Operating Systems

Christopher Kruegel
Department of Computer Science
UC Santa Barbara
http://www.cs.ucsb.edu/~chris/

CS 170 Info

- Web page: http://www.cs.ucsb.edu/~chris/cs170/index.html

- Mailing lists (one for class, one for instructors)
  cs170-users – used to disseminate information and ask fellow classmates
  cs170-admin – use to reach TA and me

Requirements

- The course requirements include
  - several projects
  - a midterm and a final exam
- The projects (and exams) are individual efforts
- The final grade will be determined according to the following weight
  - projects: 50%
  - exams: 50%
- Class participation and non-graded quizzes

Lab Projects

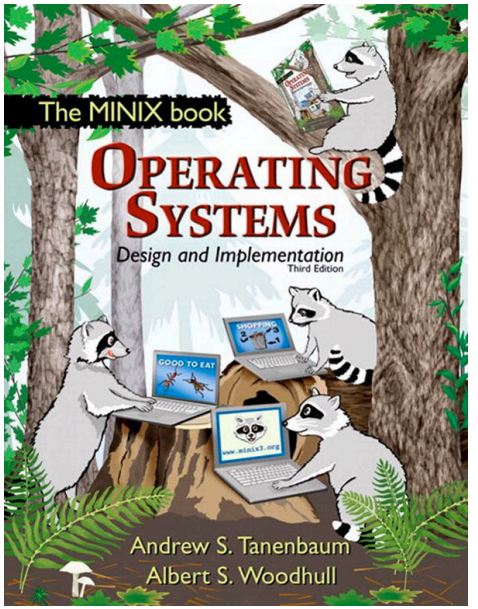
~5 programming assignments

- Shell (system calls)
- Threads (parallel execution, scheduling)
- Synchronization (semaphores, …)
- Memory (virtual memory, shared regions, …)
- File systems

Material

- The course will adopt the following book:

  Andrew S. Tanenbaum and Albert S. Woodhull
  Operating Systems (Design and Implementation)
  3rd Edition, Prentice-Hall, 2006



- The set of assignments will be updated during the course

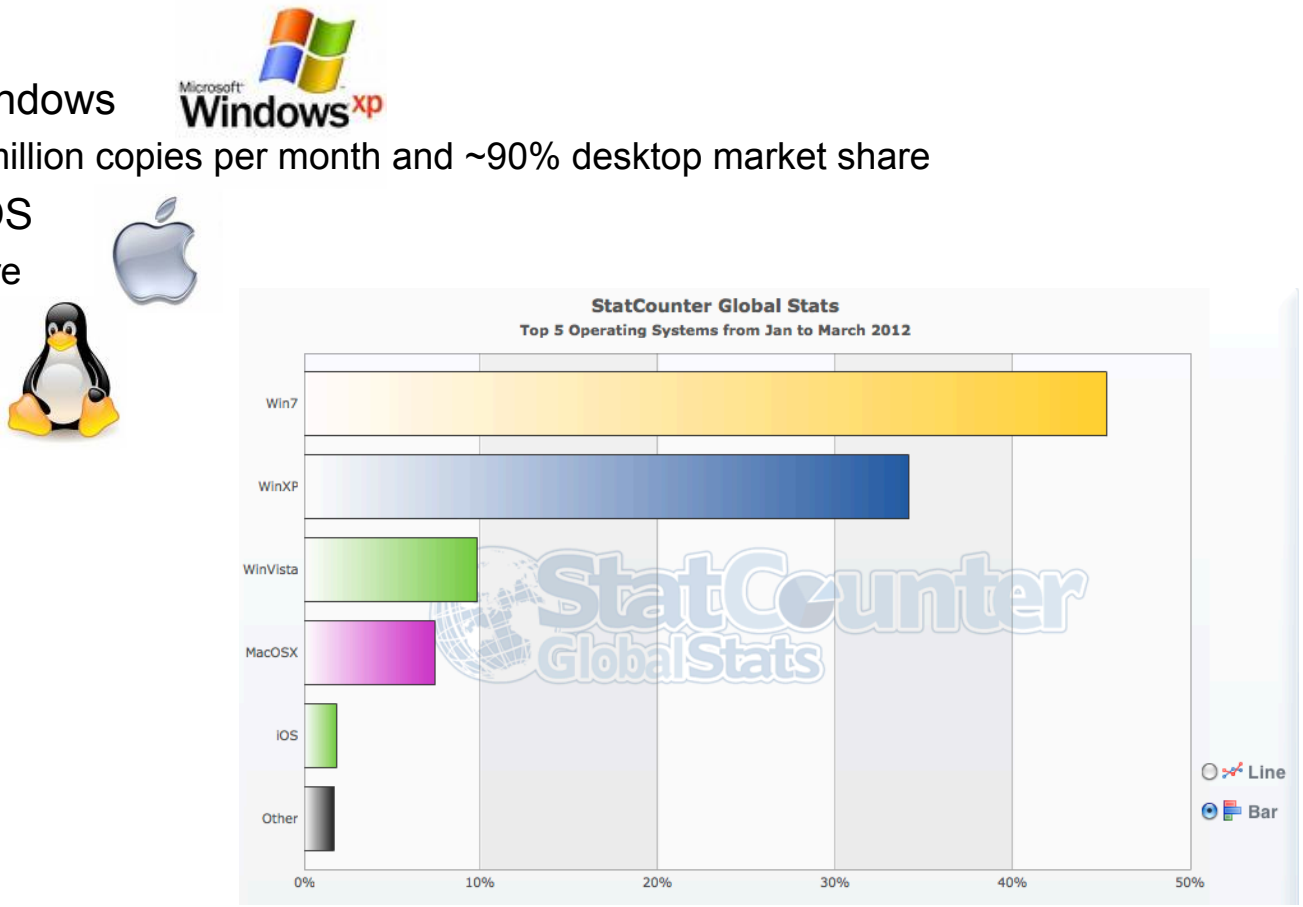
- Additional material (slides) is provided on the class Web page

