Operating System Structures

→ Operating System Services-

1.) User interface
2.) Program execution
3.) I/O operations
4.) File System Manipulation
5.) Communications
6.) Error Detection
7.) Resources Allocation
8.) Accounting
9.) Protection and Security

- Certain functions in the operating system services

→ User interface- All operating systems have a user interface varies between Command Line (CLI), Graphics User Interface (GUI)

→ Program execution - System must be able to load a program into memory and to run that program end execution, either normally (or) abnormally.

→ I/O operations - A running program may require I/O which may involve a file (or) an I/O device.

Functions useful to the user-

1) File system manipulation - read and write files and directories create and delete them, search them, list file information, permission management (Ubuntu terminal cd)
2) Communication - communications may be via shared memory or through message passing
3) Error detection - (i) may occur in the CPU and memory hardware
(ii) Appropriate action must be taken

Functions for efficient operation -

1) Resource allocation - When multiple users (or) multiple jobs are running concurrently, resources must be allocated
2) Accounting - To keep track of computer resources
3) Protection and security - Access and defense

→ User operating System Interface

- Command Interpreter allows direct command entry
(i) Sometimes implemented in kernal, sometimes by Systems program
(ii) Sometimes multiple flavours implemented shells
(iii) Primarily fetches a command from user and executes it

→ GUI -

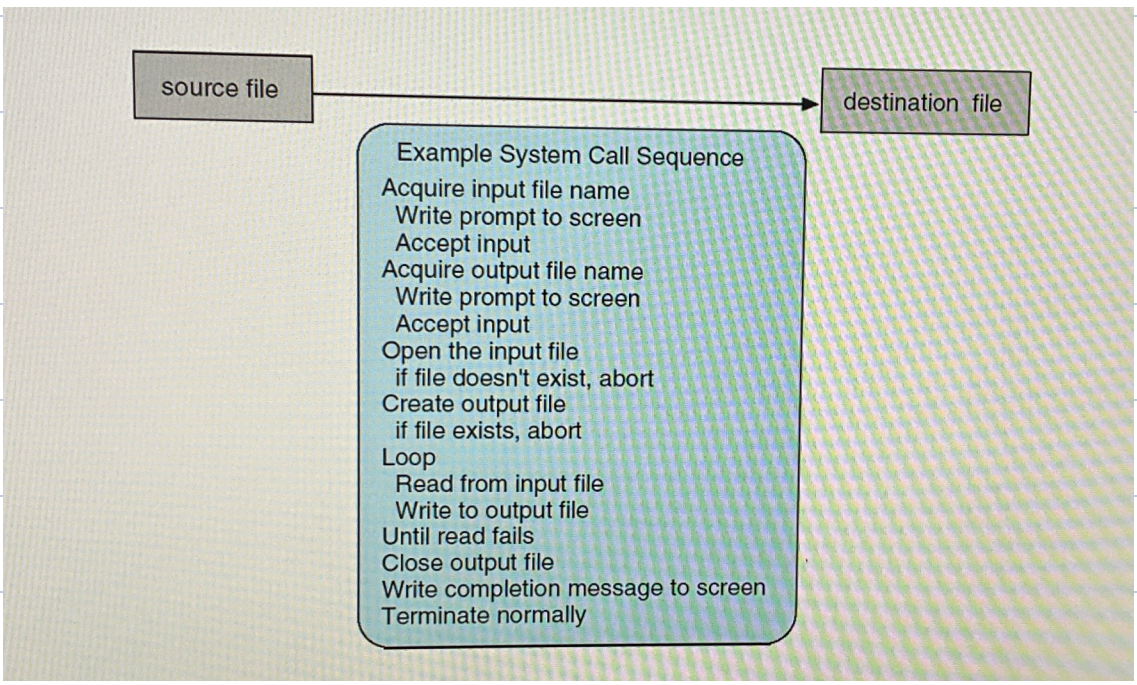
User friendly desktop interface :- (i) Mouse, keyboard and Monitor
(ii) Icons represent files, Programs, actions etc
(iii) to open directories - folder
(iv) Mouse clicking results in various actions

