Date: 19/09/2021
02:00 AM.

01. OPERATING SYSTEM.

GFG Placement course.

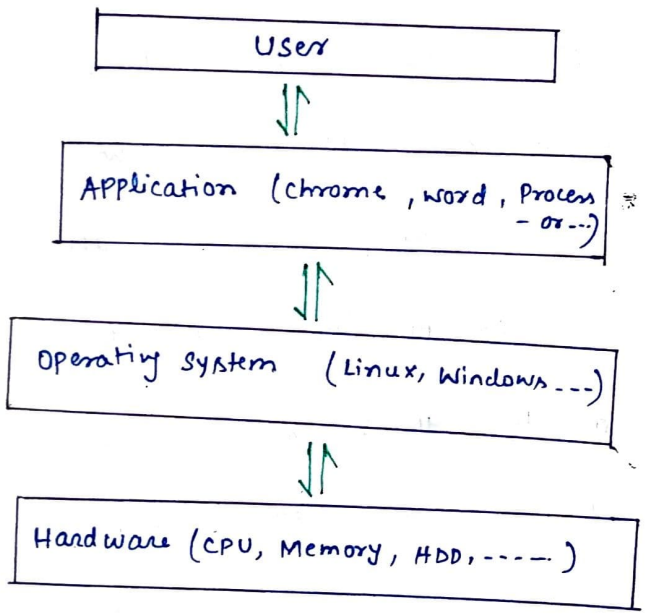
Powered by Abhishek Sharma Notes.

01. INTRODUCTION to OPERATING SYSTEM.

Operating System: Operating system is a software that takes ~~more~~ control of your machine after it is turned on.

* Whatever application you run (Ex. opening chrome, or wordpad or your own program you are running) you need to go to operating system to get your things done.

Operating system is like a manager.



* Your Application do not need to interact ~~directly~~ with Hardware, Applications ~~talk~~ ~~talk~~ talk to the operating system and Operating System talk with the Hardware.

Q) Why OS acts like a manager ?

A: Imagine you are writing an application (Ex. Word Processor) and you want to handle all the things (like if someone is typing it should be visible on screen, moving mouse cursor)

if OS was not their then we have to do all these things by our own (moving cursor, Display the data, read the data). So it will be very tough for us to handle all these situation).

Operating system provides you the basic services and you call them System calls. (Display, read, moving cursor and so on).

* This process is called Abstraction.

* Operating system provides us Resourse Management.

Resourse Management: you are doing multiple things simultaneously like you are running Chrome, firefox, mx player, VLC Media player at a same time. In your CPU, there is a limited space for running all the application by one by one manner. So operating system provides us the management of all the things done in a sequencly manner.

So OS provides us Resourse Management.

Protection: OS also provides us protection from our applications. Ex. You don't like if any Application making changes in your other application. So OS provides us Protection from doing so.

So, The services provided by OS is ⟹

i) Abstraction
ii) Resource Managment
iii) Protections.

Q) Do we need OS every time?

A) No, in simple operation we don't need OS. Like a chip. (If a chip does the simple task) then we don't need OS. But if we have complex tasks to do and we need to mange them then we need OS.

Q) Examples of operating systems?

Desktop:	Windows
	Linux
	Mac OS.
Mobile:	Android
	ios

// These are the most popular OS. There are many many OS. but these are the most popular among them.

02. Types of operating system.

According to the functionality that operating systems provides the types are:

i) single Tasking
ii) Multiprogramming and Multitasking
iii) Multithreading
iv) Multiprocessing
v) Multiuser

(1) Single Tasking OS: These OS are very very basic. They do not provide multiple functionality. They just exist in RAM. They are loaded in the RAM. Then they allow only one process to be their in the RAM at a time. and to be run at a time.

Ex: MS-DOS

Remember: Whenever you want to run a program. your program should be their in the Memory. your CPU talks only with the ~~Memory~~ programs which are their in the memory.

* Single Tasking os are very inefficient. (very very slow).

② Multiprogramming and Multitasking: This is the Great idea.

