

## Roteiro II

5. Dê uma descrição do que faz cada um desses tipos ou funções.

- **pthread\_t**: Tipo representante da thread;
- **pthread\_attr\_t**: Atributos que serão repassados para a thread;
- **pthread\_attr\_init**: Inicializa os atributos da thread;
- **pthread\_attr\_destroy**: Desaloca os atributos da thread;
- **pthread\_attr\_setdetachstate**: Adiciona um novo atributo;
- **pthread\_create**: Cria a thread com os parâmetros passados;
- **pthread\_join**: Retorna a sincronia ao código, marcando o fim das threads;
- 

6. Compile (gcc -o joinEx -pthread joinEx.c) e rode o programa. O que ele faz? Explique cada linha do resultado da saída do programa.

O programa cria três threads adiciona os devidos atributos as mesmas, e cada uma delas incrementam uma variável double “r” com valores aleatórios, que após isso printam o valor total.

```
Creating thread 0
Creating thread 1
Creating thread 2
Result: 1.073072e+15
Result: 1.073931e+15
Thread 0 has joined: status = 0
Result: 1.073973e+15
Thread 1 has joined: status = 0
Thread 2 has joined: status = 0
```

8. Dê uma descrição do que faz cada um desses tipos ou funções.

- **pthread\_mutex\_t**: Tipo do mutex;
- **pthread\_mutex\_lock**: função responsável por bloquear o mutex;
- **pthread\_mutex\_unlock**: função responsável por desbloquear o mutex;

## 9. Rode o programa. O que ele faz?

O programa cria 5 thread e cada uma tenta incrementar uma variável em comum com as demais. Ao fim ele printa o valor da variável.

```
1000 added!  
1000 added!  
1000 added!  
1000 added!  
1000 added!  
Shared data = 5000
```

## 10. Retire as diretivas de sincronização "lock" e "unlock". O que acontece com o programa? Você saberia explicar esse comportamento?

Com todas as 5 thread tentando acessar a mesma variável sem um “controlador”, acaba que apenas a thread que “reservar” primeiro a função conseguirá incrementar a variável, e as demais serão bloqueadas de interagir com o valor. É possível, dependendo do tempo de processo da thread vencedora, que outra thread consiga incrementar o valor também.

Saída do programa

```
1000 added!  
1000 added!  
1000 added!  
1000 added!  
1000 added!  
Shared data = 1000
```

Saída com a interação de duas threads

```
1000 added!  
1000 added!  
1000 added!  
1000 added!  
1000 added!  
Shared data = 2000
```

11.

- Qual o tamanho do *buffer*? 10
- Pra que serve a variável *writable* (dentro do *buffer*)? permite a incrementação do valor do *buffer*.
- Se o programa funcionasse corretamente, qual seria sua saída no final? uma concorrência entre o produtor incrementar o *buffer*, e o consumidor decrementar, mas o produtor em tese, produziria mais rápido.