Questionário - Aula 05

Arthur C. M. Barcella e Matheus P. Salazar

Explique o que é, para que serve e o que contém um TCB - Task Control Block.

É um tipo de estrutura de dados usada pelo SOP para manter o estado da tarefa.

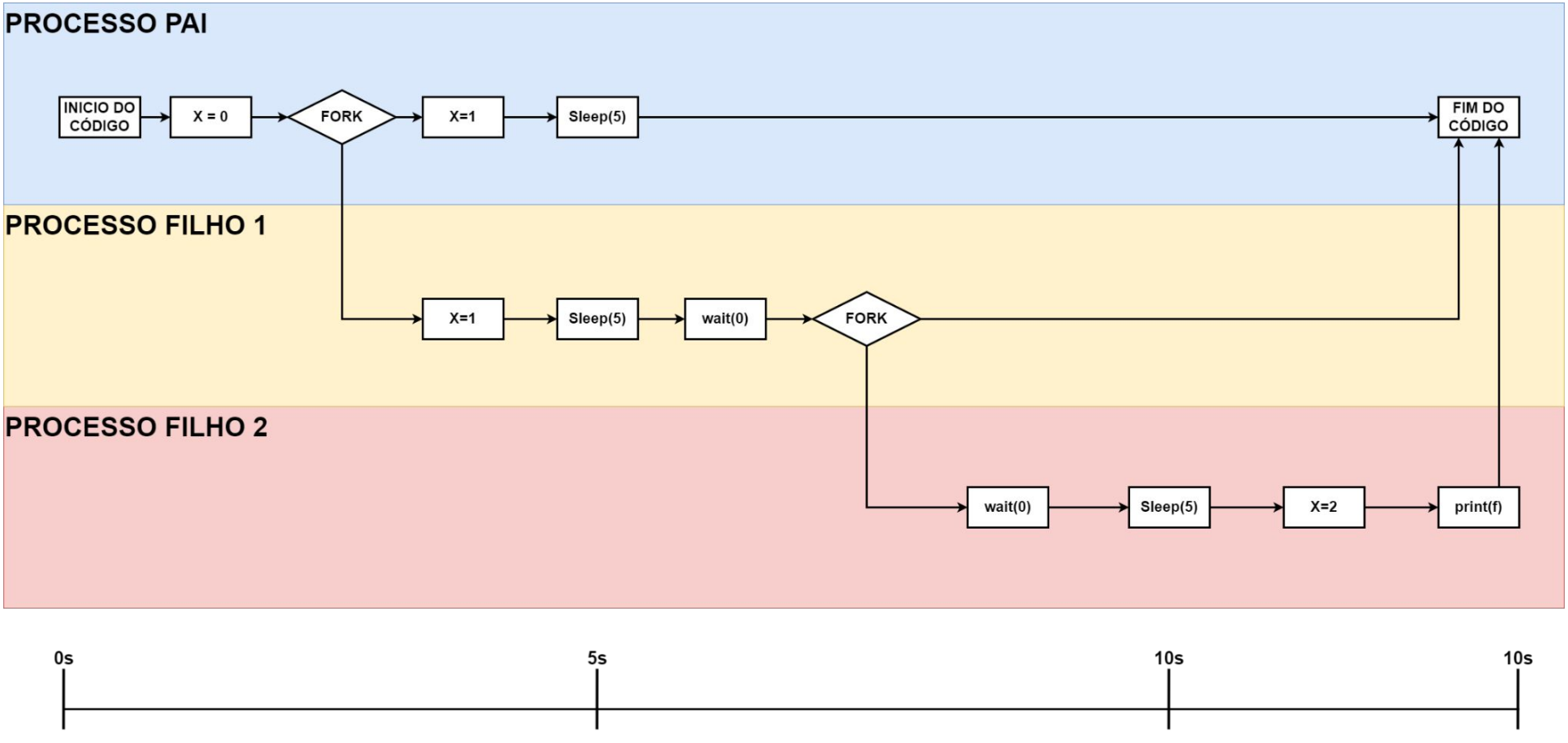
Serve para guardar o estado atual da tarefa, registradores, ponteiros, prioridade de tempo, etc.

O Task Control Block é composto por diversos parâmetros, sendo eles o “Process ID” ou identificador do processo, o “Process state” ou status do processo, “Program counter”, “Register Information”, “Scheduling information”, “Memory related information”, "Accounting information” e “Status information (I/O).

Desenhe o diagrama de tempo da execução do código a seguir, informe qual a saída do programa na tela (com os valores de x) e calcule a duração aproximada de sua execução (em múltiplos de 5 segundos).

