

SISTEMAS OPERATIVOS

Guille Rodríguez González | LS26151
Itziar Sánchez Zardoya | LS26023

ÍNDICE

FASE 1	2
REQUISITOS	2
DISEÑO	2
CUMPLIMIENTO REQUISITOS	3
ESTRUCTURAS DE DATOS	3
EXPLICACIÓN DE RECURSOS USADOS	3
PRUEBAS REALIZADAS	3
COMENTARIOS	4
CONCLUSIONES	4
FASE 2 (/find)	5
REQUISITOS	5
DISEÑO	5
ESTRUCTURAS DE DATOS	5
EXPLICACIÓN DE RECURSOS	6
PRUEBAS REALIZADAS	6
PROBLEMAS OBSERVADOS	7
CONCLUSIONES	7
FASE 3	8
REQUISITOS	8
DISEÑO	8
ESTRUCTURA DE DATOS	9
EXPLICACIÓN DE RECURSOS	9
PRUEBAS REALIZADAS	9
PROBLEMAS OBSERVADOS	10
CONCLUSIONES	10
FASE 4	11
REQUISITOS	11
DISEÑO	11
ESTRUCTURAS DE DATOS	11
EXPLICACIÓN DE RECURSOS	12
PRUEBAS REALIZADAS	12
PROBLEMAS ENCONTRADOS	13
CONCLUSIONES	13
FASE 5	13
REQUISITOS	13
DISEÑO	13
EXPLICACIÓN DE RECURSOS	14
PRUEBAS REALIZADAS	14
CONCLUSIONES	16
FASE 6	17
REQUISITOS	17
DISEÑO	17
EXPLICACIÓN DE RECURSOS	17
PRUEBAS REALIZADAS	17
CONCLUSIONES	18
CONCLUSIONES GENERALES	18

FASE 1

REQUISITOS

Para la **fase 1** de la práctica tenemos que implementar parte del cliente Dozer, una lista dinámica, funciones que trabajen con la lista, y las funciones de lectura y carga de ficheros config.dat y stock.dat

Se ha de tener en cuenta que si recibimos una interrupción SIGINT o SIGTERM, el programa deberá guardar correctamente los datos. Implementar un makefile, para compilar mediante el comando make todos los ficheros necesarios sin recibir ningún warning.

DISEÑO

Para la primera fase hemos implementado los siguientes ficheros .c y .h

Config: Contiene la estructura donde se guardara la ip, puerto e intervalo de actualización (este último solo se usará en Gekko) del servidor Gekko donde nuestro Dozer se conectará. Una vez cargada la información del fichero se pasará a una variable GLOBAL que hay en el programa con esta configuración.

Operator: Contiene la estructura donde se guardará la información del cliente (nombre, stock, lista de acciones). Al cargar la información del fichero se guardará en una variable GLOBAL de este tipo de estructura.

Al iniciar el programa hemos propuesto tener las variables de información Config y Operator como globales para poder acceder desde cualquier parte (RSI), esto incluye el socket de la conexión con Gekko.

Lectura de ficheros y carga en las variables anteriormente comentadas.

Se creará la función irq_handler para la gestión de signals SIGINT y SIGTERM, que obligará a guardar los datos correctamente en el fichero.

Se creara una función para la gestión de los comandos del menú.

Por último se añadirán constantes TRUE, FALSE, FD_SCREEN, FD_KEYBOARD, para facilitar el trabajo y el entendimiento del código.

CUMPLIMIENTO REQUISITOS

Para esta fase se han cumplido con los requisitos de lectura correcta de los ficheros, mostrando los errores necesarios. También se ha probado que al recibir un signal tipo SIGINT y SIGTERM salte la RSI, guarde los datos y salga correctamente del programa (hemos revisado que el fichero donde se guardan los datos tenga el formato correcto).

ESTRUCTURAS DE DATOS

Config

Para guardar la información de la conexión, se ha usado la siguiente estructura.

```
typedef struct {
    int update_interval;
    int host_port;
    char host_ip[16];
} Config;
```

Operator

Para guardar la información, se ha usado la siguiente estructura.

```
typedef struct {
    char name[20];
    float money;
    lista mylbexList;
} Operator;
```

EXPLICACIÓN DE RECURSOS USADOS

Hemos hecho uso de la redifinición de signals para cuando sea enviada una señal de SIGINT, SIGTERM haciendo que llame a la función closeDozer() encargada de liberar memoria dinámica, guardar fichero, cerrar el programa.

