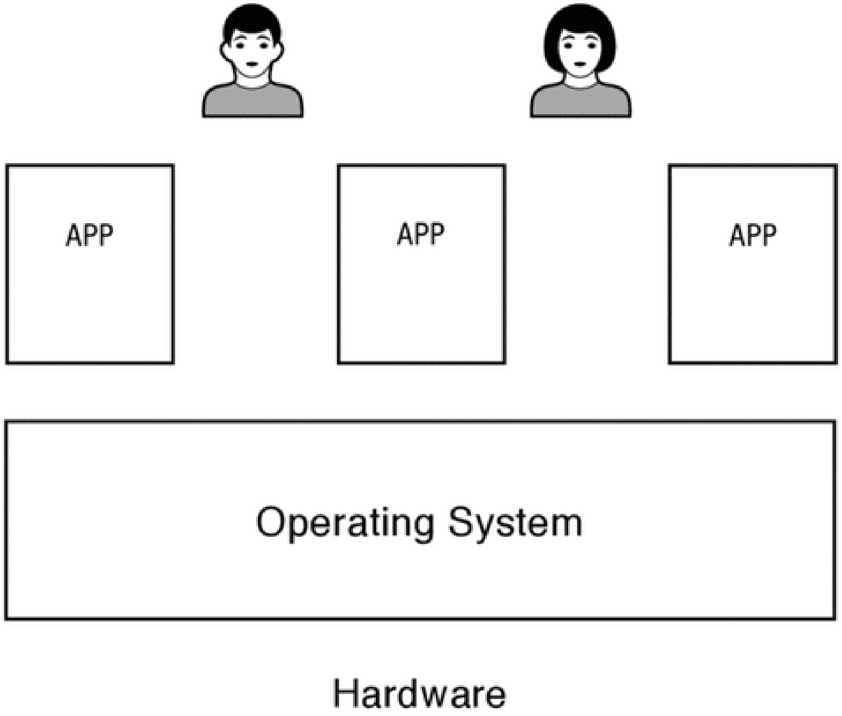
Operating systems Architecture

Operating Systems

- Low level software system that
  - manages all applications
  - implements an interface between applications and resources
  - manages available resources
- Resource manager
- Interface
- Virtual Machine

Operating System



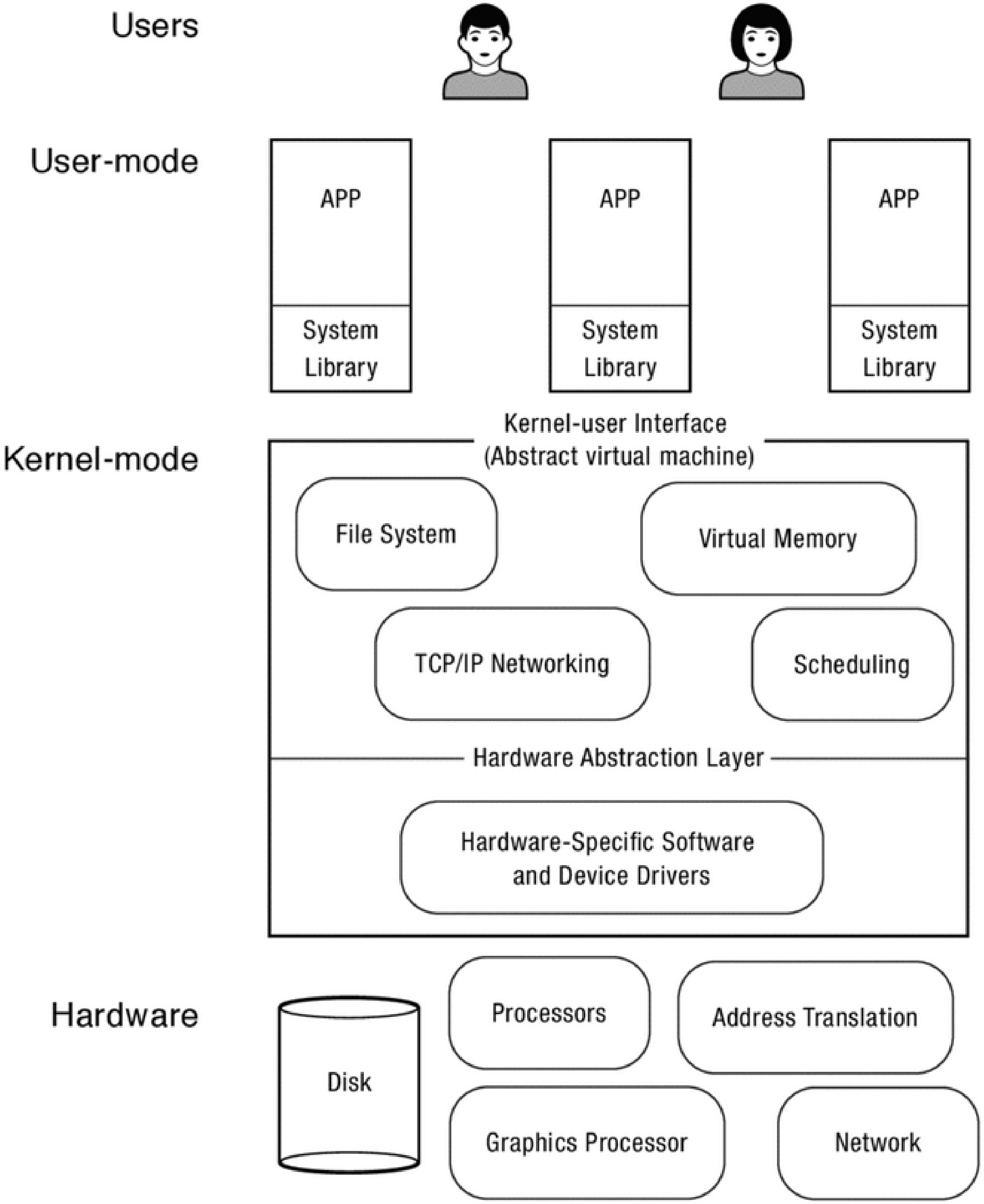
APP

APP

APP

Operating System

Hardware



Users

User-mode

APP

System Library

APP

System Library

APP

System Library

Kernel-mode

Kernel-user Interface (Abstract virtual machine)

