Introduction

Systems II

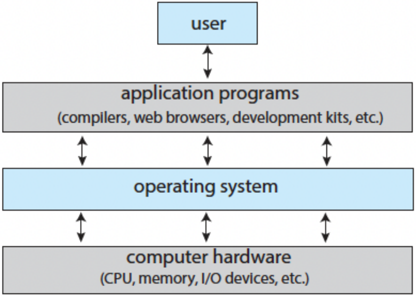
Objectives

- Describe the general organisation of a computer system and the role of interrupts.
- Describe the components in a modern multiprocesses computer system.
- The transiton from user mode to kernel mode.
- How computer systems are used in various computing environments.
- Examples of free and open-soruce operating systems.

What do operating systems do?

The computer system can be divided into:

- hardware: CPU, memory, I/O devices
- operating system
- application programs: word processor, web browsers, compilers, VLC, . . .
- user



User view

How do users see computer systems?

- laptop or a PC (montor, keyboard, ...) is designed for ease of use and not so much for resource utilisation (how hardware is shared)
- mobile devices (smartphones, tablets): touch screen + network
- voice recognition
- embedded computers

System view

Computer's point of view

OS is resource allocator

- resources: CPU time, memory space, storage space, I/O devices, ...
- OS is the manager of resources
- How to allocate resource to programs and users so that the systems operates efficiently and fairly?

OS is a control program

- Manage the execution of programs (prevents errors)
- Controls I/O devices

Definition of OS

Quick history:

- early experiments (mathematicians)
- military uses (code breaking, trajectory plotting, ...)
- government use (census calculations)
- general-purpose use (OS were born)

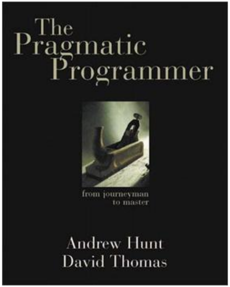
Moore's Law: # transistors doubles every 18 months

Definition of OS

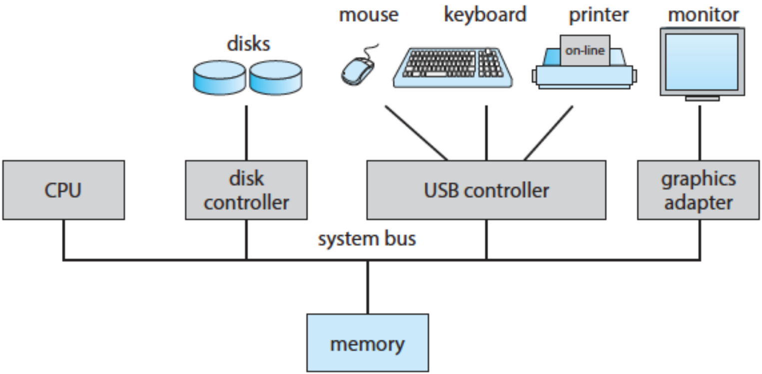
- No universally accepted definition.
- OS is the program running at all times on a computer – kernel
- system programs (device drivers, antivirus software, backup) vs. application programs
- 1998, suit against Microsoft
- middleware (Android) – framework that provides services beyond OS to application programmers
- kernel + middleware + system programs

Why study operating systems?

- Small % of programmers develop OS.
- Almost all code runs on top of OS.
- Understanding OS is crucial to efficient and secure programming.



Computer-system organisation



- device controller: local buffer + special-purpose registers
- device driver: understands the device and provides an interface to OS

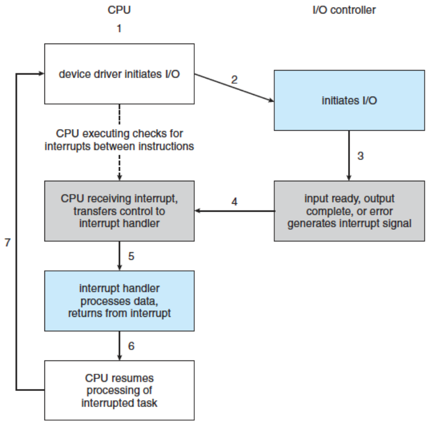
Interrupt

A typical operation: performing I/O

- device driver loads registers in the controller
- controller determines what action to take
- contoller informs the driver it has finished
- device driver returns the data to OS or status information ("device busy")
- controller informs the driver via an interrupt

