

Desktop computers

Circa 1980s

Started with resident monitor systems.

- TRS-DOS

- Apple DOS

- MS-DOS

- MacOS

File systems

- Simple single-level

There was no need for security.

- Who were you going to attack?

- Only one user.

- No hard disks.

Only one program ran at a time.

- People usually work with one program at a time.

Gradual need for multiprogramming

- Print spooling

- TSRs

- Mac Switcher

Personal computers (cont.)

Xerox (late 70s early 80s) had done some pioneering work on making computers easier to use.

This was originally on time-sharing and networked workstations.

- needed high definition graphics screens
- desktop metaphor - make the computer look like something else (even though *files* were called *files* well before this)

Macintosh (now officially called “Mac”) took this and popularized a cut-down version. MS-DOS users laughed scornfully until 1995.

Personal computers (cont.)

Eventually personal computers became as powerful as “real computers”.

Time-sharing OSs could fit on the desktop.

UNIX was the main example.

Fully featured PC operating systems were developed with

virtual memory

multiprogramming

sophisticated file systems

networking

multi-user support

e.g. MacOS X, Windows XP, Linux

Networks

1980s on

New problems

- security

- providing transparent access to resources

- developing protocols

Network Operating System

- provides file sharing

- provides communication scheme

- runs independently from other computers on the network

Distributed Operating System

- less autonomy between computers

- gives the impression there is a single operating system controlling the network.

Both are known as

loosely coupled as opposed to *tightly coupled systems*

Multiprocessor systems

Tightly coupled system – processors share memory and a clock; communication usually takes place through the shared memory.

Advantages of parallel systems

- Increased *throughput*

- Economical

- Increased reliability

 - graceful degradation

 - fail-soft systems

Symmetric multiprocessing (SMP)

- Each processor runs an identical copy of the operating system.

- Many processes can run at once without performance deterioration.

- Most modern operating systems support SMP

Asymmetric multiprocessing

- Each processor is assigned a specific task; master processor schedules and allocates work to slave processors.

- More common in extremely large systems

Real-time systems

A completely different thread of OS development.

Real-time systems have varying levels of timing constraints.

Hard real-time – must satisfy requests within definite time periods or the system fails.

e.g. robotics, air traffic control, nuclear power plants

Soft real-time – it doesn't matter too much if a time constraint is not met exactly.

e.g. Multi-media software, telephone system

Most modern OSs can handle some sort of soft real-time control.

Hard real-time control systems have to be specially designed.

PDAs or Pocket computers

PalmOS- small memory and slow processors.

The OS had to be very efficient in order to get reasonable performance. No memory protection, no virtual memory.

Windows CE – currently maintains programs in RAM when they are being used. The OS releases memory when it must or determines the program is no longer being used.

Maximum of 32 processes – does have virtual memory, each process limited to 32MB.

Both now used in phones as well as stand-alone computers.

SymbianOS – also true memory protection.

Other problems of size and battery consumption.

GUI, amount of RAM that can be kept valid.

OS design

All in one – all OS components can freely interact with each other.

MS-DOS

Early UNIX

Separate layers – see the Onion model in lecture 1.

This simplifies verification and debugging.

Very hard to get the design correct.

Can be inefficient – lots of layers to go through to get work done.

THE

OS/2

Modules – like the all-in-one but only loaded when necessary

Linux

Microkernels

Use client/server model.

Most modern general-purpose OSs use this approach.

Mach (basis for MacOS X)

Windows NT (2000)