File System

Virtual Memory

TCP/IP Networking

Scheduling

Hardware Abstraction Layer

Hardware-Specific Software and Device Drivers

Hardware

Disk

Processors

Address Translation

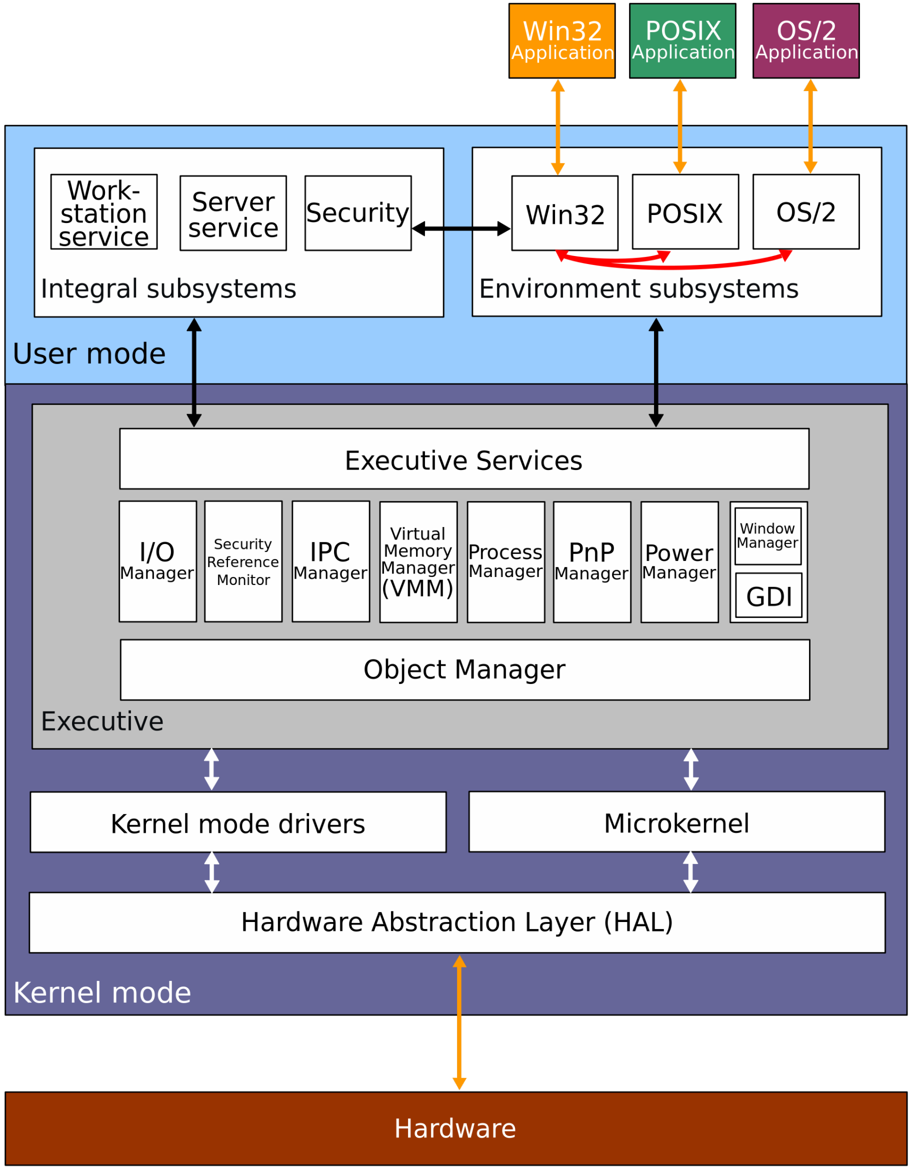
Graphics Processor

Network

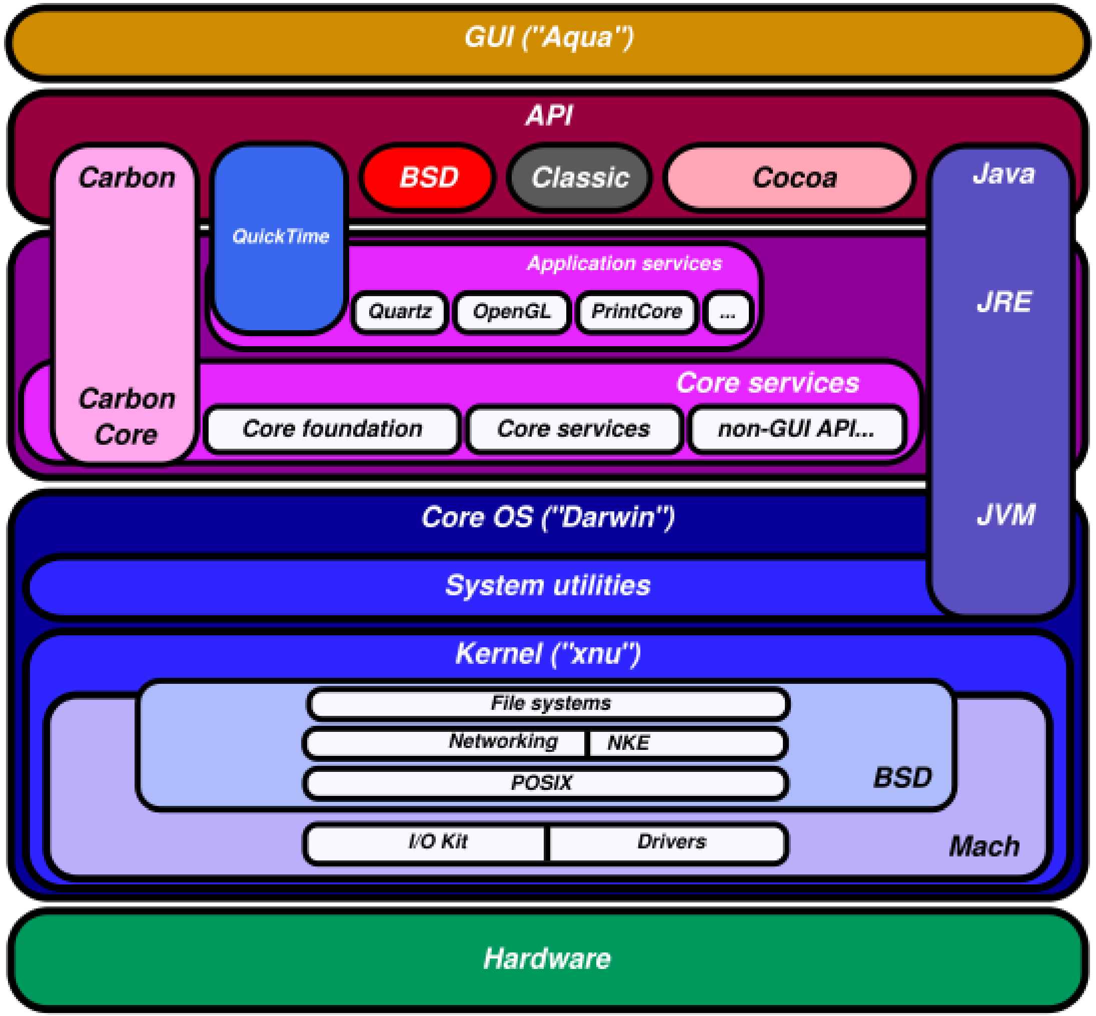
Operating System

- OS kernel
  - Code executed in privileged mode
- User space
  - Code executed in non privileged mode
- Service / daemons
  - Application that executed in the background (server)
- Utility programs
  - Application provided by the OS and executed by the user (editor, shell, compiler()
- System calls
  - functions that implement parts of the OS services or utilities
  - can be used inside the kernel
  - Manage and change internal structure.

Operating System



Win32 Application
POSIX Application
OS/2 Application
Work-station service
Server service
Security
Integral subsystems
Win32
POSIX
OS/2
Environment subsystems
User mode
Executive Services
I/O Manager
Security Reference Monitor
IPC Manager
Virtual Memory Manager (VMM)
Process Manager
PnP Manager
Power Manager
Window Manager
GDI
Object Manager
Executive
Kernel mode drivers
Microkernel
Hardware Abstraction Layer (HAL)
Kernel mode
Hardware



GUI ("Aqua")
API
Carbon
QuickTime
BSD
Classic
Cocoa
Java
JRE
Application services
Quartz
OpenGL
PrintCore
...
Carbon Core
Core services
Core foundation
Core services
non-GUI API...
JVM
Core OS ("Darwin")
System utilities
Kernel ("xnu")
File systems
Networking
NKE
POSIX
BSD
I/O Kit
Drivers
Mach
Hardware

Operating System Roles

- Referee
  - management of resources shared between application
  - Isolation of applications from each other
  - Decide which applications get which resources
- Illusionist