→ Touchscreen Interfaces -

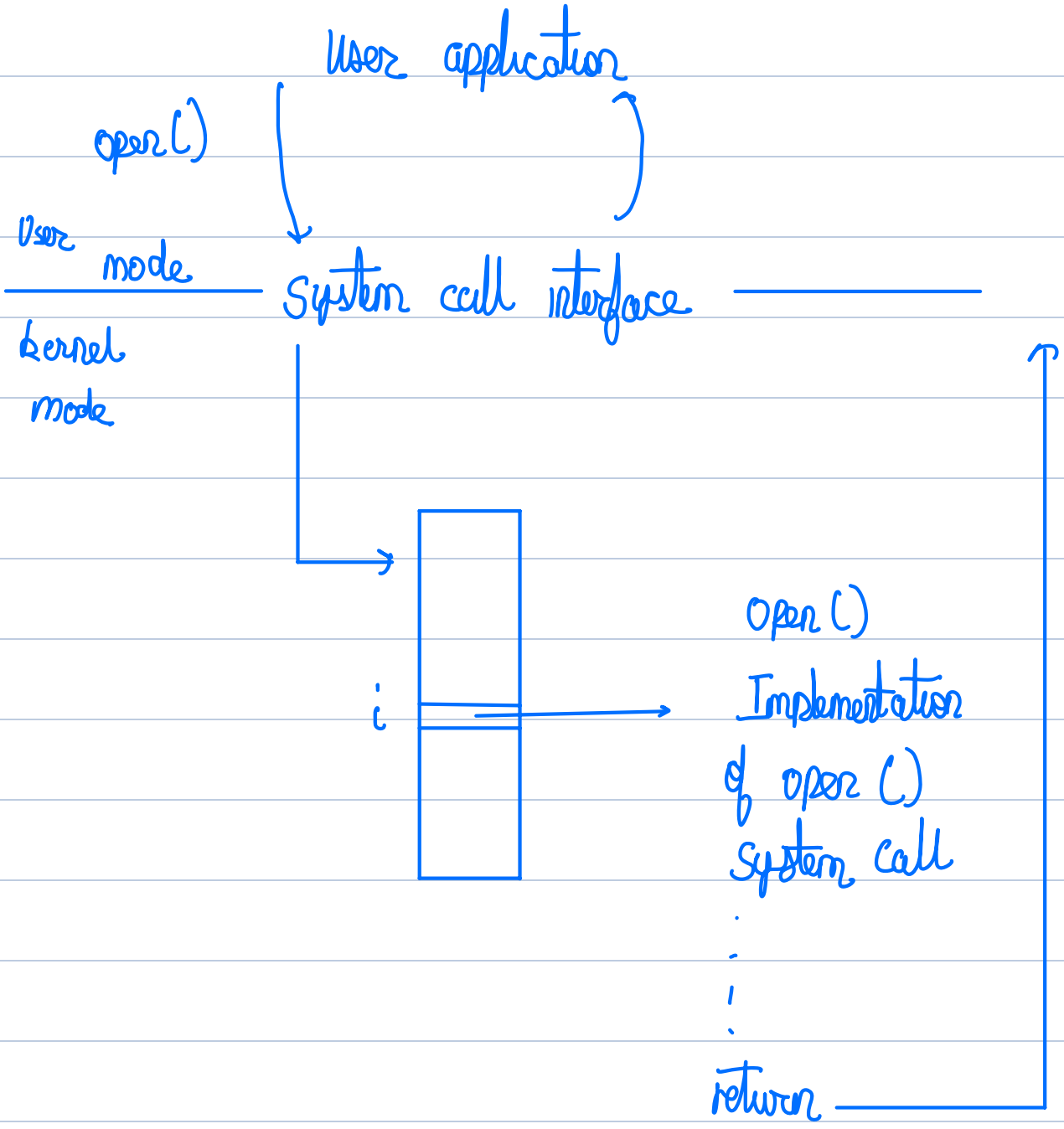
- Touchscreen devices require new interfaces
  (i) Mouse not possible or not desired
  (ii) Actions and selection based on gestures
  (iii) Virtual keyboard for text entry
  (iv) Voice Commands

