

CSE 120

Principles of Operating Systems

Fall 2001

Lecture 2: Operating System Modules,
Interfaces, and Structures

Geoffrey M. Voelker

Modules, Interfaces, Structure

- We roughly defined an OS as the layer of software between hardware and applications
- Now we're going to survey the support OSes provide to applications
 - Modules – OS services and abstractions
 - Interfaces – operations supported by components
 - Structure – how components get hooked together

OS Module Overview

- Common OS modules
 - Processes
 - Memory
 - I/O
 - Secondary storage
 - Files
 - Protection
 - Accounting
 - Command interpreter (shell)
- We'll survey each module and discuss its interface

September 19, 2001

CSE 120 – Lecture 2 – Modules, Interfaces, and Structures

3

Process Module

- An OS executes many kinds of activities
 - User programs
 - Batch jobs or command scripts
 - System programs (daemons): print spoolers, name servers, file servers, Web servers, etc.
- Each “execution entity” is encapsulated in a process
 - A process includes both the program (code, data) and execution context (PC, regs, address space, resources, etc.)
- Process module manages processes
 - Creation, scheduling, deletion, etc.

September 19, 2001

CSE 120 – Lecture 2 – Modules, Interfaces, and Structures

4

Process Interface

- Process module interface
 - Create a process
 - Delete a process
 - Suspend a process
 - Resume a process
 - Inter-process communication
 - » Transfer, share data
 - Inter-process synchronization
 - Process relationships
 - » Parent, child, process groups

September 19, 2001

CSE 120 – Lecture 2 – Modules, Interfaces, and Structures

5

Memory

- Primary memory is the direct access storage for CPU
 - Programs must be stored in memory to execute
 - Interacts with process module
- Operating systems
 - Allocate memory for programs (explicitly and implicitly)
 - Deallocate memory when needed (by rest of system)
 - Maintain mappings from virtual to physical memory (page tables)
 - Decide how much memory to allocate to each process
 - » Large space of policy decisions
 - Decide when a process should be removed from memory
 - » More policy decisions

September 19, 2001

CSE 120 – Lecture 2 – Modules, Interfaces, and Structures

6

I/O

- Much of an OS deals with device I/O
 - One of the main reasons we use OSes
 - Hundreds of thousands of lines of code in NT for I/O, drivers
- The OS provides a standard interface between programs (user or system) and devices
 - File system (disks), sockets (network), frame buffer (video)
- Device drivers are the routines responsible for controlling I/O devices
 - OS defines an interface for each class of devices (e.g., disks)
 - A driver implements interface, encapsulates device-specific knowledge (initiation and control, interrupt handling, errors)

September 19, 2001

CSE 120 – Lecture 2 – Modules, Interfaces, and Structures

7

Secondary Storage

- Secondary storage (disk) is the persistent memory
 - It endures system failures (for the most part)
- Low-level OS routines are often responsible for low-level disk functions
 - Read/write blocks
 - Schedule requests (optimize arm movement)
 - Device errors
- Usually independent of file system
 - Although there might be cooperation (e.g., free space management)
 - Low-level knowledge can help FS performance (placement)

September 19, 2001

CSE 120 – Lecture 2 – Modules, Interfaces, and Structures

8

File System

- Secondary storage devices are too crude to use directly for long-term storage
 - Read/write physical device blocks too low-level for programs
- The file system provides a much higher level, more convenient abstraction for persistent storage
 - Objects (files, directories) and interfaces (read, write, etc.)
- Files are the basic storage entity
 - A file is a named collection of persistent information
- Directories are special files that contain the names of other files + metadata (data about files, attributes)
 - Directories have all properties of files (“inheritance”)

September 19, 2001

CSE 120 – Lecture 2 – Modules, Interfaces, and Structures

9

File System Interface

- File system interface provides standard file operations
 - Existence: File/directory creation, deletion
 - Manipulation: open, read, write, append, rename, close, etc.
 - Sometimes higher-level operations
 - » File copy, change notification (NT)
 - » Records (IBM)
- File system also provides general services
 - Backup
 - Consistency
 - Accounting and quotas

