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

OS requirements

- Requirements
- Performance
  - the use of available resources should be efficient
- Security
  - users and processes should work independently, although sharing resources
- Ease of use
  - The programming interface should be simple but expressive
- Portable
  - Applications and OS should run on differente systems
    - Linux can be executed in differente HW architectures
    - A C application can execute in diffrenete OS (Linux/Unix/MAC OS X)

OS architectures

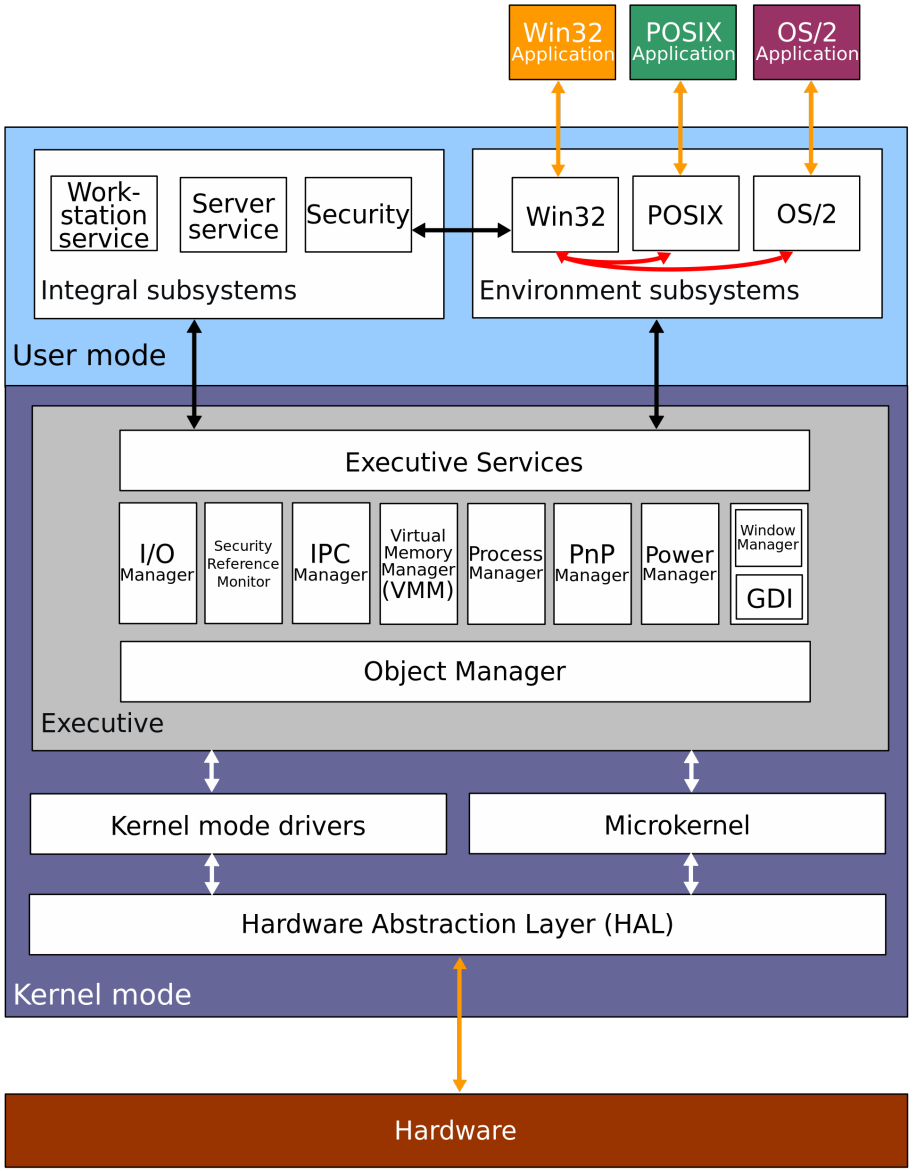
- A general purpose OS is composed of:
  - Process manager
    - multiplexes the CPU time between the multiple execution units (processes)
  - memory manager
    - controls, manages and multiplexes the access to physical and virtual memory
  - Inter-process comunication
    - Implements and handles mechanism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

