ABID ALI
2019380141

Homework Assignment

(2)

1. What is the purpose of system calls?

Ans: System calls provides the services of the OS to the user program via Application Program Interface (API). It provides an interface between a process & OS to allow user-level processes to request services of the operating system. System calls are the only entry points into the kernel system.

In general, system calls are required in the following situations:—

#If a file system requires the creation or deletion of files. Reading & writing from files also require a system call.

Creation & management of new processes.

Network connections also require system calls. This includes sending

and receiving packets.

Access to a hardware device such as a printer, scanner etc requires a system call.

2. Describe three general methods for passing parameters to the operating system.

Ans: Passing parameters to the kernel for a system call must be performed differently than when using an ordinary function call. This is because a system call is performed by the kernel itself, which typically runs in a completely different address space than the process which made the call.

Thus, it's not possible to simply place system call parameters onto the process' stack as this will not be readily available to the kernel. There are three main methods to pass the parameters required for a system call:

(1) Pass the parameters in registers (this may prove insufficient when there are more parameters than registers).

(2) Store the parameters in a block; or table, in memory and pass the address of block as a parameter in a register. This approach is used by Linus & Solaris.

(3) Push the parameters onto a stack, to be popped off by the OS. Block & stack methods do not limit the number or length of parameters passed.

3. What is the main advantage of the layered approach to system design? What are the disadvantages of the layered approach?

Ans: The layered approach is one of the operating-system structures. Since a modern OS system is large & complex structure it a common approach is to build it creating smaller modules instead of one large structure. The layered approach is one of these modular methods where the engineers partition the operating system in a number of levels & each level has different functionalities. In the layered approach the hardware is the lowest level (layer 0) & the user interface is the highest (layer N)

Advantages

- **simplicity of construction:** Since, operating systems are complex & large structures it's easier to divide them into module & engineering each one at a time. This factor made the layered approach better than the monolithic one. Higher level can implement lower level operations, in this way we don't need to know lot of details of process of implementation. So, the programmers has lot of freedom when making changes into lower levels. At the same time doesn't need to make extra changes in higher level since higher levels don't need details of lower level. So, it made debugging of designing, implementing & maintaining system easier.

- **Easier debugging and system verification**

In layered approach each layer uses operations belonging only to lower-level layers while operations of higher-levels are hidden. This means debugging made easier since the programme can start from the first level where issues arise & continue it's way to top. Since, each layer cannot use operations belonging to higher-levels after debugging the first layer the programmer can assume that layer will function correctly while trying to debug the other upper layers.

Disadvantages

- **Defining the layers :** The majority difficulty with the layered approach involves appropriately defining the various layers. Layers cann't access higher levels & they can use only the function belonging to the layer itself & any other layer. It means engineers must know before hand what function each layer need to perform. in order to avoid failure. It's very tedious because it need well thought plan.

- **Efficiency :** The layered approached tends to be less efficient than other types since each layer adds overhead to the system call. This happens because at every layer data needs to be passed on & the system call parameters might be modified. Since, there are several layers betwn the layers which made the call & the kernel it takes more time for the operating system to respond to system calls.

4. Explain the difference betⁿ a monolithic kernel and a microkernel

Ans:

Monolithic kernel

It's a single large process running entirely in a single address space. It is a single static binary file. All kernel services exist and execute in the kernel address space. The kernel can invoke functions directly. Examples of monolithic kernel based

Example: Unix, Linux

Microkernel

The kernel is broken down into seperate processes known as servers. Some of the servers run in kernel space & some run in user-space. All servers are kept seperate & run in different address spaces.

