How to program a thread (pthread)?

1) « pthread_create » : create the thread.

*pthread_create (pthread_t *thread, pthread_attr_t *attr, void *(*start_routine)(void *), void *arg)* 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().

- **thread:** An identifier for the new thread returned by the subroutine. This is a pointer to pthread_t structure. When a thread is created, an identifier is written to the memory location to which this variable points. This identifier enables us to refer to the thread.
- **attr:** An attribute object that may be used to set thread attributes. We can specify a thread attributes object, or NULL for the default values.

- **start_routine:** The routine that the thread will execute once it is created.
  `void *(*start_routine)(void *)`

  We should pass the address of a function taking a pointer to void as a parameter and the function will return a pointer to void. So, we can pass any type of single argument and return a pointer to any type. While using fork() causes execution to continue in the same location with a different return code, using a new thread explicitly provides a pointer to a function where the new thread should start executing.

- **arg:** A single argument that may be passed to start_routine. It must be passed as a void pointer. NULL may be used if no argument is to be passed.

Example :

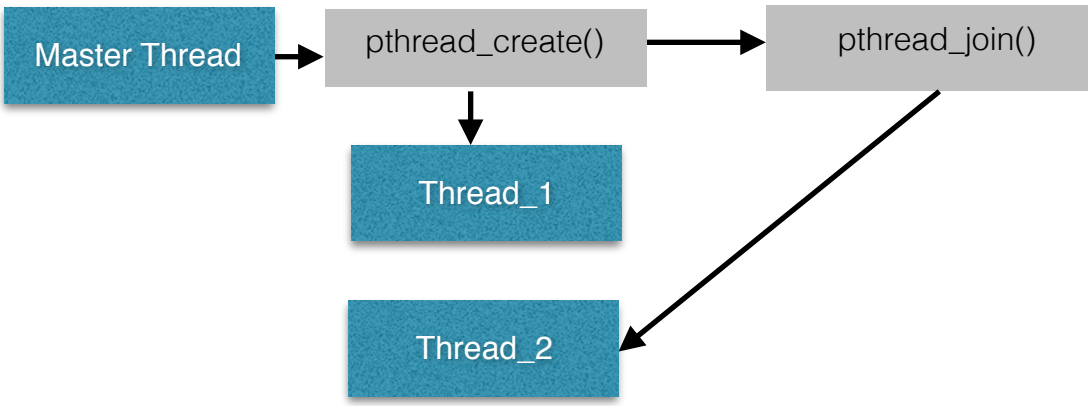
```
void *FunctionHi(void *threadid)
{
   cout << "Hi !!" << endl;
  pthread_exit(NULL);
}

int main(){
        pthread_t THB;

        pthread_create(&THB, NULL, FunctionHi, NULL);
        pthread_exit(NULL);

}
```

pthread_join
"Joining" is one way to accomplish synchronization between threads. For example:



The pthread_join() subroutine blocks the calling thread until the specified threadid thread terminates.
The programmer is able to obtain the target thread's termination return status if it was specified in the target thread's call to pthread_exit().
A joining thread can match one pthread_join() call. It is a logical error to attempt multiple joins on the same thread.
Two other synchronization methods, Mutex and condition variables, will be discussed later.

Example

```
int main (){
  pthread_t thread_one;
  pthread_create (&thread_one, NULL, FunctionHi, NULL);
  pthread_join (thread_one, NULL);
  cout << "le thread_ one :::: ok !! fin " << endl;
  return 0;
}
```