Arquitectura de Sistemas de Información

Grado en Ingeniería en Tecnología de Telecomunicación. 3º curso.
Prueba práctica

2 de julio de 2018 Tiempo total: 2 ½ h.

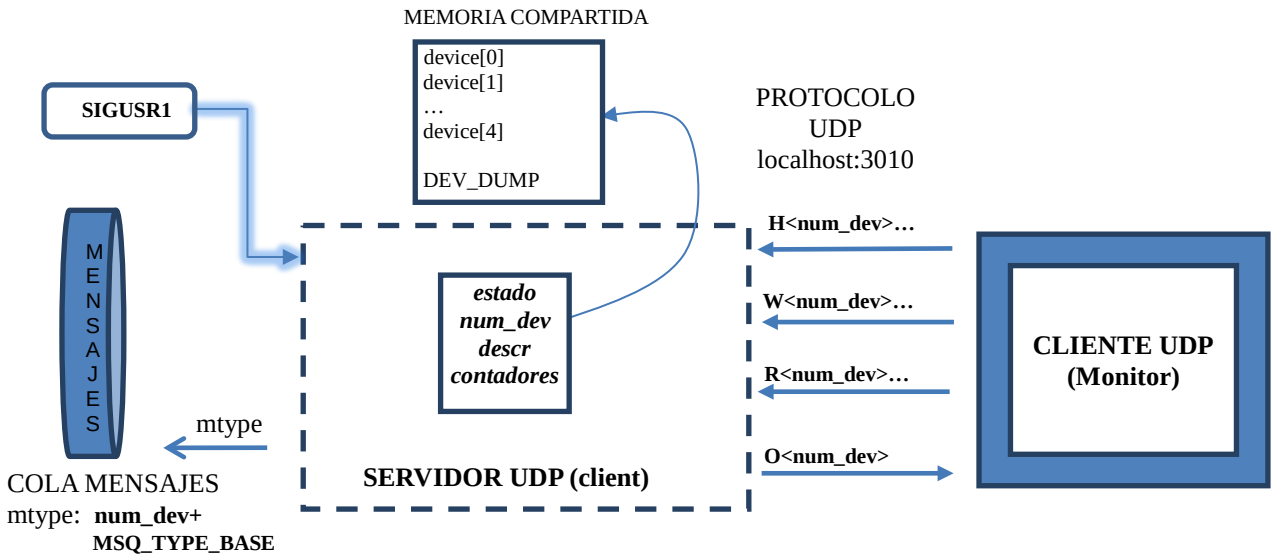
Hay que descubrir 15 claves. El aprobado se establece en 8 claves.

Descripción de la prueba práctica

El monitor funcionará como un cliente UDP que se conectará al servidor de la aplicación a desarrollar (*client.c*) para dar de alta y almacenar medidas de diferentes sensores biomédicos: pulso, tensión, temperatura, glucosa y saturación de O_2.

Los ejercicios y el monitor están diseñados de modo que permitan mantener la compatibilidad hacia atrás con los ejercicios anteriores, es decir, no hay que borrar ni comentar código de los ejercicios anteriores para hacer los siguientes. Esto permite desarrollar la aplicación cliente añadiendo en cada paso nuevas funcionalidades al desarrollo. Esta compatibilidad hacia atrás no implica que los ejercicios sean dependientes entre sí. Por el contrario, en la medida de lo posible se han diseñado de forma que se puedan desarrollar independientemente y especialmente el ejercicio 5 puede abordarse de forma prácticamente separada.

La arquitectura del sistema es la que se refleja en la siguiente figura:



A partir de la primera posición de la memoria compartida se almacenarán consecutivamente los registros de los sensores biomédicos, que denominaremos dispositivos en lo sucesivo. El número de dispositivo (*num_dev*) indicará la posición de su registro en el array de registros de la memoria compartida. Así, el dispositivo *num_dev*=0 se situará en la posición 0, inmediatamente después ira el dispositivo 1, y así hasta cinco dispositivos. Cada registro mantendrá información del dispositivo y un array de contadores para sus medidas según la estructura *shm_dev_reg* del fichero *client_ref.c*. Asociado a cada dispositivo habrá un semáforo para sincronizar el acceso a memoria.

La recepción de una señal SIGUSR1 desencadenará una operación de volcado de contenido de la memoria compartida en un mensaje que se transmitirá mediante una cola de mensajes. En la posición DEV_DUMP de la memoria compartida se encontrará el número de dispositivo(s) a

volcar de memoria mediante la cola de mensajes cuando lleguen interrupciones SIGUSR1.

Se propone realizar una jerarquía de dos procesos, en la cual el hijo implementará el servidor UDP y el padre atenderá las interrupciones SIGUSR1.

En todos los recursos, memoria compartida, semáforos y colas de mensajes se utilizará como clave el DNI del alumno sin letra codificado en hexadecimal como un long (ej: DNI:11234567G la clave sería 0x11234567L). Se recomienda abrir los recursos necesarios al inicio del programa.

Ejercicio 1. Implementación del servidor UDP

En este primer ejercicio se trata de realizar una comunicación UDP básica que consiste en esperar un comando 'H' (hello) al que se responderá con un 'O' (ok). La aplicación a desarrollar escuchará en localhost:3010.

