Jantar dos Filósofos

Gustavo Henrique Grützmann (19102053) Vitor Sorpile Geraldo (19102063)

I. INTRODUÇÃO

Este projeto tem como objetivo apresentar uma solução para o problema clássico "Jantar dos Filósofos", seguindo a descrição prevista na seção 2.5.1 (O problema do jantar dos filósofos) do livro Sistemas Operacionais Modernos, 3a edição, Andrew S. Tanembaum. Pearson, 2010. Foi desenvolvida uma solução genérica para N filósofos, que é determinado por um *define* no código. Na atividade, o programa foi executado com 5 e 10 Filósofos. A cada alteração de estado o programa mostra quais foram as alterações e deixa destacada qual foi a última alteração realizada.

II. DIAGRAMAS

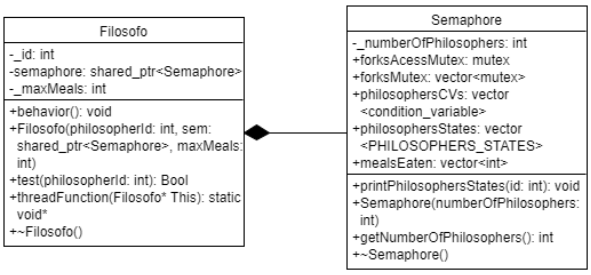

Fig. 1. Diagrama de Classes

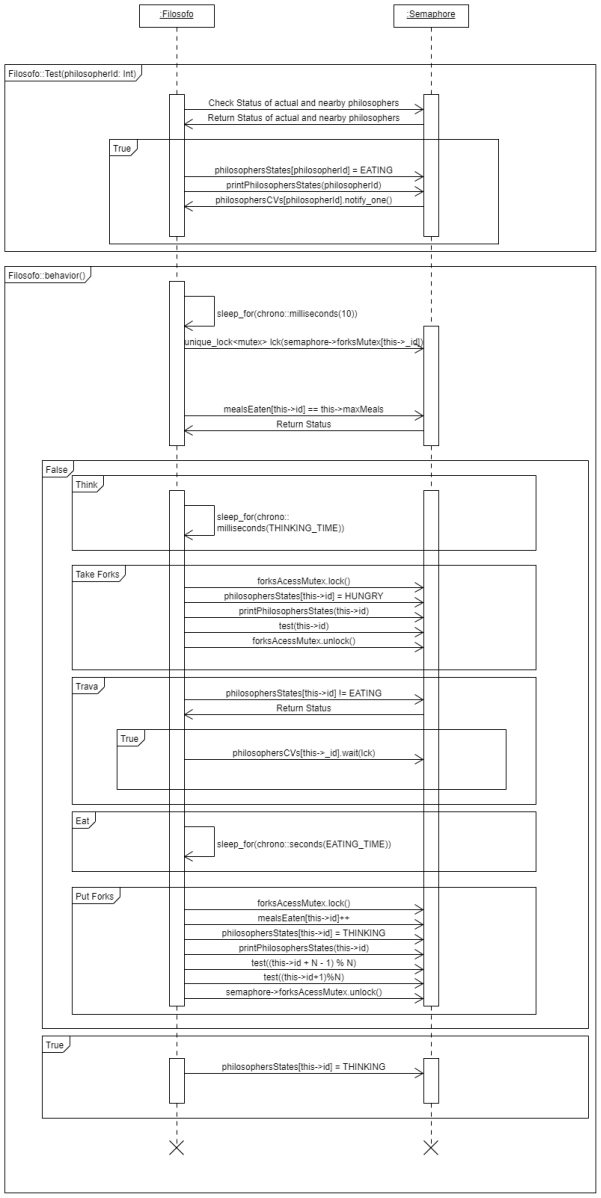

Fig. 2. Diagrama de Sequência da classe Filosofo

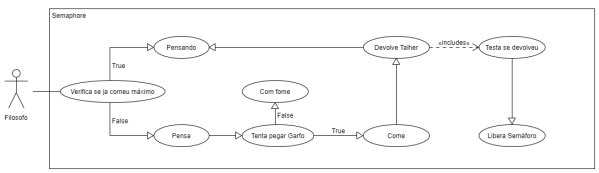

Fig. 3. Diagrama de Casos de Uso

III. Classes

A. Filosofo

Classe derivada de *std::thread* responsável por representar um Filósofo, possui um método *test()* que verifica se o filósofo correspondente ao id passado na função está com o estado *HUNGRY* e se os filósofos adjacentes não estão comendo, alterando seu estado para *EATING* e notificando sua variável condicional se estiver liberado para comer. Possui um ponteiro para um objeto da classe Semaphore, passado no construtor, e o método *behavior()*, implementado como um laço no qual o objeto fica um determinado tempo pensando, muda seu estado para *HUNGRY* e tenta pegar os dois garfos ao seu lado através do método *test()*, bloqueando sua variável condicional caso não consiga, após isso fica um tempo determinado comendo e depois devolve os garfos e verifica através do método *test()* se os filósofos adjacentes podem comer.

```
Filosofo 0 esta pensando e ja comeu 2 vezes.
Filosofo 1 esta pensando e ja comeu 2 vezes.
Filosofo 2 esta comendo  e ja comeu 1 vezes.
Filosofo 3 esta pensando e ja comeu 2 vezes.
Filosofo 4 esta comendo  e ja comeu 1 vezes.


Filosofo 0 esta pensando e ja comeu 2 vezes.
Filosofo 1 esta pensando e ja comeu 2 vezes.
Filosofo 2 esta pensando e ja comeu 2 vezes.
Filosofo 3 esta pensando e ja comeu 2 vezes.
Filosofo 4 esta comendo  e ja comeu 1 vezes.


Filosofo 0 esta pensando e ja comeu 2 vezes.
Filosofo 1 esta pensando e ja comeu 2 vezes.
Filosofo 2 esta pensando e ja comeu 2 vezes.
Filosofo 3 esta pensando e ja comeu 2 vezes.
Filosofo 4 esta pensando e ja comeu 2 vezes.
```

