Experimento #3

Sistemas Operacionais A

Introdução

Neste experimento será analisado o funcionamento dos semáforos e o funcionamento do conceito de exclusão mutua.

Um Semáforo é ferramenta utilizada para o sincronismo entre processos que garante que apenas um processo pode ter acesso a determinado recurso crítico.

Exclusão mútua é uma forma de assegurar que apenas um processo tenha acesso a uma variável ou arquivo compartilhado por vez.

Erros de sintaxe e lógica iniciais

Primeira Parte:

Linha 17: Inclusão da biblioteca stdlib.h;

Linha 74: Inicialização da variável i, responsável pelo controle de iterações simples;

Linhas 135/145: Reorganização da atribuição de valores para as structs g_sem_op1 e g_sem_op2. Também foi **corrigido o fato** da variável g_sem_op2 não ter obtido nenhum valor;

Linha 167: Modificação do valor respectivo à atribuição de permissões para a memória compartilhada que estava sendo criada;

Linha 139: Foi modificado o comando "exit" para "break";

Linha 163: Foi modificado o modo de matar os filhos. Diferente de antes, que era manualmente, agora utiliza-se uma iteração que sempre irá acompanhar o número de filhos que foram criados;

Linha 211: Tempo de dormência aumentado de 200 para 400 microsegundos;

Linha 241: Correção do parâmetro respectivo à operação que seria feita pelo comando "semop", já que anteriormente o semáforo continuaria livre, quando na verdade teria que estar sendo bloqueado neste ponto;

Linha 259: Correção do parâmetro que seria utilizado no print, de "%f7" para "%c", pois o vetor é composto por caracteres;

Segunda Parte:

Primeiramente o número de filhos foi aumentado de 2 para 8, depois foram criados mais dois semáforos, sendo o original do exemplo usado pelos filhos produtores, outro pelos filhos consumidores e outro para gerenciar acesso ao *Buffer* recém-criado, desses todos foram abertos pelo pai, permitindo acesso, com exceção do semáforo dos consumidores.

Foram criadas duas funções novas Produzir e Consumir para cada tipo de filho respectivamente, o ponto de acesso dessas funções ao *Buffer* de memória compartilhada por cada tipo é controlado por dois espaços de memória compartilhada de tamanho int, sendo um para consumidor e outro para produtor. Sendo os semáforos responsáveis por cuidar que cada filho acesse o *Buffer* sozinho e na seguinte ordem de produtor seguido por consumidor.