Implementation

- interrupt-request line
- read the interrupt number + jump to interrupt-handler routine
- device controller raises an interrupt, CPU catches the interrupt



Implementation

In modern systems, more sophisticated features are needed:

- Defer interrupt handling during critical processing.
- Efficient way to dispatch to the proper interrupt handler for a device.
- Multilevel interrupts, so that the operating system can distinguish between high- and low-priority interrupts and can respond with the appropriate degree of urgency.
- interrupt-controller hardware

Implementation

- CPUs have two interrput-request lines: nonmaskable vs. maskable
- interrupt vector
- interrupt priority levels

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Storage notation

- bit – basic unit of computer storage: 0 or 1
- byte – 8 bits
- word – computer's native unit of data
  (64-bit acrhtecture = 64-bit words = 8-byte words)
- kilobyte, megabyte, . . .
- See `https://en.wikipedia.org/wiki/Binary_prefix`

Storage structure

- main memory – RAM, random-access memory
- RAM is volatile
- EEPROM – electrically erasable programmable read-only memory
  nonvolatile, bootstrap program
- memory: array of bytes
  `load` and `store` instruction
- CPU has registers: program counter
- instruction-execution cycle:
  - fetch an instruction from memory
  - store instruction in the instruction register
  - decode instruction
  - execution of instruction

Storage structure

Why don't we keep everything in the main memory?

- To small to store everything (programs + data) permanently.
- Main memory is volatile (what happens when power turned off?)

Computers have secondary storage:
hold large quantity of data permanently

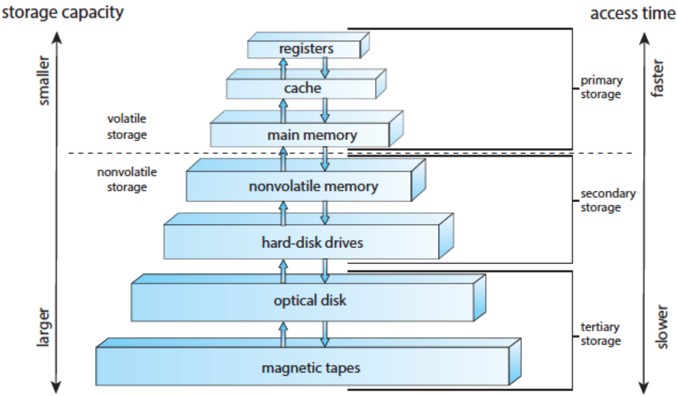
Secondary storage devices – nonvolatile memory (NVM) devices:

- hard-disk drives (HDDs)
- solid-state drives (SSDs)

Other components: cache memory, CD-ROM and blu-ray, magnetic tapes (tertiary storage), ...

Memory hierarchy

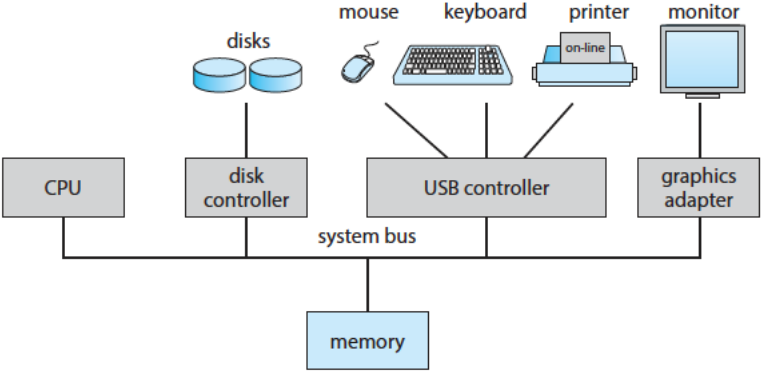
Ideal memory: fast, large, inexpensive, nonvolatile