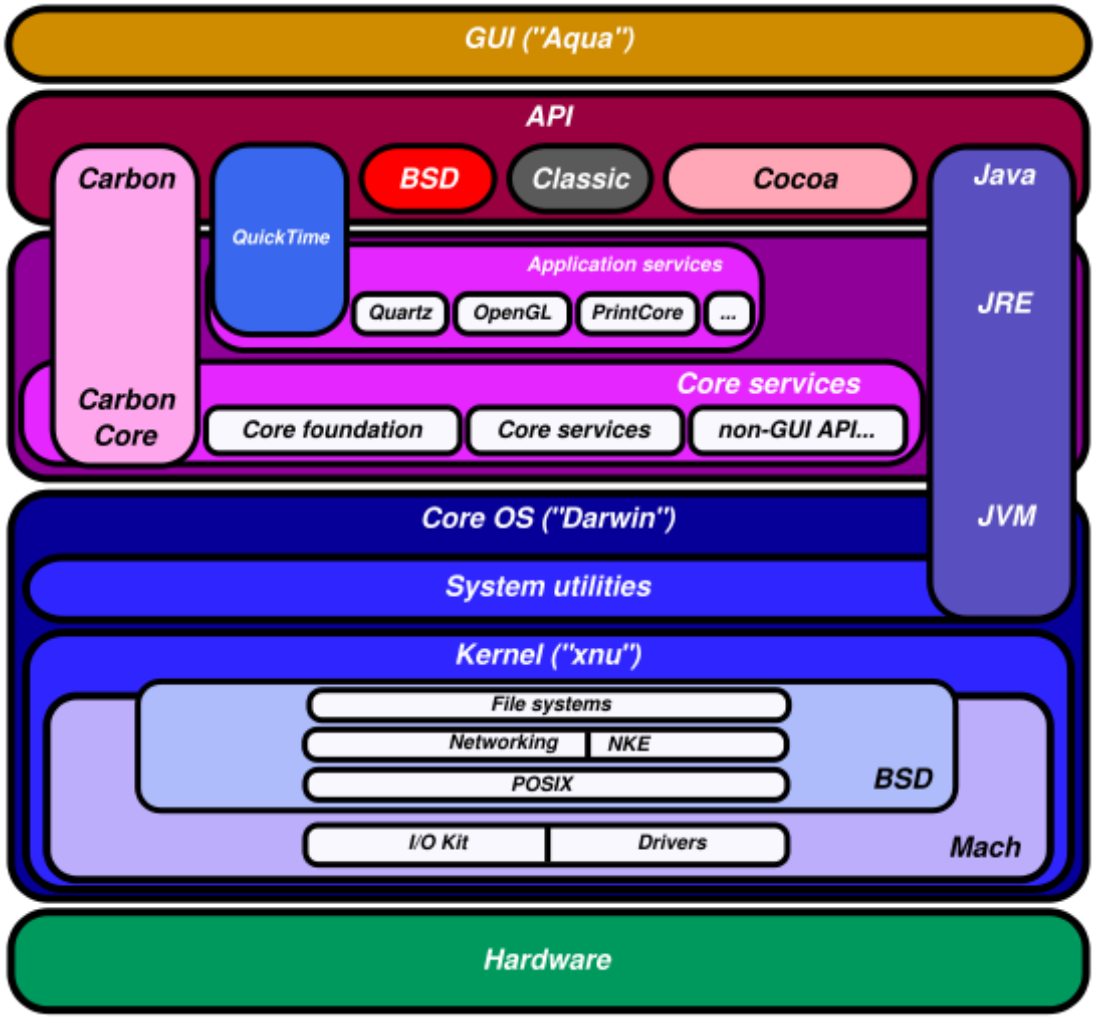
OS architectures

- A general purpose OS is composed of:
  - File System manager
    - manages and organizes data available on disks (file systems)
  - Function calls
    - Programming functions that allow applications to use OS services (memory, disk, I/O)

OS architectures

- A general purpose OS can be divided in:
  - OS kernel
    - Code executed in privileged mode
  - User space
    - Code executed in non privileged mode
  - Service /daemon
    - 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.
  - C lib
    - set of user leve functions