Fig. 4. Últimas 3 impressões de uma execução com 5 filósofos e 2 refeições

B. Semaphore

Classe responsável por controlar o acesso aos recursos compartilhados (garfos). Utiliza *mutexes* e variáveis condicionais para limitar o acesso às regiões críticas e controlar os estados dos filósofos. Seu construtor recebe o número de filósofos a serem usados no programa e inicializa as variáveis conforme necessário para o funcionamento correto. Também possui um método *get* para o número de filósofos e a função *printPhilosophersStates*, que imprime os estados de todos os filósofos, destacando o último que foi alterado.

IV. Compilação e Execução

O programa é compilado com um $Makefile$ e pode ser executado através do comando "./exercicio3 N", onde N representa o número de refeições a serem consumidas por cada filósofo.

V. Testes

A fim de verificar o correto funcionamento do programa, foram executados vários testes tanto com 5 filósofos quanto com 10 e em ambos sendo passados diferentes números de refeições a serem consumidas por cada um. As flags utilizadas na compilação foram *-Wall -Wextra -Wshadow -Werror -std=c++17 -pthread -pedantic -g* e para verificar se existiam erros de memória o programa foi executado com o *valgrind* com as flags *–leak-check=full –track-origins=yes -s*.

```
Filosofo 0 esta pensando e ja comeu 2 vezes.
Filosofo 1 esta comendo  e ja comeu 1 vezes.
Filosofo 2 esta pensando e ja comeu 2 vezes.
Filosofo 3 esta pensando e ja comeu 2 vezes.
Filosofo 4 esta pensando e ja comeu 2 vezes.
Filosofo 5 esta pensando e ja comeu 2 vezes.
Filosofo 6 esta pensando e ja comeu 2 vezes.
Filosofo 7 esta pensando e ja comeu 2 vezes.
Filosofo 8 esta comendo  e ja comeu 1 vezes.
Filosofo 9 esta pensando e ja comeu 2 vezes.


Filosofo 0 esta pensando e ja comeu 2 vezes.
Filosofo 1 esta pensando e ja comeu 2 vezes.
Filosofo 2 esta pensando e ja comeu 2 vezes.
Filosofo 3 esta pensando e ja comeu 2 vezes.
Filosofo 4 esta pensando e ja comeu 2 vezes.
Filosofo 5 esta pensando e ja comeu 2 vezes.
Filosofo 6 esta pensando e ja comeu 2 vezes.
Filosofo 7 esta pensando e ja comeu 2 vezes.
Filosofo 8 esta comendo  e ja comeu 1 vezes.
Filosofo 9 esta pensando e ja comeu 2 vezes.


Filosofo 0 esta pensando e ja comeu 2 vezes.
Filosofo 1 esta pensando e ja comeu 2 vezes.
Filosofo 2 esta pensando e ja comeu 2 vezes.
Filosofo 3 esta pensando e ja comeu 2 vezes.
Filosofo 4 esta pensando e ja comeu 2 vezes.
Filosofo 5 esta pensando e ja comeu 2 vezes.
Filosofo 6 esta pensando e ja comeu 2 vezes.
Filosofo 7 esta pensando e ja comeu 2 vezes.
Filosofo 8 esta pensando e ja comeu 2 vezes.
Filosofo 9 esta pensando e ja comeu 2 vezes.
```

Fig. 5. Últimas 3 impressões de uma execução com 10 filósofos e 2 refeições

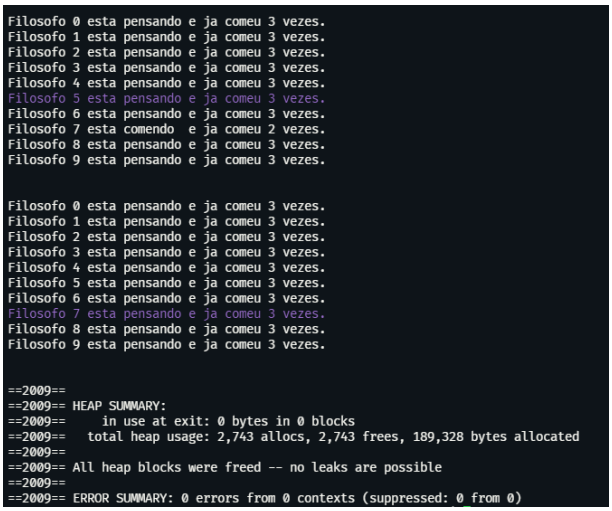

Fig. 6. Últimas 3 impressões de uma execução com o valgrind, 10 filósofos e 3 refeições