Arquitectura de Sistemas de Información

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

17 de mayo de 2017 Tiempo total: 2 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).

En este ejercicio MONITOR propone 16 claves de las cuales las 12 primeras consisten en secretos que se ocultan en diferentes recursos del sistema estudiados en las prácticas del curso. Las cuatro últimas son adicionales y se obtienen almacenando por orden las 12 primeras claves obtenidas en un array de enteros cortos sin signo (unsigned short) en una tabla en memoria compartida con desplazamiento OFF_TBL_KEY (16 bytes).

Las 12 primeras claves se han separado por temática en cuatro grupos. Existen algunas dependencias entre claves dentro y fuera de cada grupo, pero en general se pueden obtener independientemente. El orden de identificación de los secretos puede ser por tanto diferente en función de la estrategia elegida por el alumno.

En primer lugar habrá que arrancar el MONITOR. A continuación, arrancaremos nuestro cliente, pero no ejecutaremos ningún ejercicio del mismo antes de pulsar la opción 1 de MONITOR. La opción 1 de MONITOR desplegará los secretos por el sistema en diferentes recursos. A partir de aquí esperará a que el cliente vaya informándole de sus avances a través de la función de librería check_key() proporcionada. La opción 2 de MONITOR servirá para solicitarle que compruebe la correcta realización de algunos secretos aún ocultos.

En el primer grupo de secretos (1-3 ambos inclusive) están algunos relacionados con la utilización de la librería proporcionada. La clave 1 se obtendrá si se invoca de forma adecuada la función connect_MONITOR() desde el cliente. Esta llamada es imprescindible para poder comunicarse con el MONITOR y que éste muestre los avances en pantalla según se solicite mediante la opción 2. Si el alumno no es capaz de realizarla, podrá obtener suficientes secretos de los recursos, pero tendrá que utilizar la hoja de secretos en papel para demostrar sus avances, dado que no es capaz de comunicarse con MONITOR. Posteriormente se solicita al alumno que construya una pipe (tubería sin nombre) y desdoble el proceso mediante un fork en el que el hijo debe invocar a la función de la librería do_child_work() de forma adecuada (mirar el paso de parámetros) para que trabaje con la pipe. El proceso padre debe escribir en la pipe el mensaje ‘HELLO’ y tras esperar un segundo, ponerse a escuchar en la pipe la respuesta de su hijo en formato entero binario. Esta respuesta se utilizará para enviarla a MONITOR invocando adecuadamente la función check_key para el secreto 3 (check_key(3,entero)). Para informar a MONITOR de los valores descubiertos se utilizará la función check_key(clave, valor), donde el segundo parámetro no será necesariamente el valor de la clave, sino que podrá ser un dato descubierto. Por ejemplo, en el caso de la clave 3 el valor entero puede ser distinto a la clave 3.

La clave 2 se desvelará por MONITOR si se ha mandado adecuadamente el mensaje de bienvenida y la jerarquía de procesos desplegada es correcta. Igualmente acabado el trabajo del hijo, éste debe morir, pero el padre deberá esperar 1 segundo antes de terminar.

El segundo grupo de secretos (4-6) corresponde a datos enviados a la cola de mensajes con el canal (message type) correspondiente al PID del programa client y con un formato determinado:
<comando> <dato>

donde <comando> es un carácter 'A' o 'B' para indicar el tipo de comando a realizar y <dato> será un entero binario para el comando 'A' y una cadena de caracteres de tipo numérico para el comando 'B'. En ambos casos el campo dato será contiguo al campo comando (es decir, no hay separación, ni caracteres '<' o '>').

La clave 4 corresponderá al valor numérico de la cadena de caracteres del dato del comando 'B'.
La clave 5 será el valor numérico del entero binario del dato del comando 'A'.
La clave 6 corresponderá al valor numérico de una cadena de caracteres almacenada en la memoria compartida en la tabla de registros en la posición del registro indicado por la clave 5.