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

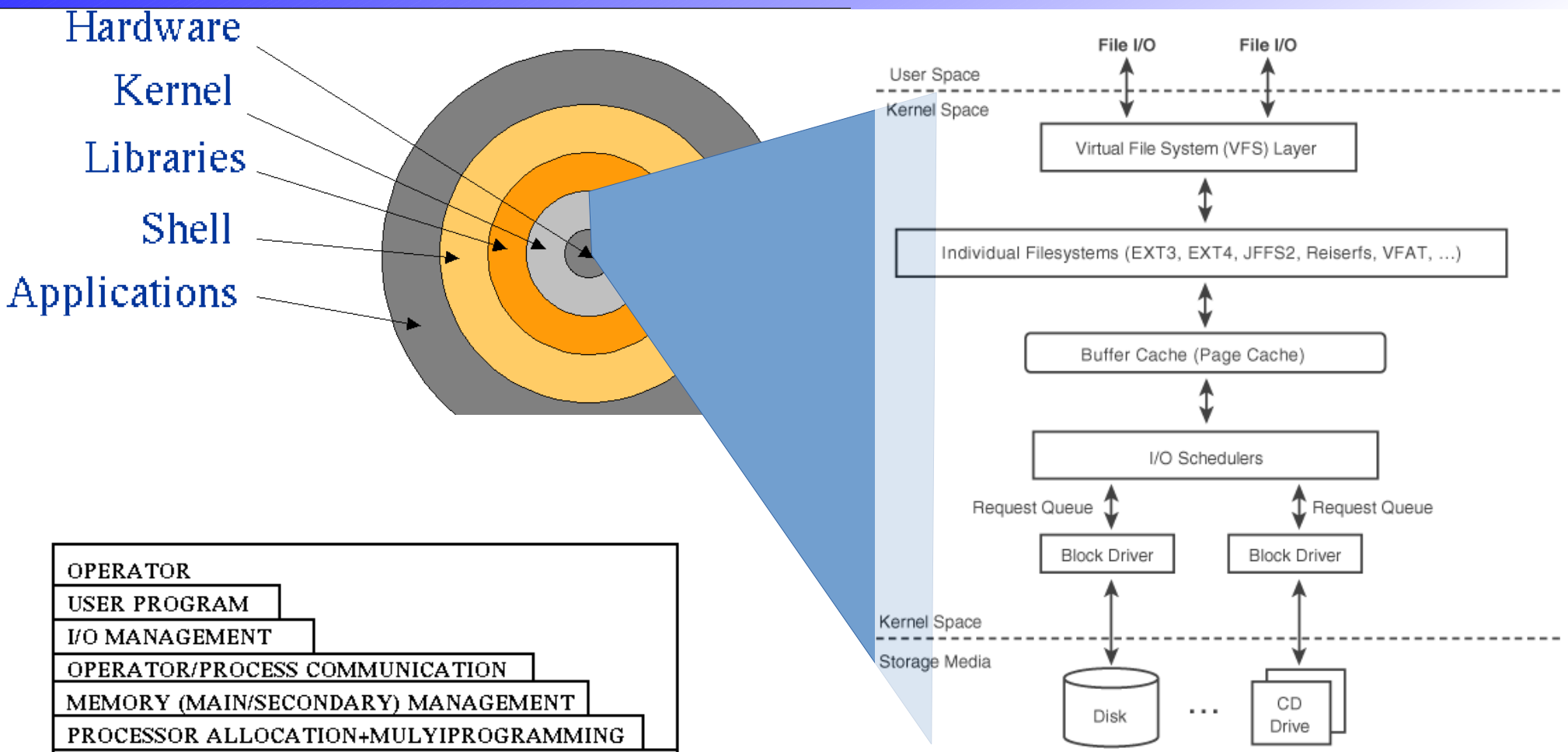

GUI ("Aqua")
API
Carbon
QuickTime
BSD
Classic
Cocoa
Java
Application services
Quartz
OpenGL
PrintCore
...
JRE
Core services
Carbon Core
Core foundation
Core services
non-GUI API...
Core OS ("Darwin")
JVM
System utilities
Kernel ("xnu")
File systems
Networking
NKE
POSIX
BSD
I/O Kit
Drivers
Mach
Hardware

OS organization

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

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

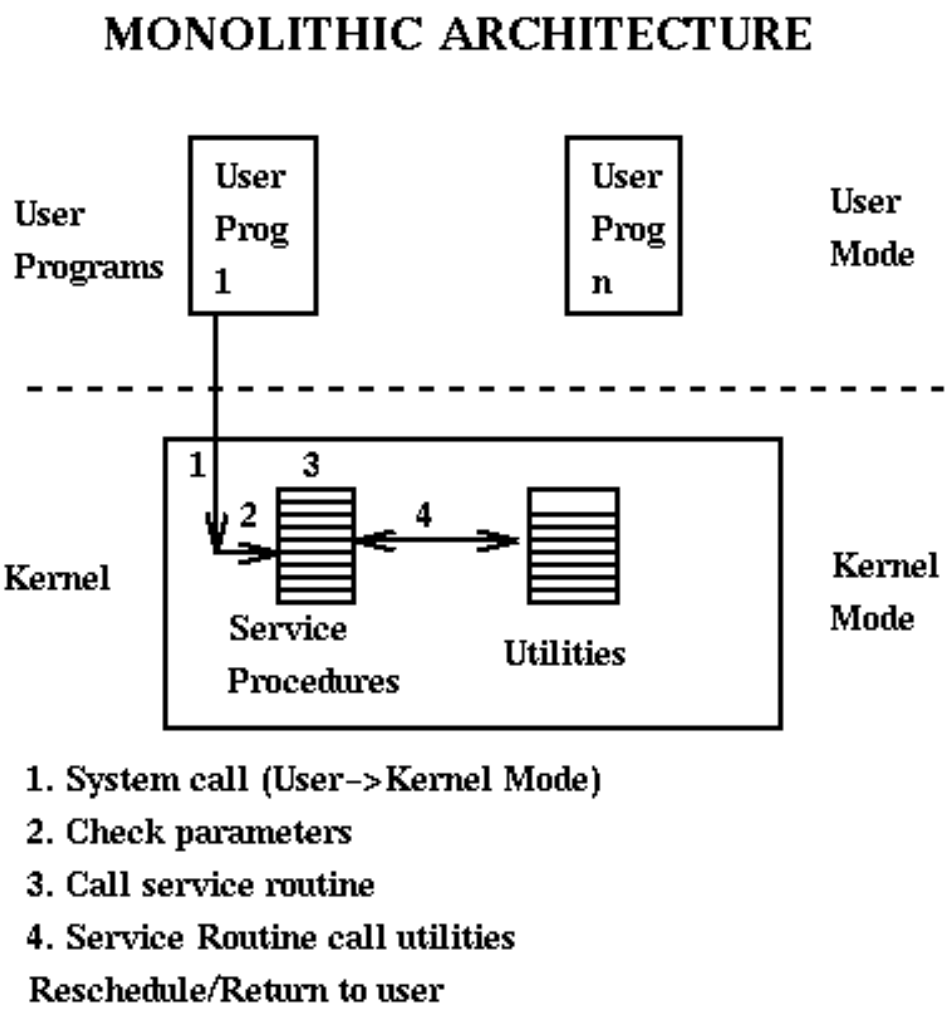
LAYERED SYSTEM (THE System, Dijkstra)

