Sistemas operacionais: Conceitos e Mecanismos

INSTITUTO FEDERAL
Santa Catarina

Capítulo 11: Mecanismo de coordenação

Integrantes do grupo: Jefferson Botitano e Leonardo Ludvig
Professor: Arliones Stevert Hoeller Junior
Disciplina: Sistemas operacionais

Campus São José

Semáforos e Mutexes

INSTITUTO FEDERAL
Santa Catarina

Mecanismo de coordenação eficiente e flexível para o controle de exclusão mútua entre n tarefas e outras aplicações,contudo a manipulação deste mecanismo é feita apenas pelas operações atômicas.

- Up(s):Incrementa o contador interno e faz a verificação caso contador negativo ou nulo significa que existe pelo menos uma tarefa suspensa,sendo assim a primeira tarefa da fila é devolvida para fila de tarefas prontas e pronta para ser executada.
- Down(s):Decrementa o contador interna e faz a verificação caso contador negativo, a tarefa solicitante é adicionada à fila do semáforo e suspendida.Caso não seja negativo a tarefa prossegue coma e sua execução.
- Eficiências: tarefas ficam suspensas nas filas de semáforos e quando recebe “sinal verde” apenas a primeira tarefa da fila é acordada.
- Justiça:as tarefas respeitam geralmente a norma de FIFO,sendo assim respeita ordem de chegada.Em outros casos de diferentes sistemas operacionais podem ser fila de prioridades.
- Independências:apenas as tarefas que solicitaram ao semáforo pela operação down(s) são consideradas na decisão de quem irá o obtê-lo,em relação a quem será a próxima tarefa ela independe de uma coordenação central e tarefas que não estejam envolvidas no acesso e apenas a lógica de programação que foi utilizada para realizar o semáforo.

Mutexes são versão simplificado do semáforo onde o contador apenas assume dois valores: 1 para livre e 0 para ocupado.

Suas aplicações podem ser para poder travar o mutex logo em seguida executar a seção crítica e destravar o mutex que seria o equivalente ao uso do up(s) e down(s).

Variáveis de condição

INSTITUTO FEDERAL
Santa Catarina

As variáveis de condição são outro mecanismo de sincronização de uso frequente na solução para os casos de acesso simultâneo a seções críticas, sendo sempre associada a uma condição lógica que pode ser aguardada por uma tarefa, como uma tarefa que está esperando o contador de outra tarefa chegar a zero por exemplo, ao invés de ser testado continuamente o estado do contador da outra tarefa é apenas avisado pela variável de condição quando o contador chega a zero, mantendo a tarefa que está aguardando a condição “dormindo” economizando processamento.

Uma variável de condição não possui a condição propriamente dita, ela está associando a condição de uma tarefa ao acionamento de outra tarefa, ou seja, ela aguarda uma determinada tarefa satisfazer sua condição para avisar a tarefa que está suspensa aguardando a condição, para isso a variável tem uma fila de tarefas que aguardam a condição e deve ser usada em conjunto com um mutex para garantir a exclusão mútua sobre o estado da condição representada pela variável.

Monitores

INSTITUTO FEDERAL
Santa Catarina

Para utilizar semáforos ou mutexes é necessário que o programador identifique explicitamente os pontos de sincronização em seu programa, porém para programas maiores e mais complexos essa abordagem acaba perdendo a sua eficiência pois se torna inviável identificar todos os pontos de sincronização no código. Para contornar o problema de esquecer a liberação de um semáforo já alocado, foram criados os monitores, estruturas de sincronização que solicitam e liberam o acesso de uma seção crítica associada a um recurso de forma transparente. Um monitor é constituído dos seguintes elementos.

- Um recurso compartilhado, visto como um conjunto de variáveis internas ao monitor.
- Um conjunto de procedimentos e funções que permitem o acesso a essas variáveis.
- Um semáforo ou mutex para garantir controle de exclusão mútua ao acessar o recurso.
- Um invariante sobre o estado interno do recurso, ou seja, uma condição sobre as variáveis internas do monitor que deve ser sempre verdadeira, que faz uma verificação formal para garantir a consistência do monitor.

2. Por que não existe operações read(s) e write(s) para ler ou ajustar o valor atual de um semáforo?

INSTITUTO FEDERAL
Santa Catarina

Caso seja implementado o write que modifica o contador interno sem verificar o controle da fila de uma tarefa precisa ser suspensa ou liberar uma tarefa para execução se quebra a lógica da implementação do semáforo.

