Unit OS1:
Overview of Operating Systems

1.2. The Evolution of Operating Systems

Copyright Notice

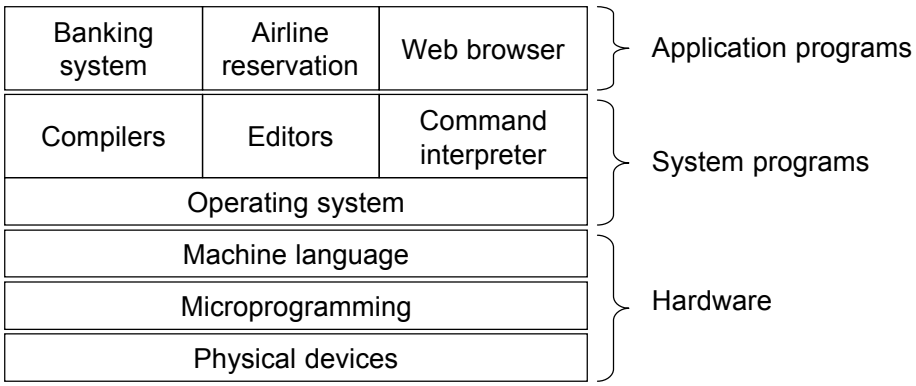
- These materials are part of the *Windows Operating System Internals Curriculum Development Kit,* developed by David A. Solomon and Mark E. Russinovich with Andreas Polze
- Microsoft has licensed these materials from David Solomon Expert Seminars, Inc. for distribution to academic organizations solely for use in academic environments (and not for commercial use)

Roadmap for Section 1.2.

- History of Operating Systems
- Tasks of an Operating System
- OS as extension of the hardware
- Main concepts: processes, files, system calls
- Operating system structuring

Operating Systems Concepts

- System software manages resources
- OS hides complexity of underlying hardware
- Layered architectures



An operating system is a program that acts as an intermediary between a user of a computer and the computer hardware. The purpose of an operating system is to provide an environment in which a user can execute programs. The primary goal is thus to make the computer system convenient to use. A secondary goal is to use the computer hardware in an efficient manner.

A computer system can be divided roughly into four components: the hardware, the operating system, the applications programs, and the users. The hardware – the central processing unit (CPU), memory, and input/output (I/O) devices – provides the basic computing resources. The application programs – such as compilers, database systems, games, and business programs – define the ways in which these resources are used to solve the computing problems of the users. There may be many different users and many different application programs. The operating system controls and coordinates the use of hardware among the various applications programs for the various users.

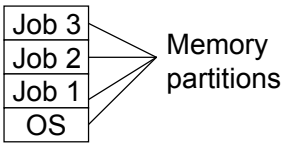
History of operating systems

- Batch processing

  The elements of the basic IBM 1401 system are the 1401 Processing Unit, 1402 Card Read-Punch, and 1403 Printer.

- Punching cards programming

Multiprocessing



Windows Operating System Internals - by David A. Solomon and Mark E. Russinovich with Andreas Polze 5

Announced October 5, 1959 and withdrawn February 8, 1971.

The following is the text of an IBM Data Processing Division press fact sheet distributed on October 5, 1959.

The all-transistorized IBM 1401 Data Processing System places the features found in electronic data processing systems at the disposal of smaller businesses, previously limited to the use of conventional punched card equipment.

These features include: high speed card punching and reading, magnetic tape input and output, high speed printing, stored program, and arithmetic and logical ability. The elements of the basic 1401 system are the 1401 Processing Unit, 1402 Card Read-Punch, and 1403 Printer. Configurations include a card system, a tape system, and a combination of the two.

The 1401 may be operated as an independent system, in conjunction with IBM punched card equipment, or as auxiliary equipment to IBM 700 or 7000 series systems. The 1401 performs functions previously requiring a number of separate machines: card reading and punching, separation of output cards, calculating, and printing.