Monolithic architecture

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

Monolithic architecture

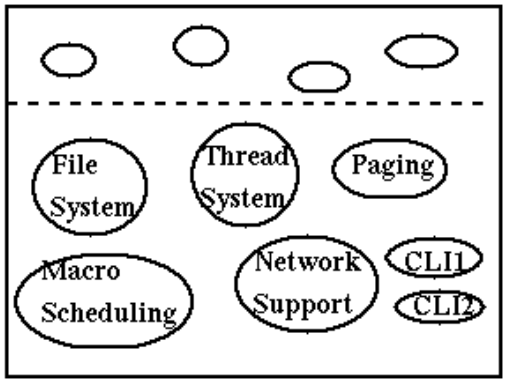
- DOS
- First Unix versions



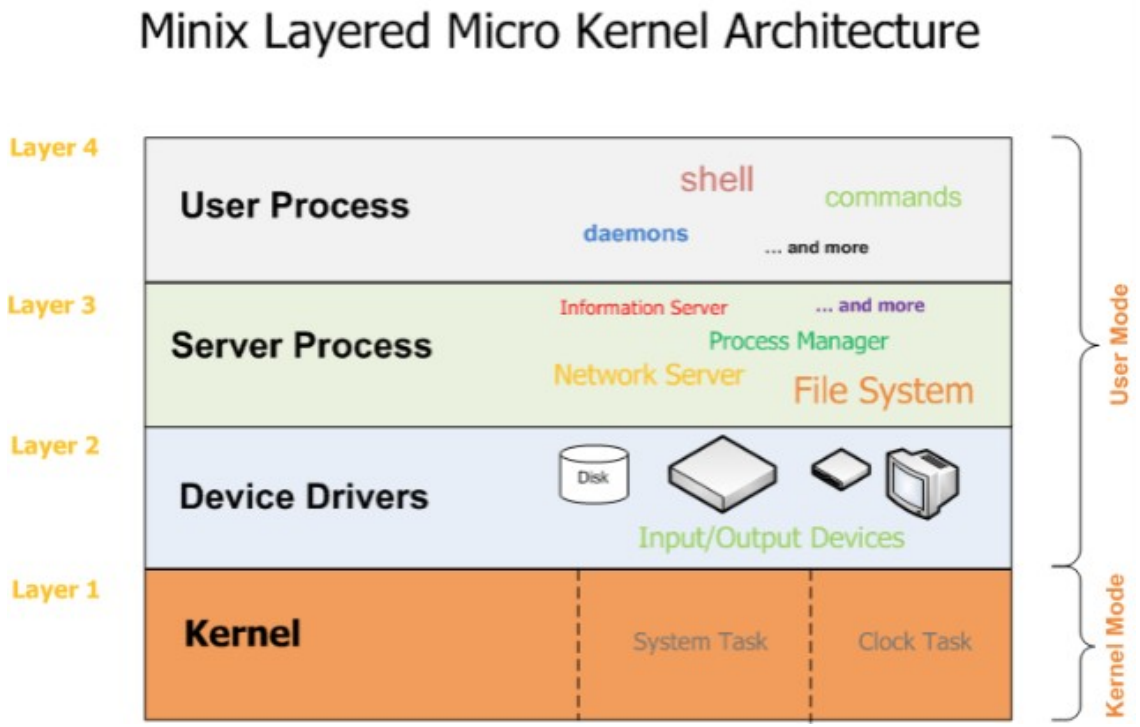
Microkernel

- 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 basic process and memory management, and 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.
- 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 kerne
- Multiple OS can be buil on top of a micro-kernel
  - Each operating system will make use of different system processes.

