PThreads - Threads em C

Engenharia de Computação
Pontifícia Universidade Católica de Campinas

Prof. Dr. Denis M. L. Martins

Criação de Thread

A thread é criada com `pthread_create` e esperada com `pthread_join`.

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

void *thread_function(void *arg) {
    printf("Olá do thread!\n");
    pthread_exit(0);
}

int main() {
    pthread_t thread1;
    pthread_create(&thread1, NULL, thread_function, NULL);
    pthread_join(thread1, NULL);
    return 0;
}
```

O que acontece se o `printf` em `thread_function` for chamado por várias threads simultaneamente?

Passagem de Argumentos

O nome "Ada" é passado como um argumento da função `thread_function`. É importante o cast do `void *` para `char *` para acessar o argumento corretamente.

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

void *thread_function(void *arg) {
    char *name = (char *)arg; // Cast do argumento para char*
    printf("Olá, %s!\n", name);
    pthread_exit(0);
}

int main() {
    pthread_t thread1;
    pthread_create(&thread1, NULL, thread_function, "Ada"); // Passa o nome como argumento
    pthread_join(thread1, NULL);
    return 0;
}
```

Múltiplas Threads

```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h> // Para sleep()

void *print_message(void *arg) {
    int thread_id = (int)(long)arg; /* Cast para obter o ID da thread */
    printf("Thread %d: Olá do thread!\n", thread_id);
    sleep(1); /* Pausa a execução da thread por 1 segundo */
    printf("Thread %d: Thread terminando...\n", thread_id);
    pthread_exit(0);
}

int main() {
    pthread_t threads[5]; /* Declara um array de threads (pthread_t) para armazenar os IDs das threads criadas */
    int thread_ids[5];

    /* Cria 5 threads */
    for (int i = 0; i < 5; i++) {
        thread_ids[i] = i + 1; /* Atribui um ID único para cada thread */
        pthread_create(&threads[i], NULL, print_message, (void *)thread_ids[i]);
    }
    /* Faz com que o programa principal espere até que as threads sejam finalizadas antes de continuar a execução */
    for (int i = 0; i < 5; i++) {
        pthread_join(threads[i], NULL);
    }
    printf("Programa principal terminado.\n");
    return 0;
}
```

Execução do Programa

Compilando o Código e Executando o Programa

```
gcc -c multiple_threads.c
gcc -o multiple_threads multiple_threads.o
./multiple_threads
```

Saída possível (a ordem em que as mensagens são impressas pode variar devido à natureza concorrente das threads)

```
Thread 1: Olá do thread!
Thread 2: Olá do thread!
Thread 3: Olá do thread!
Thread 4: Olá do thread!
Thread 5: Olá do thread!
Thread 1: Thread terminando...
Thread 2: Thread terminando...
Thread 3: Thread terminando...
Thread 4: Thread terminando...
Thread 5: Thread terminando...
Programa principal terminado.
```

Função de Thread com retorno

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

void *increment_counter(void *arg) {
    int *iptr = (int *)malloc(sizeof(int));
    *iptr = 0;
    for (int i = 0; i <= 10; i++){
        (*iptr)++;
    }
    return iptr;
}

int main() {
    pthread_t thread1;
    int *resultado;
    pthread_create(&thread1, NULL, increment_counter, NULL);

    //pthread_join permite acessar o retorno da função da thread
    pthread_join(thread1, (void *)&resultado);
    printf("Thread retornou o valor %d\n", *resultado);
    return 0;
}
```

Dúvidas e Discussão