Relatório Final: Exemplo 3 - Barreiras

De: Gabriel Henrique da Silva

RA:22020864


```
Essa parte do projeto foi feita em markdown
```

Código

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

#define NUM_THREADS 5

pthread_mutex_t barrier_mutex;
pthread_cond_t barrier_cond;
int barrier_count = 0;
```

Aqui declaramos um mutex (`barrier_mutex`), uma variável de condição (`barrier_cond`) e um contador (`barrier_count`) para controlar a barreira.

Função `barrier_wait`

```
void barrier_wait() {
    pthread_mutex_lock(&barrier_mutex);
    barrier_count++;

    if (barrier_count == NUM_THREADS) {
        barrier_count = 0;
        pthread_cond_broadcast(&barrier_cond);
    } else {
        pthread_cond_wait(&barrier_cond, &barrier_mutex);
    }
```

```
    pthread_mutex_unlock(&barrier_mutex);
}
```

A função `barrier_wait` faz as threads esperarem até que todas tenham chegado à barreira. Quando a última thread chega, ela libera todas as outras.

Função `worker`

```
void *worker(void *arg) {
    int thread_id = *((int *)arg);
    printf("Thread %d chegou na barreira\n", thread_id);
    barrier_wait();
    printf("Thread %d passou da barreira\n", thread_id);
    free(arg);
    return NULL;
}
```

Cada thread executa a função `worker` . Ela chega na barreira, chama `barrier_wait` para esperar até todas as threads estarem na barreira e, em seguida, prossegue.

Função `main`

```
int main() {
    pthread_t threads[NUM_THREADS];
    pthread_mutex_init(&barrier_mutex, NULL);
    pthread_cond_init(&barrier_cond, NULL);

    for (int i = 0; i < NUM_THREADS; i++) {
        int *thread_id = malloc(sizeof(int));
        *thread_id = i;
        pthread_create(&threads[i], NULL, worker, (void *)thread_id);
    }

    for (int i = 0; i < NUM_THREADS; i++) {
        pthread_join(threads[i], NULL);
    }

    pthread_mutex_destroy(&barrier_mutex);
    pthread_cond_destroy(&barrier_cond);
    return 0;
```

```
}
```

No `main`, inicializamos o mutex e a variável de condição, criamos as threads e esperamos que todas terminem com `pthread_join`. Após isso, destruímos o mutex e a variável de condição.