Trabalho 1A

Implementar um produtor-consumidor usando PThreads.

Marcos Vinícius Treviso (121150107)
Wolgan Ens Quepfert (121151535)

INTRODUÇÃO E INSTRUÇÕES:

Para a realização do trabalho, foi utilizada a linguagem de programação C++, porém sem o paradigma de orientação a objetos, pois seu uso dificultaria o manuseio de memória compartilhada, fazendo-se necessário a criação regras confusas e desnecessárias, e também a criação de classes extras para herança de atributos e métodos com escopo estático. Ou seja, devido aos fatos citados acima, foi optado por utilizar o ambiente de programação C++ com as lógicas funcionais do C.

A compilação pode ser feita utilizando o compilador g++, onde é necessário passar como argumento a biblioteca de pthreads.

Já a execução é feita apenas com o lançamento do executável obtido na compilação com o passamento de três parâmetros à ele: *tempo de espera a cada execução de thread*, *quantidade de produtores*, *quantidade de consumidores*, respectivamente.

O código para a realização das especificações acima é:

```
g++ -o myexec prodcons.cpp -Wall -Wextra -lpthread
./myexec <tempo.threads> <qtd.produtores> <qtd.consumidores>
```

Ou pode-se usar o script sh pré-definido pelo grupo com todas as caractrísticas requesitdas no trabalho.

```
./run.sh
```

O trabalho desenvolvido funciona apenas em ambientes familiarizados com POSIX. Foi primordialmente implementado para rodar em ambientes linux.

DESENVOLVIMENTO:

Foi resolvido o trabalho de acordo com todas as caracetísticas impostas para sua realização.

1) Ter n produtores. Assuma 3 < n < 10.

A quantidade de produtores é passada como argumento do programa e está filtrada para ser no intervalo aberto de 3 a 10.

2) Ter m consumidores. Assuma 3 < m < 10.

A quantidade de consumidores é passada como argumento do programa e está filtrada para ser no intervalo aberto de 3 a 10.

3) Não há limite de produtos produzidos.

Para isso, não foi colocada nenhuma verificação de capacidade na função de produção das caixas, ou seja, todos produtores podem produzir sem restrições de capacidade e tempo.

4) - Cada produtor pode produzir mais de um produto por vez. Considere a produção individual/por vez entre [1, 5].

A produção individual é definida aleatóriamente entre o intervalo fechado de 1 a 5. Para realizar a produção, cada thread de produção incrementa o valor de produção individual na quantidade de itens do sistema.

5) Consumos só podem ser feitos se existir algo em estoque (produzido) ou em quantidade suficiente. Considere o consumo individual/por vez entre [1, 5].

O consumo individual é definido aleatóriamente entre o intervalo fechado de 1 a 5. Para realizar o consumo, cada thread de produção decrementa o valor de produção individual na quantidade de itens do sistema.
Para a análise de consumo disponível, foi verificado se a quantidade de itens disponíveis no sistema é maior ou igual ao valor de consumo individual, onde caso a verificação retorne verdadeiro, será consumido.

6) Tanto produtores como consumidores executam infinitamente.

A execução das produções e dos consumos foram colocadas dentro de laços infinitos, onde só serão paradas quanto o tempo total de execução, imposto no item 8, é disposto.

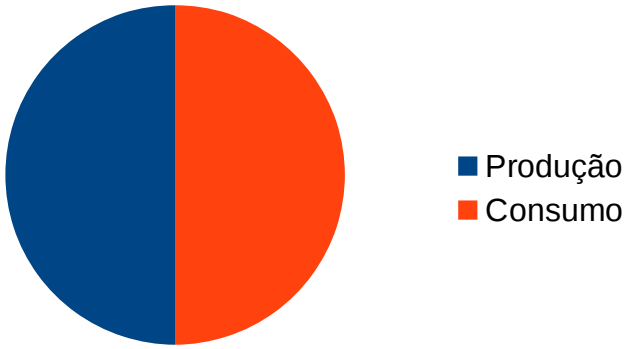
7) Utilize meios que garantam a integridade do valor em estoque.

Para garantir a integridade do valor em estoque, foi implementado a produção e o consumo com o *pthread_mutex*, onde o mesmo foi colocado em regiões críticas do código. Na produção bastou colocar o mutex na incrementação dos itens do sistema. Já no consumo, foi preciso colocar o mutex antes da análise de consumo disponível.

8) Mostre que o programa funciona corretamente por um determinado intervalo de execução. Todas as threads produziram e consumiram em algum momento?

Foi utilizado 5 produtores e 5 consumidores. Cada um produzia e consumia 4 caixas por vez. A produção e o consumo foram feitos a cada 1s. A total execução foi feita em aproximadamente 5 segundos.

O resultado do programa obtido foi:



Threads Produção	10	10	10	10	10
Threads Consumo	10	10	10	10	10

Todas as threads de produção produziram em um determinado instante, e todas threads de consumo consumiram em um determinado instante. Fazendo com que, de acordo com os dados obtidos, a quantidade final de itens no sistema fosse igual a zero.

9) Análise o comportamento do programa para n==m, n<m e n>m. Há alguma diferença? Deveria haver?

Além de analisar a quantidade de produtores e consumidores, deve-se considerar o valor de produção/consumo individual, onde isso pode influenciar diretamente nas questões de armazenamento de caixas no sistema.

Para a análise desse ítem, será considerado que a produção e consumo individual serão iguais, e como nesse caso de análise pouco importa o valor de produção, foi escolhido arbitrariamente que o valor seria 4 para ambos.

(n==m) Quando a quantidade de produtores e consumidores são iguais, os valores tendem a se neutralizar a cada execução alternada de produtores e consumidores, pois a quantidade produzida logo será consumida caso possível. No fim da execução a quantidade total de itens é zero, pois todos os consumidores conseguirão consumir as caixas produzidas.

(n<m) Quando há menos produtores que consumidores, os valores tendem a ficar a zero enquanto as threads de consumo estão executando, pois desse modo, a determinado número de threads já tera extinguido as caixas no sistema. Nesse caso, é obrigado a ser imposto políticas de verficação de integridade dos itens disponíveis no sistema, como abordado no item 7.

(n>m) Quando há mais produtores que consumidores, a quantidade de itens tende ao infinito positivo, de acordo com a razão de soma e subtração impostas pelos produtores e consumidores, respectivamente.