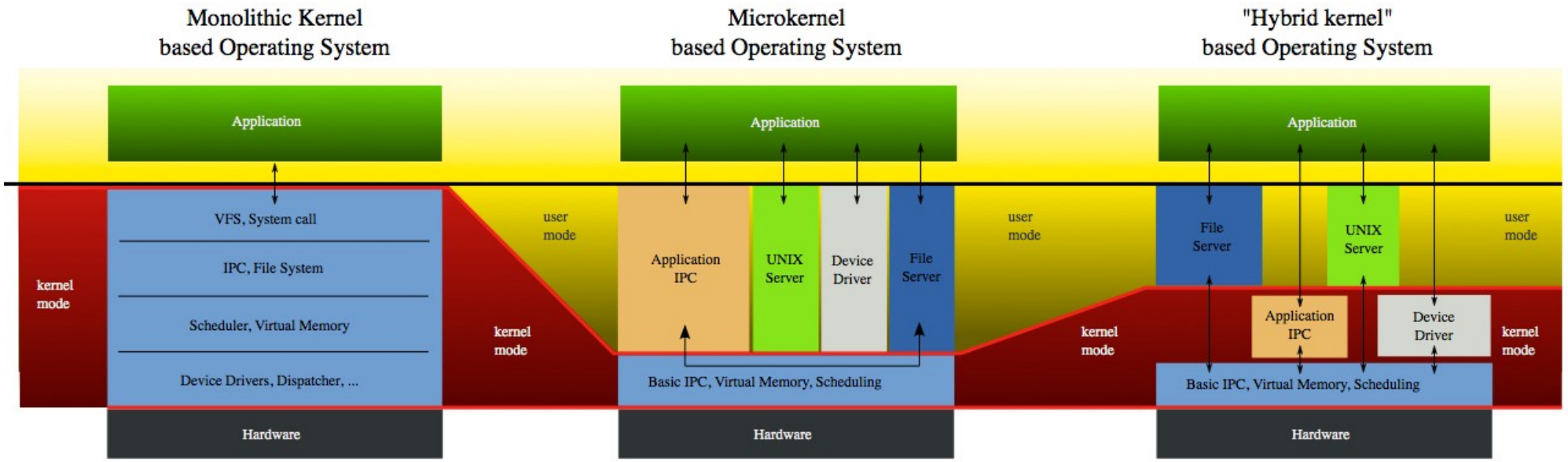
Microkernel

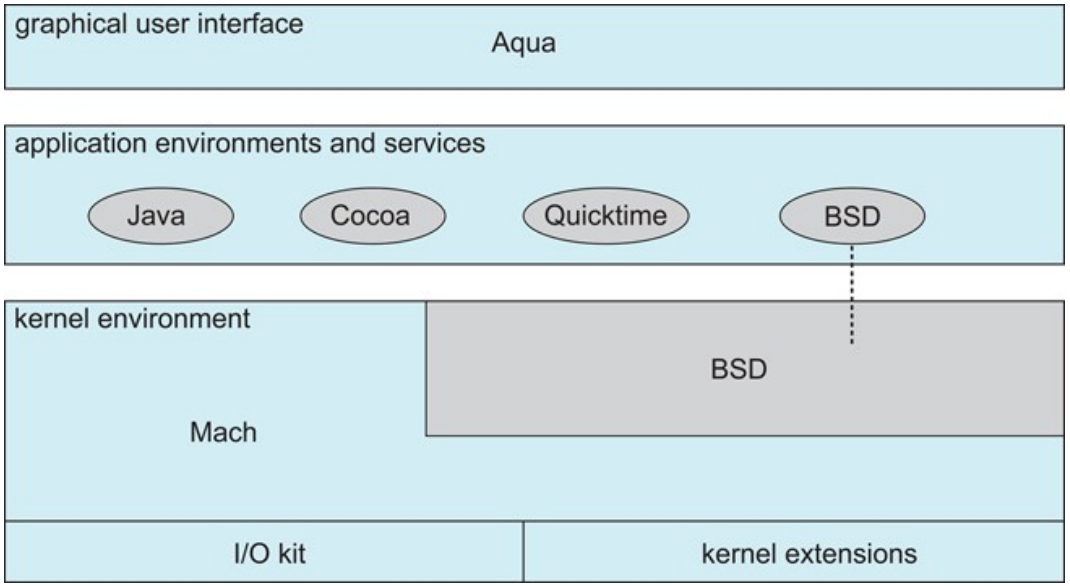


Micro–Kernel Architecture



Hibrid kernel



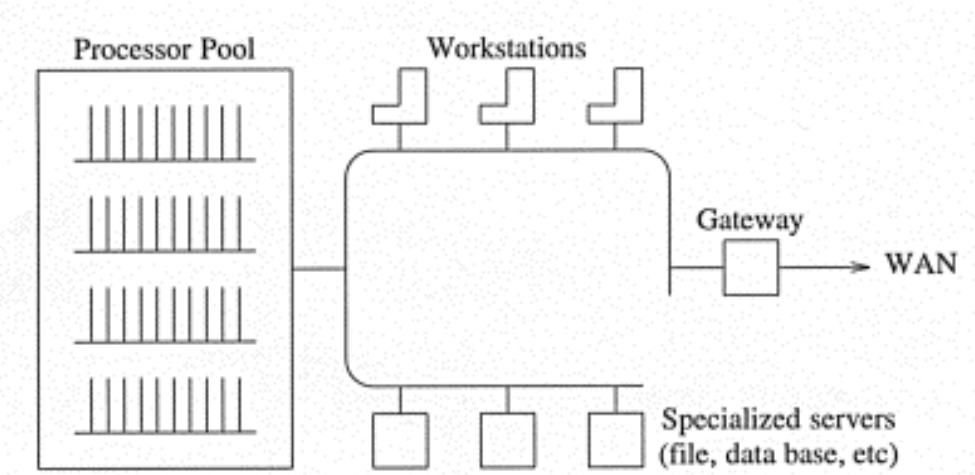


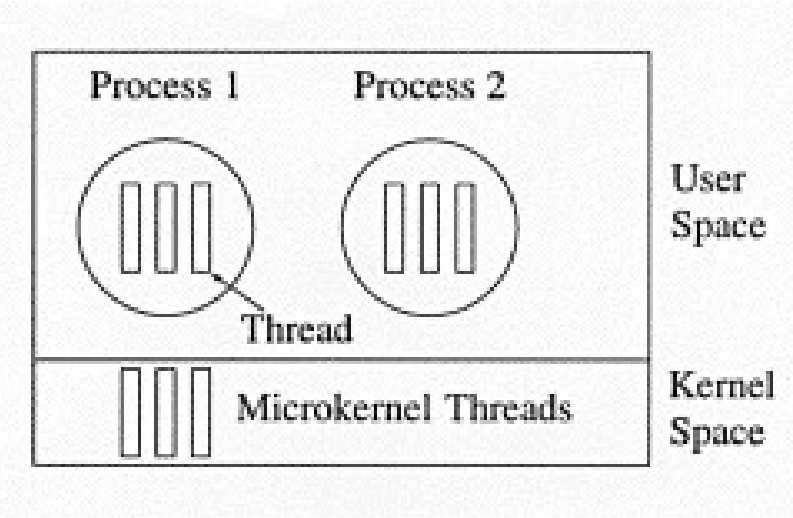
Distributed OS

- Each component/service is a separated process
  - on the same machnie
  - on diffrente machines
- Components initeractions
  - messages / Remote procediures
- 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

