Operating Systems
Chapter 1

Marie Roch
contains slides from:
Silberschatz, Galvin, and Gagne 2003
What is an Operating System?

- It is an extended machine
  - Hides the messy details which must be performed
  - Presents user with a virtual machine, easier to use

- It is a resource manager
  - Each program gets time with the resource
  - Each program gets space on the resource
What is an operating system?

- Extended machine
  - abstraction
  - standardization
What is an operating system?

- Resource manager
  - sharing
    - time
    - space
  - protection

Time sharing:
One after another please…

Space sharing
Privileged instructions

• To maintain integrity of the system, not every process should be able to execute every type of instruction.

• User mode vs. Supervisory mode
  – Prohibit privileged instructions in user mode
  – Controlled by a bit in the program status word (PSW)
  – Also known as kernel-mode
Privileged instructions

What type of instructions should be privileged?

Figure 1-1. Tanenbaum 2008

program status word (PSW)
20,000 Foot view of the last 60+ years

Vacuum tubes

1945
Thomson invented 1897
(1950s memory shown)

1955
Bardeen & Bratt’s transistor (1947)

Integrated circuits & multiprogramming

1965
Kilby & Noyce’s first IC (1959)

1980
Job 3
Job 2
Job 1
Operating system

Memory partitions

Transistors and batch systems

Personal computing

Processor Architecture

(a) A three-stage pipeline
(b) A superscalar CPU
Buses

common buses

- DMI – Direct media interface
- ISA – industry standard architecture
- PCI – peripheral component interconnect
- PCI Express
- SATA – serial advanced technology attachment
- SCSI – small computer system interface
- USB – universal serial bus

Figure 1-12. The structure of a large x86 system
Memory

Note: Mid 2010s numbers & technologies, different numbers today, but the same idea…

Fig 1.9 Tanenbaum and Bos
Caching

Basic concept of a cache:
- Store a frequently used subset of a slow device on a faster one.
- On access, check the cache first, then the slower device.
- Process should be *transparent* to user.
HW Review: Memory cache

• Small memory is placed between the CPU and RAM.
Memory Cache

• **How the cache works**
  – CPU Access/Fetch
    • Address sent out on bus
    • Cache searches table for address in O(1) time
    • Search result:
      – HIT: Address found, return contents of address
      – MISS: Address not in cache
        » Request sent to RAM
        » Result is saved into cache (might overwrite another entry)
HW Review: Memory Cache

• How the cache works
  – CPU Write
    • Address and data sent out on bus
    • Cache updates/inserts into cache
    • *Might* also write data to RAM (called write-through)

  – At times, a current cache entry will have to be overwritten when a new one is needed.
HW Review: Memory Cache

• **Cache issues**
  – Consistency: Addresses in the cache have the same value as those in RAM.
HW Review: Memory Cache

- Coherency:
  - Issue when multiple CPUs share memory
  - Any address that is in more than one CPU’s cache must have the same value in each cache it is present.
Figure 1-8.

(a) A quad-core chip with a shared L2 cache.
(b) A quad-core chip with separate L2 caches.
One base-limit pair and two base-limit pairs
HW Review: I/O Devices

• All Input/Output (I/O) mediated by a controller

• Two types of I/O
  – Programmed: CPU polls device
  – Interrupt driven: Device signals upon completion

• I/O can be
  – Synchronous
  – Asynchronous
(a) Steps in starting an I/O device and getting interrupt
(b) How the CPU is interrupted
HW Review: Magnetic disks

Structure of a disk drive

Surface 0

Surface 1

Surface 2

Surface 3

Surface 4

Surface 5

Surface 6

Surface 7

Read/write head (1 per surface)

Direction of arm motion

Structure of a disk drive
HW Review: Magnetic disks

- Anatomy of a disk drive continued

- Single platter
- Tracks – Individual rings containing sectors of data
- Blocks/Sectors – Smallest addressable data unit, typically 512 bytes
- Cylinder – Grouping of same track across platters.
HW Review: Direct Memory Access (DMA)

• Problem: CPU-managed data transfer between peripheral and RAM consumes CPU resources.

• DMA:
  – CPU indicates source/destination RAM area and instructs controller on what should be done.
  – CPU can continue executing during peripheral/RAM transfer as long as it does not need to access RAM.
  – Controller executes operation and tells CPU when it is done via interrupt.
Simplified View of Interrupt Handling

1. Device signals interrupt controller by placing signals on bus lines.

2. Controller signals the CPU and places an interrupt number associated with the device on the bus.

3. If the CPU has interrupts enabled:
   1. Save PC, SP, & PSW.
   2. Shift to supervisor mode.
   3. Execute an interrupt service routine associated with the interrupt selected from the interrupt service table.
   4. Restore PC, SP, & PSW and continue executing process.
      (Note that not all operating systems will return to the point right away.)
Operating System Concepts: Bootstrap

• How do we load the operating system?

