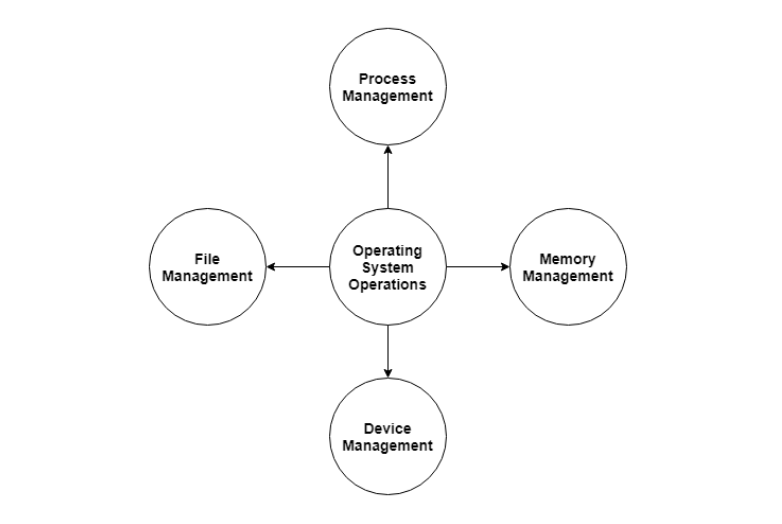
**Explain the operations of OS?**

An [**operating system**](https://www.tutorialspoint.com/operating_system/index.htm) is a construct that allows the user application programs to interact with the system hardware. Operating system by itself does not provide any function but it provides an atmosphere in which different applications and programs can do useful work.

The major operations of the operating system are process management, memory management, device management and file management. These are given in detail as follows:



**Process Management**

The operating system is responsible for managing the processes i.e assigning the processor to a process at a time. This is known as process scheduling. The different algorithms used for process scheduling are FCFS (first come first served), SJF (shortest job first), priority scheduling, round robin scheduling etc.

There are many scheduling queues that are used to handle processes in process management. When the processes enter the system, they are put into the job queue. The processes that are ready to execute in the main memory are kept in the ready queue. The processes that are waiting for the I/O device are kept in the device queue.

Memory Management

Memory management plays an important part in operating system. It deals with memory and the moving of processes from disk to primary memory for execution and back again.

The activities performed by the operating system for memory management are −

* The operating system assigns memory to the processes as required. This can be done using best fit, first fit and worst fit algorithms.
* All the memory is tracked by the operating system i.e. it nodes what memory parts are in use by the processes and which are empty.
* The operating system deallocated memory from processes as required. This may happen when a process has been terminated or if it no longer needs the memory.

**Device Management**

There are many I/O devices handled by the operating system such as mouse, keyboard, disk drive etc. There are different device drivers that can be connected to the operating system to handle a specific device. The device controller is an interface between the device and the device driver. The user applications can access all the I/O devices using the device drivers, which are device specific codes.

**File Management**

Files are used to provide a uniform view of data storage by the operating system. All the files are mapped onto physical devices that are usually non volatile so data is safe in the case of system failure.

The files can be accessed by the system in two ways i.e. sequential access and direct access −

* **Sequential Access**

The information in a file is processed in order using sequential access. The files records are accessed on after another. Most of the file systems such as editors, compilers etc. use sequential access.

* **Direct Access**

In direct access or relative access, the files can be accessed in random for read and write operations. The direct access model is based on the disk model of a file, since it allows random accesses.

**Advantages of threads over process and explain Kernel-level threads**

**Advantages of Threading**

* Threads improve the overall performance of a program.
* Threads increases the responsiveness of the program
* Context Switching time in threads is faster.
* Threads share the same memory and resources within a process.
* Communication is faster in threads.
* Threads provide concurrency within a process.
* Enhanced throughput of the system.
* Since different threads can run parallelly, threading enables the utilization of the multiprocessor architecture to a greater extent and increases efficiency.

**Kernel level**

**2. Kernel level Thread:**

**Kernel level threads are implemented and managed by the OS**.

* Kernel level threads are **implemented using system calls and Kernel level threads are recognized by the OS**.
* Kernel-level threads are **slower to create and manage compared to user-level threads**.
* **Context switching in a kernel-level thread is slower**.
* Even if one kernel-level thread performs a blocking operation, it does not affect other threads. Eg: **Window Solaris**.

