# Thread Creation - Class Participation

## Write a C/C++ program that creates two threads. Each thread should print its thread ID.

#include<pthread.h>

#include<stdio.h>

void\* print\_thread\_id(void\* arg){

printf("Thread ID: %ld\n", pthread\_self());

return NULL;

}

int main(){

pthread\_t thread1,thread2;

pthread\_create(&thread1, NULL, print\_thread\_id, NULL);

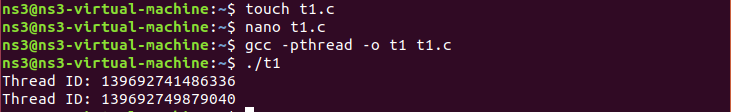
pthread\_create(&thread2, NULL, print\_thread\_id, NULL);

pthread\_join(thread1, NULL);

pthread\_join(thread2, NULL);

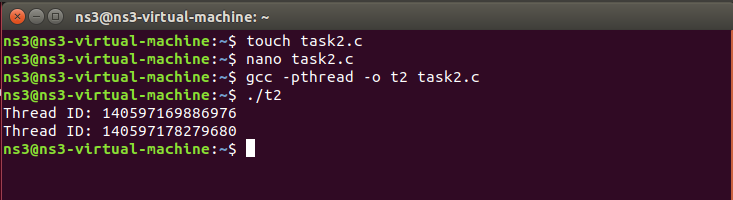
return 0;

}



## What will be the output of the following code? Explain your answer.





This program creates two threads using **pthread\_create()** function and prints their thread IDs using **pthread\_self().** **pthread\_join()** ensures that thread 1 finishes before thread 2 starts.

## How can you pass multiple arguments to a thread function in C/C++? Show an example of passing two integers to a thread and printing their sum.

#include <stdio.h>

#include <pthread.h>

// Struct to hold the arguments

typedef struct {

int a;

int b;

} thread\_args;

void\* sum(void\* arguments) {

thread\_args\* args = (thread\_args\*)arguments;

int result = args->a + args->b;

printf("Sum: %d\n", result);

return NULL;

}

int main() {

pthread\_t thread;

thread\_args args;

args.a = 5;

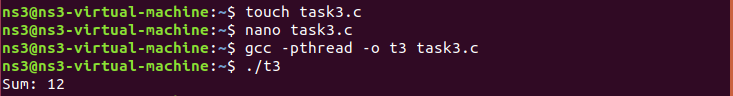
args.b = 7;

pthread\_create(&thread, NULL, sum, &args);

pthread\_join(thread, NULL);

return 0;

}



## In multithreaded programming, what is the purpose of using pthread\_join()? Write a program that demonstrates the difference between using pthread\_join() and not using it.

The **pthread\_join()** function ensures that the main thread waits for the created thread to finish execution before continuing. Without it, the main thread could finish first, leading to termination of the program before the created threads complete their work.

### Without using **thread\_join()**

#include <stdio.h>

#include <pthread.h>

void\* print\_message(void\* message) {

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

return NULL;

}

int main() {

pthread\_t thread;

pthread\_create(&thread, NULL, print\_message, "Hello! Thread created.");

printf("Main thread done\n");

return 0;

}



The program finished before creating the threads completely. Which lead to incomplete results.

### With using **thread\_join()**

#include <stdio.h>

#include <pthread.h>

void\* print\_message(void\* message) {

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

return NULL;

}

int main() {

pthread\_t thread;

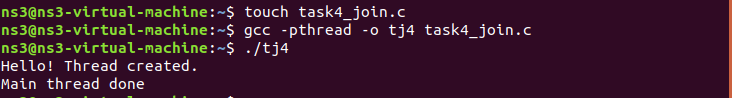
pthread\_create(&thread, NULL, print\_message, "Hello! Thread created.");

pthread\_join(thread, NULL);

printf("Main thread done\n");

return 0;

}



**pthread\_join()** ensures that the message is printed before the main thread exits. In this output we have complete and desired results.

## Modify the following code so that the threads execute in parallel but in a specific order (e.g., thread 1 executes first, then thread 2):



#include <pthread.h>

#include <stdio.h>

void\* print\_msg1(void\* arg) {

printf("Message from thread 1\n");

return NULL;

}

void\* print\_msg2(void\* arg) {

printf("Message from thread 2\n");

return NULL;

}

int main() {

pthread\_t threads[2];

pthread\_create(&threads[0], NULL, print\_msg1, NULL);

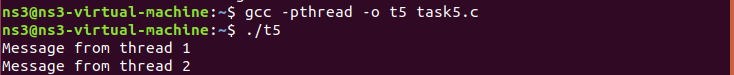
pthread\_join(threads[0], NULL);

pthread\_create(&threads[1], NULL, print\_msg2, NULL);

pthread\_join(threads[1], NULL);

return 0;

}



## What is the purpose of pthread\_attr\_t when creating a thread, and how can you use it to set thread attributes (e.g., stack size)? Provide a small example.

### Purpose:

**pthread\_attr\_t** is used to define attributes for threads, such as stack size, scheduling policy, etc.

### Code:

#include <stdio.h>

#include <pthread.h>

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

printf("Thread with custom stack size\n");

return NULL;

}

int main() {

pthread\_t thread;

pthread\_attr\_t attr;

size\_t stack\_size = 1024 \* 1024;

pthread\_attr\_init(&attr);

pthread\_attr\_setstacksize(&attr, stack\_size);

pthread\_create(&thread, &attr, thread\_func, NULL);

pthread\_attr\_destroy(&attr);

pthread\_join(thread, NULL);

return 0;

}

