disk?
  – Contiguous allocation
  – Linked files
  – Indexed files
  – Multi-level Indexed files

✎ Should know
  » How do they work?
  » Advantages?
  » Disadvantages?
File System Implementation (cnt’d)

• What is a track buffer?
• What is a buffer cache?
  – How does it work?
  – What is the replacement policy?
• How does delayed writes work?
  – Advantages?
  – Disadvantages?
• How to make file systems durable?
IO Systems

• I/O Devices characteristics?
  – Many different speeds (10 bytes/sec to GBytes/sec)
  – Different Access Patterns: block, char, net devices
  – Different Access Timing: Non-/Blocking, Asynchronous
• What are I/O Controllers?
• What is a Device Driver?
• How does the processor actually talk to the device?
• What is programmed I/O?
• What is Direct Memory Access?
• Who does an I/O device notify I/O completion or error to OS?
Protection vs Security

- **Protection**: one or more mechanisms for controlling the access of programs, processes, or users to resources
  - Page Table Mechanism
  - File Access Mechanism

- **Security**: use of protection mechanisms to prevent misuse of resources
  - Misuse defined with respect to policy
  - Requires consideration of the external environment within which the system operates

- To Assist with Protection, **Hardware** provides at least two modes: “Kernel” mode (or “protected”) and “User” mode
  - Mode set with bits in control register only accessible in kernel-mode
  - Some instructions only available in kernel mode (Privileged instructions)

- **Three Pieces to Security**
  - **Authentication**: who the user actually is
  - **Authorization**: who is allowed to do what
  - **Enforcement**: make sure people do only what they are supposed to do