Process Management

- OS manages many kinds of activities:
  - User programs
  - System programs: printer spoolers, name servers, file servers, etc.
- Each is encapsulated in a *process*
  - A process includes the complete execution context (code, data, PC, registers, OS resources in use, etc.)
  - A *process* is ***not*** a *program*
    - A process is ***one*** instance of a program ***in execution***; many processes can be running the same program
- OS must:
  - Create, delete, suspend, resume, and schedule processes
  - Support inter-process communication and synchronization, handle deadlock

Memory Management

- Primary (Main) Memory
  - Provides direct access storage for CPU
  - Processes must be in main memory to execute
- OS must:
  - Mechanics
    - Keep track of memory in use
    - Keep track of unused (“free”) memory
    - Protect memory space
    - Allocate, deallocate space for processes
    - Swap processes: memory <–> disk
  - Policies
    - Decide when to load each process into memory
    - Decide how much memory space to allocate each process
    - Decide when a process should be removed from memory

File System Management

- File System
  - Disks (secondary storage) provide long-term storage, but are awkward to use directly
  - *File system* provides files and various operations on files
    - A *file* is a long-term storage entity, a named collection of persistent information that can be read or written
    - A file system supports directories, which contain files and other directories
      - Name, size, date created, date last modified, owner, etc.
- OS must:
  - Create and delete files and directories
  - Manipulate files and directories
    - Read, write, extend, rename, copy, protect
  - Provide general higher-level services
    - Backups, accounting, quotas

Disk Management

- Disk
  - The actual hardware that sits underneath the file system
  - Large enough to store all user programs and data, application programs, entire OS
  - Persistent — endures system failures
- OS must:
  - Manage disk space at low level:
    - Keep track of used spaces
    - Keep track of unused (free) space
    - Keep track of “bad blocks”
  - Handle low-level disk functions, such as:
    - Scheduling of disk operations
    - Head movement
  - Note fine line between disk management and file system management

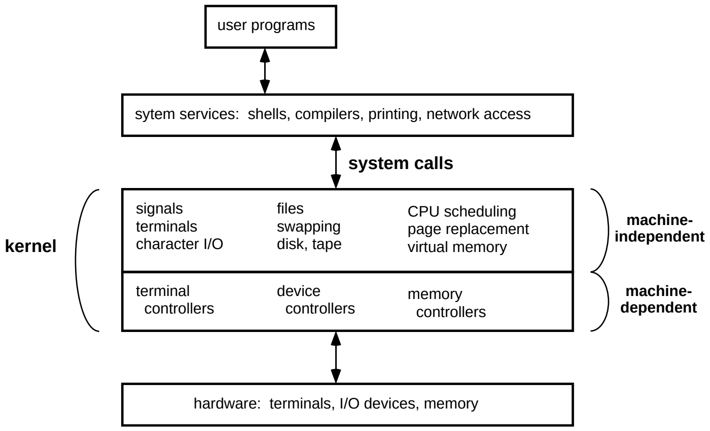
System Calls

- Process control
  - end/abort this program
  - load/execute another program
  - create/terminate a process
    - get/set attributes
    - wait specified time
    - wait for event, signal event
- File manipulation
  - create/open/read/write/close/delete file
  - get/set attributes
- Device manipulation
  - request/read/write/release device
- Information
  - get/set time/date

One OS Structure: Layers

- Another approach: layered OS
  - Divide OS into layers
  - Each layer uses services provided by next lower layer
    - User programs
    - Shell & compilers
    - CPU scheduling & memory management
    - Device drivers
    - Hardware
  - Advantages: modularity, simplicity
    - Disadvantages: performance
- Examples: THE (1968), Venus (1972), OS/2 (1988)
  - Not a very popular approach

Another OS Structure: Large Kernel



user programs

sytem services: shells, compilers, printing, network access

system calls

kernel

signals
terminals
character I/O | files
swapping
disk, tape | CPU scheduling
page replacement
virtual memory — machine-independent