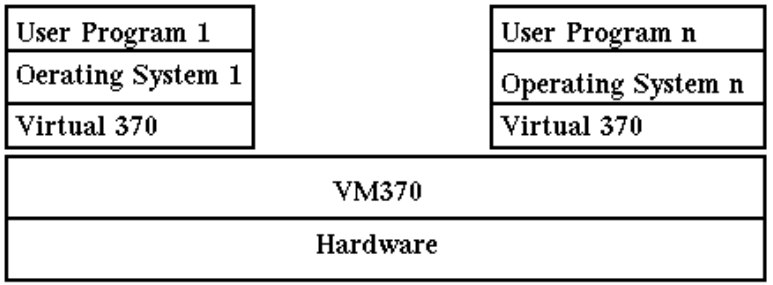




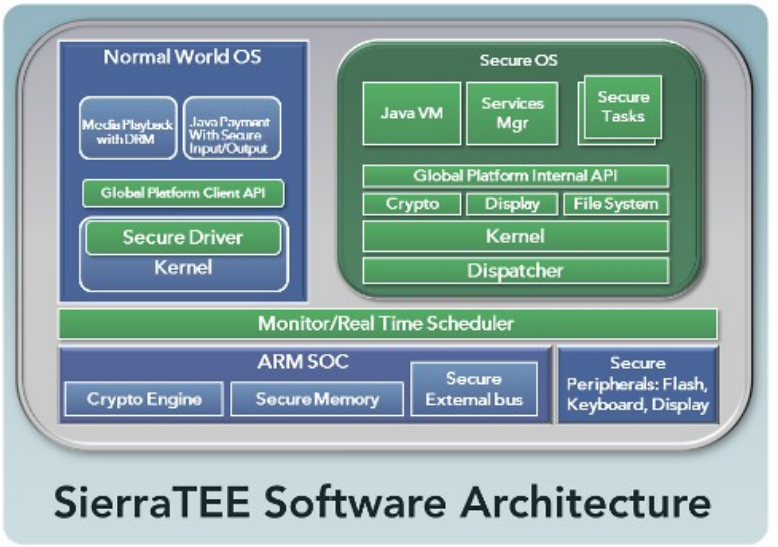
Virtual Machines

- Provide an interface that looks like independent hardware,
  - to multiple different OSes running simultaneously on the same physical hardware.
  - Each OS believes that it has access to and control over its own CPU, RAM, I/O devices, hard drives, etc.
- First appeared as the VM Operating Syste
  - for IBM mainframes in 1972

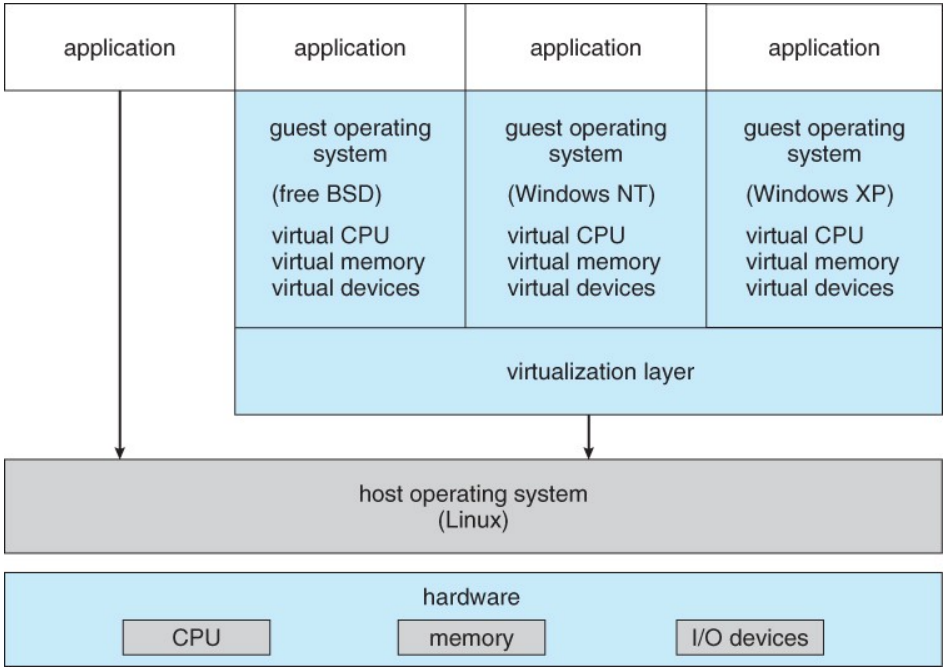
Virtual Machines

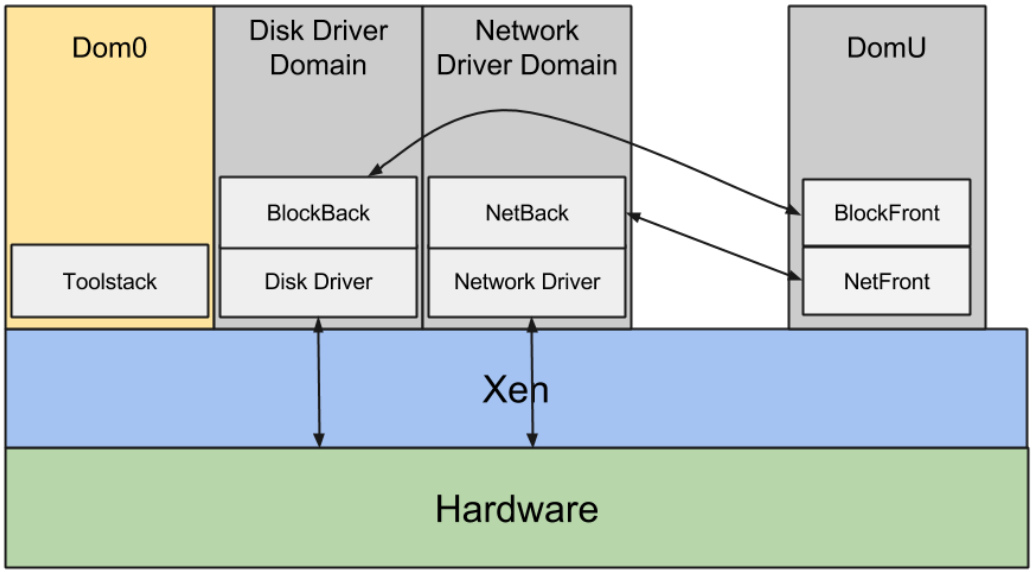


VIRTUAL MACHINE ARCHITECTURE (VM370)



SierraTEE Software Architecture





System Calls

- Application interact with the OS
  - by system calls
- Some systemn call can be invocked by the user from a C program
  - count=read(fd,buffer,nbytes)
- Or from the command line
  - read [-u fd] [-n nbytes] [-a aname] [nome1] …

