Operating Systems

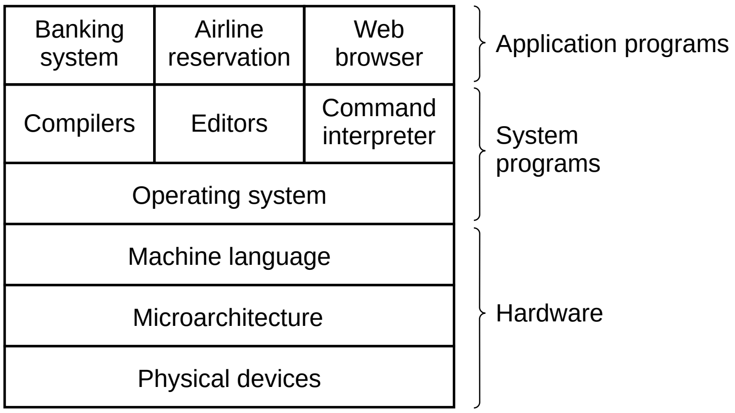
Purpose

- Standardized basis for utilizing the computer HW-resources

Functionalities

- Management of activities (processes, threads, IPC)
- Memory and file management
- Input/output: User interface, networking, special devices
- Security
- System administration and control

System Layers



Banking system | Airline reservation | Web browser — Application programs

Compilers | Editors | Command interpreter — System programs

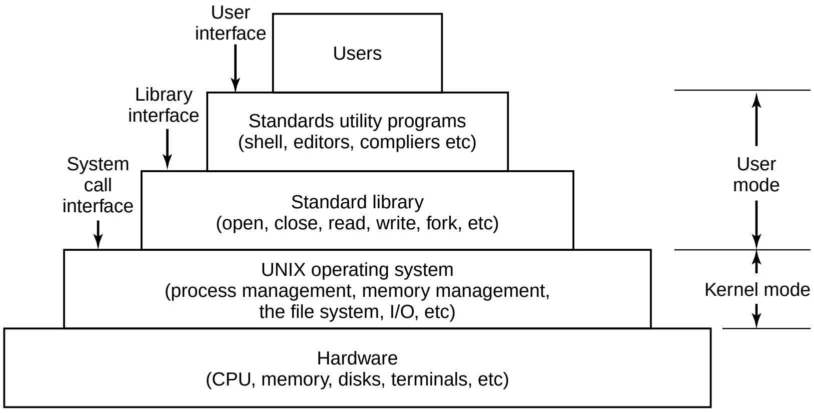
Operating system

Machine language — Hardware

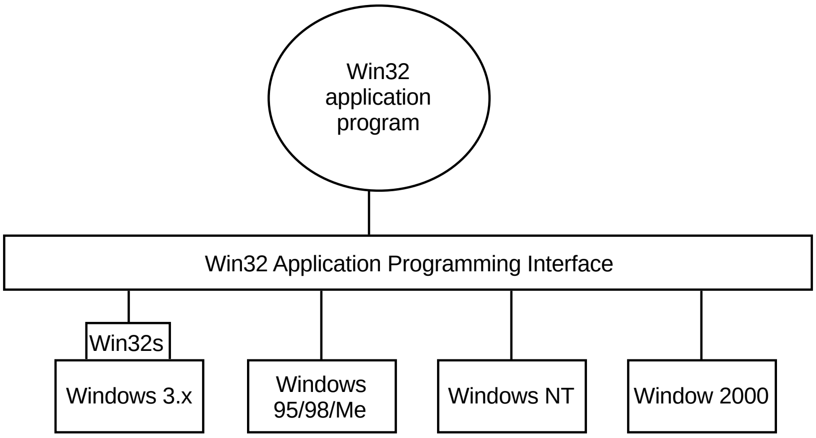
Microarchitecture

Physical devices

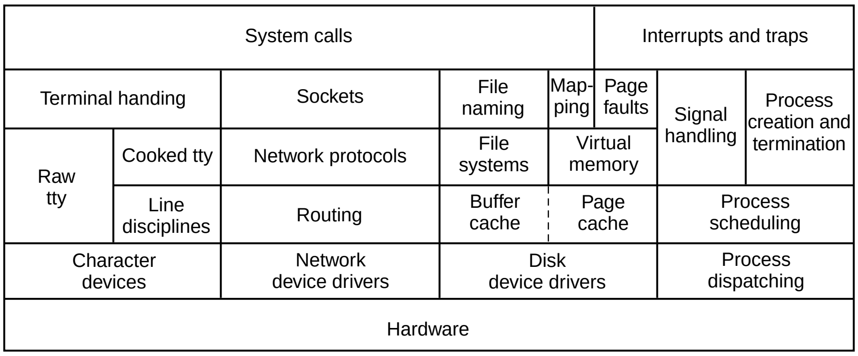
System Layers (Unix)