Example: Mac OSX & Windows NT

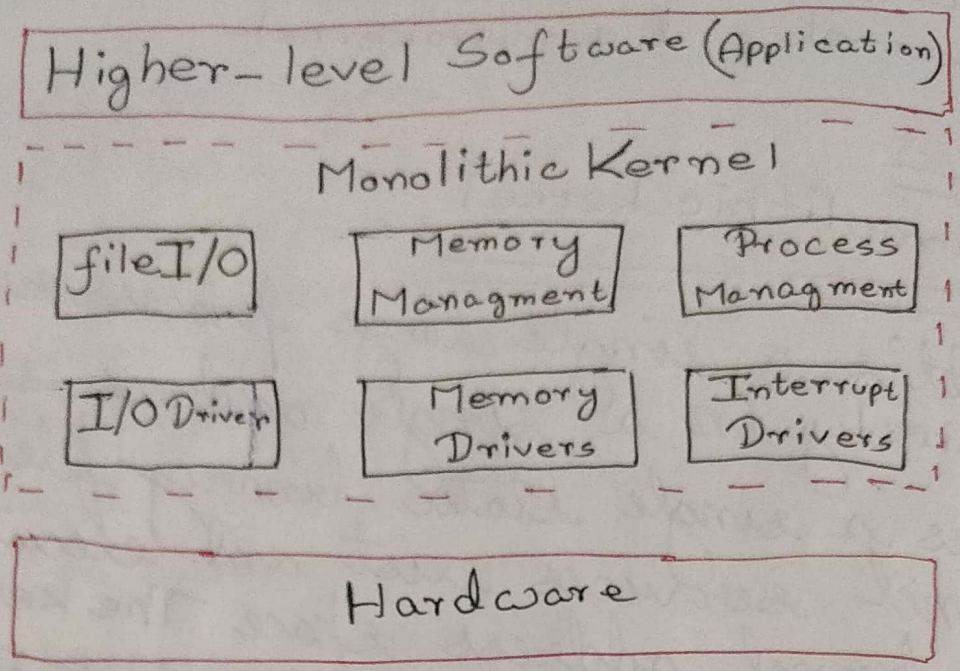

Fig: Monolithic Kernel

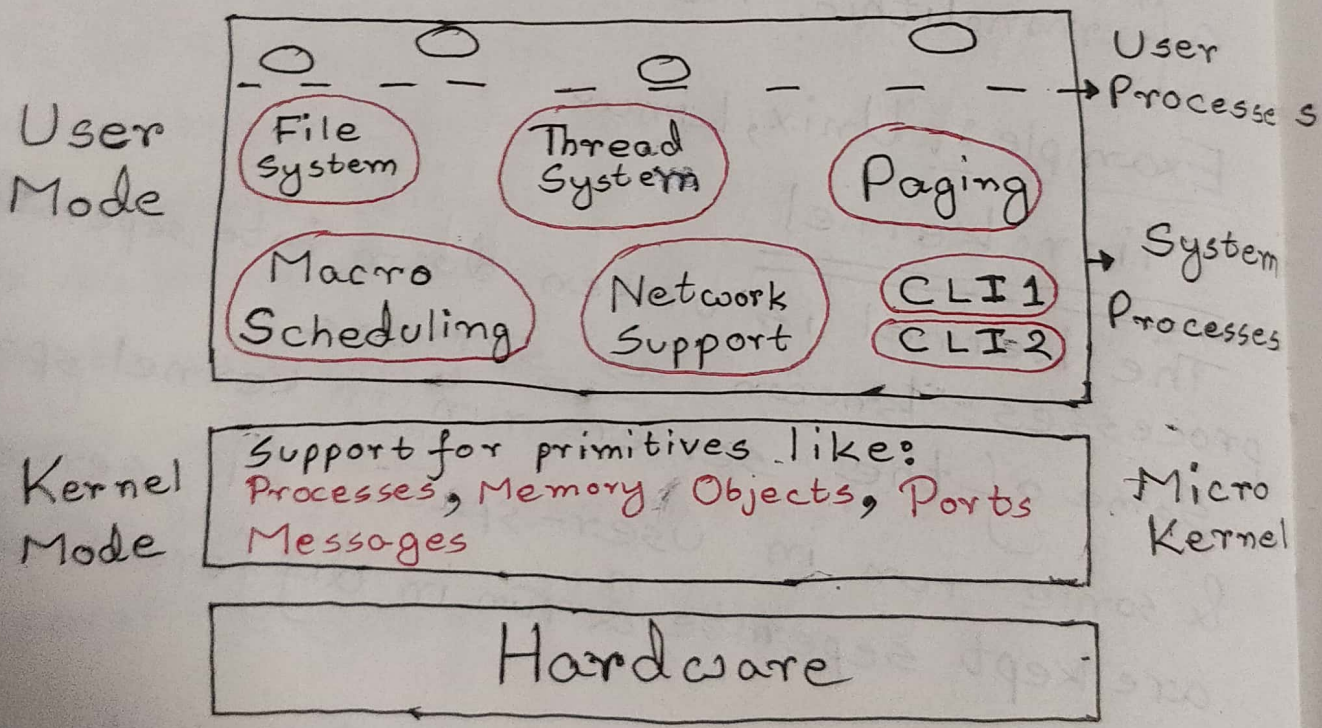

Fig: Micro-Kernel/ Architecture

Microlithic kernel

Difference betw microkernel & monolithic kernel

Microkernel kernel/ Microlithic kernel	Monolithic kernel
1) A kernel type that provides mechanisms such as low-level address space managment, thread managment and interprocess communication to implement an operating system.	1) A type of kernel in operating systems where the entire operating system coorks in the kernel space.
2) OS services & kernel are separated	2) Kernel contains the OS services
3) Slow	3) Fast
4) Failure in one component not affect the other components.	4) Failure in one component will affect the other components.
5) Easier to add new functionalities	5) Difficult to add new functionalities
6) Smaller in size	6) Larger in size