System Calls

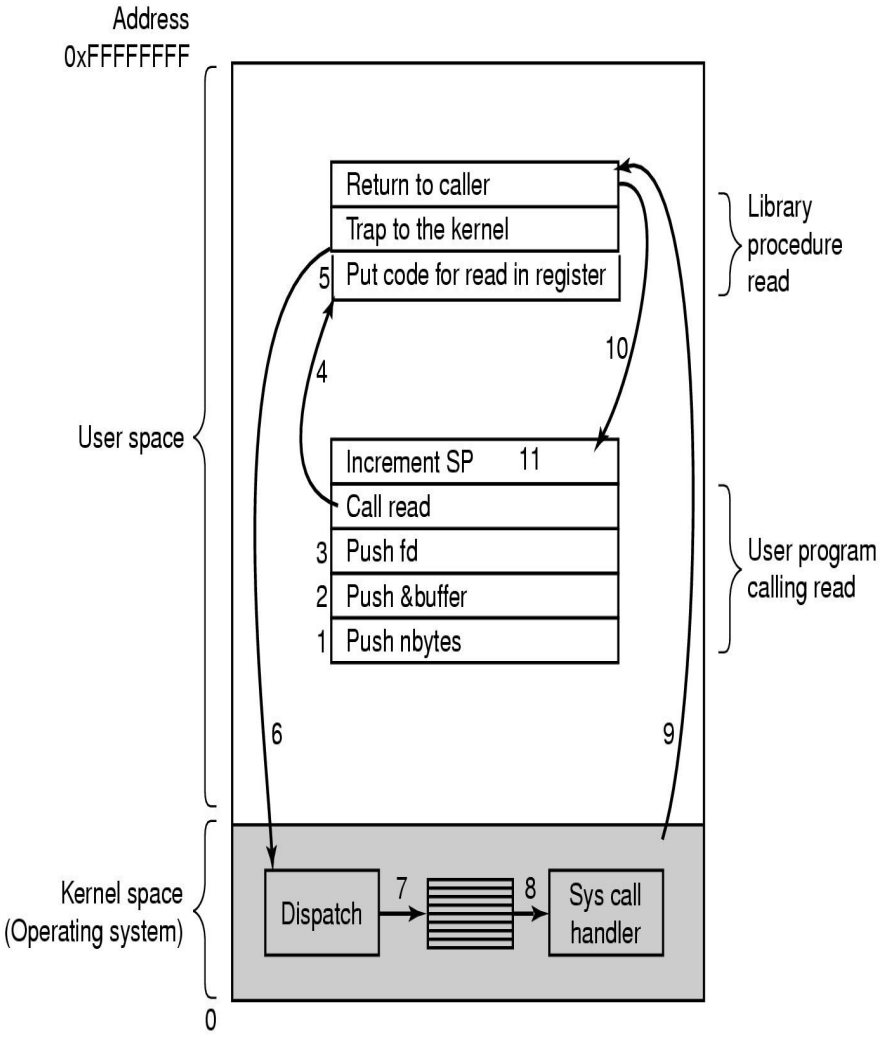
- IN Linux system calls are grouped in groups:
  - Process control: fork, execute, wait,…
  - File management: open, read, set,…
  - Device manegement: request, read, …
  - Infromation maintenance: date, ps , …
  - Comunication: send, …
- First unix version
  - 60 system calls
- Current Linux version
  - more than 300

System Calls

- Microsoft offers the **Windows API** . consists of the following functional categories:
  - Administration and Management
  - Diagnostics
  - Graphics and Multimedia
  - Networking
  - Security
  - **System Services**
  - Windows User Interface
- https://msdn.microsoft.com/en-us/library/aa383723

