COMPUTAÇÃO CONCORRENTE (MAB-117) 2019.2
Prof. Silvana Rossetto

SEGUNDO TRABALHO DE IMPLEMENTAÇÃO

PROBLEMA DE LEITORES E ESCRITORES SEM INANIÇÃO

Nome: Letícia Freire Carvalho de Sousa
DRE: 118025324

Sumário

Projeto da Solução

Funções

inicializa_log
Descrição: função que retorna copia o template(funções) do arquivo de template para o início do arquivo de log
Parâmetros: não possui
Retorno: não possui

gera_log
Descrição: função que gera o log de uma ação e o concatena no final do arquivo de log como chamada de uma função
Parâmetros: tipo(inteiro), id(inteiro), acao(inteiro)
Retorno: não possui

finaliza_log
Descrição: função que concatena no final do arquivo de log a chamada de uma função que imprime uma mensagem confirmando a validade da ordem das operações executadas, além de um comentário do python dizendo quanto tempo levou a execução do programa
Parâmetros: tempo_total(double)
Retorno: não possui

get_senha
Descrição: função que retorna uma senha de espera para um thread que deseja realizar uma leitura/escrita, além de gerar o log dessa espera
Parâmetros: tipo(inteiro), id(inteiro)
Retorno: inteiro

registra_leitura
Descrição: função que concatena o conteúdo lido por uma thread no final do seu arquivo de saída
Parâmetros: id(inteiro), conteúdo(inteiro)
Retorno: não possui

entra_fila
Descrição: função que registra a entrada de uma thread na fila de espera e usa de exclusão mútua e variável de condição para fazer com que a thread só siga seu fluxo quando ela for a primeira na fila de espera
Parâmetros: tipo(inteiro), id(inteiro), senha(inteiro)
Retorno: não possui

sai_fila
Descrição: função que registra a saída de uma thread da fila de espera e usa de exclusão mútua e variável de condição para fazer com que a "fila ande"(incrementando a senha atual) e chama a função broadcast para que todas as threads acordem e verifiquem se sua senha é a senha atual
Parâmetros: não possui
Retorno: não possui

entra_l
Descrição: função que chama a **entra_fila** para um thread leitora e espera que ela consiga sair da fila de espera, quando isso acontece, ela usa da exclusão mútua e variável de condição para esperar não haver mais um escritor escrevendo, incrementa o número de leitores ativos, gera o log de sua saída e chama a função **sai_fila**
Parâmetros: id(inteiro), senha(inteiro)
Retorno: não possui

sai_l
Descrição: função que usa da exclusão mútua para decrementar o número de leitores ativos, gerar o log de saída desse leitor e caso o número de leitores ativos seja zero, acorda um escritor
Parâmetros: id(inteiro)
Retorno: não possui

entra_e
Descrição: função que chama a **entra_fila** para um thread escritora e espera que ela consiga sair da fila de espera, quando isso acontece, ela usa da exclusão mútua e variável de condição para esperar não haver mais um escritor escrevendo nem leitores lendo, incrementa o número de escritores ativos, gera o log de sua saída e chama a função **sai_fila**
Parâmetros: id(inteiro), senha(inteiro)
Retorno: não possui

sai_e
Descrição: função que usa da exclusão mútua para decrementar o número de escritores ativos, gerar o log de saída desse leitor e acorda um escritor e todos os leitores
Parâmetros: id(inteiro)
Retorno: não possui

ler
Descrição: função a ser executada pelas threads leitoras. Ela realiza um número de iterações definido pelo usuário na entrada e a cada iteração chama a função **get_senha** para entrar na fila de execução, chama a função **entra_l**, faz uma variável de conteúdo_local receber o conteúdo da variável global, chama a função **sai_l**, e chama a função **registra_leitura** para registrar o conteúdo local no seu arquivo de saída
Parâmetros: threadid(ponteiro para void)
Retorno: NULL

escrever
Descrição: função a ser executada pelas threads escritoras. Ela realiza um número de iterações definido pelo usuário na entrada e a cada iteração chama a função **get_senha** para entrar na fila de execução, chama a função **entra_e**, registra seu id na variável global “conteúdo” e chama a função **sai_e**
Parâmetros: threadid(ponteiro para void)
Retorno: NULL