  - Provides abstraction of physical hardware
    - I/O, available memory, ...
- Glue
  - Provide standard common services
    - I/O (disk, network, ...)
    - Service implementation

Operating System requirements

- Reliability
- Availability
- Security
- Portability
- Performance
- Adoption

Operating System requirements

- Reliability
  - Capacity to work correctly
  - Hide HW faults
  - Reduce SW bugs
  - Handle attacks
- Availability
  - Time the system is working
  - Affected by frequency of failures (MTBF) and time to restore system
- Buggy system that crashes frequently but never looses information
  - Reliable but not available
- Subverted system that appears to be working correctly
  - Avalibale but not reliable

Operating System requirements

- Security
  - Guarantee that operations are not compromised by malicious attacks
  - Privacy
    - guarantee that data is only accessible to authorized users
- SW has bugs that can be exployted
- Administrator can be untrustworthy
- SW developer can be untrustworthy
- OS Design should minimize vulnerabilities
  - Computer virus
  - Downloaded code

Operating System requirements

- Portability
  - Applications are provided an abstraction for the HW
    - Resources
    - Access
- Portable abstractions do not change with time
- Abstract virtual machines
  - WIN32
  - POSIX
-

Operating System requirements

- Performance
  - Efficient use of of available resources
- Overhead
  - added cost of implementing an abstraction
  - Efficiency
    - lack of overhead
- Fairness
  - how to divide resources (memory, CPU) to multiple applications
- Response Time
  - how long a task takes to run
- Throughput
  - rate at which tasks are completed
- Performance predictability