→ System calls-

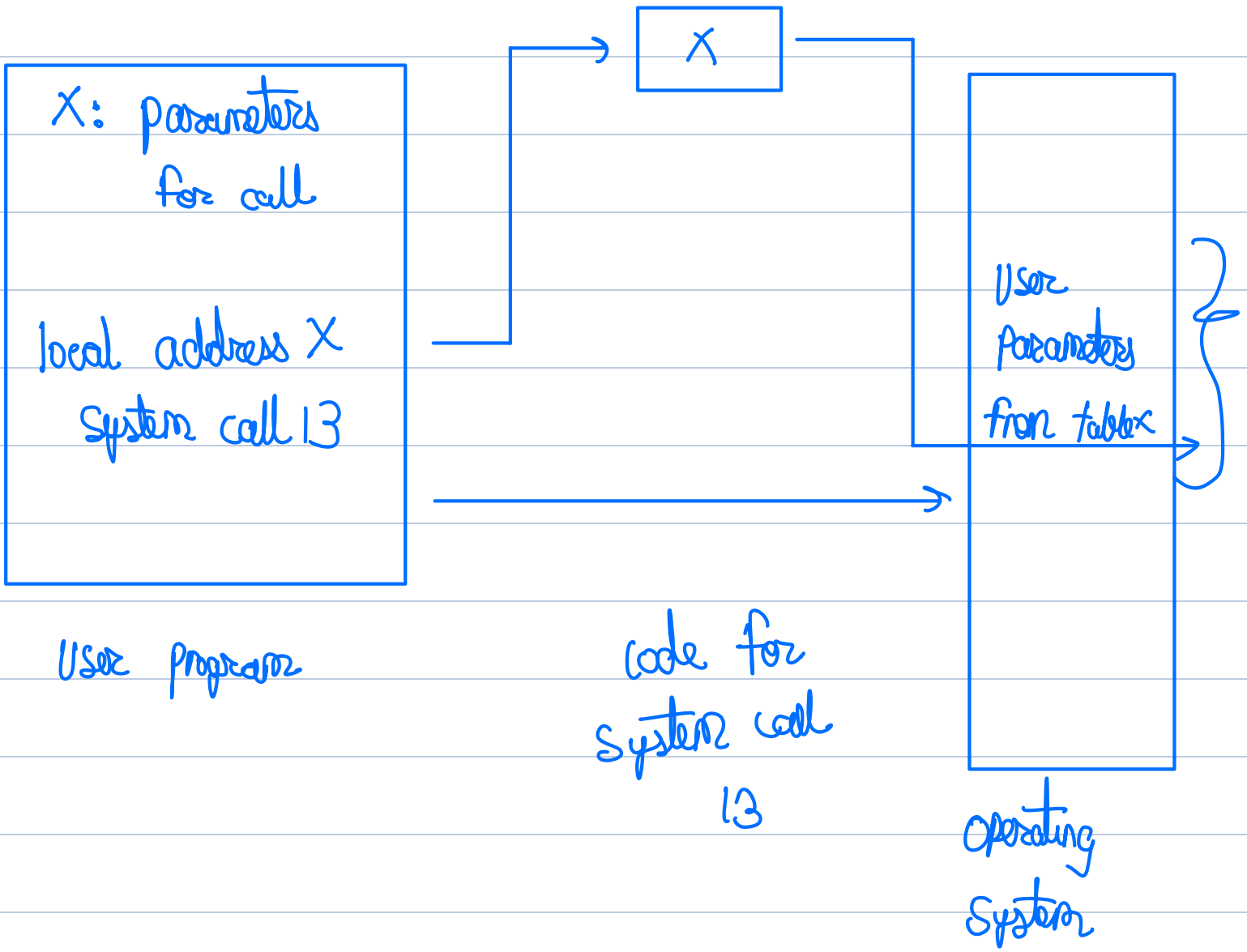
- System call is a programmatic way in which a computer program requests a service from the kernal of the operating system it is executed from



