Arquitectura de Sistemas de Información

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

18 de mayo de 2016 Tiempo total: 2 ½ h.

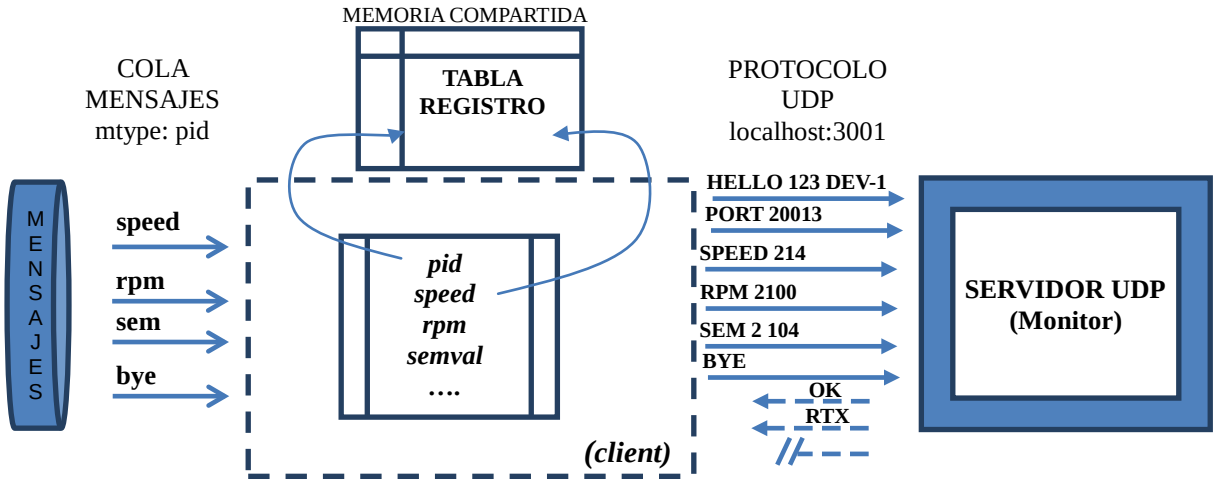
Hay que descubrir 16 claves. El aprobado se establece en la mitad de las claves

Descripción de la prueba práctica

El monitor funcionará como un servidor UDP localizado en el puerto 3001 donde se conectarán dispositivos remotos para informar del estado de determinadas variables de control: velocidad (speed), revoluciones por minuto (rpm) y estado de semáforos (sem).

El ejercicio se desarrollará progresivamente en modo incremental, añadiendo en cada paso nuevas funcionalidades al desarrollo, manteniendo la compatibilidad hacia atrás. Esto es, el ejercicio 2 será una ampliación del ejercicio 1. Así sucesivamente hasta que el ejercicio 3 implemente todas las funcionalidades (todos los secretos).

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



La tabla de registro será un array de registros en un segmento de memoria compartida donde cada registro mantendrá información de estado de cada dispositivo identificado en el sistema. El programa *client* debe identificarse en el servidor via UDP con los comandos HELLO y PORT y posteriormente esperar mensajes en la cola de mensajes (usando su pid como canal de recepción) para obtener valores de los parámetros monitorizados (*speed*, *rpm* y *sem*). Con cada mensaje recibido, debe informar al servidor UDP del estado de esos parámetros y, por otro lado, debe actualizar esos valores en el registro de datos que corresponda a su sesión en la memoria compartida.

Se pueden establecer varias sesiones de forma concurrente en distintos terminales, pero para ello cada dispositivo debe conectarse indicando un nombre diferente. Por ejemplo, en un terminal $./client device1 y en otro $./client device2. El servidor desconectará una sesión si no recibe información en un tiempo determinado (15 seg).

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).

Ejercicio 1. Comunicación UDP Básica

En este primer ejercicio se trata de realizar una comunicación UDP básica que consiste en identificarse, dar un dato y desconectarse. El servidor escuchará en localhost:3001.