Operating System requirements

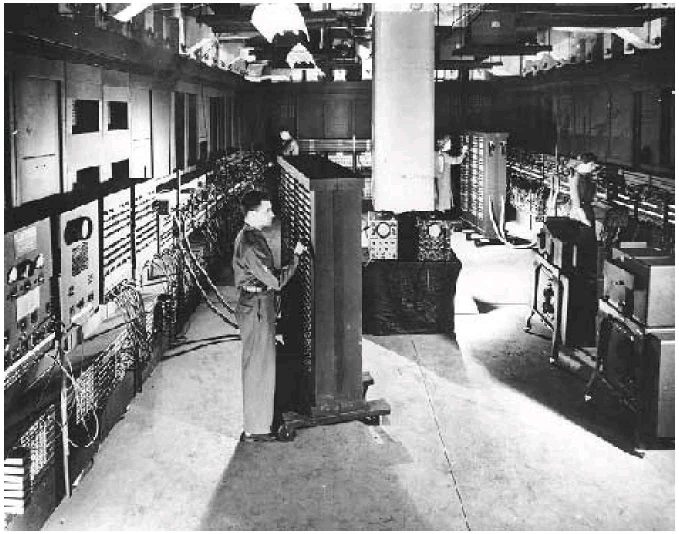
- Adoption rate depend on
  - Availability of Applications
  - Availability of hardware
-
- Open vs closed
- Standardized API
  - more guaranteed of stability
  - more guarantees of portability
- Interoperability

OS architectures

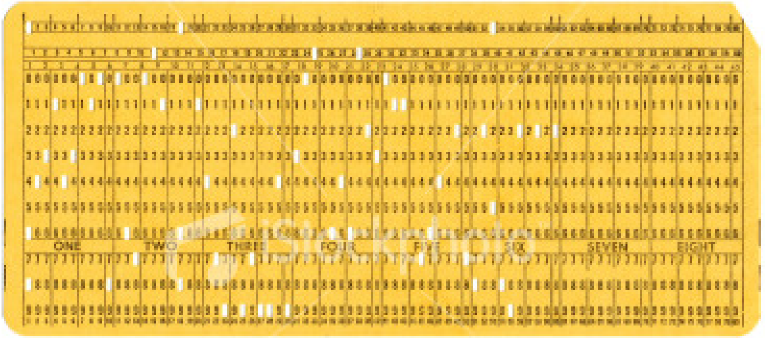
- A general purpose OS is composed of:
  - Process manager
    - multiplexes the CPU time between the multiple execution units (processes)
  - memory manager
    - controls, manages and multiplexes the access to physical and virtual memory
  - Inter-process comunication
    - Implements and handles mechanism for processes to comunicate
  - I/O manager
    - manages comunication with perifheral (keyboard/screen, disk, network)
  - User interface
    - command line interpreter
    - GUI

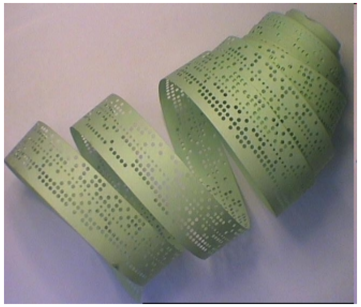
Past, Present Future

- 1945: ENIAC-Electronic Numerical INtegrator And Computer,
  - 19K válves e 1K relays
  - 200 KW
  - 167 m2



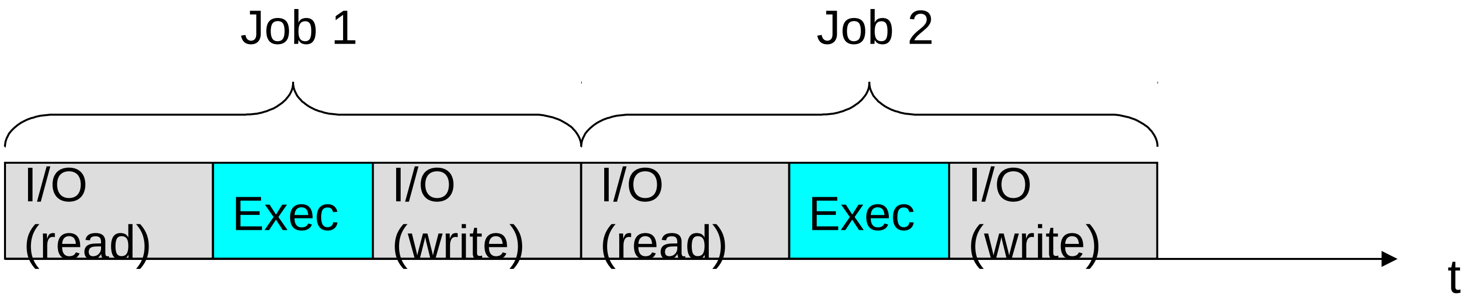
- Numeric computations
  - 5000 sums per second or 357 mult per second or 38 divs per second





Past, Present Future

- Processor idle while reading data
- Batch processing
  - Load
  - Run
  - Unload
- While a job was running
  - OS sets I/O devices for next job



