Arquitectura de Sistemas de Información

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

24 de junio de 2016 Tiempo total: 2 ½ h.

El aprobado se establece con la mitad de las claves.

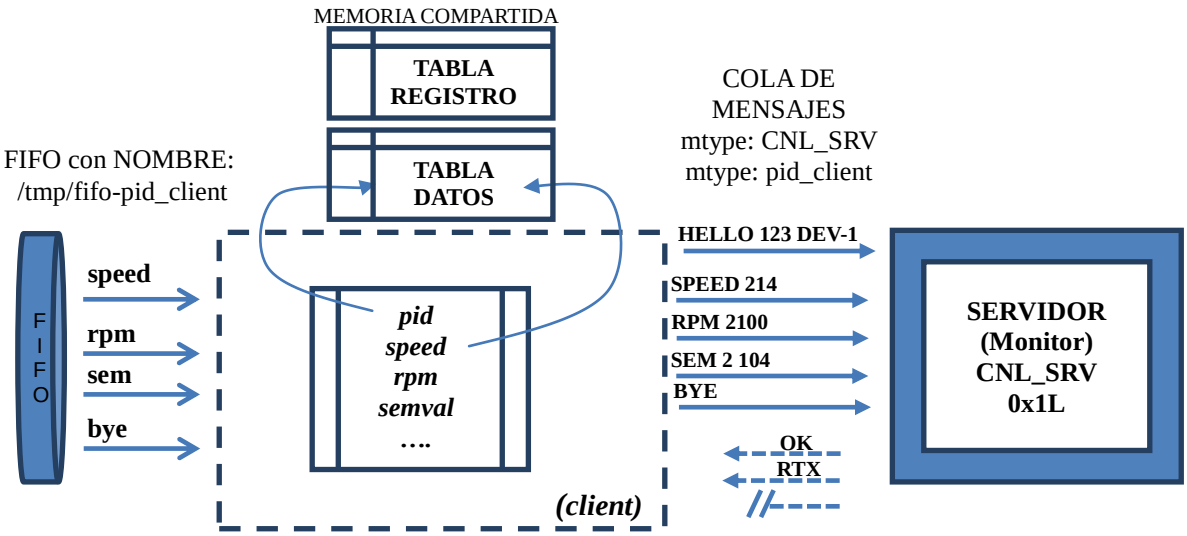
Descripción de la prueba práctica

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: para el DNI 11234567G la clave sería 0x11234567L).

El monitor funcionará como un servidor que recibirá mensajes a través de una cola de mensajes del sistema. Estos mensajes serán del tipo CNL_SRV (0x1L). Los procesos client que debe desarrollar el alumno mandarán mensajes al servidor para informar del estado de determinadas variables de control: velocidad (speed), revoluciones por minuto (rpm) y estado de semáforos (sem). El servidor responderá a cada cliente con mensajes a través de la cola identificados por un tipo coincidente con el pid del cliente enviando el correspondiente mensaje de respuesta.

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 con el comando HELLO y posteriormente esperar mensajes en la cola fifo "/tmp/fifo-pid_client" para obtener valores de los parámetros monitorizados (*speed*, *rpm* y *sem*). Con cada mensaje recibido, debe informar al servidor del estado de esos parámetros y, por otro lado, debe actualizar esos valores en el registro correspondiente de la tabla de datos. Al identificarse cada *client* en el servidor, éste reservará un registro de sesión en la tabla de registro para ese dispositivo. El índice a ese registro se usará también como índice a un registro de datos de la sesión en la tabla de datos.

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).

