Semáforos

O problema dos leitores/escritores, apresentado a seguir, consiste em sincronizar processos que consultam/atualizam dados em uma base comum. Pode haver mais de um leitor lendo ao mesmo tempo; no entanto, enquanto um escritor está atualizando a base, nenhum outro processo pode ter acesso a ela (nem mesmo leitores).

```
VAR  Acesso: Semaforo := 1;
     Exclusao: Semaforo := 1;
     Nleitores: integer := 0;
```

```
PROCEDURE Escritor();
BEGIN
    ProduzDado();
    DOWN (Acesso);
    Escreve();
    UP (Acesso);
END;
```

```
PROCEDURE Leitor();
BEGIN
     DOWN (Exclusao);
     Nleitores := Nleitores + 1;
     IF ( Nleitores = 1 ) THEN DOWN (Acesso);
     UP (Exclusao);
     Leitura();
     DOWN (Exclusao);
     Nleitores := Nleitores - 1;
     IF ( Nleitores = 0 ) THEN UP (Acesso);
     UP (Exclusao);
     ProcessaDado();
     END;
```

Semáforos

a) Suponha que exista apenas um leitor fazendo acesso à base. Enquanto este processo realiza a leitura, quais os valores das três variáveis?

b) Chega um escritor enquanto o leitor ainda está lendo. Quais os valores das três variáveis após o bloqueio do escritor ? Sobre qual(is) semáforo(s) se dá o bloqueio?

c) Chega mais um leitor enquanto o primeiro ainda não acabou de ler e o escritor está bloqueado. Descreva os valores das três variáveis quando o segundo leitor inicia a leitura.

d) Os dois leitores terminam simultaneamente a leitura. É possível haver problemas quanto à integridade do valor da variável nleitores? Justifique.

e) Descreva o que acontece com o escritor quando os dois leitores terminam suas leituras. Descreva os valores das três variáveis quando o escritor inicia a escrita.

f) Enquanto o escritor está atualizando a base, chagam mais um escritor e mais um leitor. Sobre qual(is) semáforo(s) eles ficam bloqueados? Descreva os valores das três variáveis após o bloqueio dos recém-chegados.

g) Quando o escritor houver terminado a atualização, é possível prever qual dos processos bloqueados (leitor ou escritor) terá acesso primeiro à base?

h) Descreva uma situação onde os escritores sofram starvation (adiamento indefinido).

Problema do Barbeiro Dorminhoco

```
Semaforo clientes=?;
Semaforo barbeiro=?;
Semaforo mutex=?;
int sentados=0;          /* #clientes sentados */

barbeiros() {
  while(1) {
    down(clientes); /* existem clientes? se não adormece */
    down(mutex);
    sentados --;    /* menos um cliente à espera */
    up(barbeiro);  /* menos um barbeiro adormecido */
    up(mutex);
    cortar();
  }
}

clientes() {
  down(mutex);          /* se não existem cadeiras livres */
  if (sentados < NCads) {/* vai embora; se existem entra */
    sentados ++;        /* mais um cliente à espera */
    up(clientes);       /* acorda barbeiro se necessário */
    up(mutex);          /* liberta zona crítica* /
    down(barbeiro);     /* adormece se não há barbeiros livres * /
    sentar_e_cortar();
  } else
     up(mutex);
}
```

Quais são os valores de:
- clientes ?
- barbeiro ?
- mutex ?

Problema da Padaria

```
#define N   5
typedef struct
{
        int         cliente;   /* número do cliente que deve ser atendido      */
        SEMA        numero;    /* semáforo utilizado para bloquear o vendedor */
                               /* enquanto o cliente não está pronto.          */
} TVENDEDOR;

TVENDEDOR vendedor [N] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
Int                 senha = 1;        /* senha de atendimento            */
Int               proximo = 1;        /* próximo número a ser atendido */
MUTEX       mutex_senha = 1;          /* semáforo de exclusão mútua      */
MUTEX       mutex_proximo = 1;        /* semáforo de exclusão mútua      */

void vendedor (int id)  { /* id – identificação do vendedor */

    while (1) {
        /* vê o próximo número a ser atendido */
        DOWN (&mutex_proximo);
        vendedor[id].cliente = proximo ++;
        UP (&mutex_proximo);

        /* aguarda o cliente pedir */
        DOWN (&vendedor[id].numero);
        atende_cliente ();
    }
}
```

