E.T.S.I. TELECOMUNICACIÓN
FUNDAMENTOS DE SOFTWARE DE COMUNICACIONES

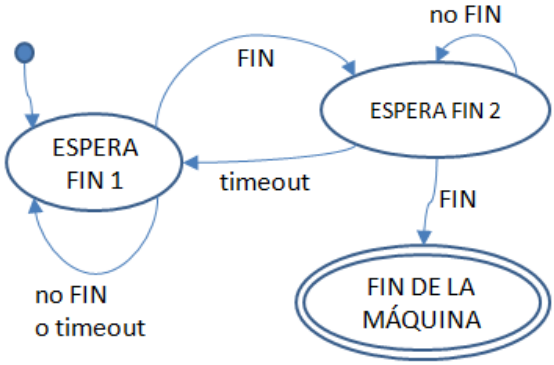
APELLIDOS, NOMBRE:
DNI:
TITULACIÓN:
NÚMERO DEL PC:

IMPORTANTE: SE ENTREGARÁN TODOS LOS FICHEROS SOLUCIÓN EN EL CAMPUS VIRTUAL, SIN COMPRIMIR

ACCESO A LA MÁQUINA VIRTUAL desde VirtualBox: **P1P2-Debian_1C_2C_2017-2018**
Usuario: **alumno**
Contraseña: **alumno**

En el directorio del usuario alumno existe una carpeta llamada FSC con códigos de referencia para los ejercicios del examen (también de directivas **`#include`**).

Ejercicio 1



Codifique un programa que implemente la máquina de estados dibujada en la figura. El programa espera eventos (mensajes de texto) provenientes, o bien desde una **`fifo`** ya existente en el directorio de ejecución, denominada "**`fsc_fifo`**", o bien desde el teclado. Si el mensaje contiene "**`fin`**" o "**`FIN`**" –no tenga en cuenta que haya espacios en blanco antes o después-, se genera el evento de tipo **`FIN`** y la máquina evoluciona. Cualquier otro mensaje genera un evento de tipo **`no_FIN`**. También se activa un temporizador de **`5,005`** segundos cada vez que se espera la llegada de un mensaje de tal forma que, si no aparece ninguno, se genera un evento de tipo **`timeout`**.

Apartado 1.1 (1 punto):

Codifique el esqueleto completo del programa incluyendo la máquina de estados que represente a la figura, aunque dejando la función **`espera_evento()`** vacía (excepto su **`return`**), de tal forma que el código compile sin errores. La comprobación de mensajes se haría dentro de **`espera_evento()`**, así que no se espera ningún tipo de código relacionado con esta finalidad en este apartado.

Se entrega un fichero de nombre ejercicio1.1.c

Apartado 1.2 (3 puntos):

Complete el programa con la función **`espera_evento()`**, donde se encuentra un **`select()`** para atender a la **`fifo`**, al teclado y al temporizador requerido de forma simultánea. Ante la llegada de un mensaje, en la misma función se comprueba si contiene "**`fin`**" o "**`FIN`**". No está permitido utilizar señales en este apartado.

Se entrega un fichero de nombre ejercicio1.2.c

Ejercicio 2

Se ha de diseñar un sistema de monitorización de la carga de procesos de un sistema Linux, que va realizando una copia de seguridad de los datos de forma remota. El sistema cuenta con un cliente (monitor) que, **cada segundo**, consulta el número de **procesos activos** del sistema, y envía ese dato a un **servidor** que los irá **almacenando en un fichero**. Antes de comenzar con la monitorización, el cliente envía al servidor el **nombre del fichero** de la copia de seguridad donde se han de almacenar los datos para su posterior análisis. Se ha de utilizar el protocolo TCP para implementar las comunicaciones, ya que se requiere fiabilidad. Los detalles de cada uno se incluyen a continuación. Ha de asegurarse que se escriben y leen los datos esperados.

Notas sobre la compilación del servidor:

1. Si el servidor se compila con la opción "`-std=c99`", entonces el `accept()` devuelve `-1` si es interrumpido por alguna señal, y hay que hacer que el servidor siga esperando clientes ante esta eventualidad.
2. Si no se compila sin la opción anterior, entonces el `accept()` funciona correctamente ante la llegada de señales.

Apartado 2.1: Servidor (3 puntos)

Funcionamiento:

1. El servidor escucha en el puerto 4950.
2. Por cada petición aceptada, el servidor hace lo siguiente:
   2.1. Lee el nombre del fichero donde guardar los datos del cliente, que viene en formato longitud + cadena.
   2.2. Mientras el cliente está conectado, éste enviará una variable entera tipo `uint16_t` con el número de procesos activos en la máquina donde se ejecuta. El servidor leerá cada dato y lo almacenará en el fichero antes de leer el siguiente.
   2.3. Una vez el cliente se desconecta, se atiende al siguiente.

Se entrega un fichero de nombre ejercicio2.1.c

Apartado 2.2: Cliente (2 puntos)

El cliente es un programa que no necesita intervención del usuario para su funcionamiento:

1. Argumentos de la línea de comandos:
   a. La dirección IP del servidor de encriptación.
   b. El nombre del fichero donde se almacenarán los datos en el servidor.
2. Antes de comenzar la monitorización, el cliente envía el nombre del fichero al servidor de esta forma: primero, la longitud del nombre (`uint8_t`), y después la cadena con el nombre.
3. Entonces comienza a monitorizar de manera infinita hasta que el usuario le envía la señal `SIGINT` (Ctrl + C). Cuando llega la esa señal, se deben liberar los recursos del sistema adecuadamente.
4. La monitorización consiste en que, cada **2 segundos**, el cliente consulta el número de procesos activos en el sistema usando la siguiente llamada:

```
#include <sys/sysinfo.h>

struct sysinfo info;
sysinfo(&info); /* llamada al sistema, chequear errores */
uint16_t dato = info.nprocs;
```

   y lo envía al servidor para que lo almacene.

Se entrega un fichero de nombre ejercicio2.2.c.

Apartado 2.3: Atención de múltiples clientes en paralelo (1 punto)

Reimplemente el servidor del Apartado 2.1 utilizando multiproceso con **`fork()`** y **`wait()`**. No está permitido ignorar señales en el servidor.

Se entrega un fichero de nombre ejercicio2.3.c