Read more at:

http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_PP1401.html

The Evolution of Operating System Functionality

- Batch Job Processing
  - Linkage of library routines to programs
  - Management of files, I/O devices, secondary storage
- Multiprogramming
  - Resource managment and sharing for multiple programs
  - Quasi-simultaneous program execution
  - Single user
- Multiuser/Timesharing Systems
  - Management of multiple simultaneous users interconnected via terminals
  - Fair resource management: CPU scheduling, spooling, mutual exclusion
- Real-Time Systems (process control systems)
  - Management of time-critical processes
  - High requirements with respect to reliability and availability

Tasks of an Operating System

- Processor management - Scheduling
  - Fairness
  - Non-blocking behavior
  - Priorities
- Memory management
  - Virtual versus physical memory, memory hierarchy
  - Protection of competing/conurrent programs
- Storage management – File system
  - Access to external storage media
- Device management
  - Hiding of hardware dependencies
  - Management of concurrent accesses
- Batch processing
  - Definition of an execution order; throughput maximization

Kernel- and User Mode Programs

Typical functionality implemented in either mode:

Kernel:

- Privileged mode
- Strict assumptions about reliability/security of code
- Memory resident
  - CPU-, memory-, Input/Output managment
  - Multiprocessor management, diagnosis, test
  - Parts of file system and of the networking interface

User Space:

- More flexible
- Simpler maintenance and debugging
  - Compiler, assembler, interpreter, linker/loader
  - File system management, telecommunication, network management
  - Editors, spreadsheets, user applications

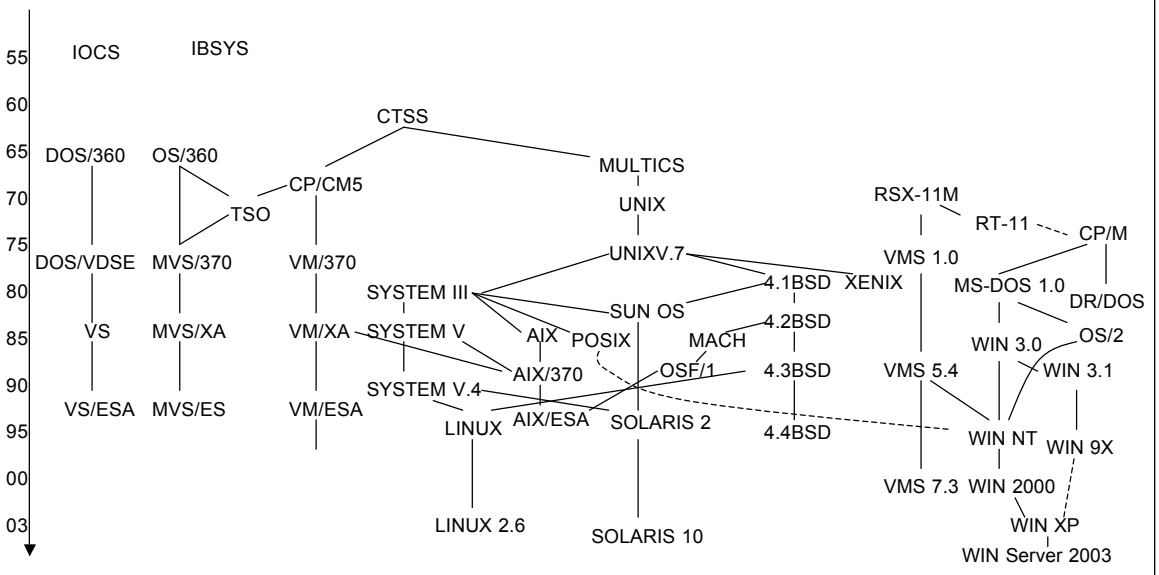
Layered Model of Operating System Concepts

nr	name	typical objects	typical operations
1	Integrated circuits	register, gate, bus	Nand, Nor, Exor
2	Machine language	instruction counter, ALU	Add, Move, Load, Store
3	Subroutine linkage	procedure block	Stack Call, JSR, RTS
4	Interrupts	interrupt handlers	Bus error, Reset
5	Simple processes	process, semaphore	wait, ready, execute
6	Local memory	data block, I/O channel	read, write, open, close
7	Virtual model	page, frame	read, write, swap
8	Process communication	channel (pipe), message	read, write, open
9	File management	files	read, write, open, copy
10	Device management	ext.memory, terminals	read, write
11	I/O data streams	data streams	open, close, read, write
12	User processes	user processes	login, logout, fork
13	Directory management	internal tables	create, delete, modify
14	Graphical user interface	window, menu, icon	OS system calls

OS acts as Extension of Hardware

- System view: layered model of OS
  - Implementation details on one layer are hidden from higher layers
- Same machine, different operating systems:
  - IBM PC: DOS, Linux, NeXTSTEP, Windows NT, SCO Unix
  - DEC VAX: VMS, Ultrix-32, 4.3 BSD UNIX
- Same OS, different machines: UNIX
  - PC (XENIX 286, APPLE A/UX)
  - CRAY-Y/MP (UNICOS - AT&T Sys V)
  - IBM 360/370 (Amdahl UNIX UTS/580, IBM UNIX AIX/ESA)
- Windows XP (or Windows NT/2000)
  - Intel i386 (i486 an NT 4.0), Alpha, PowerPC, MIPS, Itanium

