Arquitectura de Sistemas de Información

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

25 de mayo de 2018 Tiempo total: 2 ½ h.

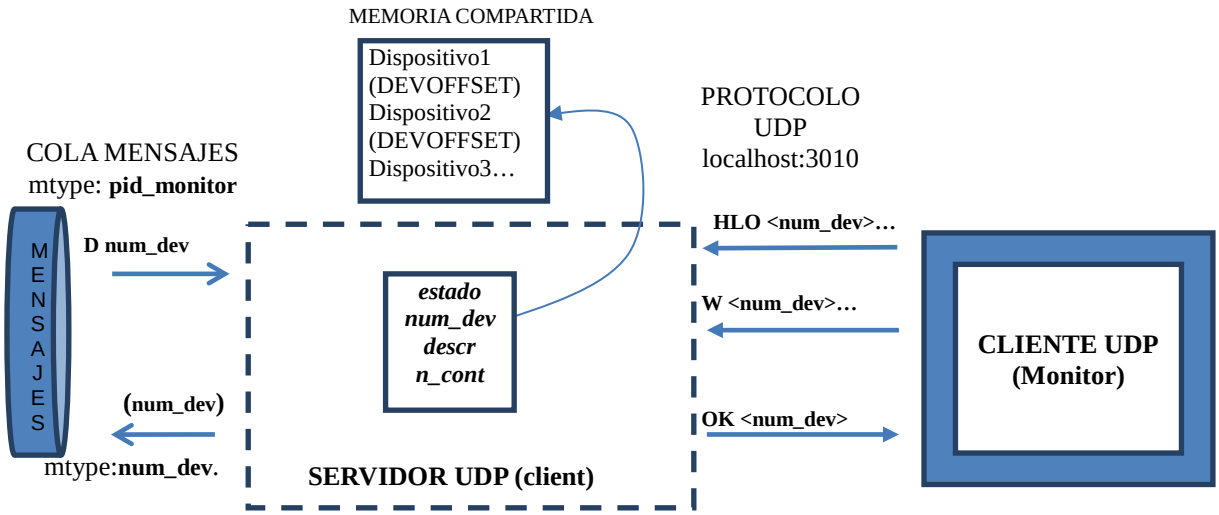
Hay que descubrir 15 claves. El aprobado se establece en 7 claves.

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 y saturación de O_2.

El ejercicio se desarrollará progresivamente en modo incremental, añadiendo en cada paso nuevas funcionalidades al desarrollo y manteniendo la compatibilidad hacia atrás. Esto es, el ejercicio 2 será una ampliación del ejercicio 1, el 3 ampliará el 2 y así sucesivamente.

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



En la memoria compartida se almacenarán los registros de los dispositivos separados entre sí por el offset DEVOFFSET. El primer dispositivo (*num_dev*=1) se situará en la posición 0 de la memoria compartida, el segundo (*num_dev*=2) en la posición DEVOFFSET, el tercero en la posición DEVOFFSET*2 y así sucesivamente. Cada registro mantendrá información del dispositivo y las medidas enviadas por el mismo. El programa *client* esperará que le lleguen peticiones *HLO* para registrar dispositivos en memoria compartida y peticiones *W* para guardar medidas del sensor. Cada mensaje recibido se deberá responder con un mensaje *OK <num_dev>*, donde *num_dev* es el número de dispositivo empleado en el mensaje de petición.

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 comienzo 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 *HLO* al que se responderá con un *OK*. La aplicación a desarrollar escuchará en localhost:3010.

El comando *HLO* será un datagrama procedente del monitor en texto claro con formato:

HLO <num_dev> <n_cont> descr

donde *num_dev* es un valor numérico en ASCII que representa el número de dispositivo, *n_cont* un valor numérico en ASCII que indica el número de contadores del dispositivo y *"descr"* una cadena con su descripción. Se descubrirá la **clave 2** si se responde al monitor con:

OK <num_dev>

Si no se es capaz de descifrar los campos del mensaje *HLO* y generar la respuesta *OK <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 *HLO* habrá que registrar el dispositivo *num_dev* en su posición correspondiente en memoria actualizando y escribiendo 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.

Si se da de alta correctamente el primer dispositivo *num_dev*=1 (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 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 ***índice valor*** (tamaño total 13 bytes)

Cada dispositivo dispone de un tamaño de memoria DEVOFFSET en el que se almacena en primer lugar el registro según la estructura *shm_dev_reg* y en la posición siguiente después de la estructura se almacenan en formato (int) las medidas que vaya recibiendo mediante comandos W. El campo ***índice*** servirá para señalar la medida (int) en la que queremos copiar el ***valor.*** Por ejemplo, *índice*=0 significa que tendremos que escribir *valor* en la primera medida, justo detrás del registro de *num_dev*. Con *índice*=1 escribiremos en la siguiente medida o posición int.

Con el comando *W* se deberá localizar la posición ***índice*** del dispositivo *num_dev* y escribir en ella el ***valor*** (int). El monitor esperará como respuesta: *OK <num_dev>* . Si se recibe correctamente se obtiene la **clave 5.**

Si atiende correctamente la primera petición *W* que realice el monitor (*num_dev*=1, *índice*=0), se obtiene la **clave 6**. Si se atiende correctamente todo el resto de peticiones *W* se consigue la **clave 7**.

Ejercicio 4. Acceso a sección crítica mediante semáforos.

El programa *client* deberá acceder a un array de 5 semáforos creado por el monitor. El primer semáforo no se utilizará y el resto (del 1 al 4) estará asociado por ese orden a cada uno de los cuatro primeros dispositivo en memoria. Por ejemplo, si se quiere acceder al dispositivo 4 habrá que actuar sobre el semáforo 4.

Si se actualizan dispositivos correctamente respetando los semáforos se obtiene la **clave 8**. En este caso, si se repone el estado de los semáforos se consigue la **clave 9**.

Ejercicio 5. Volcado de memoria mediante colas de mensajes.

La operativa de este ejercicio es diferente a la de los anteriores. El programa *client* implementará concurrentemente con el servidor UDP otro servidor que escucha en una cola de mensajes creada por él. Esperará mensajes tipo *pid_monitor* procedentes de monitor.

El monitor le enviará mensajes *Dump* con peticiones de volcado de toda la información referente a un dispositivo en memoria (registro+medidas). La sintaxis de estas peticiones es:

D num_dev

Donde *num_dev* es un entero binario. Se deberá utilizar la estructura *struct msgq_input* incluida en el fichero de referencia *client-ref.c* para la lectura de los mensajes de entrada.

Como respuesta a este comando se espera que el programa copie el dispositivo entero en memoria, incluidos las medidas almacenadas (tamaño total: DEVOFFSET), y lo envíe a la cola mediante un mensaje de tipo *num_dev*. El monitor esperará recibir este mensaje del tipo *num_dev* con el contenido exacto del dispositivo completo. Los valores posibles de *num_dev* serán de 1 a 4.

Se obtendrá la **clave 10** si como respuesta a mensajes tipo *pid_monitor* con una petición *D num_dev*, se responde con un mensaje tipo *num_dev* del tamaño adecuado.

En el caso de que el contenido del mensaje de respuesta sea exacto se obtendrá la **clave 11**. Además, si el acceso a memoria se realiza respetando los semáforos se conseguirá la **clave 12**.

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

Se obtendrá la **clave 14** si con *num_dev* = 0, se bajan los cuatro semáforos (1..4) en una única operación y no en cuatro 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

25 de mayo 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:

Ejercicio 4

Clave 8:	Clave 9:

Ejercicio 5

Clave 10:	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.