Projeto do Programa Auxiliar para Verificar Log

Estruturas

fila_execucao
Descrição: fila(estrutura do tipo *first in first out*) que guarda a ordem com que as threads solicitaram a execução de uma operação de leitura/escrita

instrucoes_ativas
Descrição: estrutura que guarda as instruções que estão sendo executadas no momento. Como a linguagem python tem como restrição não possuir multiconjuntos e nós podemos possuir mais de uma instrução igual(mais de uma leitura/escrita sendo solicitada pela mesma thread) ao mesmo tempo, eu simulo esse comportamento com um mapa que tem como chave a dupla {tipo da thread, id} e valor a quantidade de chamadas.

Funções

erro
Descrição: função que mostra uma mensagem de erro e a linha onde ocorreu o erro e para a execução do programa
Parâmetros: msg (a mensagem de erro)
Retorno: não possui

esp
Descrição: função que registra a entrada de uma instrução (dupla [tipo,id]) na **fila_execucao**
Parâmetros: tipo (0-leitor 1- escritor), id (id da thread)
Retorno: não possui

entra
Descrição: função que registra a saída de uma instrução (dupla [tipo,id]) da **fila_execucao** e sua entrada na estrutura de **instrucoes_ativas**, chama a função de **erro** se a instrução da entrada não for a primeira na **fila_execucao**
Parâmetros: tipo (0-leitor 1- escritor), id (id da thread)
Retorno: não possui

sai
Descrição: função que registra a saída de uma instrução (dupla [tipo,id]) da estrutura de **instrucoes_ativas** chama a função de **erro** se a instrução da entrada não estiver na estrutura
Parâmetros: tipo (0-leitor 1- escritor), id (id da thread)
Retorno: não possui

Testes Realizados

Teste 0

Entrada:

```
Numero de threads leitoras: 4
Numero de threads escritoras: 4
Numero de leituras: 2
Numero de escritas: 2
Nome do arquivo de log(.py): log0.py
```

Saída:

```
0.txt
     0
     3

1.txt
     0
     0

2.txt
     0
     3

3.txt
     0
     3
```

Tempo de execução: 0.0046913880

Log:

esp(0,0)
entra(0,0)
esp(0,2)
esp(0,3)
esp(1,0)
sai(0,0)
esp(0,1)
entra(0,2)
esp(1,2)
esp(1,1)
esp(1,3)
sai(0,2)
esp(0,0)
esp(0,2)
entra(0,3)
sai(0,3)
entra(1,0)
esp(0,3)
sai(1,0)
esp(1,0)
entra(0,1)
sai(0,1)
entra(1,2)
sai(1,2)
esp(0,1)
esp(1,2)
entra(1,1)
sai(1,1)
esp(1,1)
entra(1,3)
sai(1,3)
esp(1,3)

```
entra(0,0)
sai(0,0)
entra(0,2)
sai(0,2)
entra(0,3)
sai(0,3)
entra(1,0)
sai(1,0)
entra(0,1)
sai(0,1)
entra(1,2)
sai(1,2)
entra(1,1)
sai(1,1)
entra(1,3)
sai(1,3)
```

Saída do log:

```
Ordem de execuções das threads é válida
```

Teste 1

Entrada:

```
    Numero de threads leitoras: 8
    Numero de threads escritoras: 8
    Numero de leituras: 1
    Numero de escritas: 1
    Nome do arquivo de log(.py): log1.py
```