QNX RT-OS

Virtual Machines

VM/CMS

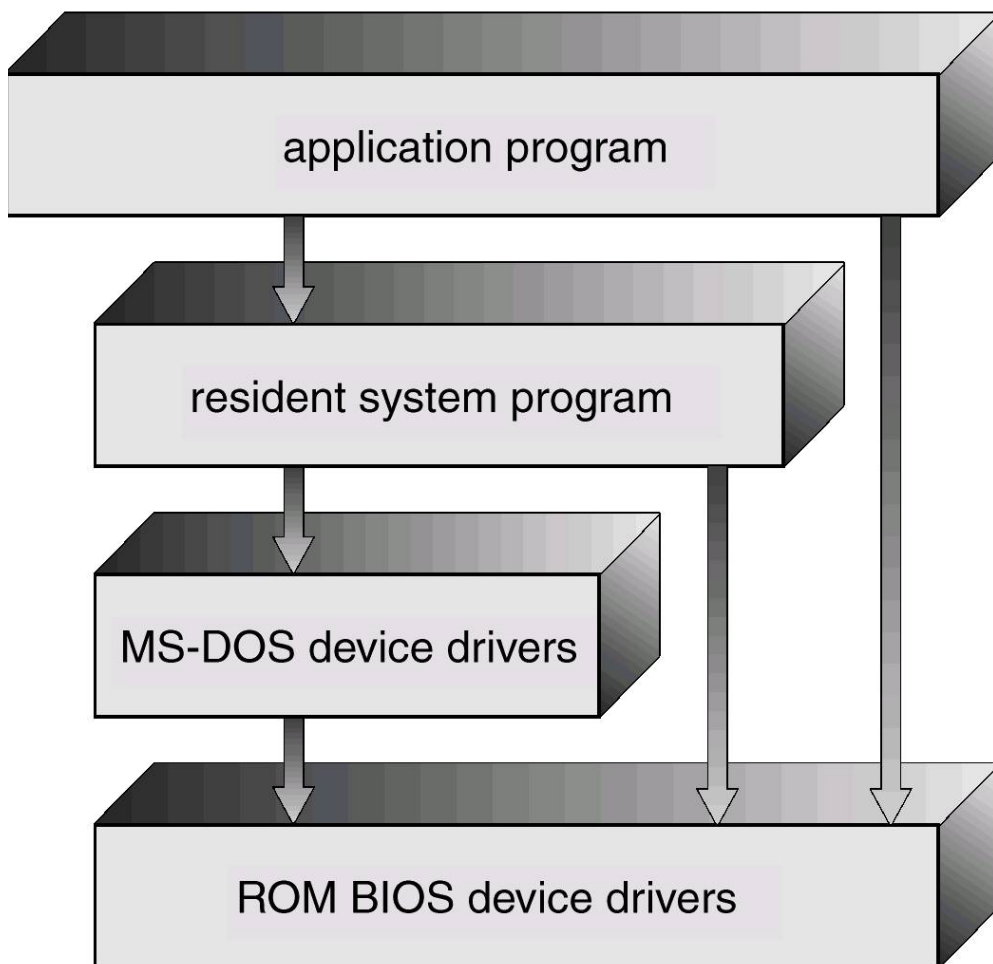
Java

MS-DOS

Written to provide the most functionality in the least space

not divided into modules

Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated



Early UNIX

The UNIX OS consists of two separable parts.

Systems programs

The kernel

Consists of everything below the system-call interface and above the physical hardware

Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level.

(the users)		
shells and commands compilers and interpreters system libraries		
<i>system-call interface to the kernel</i>		
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
<i>kernel interface to the hardware</i>		
terminal controllers terminals	device controllers disks and tapes	memory controllers physical memory

THE OS

A layered design was first used in the THE operating system.

Its six layers are as follows:

layer 5: user programs

layer 4: buffering for input and output

layer 3: operator-console device driver

layer 2: memory management

layer 1: CPU scheduling

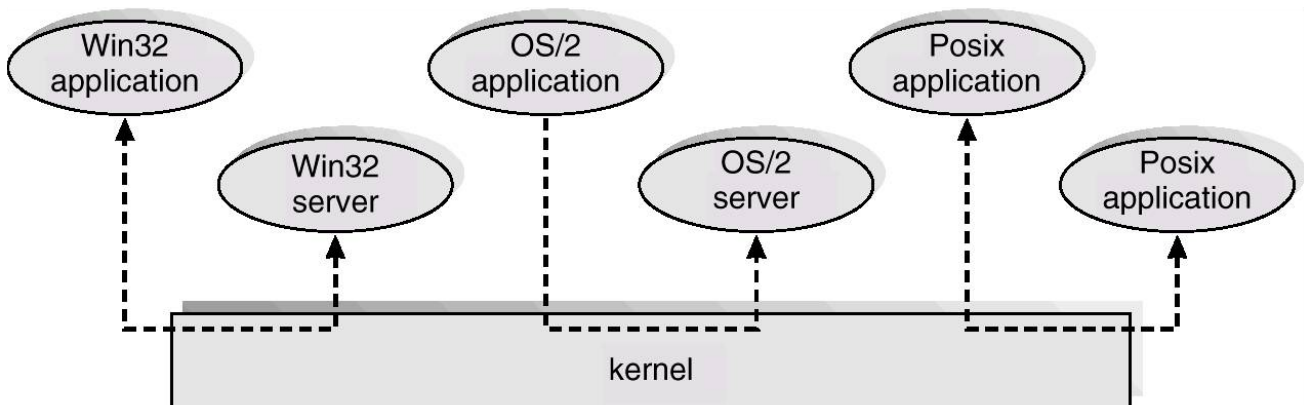
layer 0: hardware

Windows NT client/server

Windows NT (and XP) is a hybrid system.