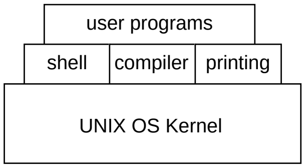
terminal controllers | device controllers | memory controllers — machine-dependent

hardware: terminals, I/O devices, memory

- The *kernel* is the protected part of the OS that runs in kernel mode
  - Critical OS data structures and device registers are protected from user programs
  - Can use privileged instructions

Coping with Hugeness

- Ideal:

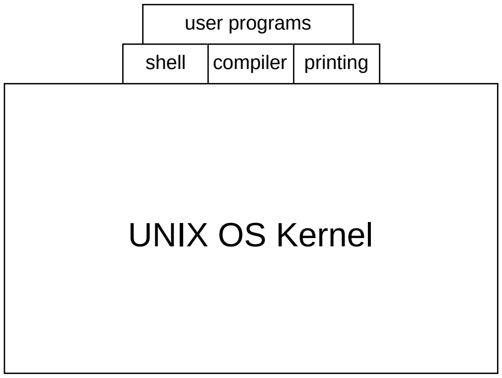


user programs

shell | compiler | printing

UNIX OS Kernel

- Reality:

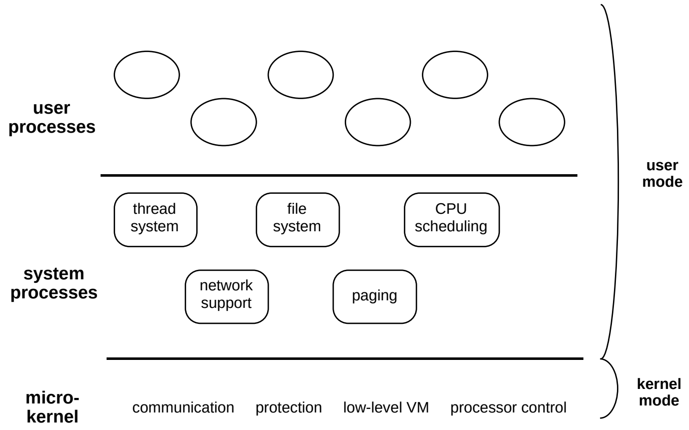


user programs

shell | compiler | printing

UNIX OS Kernel

Another OS Structure: Microkernel



user processes

user mode

thread system

file system

CPU scheduling

system processes

network support

paging

micro-kernel

communication

protection

low-level VM

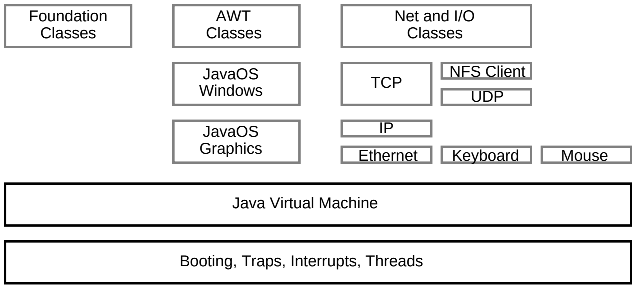
processor control

kernel mode

- Goal is to minimize what goes in the kernel, implementing as much of the OS as possible in user-mode processes
  - Better reliability, easier extension
  - Lower performance (unfortunately)
- Examples: Mach (US), Chorus (France), Windows NT

The Future? Network Operating Systems

- Sun’s JavaOS architecture:



Foundation Classes

AWT Classes

Net and I/O Classes

JavaOS Windows

TCP

NFS Client

UDP

JavaOS Graphics

IP

Ethernet

Keyboard

Mouse

Java Virtual Machine

Booting, Traps, Interrupts, Threads

Details at http://java.sun.com/doc/white_papers.html

- No disk
- OS can only run a net browser
  - Get whatever the OS needs over the net
  - Get whatever application programs are needed over the net (as Java *applets*)