El comando H será un datagrama procedente del monitor en binario con formato:

H<num_dev>descr (longitud variable)

donde *H* es un byte, *<num_dev>* es un entero que representa el número de dispositivo y *"descr"* una cadena (terminada en '\0') con su descripción (los símbolos < y > no se transmiten). Se descubrirá la **clave 2** si se responde al monitor en binario con:

O<num_dev> (5 bytes)

Si no se es capaz de descifrar los campos del mensaje *H* y generar la respuesta *O<num_dev>* adecuada, basta con responder por el socket con cualquier otro mensaje, por ejemplo, *OK* y se obtendrá la **clave 1**.

Ejercicio 2. Alta de dispositivos en memoria

Cuando llegue una petición H habrá que registrar el dispositivo *num_dev* en su posición correspondiente en memoria copiando la estructura *shm_dev_reg* definida en *client-ref.c*. El valor de *estado* deberá ser 1 para indicar que se ha ocupado el registro, de lo contrario, se entenderá que no ha sido registrado correctamente. Se comprobarán los campos num-dev, descr y estado.

Si se da de alta correctamente el primer dispositivo *num_dev*=0 (offset 0) se obtendrá la **clave 3**. Si se registra correctamente cualquier otro dispositivo en su posición se obtendrá la **clave 4**.

Ejercicio 3. Actualización y lectura de registros de dispositivos

En este ejercicio atenderemos peticiones de almacenar valores de medidas de dispositivos en su correspondiente contador de memoria compartida. Estas peticiones tienen el siguiente formato binario:

W<num_dev><contador><valor> (tamaño total 13 bytes)

donde *W* es un byte, *<num_dev>* es un entero que representa el número de dispositivo y *<contador>* es un entero que indica la posición del array en la que queremos escribir el entero *<valor>* (los símbolos < y > no se transmiten).

Con el comando *W* se deberá localizar la posición *contador* del dispositivo *num_dev* y escribir

en ella el *valor* de la medida. El monitor esperará como respuesta: *O<num_dev>* (5 bytes). Si se recibe la respuesta esperada, se obtiene la **clave 5.**

Si atiende correctamente la primera petición *W* que realice el monitor (*num_dev*=0, *contador*=0), se obtiene la **clave 6**. Si se atiende correctamente todo el resto de peticiones *W* se consigue la **clave 7**.

Las peticiones de lectura de valores de contadores tienen el siguiente formato binario:

R<num_dev><contador> (tamaño total 9 bytes)

Cuando llegue una petición R habrá que localizar el valor del contador dentro del dispositivo num_dev y devolver su contenido mediante el siguiente mensaje de respuesta:

O<num_dev><valor> (9 bytes)

Si el mensaje de respuesta es correcto se obtendrá la **clave 8**.

Ejercicio 4. Acceso a sección crítica mediante semáforos.

El programa *client* deberá acceder a un array de 5 semáforos creado por el monitor. Cada semáforo del array estará asociado por ese orden a cada uno de los cinco dispositivos en memoria.

Si las operaciones W se realizan respetando los semáforos se obtiene la **clave 9**.

Si las operaciones R se realizan respetando los semáforos se obtiene la **clave 10**.

Ejercicio 5. Volcado de memoria mediante interrupciones y colas de mensajes.

El proceso principal de la jerarquía será el encargado de atender las interrupciones SIGUSR1. Cada vez que llegue una interrupción SIGUSR1 se espera que la aplicación vuelque vía cola de mensajes el contenido completo de los dispositivos que se le indique en la posición de memoria DEV_DUMP (su contenido será *num_dev,* entero binario). Los valores posibles de *num_dev* serán de 0 a 4.

Como respuesta a esta interrupción se espera que el programa copie el dispositivo entero de la memoria y lo envíe a la cola mediante un mensaje de tipo *num_dev*+MSQ_TYPE_BASE (definida en client_ref.c). El monitor esperará recibir este mensaje con el contenido exacto del dispositivo completo.

Se obtendrá la **clave 11** si como respuesta a SIGUSR1 se obtiene un mensaje tipo *num_dev*+MSQ_TYPE_BASE con el contenido adecuado.

Además, si el acceso a memoria de los dispositivos se realiza respetando los semáforos correspondientes se conseguirá la **clave 12**.

Cuando el valor almacenado en DEV_DUMP sea ***num_dev = -1***, se espera que se haga por orden creciente el volcado de cada uno de los cinco primeros dispositivos, respondiendo en este caso a esa petición con cinco mensajes de respuesta; uno por cada dispositivo del 0 al 4. Si se responde correctamente con cinco mensajes de respuesta válidos se obtiene la **clave 13**.

Se obtendrá la **clave 14** si con *num_dev* = -1, se bajan los cinco semáforos (0..4) en una única

operación y no en cinco operaciones independientes.

Finalmente, si el ejercicio 5 se ejecuta en concurrencia con el servidor UDP de los apartados anteriores se consigue la **clave 15**.

Arquitectura de Sistemas de Información

Grado en Ingeniería en Tecnología de Telecomunicación. 3º curso.
Prueba práctica

2 de julio de 2018 Tiempo total: 2 ½ h.

Hay que descubrir 15 claves. El aprobado se establece en 8 claves

Nombre:……
Grupo:……..…… DNI:…….
Aula:……. Fila:…….. Columna:….....FIRMA:

Ejercicio 1

Clave 1:	Clave 2:

Ejercicio 2

Clave 3:	Clave 4:

Ejercicio 3

Clave 5:	Clave 6:	Clave 7:	Clave 8:

Ejercicio 4

Clave 9:	Clave 10:

Ejercicio 5

Clave 11:	Clave 12:	Clave 13:	Clave 14:	Clave 15:

Se proporciona un fichero de referencia client-ref.c con datos ajustados a las especificaciones del ejercicio.