UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
DEPARTAMENTO DE INFORMÁTICA APLICADA

INF01151 – SISTEMAS OPERACIONAIS II N
ATIVIDADE DE PROGRAMAÇÃO:
EXCLUSÃO MÚTUA IMPLEMENTADA EM SOFTWARE

OBJETIVO DA ATIVIDADE DE PROGRAMAÇÃO:

Esta atividade de Programação tem por objetivo exercitar a implementação de primitivas de exclusão mútua em software, usando como estudo de caso o Algoritmo de Lamport para resolver o problema de exclusão mútua (também conhecido como "Algoritmo da Padaria"). Esta atividade poderá ser feita em dupla. **A atividade deverá ser feita, preferencialmente, em C/C++**. Alternativamente, a atividade poderá ser feita usando a linguagem Rust.

A discussão via pseudocódigo do Algoritmo da Padaria pode ser encontrada no Tópico 5 – Exclusão Mútua (Implementação), slide 14. A sua tarefa nesta Atividade de Programação será encapsular o pseudocódigo em duas primitivas:

- void lamport_mutex_lock (int thread_id);
- void lamport_mutex_unlock (int thread_id);

Essas primitivas deverão ser implementadas em uma bibloteca (arquivo "libxxxxxxxxxx.so"), que deverá ser vinculada dinamicamente ao programa que as utiliza. Para facilitar essa implementação, você pode reusar livremente o código da implementação em C do Algoritmo de Lamport disponível no Tópico 5. Lembre-se que essa biblioteca deverá ser acompanhada de um header (arquivo .h) que conterá os protótipos das primitivas, para uso no programa.

ATENÇÃO:

Você também precisará de uma primitiva na biblioteca, por ex. lamport_mutex_init(), para inicializar as estruturas de dados usadas para indicar qual thread está escolhendo um ticket e qual o valor de ticket que cada thread possui no momento.

LEMBRE-SE:

1. Para criar uma biblioteca usando o gcc, você deverá as flags:
   -fPIC (Position Independent Code)
   -shared (para criar o binário da biblioteca, arquivo ".so" de shared object)

2. O nome da biblioteca deverá começar como "lib" e ter a extensão ".so", por exemplo "libmutex.so"

Em seguida, você deverá implementar um programa que usa essas primitivas para controlar o acesso a uma seção crítica de um código. Neste caso, utilize o código "*Exemplo de Thread Simples - Com Condição de Corrida*" disponível no Tópico 4 do Moodle.

Para simplificar:

- Utilize a mesma macro e valor de macro para indicar o número de tarefas concorrentes na biblioteca e no programa. Como sugestão, considere apenas três threads.
- Considere que cada thread irá incrementar a variável global compartilhada 3.000.000 vezes. O resultado final consistente da variável global compartilhada deverá ser 3 threads x 3.000.000 incrementos por thread = 9.000.000.

LEMBRE-SE: Para compilar um programa usando uma biblioteca não padrão, você pode manter o binário da biblioteca no próprio diretório onde irá compilar o executável. Assim, você deverá usar a flag do gcc -L ., para indicar que irá usar uma biblioteca disponível no próprio diretório. Além disso, você deverá especificar qual biblioteca pretende usar na ligação. Por exemplo:

```
gcc -L . -o programa programa.c -lmutex -lpthread
```

Execute as duas versões do programa usando, para o controle de acesso à seção crítica, a) as primitivas que implementam o Algoritmo da Padaria e b) as primitivas ***pthread_mutex_lock*** e ***pthread_mutex_unlock***, e compare o tempo de execução de ambas.

A variável de ambiente LD_LIBRARY_PATH deverá ser usada para indicar o caminho onde está a biblioteca que vocês desenvolveram.

LEMBRE-SE: Você pode usar a função built-in ***time*** do bash para medir o tempo de execução do programa. Por exemplo:

```
$ time LD_LIBRARY_PATH=. ./programa
```

ORIENTAÇÕES PARA A ENTREGA:

Como resultado desta tarefa, você deverá submeter um único arquivo ZIP contendo o código desenvolvido, um arquivo Makefile para facilitar a compilação, e um relatório demonstrando o funcionamento do projeto (com prints) e descrevendo o trabalho desenvolvido.

No relatório, você deverá descrever o sistema no qual desenvolveu o trabalho, o processo de desenvolvimento da biblioteca e do programa principal, os resultados obtidos com a execução do programa usando as primitivas de exclusão mútua implementadas em software vs. as primitivas pthread_mutex_lock e pthread_mutex_unlock, e relatar as principais dificuldades encontradas.

Em especial, preocupe-se com as perguntas a seguir:

- Você observou alguma diferença no tempo de execução entre os programas que usam primitivas de exclusão mútua implementadas em software e as disponíveis na biblioteca POSIX threads?
- Você identificou algum problema no resultado final da execução do programa usando a implementação de mutex baseada no Algoritmo da Padaria?

Este documento de trabalho prático poderá ser revisado conforme feedback recebido da turma. Prazos de entrega estão disponíveis no Moodle.