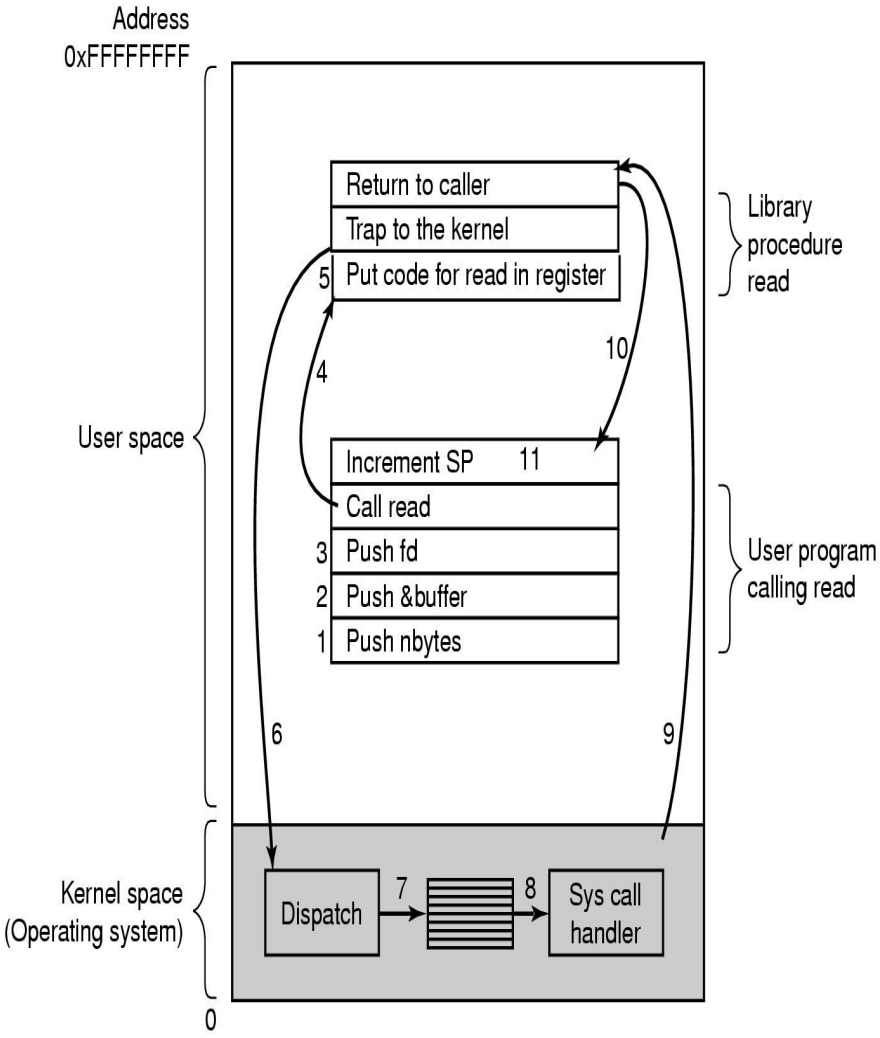
System Calls

- **read** function
- 1-3
  - arguments are lodaded into t stack
- 4
  - function read is called
- {5}
  - EAX register is initialized
    - __NR_read é 3)



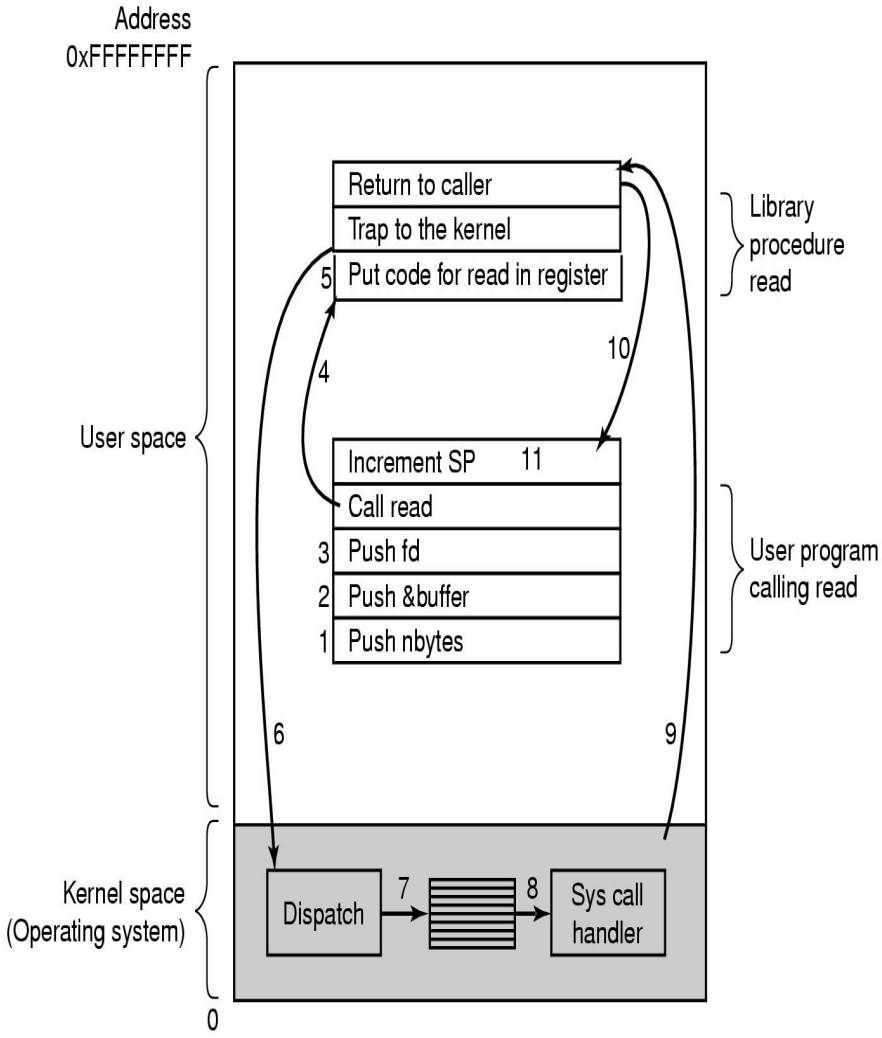
System Calls

- **read** function
- 6
  - The Trap instruction is executed
    - instrução INT 80
  - Context (register) is saved
  - Processor changes to Supervisor moc
  - PC is updated with the correct code location
- 7-8
  - Dispatch executes the correct functior
    - asmlinkage int sys_read(unsigned fd,char *buf,int count)



System Calls

- **read** function
- 9
  - Device maneger vblocks until dat transfer ends
  - After transfer IRET is executed
- 10
  - A value is returned to the C progr



System call vs Library functions

- System calls are provided by the system and are executed in the system kernel.
  - They are entry points into the kernel and are therefore NOT linked into your program.
  - The source code is not portable
  - The API is portable
- Library calls include the ANSI C standard library and are therefore portable.
  - These functions are linked into your program.
- man read
- man fread
- man man
  - section 3 section 2

fread vs read

- fread
  - man fread
  - http://opensource.apple.com/source/Libc/Libc-167/stdio.subproj/fread.c

Manual

- O comando `man` disponibiliza informação sobre o Linux.
- As páginas estão organizadas por 9 secções:
  1. Comandos de utilizador
  2. Chamadas de sistema
  3. Funções de biblioteca do C
  4. Controladores de dispositivos
  5. Ficheiros de configuração
  8. Comandos de manutenção
- O `man` procura a página a partir da secção 1. O utilizador pode indicar a secção entre o comando e o tópico.

  `man 1 read` : página do comando de utilizador `read`

  `man 2 read` : página da chamada de sistema `read`

  `man 3 fread` : página da função de biblioteca `fread`