September 19, 2001

CSE 120 – Lecture 2 – Modules, Interfaces, and Structures

10

Protection

- Protection is general mechanism throughout OS
- All objects (resources) need protection
 - Processes
 - Memory
 - Devices
 - Files
- Protection mechanisms help to prevent errors as well as prevent malicious destruction
 - E.g., running as root

Accounting

- General facility for keeping track of resource usage for all system objects
 - Quotas in the file system (Unix: “quota -v”)
 - Memory usage (Unix: “man limit”)
 - Process resource usage (Unix: “rusage <command>”)
- Resource usage might be used to bill customers
 - In world of PCs, might seem strange
 - In world of mainframes and minicomputers, crucial
 - » Departments, users billed for CPU time
 - IBM mainframe “turbo” switch

Command Interpreter (Shell)

- Process that handles
 - Handles user input (commands)
 - Manages subprocesses
 - Executes script files (files of commands)
- On some systems, CI is part of OS
 - Users constrained to use that CI (DOS)
- Others, it is just another user-level process
 - Unix shell
 - Any program can be a CI (sh, csh, ksh, bash, etc.)
- Or, there may not be a command language at all
 - MacOS (hey, where's the shell?)

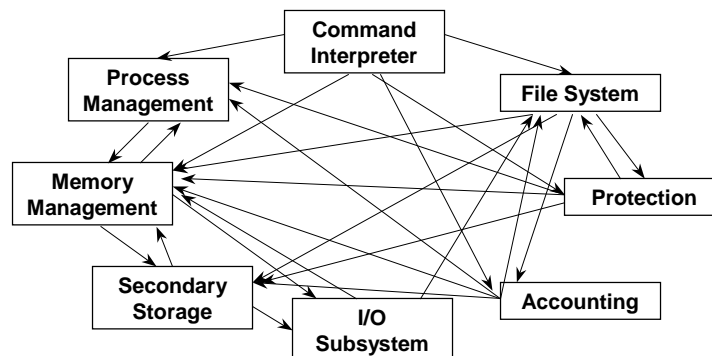
September 19, 2001

CSE 120 – Lecture 2 – Modules, Interfaces, and Structures

13

The Challenge of Structure

- It is clear what modules an OS should provide
- Not so clear how to hook them together (well)...



September 19, 2001

CSE 120 – Lecture 2 – Modules, Interfaces, and Structures

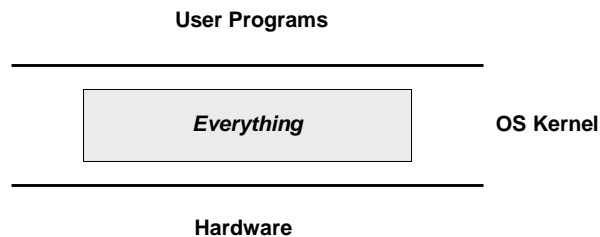
14

OS Structure

- An OS consists of all of the above modules, others we haven't discussed, system programs (privileged and non-privileged), etc.
- The challenge
 - How do we organize it all?
 - What are the modules, where do they exist?
 - How do they cooperate?
- How do we build a complex software system that is:
 - Large and complex
 - Concurrent
 - Long-lived and evolving
 - Performance-critical

Monolithic Kernels

- Traditionally, OSes such as Unix are built as a monolithic kernel (BSD, SYSV, Solaris, Linux, etc.)



Monolithic Problems

- Monolithic kernels have one great feature
 - The overhead of inter-module interaction is low
- Monolithic kernels have a number of problems
 - Hard to understand
 - Hard to modify
 - Hard to maintain
 - Unreliable (a bug in one module can take down entire system)
- From the beginning, OS designers have sought ways to organize the OS to simplify design and construction

September 19, 2001

CSE 120 – Lecture 2 – Modules, Interfaces, and Structures

17

Layering

- An early approach is layering
 - Each system service is a layer
 - Each layer defines and implements an abstraction for the layer above it
 - Layers in effect “virtualize” the layer below
- Approach first used by Dijkstra’s THE system (1968)
 - Analogous to protocol layers in networking stacks

