EXAMEN DE SISTEMAS INFORMÁTICOS DE TIEMPO REAL

Septiembre 2001

1. Definir de forma breve qué es un **Sistema Informático de Tiempo Real**. Poner un ejemplo. Explicar las diferencias entre los Sistemas de Tiempo Real controlados por **eventos** o por **tiempo**.

(1,5 puntos)

(Capítulo 1 del libro)

Un Sistema Informático de Tiempo Real se define como un sistema basado en computador el cual, ***debe responder ante estímulos generados por el entorno dentro de un periodo de tiempo finito***. No importa solo que sea capaz de generar un resultado correcto sino que éste se produzca en un tiempo determinado, asimismo interactúa con el entorno (mundo físico real) adquiriendo estímulos y estados del entorno, y generando una acción sobre dicho entorno.

El control de procesos continuos es un ejemplo de aplicación de los sistemas de tiempo real. El computador debe mantener las variables de salida del sistema siguiendo una cierta función de referencia. Cuando detecta que las variables se alejan de dicha referencia el computador debe responder variando las entradas del sistema de acuerdo a una acción de control. Esta respuesta debe ejecutarse dentro de un periodo de tiempo fijado por la dinámica del sistema y del regulador empleado.

Los STR controlados ***por eventos o interrupciones*** establecen la ejecución de un componente o tarea basándose en la aparición de una interrupción o señal generada por un evento externo. Constituyen un mecanismo eficaz para responder ante eventos externos no regulares.

Los STR controlados ***por tiempo*** operan de acuerdo a los ciclos del reloj o relojes del sistema. Este tipo de arquitecturas operan tratando los pulsos regulares del reloj como si fueran señales de interrupción. Cuando el reloj alcanza ciertos valores predefinidos, una acción apropiada es seleccionada para su ejecución. Este tipo de sistemas se utiliza cuando es preciso la ejecución de tareas periódicas o la ejecución de tareas mediante temporizadores.

2. **Describir las diferentes clases de direcciones IPv4. Explicar el uso de la máscara de subred..**

(1,5 puntos)

(Capítulo 11.3 del libro)

El direccionamiento IPv4 se realiza mediante palabras de 32 bits, agrupadas en cuatro bloques de 8 bits (byte). El direccionamiento IPv4 esta jerarquizado en 5 clases en función de la codificación de los primeros bits. Estas clases determinan el agrupamiento de los bits de la dirección en dos partes:

- **parte de red** (identifica la red)

- **parte de nodo/subred**: identifica cada nodo dentro de la red. Así mismo, permite identificar subredes en función de la **máscara de subred**.

A continuación se describen las diferentes clases:

Clase A: **RED** **SUBRED/NODO**

0xxx xxxx . xxxx xxxx . xxxx xxxx . xxxx xxxx

1111 1111 . 0000 0000 . 0000 0000 . 0000 0000 Máscara Subred Estándar

Clase B: **RED** **SUBRED/NODO**

10xx xxxx . xxxx xxxx . xxxx xxxx . xxxx xxxx

1111 1111 . 1111 1111 . 0000 0000 . 0000 0000 Máscara Subred Estándar

Clase C: **RED** **SUBRED/NODO**

110x xxxx . xxxx xxxx . xxxx xxxx . xxxx xxxx

1111 1111 . 1111 1111 . 1111 1111 . 0000 0000 Máscara Subred Estándar

Clase D: **MULTICAST (28 bits)**

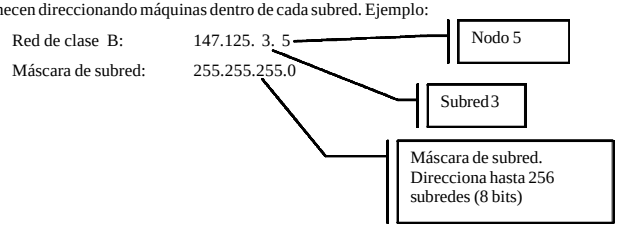
1110 xxxx . xxxx xxxx . xxxx xxxx . xxxx xxxx

