Examen de Periféricos Interfaces y Buses

Febrero 2008

1) (2 puntos)

Proponer las estructuras de datos necesarias y escribir la función de instalación del módulo para un driver (Linux) de un dispositivo que permite tres modos de funcionamiento. En los tres modos tiene las operaciones de apertura y cierre del dispositivo, así como las de lectura y escritura. Los datos que necesita el driver para los diferentes modos son los siguientes:

- Modo 1: tres números enteros
- Modo 2: tres números reales
- Modo 3: tres strings de hasta 12 caracteres

Explicar además la manera en la que se debe instalar el driver para su uso en los diferentes modos desde un programa de aplicación.

2) (2 puntos)

Escribir la función `ioctl` de un driver de Linux para un puerto serie que está instalado de manera que no se ha seleccionado el formato del dato ni la velocidad de transmisión durante la instalación. Se obliga, por tanto, a llamar al `ioctl` para programar completamente el puerto antes de su uso. El `ioctl` debe permitir especificar:

- el número de bits de stop que se usan: 1 ó 2
- el tamaño del dato: 7 u 8 bits
- la paridad siempre par
- velocidades de transmisión de 450, 1600, 6400, y 19200 bps

Diseñar la estructura de datos necesaria para el intercambio de información con la función ioctl y la codificación del comando o comandos. Si alguno de los valores pasados como argumento no está en el rango permitido, la llamada debe retornar un error.

3) (2 puntos)

En una aplicación de control el programa necesita acceder a la información suministrada por un sensor de temperatura a intervalos regulares fijos. La especificación determina que la lectura del sensor debe obtener un nuevo dato, pero si transcurridos 100 milisegundos no se obtiene un nuevo valor, se debe devolver el valor anterior.

Al sensor se puede acceder a través del *Registro de Sensor* de 16 bits en la posición 6F8h de memoria de I/O. La escritura de cualquier valor en este registro hace que el sensor comience la medida de la temperatura. Cuando tiene un valor estable produce un interrupción (IRQ3) y se puede leer la temperatura en el registro.

Escribir la función de lectura y la función manejadora de la interrupción del driver Linux que permite a la aplicación obtener el valor de la temperatura de acuerdo a las especificaciones. Diseñar las estructuras de datos necesarias y utilizar los mecanismos de sincronización más adecuados. Justificar la propuesta realizada.

4) (2 puntos)

Se desea programar una interfaz capaz de controlar dos dispositivos que permiten:

- obtener las coordenadas UTM (a través de GPS)
- obtener el valor de la altitud por medida de la presión atmosférica

Se dispone de dos funciones para el acceso al hardware que ya están escritas:

```
void lee_UTM (struct UTM *coordenada);
// Devuelve en el parámetro coordenada el valor UTM dado por
// el dispositivo

void lee_altitud (int *alt);
// Devuelve en el parámetro alt el valor de la altitud en
// metros dado por el dispositivo
```

El objetivo es permitir que el driver Linux suministre a la aplicación un valor conjunto de coordenadas y altitud (en la operación de lectura). El driver debe leer las coordenadas cada 1,5 milisegundos y la altitud cada 30 milisegundos. Estas lecturas actualizan una variable interna del driver, que es la que la aplicación podrá consultar.

Proponer el código del driver que permita realizar la actualización de coordenadas y la altitud (no es necesaria la función de lectura del driver). Se deben realizar las estructuras de datos y de programa necesarios para cumplir la especificación (también el código que pudiera ir en la inicialización del driver).

5) (2 puntos)

Escribir una función capaz pintar en pantalla la información del fabricante e identificador de dispositivo de todos los dispositivos conectados al bus PCI de un determinado sistema. La función debe devolver además el número de dispositivos conectados y se realizará teniendo en cuenta que puede formar parte del código de un driver Linux.