GSI018 – SISTEMAS OPERACIONAIS

Operating Systems – William Stallings – 7th Edition
Chapter 05 – Mutual Exclusion and Synchronization

Pedro Henrique Silva Santana – 12011BSI218 – pedro.santana@ufu.br
Victor Hugo Martins Alves – 12011BSI217 – victor.alves1@ufu.br

REVIEW QUESTIONS

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

O requerimento básico para execução de processos concorrentes é o princípio da exclusão mútua; que é a habilidade de excluir todos os demais processos do seguimento de ação enquanto um processo é garante essa operação.

5.7 List the requirements for mutual exclusion.

1. A exclusão mútua deve ser aplicada: Apenas um processo por vez é permitido em sua seção crítica, entre todos os processos que possuem seções críticas para o mesmo recurso ou objeto compartilhado.
2. Um processo que para em uma seção não crítica deve fazê-lo sem interferir com outros processos.
3. Não deve ser possível que um processo que exija acesso a uma seção crítica seja atrasado indefinidamente: sem deadlock ou starvation.
4. Quando nenhum processo estiver em uma seção crítica, qualquer processo que solicite entrada em sua seção crítica deve ser permitido para entrar sem delay.
5. Nenhuma suposição é feita sobre as velocidades relativas do processo ou o número de processadores.
6. Um processo permanece dentro de sua seção crítica apenas por um tempo finito.

5.8 What operations can be performed on a semaphore?

1 O semáforo pode ser inicializado com valor negativo;
2 A operação wait decrementa o valor do semáforo e caso o valor se torne negativo o processo executando a operação é bloqueado;
3 A operação signal incrementa o valor do semáforo e testa se o valor é igual a 0 ou negativo e, caso seja, um processo bloqueado é desbloqueado.

5.9 What is the difference between binary and general semaphores?

A diferença entre o sistema geral e binário dos semáforos é que o sistema geral pode assumir valores positivos, o zero e negativos; enquanto o binário só assume 0 ou 1.

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 thata value can only be incremented after it has been loaded into a register by a separate machine instruction.

2<= tally <= 100.

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?

2<=tally <=2* n.

n = número de processos.

5.12 Consider the following definition of semaphores. Compare this set of definitions with that of Figure 5.3. Note onedifference: 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++;
}
```

Sim, haverá diferença entre a execução do programa com base nos valores estimados para cada sistema, pois como um assume valores negativos, ele nunca seria sujeito ao primeiro “if”.