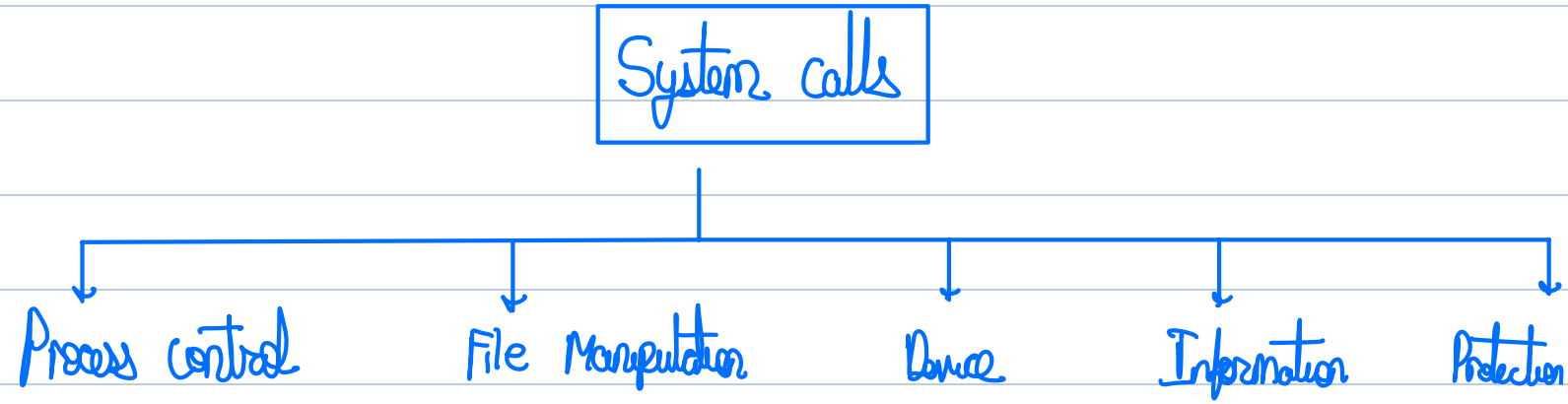
→ API- System call OS Relationship-



→ Parameter passing via table-



→ Types of system calls-



manipulation maintenance

1.) Process control -

(i) create processes (or) terminate processes
(ii) end, abort
(iii) load, execute
(iv) get process attributes, set process attributes
(v) allocate and free memory
(vi) Dump memory for corrupted system files
(vii) Debugger for debugging

2.) File management -

(i) create and delete file
(ii) open, close files
(iii) read and write files
(iv) get and set file attributes

3.) Device Management-

Deals with various resources controlled by OS
(i) request device, release device (from system with multiple users)
(ii) read, write operation
(iii) get, set device attributes
(iv) logically attach (or) detach devices

4.) Information Maintainance

- Used to transfer information between user program and OS

(i) get time or date, set time or date
(ii) get system data, set system data
(iii) get and set process, file (or) device attributes

5.) Communications-

(i) Create, delete communication connection
(ii) Send, receive messages if message passing model to