September 19, 2001

CSE 120 – Lecture 2 – Modules, Interfaces, and Structures

18

THE System

- Levels see a virtual machine provided by lower level
 - Level 1 (CPU scheduling)
 - Level 2 (memory management) sees virtual processors
 - Level 3 (console) sees VM (segments)
 - Level 4 (device buffers) sees a “virtual console”
 - Level 5 (user programs) sees “virtual” I/O drivers
- System composed of a set of sequential processes
 - Processes communicate via synchronization in memory
- Key: Each layer could be tested and verified independently

Layering Problems

- Layered systems are hierarchical, but real systems have much more complex interactions
 - File system requires VM services (data structures, buffers)
 - VM requires file system services (backing store)
 - What do you do?
- Layering not flexible enough
- Can have poor performance (depends on how layers interact)
- Systems often modeled as layered structures, but not built as such

Microkernel Structure

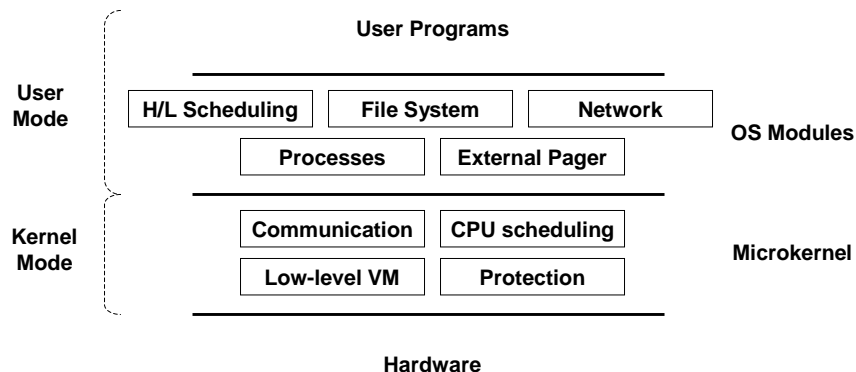
- Microkernel structures were hot in late 80s, early 90s
- Minimize kernel services, implement remaining OS modules as user-level processes
 - Better reliability (file system dies, restart it...)
 - Easier to extend and customize (start a new process)
 - Support multiple coexisting OS implementations
- But, mediocre performance
 - Overhead of inter-module interaction is high (c.f. monolithic)
- First microkernel system was Hydra (CMU, 1970)
- Others: Mach (CMU), Chorus (French Unix-like system), NT (Microsoft), OSX (Apple)

September 19, 2001

CSE 120 – Lecture 2 – Modules, Interfaces, and Structures

21

Microkernel Structure

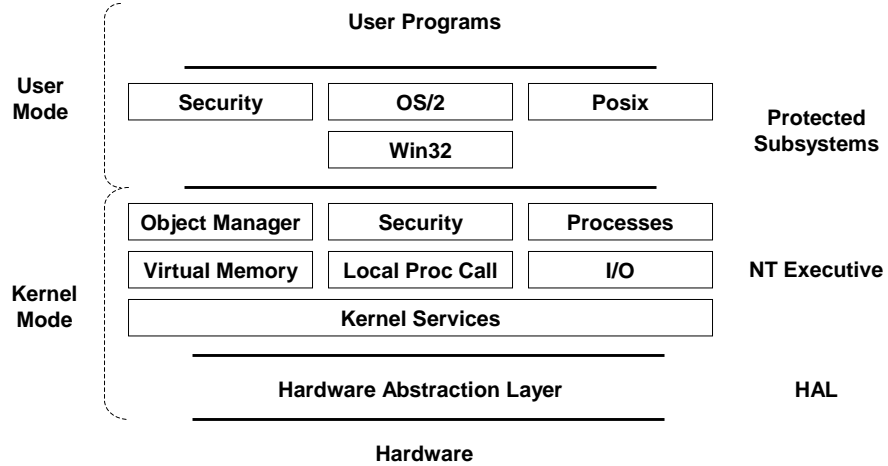


September 19, 2001

CSE 120 – Lecture 2 – Modules, Interfaces, and Structures

22

Windows NT 3.1 (1993)



September 19, 2001

CSE 120 – Lecture 2 – Modules, Interfaces, and Structures

23

Next time...

- Read Chapter 12

September 19, 2001

CSE 120 – Lecture 2 – Modules, Interfaces, and Structures

24