# ASSIGNMENT NO.:2

## TITLE: Thread management using pthread library.

- **OBJECTIVE:**
  1. Study how to use POSIX threads in Linux.
  2. Implement multithreading in Linux using C.
  3. Implement POSIX thread function for thread create,join,and exit.
- **PROMBLEM STATEMENT:**

  Implement matrix multiplication using multithreading.Application should have pthread_create,pthread_join,pthread_exit.In the program every thread must return the value and must be collected in pthread_join in the main function.Final sum of row column multiplication must be done by main thread.
- **THEORY:**
  - **THREAD:**

"A process with two **thread** of execution,running on one processor. In computer science,a **thread** of execution is the smallest sequence of programmed instructions that can be managed independently by scheduler,which is typically a part of the **Operating system.**"

A thread is a single sequence stream within in a process. Because threads have some of the properties of processes,they are sometimes called lightweight processes.In a process,threads allow multiple executions of streams.

- **THREAD VS PROCESS:**

The process and threads are independent sequences of execution,the typical difference is that threads run in separate memory spaces.

A process has a self contained execution environment that means it has a complete,private set of basic run times resources purticulary each process has its own memory space.Thread exist within a process and every process has at least one thread.

Each process provides the resources needed to execute a program.Each process is started with a single thread,known as the primary thread.

Processes are heavily dependent on system resources available while threads require minimal amounts of resources,so a process is considered as heavyweight while a thread is termed as a lightweight process.

## ➢ PTHREAD:

"POSIX Threads,usually referred to as **Pthreads**,is an execution model that exists independently from a languge,as well as a parallel execution model.It allow a program to control multiple different flows of work that overlap in time."

The thread interfaces described in this section are based on a subset of the application programming are based on a subset of the application programming interfaces (APIs) define in the POSIX standard(ANSI/IEEE Standard 1003.1,1996 Edition OR ISO/IEC 9945-1:1996)and the single UNIX Specification,version 2,1997.

## ➢ FUNCTIONS OF PTHREAD:

### 1. Pthread_create:

**Intpthread_create(pthread_t*thread,cont pthread_attr_t*attr,void*(  *start_routine)(void),void*arg);**

**ARGUMENT:**

The **pthread_create**() function starts a new thread in the calling process.  The new thread starts execution by invoking

start_routine(); arg is passed as the sole argument of

start_routine().

The new thread terminates in one of the following ways:

* It calls pthread_exit(3), specifying an exit status value that is

available to another thread in the same process that calls

pthread_join(3).

* It returns from start_routine().  This is equivalent to calling

pthread_exit(3) with the value supplied in the return statement.

* It is canceled (see pthread_cancel(3)).

* Any of the threads in the process calls exit(3), or the main thread performs a return from main(). This causes the termination of all threads in the process.

The attr argument points to a pthread_attr_t structure whose contents are used at thread creation time to determine attributes for the new thread; this structure is initialized using pthread_attr_init(3) and related functions. If attr is NULL, then the thread is created with default attributes.

Before returning, a successful call to **pthread_create**() stores the ID of the new thread in the buffer pointed to by thread; this identifier is used to refer to the thread in subsequent calls to other pthreads functions.

The new thread inherits a copy of the creating thread's signal mask (pthread_sigmask(3)). The set of pending signals for the new thread is empty (sigpending(2)). The new thread does not inherit the creating thread's alternate signal stack (sigaltstack(2)).

The new thread inherits the calling thread's floating-point environment (fenv(3)).

The initial value of the new thread's CPU-time clock is 0 (see pthread_getcpuclockid(3)).

## 2. Pthread_exit:

**Void pthread_exit(void *retval);**

## ARGUMENTS:

Retval-Return value of thread.

This routine kills the thread.the pthread_exit function never returns.If the thread is not detached,the thread id and return value may be explained from another thread by using pthread_join.

## 3.Joins:

A join is performed when one wants to wait for a thread to finish.A thread calling routine may launch multiple threads then wait for them to finish to get the results.One wait for the completion of the thread with a join.

**Int pthread_join(pthread_t thread,void,void**retval);**

The pthread_join() function waits for the thread specified by thread specified by thread to terminate.If that thread has already terminated,then pthread_join() returns immediately.The thread specified by thread must be joinable.

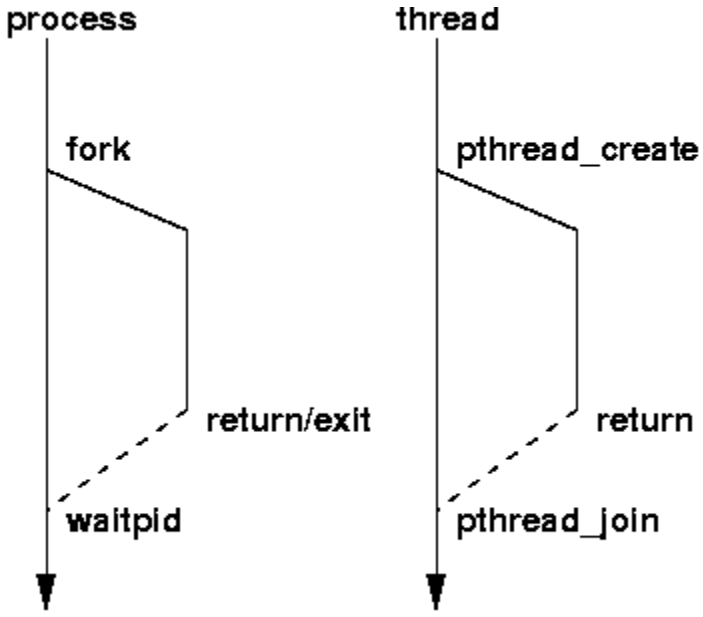


**Fig:-Pthread function implementation**

- **CONCLUSION:-**

Thus we have implemented matrix multiplication using multithreading.