Problema da Padaria

```
void cliente (void) {

    int x, i;

    /* pega um número */
    DOWN (&mutex_senha);
    x= senha ++;
    UP (&mutex_senha);

    /* verifica se há um vendedor para atendê-lo */
    atendido = 0;
    while (! atendido) {
        for (i = 0; i < N; i ++)
            if (vendedor [i].cliente== x) {
                atendido = 1;
                break;
            }
     }
     UP (&vendedor[i].numero);
     faz_pedido (i);
}
```

a) Inicialmente são criados 5 processos representando os 5 vendedores. Estes processos ficam bloqueados? Se a resposta for afirmativa, em que semáforos?

b) Chega um cliente (processo cliente é criado). Ele é atendido por um vendedor. Qual o valor das variáveis (*senha*, *proximo*, *mutex_senha* e *mutex_proximo*) quando o processo cliente está executando a função *faz_pedido* e o processo vendedor (que o atendeu) está executando a função *atende_cliente*? É possível afirmar qual processo vendedor o atendeu (número do processo vendedor)? Justifique.

c) Existe alguma situação neste algoritmo que possa causar starvation? Justifique.

d) Enquanto o primeiro cliente está sendo atendido (executando *faz_pedido*) chegam a padaria mais 9 clientes (mais 9 processos clientes são criados). Quatro destes clientes serão atendidos logo pelos vendedores. Os outros cinco ficarão esperando. Se estes cinco clientes forem embora (os processos forem "mortos"), o que acontece? Justifique.

Produtor & Consumidor - Troca de Mensagens

Observe o trecho de pseudo-código abaixo:

```
program so1_2003;

const   null = " ";
var     i, c: integer;

procedure produtor;
var p1: message;
begin
        repeat
                receive(produz, p1);
                p1 := produto;
                send(produz, p1);
        forever
end;
```

```
Procedure consumidor;

var     p2: message;

begin
        repeat
                receive(consome, p2);
                produto := p2;
                send(consome, null);
        forever
end;

begin
        create_mailbox(produz);
        create_mailbox(consome);

        for i = 1 to c do send(consome, null);
                parbegin
                        produtor;
                        consumidor;
                parend;
end.
```

Produtor & Consumidor - Troca de Mensagens

a) Existem algumas instruções erradas. Mostre quais são elas e como devem ser corrigidas para que o programa funcione corretamente, isto é, garanta a exclusão mútua através do uso de mensagens.

b) Para que o programa funcione corretamente, que características as instruções send() e receive() precisam satisfazer?

c) Seria possível usar este mesmo código num contexto com vários produtores e consumidores em paralelo?

d) Segundo o código, cada item produzido precisa ser consumido imediatamente? Justifique sua resposta.

Escalonamento

Considere os algoritmos: **FCFS** – First Come First Served; **SPN** – Shortest Process Next {min(s)} e **SRT** – Shortest Remaining Time {min(s-e)} e responda às perguntas abaixo:

Compare o desempenho dos 3 esquemas sobre o conjunto de processos abaixo algoritmos através do critério de mérito de menor *turn around normalizado médio* dado por:

$$TANM = \frac{1}{N}\sum_{i=1}^{N}\frac{Tq_i}{ts_i}$$

Assuma que as unidades de execução foram ativadas na ordem alfabética e estão todas prontas para execução.

UE	t_s	UE	t_s
A	8ms	B	4ms
C	3ms	D	5ms

Considere o trecho de código a seguir e responda às questões:

```
#define         maxsubprocs    W;
int             i, j, k, id, x= 0, y= 3, z= 5;
pid_t           pid;

int main(void) {

1.   j = -1;
2.   for (i = 0; i < maxsubprocs; i++) {
3.        pid = fork();
4.        if (pid) {
5.            x = x + j;
6.            y = y + i;
7.            z = x * y;
8.            pid = fork();
9.            if (pid == 0) {
10.               display(x, y, z);
11.           }
12.           else {
13.               display(x, y, z, i, j);
14.               z = (x * i) + (z * j);
15.           }
16.       }
17.       else {
18.           k = i + 1;
19.           j = k;
20.           x= x – 5*j;
21.           y= y - 2*i;
22.       }
23.  }
24.  display(x, y, z, i, j);
 }
```

a) Execute o código para **w = 1** e responda:
   a.1) Quantos processos estarão ativos ao final do loop, contando com o processo pai?
   a.2) Qual ou quais dos processos ativados executa a instrução 24?
   a.3) Execute o fluxo de instruções e mostre as instruções que serão executadas por cada processo ou subprocesso e o conteúdo das variáveis que serão apresentadas pelos displays.
   a.4) Quantos dos processos gerados também serão pais?

b) Para **w = 2** responda quantos processos serão ativados além do processo original (principal)?