Tente responder à pergunta sem compilar/executar o programa.

```
1  int main()
2  {
3          int x = 0 ;
4
5          fork () ;
6          x++ ;
7          sleep (5) ;
8          wait (0) ;
9          fork (0) ;
10         wait (0) ;
11         sleep (5) ;
12         x++ ;
13         printf ("Valor de x: %d\n", x) ;
14 }
```


PROCESSO PAI
INICIO DO CÓDIGO
X = 0
FORK
X=1
Sleep(5)
FIM DO CÓDIGO
PROCESSO FILHO 1
X=1
Sleep(5)
wait(0)
FORK
PROCESSO FILHO 2
wait(0)
Sleep(5)
X=2
print(f)
0s
5s
10s
10s

O que são threads e para que servem?

As threads são um fluxo de operação que ocorre dentro do núcleo do sistema operacional ou dentro de um processo.

Elas servem para permitir que um processo execute várias tarefas de forma simultânea, trazendo uma eficiência maior.

Quais as principais vantagens e desvantagens de threads em relação a processos?

Vantagens: O custo de criação das tarefas é mais baixo, trocar de contexto é mais rápido, todas as threads têm a mesma permissão, baixo uso de memória.

Desvantagens: Em caso de erro todas as threads são afetadas, (Achar mais exemplos).

Forneça dois exemplos de problemas cuja implementação multi-thread não tem desempenho melhor que a respectiva implementação sequencial.

1° Quando se tem uma tarefa pequena, pois esta não exige tanto esforço e gasto computacional, logo não precisa de múltiplas threads.

2° Quando uma tarefa depende do resultado de outra, as threads estariam paradas esperando os resultados chegarem de outra thread.

Associe as afirmações a seguir aos seguintes modelos de threads: a) many-to-one (N:1); b) one-to-one (1:1); c) many-to-many (N:M):

a. [A] Tem a implementação mais simples, leve e eficiente

b. [C] Multiplexa os threads de usuário em um pool de threads de núcleo.

c. [C] Pode impor uma carga muito pesada ao núcleo.

d. [A] Não permite explorar a presença de várias CPUs pelo mesmo processo.

e. [C] Permite uma maior concorrência sem impor muita carga ao núcleo.

Associe as afirmações a seguir aos seguintes modelos de threads: a) many-to-one (N:1); b) one-to-one (1:1); c) many-to-many (N:M):

f. [A] Geralmente implementado por bibliotecas.

g. [B] É o modelo implementado no Windows NT e seus sucessores.

h. [A] Se um thread bloquear, todos os demais têm de esperar por ele.

i. [B] Cada thread no nível do usuário tem sua correspondente dentro do núcleo.

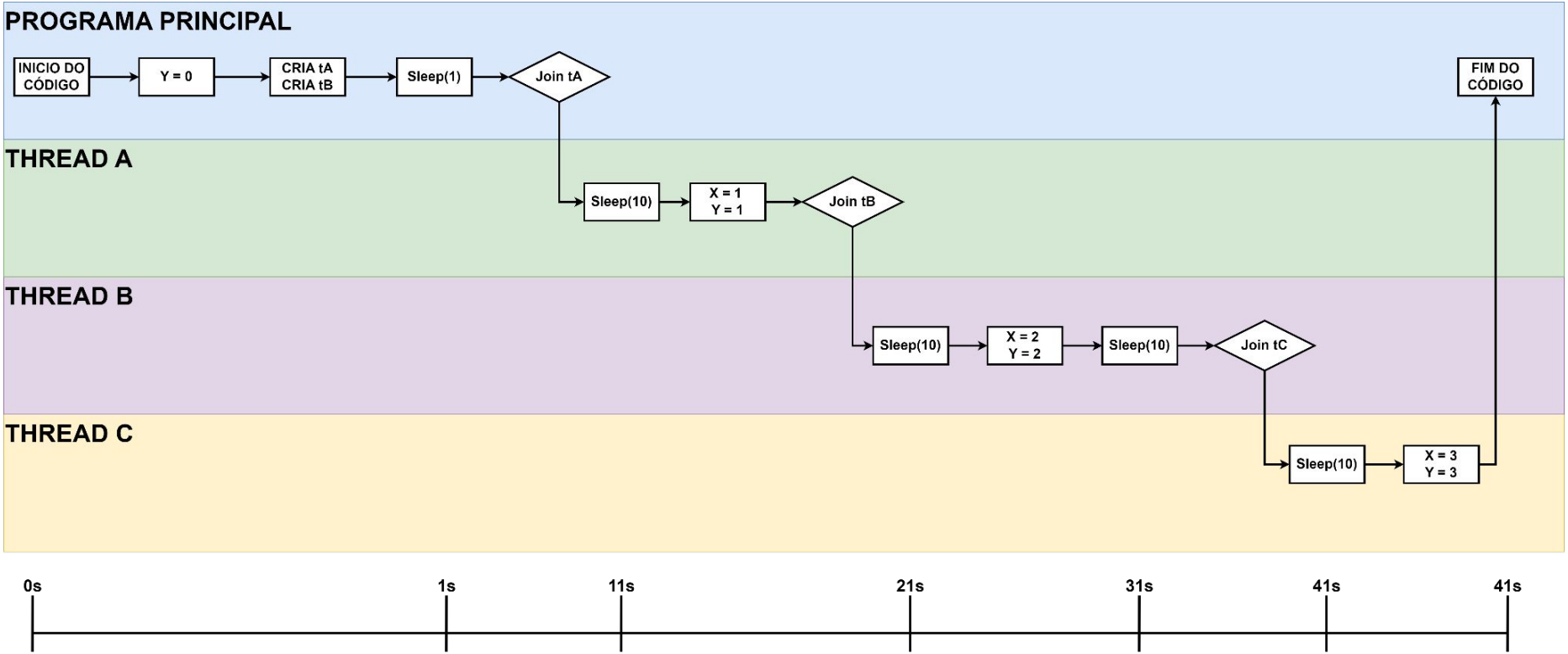
j. [C] É o modelo com implementação mais complexa.