Operating Systems Evolution



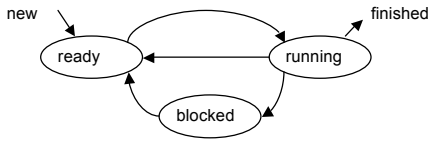
Windows Operating System Internals - by David A. Solomon and Mark E. Russinovich with Andreas Polze 11

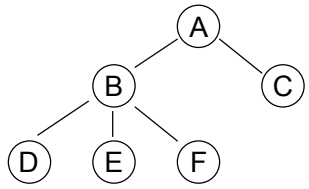
This diagram represents only a small fraction of the world of desktop operating systems. Due to space limitations, some important candidates had to be excluded from the picture.

For comprehensive diagrams showing the Unix and Windows OS evolution see:
http://www.levenez.com

Main Concepts: processes

- Processes, process table, core image
- Command interpreter, shell
- Child processes



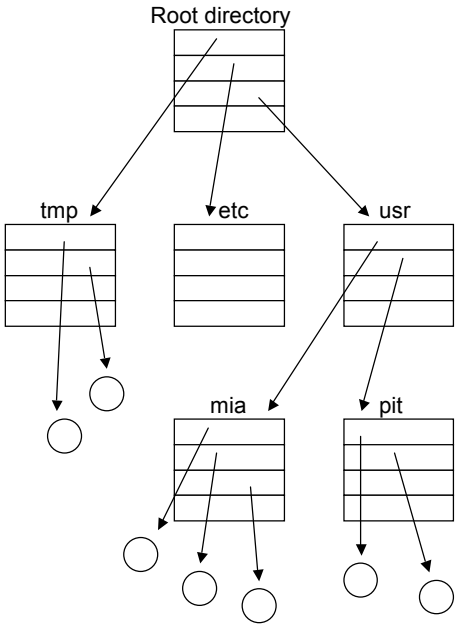


Process tree

- Scheduling, signals
- User identification, group identification

Main Concepts: Files

- Files, directories, root
- Path, working directory
- Protection, rwx bits
- File descriptor, handle
- Special files, I/O devices
- Block I/O, character I/O
- Standard input/output/error
- pipes



Main concepts: system calls

- User programs access operating system services via system calls
- Parameter transmission via trap, register, stack

  count=read(file, buffer, nbytes);
- 5 general classes of system calls:
  - Process control
  - File manipulation
  - Device manipulation
  - Information maintenance
  - communications

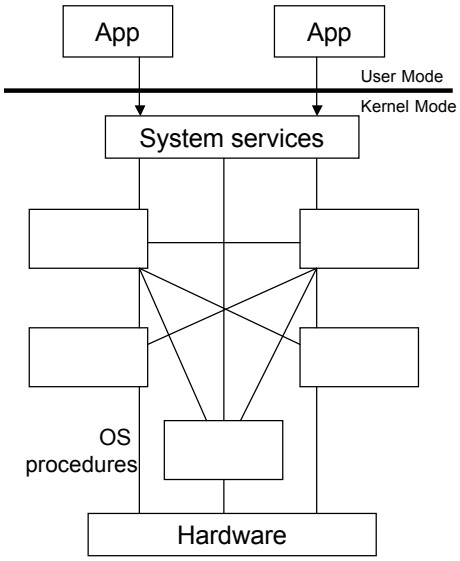
Main concepts: shell

- Command interpreter
- Displays prompt, implements input/output redirection
- Background processes, job control, pseudo terminals

```
$ date
$ date >file
$ sort <file1 >file2
$ cat file1 file2 file3 > /dev/lp1
$ make all >log 2>&1 &
```

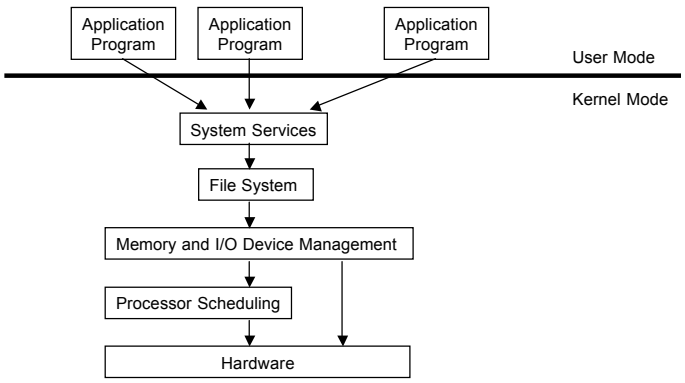
Structuring of Operating Systems

- **Monolithical systems**
- Unstructured
- Supervisor call changes from user mode into kernel mode



