Bài 13: Thư viện Pthread

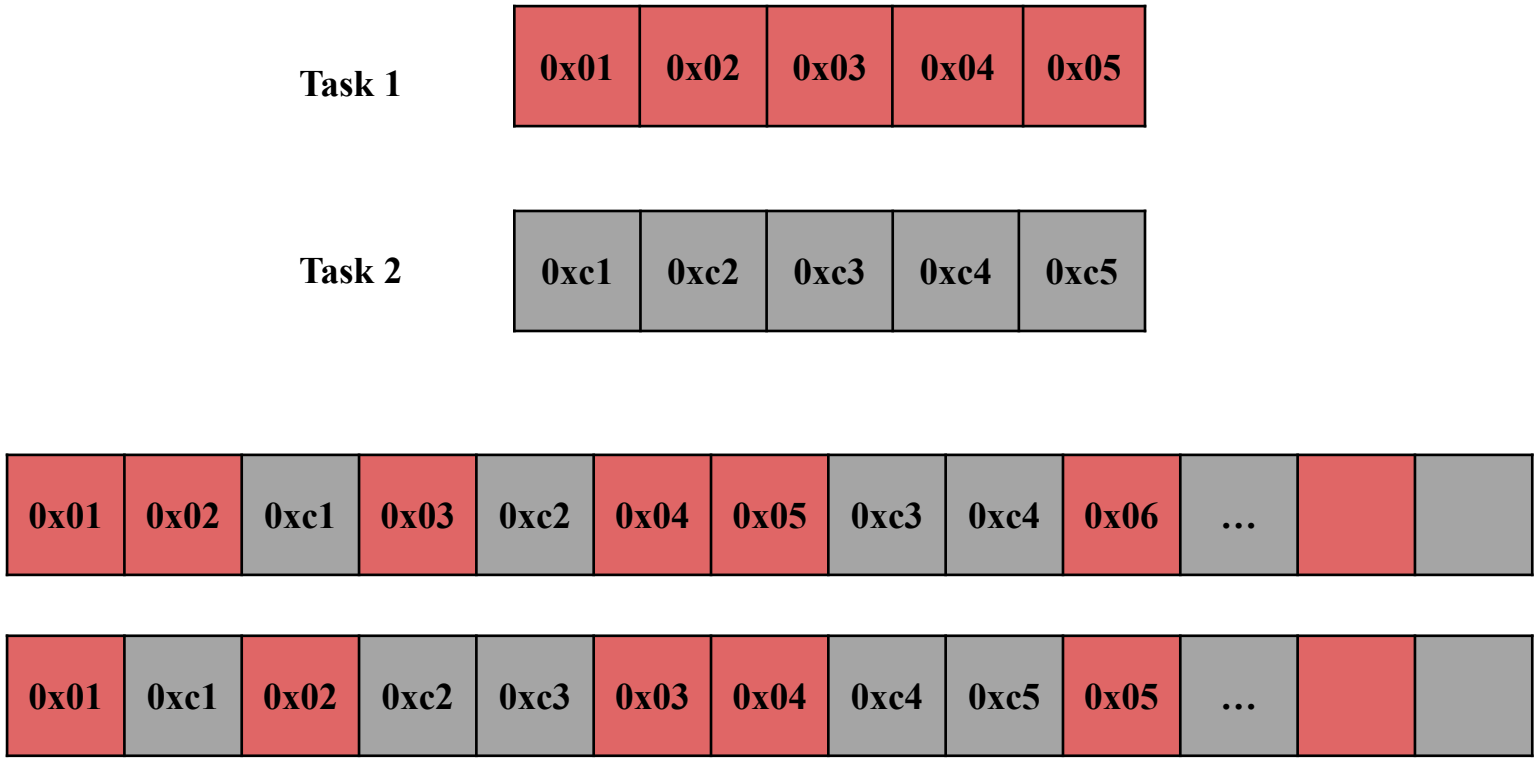
Nguyễn Hoàng Anh

Đặt vấn đề

- Giả sử chương trình thực hiện hai tác vụ liên tục (task1, task2), nếu đặt trong while(1) thì thực hiện tuần tự, task1 thực hiện xong thì mới thực hiện task2.
- Muốn cả 2 tác vụ trên thực thi song song với nhau thì làm thế nào?
- Tuần tự

```c
#include <stdio.h>
#include <unistd.h>
void task1(){
    static int count = 0;
    count++;
    printf("Count taks1: %d\n", count);
    sleep(1);
}
void task2(){
    static int count = 0;
    count++;
    printf("Count taks2: %d\n", count);
    sleep(3);
}
int main(int argc, char const *argv[]){
    while(1){
        task1();
        task2();
    }
    return 0;
}
```

```c
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
void *task1(void *data){
    static int i = 0;
    for (int i=0; i<8; i++){
        printf("count task1: %d\n", i++);
        sleep(1);
    }
}
void *task2(void *data){
    static int i = 0;
    while(1){
        printf("count task2: %d\n", i++);
        sleep(3);
    }
}
void *display(void *a){
    while(1){
        printf("%s\n", (char*)a);
        sleep(1);
    }
}
```



Bản chất đa luồng cũng là tuần tự nhưng phân chia thời gian ở các task

Khái niệm

- **Thread (luồng)** là đơn vị thực thi nhỏ nhất của một tiến trình (**Process**). Mỗi tiến trình có thể chứa nhiều thread, và các thread này chạy song song với nhau.
- Các thread trong cùng tiến trình chia sẻ tài nguyên với nhau nhưng sẽ có stack và bộ đếm chương trình riêng biệt.

C (80Gb)

VSCode (5Gb)

Chrome (5Gb)

Zalo (15Gb)

Skype:

- Chia sẻ màn hình (1 luồng)

Thư viện Pthread

- **Pthread (POSIX Threads)** là thư viện cho lập trình đa luồng trong C/C++:
  - Giúp bạn viết các chương trình có thể thực hiện nhiều tác vụ cùng lúc bằng cách chạy nhiều luồng (threads) song song. Điều này đặc biệt hữu ích trên các hệ thống có nhiều bộ xử lý hoặc bộ xử lý nhiều nhân, vì mỗi luồng có thể được phân bố cho các nhân khác nhau để chạy.
  - Xử lý nhiều công việc cùng lúc, giúp tăng tốc độ xử lý và phân tán công việc hiệu quả hơn.

Tạo mới một thread

- Sử dụng hàm **pthread_create** trong thư viện Pthread:
  - Khởi chạy một thread.
  - Chỉ định cho thread này thực thi một tác vụ cụ thể.

Tạo mới một thread

```
pthread_create(pthread_t *th, const pthread_attr_t *attr, void *(* func)(void *), void *arg)
```

- Tham số 1: một con trỏ kiểu **pthread_t**, đại diện cho một thread mới được tạo ra.
- Tham số 2: một thuộc tính của thread, đặt là **NULL** nếu giữ thuộc tính mặc định.
- Tham số 3: địa chỉ hàm muốn thực thi.
- Tham số 4: một con trỏ đối số cho hàm thuộc kiểu void.

Chờ một thread kết thúc

```
int pthread_join(pthread_t t, void **res)
```

- Tham số 1: ID của thread mà bạn muốn chờ đợi.
- Tham số 2: pointer to pointer mà kết quả trả về từ thread sẽ được lưu trữ. Nếu không cần kết quả, có thể đặt NULL .