El protocolo de comunicaciones se basa en intercambio de datagramas UDP con comandos en texto claro. Los comandos iniciales para esta sesión son:

1.- HELLO pid Nombre (Ej: HELLO 2134 DEVICE-1)

2.- PORT puerto origen de la conexión (Ej: PORT 32456)

3.- BYE

La respuesta a estos comandos (salvo para BYE que no espera respuesta y cierra el programa) será siempre OK con un comentario con la valoración del resultado:

- OK comentario // En comentario se podrá dar información del comando

La clave 1 se obtendrá al conseguir la conexión UDP con un mensaje HELLO con buen formato. La clave 2 se obtendrá al registrar el dispositivo y comprobar que el proceso *client* indicado en el pid del HELLO sigue activo. La clave 3 si se envía el comando PORT con el puerto origen utilizado que es el que servirá para hacer el seguimiento de la sesión del dispositivo. La clave 4 se obtendrá si se finaliza adecuadamente una sesión enviando el comando BYE y saliendo inmediatamente.

Nota: utilizar la función *getsockname* para asignar el puerto origen (más información en $man).

Ejercicio 2. Comunicación UDP COMPLETA sin errores

En este ejercicio se realizará una comunicación completa con todos los comandos disponibles. Se iniciará cada sesión con un comando HELLO seguido de otro PORT con los datos adecuados.

Posteriormente se esperará a recibir parámetros de estado en una cola de mensajes. Los dispositivos mandarán información de actualización de su estado a través de la cola de mensajes del sistema leyendo los mensajes de tipo coincidentes con su identificador de proceso. Los mensajes recibidos en la cola tendrán por formato:

<Byte tipo char><parámetros según comando>

Los tipos de mensajes y formatos recibidos por la cola son:

- Para SPEED: (‘2’)(<velocidad en **ascci**>) Ej: 2<324>
- Para RPM: (‘3’) (valor RPM como **int**) Ej: 3 127
- Para SEM: (‘4’) (número semáforo como **int**) Ej: 4 2
- Para el comando BYE: (‘5’) Ej: 5

Una vez recibidos los mensajes de estado procederán a reenviar al servidor UDP los comandos adecuados. Estos comandos son:

- SPEED (velocidad en ascci) Ej: SPEED 234
- RPM (revoluciones en ascci) Ej: RPM 3455
- SEM (num en ascci) (valor en ascci) Ej: SEM 1 109
- BYE

La clave 5 se conseguirá con el valor adecuado del comando SPEED o el comando RPM. La clave 6 si se completan los dos. La clave 7 si se recoge el valor del semáforo indicado desde el array de 4 semáforos. La clave 8 si se consiguen identificar dos sesiones de forma simultánea.

En el segmento de memoria compartida (SIZE:1024) en la posición 0 de esa memoria, se guarda una tabla con 4 registros de sesiones con estructura dada (*struct st_data*). Si el alumno localiza la sesión activa mediante el campo “name” en cualquiera de los 4 registros y actualiza

los campos pid, rpm y speed, puede obtener más claves. Si se actualiza uno de los parámetros de speed o rpm, se dará la clave 9. Si están ambos, la clave 10. Si se lee el valor del semáforo sometido a control de acceso utilizando primero el semáforo 0 (posición primera del array) para acceder a la sección crítica se darán las claves 11 y 7. La clave 12 se obtendrá si se actualizan en el registro de sesiones los pids de dos o más sesiones concurrentes ($./client device1 y $./client device2 en terminales distintos), cada uno en su registro.