Clase E: **Futuras Ampliaciones (27 bits)**

1111 0xxx . xxxx xxxx . xxxx xxxx . xxxx xxxx

Clase	Nodos por clase	Máscara	Dir. Comienzo	Dir. final
A	16.777.216	255.0.0.0	0.0.0.0	127.255.255.255
B	65536	255.255.0.0	128.0.0.0	191.255.255.255
C	256	255.255.255.0	192.0.0.0	223.255.255.255
D	-	-	224.0.0.0	239.255.255.255
E	-	-	240.0.0.0	255.255.255.255

La máscara de subred permite dividir una red en subredes, colocando en la máscara de subred un uno en aquellos bits libres que deseamos que codifiquen una subred. El resto de bits, cuya máscara es cero, permanecen direccionando máquinas dentro de cada subred. Ejemplo:



3. **Para la red mostrada en la figura se pide**:

   a) Asignar una dirección IP de Red de clase C (cualquiera que sea válida).

   b) Especificar la máscara de subred adecuada para cada una de las subredes

   c) Asignar las direcciones IP de cada subred.

   d) Asignar las direcciones IP a cada nodo (Estaciones y Routers).

   e) Asignar la dirección IP de la pasarela (router) por defecto de cada nodo.

(2 puntos)

Nota: la selección de las direcciones concretas es libre siempre que cumpla los requerimientos del problema y sean direcciones IP válidas (no privadas).

Utilizar direcciones correlativas para los nodos en cada subred. Basta con indicar la primera y última dirección de cada nodo.

La solución se puede entregar sobre el gráfico del enunciado o bien sobre una reproducción correcta del mismo.

(Capítulo 11 del libro)

a) Una red de clase C se identifica por la secuencia 110xxxxx en los bits más significativos de la dirección IP por lo que el rango de direcciones IP será;

192.0.0.0 a 223.255.255.0

Excluyendo el rango de direcciones privadas 192.168.00 – 192.168.255.0 podemos elegir cualquiera.

Tomaremos por ejemplo la dirección de red **194.100.100.0** que está en el rango de direcciones Europeas(194.0.0.0 – 195.255.255.0)

b) Ya que nuestra red está formada por tres subredes necesitamos dos bits del campo de subred/nodo (cuarto byte) para direccionar las subredes. Para ello activaremos los dos bits más significativos de la máscara de subred en este campo (cuarto byte).

Los 6 bits restantes nos permiten direccionar 2^6**-2=62** nodos. (dos direcciones están reservadas para direccionar la subred y el broadcast). Estos son **insuficientes** para la tercera subred ya que dispone de 62 PCs más un router por lo que se necesitan 63 direcciones IP para nodos. En cambio en el resto de subredes sobran direcciones y nos sobra una subred

Para solucionarlo ajustaremos al máscara de subred en función del tamaño de la subred asegurándonos de que no se solapen direcciones entre subredes diferentes. Para ello juntaremos dos subredes contiguas para formar una subred de mayor tamaño:

- Para las dos redes de 16 y 32 PCs utilizaremos una máscara de subred con dos bits (bits más significativos)

255.255.255.192 -> (hasta 62 nodos)

- Para la red de 62 PCs utilizaremos una máscara de subred con un bit (bit más significativo) [esto equivale a agrupar dos subredes]

255.255.255.128 -> (hasta 2^7-2= 126 nodos)

c) Ya que máscara de subred es variable debemos realizar cuidadosamente la asignación de direcciones de subred para evitar que se solapen direcciones. Empezaremos numerando la de mayor dimensión asignándole un 0 al único bit usado para esta subred (más significativo)

- Subred Inferior (62 PCs + router): **194.100.100.0 (mask:255.255.255.128)**
  El rango de direcciones para nodos será desde la 194.100.100.1 a la 194.100.100.127

