N-1 produtores e consumidor

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

I. INTRODUÇÃO

Este projeto tem como objetivo apresentar uma solução para o problema clássico "Produtor-Consumidor", porém limitando o número de consumidores e de refeições produzidas. Nesse contexto, foi desenvolvida uma solução genérica para N produtores e N consumidores que são determinados por *defines* no código e para cada *thread* produtora é passada uma função *static* da classe Chef e para as consumidoras é utilizado o operador () da classe Atendente.

II. DIAGRAMAS

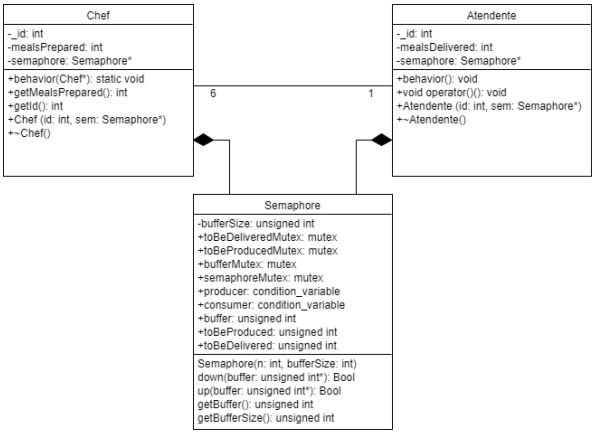

Fig. 1. Diagrama de Classes

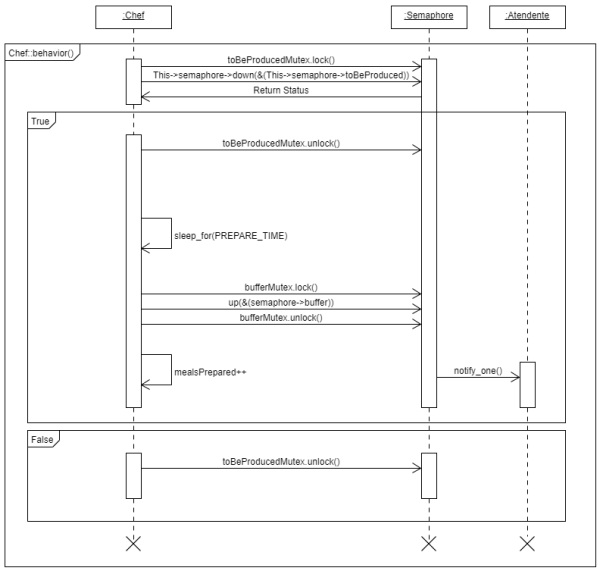

Fig. 2. Diagrama de Sequência da classe Chef

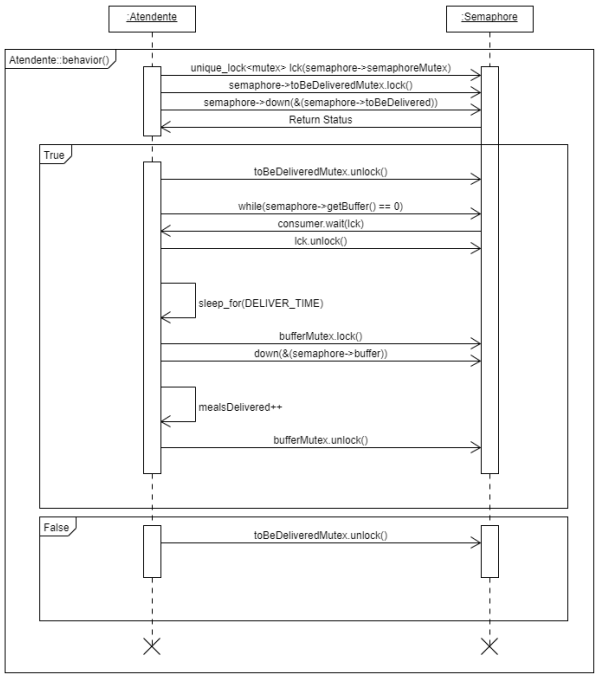

Fig. 3. Diagrama de Sequência da classe Atendente

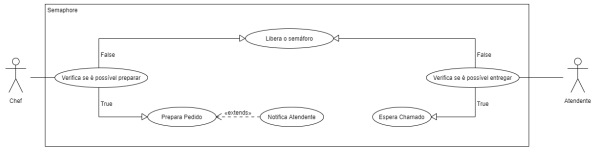

Fig. 4. Diagrama de Casos de Uso

III. CLASSES

A. Chef

Classe responsável por representar um produtor, possui dois métodos *get*, um para o id que é passado para o construtor e outro para o contador de refeições produzidas, um ponteiro para um objeto da classe Semaphore, também passado no construtor, e o método *static behavior()* que define seu comportamento, implementado como um laço no qual o objeto verifica através do semáforo se ainda é preciso produzir refeições, caso sim, produz e notifica um consumidor, caso não, quebra o laço.

B. Atendente

Classe responsável por representar um consumidor, possui um ponteiro para um objeto da classe Semaphore, passado no

construtor, e o método *behavior()* que define seu comportamento, implementado como um laço no qual o objeto verifica através do semáforo se ainda é preciso entregar refeições, caso sim, entrega ou espera até uma ser produzida, caso não, quebra o laço. Também possui a sobrecarga do operador () que chama o método *behavior()* e é usado para definir o que é executado pela *thread*.

C. Semaphore