Resultados da execução do programa exemplo

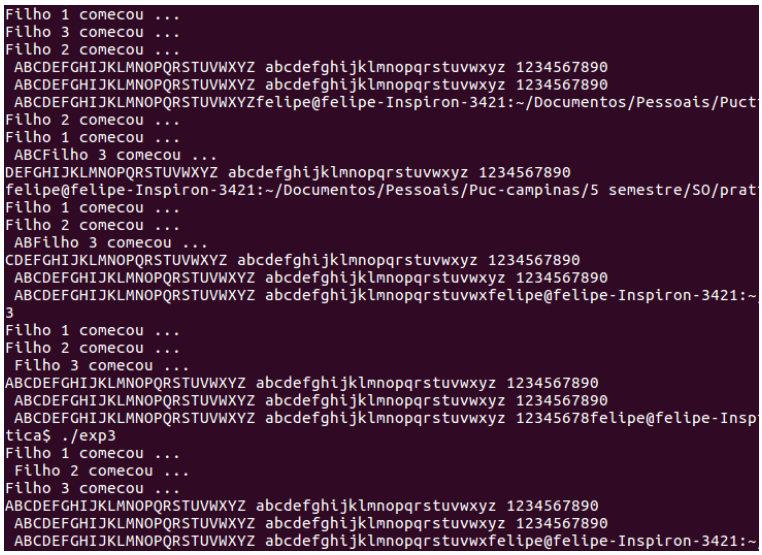

Figura 1: Programa exemplo com semáforos

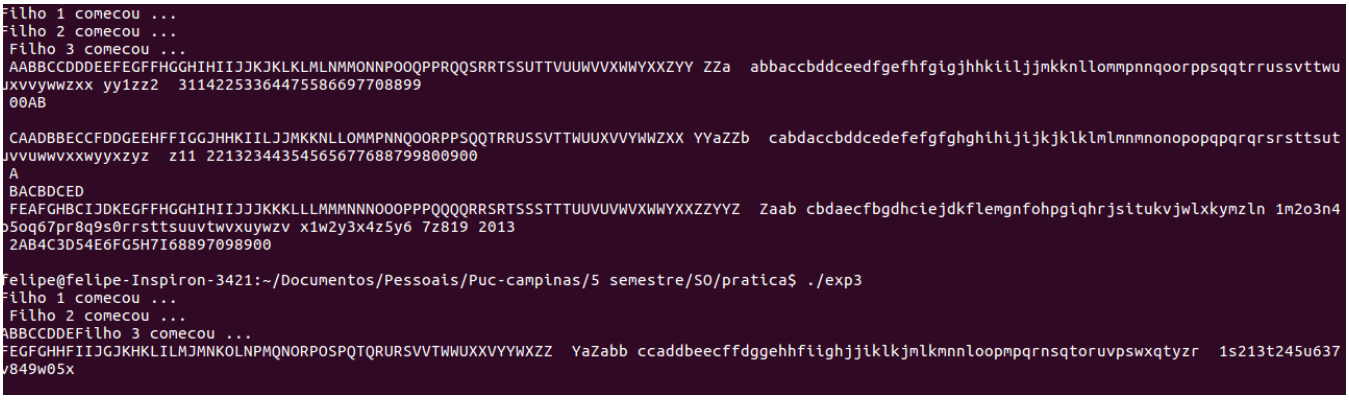

Figura 2: Programa Exemplo sem semáforo

Resolução dos exercícios

1) Uma região por ser crítica tem garantida a exclusão mútua? Justifique.

Não, pois a exclusão mutua é estipulado pelo programador, por ser uma técnica para resolver a disputa entre processos, mas não é necessariamente tratada na região critica.

2) É obrigatório que todos os processos que acessam o recurso crítico tenham uma região crítica igual?

Não. Dizer que todos são obrigados a ter uma região crítica igual é um equívoco, pois podem existir processos que acessam outros recursos ou que nem cheguem a acessar a região crítica.

3) Porque as operações sobre semáforos precisam ser atômicas?

Para que não ocorra nenhum tipo de interrupção enquanto os processos estão executando, já que nesse caso, se faz necessário que o processo comece e termine sem que seja escalonado.

4) O que é uma diretiva ao compilador?

É um comando que define se uma parte específica do código será compilada ou não, sendo dirigido ao pré-processador.

5) Porque o número é pseudoaleatório e não totalmente aleatório?

Porque é gerado por um processo determinístico, ou seja, um número obtido de alguma forma variada passa por diversas contas para ter seu valor modificado, de maneira que se torne praticamente aleatório. Um número pseudoaleatório é mais fácil de gerar do que um número totalmente aleatório.

Resolução dos exercícios do código

Pergunta 1: Se usada a estrutura g_sem_op1 terá qual efeito em um conjunto de semáforos?

Há estrutura g_sem_op1 após as mudanças necessitarias possui os valores que são usados pelo semop para deixar os semáforos com sinal de aberto;

Pergunta 2: Para que serve esta operação semop(), se não está na saída de uma região crítica?

A função semop está sendo usada apenas para preparar o semáforo, ela é usada para efetuar operação em um semáforo.

Pergunta 3: Para que serve essa inicialização da memória compartilhada com zero?

Para evitar que haja lixo no seu endereço.

Pergunta 4: se os filhos ainda não terminaram, semctl e shmctl, com o parametro IPC-RMID, não permitem mais o acesso ao semáforo / memória compartilhada?

Só é possível acessar o semáforo ou memória compartilhada nesta ocasião caso ocorra falha na remoção.

Pergunta 5: quais os valores possíveis de serem atribuídos a number?