Parts are layered and in recent versions some of the layers have been merged to improve performance.

But many OS services are provided by user-level servers
e.g. the environmental subsystems.



Virtual machines

Virtual machine systems can give everyone the OS (and hardware) that they want.

IBM's VM provided an exact copy of the hardware to the user.

Advantages

- You can choose your OS.

- You can modify or develop new OSs without crashing machines or having to reboot.

- An extra level of safety – each user's virtual machine was completely separate from all the others.

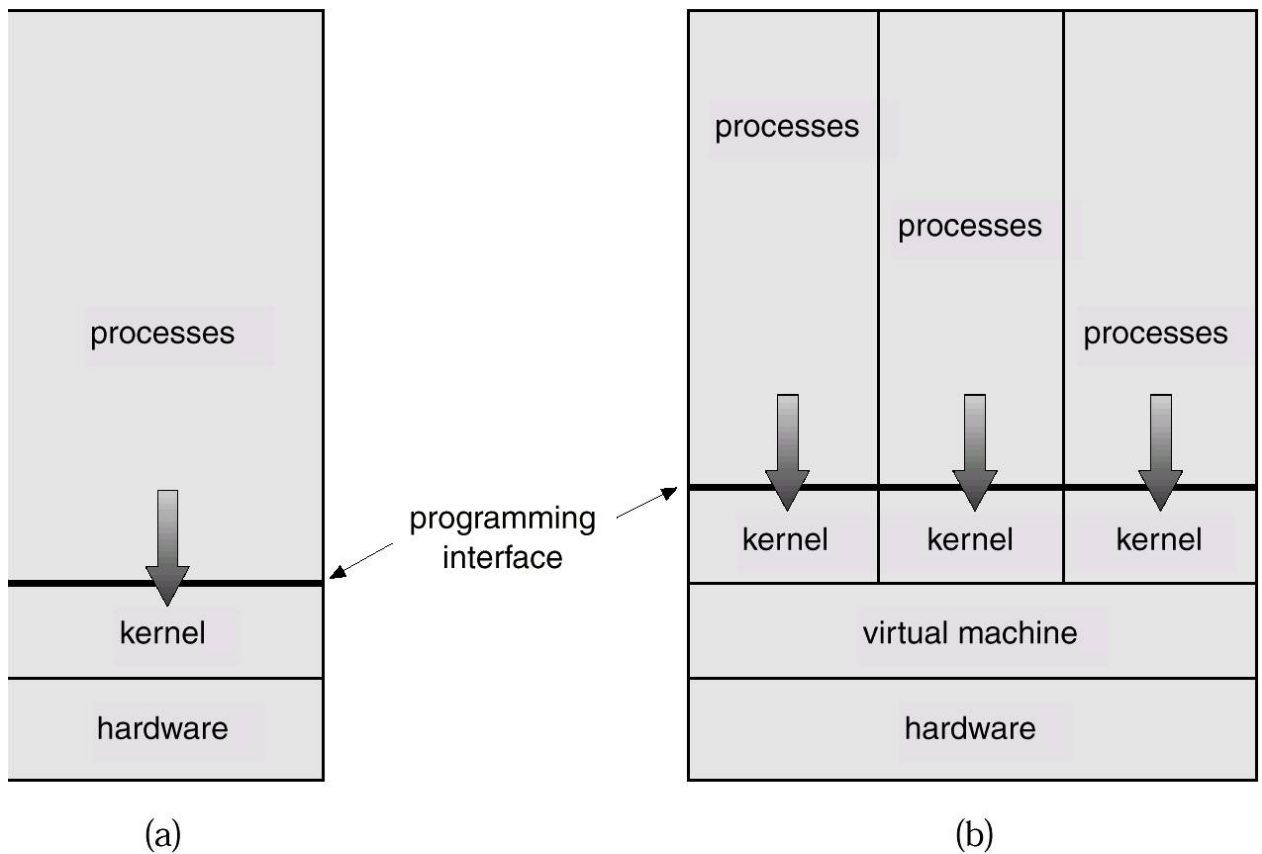
- Tricky to implement.