La tabla de registros estará situada en memoria compartida en un desplazamiento OFF_REG_KEY (48 bytes) y cada registro consiste en un entero binario o en un array de cuatro caracteres, incluido el terminador (se proporciona una union un_reg_keys para ver su formato en "client.h"). Es decir, un registro de esta tabla puede almacenar la clave como entero de cuatro bytes o como una cadena de caracteres de cuatro bytes terminada en null.

El tercer grupo (7-9) corresponde a secretos almacenados en memoria compartida.
La clave 7 es un entero binario situado en la memoria compartida en la posición OFF_FIRST_KEY (2 bytes de desplazamiento).
La clave 8 es un entero binario situado en la tabla de registros en la posición indicada por la clave 7.
La clave 9 corresponde al valor numérico de la cadena de caracteres numéricos situada en la tabla de registros en la posición indicada por la clave 8.

El cuarto grupo de claves (10-12) corresponde a secretos asociados a manejo de semáforos. Para ello MONITOR habrá creado un array de NSEMS (3) semáforos donde el semáforo de posición 0 se usará para control de acceso a la clave 12 que estará en memoria compartida como un entero binario en la posición OFF_DATA_SEM (10). El semáforo de posición 1 se podrá leer directamente y su valor corresponderá a la clave 10. Si se quiere confirmación de estas claves de MONITOR habrá que usar la función check_key. La clave 11 se podrá obtener si el alumno actualiza el valor del semáforo de posición 2 al valor obtenido del semáforo 1. Para descubrir esta clave se debe utilizar la opción 2 de MONITOR.

Las claves adicionales (13-16) se podrán obtener a través de la opción 2 de MONITOR siempre y cuando se actualice la tabla de claves (correspondiente a las 12 primeras claves) en memoria compartida. Esta tabla de claves consiste en un array de valores de tipo unsigned short que estarán en el desplazamiento OFF_TBL_KEY (16 bytes) de memoria compartida y que corresponde al alumno actualizar según vaya consiguiendo sus claves.

Por cada grupo de 3 claves completado, se proporcionará una clave adicional. Eso quiere decir que se pueden conseguir hasta 4 claves adicionales si se completan las 12 exigidas.

Arquitectura de Sistemas de Información

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

17 de mayo de 2017 Tiempo total: 2 h.

El aprobado se establece con la mitad de las claves.

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

FIRMA:

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

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

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

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

En el fichero "**client.h**" se definen los prototipos de las funciones de la librería a utilizar para la comunicación con MONITOR. También se definen constantes relacionadas con el tamaño de la memoria compartida, offsets de distintos bloques dentro de la memoria, el tamaño del array de semáforos, la unión que define la organización de la tabla de registros, etc.

Para compilar el "**client.c**" habrá que utilizar el compilador "**cc**" ("gcc" no está instalado correctamente en los centros de cálculo) añadiendo en la compilación la librería "**libexamen7.a**".

#include “client.h”

int **connect_MONITOR**(char *dni);

DESCRIPTION

This function makes an internal connection to MONITOR application and prepares the environment to get some special services in order to check key values.

The service must be initialized to a specific user indicated by dni. MONITOR application must be running when the function is called.

RETURN VALUES:

-1: On error accessing to MONITOR interface
1: On success

#include “client.h”

int **check_key** (int key, int value)

DESCRIPTION

This function informs MONITOR about a specific value related to a key. You have a credit (a maximum number of attempts) to guess a key value, if you run out of this credit this key will be blocked. To use this function, previously, you must have connected to MONITOR via connect_MONITOR call.

RETURN VALUES:

-1: problems to communicate with MONITOR
1: Informative request to MONITOR processed

#include “client.h”

int **do_child_work**(int fildes[2]);

DESCRIPTION

This function works with a previoulsy created pipe whose file descriptors are passed in a fildes array. It waits for a Wellcome message (“HELLO”) an after two seconds it writes an integer value associated with a key that users must use to get the actual value of the key. Before ending file descriptors will be closed. A previous connection to MONITOR must be stablished via connect_MONITOR call.

Warning: The parameter fildes must be a pointer to the array of two descriptors generated via pipe() function.

RETURN VALUES:

-1: problems with the pipe
0: problems with welcome message or initialization
1: on successful completion