PRUEBAS REALIZADAS

Cambiado el nombre del fichero stock.dat a stock.data

```
ls26151@vela:~/SO$ ./dozer.exe
[Dozer] - ERROR: No se pudo abrir el fichero de stock [stock.dat]
ls26151@vela:~/SO$
```

Cambiado el nombre del fichero config.dat a config.data

```
ls26151@vela:~/SO$ ./dozer.exe  
[Dozer] - ERROR: No se pudo abrir el fichero de configuracion del servidor [config.dat]  
ls26151@vela:~/SO$
```

Al salir del Dozer pulsando CTRL+C o bien desde la pequeña Shell que en esta fase solo tiene un único comando “exit”, podemos ver que sale correctamente (foto tomada posteriormente, de ahí que salga el mensaje de conexión con Gekko).

```
guilles>^C[Dozer] - ERROR: La conexion con Gekko se ha cerrado.  
Sayonara  
ls26151@vela:~/SO$ ./dozer.exe  
  
guilles>exit  
[Dozer] - ERROR: La conexion con Gekko se ha cerrado.  
Sayonara  
ls26151@vela:~/SO$
```

COMENTARIOS

Esta fase ha sido corta, pero hemos tenido que trabajar e implementar las signals correctamente.

CONCLUSIONES

En esta fase hemos refrescado cómo se hacían listas de primero, repasado los punteros y los arrays variables. Además hemos introducido los signals.

FASE 2 (/find)

REQUISITOS

En la fase 2 debemos crear la Shell en Dozer e implementar dos comandos de la Shell.

```
show stock  
show me the money
```

Inicializar Gekko, esto significa leer los ficheros de configuración de Gekko y cargarlos en la memoria.

DISEÑO

Por la parte de Dozer, se ha aumentado las opciones de la función menú. Cuando el usuario hace una llamada a show stock, llamaremos a una función encargada de mostrar la información de stock que dispone el usuario.

Por la parte de Gekko, se creará también una variable global Config donde guardaremos la información de conexión del servidor TumblingDice. Para ver que realmente el programa guarda los datos. También cargar la información del listado de acciones.

ESTRUCTURAS DE DATOS

Para esta fase se ha reaprovechado la estructura Config para guardar los datos de conexión con TumblingDice.

Ibex

Para guardar la información de **un** Ibex, se ha creado la siguiente estructura.

```
typedef struct{  
    char code[5];  
    float value;  
    long long int quantity;  
} Ibex;
```

EXPLICACIÓN DE RECURSOS

Para esta fase se han definido los signals para el programa Gekko y se ha creado también una variable global (array de Ibex) para tener el listado de Ibex accesible desde cualquier parte del código, ya que posteriormente los Dozer se conectarán.

Por la parte del cliente Dozer, no ha habido que añadir nada nuevo, tan solo hacer un mostrado de los datos que ya se cargaron en la fase 1 (implementando nuevas funciones).

Para esta fase no ha hecho falta crear ningún thread, ni semáforo.

PRUEBAS REALIZADAS

Ejecutando Dozer

```
ls26023@vela:~/SO> ./dozer.exe

Benvingut al Dozer!
IP-SERVER: 192.168.50.76:8471

janires>
```

Probando el comando show stock

```
guilles>show stock

BBVA-500
SAB-1000
TEF-50
```

Probando el comando show me the money

```
guilles>show me the money

12000.50€

guilles>
```

Probando el comando exit

```
janires>exit
Sayonara
```

Probando un comando erroneo

```
janires>show show  
Comanda incorrecta
```

Para el programa Gekko hemos mostrado por pantalla el listado de Ibex y la configuracion cargada del TumblingDice y ha salido todo correctamente.

PROBLEMAS OBSERVADOS

Como siempre pasa al tratar ficheros con C, hemos tenido problemas para leer los ficheros de configuración.

CONCLUSIONES

En esta fase hemos leído ficheros e implementado la Shell. En esta fase se han tenido que tratar todos los posibles errores que el cliente puede introducir.

FASE 3

REQUISITOS

Para la fase 3 hemos de programar a Gekko para que pueda hacer de cliente con TumblingDice y actualizar los valores de la lista de Ibex. Deberá actualizar cada cierto tiempo la información de los Ibex.

DISEÑO

Para la fase 3 hemos de programar a Gekko para que pueda hacer de cliente con TumblingDice y actualizar los valores de la lista de Ibex. Deberá actualizar cada cierto tiempo la información de los Ibex.

Para ello haremos uso de un thread que gestionará la comunicación con TumblingDice y actualizará los valores del Ibex. Necesitaremos 3 mutex / semáforos.