Layered OS

- Each layer is given access only to lower-level interfaces

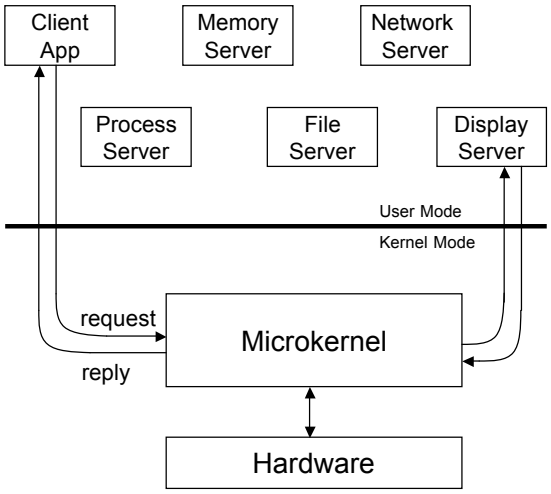


Microkernel OS (Client/server OS)

Kernel implements:

- Scheduling
- Memory Management
- Interprocess communication (IPC)

User-mode servers



Windows Operating System Internals - by David A. Solomon and Mark E. Russinovich with Andreas Polze 18

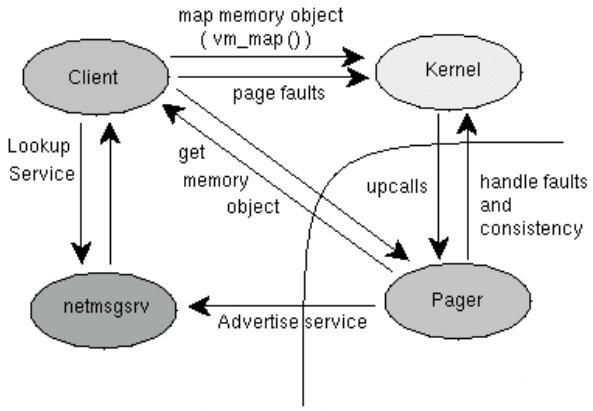
Is Windows a Microkernel-Based System?

Although some claim it as such, Windows isn't a microkernel-based operating system in the classic definition of microkernels, where the principal operating system components (such as the memory manager, process manager, and I/O manager) run as separate processes in their own private address spaces, layered on a primitive set of services the microkernel provides. For example, the Carnegie Mellon University Mach operating system, a contemporary example of a microkernel architecture, implements a minimal kernel that comprises thread scheduling, message passing, virtual memory, and device drivers. Everything else, including various APIs, file systems, and networking, runs in user mode. However, commercial implementations of the Mach microkernel operating system typically run at least all file system, networking, and memory management code in kernel mode. The reason is simple: the pure microkernel design is commercially impractical because it's too inefficient.

Mach Microkernel OS
Extended Memory Managment

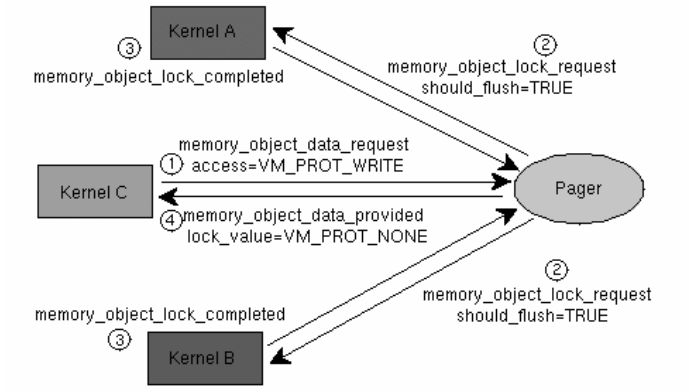
Paging handled by user-space server

Port: comm. endpoint, network-wide



Mach Microkernel OS
Distributed Shared Memory System

- Access remote memories, port access rights - ACL



Windows 2000/NT background/history

Dave Cutler:

