**2a Chamada/Final 2023.1**

1. Defina modo núcleo (modo *kernel).* Qual a importância para sistemas operacionais? (1,5)
2. Explique o significado de chamada ao sistema (system call). Como interrupção é utilizada nesta funcionalidade? (2,0)
3. Explique e mostre um exemplo do algoritmo de escalonamento *Multilevel Feedback Queue (MFQ). (2,0)*
4. O código abaixo possui a possibilidade de ter *deadlock*. Por quê? Como *deadlock* poderia ser removido do código? (2,0)  
   pthread\_mutex\_t mutex1 = PTHREAD\_MUTEX\_INITIALIZER;  
   pthread\_mutex\_t mutex2 = PTHREAD\_MUTEX\_INITIALIZER;

| vovoid thread1() {  while(1) {  pthread\_mutex\_lock(mutex2);  pthread\_mutex\_lock(mutex1);  regiao\_critica();  pthread\_mutex\_unlock(mutex2);  pthread\_mutex\_unlock(mutex1);  }} | vovoid thread2() {  while(1) {  pthread\_mutex\_lock(mutex1);  pthread\_mutex\_lock(mutex2);  regiao\_critica();  pthread\_mutex\_unlock(mutex1);  pthread\_mutex\_unlock(mutex2);  }} |
| --- | --- |

1. Implemente um sistema de controle de catraca de um evento esportivo usando pthreads. O sistema é composto por *N* *threads,* no qual cada catraca é representada por uma *thread*. Quando uma pessoa passa por uma catraca, a respectiva *thread* é responsável em incrementar em uma unidade a variável global *quantidadePublico*. Se a variável global atingir o valor igual a MAXIMO, o sistema não pode aceitar mais nenhuma pessoa no evento esportivo. Assuma que cada *thread* será implementada usando um laço infinito e a função *void* *receberPessoa()* indica que alguém está tentando passar pela catraca. **Além da implementação, explique brevemente se barreira é uma técnica apropriada para esse sistema. Obs: Não precisa mostrar a criação das threads, mas você deve demonstrar a utilização de outros recursos necessários para a resposta. Assuma que a função *receberPessoa* está implementada(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 );
2. ***Somente para quem estiver fazendo a 2a chamada do 1o EE*.** Considerando a arquitetura x86, explique resumidamente o funcionamento do bootloader. Quais as etapas necessárias e os cuidados na implementação.