Mutex alarm (para no realizar esperas activas entre actualización y actualización).

Lo primero que realizaremos al iniciar Gekko (después de lo explicado en anteriores fases) será lanzar el Thread y este automáticamente llamara a `updateIbexList`. Para evitar problemas de conexión al actualizar, hemos pensando que la mejor forma de programarlo es, hacer que no se llame a la función conectar y luego actualizar, si no que se llame directamente a la función `update` y esta si hace falta, intente conectar con TumblingDice.

Si al cabo de un tiempo TumblingDice cae, Gekko intentara reconectar N veces (depende de un parámetro indicado en el `gekko.h`) esperando N segundos (también parámetro). Si después de llegar al máximo de reintentos no ha podido conectar con Tumbling, lanzara una interrupción SIGINT que hará la llamada a cerrar todo Gekko correctamente.

Mientras esté conectado con TumblingDice, enviara cada cierto tiempo una petición de actualización de datos. Si la trama es correcta Tumbling responderá con los valores de los Ibex.

Por cada Ibex, Gekko actualizará. En caso de no encontrar un Ibex en su listado, Gekko saldrá de la función con un código de error y entonces no esperara los segundos indicados para lanzar otra actualización, sino que esperara los N segundos de intento de reconexión.

Por último cuando el programa Gekko cierre, liberara toda la memoria dinámica y enviará una trama de desconexión al servidor TumblingDice (aun habiendo sido cerrado mediante CTRL+C).

ESTRUCTURA DE DATOS

Para comunicarnos con TumblingDice necesitaremos tramas, por lo que hemos creado una estructura llamada Frame que contendrá los datos solicitados en el anexo de la memoria. Además de tener funciones de creación automática de tramas. De esta forma nuestro código de la aplicación será más limpio y menor.

Frame

Para facilitar la comunicación entre los procesos Gekko y TumblingDice.

```
typedef struct {  
    char source[14];  
    char type;  
    char data[100];  
} Frame;
```

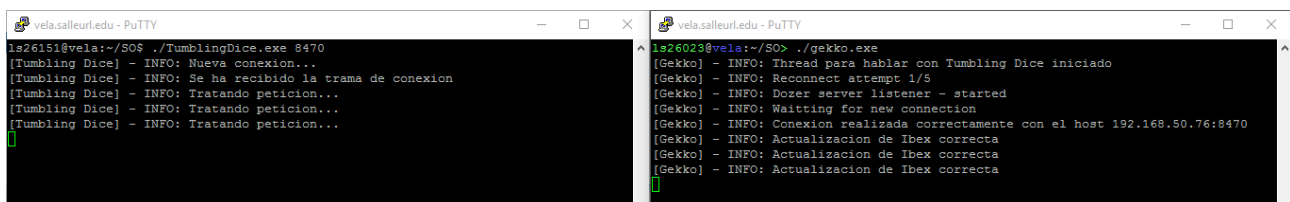
EXPLICACIÓN DE RECURSOS

Usaremos el mutex_alarm para lanzar la actualización de los Ibex.

El thread permitirá que Gekko pueda atender a TumblingDice de forma paralela a la vez que atiende a los Dozer (en las siguientes fases).

PRUEBAS REALIZADAS

Lanzando el programa, Gekko conecta automáticamente con Tumbling y solicita la información de los Ibex. Podemos ver que cada cierto tiempo realiza una nueva solicitud.



```
ls26151@vela:~/SO$ ./TumblingDice.exe 8470  
[Tumbling Dice] - INFO: Nueva conexion...  
[Tumbling Dice] - INFO: Se ha recibido la trama de conexion  
[Tumbling Dice] - INFO: Tratando peticion...  
[Tumbling Dice] - INFO: Tratando peticion...  
[Tumbling Dice] - INFO: Tratando peticion...  
ls26023@vela:~/SO$ ./gekko.exe  
[Gekko] - INFO: Thread para hablar con Tumbling Dice iniciado  
[Gekko] - INFO: Reconnect attempt 1/5  
[Gekko] - INFO: Dozer server listener - started  
[Gekko] - INFO: Waiting for new connection  
[Gekko] - INFO: Conexion realizada correctamente con el host 192.168.50.76:8470  
[Gekko] - INFO: Actualizacion de Ibex correcta  
[Gekko] - INFO: Actualizacion de Ibex correcta
```

