UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

DANIEL LA RUBIA ROLIM - DRE: 115033904

PAULA MACEDO DA CRUZ - DRE: 113049909

LEITORES E ESCRITORES SEM INANIÇÃO

Rio de Janeiro

2019

Segundo Trabalho de Implementação - 2019/2
Disciplina: Computação Concorrente
Professora: Silvana Rossetto

Descrição do problema

O trabalho consiste em implementar dois programas: um principal e outro auxiliar e arquivo log. O programa principal trata-se de um problema clássico de concorrência, o problema dos leitores/escritores. A implementação do trabalho deve garantir ausência de inanição entre threads e deve respeitar as seguintes condições lógicas:

• mais de um leitor pode ler ao mesmo tempo;
• apenas um escritor pode escrever de cada vez;
• não é permitido ler e escrever ao mesmo tempo.

O programa auxiliar verifica se a execução do programa principal ocorreu com sucesso.

Solução do problema

O trabalho apresenta dois programas, o principal implementado em C e o auxiliar responsável pela verificação do programa principal, implementado em Python. O programa principal gera um arquivo log que é verificado pelo programa auxiliar, avaliando se execução do programa principal foi realizada com sucesso.

No programa main, a solução utilizada faz uso de mutex para controle de acesso às áreas críticas e o uso de variável de condição. Para resolver a questão da inanição, as threads escritoras apresentam prioridade com relação às threads leitoras. Basicamente a permissão de escrita e leitura é baseada em turnos. Quando as threads são inicializadas, elas são controladas por mutex para que apenas uma por vez verifique a condição 'turn'. Turn é um contador modular que controla a vez da thread que deverá realizar a operação.

Divisão do projeto

Programa Principal (main.c) -> Implementado em C, realiza as execuções das threads leitoras e escritoras respeitando as condições lógicas e outras restrições solicitadas pelo trabalho.

Programa Auxiliar (checkOperations.py) -> Implementado em Python, tem como entrada o log gerado pelo programa principal. Responsável por checar se a execução do programa principal foi bem sucedida ou não.

Makefile -> Instrui como compilar e como realizar o link entre os programas.

Modularidade do programa

A solução implementada não é extensa e nem utiliza-se de muitas estruturas de dados, logo, não consideramos tão importante a existência de diversos arquivos.

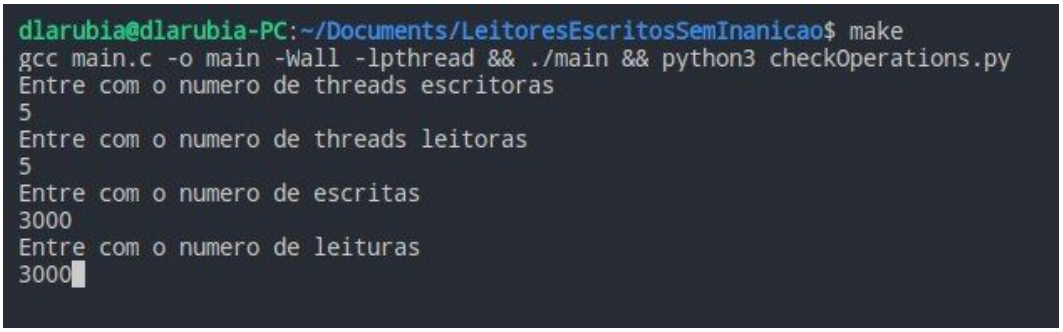
Ambiente utilizado

O ambiente de testes utilizados neste trabalho é Unix. A versão do compilador gcc utilizado foi a 9.1. A máquina utilizada possui 4 processadores.

Testes realizados e resultados

O programa foi compilado utilizando o makefile disponível no github.

Como na descrição do trabalho não está claro se a entrada é por linha de comando ou entrada padrão (ler do teclado), optamos por implementar pela entrada padrão.