The idea is to ~~to~~ use single CPU to allow multiple processes run concurrently. (Read in DBMS).

* We have multiple programs in the memory and they can execute concurrently.

* Multitasking systems are more responsive.

* Multiprogramming and Multitasking os are very efficient. (very fast).

③ MULtithreading:

In Multithreading, you have multiple Threads running within a process in interlived fashion.

Threading: A Thread is called smallest unit of execution that can be assigned to CPU. A process can single thread or multiple thread.

Advantages of Multithreading.

i) More responsive process.

ii) Switching from one thread to another thread ~~is~~ within a process is less costly.

* Multithreading is implemented by almost all known os. like, linux, windows

④ Multiprocessing :

Having an operating system supports for multiple ~~processes~~ processors.

So far we have talked about single processors.

There can be multiple processors available in computers.

and your OS should be having the capability of utilise all the processors. (probably divide multiple thread into multiple processors, divide multiple processes into multiple processors). All this can be implemented by Multi processing OS.

These days if we talk about Desktop OS. They all have multiprocessing support in them. (like, Dual core, Quad core, Octa core).

⑤ Multiuser OS :

Early OS allows only one user to use the OS at a time.

All modern operating system (Windows 10, Linux, and so on) They provide multi user support.

Summary: Modern OS. they all have multiprogramming and Multitasking, Multithreading, Multiprocessing and Multi user features. (Ex. Windows 10, Linux and so on).

Happy Coding ☺

NOTES provided by Abhishek Sharma.
Ph 8290903490.
Date: 19/09/2021 Time: 8:00 Am.

Abhishek Sharma Notes.
Ph. 6290903490.

03. Multithreading Introduction

We will be covering the following topic in this video.

i) Multitasking Vs. Multithreading
ii) Some Real world Examples.
iii) Advantages and Disadvantages.

---

i) Multitasking Vs. Multithreading.

Multitasking: we can do multiple task at a same time.

2 or more tasks. Ex. Listening to music and browse the web at a same time.

Here task means process: (Ex. one Process. 2nd Process).

Multithreading: In Multithreading we can do multiple things within a process. Ex. When you Browsing a web you can download also.

Here
* Download and Browsing both the process can be done Simultenously with the Browser process.

* Real world Examples of Multithreading.

i) **Word - Processor:** Typing, Saving, formatting and spell - checking Happen at the same time.

ii) **Web server:** Apache HTTP Server uses Thread pools. (i.e. idea of these Thread pools is to have fast Response for HTTP Request).

Abhishek Sharma Notes.
Ph. 6290903490.

iii) IDEs: - Modern IDEs do compiler error checks while you are writing code.

iv) Games :- Modern Games, Multiple objects are implemented as different threads.

* Advantages of Multithreading:

i) Parallelism and Improved performance.

ii) Responsiveness

iii) Better Resourse ~~uti~~ utilization.

* Disadvantages of Multithreading:

i) Difficult to write, Test and debug code.

ii) It can lead to Deadlock and Race conditions when you have shared variables.

Examples of Race condition.

Thread 1	Thread 2
Read x	
x++	
write x	
	Read x
	x++
	write x

1st sequence of Execution

Initially $x = 5$, Finally $x = 7$.

Thread 1	Thread 2
Read x	
x++	
	Read x
	x++
Write x	
	Write x.

2nd ~~Execution~~ sequence of execution.

Initially $x = 5$, Finally $x = 6$
(Expected Result was = 7)

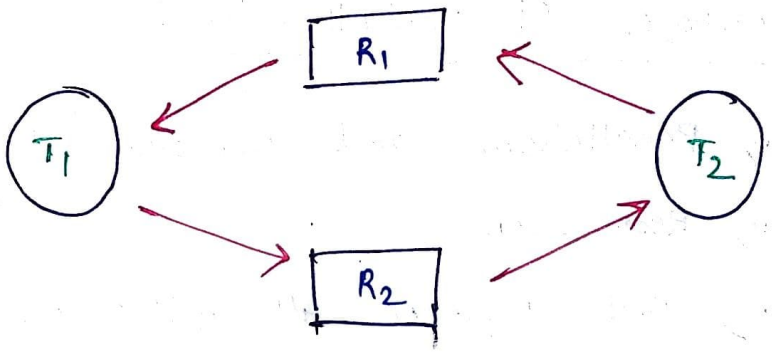
This is the Example of Race condition that might Happen when we have multiple thread Accessing the same memory. Or same variable.