Si cerramos el proceso TumblingDice, Gekko se da cuenta y reintenta la conexión (en este caso 5 veces), consiguiendo reconectar si TumblingDice revive antes de que pasen los 5 reintentos.

```

ls26151@vela:~/SO$ ./TumblingDice.exe 8470
[Tumbling Dice] - INFO: Nueva conexion...
[Tumbling Dice] - INFO: Se ha recibido la trama de conexion
^C
ls26151@vela:~/SO$ ./TumblingDice.exe 8470
[Tumbling Dice] - INFO: Nueva conexion...
[Tumbling Dice] - INFO: Se ha recibido la trama de conexion
[Tumbling Dice] - INFO: Tratando peticion...

ls26023@vela:~/SO$ ./gekko.exe
[Gekko] - INFO: Thread para hablar con Tumbling Dice iniciado
[Gekko] - INFO: Reconnect attempt 1/5
[Gekko] - INFO: Dozer server listener - started
[Gekko] - INFO: Waiting for new connection
[Gekko] - INFO: Conexion realizada correctamente con el host 192.168.50.76:8470
[Gekko] - ERROR: No se ha recibido respuesta o bien el socket se ha caido.
*[[A[Gekko] - INFO: Reconnect attempt 1/5
[Gekko] - ERROR: No se ha podido conectar con 192.168.50.76:8470
[Gekko] - INFO: Reconnect attempt 2/5
[Gekko] - ERROR: No se ha podido conectar con 192.168.50.76:8470
[Gekko] - INFO: Reconnect attempt 3/5
[Gekko] - ERROR: No se ha podido conectar con 192.168.50.76:8470
[Gekko] - INFO: Reconnect attempt 4/5
[Gekko] - INFO: Conexion realizada correctamente con el host 192.168.50.76:8470
[Gekko] - INFO: Actualizacion de Ibex correcta
  
```

Si Tumbling no revive, entonces Gekko sale de su ejecución.

```

ls26151@vela:~/SO$ ./TumblingDice.exe 8470
[Tumbling Dice] - INFO: Nueva conexion...
[Tumbling Dice] - INFO: Se ha recibido la trama de conexion
^C
ls26151@vela:~/SO$ ./TumblingDice.exe 8470
[Tumbling Dice] - INFO: Nueva conexion...
[Tumbling Dice] - INFO: Se ha recibido la trama de conexion
[Tumbling Dice] - INFO: Tratando peticion...
^C
ls26151@vela:~/SO$

[Gekko] - ERROR: No se ha podido conectar con 192.168.50.76:8470
[Gekko] - INFO: Reconnect attempt 4/5
[Gekko] - INFO: Conexion realizada correctamente con el host 192.168.50.76:8470
[Gekko] - INFO: Actualizacion de Ibex correcta
[Gekko] - ERROR: No se ha recibido respuesta o bien el socket se ha caido.
[Gekko] - INFO: Reconnect attempt 1/5
[Gekko] - ERROR: No se ha podido conectar con 192.168.50.76:8470
[Gekko] - INFO: Reconnect attempt 2/5
[Gekko] - ERROR: No se ha podido conectar con 192.168.50.76:8470
[Gekko] - INFO: Reconnect attempt 3/5
[Gekko] - ERROR: No se ha podido conectar con 192.168.50.76:8470
[Gekko] - INFO: Reconnect attempt 4/5
[Gekko] - ERROR: No se ha podido conectar con 192.168.50.76:8470
[Gekko] - INFO: Reconnect attempt 5/5
[Gekko] - ERROR: No se ha podido conectar con 192.168.50.76:8470
[Gekko] - ERROR: Despues de 5/5 intentos se cierra Gekko server.
[Gekko] - INFO: Cerrando el programa
  
```

PROBLEMAS OBSERVADOS

Para realizar la reconexión hubo unos problemas con la escritura del socket haciendo que el programa saliera de su ejecución sin dar ningún mensaje del tipo core dumped, segmentation fault. Esto lo miramos con Toni y finalmente conseguimos que funcionara. Se debía a que no cerrábamos el socket antes de volver a reintentar.

CONCLUSIONES

En esta fase hemos tenido que aplicar conocimientos nuevos, los sockets. El Gekko se tenía que conectar con el TumblingDice.

FASE 4

REQUISITOS

En esta fase 4 hemos de realizar la conexión de los Dozer con el servidor Gekko. Gekko debe poder atender a N clientes Dozer a la vez.

DISEÑO

Implementaremos un servidor multithread para atender a las diferentes conexiones de los Dozer, de manera que permitamos la conexión múltiple de los dozer. De esta forma podremos ir comunicándonos con ellos sin problemas.