Storage structure

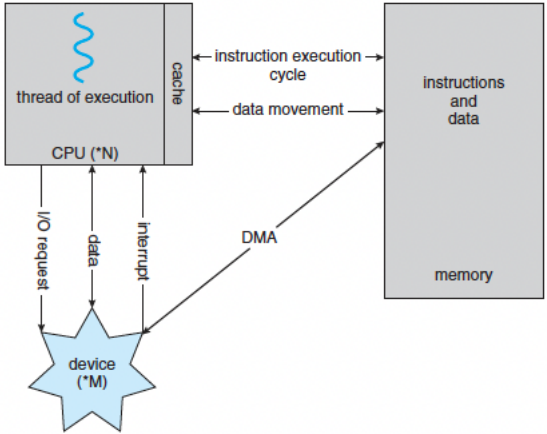
- semiconductor memory (electronic circuits)
- When we say memory we mean volatile storage
- Nonvolatile storage (NVS):
  - mechanical (HDDs, optical disks)
  - electrical (flash memory, SSDs)

I/O structure



- interrupt-driven I/O

Direct memory access (DMA)



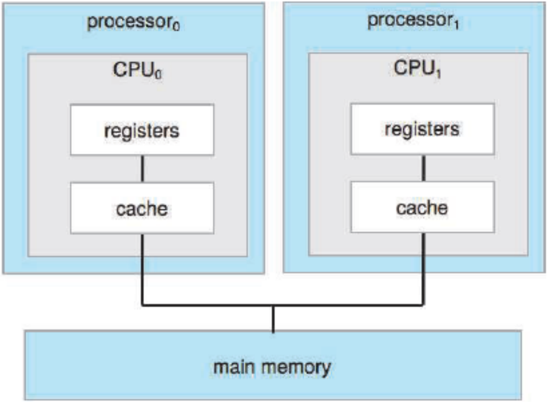
Computer-system architecture

- single-processor systems
- multiprocessor systems

Definitions:

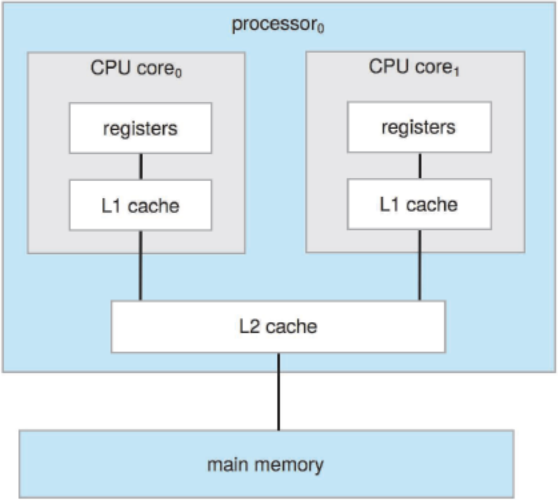
- CPU – the hardware that executes instructions
- processor – physical chip that contains one or more CPUs
- core – The basic computation unit of the CPU
- multicore – including multiple computing cores on the same CPU
- multiprocessor – including multiple processors

Symmetric multiprocessing (SMP)

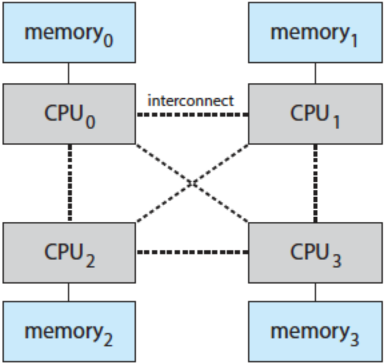


Multicore systems

Dual-core design:

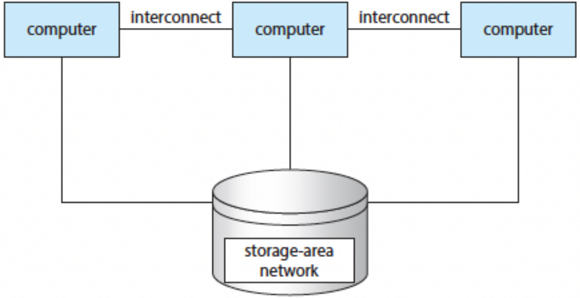


Non-uniform memory access (NUMA)



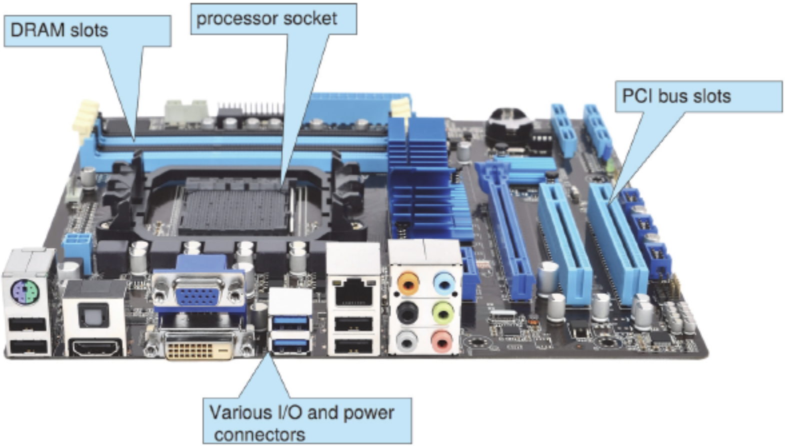
Clustered systems

Individual computers + LAN + shared storage



- high-availability services
- high-performance computing

PC motherboard



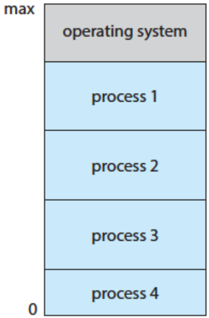
OS operations

- bootstrap program
- OS kernel
- system daemons (on Linux: `systemd`)

- trap (exception) is a software-generated interrupt
- system call

Multiprogramming & multitasking

- A process is a program in execution.
- A process occasionally has to wait for an I/O operation to complete.
- increase CPU utilisation



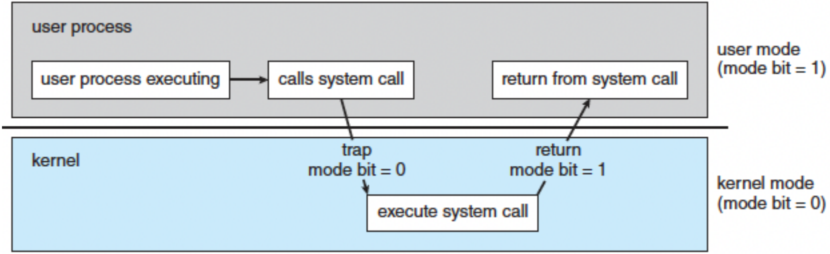
- multitasking: frequent switches (fast response time)

Multiprogramming & multitasking

- several processes in memory $\Longrightarrow$ memory management
- several processes ready to run $\Longrightarrow$ CPU scheduling
- processes isolated from one another:
  physical memory vs. logical memory
  execution of processes that are not entirely in the memory
  running programs that are larger than physical memory
- filesystem: protect files from inappropriate use
- mechanism for process communication and synchronisation

Dual-mode operation

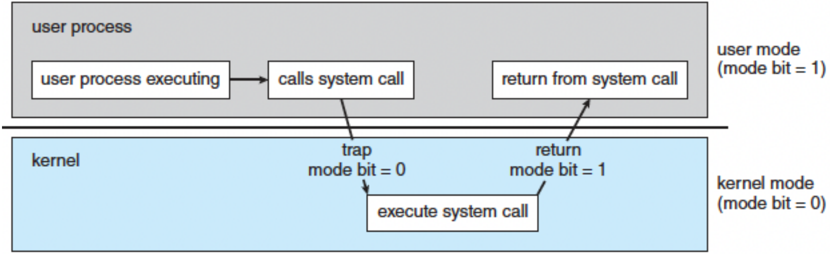
- Incorrect (or malicious) programs should not affect other programs or the OS.
- user mode vs. kernel mode (system mode)
- supported by hardware – mode bit: 0 = kernel, 1 = user