User interface
Users
Library interface
Standards utility programs
(shell, editors, compliers etc)
System call interface
Standard library
(open, close, read, write, fork, etc)
UNIX operating system
(process management, memory management,
the file system, I/O, etc)
Hardware
(CPU, memory, disks, terminals, etc)
User mode
Kernel mode

Role of API (Windows)


Win32 application program
Win32 Application Programming Interface
Win32s
Windows 3.x
Windows 95/98/Me
Windows NT
Window 2000

Unix-like systems



System calls

Interrupts and traps

Terminal handing

Sockets

File naming

Map-ping

Page faults

Signal handling

Process creation and termination

Raw tty

Cooked tty

Network protocols

File systems

Virtual memory

Line disciplines

Routing

Buffer cache

Page cache

Process scheduling

Character devices

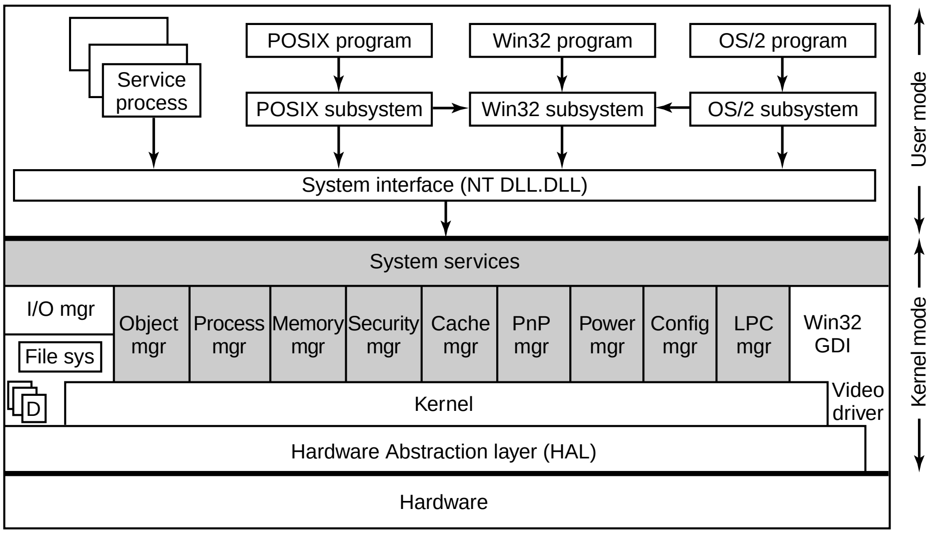
Network device drivers

Disk device drivers

Process dispatching

Hardware

Windows NT/2000/XP/ . . .



Service process

POSIX program

Win32 program

OS/2 program

POSIX subsystem

Win32 subsystem

OS/2 subsystem

System interface (NT DLL.DLL)

System services

I/O mgr

File sys

D

Object mgr

Process mgr

Memory mgr

Security mgr

Cache mgr

PnP mgr

Power mgr

Config mgr

LPC mgr

Win32 GDI

Kernel

Video driver

Hardware Abstraction layer (HAL)

Hardware

User mode

Kernel mode

Operating System Topics

- Process management
- Memory management
- Synchronization and communication (IPC)
- I/O
- Deadlocks
- File Systems
- Networking
- Security
- Real-time and multimedia
- Distribution