• Simplified overview for Intel architectures:
  • Basic input/output system (BIOS) initializes and executes the power on self test (POST)
  • BIOS queries buses and performs low level configuration of the peripherals (plug and play)
  • BIOS loads the boot sector of the designated I/O device.
  • The boot sector contains a small program which typically loads and executes a more sophisticated loader program.
  • The loader program loads the operating system and the initial process is started.
Concepts: Processes

- A process tree
  - A created two child processes, B and C
  - B created three child processes, D, E, and F
(a) A potential deadlock. (b) an actual deadlock.
Concepts: File systems

File system for a university department

Tanenbaum 2001
Concepts: File Systems

- Before mounting,
  - files on network drive, USB drive, etc. are inaccessible
- After mounting network drive on b,
  - files on network drive are part of file hierarchy
Concepts:
Interprocess communication

Two processes connected by a pipe
Concepts: Function calls

- Run-time models allow compiled programs to:
  - pass parameters
  - resolve variable scope
  - track subroutine invocation

- We need to understand a small part of this to understand how user programs communicate with the operating system.
Concepts: Function calls

- **Local variables and actual arguments**
- **Global variables (ick!) and variables allocated from the heap (e.g. using `malloc/calloc/new`)**
- **Program instructions**

The diagram illustrates the process memory with different sections:

- **Stack**
- **Gap**
- **Data**
- **Text**

Address (hex):
- FFFF
- 0000

Tanenbaum 2001
Concepts: Function calls

foobar(alpha, beta)

• let alpha = 5, beta = 2

• Push arguments
  (typically in reverse order)

• Call function read
  push return address on stack
  execute jump subroutine instruction

• Pop stack upon return

stack growth

- 2
- 5
- 0x3002A
Concepts: System calls

- Typically starts by calling a library function
- Library function sets a register indicating what system functionality should be performed
- Executes a special instruction called a TRAP which triggers a software interrupt.
- Kernel executes and returns with RTE
Concepts: System Calls

Execution of system call:
read (fd, buffer, nbytes)
Concepts: System Calls

- Permits processes to make requests of the operating system
- Sample process management calls from UNIX

<table>
<thead>
<tr>
<th>Process management</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid = fork()</td>
<td>Create a child process identical to the parent</td>
</tr>
<tr>
<td>pid = waitpid(pid, &amp;statloc, options)</td>
<td>Wait for a child to terminate</td>
</tr>
<tr>
<td>s = execve(name, argv, environp)</td>
<td>Replace a process’ core image</td>
</tr>
<tr>
<td>exit(status)</td>
<td>Terminate process execution and return status</td>
</tr>
</tbody>
</table>

Figure 1-18 Tanenbaum 2001
Concepts

• Memory management
• Security
• Shells permit the user to communicate with the operating system
  – text-oriented command interpreter
  – graphical user interface (GUI)
Operating System Architecture

- Monolithic systems
  - Traditional program design
  - Collection of procedures
## Architecture: Layered Systems

<table>
<thead>
<tr>
<th>Layer</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>The operator</td>
</tr>
<tr>
<td>4</td>
<td>User programs</td>
</tr>
<tr>
<td>3</td>
<td>Input/output management</td>
</tr>
<tr>
<td>2</td>
<td>Operator-process communication</td>
</tr>
<tr>
<td>1</td>
<td>Memory and drum management</td>
</tr>
<tr>
<td>0</td>
<td>Processor allocation and multiprogramming</td>
</tr>
</tbody>
</table>

## Structure of the THE operating system
Architecture: Virtual Machines

- Provides *emulation* of multiple *physical machines* called virtual machines.
Type 1 hypervisors require hardware support.

Figure 1-29. (a) Type 1 hypervisor. (b) Type 2 hypervisor.
Architecture: VM Components

• Hypervisor (monitor)
  – Interfaces with hardware.
  – Manages sharing of physical resources between virtual machines.
  – Updates data structures representing virtual machines.
  – Provides emulation of hardware for virtual machines.
Architecture: Virtual machines

• Software interrupts

  1. User code executes TRAP
  2. Determine which VM caused TRAP
  3. Set hardware emulation in appropriate VM.
  4. Control transferred to VM “kernel”

- Hardware interrupts similar, except hypervisor handles physical device and sets up virtual device.
Architecture: Virtual machines

- Execution of privileged instructions

1. "kernel" attempts to execute privileged instruction.
2. Monitor interrupted with privilege violation
3. If simulated PSW has supervisor mode set, monitor executes/simulates the privileged instruction on behalf of the VM operating system.
Microkernels

Figure 1-26. Structure of the MINIX 3 system.

Essential services run in user mode and make system calls to a very small kernel. User processes send messages to the services.
Other Architectures

• Exokernels: Similar to virtual machines
  – Each machine allocated a subset of the resources.
  – Saves a mapping layer as exokernels partition resources.
    e.g. this part of the disk belongs to exokernel 3
Review: C/C++ compilation

What happens if variables or functions are declared in defs.h?
C/C++ compilation

- preprocessed source files
  - contains stdio.h
def.h
io.c
- contains
defs.h
main.c

Object files
- io.o
- main.o

Compiler
C/C++ linking

The linker is usually invoked by the compiler transparently.