Laboratorio de Software de Comunicaciones
E.T.S.I. Telecomunicación

Prácticas de reacción a múltiples eventos de Entrada/Salida

1. Se quiere implementar un programa chat de conversación que esté atento a los mensajes que un usuario quiere mandar desde teclado y a la recepción de otros provenientes de otros usuarios (en este caso, simulados por una fifo de nombre /tmp/fsc_chat). Para que el programa no crezca en dificultad, por el momento, no se va a realizar envío ninguno de mensajes, tan solo recepción. De esta forma, los mensajes que el usuario introduzca por teclado se imprimirán únicamente por pantalla.
   Para enviar mensajes por la fifo al proceso del chat se utilizará una consola aparte, a través del comando *cat*[1]:

   cat > nombre_fifo
   ...escribir texto en la fifo... hasta que se pulse
   CRTL-D

   a. Utilice un select() sobre los dos descriptores de fichero: el de teclado y el de la fifo. Si hay datos listos para lectura por cualquiera de ellos, el select() devolverá por referencia el descriptor (dentro de un conjunto de descriptores) adecuado sobre el que realizar la lectura de datos con read()
   b. Permita que se dibuje algo por pantalla cada tres segundos en el caso anterior (por ejemplo un punto "."), si no hay datos ni de teclado ni de la fifo. Utilice el último argumento de select() para activar este temporizador cada vez que se haga la llamada.

   c. El ejercicio, hasta ahora, plantea serios fallos de diseño. Quizá uno de los más graves consiste en que, cuando el cliente (la consola) se cierra, si se vuelve a abrir no se pueden volver a enviar datos al servicio de chat que sigue en ejecución. Para evitarlo, implementa un cliente de chat que se comunique con el servicio anterior de una forma más robusta:
      -El cliente de chat se va a dedicar, en bucle, a leer datos del teclado y enviarlos por la fifo. Pero antes de esto, tendrá en cuenta lo siguiente:
      -El servicio de chat NO abrirá la fifo hasta que el cliente entre en ejecución. Es decir, lo primero que hará el cliente al arrancar su ejecución es enviar la señal SIGUSR1 al servicio de chat, que la manejará para enterarse de la llegada del cliente. Es en ese momento cuando se realizará la apertura de la fifo en ambos programas: cuidado porque el cliente no será capaz de enviar la señal al servidor si antes hace su open de la fifo ¡se quedaría bloqueado! Tiene que hacerlo después.

[1] El comando *cat* sirve para mostrar por pantalla el contenido de un fichero que se le pasa como argumento. Ejemplo: *cat datos.txt*. Sin embargo, si se utiliza sin argumentos y redirigiendo su salida a un fichero (por eso se utiliza el operador >), permite que los mensajes que escriba el usuario por teclado se escriban en dicho fichero hasta pulsar simultáneamente las teclas Ctrl-D

-El cliente podrá mandar la señal SIGUSR1 al servicio de chat si antes conoce el *pid* del mismo ¿Cómo? Hay diversas maneras de conseguirlo, por ejemplo, si el servicio de chat saca por pantalla su *pid* (porque se ejecuta antes) y el cliente se ejecuta después pasándole dicho *pid* como argumento. Otra opción consiste en que el servicio de chat escriba su *pid* en un fichero (por ejemplo en /tmp/pid_chat) y el cliente lo lea de ahí al ejecutarse.

En los ejercicios se tendrá especial cuidado en controlar los errores en las llamadas al sistema.