Operating Systems

- Let us do amazing things …
  - allow you to run multiple programs at the same time
  - protect all other programs when one app crashes
  - allow programs to use more memory that your computer has RAM
  - allow you to plug in a device and just use it (well, most of the time)
  - protects your data from fellow students on CSIL

What is the most-used OS?

- Desktops
  - Microsoft Windows
    - sells 10 million copies per month and ~90% desktop market share
  - Apple Mac OS
    - ~8% share
  - Linux
    - negligible



What is the most-used OS?

- But wait … what about embedded devices?
  - order of magnitude more devices

    iTron (several billion installations)

    Wind River (VxWorks) – market leader, “Lord of Toasters”

    Linux is growing rapidly

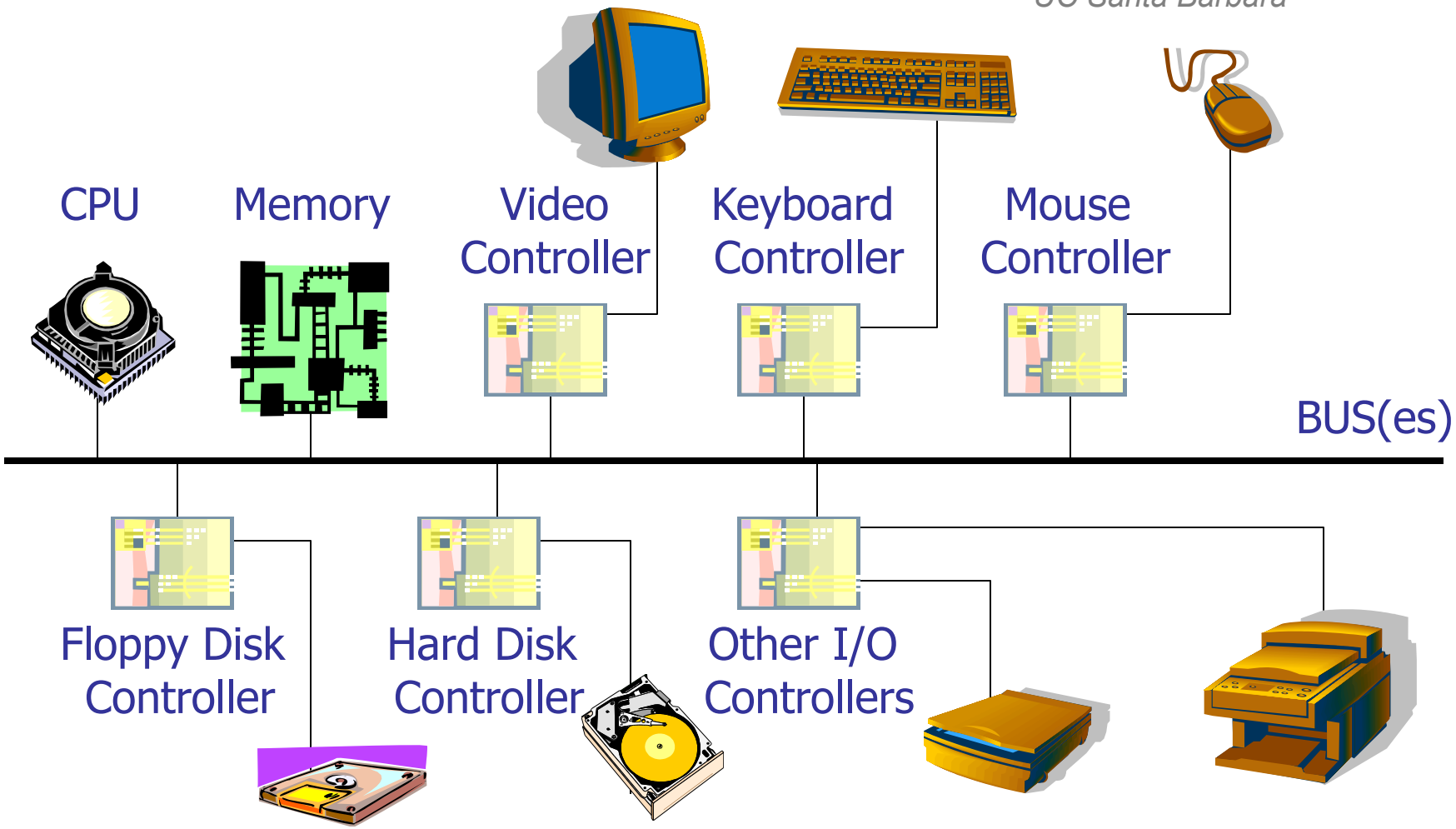
    ~~Symbian and cell phones (73 million in 2008)~~

- Smartphone ecosystem is moving extremely fast (600M forecast for ‘12)
  - Android 49% (Linux-based)
  - iOS 19% (derived from Mac OS X – Darwin/Unix foundation)
  - BlackBerry 13% (proprietary OS developed by RIM)
  - Windows 11%
  - Symbian 5%

Outline

- Introduction to Operating Systems
- Processes and Threads
- IPC, Synchronization, and Deadlocks
- Memory Management
- Input/Output
- File Systems
- Security

In The Beginning There Was Hardware



Central Processing Unit

- Fetches instructions from memory and executes them
- Characterized by an *instruction set*
  - Loads and stores values to/from memory/registers
  - Performs simple operations (add, and, xor)
  - Jumps to locations
- Contains a set of *registers*
  - Program counter
  - Stack pointer
  - PSW (Program Status Word)
    - Kernel mode: total access to memory/registers and instructions
    - User mode: limited access to memory/registers and subset of instructions

Memory

- Set of locations that can hold values
- Organized in a hierarchy of layers
  - Registers (access time ~1 nsec)
  - Cache memory (access time ~2 nsec)
  - Main memory - RAM (access time ~10 nsec)
  - Hard disk (access time ~10 msec)
- Read Only Memory (ROM) used to store values ... permanently

I/O Devices

- Controllers connected to the bus
- Device connected to a controller
- The controller provides an interface to access the device resources/functionalities
  - done by storing values into device registers
- Memory mapped access
  - device registers mapped into memory region
- Dedicated I/O
  - special CPU instructions