O que define os valores que serão obtidos basicamente é o módulo colocado logo após a operação que modifica o número obtido. A operação módulo limita os números a serem obtidos, fazendo com que estejam no intervalo de 0 até “n”, sendo “n” o segundo parâmetro da operação. Nesse caso, como se trata de quantidade, soma-se 1 ao valor obtido para que nunca seja 0

Resultados da Tarefa 2

Com o PROTECT definido:

```
Filho 1 comecou como produtor ...
Filho 3 comecou como produtor ...
Filho 5 comecou como consumidor...
Filho 2 comecou como produtor ...
Filho 7 comecou como consumidor...
Filho 4 comecou como produtor ...
Filho 6 comecou como consumidor...
Filho 1------------
Caracteres produzidos: Filho 8 comecou como consumidor...
ABC
Indice: 3, indice global produtor: 0
--------------------
Filho 5------------
Caracteres consumidos: ABC
tmp_index: 3, indice global: 0
--------------------
Filho 3------------
Caracteres produzidos: D
Indice: 4, indice global produtor: 3
--------------------
Filho 7------------
Caracteres consumidos: D
tmp_index: 4, indice global: 3
--------------------
Filho 1------------
Caracteres produzidos: EFGH
Indice: 8, indice global produtor: 4
--------------------
Filho 5------------
Caracteres consumidos: EF
tmp_index: 6, indice global: 4
--------------------
Filho 3------------
Caracteres produzidos: IJKL
Indice: 12, indice global produtor: 8
--------------------
Filho 7------------
Caracteres consumidos: GH
tmp_index: 8, indice global: 6
--------------------
Filho 1------------
Caracteres produzidos: MNO
Indice: 15, indice global produtor: 12
--------------------
```

Sem o PROTECT definido:

```
Filho 1 comecou como produtor ...
Filho 2 comecou como produtor ...
Filho 3 comecou como produtor ...
Filho 4 comecou como produtor ...
Filho 5 comecou como consumidor...
Filho 6 comecou como consumidor...
Filho 7 comecou como consumidor...
Filho 1------------
Filho 8 comecou como consumidor...
Caracteres produzidos: Filho 2------------
Caracteres produzidos: Filho 3------------
Caracteres produzidos:
Indice: 1, indice global produtor: 0
AC
--------------------
Filho 3------------
Indice: 3, indice global produtor: 0
Caracteres produzidos: --------------------

Indice: 3, indice global produtor: 3
Filho 1------------
--------------------
Caracteres produzidos: Filho 3------------
Caracteres produzidos:
Indice: 5, indice global produtor: 3
--------------------
Filho 3------------
Caracteres produzidos: BD
Indice: 5, indice global produtor: 5
--------------------
Filho 2------------
Caracteres produzidos: Filho 4------------
Caracteres produzidos: E
Indice: 5, indice global produtor: 5
--------------------
F
Filho 1------------
Caracteres produzidos: Indice: 8, indice global produ
tor: 5
--------------------
Filho 3------------
Caracteres produzidos: G
Indice: 8, indice global produtor: 8
--------------------
Filho 2------------
Caracteres produzidos: H
Indice: 9, indice global produtor: 8
--------------------
Filho 4------------
Caracteres produzidos:
Indice: 10, indice global produtor: 9
```

Análise dos Resultados

Foi observado que os resultados variam de uma forma bem expressiva quando submetidos ao uso dos semáforos. Sem os semáforos, é possível verificar resultados sem ordem, totalmente aleatórios. Isso se deve porque os diversos processos estão acessando os mesmos recursos de maneira concorrente e, eventualmente, irão atualizar os mesmos de maneira errada e sem coerência, provocando assim inconsistência nos dados.

Já com o uso de semáforos, todos os processos seguem uma ordem de acesso, impedindo que haja incoerência na atualização dos recursos. Deste modo, é possível observar resultados plenamente legíveis, o que indica que há uma cooperação ordenada entre os processos, mesmo que estejam executando suas tarefas “simultaneamente”.

Conclusão

Foi possível concluir que os semáforos exercem um grande papel ao evitar que dados que podem ser acessados por múltiplos processos se tornem inconsistentes. Isso se deve ao fato de que são justamente os semáforos que garantem a exclusão mútua, que deve estar presente para que haja ordem no seu acesso, permitindo que apenas um processo consiga utilizar o recurso compartilhado de cada vez.

Alunos:

Raissa Furlan Davinha RA: 15032006

José Carlos Clausen Neto RA: 15055825

Luiz Felipe Masson Zerbetto RA: 15166804

Erick Felipe Campo Dall'orto de Souza RA: 13052170