Process:Running an task.

\*To perform multiple application using command prompt there should be link between them, that link is thread, and thread is under controll of shell.

\*Light weighted thrreads or process

Concurrency:Refers to the ability of a system to perform multiple tasks(functions) simultaneously or handle multiple operation within overlapping time frames.

Necessity and benefits of concurrency:

\*Responsiveness

\*Utilizing multi core processors

\*Efficient Resource Utilization

\*Scalability:Efficiently and effectivly

\*Improved Throughput

what is concurrency in programming?

MultiThreading:multiple threads with in a single process are created to perform different parts of a task or handle multiple taks concurrently.

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

MultiProcessing:separate process 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 collabaration.

Asynchronous Programming:Non-blocking functions are used to handle tasks such as I/O operations without blocking the main thread.

Concurrency vs Paralleslism:

\*concurrency is about dealing with multiple taks at once or having multiple tasks in progress.

\*Parallelism is about executing multiple tasks truly simultaneously,typically achieved by having multiple CPU cores.

\*fork is method to create multiprocessing.

\*multithreading are lighter compare to multiprocessing because multiprocessors will have it own address which will leads to memory wastage.so, we commonly use

multithreading.

Traditional view of process:

\*process=process context+code,data and stack

(or)

\*process=thread+code,data and kernel context

A process with multiple threads

\*multiple threads can be associated with a process.

\*Each thread has its own logical control flow

\*Each thread shares the same code, data, and kernel context

\*Each thread has it's own thread id(TID)

Threads vs Processes

how threads and processes are similar Different

\*Each has its own logical control flow \*Threads share code,data,processes do not

\*Each is contet switched \*Threads are somewhat expensive less expensive than processes

Posix Threads(Pthreads)interface

pthreads:standard inteface for ~60 functions that manipulate threads

\*Creating and reaping threads

syntax:pthread\_create,pthread\_join

\*Determining your thread ID

syntax:pthread\_self

\*Terminating threads

syntax:pthread\_cancel,pthread\_exit

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

\*Synchronizing access to shared variables

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

/\*#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("\nAfter Completing (main}\n");

return 0;

}

void printHello()

{

printf("\nHello World\n");

}

\*/

\*(context switching)(synchronization)to avoid race condition we use lock, to use lock we need to first intiate the mutex and then it will appply.

we use race condition when there is shared resource between two threads are more.

\*when we detach a thread what will happen?