Infraestrutura de Software 2022.1

2a Chamada/Final

1. O que é Spin Lock? Mostre um exemplo. (2,0)

2. Veja os códigos abaixo. (a) e (b) são códigos de aplicações distintas.

| …  pthread\_cond\_wait(&var\_cond,&mutex);  pthread\_mutex\_lock(&mutex);  regiao\_critica();  pthread\_mutex\_unlock(&mutex);  (a) | …  pthread\_cond\_signal(&var\_cond);  pthread\_mutex\_lock(&mutex);  regiao\_critica();  pthread\_mutex\_unlock(&mutex);  (b) |
| --- | --- |

Assumindo que as variáveis foram devidamente inicializadas, existe algum mau uso das funcionalidades de *pthreads*? Justifique sua resposta para cada item (2,0)

3. Explique o que seria *System Calls* (chamadas ao sistema) e a relação com o conceito máquina estendida (2,0).

4. Implemente um sistema nacional bancário usando *pthread*. Este é composto por uma *thread* banco central e outras denominadas banco. A *thread* banco\_central informará quando os bancos deverão abrir ou fechar. Os bancos ficam dormindo até receber a notificação do banco central para abertura do expediente. Após abrirem, voltam a ficar dormindo até receberem a notificação (pelo banco central) para mudar o estado para fechado. Este ciclo deve ser executado em um laço infinito. Assuma a função void timer(int horas) para fazer a contagem de tempo de forma síncrona, e o banco\_central deverá enviar uma notificação a cada 12 horas. **Ademais, responda se existe a possibilidade de deadlock no código.**

**Obs: Não precisa mostrar a criação das threads, mas você deve demonstrar a utilização de outros recursos necessários para a resposta. (4,0).**

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\_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 );