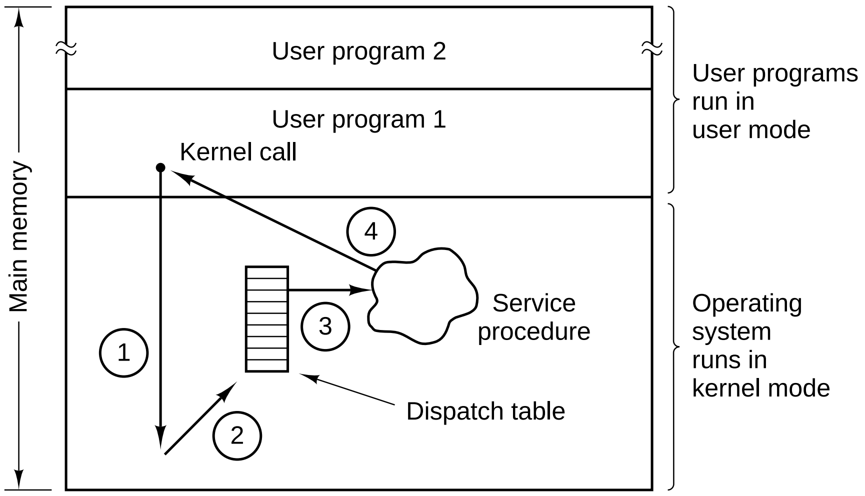
Process Management

- Provide multiple, *concurrent activities*
- Ensure proper *progress* of activities
- Ensure *protection* between unrelated activities

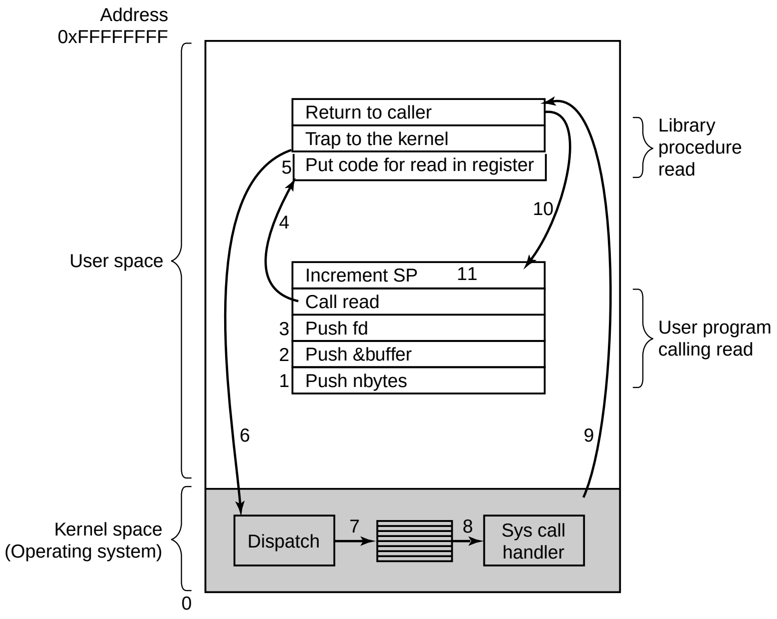
Means

- Resources associated with *program contexts* (processes)
- Activities represented by *execution contexts* (threads)
- *Context switching* between activities
- *Scheduling strategies* for swithcing
- HW support: *Kernel mode* + *memory management unit*

System Call



System Call (Elaborate)



Contexts

Program (Process)

- Administrative unit holding resources:
  - *Memory address space(s)*
  - *Name space(s)*
  - *Access points* (handles) to other resources, especially files
  - One or more *virtual processors*
  - Security, accounting, and other *information*

Execution (Thread state)

- State of a virtual processor:
  - *Registers* (including PC and SP)
  - *Stack*
  - Scheduling *information* (thread state, priority, . . .)
  - Associated program context

Process Creation

Unix

- `fork()`
  Makes an identical copy of calling process except for:
  - Pending signals
  - Return value
- `execve(`*file*`,`*args*`,`*env*`)`
  Loads a new program into current process and starts it

Windows

- `CreateProcess(`*file*`,`*args*`, ...)` (10 parameters)
  Loads a program into a new process and starts it

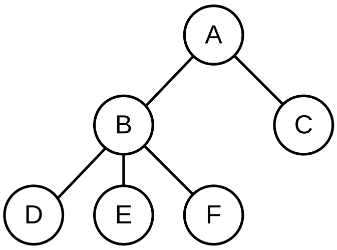
A Unix Shell

```
while (TRUE) {                              /* repeat forever /*/
    type_prompt( );                         /* display prompt on the screen */
    read_command(command, params);          /* read input line from keyboard */

    pid = fork( );                          /* fork off a child process */
    if (pid < 0) {
        printf("Unable to fork0);           /* error condition */
        continue;                           /* repeat the loop */
    }

    if (pid != 0) {
        waitpid (–1, &status, 0);           /* parent waits for child */
    } else {
        execve(command, params, 0);         /* child does the work */
    }
}
```

Process Relationsships

Unix

- Creator becomes *parent* — only parents can await termination



Windows

- All peers — may pass references