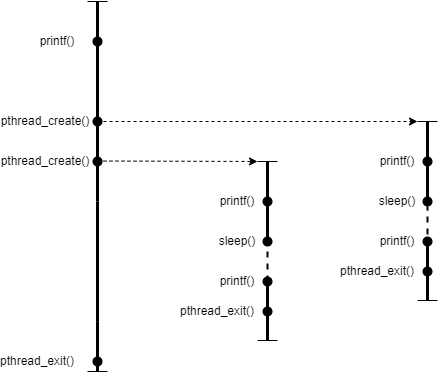
**Threads**

**Marcos Vinícius Simão de Souza – 1591916**

**Exercício 1:**

As threads são criadas no contexto de um processo e compartilham de um mesmo endereço. Cada thread gerencia o seu contexto de execução, também possuem seu próprio contador de programa, apontador de pilha e registradores. A biblioteca **p\_thread.h** é utilizada para interface de programação das threads. A função p\_thread\_t é o identificador da thread, **p\_thread\_create** e **p\_thread\_exit** são utilizadas para criação e finalização das threads onde a função de criação tem como parâmetro uma variável onde será armazenado o valor de retorno da thread.

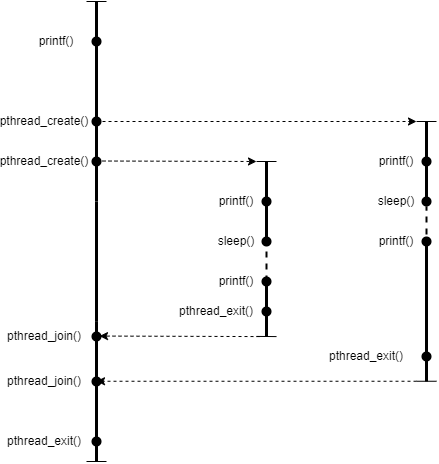
O exercício é inicializado pelo sistema operacional threads de maneira sequencial mas não ficam prontas na mesma ordem de inicialização pois seus valores de retorno são armazenados no mesmo local, os prints nos mostram a sequência em que são executadas. Como cada thread opera de forma independente sua criação, ativação e finalização não é sequencial e quanto mais tempo levar para executar, menor a chance de execução sequencial.

****

**Exercício 2:**

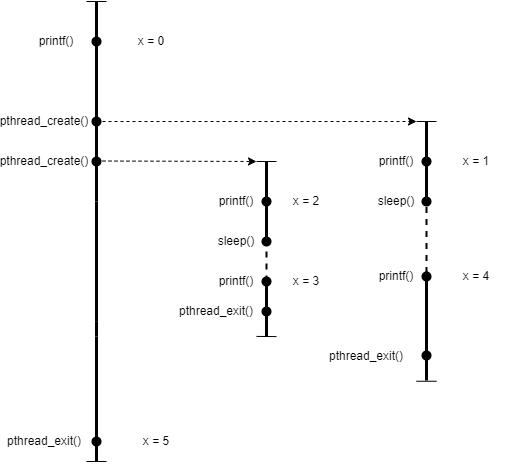
* pthread\_attr\_t: atributos da thread;
* pthread\_attr\_init: inicializa os atributos da thread;
* pthread\_attr\_: trata sobre encadeamento de atributos de thread, se são independentes ou não;
* pthread \_create\_joinable: aguarda retorno da thread;
* pthread\_join: aguarda a finalização da thread;
* pthread\_attr\_destroy: destrói a thread.

O funcionamento do exercício 2 é parecido com o anterior, diferindo em poder utilizar função que esperam a execução da thread. Ao final do programa a função main() aguarda o encerramento das threads e destrói os atributos criados finalizando de vez a execução.

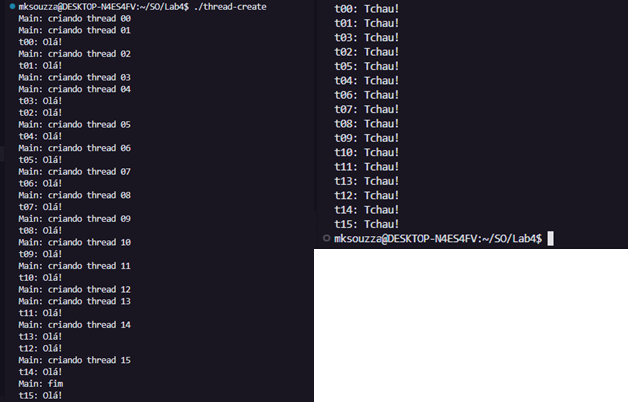
****

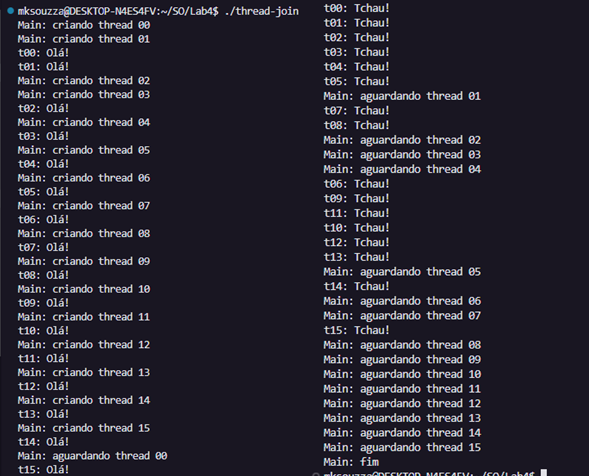
**Exercício 3:**

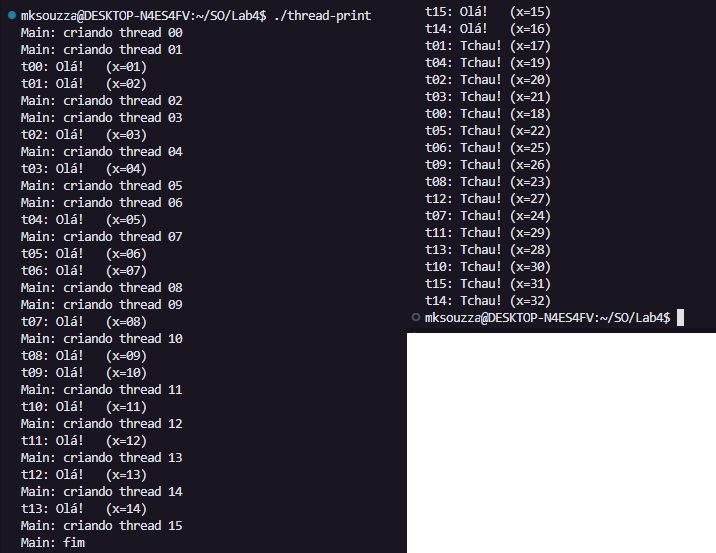
Foi adicionado uma variável x que é compartilhada entre as threads e sofre incrementação a cada thread executada.



**Terminal:**

****

****

****