Considerando as implementações de threads N:1 e 1:1 para o trecho de código a seguir;
a) Desenhe os diagramas de execução;
b) Informe as durações aproximadas de execução;
c) Indique a saída do programa na tela;

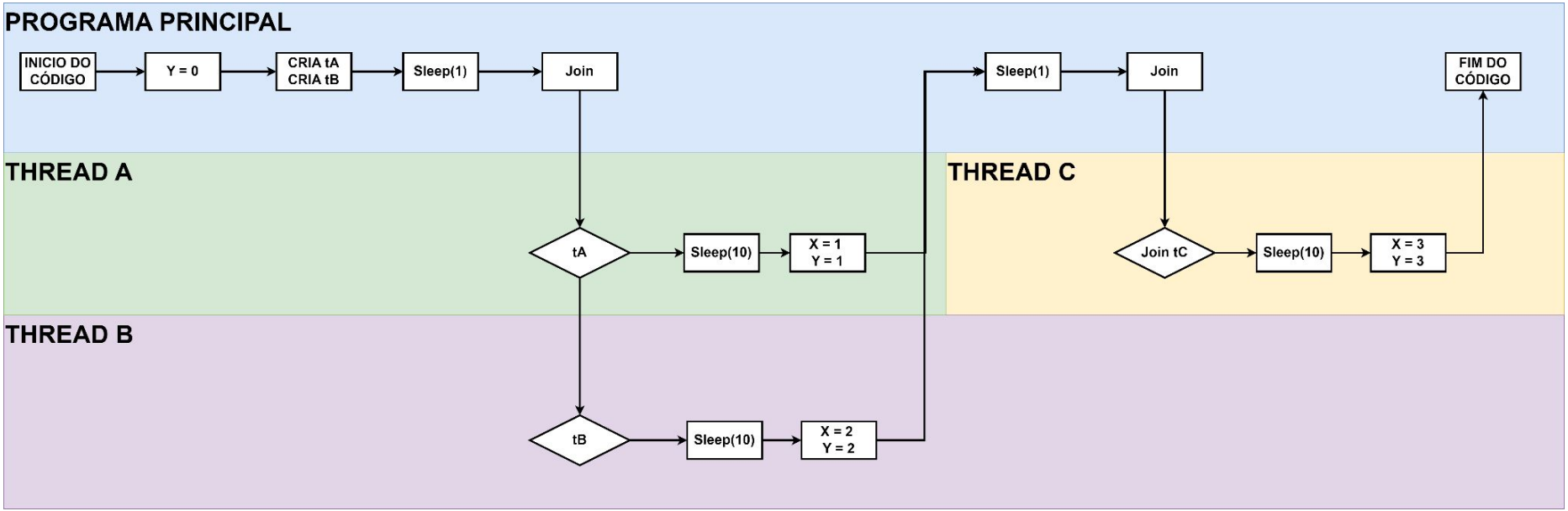
Considere a operação sleep() como uma chamada de sistema (syscall). A chamada “thread_create” cria uma nova thread, “thread_exit” encerra a thread corrente e “thread_join” espera o encerramento da thread informada como parâmetro

```
1  int y = 0 ;
2
3  void threadBody
4  {
5      int x = 0 ;
6      sleep (10) ;
7      printf ("x: %d, y:%d\n", ++x, ++y) ;
8      thread_exit();
9  }
10
11 main ()
12 {
13     thread_create (&tA, threadBody, ...) ;
14     thread_create (&tB, threadBody, ...) ;
15     sleep (1) ;
16     thread_join (&tA) ;
17     thread_join (&tB) ;
18     sleep (1) ;
19     thread_create (&tC, threadBody, ...) ;
20     thread_join (&tC) ;
21 }
```

Considerando a implementação de threads “N:1” :



Considerando a implementação de threads “1:1” :



Questionário - Aula 04

Arthur C. M. Barcella e Matheus P. Salazar