Infraestrutura de Software 2022.1 - 2o EE

1. Explique a técnica de escalonamento earliest deadline first (EDF) para sistemas de tempo real críticos*.* O sistema abaixo é escalonável? (1,5)

T1: (c=15,d=20,p=20); T2:(c=5,d=10,p=10)

Obs: c = tempo de computação, p = período, d = deadline.

2. Em gerenciamento de memória, o que seria proteção e relocação? Os processos precisam estar cientes destas técnicas em relação aos seus respectivos espaços de endereçamento? (1,5)

3 Explique como diretórios podem ser implementados em sistemas de arquivos. (2,0)

.

4. Sobre memória virtual , o que seria a tabela de páginas? Explique através de exemplos os algoritmos de substituição de página *segunda chance* e *menos usada. recentemente* (2,0)

5. **Implemente um sistema** de gerenciamento de semáforo de trânsito, o qual é composto de um semáforo de veículos, um semáforo de pedestres e um dispositivo controlador. Os semáforos e o controlador são representados cada um por uma *thread*, e todos executam em um laço infinito. Uma variável global denominada *estado* determina qual semáforo (veículo ou pedestre) estará verde. O valor 0 indica verde para os veículos e vermelho para pedestres. De forma contrária, o valor 1 indica vermelho para os veículos e verde para os pedestres. A cada 50 segundos, o controlador muda o valor da variável *estado* (trocando de 0 para 1 ou vice-versa ) e notifica as *threads* semáforo de veículos e pedestres. Após acordarem e lerem o valor da variável *estado*, ambas as *threads* representando os semáforos mudam seu estado interno (ABERTO ou FECHADO) e, em seguida, dormem aguardando por uma nova notificação do controlador. Assuma a existência de uma função *void timer(int tempoSegundos)* que realiza a contagem de tempo de forma síncrona (bloqueante). **Além da implementação, explique brevemente se barreira é uma técnica apropriada para esse sistema. (3,0)**

**Obs: Não precisa mostrar a criação das threads, mas você deve demonstrar a utilização de outros recursos necessários para a resposta. (2,5).**

int pthread\_mutex\_unlock(pthread\_mutex\_t \*mutex); int pthread\_mutex\_lock(pthread\_mutex\_t \*mutex); int pthread\_cond\_wait(pthread\_cond\_t \*cond, pthread\_mutex\_t \*mutex); int pthread\_cond\_signal(pthread\_cond\_t \*cond);int pthread\_cond\_broadcast(pthread\_cond\_t \**cond*); int pthread\_mutex\_init(pthread\_mutex\_t \*mutex, const pthread\_mutexattr\_t \*attr);int pthread\_mutex\_destroy(pthread\_mutex\_t \*mutex); int pthread\_join(pthread\_t *thread*, void \*\**value\_ptr*); int pthread\_barrier\_wait(pthread\_barrier\_t \*barrier); int pthread\_barrier\_init(pthread\_barrier\_t \* barrier, const pthread\_barrierattr\_t \* attr, unsigned int count );

.