Disk

- One or more metal platters that rotate at a constant speed (e.g., 5400 rpm, 7200 rpm, …)
- Each platter has many concentric *tracks*
- Corresponding tracks in different platters compose a *cylinder*
- Each track is divided in *sectors*
- A mechanical arm with a set of heads (one per surface) moves on platters and reads/writes sectors
  - Move to the right track (1 to 10 msec)
  - Wait for the sector to appear under the head (5 to 10 msec)
  - Perform the actual read/write

Buses

- Used to transfer data among components
- Different functions, speeds, number of bytes transferred

- Cache bus
- Memory bus (FrontSide Bus, QuickPath Interconnect)

- ISA (Industry Standard Architecture) bus
  - 8.33 MHz, 2 bytes, 16.67 MB/sec
- PCI (Peripheral Component Interconnect) bus
  - 66 MHz, 8 bytes, 528 MB/sec
- PCIe (PCI Express)
- USB (Universal Serial Bus)
- SCSI (Small Computer System Interface) bus
- IEEE 1394/FireWire bus

There Be Power...

There Be Power...

- CPU starts and loads instructions starting at 0xffffffff0
- Instruction jumps to BIOS code
- BIOS (Basic Input/Output System) is started
  - Performs basic tests (memory, keyboard, etc) – POST (power on self test)
  - Determines the “boot device” (Hard disk, Floppy, CD-ROM)
  - Loads the contents of the first physical sector (the Master Boot Record - MBR - Cyl 0, Head 0, Sect 1) in memory 0x7C00 - 0x7DFF
  - Jumps to 0x7C00
- MBR code finds an “active” file system, loads the corresponding boot sector in memory, and jumps to it
- The boot sector code loads the *operating system*

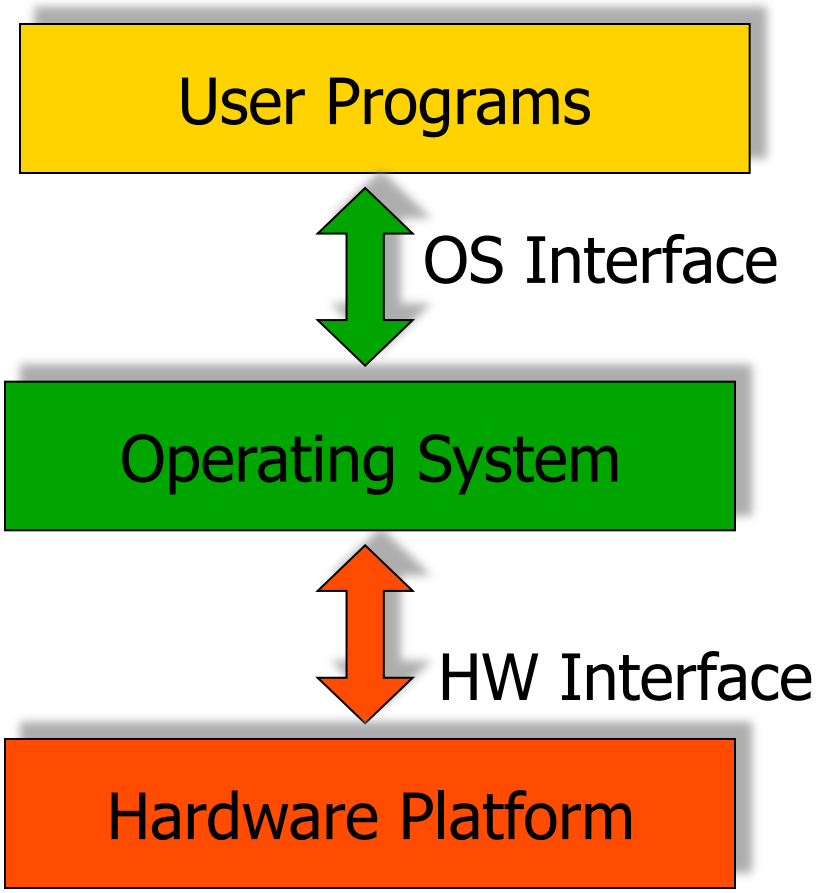
A few words about the BIOS

- Two main components
  - Boot services
    - initialize hardware (including RAM)
    - read and load boot code
    - transfer control
  - Runtime services
    - basic routines for accessing devices
    - can display menus, boot from devices (and even network)
    - access to clock, NVRAM, …
    - OS typically bring their own device drivers
- Developments
  - Standard PC BIOS around for a long time
  - recently, Unified Extensible Firmware Interface (UEFI) started as replacement
  - UEFI is more general, supports boot from large disks

The Operating System

- Where does an operating system fit in a computing system?
- What does an operating system do?
- Why do we need an operating system?
- How is an operating system structured?