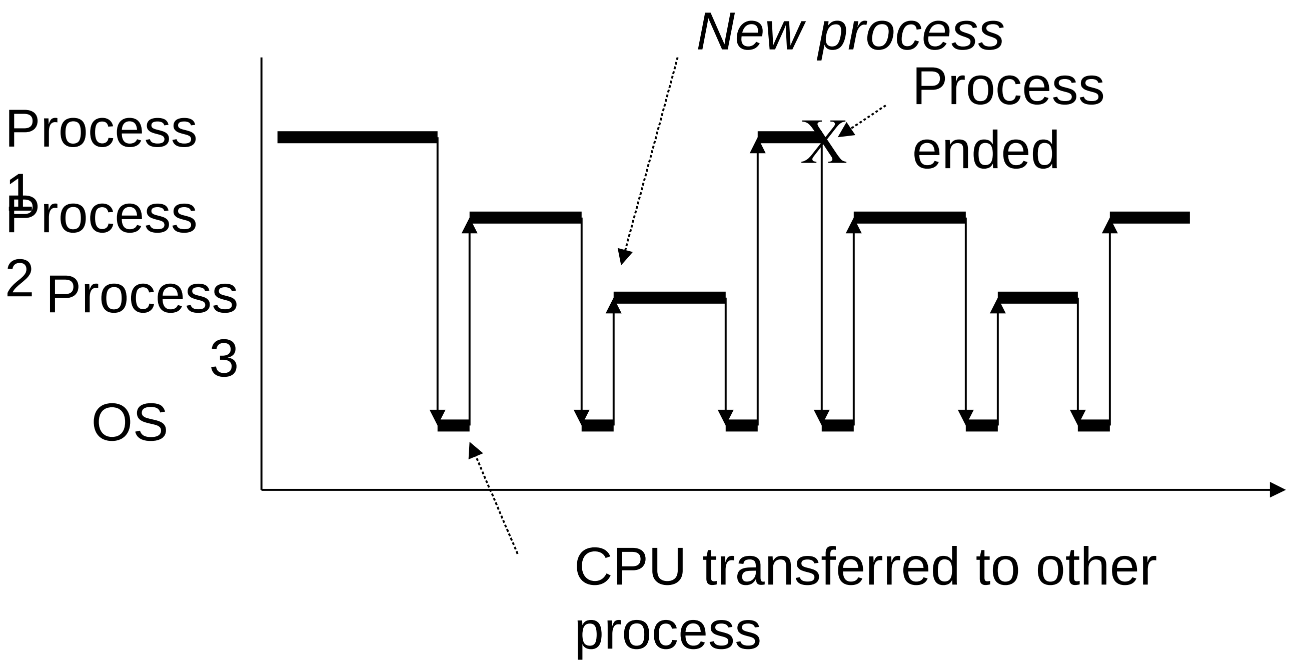
Past, Present Future

- Processor idle while reading data
- Multitasking / multiprocessing
  - multiple programs loaded into memory
  - OS selects program to execute
  - Blocking programs (I/O) do not use CPU
- Efficeint
  - if queu of tasks is long
  - I/O devices feed processor
- requires program isolation

Past, Present Future

- Processor idle while reading data
- User can interact with the programs
  - multiple display/keyboards
  - make program idle most of the time
- Time-sharing
  - multiple programs loaded into memory
  - Programs from different users
  - OS selects program to execute
  - Blocking programs (I/O) do not use CPU
- Efficient
  - if programs require I/O
- Requires program isolation
- Requires user isolation

- CPU Usage



New process

Process ended

Process 1

Process 2

Process 3

OS

X

CPU transferred to other process

- Memory usage

Process 1

Process 2

Process 3

Operating System

Past, Present Future

- Modern OS
  - Desktop
  - Smartphone
  - Server
  - Embedded
  - Virtual machines
  - Server clusters
  - Cloud

- Future OS
  - VLS datacenters
  - VLS Multicore systems
  - Mobile computing
  - Heterogenous systems
  - Large Storage
  - IoT

OS organization

- Layered
- Monolithic
- Micro kernel
- Distributed
- VM based

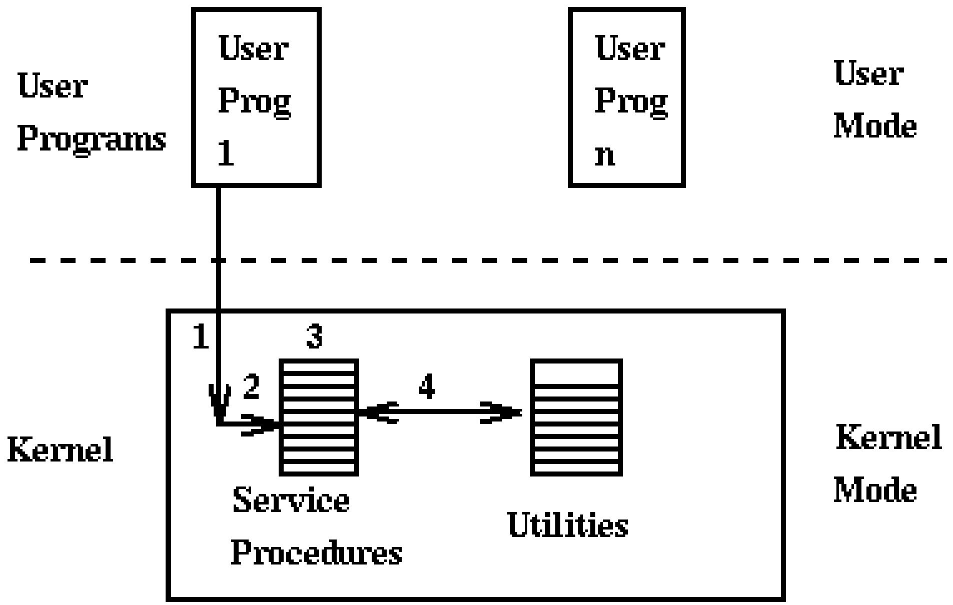
Monolithic architecture

- OS composed of a single module
  - Although using data abstraction
  - Although using layered approach
- All data and code use same memory space
  - low security mechanisms (one driver can mess other drivers)
  - Difficult to evolve (reboot of system needed)
- Easy to implement
- Low overhead

