GSI018 – SISTEMAS OPERACIONAIS

Operating Systems – William Stallings – 7th Edition

Chapter 05 – Mutual Exclusion and Synchronization

REVIEW QUESTIONS

https://quizlet.com/233237383/chapters-5-8-flash-cards/

5.1 List four design issues for which the concept of concurrency is relevant.

Comunicação entre processos, compartilhamento e competição por recursos, sincronização de atividades entre múltiplos processos e alocação de tempo do processador para processos.

5.2 What are three contexts in which concurrency arises?

Aplicações múltiplas, aplicações estruturadas e estrutura dos sistemas operacionais.

5.3 What is the basic requirement for the execution of concurrent processes?

A habilidade de garantir a exclusão mútua.

5.4 List three degrees of awareness between processes and briefly define each.

Processos se desconhecem, são processos independentes que não trabalharão juntos. Processos se conhecem indiretamente, trata-se de processos que compartilham o mesmo objeto, como um E/S buffer. Processos se conhecem diretamente, são processos capazes de comunicarem entre si através de seus ID's e são desenhados para trabalharem juntos em uma atividade.

5.5 What is the distinction between competing processes and cooperating processes?

Processos concorrentes precisam acessar o mesmo recurso ao mesmo tempo, por exemplo um arquivo ou disco. Processos cooperativos compartilham acesso a um mesmo objeto, como buffer de memória, ou são capazes de se comunicarem, e cooperam na execução de uma aplicação ou trabalho.

5.6 List the three control problems associated with competing processes and briefly de-fine each.

Exclusão mútua: espera ocupa é empregada, o que gera consumo de processador. Starvation: alguns dos processos esperando para entrar na região crítica podem ter que esperar indefinidamente. Deadlock: quando processos com prioridades diferentes estiverem acessando a região crítica e o outro solicitando a região crítica.

5.7 List the requirements for mutual exclusion.

Garantir que 02 ou mais processos não estejam simultaneamente dentro de suas regiões críticas referentes ao mesmo recurso compartilhado. Mútua exclusão deve ser contemplada de forma independente da velocidade relativa dos processos ou mesmo número de

processadores. Nenhum processo executando fora da região crítica pode bloquear outro processo. Nenhum processo espere um tempo arbitrariamente longo para executar uma região crítica.

5.8 What operations can be performed on a semaphore?

O semáforo pode ser inicializado com valor negativo. A operação *wait(s)* decrementa o valor do semáforo e caso o valor tenha se tornado negativo o processo executando a operação é bloqueado. A operação *signal(s)* incrementa o valor do semáforo e testa se o valor é 0 ou negativo e em caso afirmativo um processo bloqueado é desbloqueado.

5.9 What is the difference between binary and general semaphores?

Um semáforo binário assume somente dois valores, 0 e 1. Já um semáforo geral pode assumir valores positivos, zero e negativos.

PROBLEMS

5.4 Consider the following program.

```
const int n = 50;
int tally;
 void total( ) {
        int count;
        for( count = 1; count<= n; count++) {
                tally++;
        }
}
void main( ) {
        tally = 0;
        parbegin (total (), total () );
        write (tally);
}
```

a) Determine the proper lower bound and upper bound on the final value of the shared variable tally output by this concurrent program. Assume processes can execute at any relative speed and that a value can only be incremented after it has been loaded into a register by a separate machine instruction.

Analisando o código, e as chamadas de funções, podemos assumir que tally estará entre 2<= tally <= 100. Os passos são seguidos dessa forma:

a. Tally recebe zero, é chamada a função parbegin. Processo A executa um total() e processo B executa outro.
b. Processo A incrementa tally, mas perde o controle para o Processo B que executa o laço for na função total().
c. Processo A volta a ser executado e guarda o valor 1 em tally.
d. Processo B volta a ser executado e guarda o valor 1 em tally.
e. Processo A executa as 49 remanescentes operações de soma, ficando com 50.
f. Processo B soma mais 1 em seu registrador.

b. Suppose that an arbitrary number of these processes are permitted to execute in parallel under the assumptions of part (a). What effect will this modification have on the range of final values of tally ?

Nesse caso, temos que o mínimo valor de tally será 2. Já o máximo é o número de processos vezes 50 (N*50).

2<= tally <= 50 * número de processos

5.12 Consider the following definition of semaphores. Compare this set of definitions with that of Figure 5.3 . Note one difference: With the preceding definition, a semaphore can never take on a negative value. Is there any difference in the effect of the two sets of definitions when used in programs? That is, could you substitute one set for the other without altering the meaning of the program?

```
void semWait( s ) {
        if( s.count > 0 ) {
                s.count--;
        } else {
                place this process in s.queue;
                block;
        }
}
void semSignal( s ) {
        if( there is at least one process blocked on semaphore s) {
                remove a process P from s.queue;
                place process P on ready list;
        }
        else s.count++;
}
```

Ambos funcionam da mesma forma. A diferença está na definição da figura 5.3 que permite verificar quantos processos estão na lista de espera.