Tuesday, March 29, 2022 10:24 AM

Process - Program in execution

For every new process ---- what is loaded in RAM ? EXE file ----SPACE in the RAM (Code, Data ,Stack ,Heap)

+

Creating PCB (pid etc) ---ID Card

When many processes are waiting in the ready q they execute in RR ---- Multiprocessing

Thread = **Path** of program execution (Sequence of instructions that will execute from start to end)

A process must have at least ONE THREAD !!!!!!

Every process has at least one path of execution----Default thread = MAIN THREAD !!!

DEFAULT PATH is defined by

starting point **void main() {**

… statements

function calls

……

end point **} //end of main**

Thread Life Cycle = Process Life Cycle = life cycle state = created, ready, wait, execution, terminate

Process Create State	Thread Create State
New Code, data , stack , heap space created on the RAM	A new stack space is added to the process
	Light Weight process
PCB is created with PID etc	TCB is created with TID etc

MULTITHREADING = We can create MORE than one thread in a single PROCESS, threads will compete with each other in RR .

For each thread a new stack in the process space and a new TCB is created!!!!

WHY do we need many threads in a single process ? We want to start many tasks such that one task is not waiting for another task to complete ----- Simultaneous execution !!!!

Sequential execution	**Simultaneous execution**
Single Thread Model	Multi thread model
task1 task2 task3 Task3 will start only after	task1 starts ----return to ready Q task2 starts -----return to ready Q task3 starts -----return to ready Q task1 resumes -------- return to RQ Task3 resumes --- return to RQ

task2 is done Task2 will start only after task1 is done First task1 will start ----- finish Then only task2 will start --- finish Then only task3 will start ……finish	task3 resume ------return to RQ This will go on in RR till the tasks are not done!!! All the tasks are started without waiting for other task to finish and they will run in RR . E**ach task is running in a separate thread**

What are the common examples of Multithreading ? -------

Process	Threads
Word Document MS WORD	Editor thread , spell checker thread , auto save thread
Zoom	Recording thread , chat thread , screen share thread , audio thread, video thread , join thread
Eclipse IDE	Editor thread , auto assist thread , command prompt thread, error display
Game	Timer thread , music thread, player thread , ball threads, guidelines thread, score thread
Window Media player	Audio thread, edit playlist , volume thread ,…….

How to implement Multithreaded application Using C on Ubuntu ?

we will use a Linux library called as **pthread** = this is also called as POSIX thread

POSIX API - to create a thread

pthread_create (*threadID , *returnvalueof function, * function-path-of-execution, *args-to-function)

pthread_create(&th1, NULL , (void*) editor , NULL)
pthread_create(&th2, NULL , (void*) spellchecker , NULL)

_____

POSIX TYPE

pthread_t = inbuilt data type of unsigned long int = it stores the thread ID

_____

POSIX API for a thread to check its own thread ID = POSIX API == **pthread_self()**

_____

POSIX API to pause a thread till another thread is complete --- pthread_join()

_____

If main thread is creating th1 and th2 !!!
Main is the parent of th1 and th2 ---- if main thread terminates then the other threads are forcefully terminated .

_____

Windows is a Multhithreaded OS

Kernel Threads manage User Threads

This management may happen in many ways

1	One to one	One kernel thread manages one user thread
2	One to many	One kernel thread manages many user threads
3	Many to many	Few kernel threads manage all user threads

_____

SHELL PROGRAMMING ----------

SHELL --- Terminal -----BASH is the default shell .

We can write a program using BASH commands ----- SHELL PROGRAM

SOURCE CODE = XYZ**.sh**

Shell commands can be executed
1. On the prompt
2. Through the shell program

Ex...

1. Write a shell script to print HELLO WORLD , RUN it
   No compilation is needed to run a shell program. SHELL is an **interpreter** -- it reads the input file line by line and then it executes it line by line !

gcc	bash
Compiler	Interpreter
It generates output file using -o option	No output file is generated here
We run the output file	We run the source file

2. WASS to declare a variable and print its value , Run it
   x=10

Use a META CHARCHTER **$ to print the value of x in echo**
If u add spaces between = then not allowed

3. Write a shell script to accept a number from the user and show the number.
Accept two numbers from the user and show the numbers

4. accept a path and folder name from the user and create a folder of that name in the pathfolder , show content of path folder

5. accept a path and folder name and remove the folder from the path folder , show content of path folder

EX **...**TRY creating one more thread autosave in the same program written in class!!!