Monolithic architecture

- DOS
- First Unix versions

MONOLITHIC ARCHITECTURE

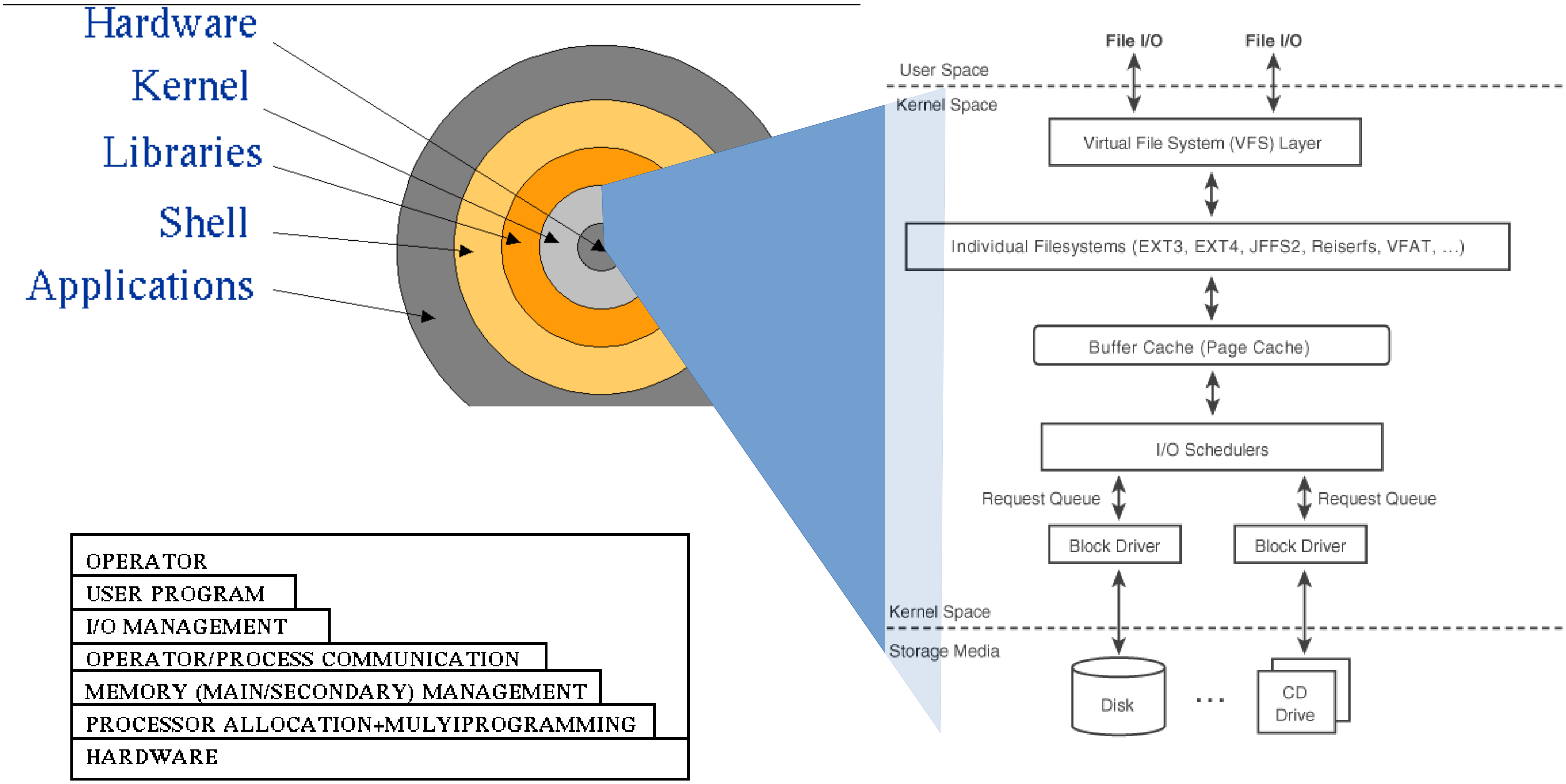


1. System call (User->Kernel Mode)
2. Check parameters
3. Call service routine
4. Service Routine call utilities

Reschedule/Return to user

Layered OS

- Components are divided into layers
  - grouping similar components
- Each layer only interacts with:
  - the bottom layer - requesting services
  - to top layer - answering requests
- Higher level layer
  - Applications
- lowest level layer
  - hardware
- Advantaged
  - good structure, well defined interface, ...
- Disadvantages
  - can be slow, may be difficult to define layers.



OPERATOR
USER PROGRAM
I/O MANAGEMENT
OPERATOR/PROCESS COMMUNICATION
MEMORY (MAIN/SECONDARY) MANAGEMENT
PROCESSOR ALLOCATION+MULYIPROGRAMMING
HARDWARE

LAYERED SYSTEM (THE System, Dijkstra)

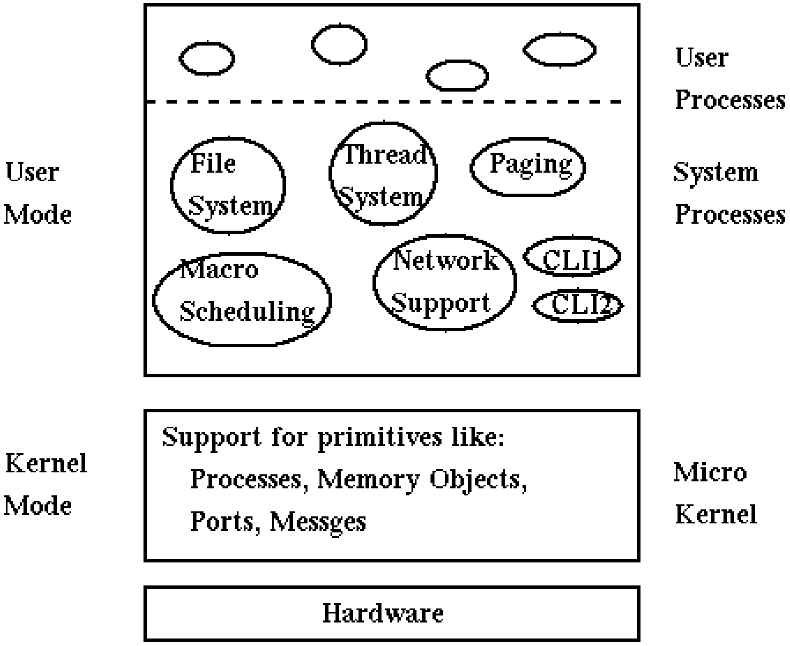
Microkernel

- Removes from kernel as much functionality as possible,
  - limiting the amount of code executed in privileged mode
  - allow easy modifications and extensions
- Most microkernels provide
  - process management
  - memory management
  - message passing between other services
- Security and protection can be enhanced
  - most services are performed in user mode, not kernel mode.
- System expansion can also be easier,
  - only involves adding more system applications, not rebuilding a new kernel.

