Kernel :- The kernel is the heart of an operating system. It is the central control module of an os that provides basic system facilities.

→ It operates in a separate address space.
→ It is that part of an OS, which loads first & remains in the main memory
→ Since it stays in the main memory, it should be kept as small as possible.
→ It provides all essential services required by the other parts of the OS & other applications.
→ It is responsible for various tasks like memory management, process management, disk management etc.
→ It interfaces directly with the h/w.

Operating systems are broadly classified into following categories as per their structural architecture

(1) Monolithic Architecture of OS :-

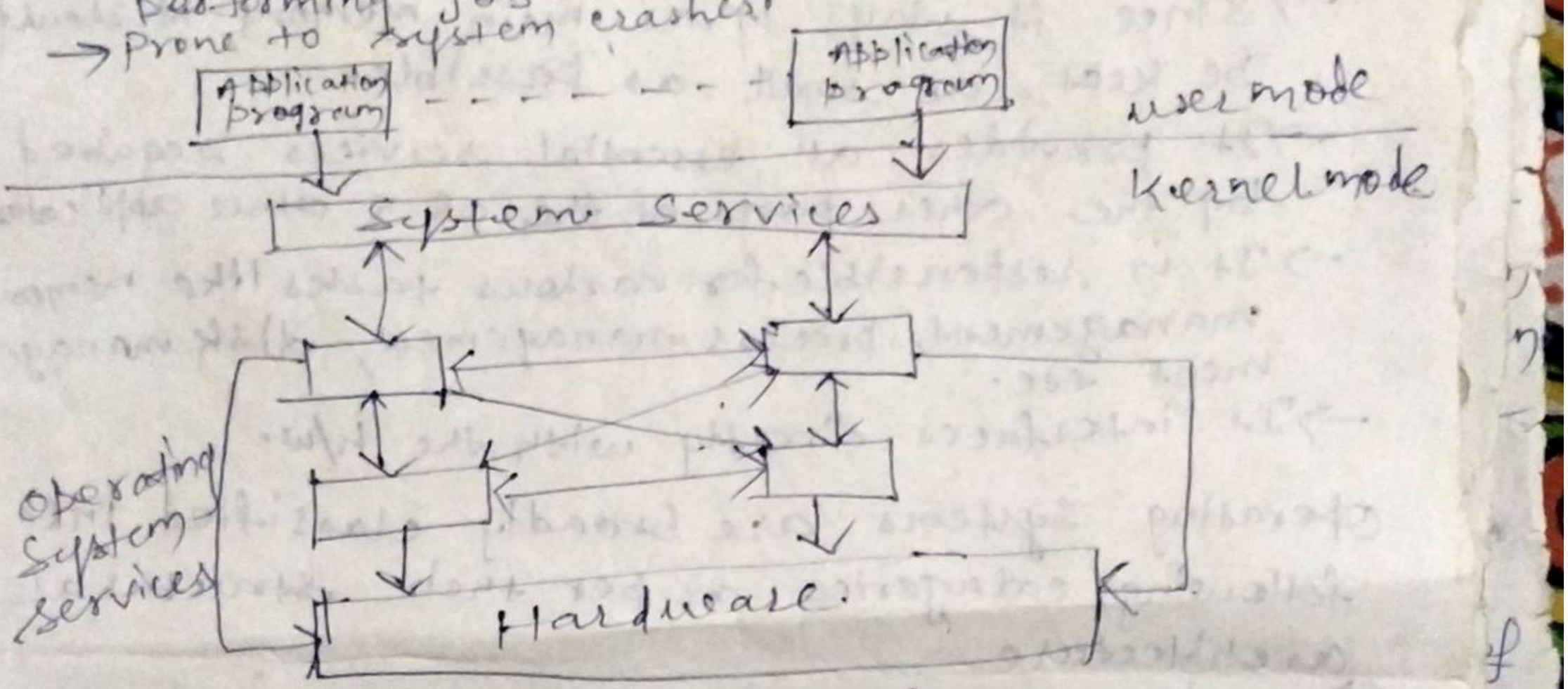
→ It is the oldest architecture used for developing an OS.
→ In this approach, the entire OS runs as a single program in the kernel mode or supervisor mode. So it can have direct access to the system data & h/w.
→ In monolithic kernels mode, most OS services such as process management, memory management, device management, file management, interprocess communication etc are provided by the kernel. So the kernel has a large monolithic structure.
→ Monolithic kernel gets loaded completely into the memory at boot time.
→ It consumes a large amount of memory space & increases the complexity of the system.
→ The components of a monolithic OS are

organized haphazardly & any procedure can call any other procedure. This makes the system complex & difficult to understand.

→ It provides better application performance but doesn't provide data hiding due to the visibility of procedures to each other.

→ since all the procedures run in same address space, so no message passing or context switching is required, while the kernel is performing job.

→ Prone to system crashes.



Ex MS-DOS, original UNIX OS

② Layered Architecture of OS:-

→ Here the OS is organized as a hierarchy of layers, where one layer is present on the top of the other.

→ The 1st system constructed in this way was the THE system developed by E.W. Dijkstra in 1968.

→ It has 6 layers.

layer 5: user programs.
layer 4: buffering for I/O device
layer 3: Operator-console device driver
layer 2: Memory Management
layer 1: CPU scheduling
layer 0: H/W

The bottom layer was the hardware. The next layer implemented CPU scheduling & so on.

→ Here each layer can provide a set of functions that other module can call.

Advantages 1. One layer can be replaced without affecting other layer.

