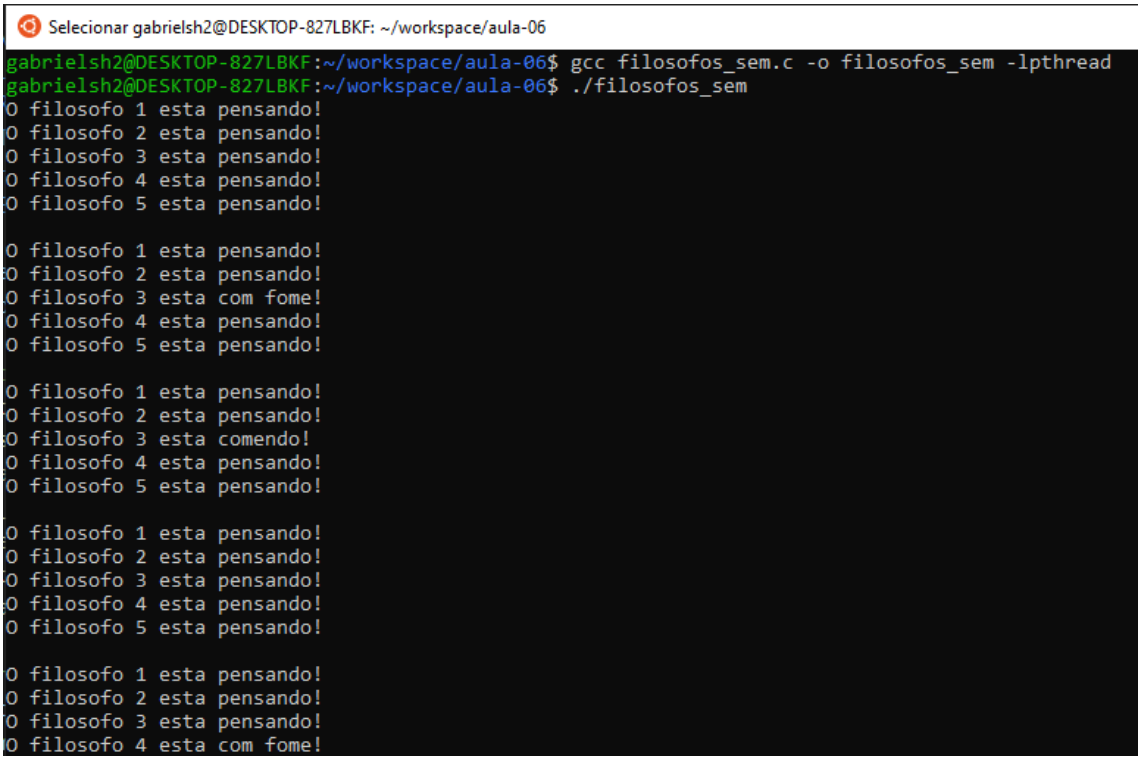
Atividade da Aula 06 (15/09/2021)

Aluno: Gabriel Hoffmann

Curso: Ciência da Computação

Execute os programas feitos em MUTEX e semáforos e descreva em poucas palavras as diferenças na lógica da programação observada:

O exercício usando semáforos utiliza uma lógica em que consiste em criar um array de 5 estados que pode ter os valores das constantes THINKING (0), HUNGRY (1) e EATING (2) que significam o estado de cada filósofo. Os 5 filósofos são criados no estado THINKING e é gerado um tempo aleatório em que permanecem nesse estado. Após o fim desse tempo o estado do filósofo muda para HUNGRY e é verificado se os filósofos adjacentes estão no estado EATING para, pois só é permitido comer se os garfos estiverem disponíveis e se estiverem disponíveis o estado do filósofo muda para EATING. Após um tempo aleatório comendo, o filósofo volta para THINKING e é feito o teste com os filósofos adjacentes. O processo repete em looping até ser cancelado



No exercício utilizando MUTEX temos um número máximo de 40 refeições, um array de garfos que começam todos disponíveis, um array de id para os filósofos, um array de refeições feitas separadas por filósofos e um array do número de tentativas também separado por filósofos. Pelas propriedades já vemos que a lógica abordada será bem diferente da primeira.

Os filósofos vão sempre olhar para a disponibilidade do garfo da esquerda e direita que é definida por 0 e 1 na posição do garfo no array de garfos. Caso os garfos estejam disponíveis (valor 1) o filósofo fica um tempo aleatório comendo e ocupando os dois garfos, após finalizar os garfos são liberados e o filósofo descansa por um tempo aleatório.

```
gabrielsh2@DESKTOP-827LBKF:~/workspace/aula-06$ gcc filosofos_mutex.c -o filosofos_mutex -lpthread
gabrielsh2@DESKTOP-827LBKF:~/workspace/aula-06$ ./filosofos_mutex
Filosofo 0: Eu estou tentando comer!
Filosofo 0: esquerda=1
Filosofo 0: Peguei o garfo esquerdo!
Filosofo 0: Direita=1
Filosofo 0: Peguei os dois garfos!
Filosofo 0: Eu estou comendo!

Filosofo 2: Eu estou tentando comer!
Filosofo 2: esquerda=1
Filosofo 2: Peguei o garfo esquerdo!
Filosofo 2: Direita=1
Filosofo 2: Peguei os dois garfos!
Filosofo 2: Eu estou comendo!

Filosofo 1: Eu estou tentando comer!
Filosofo 1: esquerda=0
Filosofo 1: Não pode pegar o garfo esquerdo!

Filosofo 3: Eu estou tentando comer!
Filosofo 3: esquerda=0
Filosofo 3: Não pode pegar o garfo esquerdo!

Filosofo 4: Eu estou tentando comer!
Filosofo 4: esquerda=1
Filosofo 4: Peguei o garfo esquerdo!
Filosofo 4: Direito=0
```

Questões Teóricas

Em um sistema computacional multiprocessado, onde o sistema operacional realiza escalonamento de tarefas do tipo preemptivo, três processos (P1, P2 e P3) compartilham recursos (R1, R2 e R3). Os processos P1 e P2 concorrem entre si ao acesso do recurso R1, enquanto P2 e P3 concorrem entre si ao acesso dos recursos R2 e R3. Os recursos R1 e R3 são preemptíveis, ou seja, podem sofrer preempção; R2 é um recurso não preemptível. Todos os três processos usam o mesmo mecanismo de exclusão mútua para garantir acesso exclusivo em suas seções críticas. Com base nesse cenário, é correto afirmar que:

a) Não é possível ocorrer deadlock entre os três processos.

Analise as seguintes afirmativas.

I. Condições de corrida podem ocorrer se múltiplas *threads* fazem leituras de um dado compartilhado, mesmo que nenhuma realize escritas.

II. O uso de *mutex* para a exclusão mútua em seções críticas garante que não haja condição de corrida, porém pode ocasionar *deadlocks* se não for corretamente empregado.

III. Monitores são baseados em um tipo abstrato de dados e um controle de acesso aos dados. Apenas funções do monitor acessam os dados e apenas uma *thread* ou processo pode executar funções de um monitor por vez.

IV. Semáforos têm duas operações, P( ) e V( ), sendo que apenas a operação P( ) pode bloquear um processo ou *thread*.

A análise permite concluir que

d) apenas as afirmativas II, III e IV são verdadeiras.