Microkernel

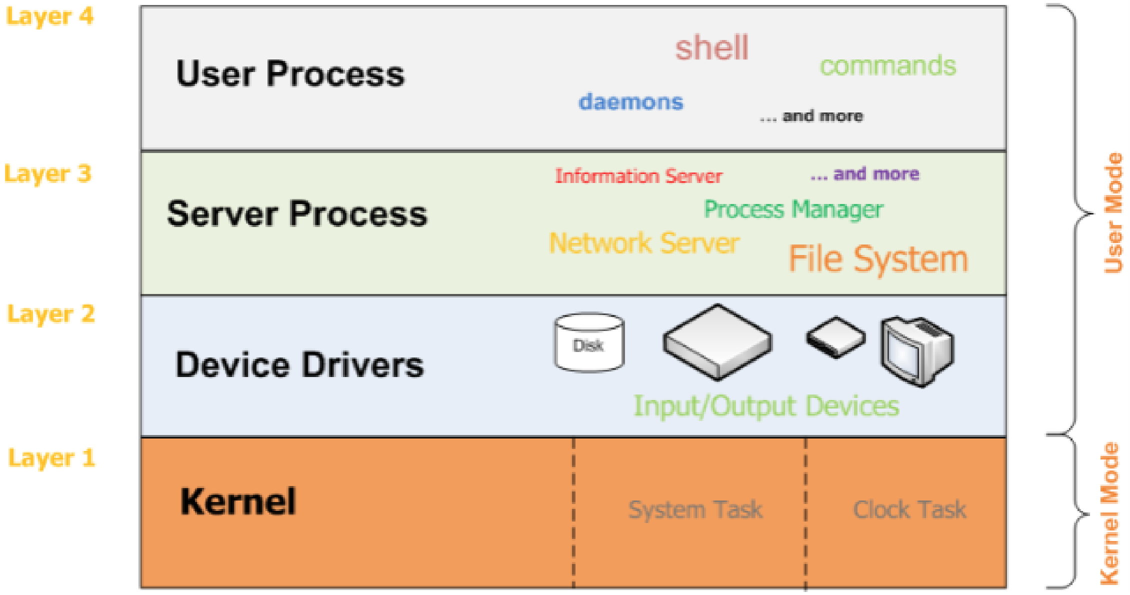
- Windows NT was originally microkernel
  - but suffered from performance problems relative to Windows 95.
  - NT 4.0 improved performance by moving more services into the kernel
- Multiple OSs can be buils on top of a micro-kernel
  - Each operating system will make use of different system processes.