Ejercicio 3. Sesión UDP COMPLETA con pérdidas

En una comunicación UDP se pueden producir pérdidas de datagramas o vencimiento de temporizadores que producen solicitudes de retransmisión. El alumno debe poder implementar estos procedimientos en el protocolo. En este ejercicio Monitor simulará la pérdida de datagramas y/o solicitudes de retransmisión de forma aleatoria para ver cómo responde la implementación del protocolo del cliente.
Ante la recepción de un comando desde el cliente, el servidor puede responder de formas diferentes:

- OK comentario //comando recibido correctamente
- RTX comentario //solicitud de retransmisión del último comando
- Sin respuesta //a los **4 segundos** el cliente debe retransmitir el comando

Monitor simulará en cada sesión procedimientos de retransmisión (RTX) y de pérdida de datagrama. Si el cliente responde adecuadamente en la primera sesión a las solicitudes de retransmisión (RTX) obtendrá la clave 13. Si lo hace en sesiones concurrentes obtendrá la clave 15. Igualmente si completa una sesión con procedimientos de retransmisión por temporización podrá obtener la clave 14 y si lo hace en sesiones concurrentes la clave 16.

Nota:
Para implementar el procedimiento de retransmisión por temporización se propone usar una señal temporizada (alarm(4)) que pueda interrumpir la lectura bloqueante del socket. En la implementación última del sistema algunas señales realmente no interrumpen la lectura bloqueante, salvo que se usen funciones de enmascaramiento. Para evitar este problema, se propone usar la función signal_EINTR() proporcionada en vez de la clásica signal(). Con ello, se podrá interrumpir las lecturas bloqueantes por temporización.

Ejercicio 4. Generación de sesiones en memoria.

Puede ocurrir que el servidor UDP no esté disponible, en ese caso los dispositivos del sistema pueden crear sus propias sesiones en los registros de memoria compartida. En la posición 0 de la memoria se situará una tabla de 4 registros (tipo *struct st_data*) que tiene datos de las sesiones del sistema. El semáforo 3 del array (el cuarto valor) se utilizará para obtener acceso exclusivo a la tabla de sesiones y poder así reservar un registro.

Este ejercicio está pensado como un complemento a los anteriores. Realmente se pueden conseguir todas las claves con los 3 primeros ejercicios. Si un alumno tiene problemas en desarrollar el protocolo UDP, puede conseguir determinadas claves (hasta 8: las de inicio de sesión y la actualización de datos en memoria) por este procedimiento alternativo. Algunas de las claves exigirán dos sesiones en concurrencia.

La operativa del sistema será que el cliente debe dar de alta un registro disponible. Para ello tendrá que cambiar el estado (ST_FREE) del registro a 1 (ST_PID) y rellenar los datos de pid y name del registro. El campo name tiene tamaño 16, pero conviene no usar más de 8 caracteres. El monitor detectará las nuevas sesiones y si todo está correcto enviará mensajes a la cola de mensajes que debe atender el cliente y actualizar los datos en memoria tal y como se pide en los apartados anteriores.

Arquitectura de Sistemas de Información

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

18 de mayo de 2016 Tiempo total: 2 ½ h.

Hay que descubrir 16 claves. El aprobado se establece en la mitad de las claves

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

Ejercicio 1

Clave 1:	Clave 2:	Clave 3:	Clave 4:

Ejercicio 2

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

Clave 9:	Clave 10:	Clave 11:	Clave 12:

Ejercicio 3

Clave 13:	Clave 14:	Clave 15:	Clave 16:

Ejercicio 4
Permite conseguir alguna de las claves precedentes utilizando sesiones en memoria compartida. Se pueden conseguir las claves 2,4,5,6,9,10 con una sesión buena y la 8 y 12 con sesiones concurrentes

Se proporciona un fichero de referencia client-ref.c con datos ajustados a las especificaciones del ejercicio. Se pone aquí la estructura de registro de datos usada para la tabla de sesiones en memoria:

```
#define ST_FREE 0
#define ST_PID 1
#define ST_DATA 2
#define LEN_NAME 16     //Usar como máximo nombres de longitud 8
struct st_data {
        int state;      // State of register
        char name[LEN_NAME];    // Name of device
        int speed;      // velocidad
        int rpm;        // revoluciones por minuto
        int port;       // Original port
        int sem;        // Sem number
        int semval;     // Sem value
        pid_t pid;      // Process identifier
};
```