

# PThreads

PThreads have type `pthread_t`.

There are five basic thread functions:

`pthread_create` creates a new thread within a process.

`pthread_join` waits for another thread to terminate.

`pthread_self` returns the id of the calling thread.

`pthread_equal` compares thread ids to see if they refer to the same thread.

`pthread_exit` terminates the currently running thread.

```
#include <pthread.h>

int pthread_create(pthread_t *tid, const pthread_attr_t *attr,
                  void *(*my_function)(void *), void *arg);
```

On creation of a new thread, its thread id is returned to the calling thread.

Each thread has *attributes* (priority, stack size, etc.).

Normally, we specify the default attributes by passing the argument NULL.

When we create a new thread, we specify a function `my_function` for it to execute.

The thread begins by executing `my_function`.

The thread ends by returning from `my_function` or by calling `pthread_exit`.

`arg` is an argument, often NULL, passed to `my_function`.

Example:

```
pthread_t thread1
```

```
pthread_create(&thread1, NULL, &function1, NULL)
```

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

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

If there are no other threads joined with `thread`, then `pthread_join` suspends the calling thread and waits for `thread` to terminate.

Otherwise, the result is undefined.

If the *join* is successful, `pthread_join` will return 0.

If `status` is not `NULL`, then it will point to the return value of `thread`.

On failure, `pthread_join` will return an error number.

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

```
pthread_t pthread_self();
```

The `pthread_self` function returns the thread id of the calling thread.

This allows the calling to find out its own id.

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

```
int pthread_equal(pthread_t thread1, pthread_t thread2);
```

This function compares the thread ids `thread1` and `thread2`, and returns a non-

zero value if they refer to the same thread.

Otherwise, 0 is returned.

```
#include <pthread.h>

void pthread_exit(void *status);
```

The calling thread is terminated.

The value of `status` is returned to the thread that successfully joins the terminating thread.

If the exiting thread is the last thread to terminate, then the parent process terminates.

There are two other ways for a thread to terminate.

1. The function `my_function` returns.

In that case, its return value takes the place of `status`.

The function's return value is the exit status of the function (since it returns a void pointer).

2. `main()` exits or any thread calls `exit`.

In this case, `pthread_exit` is called implicitly for all threads.

## Example: The ”Hello World” Program

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

void *print_message(void *ptr);

main()
{
    pthread_t thread1, thread2;
    char *message1 = "Hello";
    char *message2 = "World";

    pthread_create(&thread1, NULL, &print_message, &message1);
    pthread_create(&thread2, NULL, &print_message, &message2);

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

    return 0;
}

void *print_message(void *ptr)
{
    char *message;
    message = (char *) ptr;
    printf("%s", message);
}
```

There are some problems with this example.

What are they?