

LAB-10

Aim: Thread creation and Termination. Synchronization using mutex lock and unlock. (Use of `pthread_create`, `pthread_join`, `pthread_mutex_lock` and `pthread_mutex_unlock` library functions of Pthread library).

Explanation:

Pthread_create:

```
#include <pthread.h>
```

```
int pthread_create(pthread_t *thread, const pthread_attr_t *attr,  
                  void *(*start_routine) (void *), void *arg);
```

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 `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** 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.

On success, **pthread_create()** returns 0; on error, it returns an error number, and the contents of `thread` are undefined.

Pthread_join:

```
#include <pthread.h>
```

```
int pthread_join(pthread_t thread, void **retval);
```

Compile and link with `-pthread`.

The **pthread_join()** function waits for the 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.

On success, **pthread_join()** returns 0; on error, it returns an error number.

Pthread mutex lock:

#include <pthread.h>

int pthread_mutex_lock(pthread_mutex_t *mutex);

The mutex object referenced by *mutex* shall be locked by calling *pthread_mutex_lock()*. If the mutex is already locked, the calling thread shall block until the mutex becomes available. This operation shall return with the mutex object referenced by *mutex* in the locked state with the calling thread as its owner.

If successful, the *pthread_mutex_lock()* and *pthread_mutex_unlock()* functions shall return zero; otherwise, an error number shall be returned to indicate the error.

Pthread mutex unlock:

#include <pthread.h>

int pthread_mutex_unlock(pthread_mutex_t *mutex);

The *pthread_mutex_unlock()* function shall release the mutex object referenced by *mutex*.

If successful, the *pthread_mutex_lock()* and *pthread_mutex_unlock()* functions shall return zero; otherwise, an error number shall be returned to indicate the error.

Programs:

1. Write a program to create a thread using pthread_create.

Solution logic:

The program should create a thread and print some message using the threaded function.

2. Write a program to pass a character string to the threaded function.

Solution logic:

The program should create a thread and pass a character string to threaded function. The threaded function should print the given string.

3. Write a program to implement simple calculator using threads.

Solution logic:

The program should create four threads for four calculator operations and print the results.