Questionário 05

✂--- Implementação de tarefas* ---✂

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

2. 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 }
```

3. O que são threads e para que servem?

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

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

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

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

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

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

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

   f.[ ] Geralmente implementado por bibliotecas.

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

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

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

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

7. 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 e 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.

*Baseado no conteúdo do livro "Sistemas Operacionais: Conceitos e Mecanismos" do Prof. Carlos A. Maziero (UFPR).

```c
int y = 0 ;

void threadBody
{
    int x = 0 ;
    sleep (10) ;
    printf ("x: %d, y:%d\n", ++x, ++y) ;
    thread_exit();
}

main ()
{
    thread_create (&tA, threadBody, ...) ;
    thread_create (&tB, threadBody, ...) ;
    sleep (1) ;
    thread_join (&tA) ;
    thread_join (&tB) ;
    sleep (1) ;
    thread_create (&tC, threadBody, ...) ;
    thread_join (&tC) ;
}
```

8. Indique quantas letras "X" serão impressas na tela pelo programa abaixo quando for executado com a seguinte linha de comando: ./a.out 4 3 2 1

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
        pid_t pid[10];
        int i;

        int N = atoi(argv[argc-2]);

        for (i=0; i<N; i++)
                pid[i] = fork();
        if (pid[0] != 0 && pid[N-1] != 0)
                pid[N] = fork();
        printf("X");
        return 0;
}
```