Classe responsável por controlar o acesso aos recursos compartilhados (buffer entre produtores e consumidores, contador de refeições a serem produzidas e contador de refeições a serem entregues). Utiliza *mutexes* e variáveis condicionais para limitar o acesso às regiões críticas e notificar os consumidores quando uma refeição é produzida e possui os métodos *down* e *up* para controlar se é possível retirar/acrescentar ao buffer e dois métodos *get*, um para o tamanho do buffer e outro para o buffer em si.

IV. Compilação e Execução

O programa é compilado com um $Makefile$ e pode ser executado através do comando "./exercicio1 N", onde N é um argumento opcional, por padrão assume o valor 50, que representa quantas refeições serão produzidas e consumidas.

V. Testes

A fim de verificar o correto funcionamento do programa, foram executados vários testes com diferentes números de refeições a serem produzidas e consumidas. 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*.

```
vitorsorpile@DESKTOP-KNUPDEI:~/UFSC/SO/1_N-1_produtores_e_consumidor$ ./exercicio1
Chefe 1 produziu 9
Chefe 2 produziu 9
Chefe 3 produziu 8
Chefe 4 produziu 8
Chefe 5 produziu 8
Chefe 6 produziu 8
O chefe menos ocioso foi o 1
O chefe mais ocioso foi o 3
```

Fig. 5. Saída padrão

```
vitorsorpile@DESKTOP-KNUPDEI:~/UFSC/SO/1_N-1_produtores_e_consumidor$ ./exercicio1 150
Chefe 1 produziu 25
Chefe 2 produziu 26
Chefe 3 produziu 26
Chefe 4 produziu 25
Chefe 5 produziu 24
Chefe 6 produziu 24
O chefe menos ocioso foi o 2
O chefe mais ocioso foi o 5
```

Fig. 6. Saída com 150 refeições

```
valgrind --leak-check=full --track-origins=yes -s ./exercicio1
==22732== Memcheck, a memory error detector
==22732== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==22732== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==22732== Command: ./exercicio1
==22732==
Chefe 1 produziu 8
Chefe 2 produziu 8
Chefe 3 produziu 9
Chefe 4 produziu 9
Chefe 5 produziu 9
Chefe 6 produziu 7
O chefe menos ocioso foi o 3
O chefe mais ocioso foi o 6
==22732==
==22732== HEAP SUMMARY:
==22732==     in use at exit: 0 bytes in 0 blocks
==22732==   total heap usage: 31 allocs, 31 frees, 76,256 bytes allocated
==22732==
==22732== All heap blocks were freed -- no leaks are possible
==22732==
==22732== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Fig. 7. Saída com o valgrind