host name (or) process name

(iii) transfer status information

(iv) attach and detach remote devices

6.) Protection

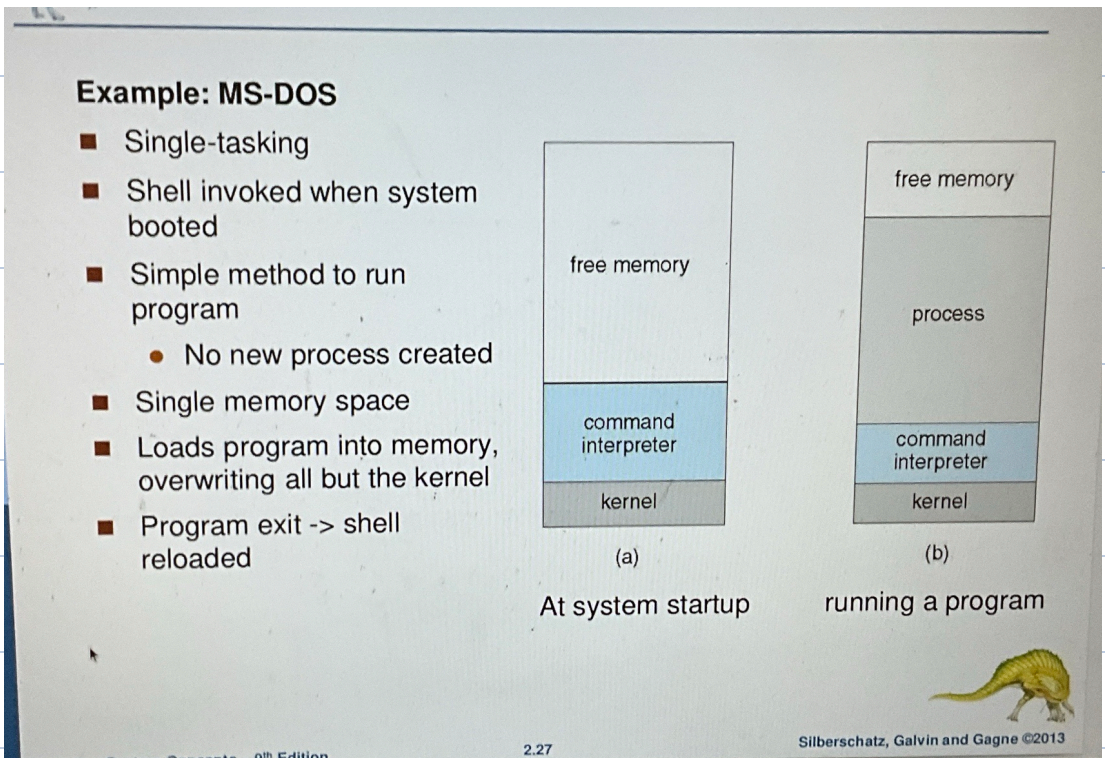
- provides mechanisms for controlling access to the system resources

(i) control access to resources

(ii) get and set permission

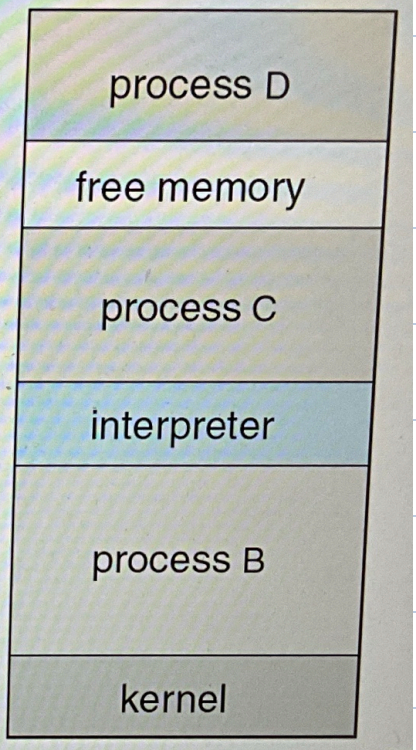
(iii) allow and deny user access for resources

→ Variant in process control -



Example: FreeBSD

- Unix variant
- Multitasking
- User login -> invoke user's choice of shell
- Shell executes fork() system call to create process
  - Executes exec() to load program into memory and execute
  - Shell waits for process to terminate or continues with user commands
- Process exits with:
  - code = 0 – no error
  - code > 0 – error code