Saída:

```
0.txt
    0

1.txt
    0

2.txt
    0

3.txt
    0

4.txt
    0

5.txt
    1

6.txt
    3

7.txt
    7
```

```
Tempo de execução: 0.0058813450

Log:
    esp(0,0)
    entra(0,0)
    esp(0,1)
    sai(0,0)
    esp(0,3)
    entra(0,1)
    esp(0,4)
    esp(1,1)
    esp(0,5)
    esp(1,7)
    sai(0,1)
    esp(0,7)
    esp(1,4)
    esp(1,6)
    esp(1,0)
    esp(0,2)
    esp(1,5)
    esp(1,3)
    esp(0,6)
    entra(0,3)
    esp(1,2)
    sai(0,3)
    entra(0,4)
    sai(0,4)
    entra(1,1)
    sai(1,1)
    entra(0,5)
    sai(0,5)
    entra(1,7)
```

```
sai(1,7)
entra(0,7)
sai(0,7)
entra(1,4)
sai(1,4)
entra(1,6)
sai(1,6)
entra(1,0)
sai(1,0)
entra(0,2)
sai(0,2)
entra(1,5)
sai(1,5)
entra(1,3)
sai(1,3)
entra(0,6)
sai(0,6)
entra(1,2)
sai(1,2)
```

Saída do log:

```
Ordem de execuções das threads é válida
```

Teste 2

Entrada:

```
    Numero de threads leitoras: 10
    Numero de threads escritoras: 2
    Numero de leituras: 2
    Numero de escritas: 4
    Nome do arquivo de log(.py): log2.py
```

Saída:

```
0.txt
    0
    0

1.txt
    0
    0

2.txt
    0
    1

3.txt
    0
    1

4.txt
    0
    1

5.txt
    1
    1
```

```
6.txt
     0
     1

7.txt
     0
     1

8.txt
     0
     1

9.txt
     0
     0

Tempo de execução: 0.0204028560

Log:
     esp(0,0)
     entra(0,0)
     esp(0,1)
     esp(0,2)
     sai(0,0)
     entra(0,1)
     esp(0,3)
     sai(0,1)
     entra(0,2)
     esp(0,6)
     esp(0,8)
     esp(0,7)
     esp(1,0)
```

```
esp(0,0)
esp(0,9)
esp(0,1)
esp(0,4)
sai(0,2)
esp(1,1)
esp(0,2)
entra(0,3)
esp(0,5)
sai(0,3)
esp(0,3)
entra(0,6)
sai(0,6)
entra(0,8)
sai(0,8)
esp(0,6)
esp(0,8)
entra(0,7)
sai(0,7)
entra(1,0)
esp(0,7)
sai(1,0)
esp(1,0)
entra(0,0)
sai(0,0)
entra(0,9)
sai(0,9)
esp(0,9)
entra(0,1)
sai(0,1)
entra(0,4)
sai(0,4)
entra(1,1)
```

```
sai(1,1)
esp(1,1)
entra(0,2)
sai(0,2)
entra(0,5)
sai(0,5)
entra(0,3)
sai(0,3)
entra(0,6)
sai(0,6)
entra(0,8)
sai(0,8)
entra(0,7)
sai(0,7)
entra(1,0)
sai(1,0)
esp(1,0)
entra(0,9)
sai(0,9)
entra(1,1)
sai(1,1)
esp(1,1)
entra(1,0)
sai(1,0)
esp(1,0)
entra(1,1)
sai(1,1)
esp(1,1)
entra(1,0)
sai(1,0)
entra(1,1)
sai(1,1)
esp(0,4)
```

```
entra(0,4)
sai(0,4)
esp(0,5)
entra(0,5)
sai(0,5)
```

Saída do log:

```
Ordem de execuções das threads é válida
```

Teste 3

Entrada:

```
    Numero de threads leitoras: 4
    Numero de threads escritoras: 4
    Numero de leituras: 2
    Numero de escritas: 2
    Nome do arquivo de log(.py): log3.py
```

