Recap

What is OS?

Basic hardware organization

Interrupts

Different Architectures.

Operations (Timers / dual mode bit).

What are the services provided by the OS to the applications and system programs?

1. File access services.
2. Error detection and handling
3. Protection and Security. + Permissions.
   a process should not access other processes' memory / data.
4. User Interface
   - CLI Command line Interface
   - GUI Graphical User Interface.
5. Program execution services.
6. I/O access.
7. Allocating resources (e.g memory) to processes.
8. Accounting

How does the OS provide these services?

A system call Syscall provides an interface for these services.

Ex

Copy a File

1. Aequire 2 file names
   - → Ask the user (print/write ... to the CLI)
   - → Read file names.
2. Open Files
   - Syscall.
   - errors can occur.
3. loop reading + writing
   - Syscalls.
4. Close files (syscall)
5. exit Syscall.

Typically an API is used

Application Programming Interface

rather than the native syscall.

API: provides a set of functions (includes parameters and return values) to the programmer to use.

WIN32 API
Java API
POSIX API

Why?

(1) Portability: applications would run on any system that supports the API.

(2) Simple: more simple than the syscall.

Caller needs to know the interface and the result of invoking a syscall (nothing internal)

Types of System calls

1. Process Control (create (func), exit, wait, suspend, ...)
2. File manupilation (read, write, open, close, delete, ...)
3. Device manupilation (aquire, release, ...)
4. Information maintaineree (set/get date/time)
5. Communication (send, receive, socket, ...)
6. Protection (set/get permissions (chmod)).
7. Memory Management (mmap, mlock, munlock, ...)

How to structure the OS?

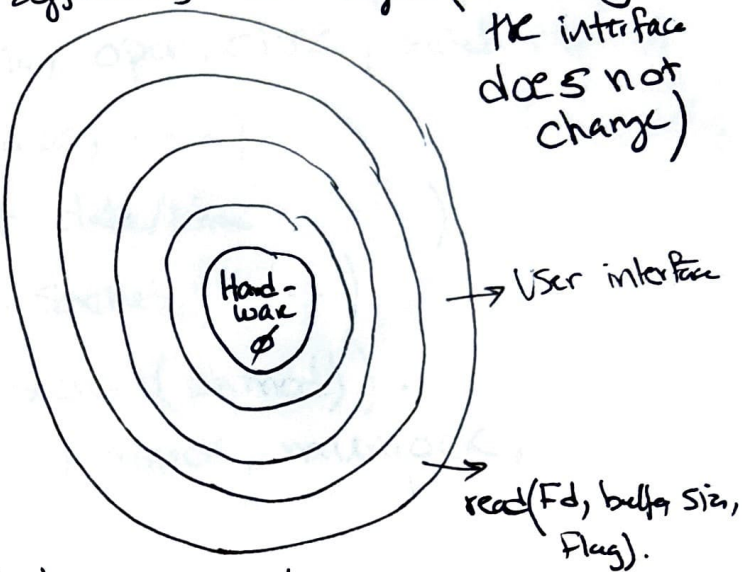
Complex - large - Extinsible Challenges

1. Simple
   started simple but grow over time
   DOS Disk Operating System
   Interface was not well defined.

2 Layered

Modularize system design.

a layer can be changed without affecting other layers (as long as the interface does not change)



Adv

1. Easy to debug.
2. Information hiding (no need to worry about the details).

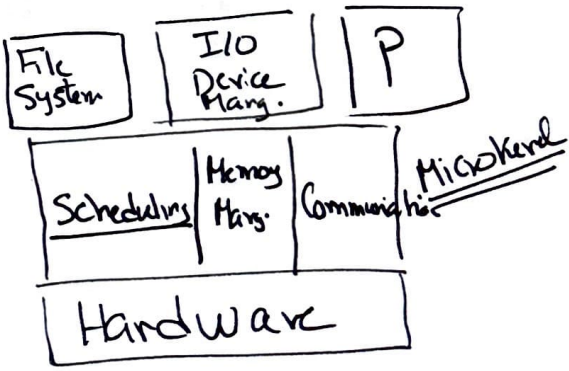
Disadv

1. How to define the number of layers and map functionalities to them?
2. less efficient due to overhead in using multiple interfaces (parameters passed between layers).

3 MicroKernels

Idea: Kernel is very small.

all non-essential components are implemented as system and user programs.



Kernel provides communication between process. (via message exchanges).

Adv

+ Extensible (easy to extend).
+ Easy to debug.
+ a failure of a component does not crash the kernel.

DIS/adv

→ less efficient due to message passing?.

---

(4) Modules

Kernel has a set of core component.
links to additional services that can be loaded at run time / boot)