Sistemas Operativos

Uso de semáforos

Curso 2021

Facultad de Ingeniería, UDELAR

Agenda

1. Alicia y Bernardo

2. Productor - Consumidor

3. Lectores - Escritores

4. Filósofos comensales

Alicia y Bernardo

Alicia y Bernardo

- Se tienen dos procesos que necesitan el acceso mutuo-excluido a una zona crítica.

```
INIT(S, 1);
```

```
procedure Alicia()
   repeat
      P(S);
      acceder_Ali;
      V(S);
      otras tareas;
   until False
end procedure
```

```
procedure Bernardo()
   repeat
      P(S);
      acceder_Ber;
      V(S);
      otras tareas;
   until False
end procedure
```

Productor - Consumidor

Productor - Consumidor I

- Se tiene un conjunto de procesos Productor que almacenan datos en un buffer y un conjunto de procesos Consumidor que sacan datos del buffer.
- En primera instancia, el problema con buffer infinito.

Solo se puede sacar del buffer solo cuando hay algo

Productor - Consumidor I

```
INIT(N, 0);          ▷ Asumo que buffer inicia vacío

procedure                          procedure
productor()                        consumidor()
   repeat                             repeat
      producir();                        P(N);
      guardar();                         tomar();
      V(N);                              consumir();
   until False                        until False
end procedure                      end procedure
```

Productor - Consumidor II

- Se tiene un conjunto de procesos Productor que almacenan datos en un buffer y un conjunto de procesos Consumidor que sacan datos del buffer.
- Seguimos con el problema con buffer infinito.

Sacar del buffer solo cuando hay.

Mutuoexcluir el acceso al buffer.

Productor - Consumidor II

```
INIT(S, 1);                                  ▷ Mutex
INIT(N, 0);          ▷ Asumo que buffer inicia vacío
```

```
procedure
productor()
   repeat
      producir();
      P(S);
      guardar();
      V(S);
      V(N);
   until False
end procedure
```

```
procedure
consumidor()
   repeat
      P(N);
      P(S);
      tomar();
      V(S);
      consumir();
   until False
end procedure
```

Productor - Consumidor III

- Se tiene un conjunto de procesos Productor que almacenan datos en un buffer y un conjunto de procesos Consumidor que sacan datos del buffer.
- Un problema más realista, buffer finito.

Sacar del buffer solo cuando hay.

Poner en el buffer solo cuando no está lleno.

Mutuoexcluir el acceso al buffer.

Productor - Consumidor III

```
INIT(S, 1);                                   ▷ Mutex
INIT(N, 0);              ▷ Asumo que buffer inicia vacío
INIT(E, TAM_BUFFER);           ▷ Asumo que inicia vacío

procedure productor()                procedure consumidor()
    repeat                               repeat
        producir();                          P(N);
        P(E);                                P(S);
        P(S);                                tomar();
        guardar();                           V(S);
        V(S);                                V(E);
        V(N);                                consumir();
    until False                          until False
end procedure                        end procedure
```

Lectores - Escritores

Lectores - Escritores I

- Se tiene un conjunto de procesos Lectores que acceden en modo lectura a un recurso (p. ej. una BD)
- Y un conjunto de procesos Escritores que acceden en modo escritura al mismo recurso.

Puede haber muchos Lectores accediendo al mismo tiempo, pero el acceso de los Escritores tiene que ser de a uno.

Comportamiento asimétrico entre los Lectores y Escritores.

Lectores - Escritores I

Motivación:

- El acceso de lectura y escritura a una variable no es una operación atómica
- Ejemplo: sumar uno a una variable:
  1. MOV (de memoria a registro)
  2. INC (para sumar uno)
  3. MOV (de registro a memoria)
- Si se hacen varias sumas en forma concurrente pueden dar resultados equivocados
- La lectura de la variable consiste solamente en un MOV y puede hacerse en paralelo

Lectores - Escritores I

```
INIT(E, 1);     ▷ Mutex acceso
INIT(mL, 1); ▷ Mutex lectores
cantLect = 0;

procedure Escritor()
   repeat
      P(E);
      escribir();
      V(E);
   until False
end procedure
```

```
procedure Lector()
   repeat
      P(mL);
      cantLect := cantLect + 1;
      if cantLect = 1 then
         P(E);
      end if
      V(mL);

      leer();

      P(mL);
      cantLect := cantLect - 1;
      if cantLect = 0 then
         V(E);
      end if
      V(mL);
   until False
end procedure
```

Lectores - Escritores II

- Se tiene un conjunto de procesos Lectores que acceden en modo lectura a un recurso (p. ej. una BD)
- Y un conjunto de procesos Escritores que acceden en modo escritura al mismo recurso.

Puede haber muchos Lectores accediendo al mismo tiempo, pero el acceso de los Escritores tiene que ser de a uno.

Comportamiento asimétrico entre los Lectores y Escritores.

Prioridad cuando llega un Escritor, y hay Lectores leyendo, por sobre los próximos Lectores que lleguen.

Lectores - Escritores II

```
INIT(E, 1);                                          ▷ Mutex buffer
INIT(mL, 1);                                         ▷ Mutex lectores
INIT(mE, 1);                                         ▷ Mutex escritores
INIT(try, 1);                                        ▷ Intento lectores
cantEsc = cantLect = 0;
```

```
procedure Escritor()
  repeat
    P(mE);
    cantEsc := cantEsc + 1;
    if cantEsc = 1 then
      P(try);
    end if
    V(mE);

    P(E);
    escribir();
    V(E);

    P(mE);
    cantEsc := cantEsc - 1;
    if cantEsc = 0 then
      V(try);
    end if
    V(mE);
  until False
end procedure
```

```
procedure Lector()
  repeat
    P(try);
    P(mL);
    cantLect := cantLect + 1;
    if cantLect = 1 then
      P(E);
    end if
    V(mL);
    V(try);

    leer();

    P(mL);
    cantLect := cantLect - 1;
    if cantLect = 0 then
      V(E);
    end if
    V(mL);
  until False
end procedure
```

Filósofos comensales

Filósofos comensales

- Hay 5 filósofos que viven sentados alrededor de una mesa pensando y comiendo.
- En la mesa, circular se dispone de 5 platos y 5 tenedores. Para comer se necesita utilizar 2 tenedores
- Cuando un filósofo quiere comer, primero agarra un tenedor y luego el otro.

Filósofos comensales: primera solución

```
INIT(T[i], 1);                          ▷ Mutex tenedor i
procedure filosofo_i()
   repeat
      P(T[i]);
      P(T[i+1 mod 5]);
      comer();
      V(T[i]);
      V(T[i+1 mod 5]);
      pensar();
   until False
end procedure
```

Filósofos comensales

Problema:

- Si los 5 quieren comer a la vez quedan en deadlock.

Soluciones:

- Solo permitir comer a 4 a la vez.
- Tener un administrador que solo permita tomar los dos tenedores juntos o ninguno.
- Numerar los tenedores y que solo se puedan tomar en orden ascendente (o descendente).

Filósofos comensales: solo comen 4

```
INIT(N, 4);                          ▷ Mutex comensales
INIT(T[i], 1);                        ▷ Mutex tenedor i
procedure filosofo_i()
   repeat
      P(N);
      P(T[i]);
      P(T[i+1 mod 5]);
      comer();
      V(T[i]);
      V(T[i+1 mod 5]);
      V(N);
      pensar();
   until False
end procedure
```