**Creating a Simple Thread in Linux**

**Understanding the Basics**

Before we dive into the code, let's understand the key concept and function:

* **pthread library:** Provides functions for creating and managing threads in Linux.
* **pthread\_create:** The primary function used to create a new thread.

**Code Implementation**

Here's a simple C program that creates a thread:

C

#include <stdio.h>

#include <pthread.h>

void \*thread\_function(void \*arg) {

printf("Hello from child thread!\n");

return NULL;

}

int main() {

pthread\_t thread\_id;

int rc = pthread\_create(&thread\_id, NULL, thread\_function, NULL);

if (rc) {

printf("Error: pthread\_create failed with code %d\n", rc);

exit(EXIT\_FAILURE);

}

printf("Hello from main thread!\n");

pthread\_exit(NULL);

return 0;

}

Use code [with caution.](/faq#coding)

**Explanation**

1. **Include necessary headers:** stdio.h for input/output and pthread.h for thread-related functions.
2. **Define the thread function:** thread\_function is the entry point for the new thread. It simply prints a message.
3. **Main function:**
   * Declares a variable thread\_id to store the thread ID.
   * Calls pthread\_create to create a new thread. The arguments are:
     + &thread\_id: Pointer to store the thread ID.
     + NULL: Default thread attributes.
     + thread\_function: Function to be executed by the thread.
     + NULL: No arguments passed to the thread function.
   * Checks the return value of pthread\_create for errors.
   * Prints a message from the main thread.
   * Terminates the main thread using pthread\_exit.

**Compiling and Running**

To compile and run this code:

1. Save the code as a .c file (e.g., thread.c).
2. Open a terminal and navigate to the directory containing the file.
3. Compile the code using the GCC compiler:

Bash

**gcc thread.c -o thread -lpthread**

Use code [with caution.](/faq#coding)

The -lpthread flag links the pthread library.

1. Run the executable:

Bash

./thread

Use code [with caution.](/faq#coding)

The output should be:

Hello from child thread!

Hello from main thread!

**Important Notes**

* The order of output is not guaranteed. The child thread might print its message before or after the main thread.
* For more complex scenarios, you might need to use thread attributes, synchronization mechanisms (like mutexes, semaphores), and thread joining.
* Always check the return values of pthread functions for error handling.

This is a basic example of thread creation. You can expand on this by creating multiple threads, passing arguments to thread functions, and using synchronization mechanisms as needed.