Caso seja implementado o read para decisão de acesso a seção crítica pode tornar a própria leitura uma nova seção crítica pois diferente das operação up e down que são implementadas de forma atômica garantindo assim a operação sem ser interrompido.

3.Mostre como pode ocorrer violação da condição de exclusão mútua se as operações down(s) e up(s) sobre semáforos não forem implementadas de forma atômica.

INSTITUTO FEDERAL
Santa Catarina

Se as operações não forem implementada de forma atômica, pode ocorrer uma violação da exclusão mútua entre o processo de incrementar/decrementar e testar o contador, gerando assim um erro de acesso simultâneo a uma seção crítica.

4.Em que situações um semáforo deve ser inicializado em 0, 1 ou n > 1?

INSTITUTO FEDERAL
Santa Catarina

Caso inicializado em 0: caso receber um down a tarefa será bloqueada,até que outra tarefa de um up.

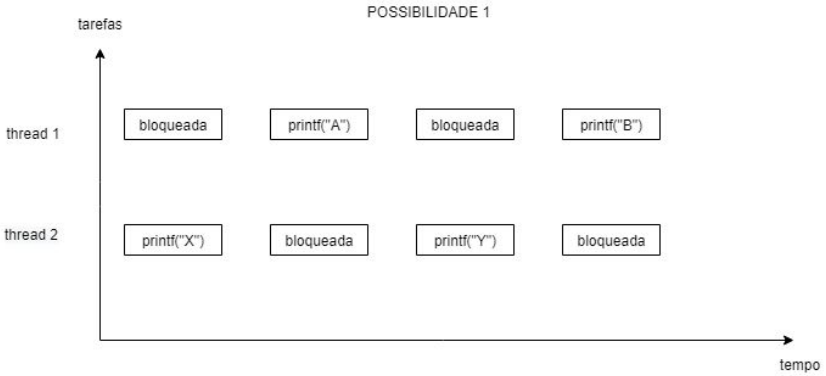
Caso inicializado em 1: caso receber um down a primeira tarefa poderá seguir com sua execução.

Caso inicializado maior que 1: significa que se pode ter mais de uma tarefa acessando a seção crítica simultâneamente.

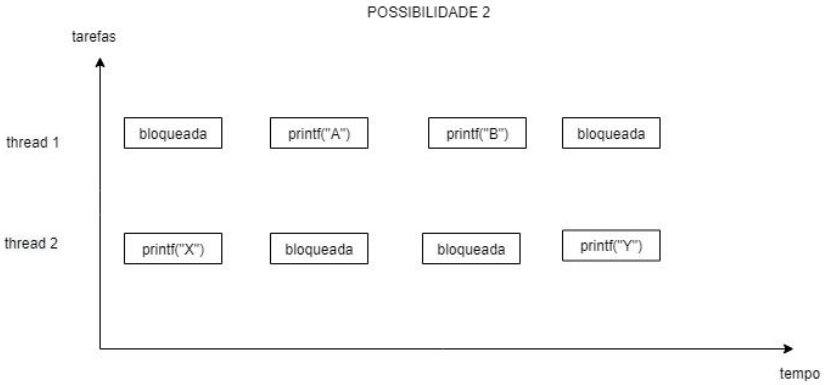
5.Desenhe o diagrama de tempo da execução e indique as possíveis saídas para a execução concorrente das duas threads cujos pseudo-códigos são descritos a seguir. Os semáforos s1 e s2 estão inicializados com zero (0).

INSTITUTO FEDERAL
Santa Catarina

```
thread1 ()
{
    down (s1) ;
    printf ("A") ;
    up (s2) ;
    printf ("B") ;
}
```




```
thread2 ()
{
    printf ("X") ;
    up (s1) ;
    down (s2) ;
    printf ("Y") ;
}
```



Existem duas possibilidades de diagrama de tempo pelo fato de depender do escalonador do sistema.

Primeiramente será impresso na tela “X” da thread2 enquanto a thread1 aguarda no semáforo s1.

Segundamente será impresso na tela “A” da thread1 pelo fato da thread2 ter dado up no semáforo s1.

Terceiramente que dependerá do escalonador do sistema operacional que resultará em duas possibilidades de ser impresso na tela “Y” da thread2 e em seguida sendo impresso na tela “B” da thread1 e como segunda possibilidade o processo ao contrário.