- privileged instructions

System calls

- System calls provide an interface for a user program to ask the OS to perform a privileged instruction on the user program's behalf.
- `trap` or `syscall` instruction



Timer

- Do not allow programs to get stuck in an infinite loop
- A timer raises interrupts periodically.
- On Linux: `HZ` parameter (kernel configuration) = number of interrupts per second

```
$ cat /boot/config-version | grep "CONFIG_HZ"
```

Resource management

OS is a resource manager.

Resources:

- CPU time
- memory space
- file-storage space
- I/O devices, ...

Process management

- A program in execution = process.
- A process needs certain resources.
- A program by itself is not a process.
- A program is a passive entity, a process is an active entity.
- A single-threaded process has one program counter specifying the next instruction to execute.
- A multithreaded process has multiple program counters: one for each thread.

Process management

Job of the OS:

- Creating and deleting (both user and system) processes
- Scheduling processes and threads on the CPUs
- Suspending and resuming processes
- Providing mechanisms for process communication
- Providing mechanisms for process synchronisation

Memory management

- Main memory = a large array of bytes
- Each byte has its own address.
- The CPU reads instructions from main memory during the instruction-fetch cycle.
- To process data from disk, those data must first be transferred to main memory.
- For a program to be executed, it must be loaded into memory.
- To improve the utilisation of the CPU and responsiveness, general-purpose computers must keep several programs in memory.

Memory management

Job of the OS:

- Keeping track of which parts of memory are currently being used and which process is using them.
- Allocating and deallocating memory space as needed.
- Deciding which processes (or parts of processes) and data to move into and out of memory.

Memory management

Job of the OS:

- Keeping track of which parts of memory are currently being used and which process is using them.
- Allocating and deallocating memory space as needed.
- Deciding which processes (or parts of processes) and data to move into and out of memory.

File-system management

- File = a logical storage unit
- The operating system maps files onto physical media and accesses these files via the storage devices.
- Computers can store information on several different types of physical media.
- Files represent programs (both source and object forms) and data.
- Files are organised into directories to make them easier to use.
- When multiple users have access to files, we have to control which user may access a file and how that user may access it (read, write, execute).

File-system management

Job of the OS:

- Creating and deleting files
- Creating and deleting directories to organize files
- Supporting primitives for manipulating files and directories
- Mapping files onto mass storage
- Backing up files on stable (nonvolatile) storage media

Mass-storage management

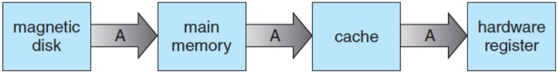
modern computer systems use HDDs and NVM devices as the principal on-line storage media for both programs and data.
Job of the OS:

- Mounting and unmounting
- Free-space management
- Storage allocation
- Disk scheduling
- Partitioning
- Protection

Cache management

Level	1	2	3	4	5
Name	registers	cache	main memory	solid-state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25-0.5	0.5-25	80-250	25,000-50,000	5,000,000
Bandwidth (MB/sec)	20,000-100,000	5,000-10,000	1,000-5,000	500	20-150
Managed by	compiler	hardware	operating system	operating system	operating system
Backed by	cache	main memory	disk	disk	disk or tape

- replacement-algorithms for software-controlled caches
- movement of information and cache coherency



I/O system management

- Hide the detailes of specific hardware devices from the user.
- UNIX: I/O subsystem components:
  - A memory-management component (includes buffering, caching, and spooling)
  - General device-driver interface
  - Drivers for specific hardware devices

Security and protection

- Protection is any mechanism for controlling the access of processes or users to the resources defined by a computer system.
- The job of security is to defend a system from external and internal attacks.