Ejercicio 1. Comunicación Básica

En este primer ejercicio se trata de realizar una comunicación básica que consiste en crear y abrir la fifo, identificarse, enviar un mensaje de desconexión y desconectarse. El servidor escuchará en la cola en el CNL_SRV. El formato del mensaje que espera recibir el servidor es:

<mtype (long=0x01)><pid (int=pid_client)><comando (texto comando)>

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

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

2.- 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

Los mensajes de respuesta tendrán el formato de mensaje:

<mtype (long=pid_client)><respuesta (texto respuesta)>

La clave 1 se obtendrá al conseguir la conexión con un mensaje HELLO con buen formato. La clave 2 se obtendrá cuando el monitor registre el dispositivo y compruebe que el proceso *client* indicado en el pid del HELLO sigue activo. La clave 3 si se crea y se abre adecuadamente la cola fifo del proceso client. Se trata de una fifo con nombre "/tmp/fifo-pid_client" (donde pid_client es el número de proceso del programa client en ejecución). Esta clave se desvelará cuando monitor compruebe que se ha podido acceder a ella en modo escritura. La clave 4 se obtendrá si se finaliza adecuadamente una sesión enviando el comando BYE y saliendo inmediatamente.

Ejercicio 2. Comunicación COMPLETA sin errores

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

Posteriormente se esperará a recibir parámetros de estado en la cola fifo con nombre asociada al proceso (Ej: /tmp/fifo-1232 para un pid del proceso *client* 1232). Los dispositivos mandarán información de actualización de determinados parámetros (speed, rpm, etc…) a través de esa fifo con nombre. Los mensajes recibidos en la fifo tendrán por formato:

<Byte tipo char><parámetros según comando>

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

- Para SPEED: ('2')(<valor velocidad en **float**>) Ej: 2 127.56
- Para RPM: ('3') (valor RPM en **ascci**) Ej: 3<324>
- Para SEM: ('4') (número semáforo como **int**) Ej: 4 2
- Para el comando BYE: ('5') Ej: 5

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

- SPEED (velocidad en ascci) Ej: SPEED 127.56

- RPM (revoluciones en ascci) Ej: RPM 324
- SEM (num en ascci) (valor en ascci) Ej: SEM 2 109
- 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.

Además de enviar un mensaje al servidor, los mensajes recibidos en la fifo deben actualizar los datos correspondientes en los registros de memoria compartida. En el segmento de memoria compartida (SIZE:1024) en la posición 0 de esa memoria, se guarda la tabla de registro de sesiones (4 sesiones max) con estructura dada (*struct st_reg*). En la posición 300 de la memoria compartida se guarda una tabla de datos con estructura dada (struct st_data) con tantos registros como sesiones potenciales. En esta tabla se guardarán los datos de los parámetros de cada sesión. Si una sesión se activa en el registro 2, se usará ese índice para almacenar sus datos en el registro 2 de la tabla de datos.

El programa client debe localizar el índice (o posición) del registro en la tabla de sesiones donde se haya dado de alta su sesión por el servidor utilizando como parámetro de búsqueda el campo "name". Una vez hallado el registro correspondiente deberá actualizar el campo pid de dicho registro con su pid.

Con ese índice podrá actualizar los datos de los parámetros leídos en la tabla de datos en memoria compartida. 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 COMPLETA con pérdidas

En una comunicación se pueden producir pérdidas de mensajes 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 mensajes 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 mensaje. 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 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_reg*) 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 de la cola de mensajes, 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 fifo con nombre “/tmp/fifo-pid_client” 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

24 de junio de 2016 Tiempo total: 1 ½ a 2 ½ h.

El aprobado se establece con 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 sesiones y la estructura de datos usada para la tabla de sesiones en memoria (posición 0) y la tabla de datos (posición relativa 300):

```
#define ST_FREE 0
#define ST_PID 1
#define ST_DATA 2
#define LEN_NAME 16     // Usar como máximo nombres de longitud 8

#define OFF_REG_TBL     0
#define OFF_DATA_TBL    300

struct st_reg {
        int state;      // State of register
        char name[LEN_NAME];    // Name of device
        pid_t pid;      // Process identifier
};
struct st_data {
        float speed;    // Speed in float format
        int rpm;        // RPM as int
        int sem;        // Sem number
        int semval;     // Sem value
};
```