Where?


User Programs
OS Interface
Operating System
HW Interface
Hardware Platform

What?

- The operating system is a *resource manager* that provides an orderly and controlled allocation of resources
- The operating system is an implementer of *multiple virtual (extended) machines* that are *easier to program* than the underlying hardware
- Goal:
  - Each program/application can be developed as if the whole computer were dedicated to its execution

Resource Management

- *Multiplexing*
  - creating an illusion of multiple (logical) resources from a single (physical) one
- *Allocation*
  - keep track of who has the right to use what
- *Transformation*
  - creating a new resource (logical) from existing resource (physical) primarily for “ease of use”
- An OS multiplexes, allocates, and transforms HW resources

Types of Multiplexing

- Time multiplexing
  - time-sharing
  - scheduling a serially-reusable resource among several users
    - e.g., CPU or printer
- Space multiplexing
  - space-sharing
  - dividing a multiple-use resource up among several users
    - e.g., memory, disk space

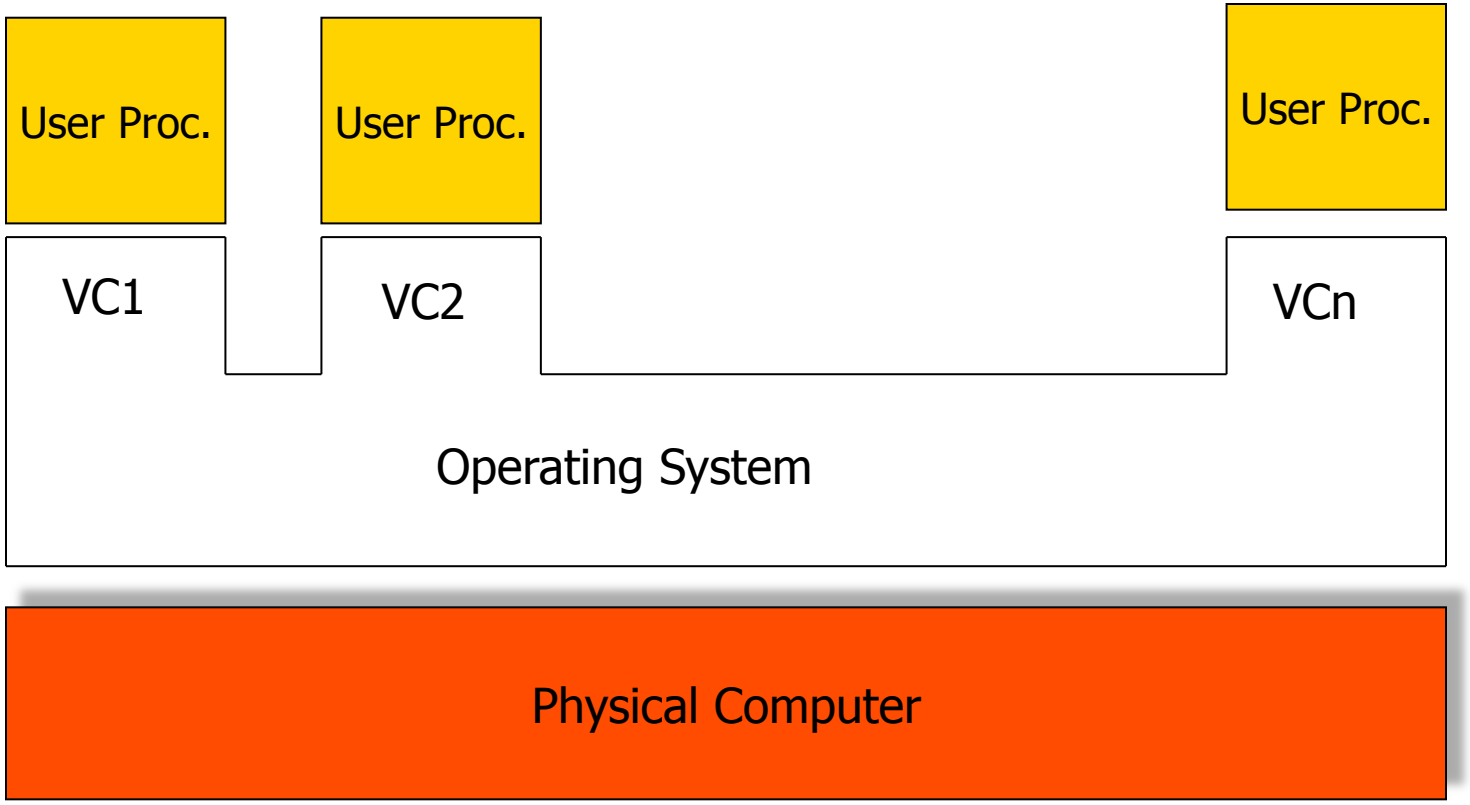
Multiple Virtual Computers

- Multiple processors
  - capability to execute multiple flows of instructions at the same time
- Multiple memories
  - capability to store information of multiple applications at the same time
- Access to file system as an abstraction of the disk
- Access to other I/O devices in abstract, uniform ways
  - e.g., as objects or files

Virtual Computers

- OS creates multiple processes (simulated processors) out of the single CPU
  - time-multiplexing the CPU
- OS creates multiple address spaces (memory for a process to execute in) out of the physical memory (RAM)
  - space-multiplexing of the memory
- OS implements a file-system and I/O system so that processes use and share the disks and I/O simultaneously
  - space-multiplexing the disk and time-multiplexing the I/O channels
- OS creates multiple virtual computers from a single physical machine

Virtual Computers



OS Interface – Virtual Processors

- Nearly the same interface as the physical CPU
- OS removes privileged operations
  - PSW determines if the code is either “user code” or “OS code”
  - changes in status are strictly regulated...
- OS adds instructions (system calls)
  - create new virtual computers (processes)
