**Infraestrutura de Software - 2º Chamada/Final (2018.2)**

1. Alguns sistemas operacionais famosos adotam um núcleo monolítico e outros micronúcleo. Faça um comparativo de ambas formas de implementação (1,75).

2. Explique como um sistema de arquivo é organizado em um dispositivo de armazenamento (ex: disco rígido) (1,75).

3. Quais diferenças principais entre um algoritmo de escalonamento para sistemas computacionais tradicionais e sistemas computacionais de tempo real? Explique (1,5).

4. Considere um sistema de *e-commerce*, no qual há 2 tipos de robôs: (i) robôs que coletam itens no depósito; (ii) e outros robôs que repõem o estoque do depósito**.** O depósito possui um tamanho limitado **N**. **disp** é a variável que indica quantos itens estão presentes no depósito*.* Quando o  depósito estiver cheio, os robôs repositores deverão aguardar (“dormindo”) a liberação de algum espaço livre no depósito . Caso não haja item disponível, os robôs coletores deverão “dormir”. Adicionalmente, quando um robô coletor pega um item, espaço é liberado no depósito para reposição. Implemente o código em C usando pthreads para este sistema. Você deverá implementar o código  dos robôs coletores e repositores. Para facilitar a implementação, assuma a existência das funções *inserirItemDepósito(int item) removerItemDepósito(),a*s quais fazem a inserção e remoção de itens, respectivamente*.* As threads executam dentro de um laço infinito. Ademais, os valores a serem inseridos no depósito podem ser aleatórios (sem um significado explícito).

**Não precisa mostrar a criação das threads na função main**, mas precisa mostrar o uso/inicialização dos recursos solicitados nesta questão. **Neste problema, há condição de disputa? Caso positivo, tem região crítica? Explique.** (5,0)

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\_mutex\_init(pthread\_mutex\_t \*mutex, const pthread\_mutexattr\_t \*attr);int pthread\_mutex\_destroy(pthread\_mutex\_t \*mutex);