Resumo | Threads – Parte 1

👝 Threads

- Threads são unidades de execução dentro de um processo.
- Compartilham memória, arquivos abertos e outros recursos do processo pai.
- São mais leves e eficientes do que processos para tarefas concorrentes.

Exemplo de uso básico:

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

void* minha_thread(void* arg) {
    printf("Olá da thread!\n");
    return NULL;
}

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

🔐 Mutex

- Mutex (mutual exclusion) é usado para proteger regiões críticas do código.
- Impede que duas ou mais threads acessem o mesmo recurso simultaneamente.

Exemplo com `pthread_mutex_t`:

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

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
int contador = 0;

void* incrementar(void* arg) {
    pthread_mutex_lock(&mutex);
    contador++;
    pthread_mutex_unlock(&mutex);
    return NULL;
}
```

🚦 Semáforos

- Semáforos controlam o acesso concorrente a recursos compartilhados.
- Permitem múltiplas entradas dependendo do valor inicial.

Exemplo com `sem_t`:

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

sem_t semaforo;

void* tarefa(void* arg) {
    sem_wait(&semaforo);
    printf("Thread entrou na seção crítica.\n");
    sem_post(&semaforo);
    return NULL;
}

int main() {
    sem_init(&semaforo, 0, 1); // Semáforo binário
    pthread_t t1, t2;
    pthread_create(&t1, NULL, tarefa, NULL);
    pthread_create(&t2, NULL, tarefa, NULL);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    sem_destroy(&semaforo);
    return 0;
}
```

🔧 Explicação das Funções Importantes

✅ `pthread_create`

int pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine) (void *), void *arg);

int pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine) (void *), void *arg);

- Cria uma nova thread.
- `thread`: ponteiro para o ID da thread.
- `attr`: atributos da thread (use `NULL` para padrão).
- `start_routine`: função que a thread executará.
- `arg`: argumento passado para a função da thread.

✅ `pthread_join`

int pthread_join(pthread_t thread, void **retval);

int pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine) (void *), void *arg);

- Espera o término de uma thread.
- `thread`: ID da thread.
- `retval`: ponteiro para o valor de retorno da thread (pode ser `NULL`).

✅ `pthread_exit`

void pthread_exit(void *retval);

void pthread_exit(void *retval);

- Termina a thread chamadora.
- `retval`: valor a ser retornado (acessado via `pthread_join`).

✅ `sem_init`

int sem_init(sem_t *sem, int pshared, unsigned int value);

int sem_init(sem_t *sem, int pshared, unsigned int value);

- Inicializa um semáforo.
- `sem`: ponteiro para o semáforo.
- `pshared`: `0` para threads no mesmo processo.
- `value`: valor inicial do semáforo.

✅ `sem_wait`

int sem_wait(sem_t *sem);

int sem_wait(sem_t *sem);

- Decrementa o semáforo.
- Bloqueia se o valor for `0` (espera até liberar).

✅ `sem_post`

int sem_post(sem_t *sem);

int sem_post(sem_t *sem);

- Incrementa o semáforo.
- Libera outra thread que estiver esperando.