→ Operating System Structure -

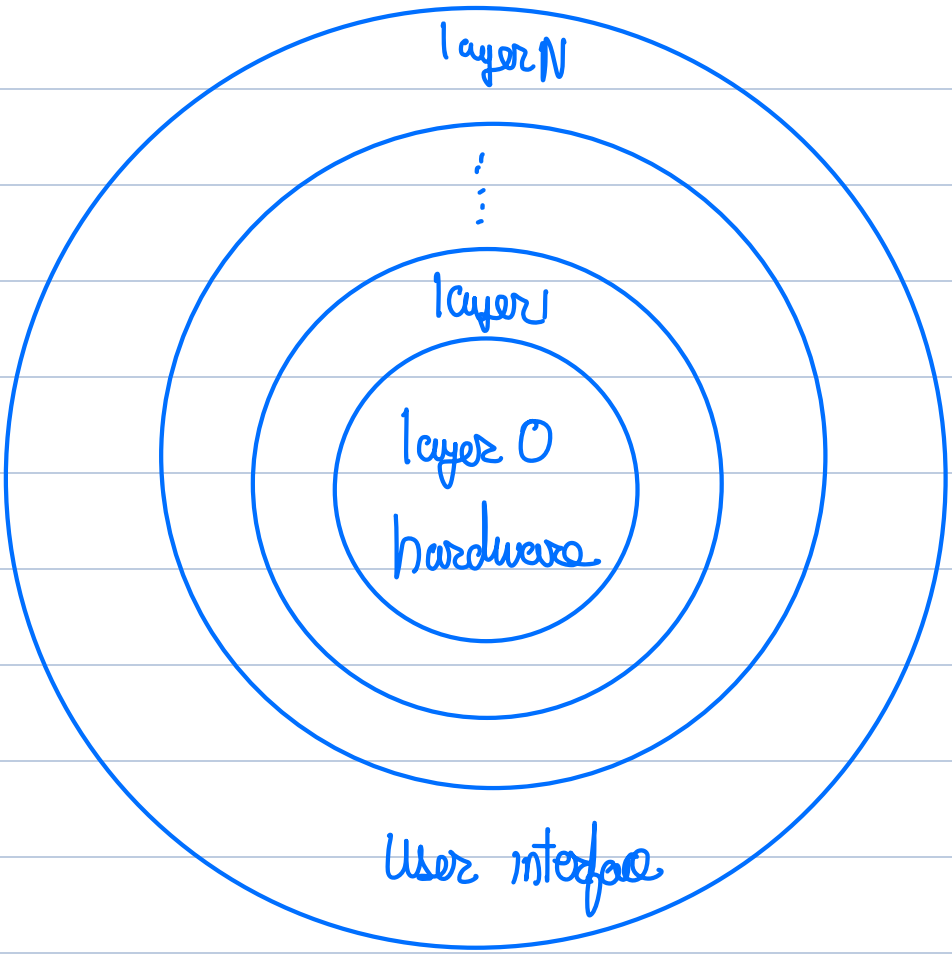
Simple Structure - MSDOS

More complex - UNIX

Layered - an abstraction

Microkernel - Mach

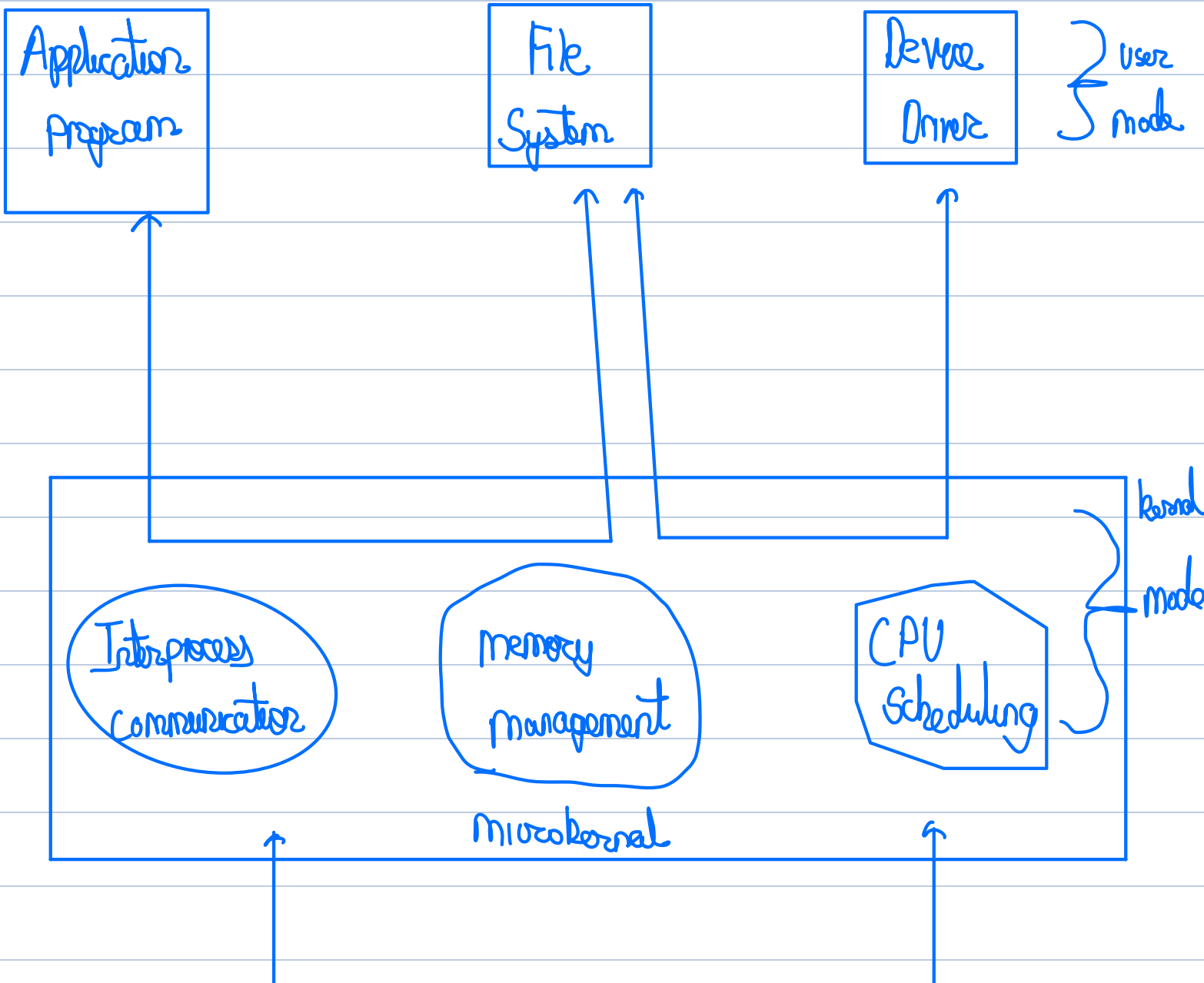
→ Layered approach -



- Divided into a number of layers
- Highest layer is the user interface
- Lowest layer is the hardware
- With modularity, layers are selected such that each uses functions and services of only lower layer capabilities
- Construction of the layers requires careful design because upper layers only make use of lower layers capabilities

- Less efficient as each layer adds overhead to System call. Each layer will pass parameter, modify and so on.
- Thus, System calls takes longer than the one on non-layered system.

→ Microkernal System Structure:



- Benefits of microkernel system:-

1) Easier to extend a microkernel
2) Easier to port the operating system to new architectures
3) More reliable
4) More secure as the remaining operating system remains unaffected and keeps running properly even if a microkernel fails.

→ Monolithic vs microkernal-

Monolithic	Microkernal
- A kernal where the entire OS works in the kernal space	- A kernal type that provides mechanism such as low level address space management, thread management and interprocess communication to implement an operating system
Kernal contains the OS services	OS services and kernal are seperated
Fast	Slow
Larger in size	Smaller in size
Difficult to add new functionalities	Easy to add new functionalities

X ——— X