Disadvantages

- Some resources allocated to one VM can't be shared by another. (Virtual networks)

- An extra layer of complexity. Each layer can provide its own bugs.

Design of VM



CPU scheduling can create the appearance that users have their own processor.

Spooling and a file system can provide virtual line printers and virtual disks (minidisks).

A normal user time-sharing terminal serves as the virtual machine operator's console.

Virtual user and kernel modes are provided.

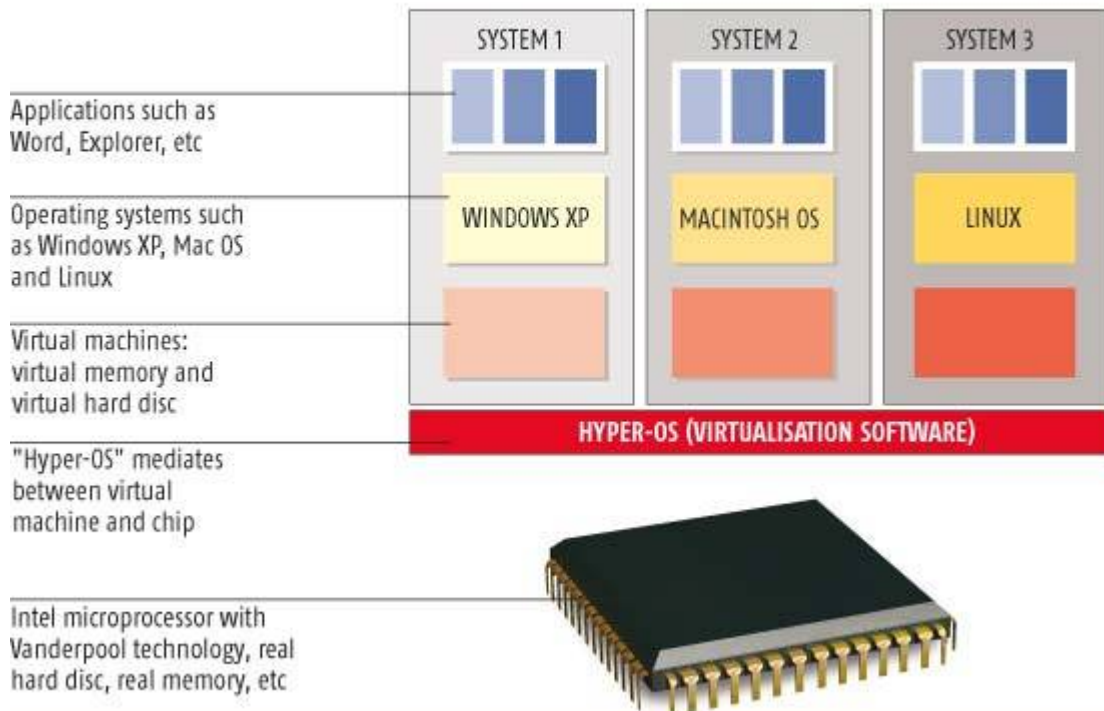
VM did very little simulation. System calls caused traps to the VM and calls back to the correct kernel.

Only privileged instructions needed to be simulated.

Vanderpool and Xen

AFTER PENTIUM

Intel's Vanderpool processor is designed to allow several operating systems to run simultaneously



Xen – software approach. Requires modifications to the OS source code to use the Xen layer. Therefore no Windows version (there was an unofficial one for research purposes).

Java VM and simulators

A different approach

Implement a different architecture on top of any hardware/OS.

Complete simulation – unless clever JIT compilation techniques are used it is significantly slower.

Earlier versions – late 70's UCSD Pascal system.

Simulating one architecture on another.

Amiga on a PC.

VirtualPC on a Mac.

VirtualPC on a PC.

VMware takes the IBM VM approach.

Run Linux on Windows.

Run Windows on Linux.

Before the next lecture

Revise your understanding of System Calls

read the textbook section 3.3 System Calls

Read textbook

3.5 System Structure

3.6 Virtual Machines

3.7 Java

3.1.8 Command-Interpreter System