Examen Sistemas Concurrentes y Distribuidos
13 de Septiembre de 2012

Apellidos, Nombre:
Grupo (M/T):

1) (1 punto) Concepto de semáforo. Similitudes y diferencias con las variables de condición de los monitores.

2) (1 punto) Condiciones de los protocolos de E/S en algoritmos de espera ocupada. Discutir su cumplimiento en el algoritmo de Dekker.

3) (1 punto) Ventajas e inconvenientes entre las posibilidades que ofrece el paso de mensajes asíncronos frente a los sincronos.

4) (1 punto) Principales características de la Llamada a Procedimiento Remoto.

5) (2 puntos) Tenemos un sistema con `t` terminales e `i` impresoras. En cada terminal se ejecuta un proceso que usa recursos propios y accede a utilizar alguna de las `i` impresoras (siendo las `i` impresoras recursos comunes a los `t` terminales). Programar los procesos utilizando semáforos para el acceso a una impresora sea en exclusión mutua pero pudiendo haber varios procesos a la vez en las impresoras, cada uno en una distinta.

Solución:

- libre: array[0..i-1] de boolean, que tendrá un valor de Verdad si la impresora i está libre.
- impresoras: semáforo, que estará inicializado a i e indicará el número de impresoras disponibles en cada momento.
- mutex: semáforo, para la exclusión mutua del arras libre y está inicializado a 1.

```
Proceso terminal

variables
   k : entero

........ sentencias anteriores al uso de una impresora.

wait(impresoras)
wait(mutex)
   k = 0
   Mientras No libre[k] hacer
      k++
   finMientras
   *** Buscamos la impresora que está libre
```

```
   libre[k] = Falso *** La ocupamos
signal(mutex)

UsarLaImpresora(k)

wait(mutex)
   libre[k] = Verdad *** Liberamos la impresora
signal(mutex)
signal(impresoras)

..... sentencias posteriores al uso de una impresora
```

Suponemos que tenemos una serie de procesos terminales lanzados que intentarán hacer uso de las impresoras del sistema.

6) (2 puntos) Supongamos tres procesos: Pa, Pb y Pc de tal forma que Pa sólo puede escribir la letra 'a', Pb la letra 'b' y Pc la letra 'c'. Los tres procesos están en un bucle de 1 a 5. Se pide sincronizar los tres procesos para que se cumplan las siguientes condiciones:

1. Una 'b' siempre se imprime antes que una 'c'.
2. Las b's y las c's deben alternarse en la cadena de salida, de tal forma que después de la primera 'b' se ha impreso, ninguna otra 'b' puede ser impresa hasta que no se imprima una 'c'. De igual forma, cuando se imprime una 'c' no puede volver a imprimir otra hasta que no se imprima una 'b'.
3. El número total de b's y c's presentes en la cadena de salida en un instante determinado no puede exceder el numero de a's en la cadena de salida en ese mismo instante.

   La solución será programada mediante el paso de mensajes asíncronos.

Solución:

- buzones : array [1..3] de caracteres. Donde se enviarán los caracteres a imprimir.
- buzonesRespuesta : arras [1..3] de enteros. Buzón para la sincronización para imprimir el carácter.

```
Proceso ImprimeA

variables
  cont : entero
  respuesta : entero
```

```
Para cont = 1 hasta 5 hacer
   enviar (buzon[1], 'a')
   recibir (buzonRespuesta[1], respuesta)
   escribir ('a')
finPara

Proceso ImprimeB

variables
   cont : entero
   respuesta : entero

Para cont = 1 hasta 5 hacer
   enviar (buzon[2], 'b')
   recibir (buzonRespuesta[2], respuesta)
   escribir ('b')
finPara

Proceso ImprimeC

variables
   cont : entero
   respuesta : entero

Para cont = 1 hasta 5 hacer
   enviar (buzon[3], 'c')
   recibir (buzonRespuesta[3], respuesta)
   escribir ('c')
finPara

Proceso Controlador

variables
   numeroDeA : entero
   tocaB : boolean
   letra : carácter

numeroDeA = 0
tocaB = verdad

Repetir
   Select
```

```
      recibir (buzon[1], letra)
      numeroDeA++
      enviar (buzonRespuesta, numeroDeA)
   or
      Mientras (numeroDeA>0) Y tocaB =>
         recibir (buzon[2], letra)
         numeroDeA--
         tocaB = Falso
         enviar(buzonRespuesta[2], numeroDeA)
   or
      Mientras (numeroDeA>0) Y (No tocaB) =>
         recibir (buzon[3], letra)
         numeroDeA--
         tocaB = Verdadero
         enviar(buzonRespuesta[3], numeroDeA)
   finSelect
ParaSiempre
```

Suponemos que los tres procesos que imprimen y el controlador se encuentran ya lanzados en el sistema.

7) (2 puntos) Diseñar un proceso controlador que provoque que los dos primeros procesos que lo invoquen sean suspendidos y el tercero los despierte, y así cíclicamente. La solución será programada mediante el paso de mensajes síncronos.

Solución:

Utilizaremos dos buzones de sincronización:

- pidoPermiso
- permisoConcedido

La solución para los diferentes procesos vendrá parametrizada por un valor que nos lo identificará.

```
Proceso Proc(i)

Repetir
   ...
   pidoPermiso[i] ! any
   permisoConcedido[i] ? any
   ...
ParaSiempre
```

```
Proceso Controlador

variables
   identificadores : array [1..3] de enteros;
   // almacena el identificador del proceso bloqueado
   cont1,cont2 : entero

cont1=0
Repetir
   cont1++
   Select
      Para cont2 = 1 hasta NUMEROPROCESOS replicate
         pidoPermiso[cont2] ? any
         identificadores[cont1] =cont2
   or
      terminate
   finSelect

   Si cont1 == 3 entonces
       Para conté = 1 hasta 3 Hacer
          permisoConcedido[identificadores[cont2]] ! any
       FinPara
       cont1 = 0
    FinSi
ParaSiempre
```