Abhishek sharma Notes.
Ph. 6290903490

* The Other Disadvantage is Deadlock.

In Multithreading, A situation can arrive when multiple thread involved in Deadlock can't further proceed atall.

Example of Deadlock:



Thread
∧ T_1 is Holding the Resourse R_1 and waiting for Resourse R_2 and another Thread T_2 is waiting for R_1 and Holding R_2.

Hence None of the 2 Thread can procedd further.

T_1 is not ~~refer~~ Proceed further until R_2 is Released and T_2 is " " " " R_1 is " .

So In Multithreading this is the condition of Deadlock.

* Hence in Multithreading there are many advantages and also have some disadvantages. It requires expresience Programmer to write Multithread Program.

04. Thread vs Process

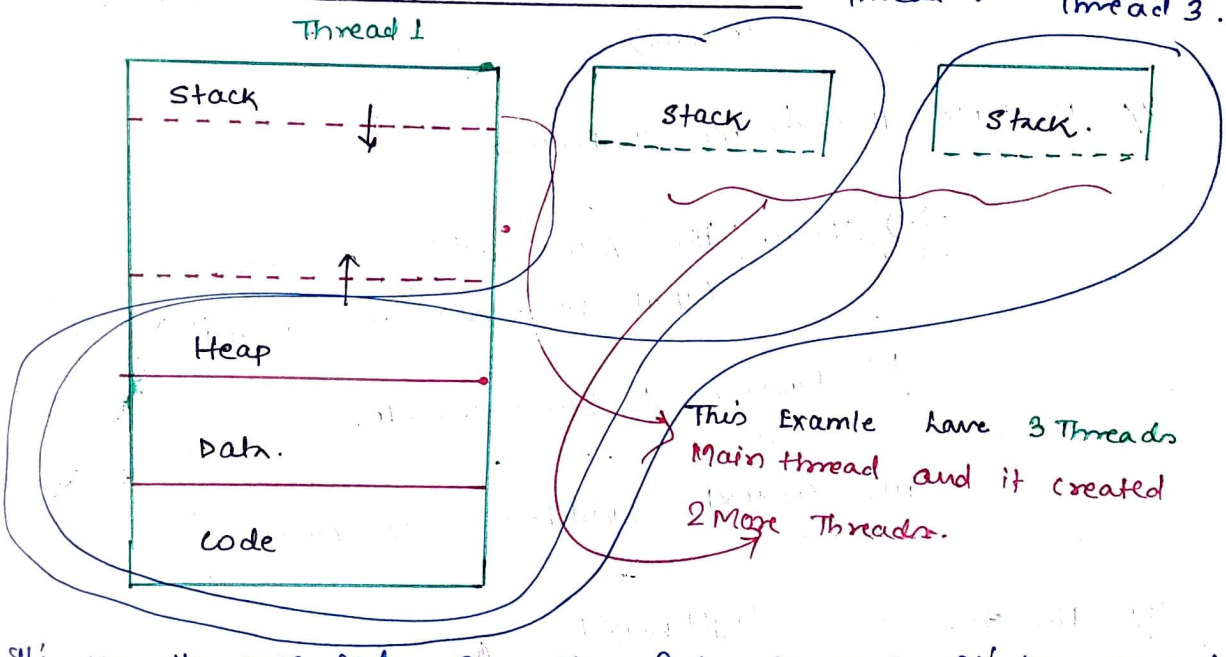
Abhishek Sharma Notes.
ph. 6290903490

Process : code, Data, Files, Heap, Stack etc.

Thread : Stack.

