Lista 5

1. Considere uma fila de espera *Ready* com n processos. Quantos tipos diferentes de escalonamentos podem ser considerados?

2. Considere um algoritmo de escalonamento *Round Robin* e um processo limitado por E/S com a seguintes características:

   - o processo roda por 1 quantum e fica bloqueado por 4 quanta;
   - o processo precisa rodar por um total de 6 quanta (ficando bloqueado 5 vezes).

   Imagine que ao mesmo tempo há outros 4 processos limitados por CPU no estado *Ready*.

   Qual é o valor do *turnaround time* T_r do processo limitado por E/S se ele voltar ao fim da fila *Ready* quando ficar desbloqueado? Imagine que os processos desbloqueados entram na fila *Ready* depois do processo que acabou de rodar por 1 quantum.

3. Considere cinco processos com as seguintes características

   - $P_1 \quad T_s = 10$
   - $P_2 \quad T_s = 1$
   - $P_3 \quad T_s = 2$
   - $P_4 \quad T_s = 1$
   - $P_5 \quad T_s = 5$

   Aqui T_s é o tempo de CPU necessário para cada processo rodar. Os processos chegam na ordem da lista acima, todos eles no instante 0. Considere os algoritmos FCFS (First-Come-First-Served), RR (Round Robin) com quantum = 1 e SPN (Shortest Process Next). Qual o valor do *turnaround time* T_r e qual a razão T_r/T_s para cada processo usando os algoritmos acima? Que algoritmo apresenta o menor valor médio da razão T_r/T_s?

4. Suponha que um algoritmo de escalonamento favoreça aqueles processos que usaram recentemente o menor tempo do processador. Por que este algoritmo irá favorecer os programas limitados por I/O e não deixará que os programas limitados por CPU fiquem em inanição permanentemente (*starvation*)?

Lista 6

1. Modifique o código (com semáforos) para o problema dos leitores e escritores para impedir *starvation* dos escritores.

2. Mostre que, se as operações *wait* e *signal* não forem executadas atomicamente, a exclusão mútua pode ser violada.

3. Considere a seguinte solução do problema do produtor-consumidor no caso de um *buffer* com capacidade finita N:

```
semaphore a = 0;
semaphore b = 1;
semaphore c = N;

void producer(void)
{
  int item;

  while (TRUE)
  {
    produce_item(&item);
    down(&c);
    down(&b);
    enter_item(item);
    up(&b);
    up(&a);
  }
}

void consumer(void)
{
  int item;

  while (TRUE)
  {
    down(&a);
    down(&b);
    remove_item(&item);
    up(&b);
    up(&c);
```

```
      consume_item(item);
    }
  }
```

Qual é o significado dos três semáforos usados acima?

4. Qual das duas soluções seguintes do problema dos produtores e do consumidor (com "buffer" infinito, usando semáforos binários) é correta?

   O que pode acontecer com a solução errada?

```
I) rotina do produtor
   begin;
     repeat;
       produce;
       waitB(s);
       append;
       n := n + 1;
       if n=1 then
         signalB(delay);
       signalB(s);
     forever;
  end;

  rotina do consumidor
  begin;
    waitB(delay);
    repeat;
      waitB(s);
      take;
      n := n - 1;
      signalB(s);
      consume;
      if n=0 then
        waitB(delay) ;
    forever;

  end;
```

```
II) rotina do produtor
    begin;
      repeat;
        produce;
        waitB(s);
        append;
        n := n + 1;
        if n=1 then
          signalB(delay);
        signalB(s);
      forever;
    end;

    rotina do consumidor
    begin;
      waitB(delay);
      repeat;
        waitB(s);
        take;
        n := n - 1;
        m := n;
        signalB(s);
        consume;

        if m=0 then
          waitB(delay);
      forever;
    end;
```