Saída:

```
0.txt
    0
    2

1.txt
    1
    1

2.txt
    1
    1

3.txt
    1
    1
```

Tempo de execução: 0.0065737480

Log:

```
    esp(0,0)
    entra(0,0)
    esp(1,1)
    esp(0,2)
```

```
esp(0,3)
esp(0,1)
esp(1,0)
sai(0,0)
esp(1,2)
esp(0,0)
entra(1,1)
esp(1,3)
sai(1,1)
esp(1,1)
entra(0,2)
sai(0,2)
entra(0,3)
esp(0,2)
sai(0,3)
entra(0,1)
esp(0,3)
sai(0,1)
entra(1,0)
esp(0,1)
sai(1,0)
esp(1,0)
entra(1,2)
sai(1,2)
esp(1,2)
entra(0,0)
sai(0,0)
entra(1,3)
sai(1,3)
esp(1,3)
entra(1,1)
sai(1,1)
entra(0,2)
```

```
sai(0,2)
entra(0,3)
sai(0,3)
entra(0,1)
sai(0,1)
entra(1,0)
sai(1,0)
entra(1,2)
sai(1,2)
entra(1,3)
sai(1,3)
```

Saída do log:

```
Ordem de execuções das threads é válida
```

Teste 4

Entrada:

```
Numero de threads leitoras: 4
Numero de threads escritoras: 4
Numero de leituras: 4
Numero de escritas: 4
Nome do arquivo de log(.py): log4.py
```

Saída:

```
0.txt
     0
     0
     0
     0

1.txt
     0
     1
     1
     1

2.txt
     0
     1
     1
     1

3.txt
     0
     1
     1
     1
```

```
Tempo de execução: 0.0063599660

Log:
    esp(0,0)
    entra(0,0)
    esp(0,1)
    sai(0,0)
    entra(0,1)
    esp(0,2)
    esp(0,3)
    esp(1,0)
    sai(0,1)
    esp(0,0)
    esp(1,3)
    esp(1,2)
    esp(1,1)
    esp(0,1)
    entra(0,2)
    sai(0,2)
    entra(0,3)
    sai(0,3)
    esp(0,2)
    entra(1,0)
    sai(1,0)
    esp(0,3)
    entra(0,0)
    esp(1,0)
    sai(0,0)
    esp(0,0)
    entra(1,3)
```

```
sai(1,3)
entra(1,2)
esp(1,3)
sai(1,2)
esp(1,2)
entra(1,1)
sai(1,1)
esp(1,1)
entra(0,1)
sai(0,1)
entra(0,2)
esp(0,1)
sai(0,2)
entra(0,3)
esp(0,2)
sai(0,3)
entra(1,0)
esp(0,3)
sai(1,0)
esp(1,0)
entra(0,0)
sai(0,0)
entra(1,3)
esp(0,0)
sai(1,3)
esp(1,3)
entra(1,2)
sai(1,2)
esp(1,2)
entra(1,1)
sai(1,1)
esp(1,1)
entra(0,1)
```

```
sai(0,1)
entra(0,2)
esp(0,1)
sai(0,2)
entra(0,3)
esp(0,2)
sai(0,3)
esp(0,3)
entra(1,0)
sai(1,0)
esp(1,0)
entra(0,0)
sai(0,0)
entra(1,3)
sai(1,3)
esp(1,3)
entra(1,2)
sai(1,2)
esp(1,2)
entra(1,1)
sai(1,1)
esp(1,1)
entra(0,1)
sai(0,1)
entra(0,2)
sai(0,2)
entra(0,3)
sai(0,3)
entra(1,0)
sai(1,0)
entra(1,3)
sai(1,3)
entra(1,2)
```

```
sai(1,2)
entra(1,1)
sai(1,1)
```

Saída do log:

```
Ordem de execuções das threads é válida
```