Most of the Modern applications are Multithreated. bcz. Multithreading does better CPU ilization and takes advantage of Multicore processor.

How process Looks Like in our Memory.



It's Has its own code segment, Data segment, It's own Heap of Meaory and it's own stack. Stack is needed to manage function calls. Typically a process is a programming execution and in program you have a main function and this main function might be calling other fuction. and so on.

All the 3 Thread share the same Heap, Data, code section.

Abhishek Sharma Notes.
Ph. 6290903490.

* If your process is not multithreaded then it will have only one stack and we will called such process Single Thread of Execution.

If your process is Multiple Threaded they you need multiple stack bcz Multiple function might be running concurrently ~~and~~ Or. parallely.

Difference b/w Thread and process.

1) Threads are faster to create and Terminate. (easier)

2) Multiple Threads in a process

   i) Share the same address space. (That we saws in our Example).

   ii) Easiar to communicate.

   iii) Context switching is easier.

3) Threads are light weight (consume less resources). bcz it is easier to create and Terminate.

05. USER Threads Vs. Kernel Threads.

User Thread: A process is creating multiple threads and Kernel is not aweare of this multiple threads. These Threads are managed by the process itself.

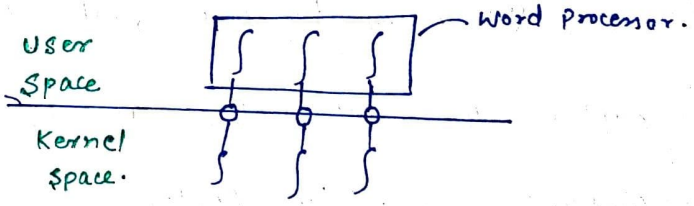
Kernel Threads: It is Managed purely in the Kernel space.

Abhishek Sharma Notes.
ph. 6290903490.

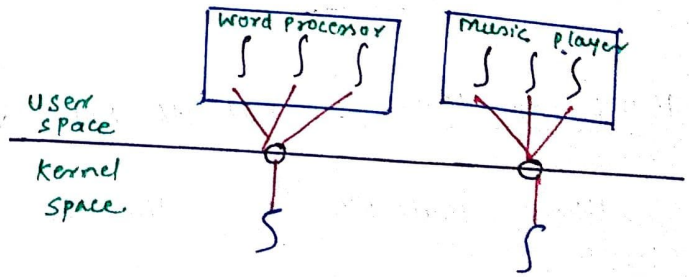
	User Managed Thread	Kernel Managed Thread
Management	In user space.	In kernel space.
context switching	Fast.	Slow
Blocking.	One Thread might block all other Threads.	A Thread blocks itself only.
Multicore or Multiprocessor	Cannot take advantage of multicore system. Only concurrent execution on single processor.	Take full advantage of Multicore system.
Creation/ Termination	Fast	Slow.

* Mapping of user Threads to Kernel Threads.

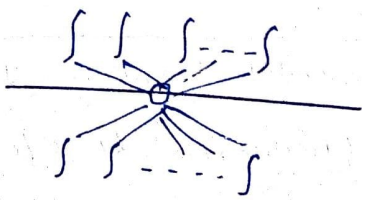
One to one



Many to one.



Many to Many.



Abhishek Sharma NOTES.
Ph. 6290903490

One to one.

If we take a look on One to One structure. In this Every user thread creates a Kernel Thread.

This is the most popular. This is suppored by Windows and Linux both.

This is the Mapping b/w user Thread and Kernel Thread.

In one to one context switching is slightly slowlyer compared to many to one. In one to one Blocking One Thread doesn't block all the Thread.

Many to one: This is purely user managed multithreading System. Bcz. Kernel is managing only one thread and their are multiple Threads which are running on top of it (in the ~~space~~ user space Area).

In many to one, if one thread calls the blocking System then all other threads are also blocked. context switching is faster

Many to Many.: It is not very much common. Here Multiple user threads are mapped with multiple Kernal Threads.

Real world Multithreading Example: Java Multithreading Libraries, porgus Library (Implemented by C/C++).