Imprimir en PDF

threadsC.ipynb

Contenido


```
# Verificar la versión de GCC
!gcc --version
```

```
gcc (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Sin utilizar threads

```
%%writefile hellothreadsC.c
#include <stdio.h>
int main( ){
  printf("Hola, aún no estoy usando los hilos con C");
  return 0;
}
```

```
Overwriting hellothreadsC.c
```

```
!gcc hellothreadsC.c -o hellothreadsC
!./hellothreadsC
```

```
Hola, aún no estoy usando los hilos con C
```

Algo de punteros

```
%%writefile punteros.c
#include <stdio.h>
int main() {
    int num = 42;        // Declaración de una variable entera
    int *p;              // Declaración de un puntero a entero
    p = &num;            // Asignación de la dirección de memoria de 'num' al p

    printf("Valor de num: %d\n", num);             // Imprime el valor de 'num'
    printf("Dirección de num: %p\n", &num);        // Imprime la dirección de m
    printf("Valor de p: %p\n", p);                 // Imprime el valor de 'p',
    printf("Valor apuntado por p: %d\n", *p);      // Imprime el valor de la va

    return 0;
}
```

```
Overwriting punteros.c
```

```
!gcc punteros.c -o punteros
!./punteros
```

```
Valor de num: 42
Dirección de num: 0x7ffff0583dcc
Valor de p: 0x7ffff0583dcc
Valor apuntado por p: 42
```

Uso de Hilos en C: definición de hilos

```
%%writefile hellothreadsC.c
#include <stdio.h>
#include <pthread.h>

#define NUM_THREADS 5 // Definimos una constante para el número de hilos a crea

// Esta función será ejecutada por cada hilo creado.
// Imprime un mensaje de saludo y el identificador del hilo.
void *print_hello(void *thread_id) {
    int id = *((int*) thread_id);
    printf("Hola, soy el hilo %d\n", id);
    pthread_exit(NULL); // Termina la ejecución del hilo.
}

int main() {
    pthread_t threads[NUM_THREADS]; // Creamos un arreglo de hilos.
    int thread_ids[NUM_THREADS]; // Creamos un arreglo de identificadores de hi

    // Creamos NUM_THREADS hilos y los almacenamos en el arreglo threads.
    // Cada hilo ejecuta la función print_hello y le pasamos como argumento el
    for (int i = 0; i < NUM_THREADS; i++) {
        thread_ids[i] = i; // Almacenamos el identificador del hilo en el arreg
        pthread_create(&threads[i], NULL, print_hello, (void*) &thread_ids[i]);
    }

    printf("Hola soy el hilo principal\n"); // Mensaje de saludo del hilo princ
    pthread_exit(NULL); // Termina la ejecución del hilo principal.
    return 0;
}
```

```
Overwriting hellothreadsC.c
```

```
# Importante agregar: "-pthread" para el correcto funcionamiento. Esta opción l
!gcc -pthread  hellothreadsC.c -o hellothreadsC
!./hellothreadsC
```

```
Hola, soy el hilo 0
Hola, soy el hilo 2
Hola, soy el hilo 1
Hola, soy el hilo 3
Hola soy el hilo principal
Hola, soy el hilo 4
```

Uso de Hilos en C: uso del Join

```
%%writefile hellothreadsC.c
#include <stdio.h>
#include <pthread.h>

#define NUM_THREADS 5 // Definimos una constante para el número de hilos a crea

// Esta función será ejecutada por cada hilo creado.
// Imprime un mensaje de saludo y el identificador del hilo.
void *print_hello(void *thread_id) {
    int id = *((int*) thread_id);
    printf("Hola, soy el hilo %d\n", id);
    pthread_exit(NULL); // Termina la ejecución del hilo.
}

int main() {
    pthread_t threads[NUM_THREADS]; // Creamos un arreglo de hilos.
    int thread_ids[NUM_THREADS]; // Creamos un arreglo de identificadores de hi

    // Creamos NUM_THREADS hilos y los almacenamos en el arreglo threads.
    // Cada hilo ejecuta la función print_hello y le pasamos como argumento el
    for (int i = 0; i < NUM_THREADS; i++) {
        thread_ids[i] = i; // Almacenamos el identificador del hilo en el arreg
        pthread_create(&threads[i], NULL, print_hello, (void*) &thread_ids[i]);
    }

    // Esperamos a que cada hilo termine su ejecución antes de que el hilo prin
    for (int i = 0; i < NUM_THREADS; i++) {
        pthread_join(threads[i], NULL);
    }

    printf("Hola soy el hilo principal\n"); // Mensaje de saludo del hilo princ
    pthread_exit(NULL); // Termina la ejecución del hilo principal.
    return 0;
}
```

```
Overwriting hellothreadsC.c
```

```
# Importante agregar: "-pthread" para el correcto funcionamiento. Esta opción l
!gcc -pthread  hellothreadsC.c -o hellothreadsC
!./hellothreadsC
```

```
Hola, soy el hilo 0
Hola, soy el hilo 1
Hola, soy el hilo 2
Hola, soy el hilo 3
Hola, soy el hilo 4
Hola soy el hilo principal
```

Uso de Hilos en C: uso de semáforos

```
%%writefile hellothreadsC.c
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <unistd.h>

#define NUM_THREADS 5 // Definimos una constante para el número de hilos a crea
sem_t semaphore; // Declaramos la variable semáforo.

// Esta función será ejecutada por cada hilo creado.
// Imprime un mensaje de saludo y el identificador del hilo.
void *print_hello(void *thread_id) {
    int id = *((int*) thread_id);
    sem_wait(&semaphore); // Esperamos a que el semáforo se libere.
    printf("Hola, soy el hilo %d\n", id);
    sleep(2);
    sem_post(&semaphore); // Liberamos el semáforo.
    pthread_exit(NULL); // Termina la ejecución del hilo.
}

int main() {
    pthread_t threads[NUM_THREADS]; // Creamos un arreglo de hilos.
    int thread_ids[NUM_THREADS]; // Creamos un arreglo de identificadores de hi
    sem_init(&semaphore, 0, 2); // Inicializamos la variable semáforo.

    // Creamos NUM_THREADS hilos y los almacenamos en el arreglo threads.
    // Cada hilo ejecuta la función print_hello y le pasamos como argumento el
    for (int i = 0; i < NUM_THREADS; i++) {
        thread_ids[i] = i; // Almacenamos el identificador del hilo en el arreg
        pthread_create(&threads[i], NULL, print_hello, (void*) &thread_ids[i]);
    }

    // Esperamos a que cada hilo termine su ejecución antes de que el hilo prin
    for (int i = 0; i < NUM_THREADS; i++) {
        pthread_join(threads[i], NULL);
    }

    printf("Hola soy el hilo principal\n"); // Mensaje de saludo del hilo princ
    pthread_exit(NULL); // Termina la ejecución del hilo principal.
    sem_destroy(&semaphore); // Destruimos la variable semáforo.
    return 0;
}
```

```
Overwriting hellothreadsC.c
```

```
# Importante agregar: "-pthread" para el correcto funcionamiento. Esta opción l
!gcc -pthread  hellothreadsC.c -o hellothreadsC
!./hellothreadsC
```

```
Hola, soy el hilo 0
Hola, soy el hilo 1
Hola, soy el hilo 2
Hola, soy el hilo 3
Hola, soy el hilo 4
Hola soy el hilo principal
```