Microkernel

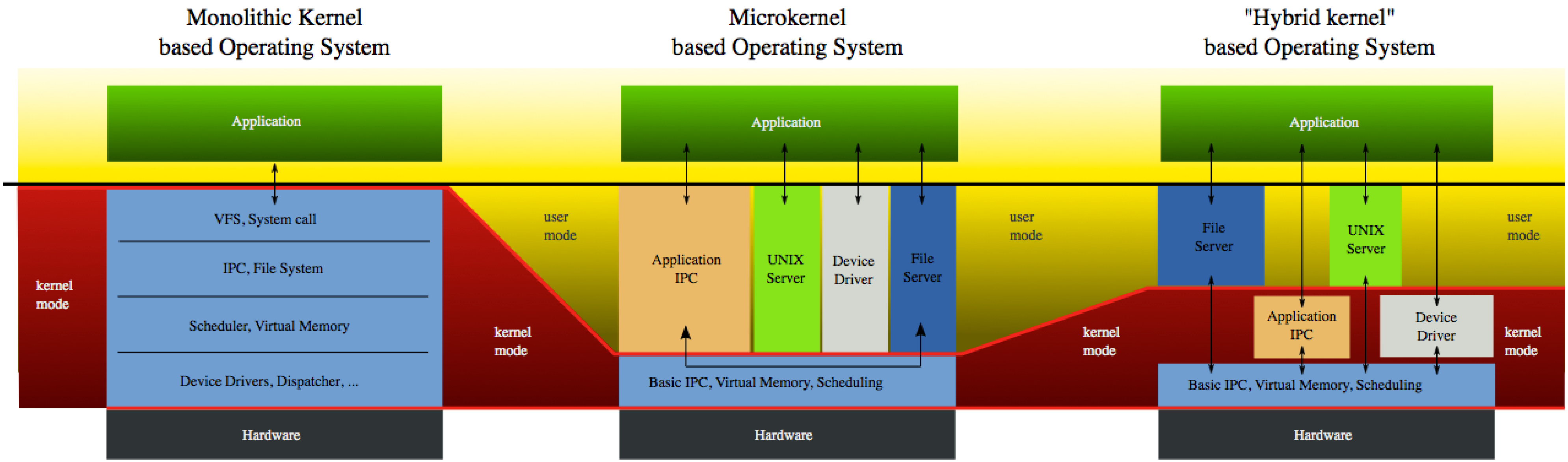


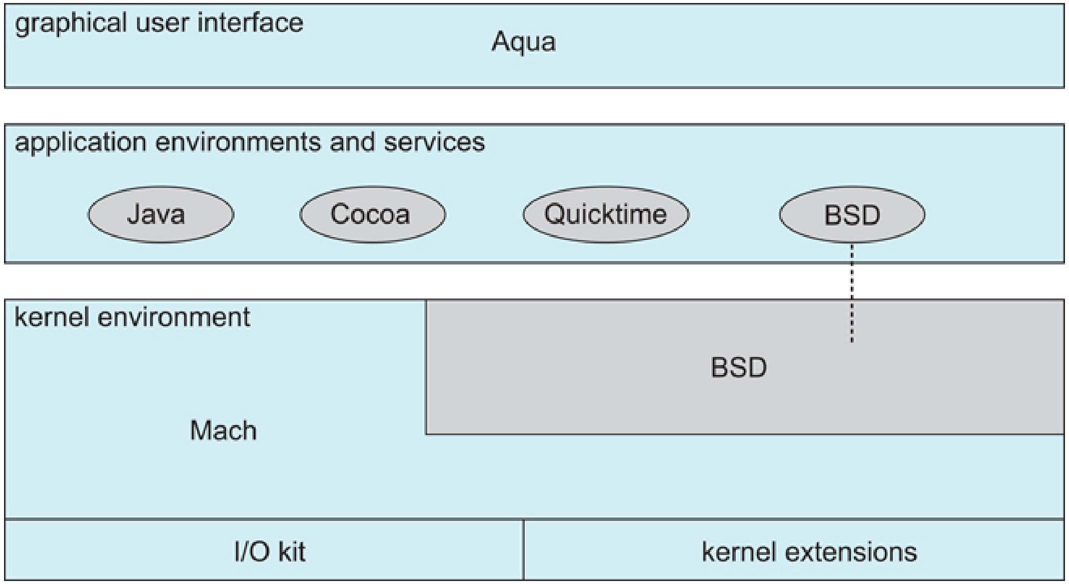
Micro–Kernel Architecture

Minix Layered Micro Kernel Architecture



Hibrid kernel



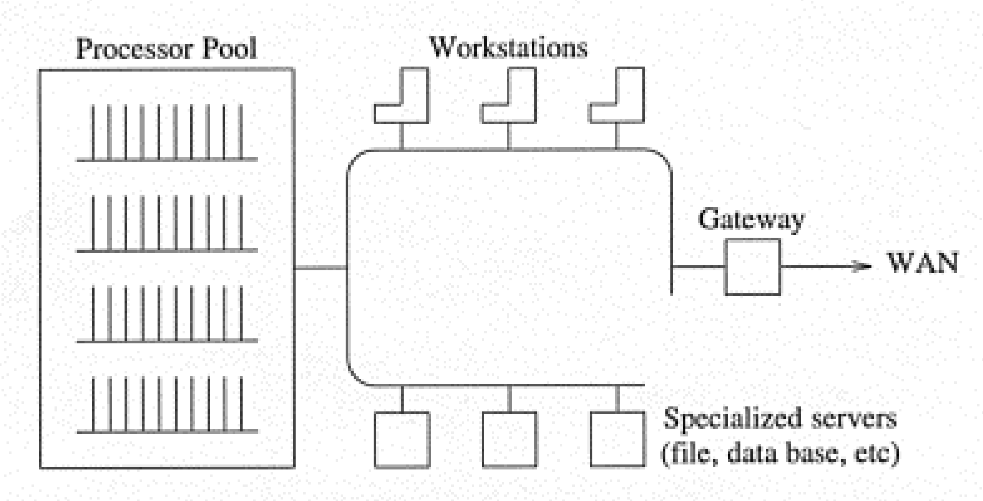


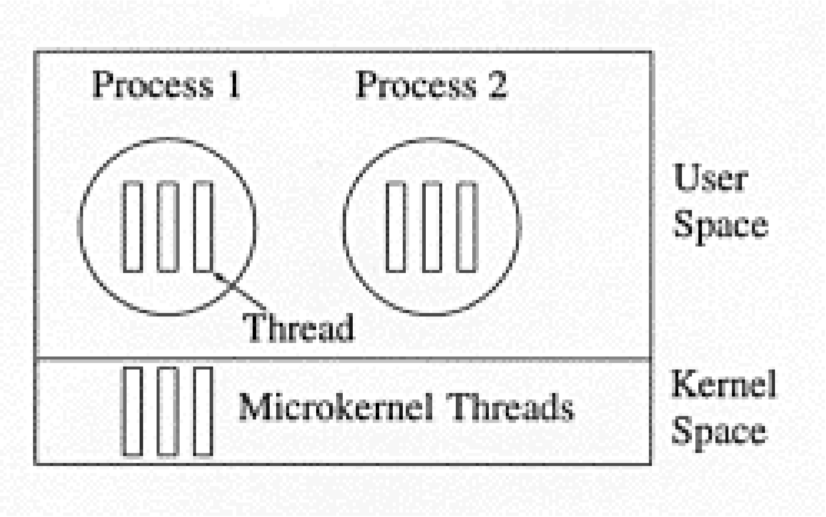
Distributed OS

- Each component/service is a separated process
  - on the same machine
  - on different machines
- Components interactions
  - Messages / Remote procedures
- Distributed File System
- Distributed Memory
- Distributed Processes

Amoeba

- The Bullet Server
  - Used for file storage
- The Directory Server
  - Used for file naming
  - Maps from names to capabilities
- The Replication Server
  - Used for fault tolerence and performance
- The Run Server
  - Run server manages the processor pools
- The Boot Server
  - Ensures that servers are up and running
  - If it discovers that a server has crashed,
    - it attempts to restart it, otherwise selects another processor to provide the service


Processor Pool
Workstations
Gateway
WAN
Specialized servers
(file, data base, etc)


Process 1
Process 2
User
Space
Thread
Microkernel Threads
Kernel
Space