Programación declarativa aplicada

Curso 2016/17. Ejercicios – Librerías OTP (Hoja 1)

Instrucciones:

- Solo es necesario entregar el Ejercicio 1.
- Fecha límite: 21 de diciembre de 2016.
- Evaluación: 0.3 puntos (máx.) en la nota final, si se entrega antes de la fecha límite.

1. El objetivo de este ejercicio es implementar un *servidor de trabajos* mediante el comportamiento `gen_server` de las librerías OTP. El servidor mantiene una cola de trabajos pendientes. Un cliente puede:

   - Enviar un trabajo nuevo al servidor para que lo añada al final de la cola.
   - Obtener el trabajo situado al principio de la cola para realizarlo.
   - Notificar al servidor que un trabajo obtenido previamente ha sido finalizado.

   El servidor almacena los *trabajos pendientes* en una lista de funciones. Para realizar uno de los trabajos tan sólo hay que llamar a la función correspondiente. Además de la lista de trabajos pendientes, el servidor también mantiene una lista de *trabajos en progreso*, que son los que han sido asignados a clientes, pero aún no han sido finalizados. Se trata de una lista de tuplas `{Ref, Pid}`, donde `Ref` es un número de referencia unívoco (ver función `make_ref/0`) y `Pid` es el identificador del proceso que está realizando el trabajo.

   Implementa mediante el comportamiento `gen_server` un servidor que procese tres tipos de peticiones:

   `{nuevo_trabajo, F}`
   : Añade un nuevo trabajo al final de la cola.

   `obtener_trabajo`
   : Obtiene el trabajo situado al principio de la cola. El servidor generará una referencia para ese trabajo y devolverá al cliente la tupla `{Ref,F}` con la referencia y la función a ejecutar para realizar el trabajo, o bien el átomo `no` para indicar que no hay más trabajos pendientes en la cola.

     Indicación: Cuidado con el argumento `From` de la función `handle_call`. Este argumento es una tupla `{Pid, Tag}`, donde `Pid` es el identificador del proceso que realiza la petición y `Tag` es un número de referencia generado automáticamente. Puedes ignorar este último para realizar el ejercicio.

   `{trabajo_terminado, Ref}`
   : Notifica al servidor que el trabajo con referencia `Ref` ha sido realizado. El servidor ha de comprobar que el trabajo con identificador `Ref` está en la lista de trabajos en progreso, y que el cliente que realiza la petición es el mismo que el que solicitó el trabajo previamente. El servidor devolverá al cliente el átomo `ok` o `error` según se hayan cumplido o no estas condiciones.

     Indicación: Puedes utilizar las funciones `keyfind/3` y `keydelete/3` del módulo `lists`.

   Define unas funciones llamadas `iniciar`, `nuevo_trabajo`, `obtener_trabajo` y `trabajo_terminado` que arranquen el servidor y envíen los mensajes correspondientes.

2. Modifica el ejercicio anterior para asignar un tiempo límite (en segundos) a los trabajos, para que el servidor pueda despedir a los clientes más tardones. De este modo, si un cliente solicita al servidor un trabajo con tiempo `X`, y no lo termina antes de `X` segundos, el servidor deberá enviar un mensaje `estas_despedido` al cliente y volver a colocar el trabajo en la cola de trabajos pendientes.

Indicación: Necesitarás un proceso adicional que envíe notificaciones periódicas al servidor cada segundo.

3. Implementa, utilizando `gen_server`, el módulo *escucha* propuesto en la Hoja de Ejercicios 1 de concurrencia (Ejercicio 4).