- OS Developer at DEC since 1971
- RSX-11M, PDP-11 (16 bit mini) **„Size is the Goal“**
  - Multitasking, hierarchical file system, real-time scheduling
  - Application swapping, utilities
  - 32 kb of memory (!)
  - 16 kb Kernel, 16 kb utilities, overlay structures, assembly language
  - Time-to-market: 18 months
- Lack of address bits: VAX architecture (32 bit)
  - Most successful architecture in ´70s and ´80s

DEC Virtual Memory System (VMS) and MS Windows NT

- Cutler was leader of VMS development effort
- VAX-11 hardware had PDP-11 compatibility mode
  - RSX-11M was the compatibility environment to be supported by VMS
  - Binary and file system compatibility
- Biggest mistake: VMS written in assembly language
  - Size restrictions, no compiler available, engineering expertise
- Summer `88: call from Bill Gates
  - New OS for PC architecture
  - Portability, security, POSIX, compatibility, multiprocessor, extensibility
  - Similar goals as for PDP-11/VAX transition
- Windows NT came to market in 1993

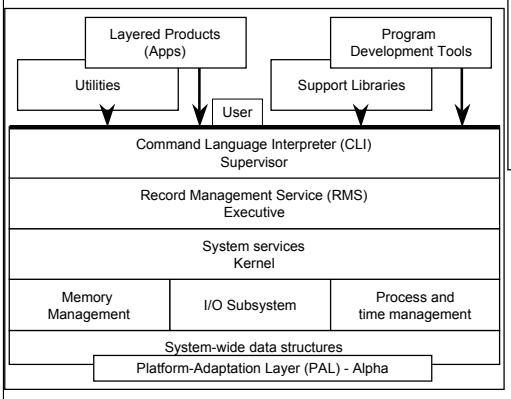
DEC had no fewer than 10 PDP-11 operating systems

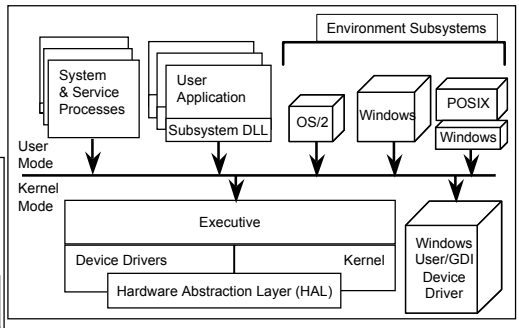
The first time-sharing system for the PDP-11 was written by a user and carried to Digital when he joined the company as an employee. This was RSX-11A (Resource Sharing Executive). RSX-11D was much improved and added a file system, Files-11. RSX-11M (M for medium) was a scaled down version of the OS.

Cutler was involved with RSX-11M, he was leader of the VMS development effort and worked later on MicroVAX project

VMS and Windows NT
- a bird's-eye view on architectures

Layered design for VAX/VMS operating system





Windows NT high-level architecture

Windows Operating System Internals - by David A. Solomon and Mark E. Russinovich with Andreas Polze 23

The VMS operating system and the VAX architecture were designed at the same time - several features of the VAX architecture can be traced to specific usages in the operating system.

One particular design feature is the four modes of protection enforced by the hardware (Kernel, Executive, Supervisor, and User).

The full instruction set of the VAX is available to programs executing in kernel mode. The entire VMS database (scheduler, IS, and memory) is accessible only in kernel mode. Most of the system services operate in kernel mode.

The executive modes is reserved for RMS, the Record Management Services, responsible for managing all files on disk. The Command Language Interpreter (CLI) executes in supervisor mode, calling both RMS and VMS kernel services. The least privileged and by far the greatest number of programs execute in user mode, as do the utilities, compiler, editors, and certain Digital products.

The architecture of Windows NT also uses a layered design. However, Windows NT uses only two different protection modes, namely kernel mode (privileged) and user mode. Interestingly enough, the VAX/VMS design has influenced the feature set of other processor architectures as well - even the Intel 80368 series of CPUs supports four distinct protection modes (ring 0..3).

Further Reading

- Dennis M. Ritchie,
  The Evolution of the Unix Time-sharing System,
  - in Proc. of Lang. Design and Programming Meth. Conf., Sydney, Australia, Sept 1979, Lecture Notes in Computer Science #79, Springer-Verlag, 1980.
- David Donald Miller,
  OpenVMS Operating System Concepts,
  - 2nd Ed., Digital Press, 1997.
  - History of Digital Operating Systems (pp. 447 ff.)
- Mark E. Russinovich and David A. Solomon,
  Microsoft Windows Internals,
  - 4th Edition, Microsoft Press, 2004.
  - Historical Perspective (pp. xix ff.)