Tela de execução programa principal

Caso 1

Número de Threads Leitoras = 5
Número de Threads Escritoras = 5
Quantidade de escritas = 3000
Quantidade de leituras = 3000

```
Fim da execução do programa principal.


A thread escritora 0 realizou 19.9% das escritas. Aguardou permissão 596 vezes no total.
A thread escritora 1 realizou 20.67% das escritas. Aguardou permissão 603 vezes no total.
A thread escritora 2 realizou 20.67% das escritas. Aguardou permissão 606 vezes no total.
A thread escritora 3 realizou 20.23% das escritas. Aguardou permissão 590 vezes no total.
A thread escritora 4 realizou 18.53% das escritas. Aguardou permissão 530 vezes no total.
A thread leitora 5 realizou 19.27% das leituras. Aguardou permissão 698 vezes no total.
A thread leitora 6 realizou 19.87% das leituras. Aguardou permissão 688 vezes no total.
A thread leitora 7 realizou 22.27% das leituras. Aguardou permissão 685 vezes no total.
A thread leitora 8 realizou 19.77% das leituras. Aguardou permissão 630 vezes no total.
A thread leitora 9 realizou 18.83% das leituras. Aguardou permissão 644 vezes no total.

Quantidade de escritas esperadas: 3000
Quantidade de escritas feitas: 3000
Quantidade de leituras esperadas: 3000
Quantidade de leituras feitas: 3000

Quantidade de threads escritoras utilizadas: 5
Quantidade de threads leitoras utilizadas: 5
Quantidade de vezes em que foi utilizada técnica de sincronização (aguardando permissão): 6270
O programa foi executado corretamente.
Execução encerrada.
```

Impressão da tela do programa auxiliar para o caso 1

Caso 2

Número de threads leitoras = 10
Número de threads escritoras = 1
Quantidade de escritas = 100000
Quantidade de leituras = 100000

```
Fim da execução do programa principal.


A thread escritora 0 realizou 100.0% das escritas. Aguardou permissão 4156 vezes no total.
A thread leitora 1 realizou 10.14% das leituras. Aguardou permissão 80369 vezes no total.
A thread leitora 2 realizou 9.93% das leituras. Aguardou permissão 80724 vezes no total.
A thread leitora 3 realizou 10.06% das leituras. Aguardou permissão 79575 vezes no total.
A thread leitora 4 realizou 9.9% das leituras. Aguardou permissão 79283 vezes no total.
A thread leitora 5 realizou 10.05% das leituras. Aguardou permissão 80457 vezes no total.
A thread leitora 6 realizou 10.2% das leituras. Aguardou permissão 80122 vezes no total.
A thread leitora 7 realizou 9.93% das leituras. Aguardou permissão 79825 vezes no total.
A thread leitora 8 realizou 9.88% das leituras. Aguardou permissão 81083 vezes no total.
A thread leitora 9 realizou 10.17% das leituras. Aguardou permissão 80397 vezes no total.
A thread leitora 10 realizou 9.73% das leituras. Aguardou permissão 80263 vezes no total.

Quantidade de escritas esperadas: 100000
Quantidade de escritas feitas: 100000
Quantidade de leituras esperadas: 100000
Quantidade de leituras feitas: 100000

Quantidade de threads escritoras utilizadas: 1
Quantidade de threads leitoras utilizadas: 10
Quantidade de vezes em que foi utilizada técnica de sincronização (aguardando permissão): 806254
O programa foi executado corretamente.
Execução encerrada.
```

Impressão da tela do programa auxiliar para o caso 2

Caso 3

Número de threads leitoras = 1
Número de threads escritoras = 10
Quantidade de escritas = 500000
Quantidade de leituras = 500000

```
Fim da execução do programa principal.


A thread escritora 0 realizou 10.0% das escritas. Aguardou permissão 50003 vezes no total.
A thread escritora 1 realizou 10.0% das escritas. Aguardou permissão 49999 vezes no total.
A thread escritora 2 realizou 10.0% das escritas. Aguardou permissão 49999 vezes no total.
A thread escritora 3 realizou 10.0% das escritas. Aguardou permissão 50000 vezes no total.
A thread escritora 4 realizou 10.0% das escritas. Aguardou permissão 49999 vezes no total.
A thread escritora 5 realizou 10.0% das escritas. Aguardou permissão 50002 vezes no total.
A thread escritora 6 realizou 10.0% das escritas. Aguardou permissão 49999 vezes no total.
A thread escritora 7 realizou 10.0% das escritas. Aguardou permissão 49999 vezes no total.
A thread escritora 8 realizou 10.0% das escritas. Aguardou permissão 49999 vezes no total.
A thread escritora 9 realizou 10.0% das escritas. Aguardou permissão 50000 vezes no total.
A thread leitora 10 realizou 100.0% das leituras. Aguardou permissão 446110 vezes no total.

Quantidade de escritas esperadas: 500000
Quantidade de escritas feitas: 500000
Quantidade de leituras esperadas: 500000
Quantidade de leituras feitas: 500000

Quantidade de threads escritoras utilizadas: 10
Quantidade de threads leitoras utilizadas: 1
Quantidade de vezes em que foi utilizada técnica de sincronização (aguardando permissão): 946109
O programa foi executado corretamente.
Execução encerrada.
```

