Let's look at an example program exercising the above pthread functions:

[?](https://randu.org/tutorials/threads/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | #include <stdio.h>  #include <stdlib.h>  #include <pthread.h>    #define NUM\_THREADS 2    /\* create thread argument struct for thr\_func() \*/  typedef struct \_thread\_data\_t {    int tid;    double stuff;  } thread\_data\_t;    /\* thread function \*/  void \*thr\_func(void \*arg) {    thread\_data\_t \*data = (thread\_data\_t \*)arg;      printf("hello from thr\_func, thread id: %d\n", data->tid);      pthread\_exit(NULL);  }    int main(int argc, char \*\*argv) {    pthread\_t thr[NUM\_THREADS];    int i, rc;    /\* create a thread\_data\_t argument array \*/    thread\_data\_t thr\_data[NUM\_THREADS];      /\* create threads \*/    for (i = 0; i < NUM\_THREADS; ++i) {      thr\_data[i].tid = i;      if ((rc = pthread\_create(&thr[i], NULL, thr\_func, &thr\_data[i]))) {        fprintf(stderr, "error: pthread\_create, rc: %d\n", rc);        return EXIT\_FAILURE;      }    }    /\* block until all threads complete \*/    for (i = 0; i < NUM\_THREADS; ++i) {      pthread\_join(thr[i], NULL);    }      return EXIT\_SUCCESS;  } |

This program creates NUM\_THREADS threads and prints their respective user-assigned thread id. The first thing to notice is the call to pthread\_create() in the main function. The syntax of the third and fourth argument are particularly important. Notice that the thr\_func is the name of the thread function, while the fourth argument is the argument passed to said function. Here we are passing a thread function argument that we created as a thread\_data\_t struct. Of course, you can pass simple data types as pointers if that is all that is needed, or NULL if no arguments are required. However, it is good practice to be able to pass arguments of arbitrary type and size, and is thus illustrated for this purpose.   
  
A few things to mention:

* Make sure you check the return values for all important functions.
* The second argument to pthread\_create() is NULL indicating to create a thread with default attributes. The defaults vary depend upon the system and pthread implementation.
* Notice that we have broken apart the pthread\_join() from the pthread\_create(). Why is it that you should not integrate the pthread\_join() in to the thread creation loop?
* Although not explicitly required to call pthread\_exit() at the end of the thread function, it is good practice to do so, as you may have the need to return some arbitrary data back to the caller via pthread\_join().