Operating Systems
(1DT020 & 1TT802)

Lecture 1
Introduction

March 24, 2009

Léon Mugwaneza

http://www.it.uu.se/edu/course/homepage/os/vt09
Who am I?

- Léon Mugwaneza
  - Visiting lecturer
  - Senior lecturer
  Computer science department
  ESIL/Université de la Méditerranée
  Marseille, France
Goals for Today

• What is an Operating System?
  – and – what is it not?
• Examples of Operating Systems design
• What is in this course?
• Why study Operating Systems?
• Also “How does this course operate?”

Note: Some slides and/or pictures in the following are adapted from slides ©2005 Silberschatz, Galvin, and Gagne, others from Kubiatowicz - CS162 ©UCB Fall 2007 (University of California at Berkeley)
What is an Operating System?

- Software that makes computers easier to use
  - What is a computer?
  - What is software?

- A computer system is composed of hardware and software
Computing devices everywhere
Computer System hardware Organization

- **Computer-system  operation**
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles
Computer-System Operation

• I/O devices and the CPU can execute concurrently
• Each device controller is in charge of a particular device type
• Each device controller has a local buffer
• CPU moves data from/to main memory to/from local buffers
• I/O is from the device to local buffer of controller
• Device controller informs CPU that it has finished its operation by causing an *interrupt*
**Computer hardware variety**

• Every piece of computer hardware is different
  – Different CPU
    » Pentium, PowerPC, ColdFire, ARM, MIPS
  – Different amounts of memory, disk, …
  – Different types of devices
    » Mice, Keyboards, Sensors, Cameras, Fingerprint readers
  – Different networking environment
    » Cable, DSL, Wireless, Firewalls, …

• Questions:
  – Does the programmer need to (re)write his own code for common tasks?
  – Does the programmer need to write a single program that performs many independent activities?
  – Does every program have to be altered for every piece of hardware?
  – Does a computer system run only one user program at time?
  – Does a faulty program crash everything?
  – Does every program have access to all hardware?
OS Tool: Virtual Machine Abstraction

Application

Operating System

Hardware

- Software Engineering Problem:
  - Turn hardware/software peculiarities into what programmers want/need
  - Optimize for convenience, utilization, security, reliability, etc...

- For Any OS area (e.g. file systems, virtual memory, networking, scheduling):
  - What’s the hardware interface? (physical reality)
  - What’s the application interface? (nicer abstraction)
Virtual machines

• Software emulation of an abstract machine
  – Make it look like hardware has features you want
  – Programs from one hardware & OS on another one

• Programming simplicity
  – Each running program thinks it has all memory/CPU time
  – Each running program thinks it owns all devices
  – Different devices appear to have same interface
  – Device interfaces more powerful than raw hardware
    » Bitmapped display ⇒ windowing system
    » Ethernet card ⇒ reliable, ordered, networking (TCP/IP)

• Fault isolation
  – Programs unable to directly impact other programs
  – Bugs cannot crash whole machine

• Protection and portability
  – Java interface safe and stable across many platforms
Definition: An operating system implements a virtual machine that is (hopefully) easier and safer to program and use than the raw hardware.
Virtual Machines (con’t): Layers of OSs

- Useful for OS development
  - When OS crashes, restricted to one VM
  - Can aid testing programs on other OSs
What does an Operating System do?

- **Silerschatz and Gavin:**
  - "An OS is Similar to a government"
  - Raises the question: does a government do anything useful by itself?

- **Coordinator and Traffic Cop:**
  - Manages all resources
  - Settles conflicting requests for resources
  - Prevent errors and improper use of the computer

- **Facilitator:**
  - Provides facilities that everyone needs
  - Standard libraries, Windowing systems
  - Make application programming easier, faster, less error-prone

- **Some features reflect both tasks:**
  - E.g. File system is needed by everyone (Facilitator)
  - But File system must be protected (Traffic Cop)
What is an Operating System,... Really?

• Most Likely:
  – Memory Management
  – I/O Management
  – CPU Scheduling
  – Communications? (does email belong in OS?)
  – Multitasking/multiprogramming?

• What about?
  – File System?
  – Multimedia Support?
  – User Interface?
  – Internet Browser? 😊

• Is this only interesting to Academics??
Course Administration

• Instructor:
  Léon Mugwaneza (lectures and labs)
  email: leon.mugwaneza@it.uu.se

• Course homepage: [http://www.it.uu.se/edu/course/homepage/os/vt09](http://www.it.uu.se/edu/course/homepage/os/vt09)

• Labs: 3 labs
  – 1st lab on March 30 (material on the course homepage by this Thursday noon)
  – Deadline will be announced later

• Evaluation
  – 1 written exam - closed books
  – Labs – you have to complete the 3 labs

• Course Registration
  – Tick your name on one of the 2 lists
  or
  – Fill in your name (and other data) on the appropriate sheet (1DT020 or 1TT802)
Textbook

• Text: Operating Systems Concepts, 8th Edition Silberschatz, Galvin, Gagne

• Online supplements
  – See book web page and “Student companion site” (links on course homepage)
  – Include slides from the editor, appendices, and solutions to practice exercises

• Note: Operating Systems Concepts, 7th ed. is also OK.
Topic Coverage


- 1 lecture: Introduction (what is an operating system?)
- 3 lectures: Processes, threads, process dispatching and scheduling
- 3 lectures: Process communication and Synchronization
- 2 lectures: Virtual memory (mechanism and policies)
- 3 lectures: File systems and mass storage devices
- 1 lecture: I/O systems
- 1 lecture: Protection and security
- 1 lecture: Review
Operating System Definition (Cont.)

