**What is the process?**

process is the execution of a program

**what is a thread?**

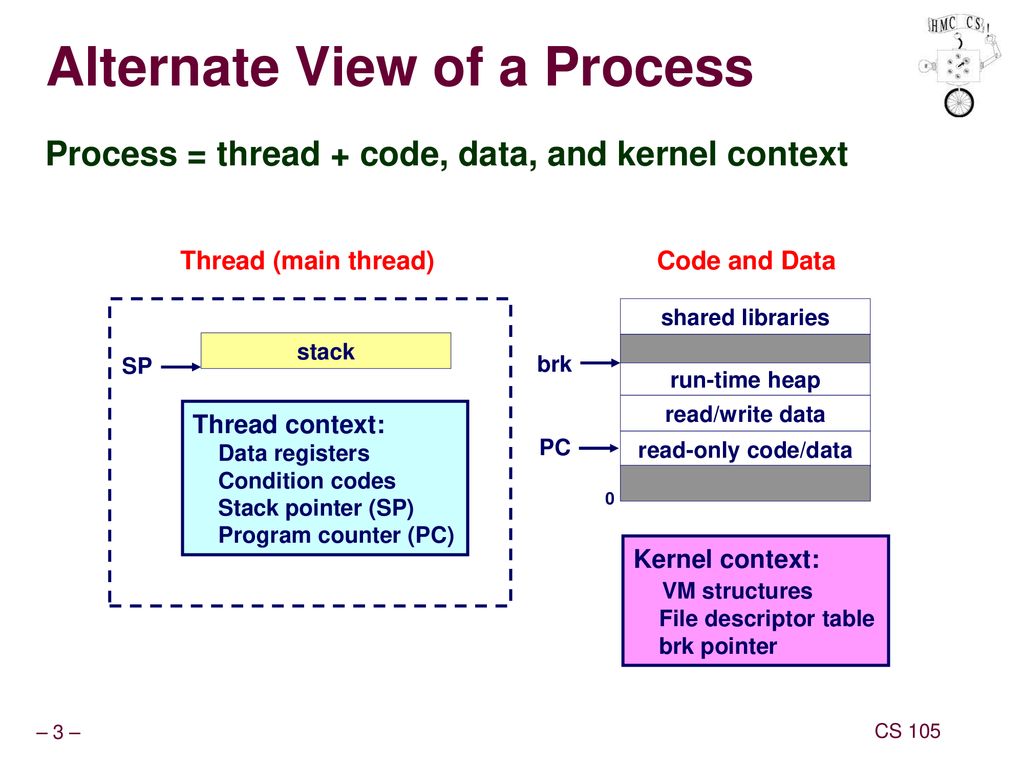
Thread is a lightweighted process

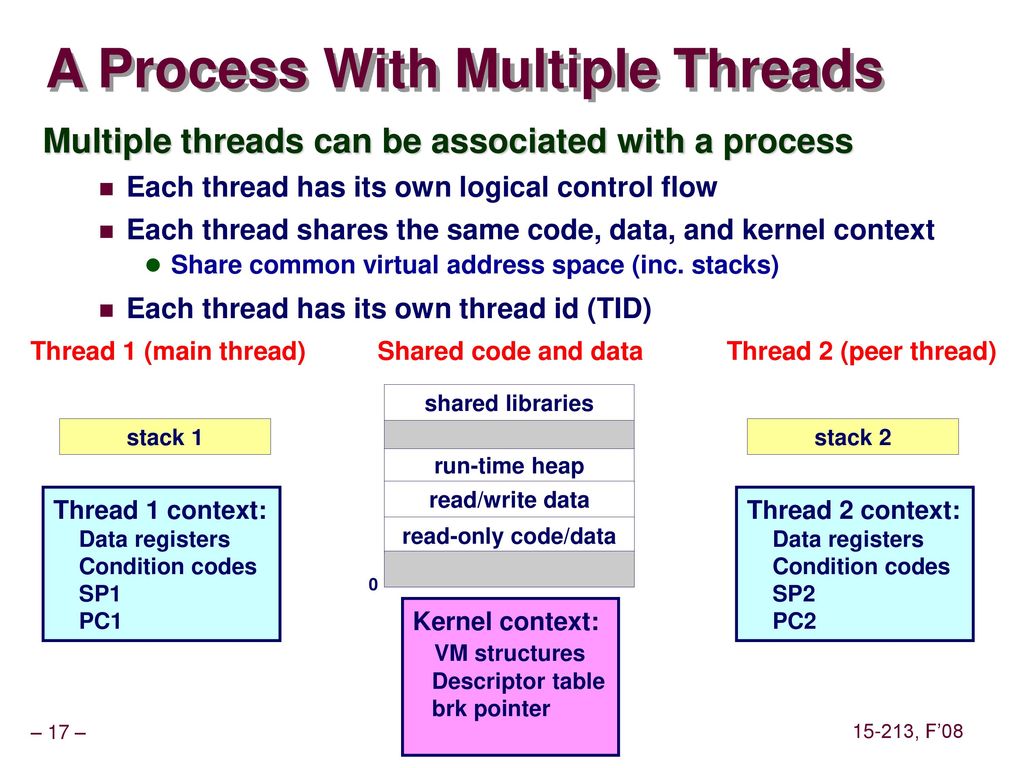
**what is concurrency ?**

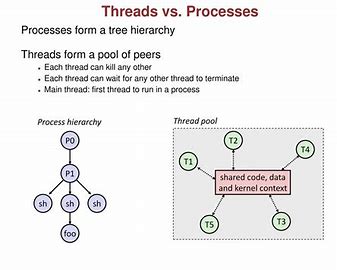
Refers to the ability of a system to perform multiple tasks(functions) simultaneously or handle multiple operation within overlapping time frames.

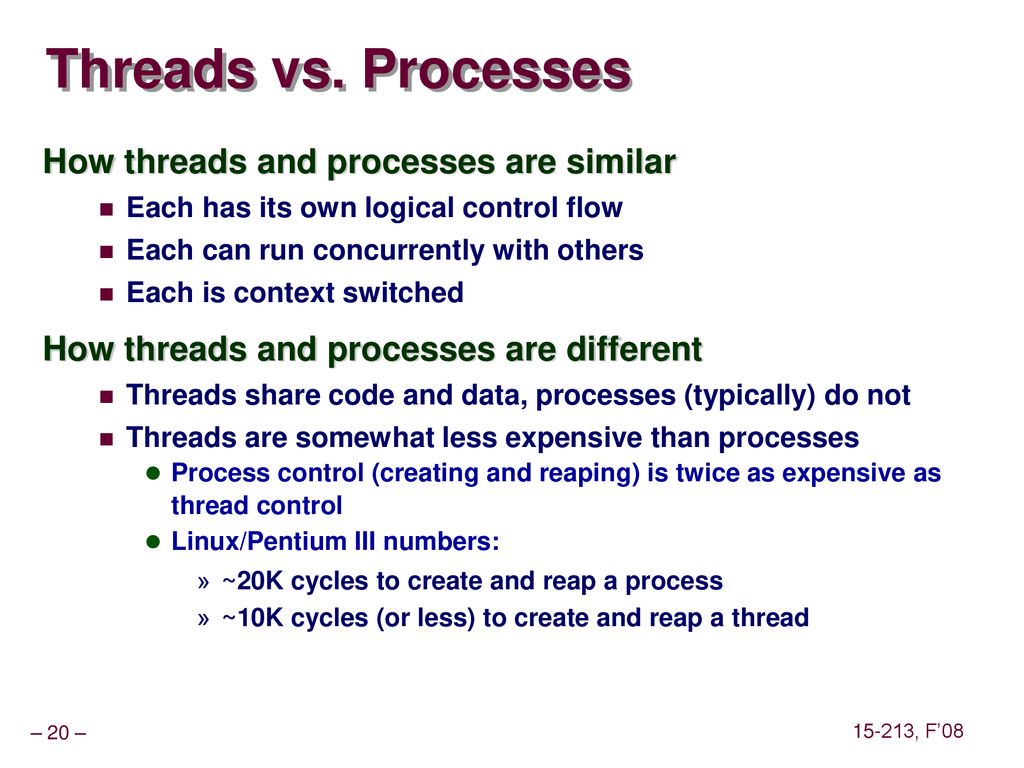
Necessity and benefits of concurrency:

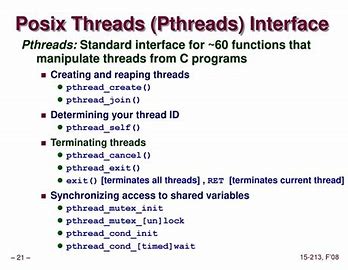
* Responsiveness
* Utilizing multi core processors
* Efficient Resource Utilization
* Scalability:Efficiently and effectivly
* Improved Throughput











**What is Multi Threading?**

**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.

**What is MultiProcessing?**

**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.

#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?

**Why we are supposed to destroy mutex?**

Mutex destruction is necessary to free system resources and prevent memory leaks. It ensures proper cleanup and avoids undefined behavior when the mutex is no longer needed.