  - communicate with other VCs
  - allocate memory
  - perform input and output operations I/O
  - access the file system

OS Interface – Virtual Memory

- Memory of the Virtual Computer is similar to the hardware memory (i.e., a sequence of words), and it is accessed the same way
- The OS divides up the memory into parts and gives each part to each virtual computer
- OS creates an illusion that each virtual computer has a memory starting from address 0x0000
- OS creates an illusion that the virtual computer has more memory than the physical memory

OS Interface – Virtual File System

- Secondary storage provides long-term storage for data
- Storage is done physically in term of disk sectors and virtually in terms of disk files
- The virtual computer sees a file system consisting of named files with arbitrary size

OS Interface – Virtual I/O

- I/O operations of the virtual computer are completely different from the I/O operations of the physical computer
- The physical computer has devices with complex control and status registers
- In contrast, the virtual I/O is simple and easy to use
- In fact, in most OSes (e.g., UNIX) virtual I/O abstraction is almost identical to the file-system interface giving rise to uniformity of treatment with respect to all types of I/O devices including disks, terminals, printers, network connections
- Each VC sees a dedicated I/O device: the actual hardware is space/ time multiplexed by the OS

Operating System Services

- The programs running on virtual computers access the operating system services through *system calls*
- A system call is carried out by
  - Storing the *parameters* of the system calls in specific locations (registers, memory)
  - Calling a “software interrupt” or “trap”
- Switch to kernel mode: the OS is notified and takes control of the situation

Do We Need an OS?

- For a specialized application (e.g., a microwave oven, a car), an OS may not be needed
- The hardware can directly be programmed with the rudimentary functionalities required by these applications
- A general-purpose computer, on the other hand, needs to run a wide range of user programs
- For such a system, an OS is indeed necessary
- Otherwise, each user will need to program its own operating system services
- An OS can do this for once and make it available to the user programs