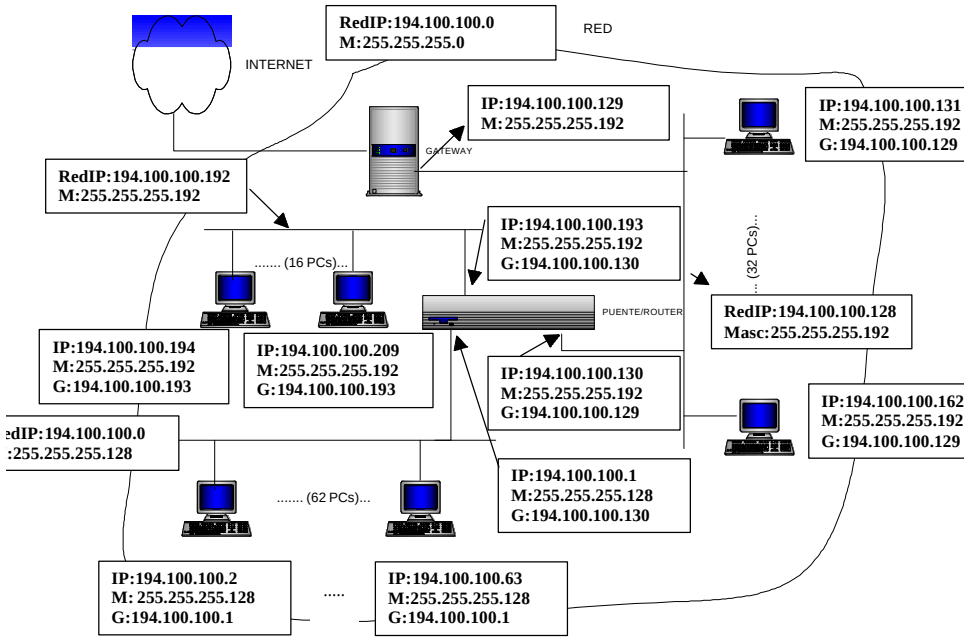
Para las otras dos subredes utilizaremos los valores **10** y **11** en los bits de subred, de este modo no se solapan con los anteriores (**1** en el bit más significativo):

- Subred Derecha (32PCs + router+ pasarela): **194.100.100.128(mask:255.255.255.192)**
  El rango de direcciones para nodos será desde la 194.100.100.129 a la 194.100.100.191

- Subred Izquierda Superior (16PCs + router): **194.100.100.192(mask:255.255.255.192)**
  El rango de direcciones para nodos será desde la 194.100.100.193 a la 194.100.100.255

Como podemos observar no se solapan ningún rango de direcciones por lo que el enrutamiento dentro de la red funcionará correctamente.

d) y e) sobre el gráfico:



4. Explicar en que consiste el paralelismo y el pseudoparalelismo. Comentar cual es el algoritmo de planificación más adecuado para implementar el pseudoparalelismo.

(1,5 puntos)

- Solución en capítulo 3 del libro de la asignatura, página 47, y capítulo 4 (pag. 73-74)

5. Semáforos: definición y funcionamiento. Poner un ejemplo.

(1,5 puntos)

- Solución en capítulo 7 del libro de la asignatura, páginas 119 y 120

6. Realizar un programa que lance un thread que ejecute una función que multiplique dos números. Los dos números se deben pasar a la función como argumentos en la llamada de creación del thread. Los valores de los dos números a multiplicar se deben inicializar en la función main.

(2 puntos)

(Capítulo 6)

```
//Programa que muestra el paso de parámetros en threads
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
/* Prototipos de las funciones que ejecutan los threads */
void *func1 (void *);
// Estructura que contiene los datos a pasar como parámetros
// Un único parámetro se puede pasar directamente con el operador &
typedef struct
{
      int dato1,dato2;
}datos;
/* Declaración de los threads */
pthread_t thread1, thmain;
pthread_attr_t attr;    /*atributos de los threads*/
/* Definición de las función func1 */
void *func1 (void *arg)
{
      int a,b;
      datos *p= (datos *) (arg);
      pthread_t tid = pthread_self();  /*se asigna un identificador de
thread*/
      a=(p->dato1);
      b=(p->dato2);
      printf("Soy el thread 1 y voy a ejecutar func1 \n");
      printf("La multiplicación es %d\n",a*b);
      printf("Soy el thread 1 y he terminado de ejecutar func1\n");
      pthread_exit(NULL);     /* Provoca la terminación del thread*/
}
/*Función main*/
void main(void)
{
      datos param;
      param.dato1=6;
      param.dato2=6;
      thmain = pthread_self();
      pthread_attr_init (&attr);
      printf("Soy la función main y voy a lanzar el thread \n");
      pthread_create (&thread1, &attr, func1, &param);
      printf("Soy main: he lanzado el threads y termino\n");
      pthread_exit(NULL);
}
```


Campus de Elche. Avda. del Ferrocarril s/n – 03202 ELCHE