**MULTI THREADING:**

Process---execution of a task.

To perform multiple application using command prompt there should be link between them, that link is called thread.

These are light weighted threads or process.

**Concurrency:** Ability of a system to perform multiple tasks simultaneously or handle multiple tasks in a single time frame.

**1.Responsiveness:** Concurrency allows applications to remain responsive to user actions even when performing long-running tasks in the background.

**2. Utilizing multi-core Processors**: Concurrency allows software to utilize these multiple cores by distributing tasks across them, resulting in faster and more efficient execution.

**3.Efficient Resource Utilization:** Concurrency allows programs to better utilize available system resources by avoiding idle times.

**4.Scalability:** Concurrency enables applications to scale better, especially in server environments where multiple clients or tasks must be handled simultaneously.

**5.Improved Throughput:** Throughput, or the amount of work an application can process in a given amount of time, is often increased by handling multiple tasks concurrently.

**CONCURRENCY IN PROGRAMMING:**

**Multithreading:** Multiple threads within a single process are created to perform different parts of a task or handle multiple tasks concurrently.

**Threads share the same memory space,** which allows for efficient communication but requires careful synchronization to prevent race conditions.

**Multiprocessing:** Separate processes are created to handle different tasks concurrently.

**Each process has its own memory space,** which is safer but requires inter-process communication methods like pipes or shared memory for collaboration.

**Asynchronous Programming:** Non-blocking functions are used to handle tasks such as I/O operations without blocking the main thread, allowing other operations to proceed concurrently. This is common in languages like JavaScript and Python.

**Concurrency vs Parallelism:**

Concurrency is about dealing with multiple tasks at once or having multiple tasks in progress.

Parallelism is about executing multiple tasks truly simultaneously, typically achieved by using multiple CPU cores.

**Methods to Achieve Concurrency in C:**

1.Multithreading Using POSIX Threads(pthreads)

2.Using fork() for Multiprocessing

3.Asynchronous I/O

4.Atomic Operations

5.Semaphores

6.Message Queues

**Traditional view of a process:**

Process = process context + code, data, and stack

(or)

Process = thread + code, data, and kernel context-----alternative process

**A Process with Multiple Threads:**

Multiple Threads can be associated with a process

Each thread has its own logical control flow (sequence of PC values)

Each thread shares the same code, data, and kernel context.

Each thread has its own thread id(TID).

**Threads vs Processes**

**How threads and processes are similar:**

Each has its own logical control flow.

Each can run concurrently.

Each is context Switched

**How threads and processes are different:**

Threads share code and data, processes(typically) do not.

Threads are some what less expensive than processes.

**Creating and reaping threads:**

pthread\_create, pthread\_join

**Determaining your thread ID:**

pthread\_self

**Terminating threads:**

pthread\_cancel, pthread\_exit

exit[terminates all threads], return [terminates current thread]

**Synchronizing access to shared variables:**

pthread\_mutex\_init, pthread\_mutex\_[un]lock

**EX:**

#include <stdio.h>

#include<stdlib.h>

#include<pthread.h>

void printHello();

int main() {

pthread\_t tid;

int status=0;

printf("\nIn the Main Function\n");

printf("\nI am Main Program/Process/MainThread\n");

status = pthread\_create(&tid,NULL,printHello,NULL);

pthread\_join(tid,NULL);

printf("\n After completing (Main)\n");

sleep(1);

return 0;

}

void printHello()

{

printf("\nHello World\n");

}

If I want to convert to structure we need to typecast into the void \* type to catch it and execute that

**Why we use semicolon in do while, Structures, Unions etc?**