2. The main benefit of this approach is modularity, i.e each layer can access to only those layers that are at a lower level than that layer. For eg:- Nth layer can access services provided by the N-1 layer & provides services to N+1th layer. So user program at layer 5 doesn't have to worry about process, memory or I/O management.

Disadvantages

1. It requires appropriate definition of various layers.
2. Here each layer adds an overhead to the system call. So the system call takes longer time that of a non-layered system.

Ex- Venus, OS/2, UNIX etc, ~~xxxxxx~~, multics

(3) Microkernel Architecture of OS :-

→ A microkernel was a kernel that was micro in size. So the kernel is the core code of the OS which provides the minimal facilities, for the implementation of the OS services. It runs in kernel mode.

→ The only services provided by the kernel are interprocess communication, low-level device management, a limited amount of process management & memory management.

→ All other OS services such as file management, additional process management, memory management, device driver, program & other system services are implemented in the user space or user mode.

Advantages

1. It consumes less space in the main memory.

(4)

(2) It is highly modular in nature. So the implementation of os in this mode is easier & also implementation is easier.

(3) Easy to modify since most of the services are implemented as user level processes.

(4) Access to the system resources is more restricted. So more secure.

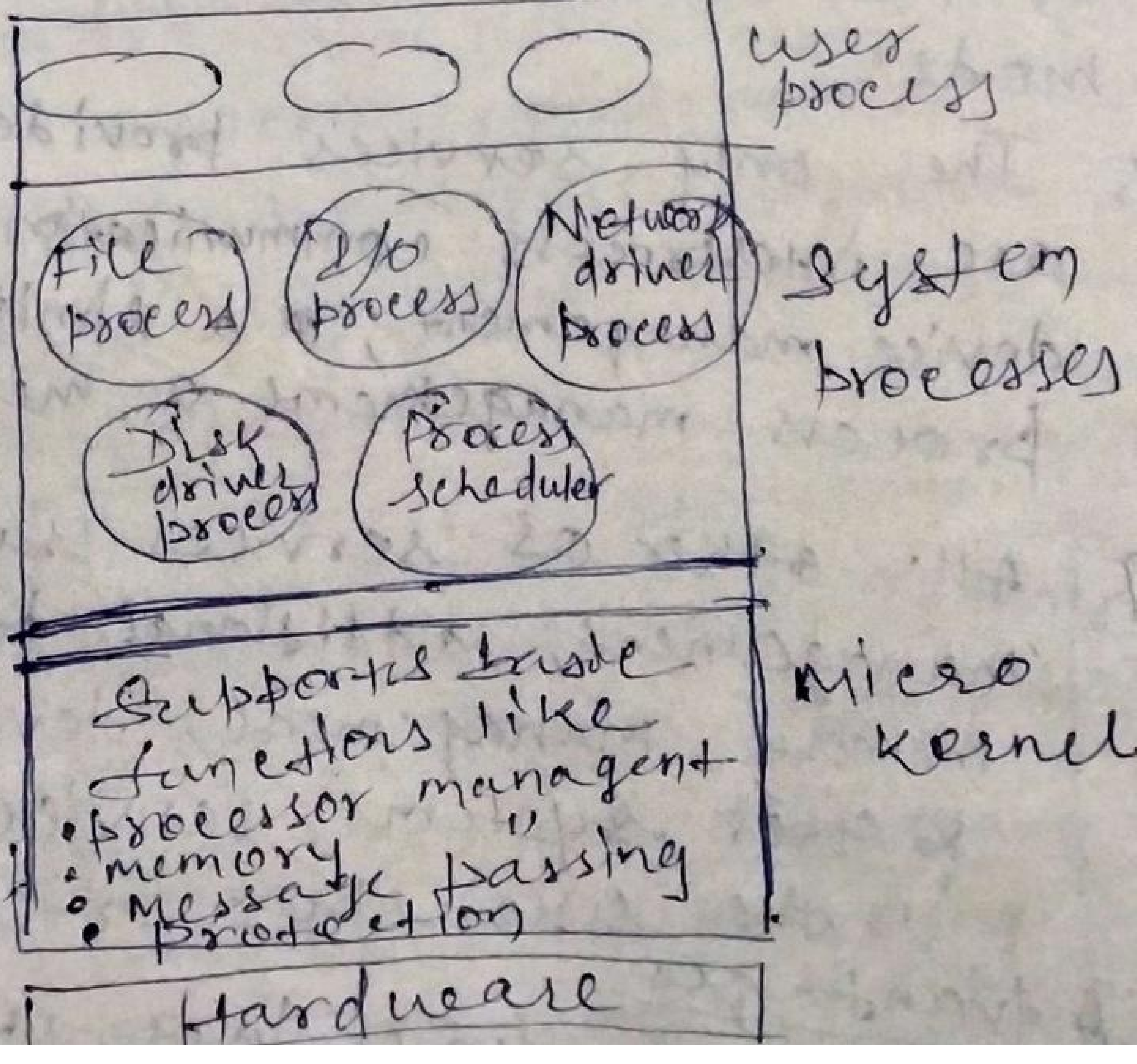
(5) Failure of one function can crash a single component but not the entire system.

(6) More reliable since less code is running in kernel mode

Disadvantages

1. Here message based interprocess communication is used to communicate between processes. So message passing between processes & microkernel requires more context switching, which is an additional overhead.

2. A request may be serviced faster in monolithic kernel mode than that of microkernel mode. But still the performance of microkernel is not poorer in practice, because other factors dominates this.



Ex- Windows NT, MACH, QNX, GNU Hurd kernel & etc