Impressão da tela do programa auxiliar para o caso 3

Caso 4

Número de threads escritoras = 3
Número de threads leitoras = 6
Quantidade de leituras = 1000
Quantidade de escritas = 1000

```
Fim da execução do programa principal.


A thread escritora 0 realizou 32.9% das escritas. Aguardou permissão 199 vezes no total.
A thread escritora 1 realizou 36.4% das escritas. Aguardou permissão 196 vezes no total.
A thread escritora 2 realizou 30.7% das escritas. Aguardou permissão 164 vezes no total.
A thread leitora 3 realizou 16.2% das leituras. Aguardou permissão 335 vezes no total.
A thread leitora 4 realizou 14.4% das leituras. Aguardou permissão 371 vezes no total.
A thread leitora 5 realizou 18.9% das leituras. Aguardou permissão 429 vezes no total.
A thread leitora 6 realizou 17.2% das leituras. Aguardou permissão 371 vezes no total.
A thread leitora 7 realizou 17.7% das leituras. Aguardou permissão 436 vezes no total.
A thread leitora 8 realizou 15.6% das leituras. Aguardou permissão 354 vezes no total.

Quantidade de escritas esperadas: 1000
Quantidade de escritas feitas: 1000
Quantidade de leituras esperadas: 1000
Quantidade de leituras feitas: 1000

Quantidade de threads escritoras utilizadas: 3
Quantidade de threads leitoras utilizadas: 6
Quantidade de vezes em que foi utilizada técnica de sincronização (aguardando permissão): 2855
O programa foi executado corretamente.
Execução encerrada.
```

Impressão da tela do programa auxiliar para o caso 4

Observação: Valores distintos para leitura e escrita podem causar deadlocks na execução do programa. Tal problema pode ser corrigido com os trechos de códigos abaixo:

Escritores continuam enviando broadcast para leitores quando são encerradas as escritas:

```
while(writings == 0 && readings > 0) {
            pthread_cond_broadcast(&permissionToRead);
            printf("Leituras restantes: %d", readings);
        }
```

Leitores continuam enviando broadcast para escritores quando são encerradas as leituras:

```
while(readings == 0 && writings > 0) {
            turn = (turn + 1) % 2;
            pthread_cond_signal(&permissionToWrite);
            printf("Escritas restantes: %d", writings);
        }
```

Conclusão

Implementar a solução para um programa concorrente com threads leitoras/ escritoras e com ausência de inanição não é algo tão trivial, pois é necessário se pensar na melhor estratégia a ser utilizada. O uso de prioridade para escrita foi uma das estratégias utilizadas. Estabelecer como deveria ser implementada essa prioridade, que fatores deveriam ser relevantes ou não, e como fazer isso de uma forma simplificada também não foi algo trivial, pois a cada modificação do código, fez com que surgisse situações de starvation ou até mesmo deadlock.

Gerar o arquivo texto de saída do programa principal também não foi algo simples, pois não nos recordamos muito bem deste tópico. Logo, a integração da saída do programa principal com o auxiliar foi um problema.

Novamente, tivemos dificuldades de entender o enunciado do trabalho, logo, o mesmo foi implementado de acordo com nossa interpretação em relação ao que foi pedido e de acordo com nossas limitações de tempo e conhecimento para a realização do mesmo.