• No universally accepted definition
• “Everything a vendor ships when you order an operating system” is good approximation
  – But varies widely
• “The one program running at all times on the computer” is the kernel.
  – Everything else is either a system program (ships with the operating system) or an application program
What if we didn’t have an Operating System?

- Source Code $\Rightarrow$ Compiler $\Rightarrow$ Object Code $\Rightarrow$ Hardware
- How do you get object code onto the hardware?
- How do you print out the answer?
- Once upon a time, had to toggle in program in binary and read out answer from LED’s!

Altair 8080
Simple OS: What if only one application?

• **Examples:**
  – Very early computers
  – Early PCs
  – Embedded controllers (elevators, cars, etc)

• **OS becomes just a library of standard services**
  – Standard device drivers
  – Interrupt handlers
  – Libraries (math, …)
MS-DOS Layer Structure

application program

resident system program

MS-DOS device drivers

ROM BIOS device drivers
More complex OS: Multiple applications

• Full coordination and protection
  – Manage interactions between different users
  – Multiple programs running simultaneously
  – Multiplex and protect hardware resources
    » CPU, Memory, I/O devices like disks, printers, etc

• Facilitator
  – Still provides standard libraries, facilities

• Would this complexity make sense if there were only one application that you cared about?
Example: Protecting programs from each other

• Problem: Run multiple applications in such a way that they are protected from one another

• Goal:
  – Keep user programs from crashing OS
  – Keep user programs from crashing each other
  – [Keep parts of OS from crashing other parts?]

• Simple Policy:
  – Programs are not allowed to read/write memory of other programs or of Operating System

• (Some of the required) Mechanisms:
  – Address translation
  – Dual mode operation
Address Translation

• **Address Space**
  - A group of memory addresses usable by something
  - Each program and kernel has potentially different address spaces.

• **Address Translation:**
  - Translate from Virtual Addresses (emitted by CPU) into Physical Addresses (of memory)
  - Mapping *often* performed in Hardware by Memory Management Unit (MMU)
Example of Address Translation

Translation Map 1

Translation Map 2

Physical Address Space

- Translation helps protection:
  - Control translations, control access
  - Should users be able to change Translation map???
Dual Mode Operation

- **Hardware** provides at least two modes:
  - “Kernel” mode (or “supervisor” or “protected”)
  - “User” mode: Normal programs executed

- **Some instructions/ops prohibited in user mode:**
  - Example: cannot modify page tables in user mode
    » Attempt to modify ⇒ Exception generated

- **Transitions from user mode to kernel mode:**
  - System Calls, Interrupts, Other exceptions

![Diagram of mode transition](image)
UNIX System Structure

**User Mode**

- Applications (the users)
- Standard Libs
  - shells and commands
  - compilers and interpreters
  - system libraries

**Kernel Mode**

- **system-call interface to the kernel**
  - signals terminal handling
  - character I/O system
  - terminal drivers
  - file system
  - swapping block I/O system
  - disk and tape drivers
  - CPU scheduling
  - page replacement
  - demand paging
  - virtual memory

**Hardware**

- terminal controllers
  - terminals
- device controllers
  - disks and tapes
- memory controllers
  - physical memory
OS Systems Principles

• OS as illusionist:
  – Make hardware limitations go away
  – Provide illusion of dedicated machine with infinite memory and infinite processors

• OS as government:
  – Protect users from each other
  – Allocate resources efficiently and fairly

• OS as complex system:
  – Constant tension between simplicity and functionality or performance

• OS as history teacher
  – Learn from past
  – Adapt as hardware tradeoffs change
Why Study Operating Systems?

• **OS are complex systems:**
  – How can you manage complexity for future projects?

• ** Buying and using a personal computer:**
  – Why different PCs with same CPU behave differently
  – How to choose a processor (Opteron, Itanium, Celeron, Pentium, Hexium)? [Ok, made last one up]
  – Should you get Windows XP, Vista, Linux, Mac OS …?

• **Security, viruses, and worms**
  – What exposure do you have to worry about?

• **Discover what is in the black box! 😊**
Summary

• Operating systems provide a virtual machine abstraction to handle diverse hardware
• Operating systems coordinate resources and protect users from each other
• Operating systems simplify application development by providing standard services
• Operating systems can provide an array of fault containment, fault tolerance, and fault recovery