------------------------Souvik Kr. Parui--------------------------------------

POSIX Threads

pthread: declare, create, threaded function, and thread join

☞ One method of achieving parallelism is for multiple processes
   to cooperate and synchronize through shared memory.

☞ An alternative approach uses multiple threads of execution
  in a single address space.

☞ Thread: A single sequence of instructions executed within a process.
           Threads in the same process share the same memory space and resources.

Thread Creation and Execution: In a multithreaded program:

Threads are created as independent units of execution.
Each thread runs concurrently with the main thread and other threads.
Threads share the process memory but execute independently.

Steps:

☞ The program begins execution in the main() function.

☞ A thread is created using pthread_create.

☞ Each thread starts executing its function independently.

☞ The main() thread can wait for threads to finish using pthread_join or continue independently.

Creating a Thread

```
#include <pthread.h>
int pthread_create(pthread_t *restrict thread,const pthread_attr_t *restrict
attr,void *(*start_routine)(void *),void *restrict arg);
```

pthread_create(): The pthread_create function creates a thread.The POSIX pthread_create automatically makes the thread runnable without requiring a separate start operation.

thread: The parameter of pthread_create points to the ID of the newly created thread.

attr: The attr parameter represents an attribute object that encapsulates the attributes of a thread. If attr is NULL, the new thread has the default attributes.

Thread Joining

```
#include <pthread.h>
int pthread_join(pthread_t thread, void **value_ptr);
```

thread: The pthread_join function suspends the calling thread until the target thread, specified by the first parameter, terminates.

value_ptr: The value_ptr parameter provides a location for a pointer to the return status that the target thread passes to pthread_exit or return.

Thread Exit

```
-----------------------
#include <pthread.h>
void pthread_exit(void *value_ptr);
```

pthread_exit: A call to exit causes the entire process to terminate; a call to pthread_exit causes only the calling thread to terminate.
value_ptr: The value_ptr value is available to a successful pthread_join. Put NULL not to return any status.

Example : Basic Thread Creation and Synchronization

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

// Thread function

void *print_message(void *arg)
{
    char *message = (char *)arg;

    printf("%s\n", message);

    pthread_exit(NULL);
}

int main() {
    pthread_t thread1, thread2;

    pthread_create(&thread1, NULL, print_message, "Hello from Thread 1");
    pthread_create(&thread2, NULL, print_message, "Hello from Thread 2");

    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);

    return 0;
}
```

&thread1: A pointer to the thread identifier variable.
NULL: Uses the default thread attributes.
print_message: The function the thread will execute.
"Hello from Thread 1": The argument passed to the print_message function.

pthread_join: Wait for threads to finish. Blocks the main thread until the specified thread finishes.

Passing Parameters to Threads

Passing an Integer Array :

```
void *arraypass(void *arg);
int main()
{
int arr[]={10,20,30,40};
pthread_t tid;
pthread_create(&tid,NULL,arraypass,(void *)arr);
pthread_join(tid,NULL);
printf("Bye....main thread\n");
return 0;
}
void *arraypass(void *arg)
{
int *ar,i;
ar=(int *)arg;
for(i=0;i<4;i++){
printf("Received:arr[%d]=%d\n",i,*(ar+i));
/*or printf("Received:arr[%d]=%d\n",i,ar[i]); */
}
pthread_exit(NULL);
}
```

Passing a String:

```
#include<stdio.h>
#include<pthread.h>
void *stringpass(void *arg);
int main(){
char *msg="ITER";
pthread_t t;
pthread_create(&t,NULL,passint,(void *)msg);
pthread_join(t,NULL);
return 0;
}
void *stringpass(void *arg)
{
char *str;
str=(char *)(arg);
printf("String received=%s\n",str);
pthread_exit(NULL);
}
```

POSIX	function Description
pthread_create	Create a thread
pthread_join	Wait for a thread
pthread_detach	Set thread to release resources
pthread_exit	Exit a thread without exiting process
pthread_self	Find out own thread ID
pthread_equal	Test two thread IDs for equality
pthread_kill	Send a signal to a thread
pthread_cancel	Terminate another thread

☞ pthread_create: Starts a new thread and runs the specified function.
☞ pthread_join: Waits for a thread to complete and optionally retrieves its result.
☞ pthread_exit: Terminates a thread and optionally provides a return value.
☞ Concurrency: Threads run independently, and their order of execution is not guaranteed.

Question : Write a C program that demonstrates the use of POSIX threads (pthread), including declaration, creation, a threaded function, and joining threads:

☞ Main thread creates 3 threads.
☞ Each thread prints a message and computes a result.
☞ Main thread waits for each thread to finish and retrieves its result.
☞ Program prints results and exits.

Output :

```
Thread 1 created.
Thread 2 created.
Thread 3 created.
Hello from thread 1!
Hello from thread 2!
Hello from thread 3!
Thread 1 finished with result: 2
Thread 2 finished with result: 4
Thread 3 finished with result: 6
All threads have completed.
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>


// Threaded function that will be executed by the threads

void *thread_function(void *arg)
{
    int thread_num = *((int *)arg);    // Retrieve the thread number from the argument
    printf("Hello from thread %d!\n", thread_num);

    // Perform some task (e.g., increment a number)

    int result = thread_num * 2;   // Example computation
    pthread_exit((void *)result);  // Return the result to the main thread
}


int main()
{
    pthread_t threads[3];      // Declare an array of thread identifiers

    int thread_args[3];        // Arguments to be passed to the threaded function

    int i;
    void *retval;              // Pointer to hold return value from a thread
    // Create 3 threads

    for (i = 0; i < 3; i++)
    {
        thread_args[i] = i + 1; // Assign thread number

        if (pthread_create(&threads[i], NULL, thread_function, (void
*)&thread_args[i]) != 0)
         {
            perror("Error creating thread");
            exit(1);
          }
        printf("Thread %d created.\n", i + 1);
    }

    // A loop to wait for all 3 threads to finish. Join threads and retrieve their
results

    for (i = 0; i < 3; i++) {
        if (pthread_join(threads[i], &retval) != 0) {  // Blocks the main thread
until the thread threads[i] finishes. &retval: Captures the value returned by the
thread.
            perror("Error joining thread");
            exit(1);
        }
        printf("Thread %d finished with result: %ld\n", i + 1, (long)retval);
    }

    printf("All threads have completed.\n");
    return 0;
}
```

Explain :

void *thread_function(void *arg): executed by each thread. It takes a void * argument that allows passing any type of data.

int thread_num = *((int *)arg);: cast to an integer pointer and dereferenced to retrieve the value passed during thread creation (thread number).

thread_t threads[3];: Declares an array to hold thread identifiers for 3 threads. Each thread has a unique pthread_t value.

int thread_args[3];: An array to store arguments for each thread (e.g., thread numbers: 1, 2, 3).

void *retval;: A pointer to store the value returned by a thread after it finishes.

pthread_create(&threads[i], NULL, thread_function, (void *)&thread_args[i]): Creates a thread.

&threads[i]: Pointer to store the thread identifier.
NULL: Default thread attributes.

```
thread_function: The function to execute in the thread.
(void *)&thread_args[i]: Argument passed to the thread.
```