También deberemos implementar 1 lista dinámica de clientes conectados, para guardar la información de su nombre de operador y el file descriptor del socket.

Como los dozer consultaran la lista, necesitaremos implementar unos mutex nuevos para implementar el código lectores – escritores.

También necesitaremos proteger la lista de conectados, para cuando varios dozer se conecten o se desconecten.

Por último cuando el Dozer se desconecte avisará a Gekko con una trama de desconexión para que este cierre el thread y el socket asociado a el dozer, además de eliminarlo de la lista de conectados. Si Gekko cierra su proceso, antes de finalizar totalmente su ejecución enviara una trama de desconexión a los Dozer y estos lanzarán una señal SIGINT sobre si mismos para cerrar su proceso.

ESTRUCTURAS DE DATOS

Para esta fase se ha reaprovechado la estructura Frame para los Dozer.

Dozer

Para guardar la información de un Dozer en el servidor Gekko..

```
typedef struct{
    char sOperador[14];
    int nFdSocket;
    char data_enviar[100];
}Dozer;
```

EXPLICACIÓN DE RECURSOS

Para esta fase hemos creado un Thread en el Dozer para escuchar a Gekko, hemos ampliado a Gekko en un servidor multicliente que gestionará cada Dozer en un nuevo Thread.

Hemos tenido que implementar los mutex (lector – escritor), y el mutex para la lista de conectados.

Hemos creado lista de clientes conectados como variable Global.

PRUEBAS REALIZADAS

Cuando un cliente se conecta, Gekko muestra un mensaje en la terminal

```
ls26023@vela:~/SO$ ./dozer.exe
Benvingut al Dozer!
IP-SERVER: 192.168.50.76:8471
janires>

[Gekko] - INFO: Actualizacion de Ibex correcta
[Gekko] - INFO: Conexion cerrada del operador: janires
[Gekko] - INFO: Actualizacion de Ibex correcta
[Gekko] - INFO: Actualizacion de Ibex correcta
[Gekko] - INFO: Waitting for new connection
[Gekko] - INFO: Conexion aceptada para dozer con operador: janires
```

Cuando un cliente desconecta avisa a Gekko

```
janires>exit
Sayonara
ls26023@vela:~/SO$

[Gekko] - INFO: Actualizacion de Ibex correcta
[Gekko] - INFO: Actualizacion de Ibex correcta
[Gekko] - INFO: Conexion cerrada del operador: janires
[Gekko] - INFO: Actualizacion de Ibex correcta
```

Conexión múltiple

```
ls26023@vela:~/SO$ ./dozer.exe
Benvingut al Dozer!
IP-SERVER: 192.168.50.76:8471
janires>

ls26023@vela:~/SO2$ ./dozer.exe
Benvingut al Dozer!
IP-SERVER: 192.168.50.76:8471
guille>

ls26023@vela:~/SO$ ./gekko.exe
[Gekko] - INFO: Dozer server listener - started
[Gekko] - INFO: Thread para hablar con Tumbling Dice iniciado
[Gekko] - INFO: Waiting for new connection
[Gekko] - INFO: Reconnect attempt 1/5
[Gekko] - INFO: Conexion realizada correctamente con el host 192.168.50.76:8470
[Gekko] - INFO: Actualizacion de Ibex correcta
[Gekko] - INFO: Conexion aceptada para dozer con operador: guille
[Gekko] - INFO: Waiting for new connection
[Gekko] - INFO: Conexion aceptada para dozer con operador: janires
[Gekko] - INFO: Waiting for new connection
```

Desconexión cuando Gekko cae

```
ls26023@vela:~/SO$ ./dozer.exe
Benvingut al Dozer!
IP-SERVER: 192.168.50.76:8471
janires>
[Dozer] - ERROR: El servidor Gekko ha enviado señal de que cierra. Este terminal se cerrara
Sayonara
ls26023@vela:~/SO$

ls26151@vela:~/SO$ ./dozer.exe
Benvingut al Dozer!
IP-SERVER: 192.168.50.76:8471
guilles>
[Dozer] - ERROR: El servidor Gekko ha enviado señal de que cierra. Este terminal se cerrara
Sayonara
ls26151@vela:~/SO$

ls26023@vela:~/SO$ ./gekko.exe
[Gekko] - INFO: Thread para hablar con Tumbling Dice iniciado
[Gekko] - INFO: Reconnect attempt 1/5
[Gekko] - INFO: Conexion realizada correctamente con el host 192.168.50.76:8470
[Gekko] - INFO: Dozer server listener - started
[Gekko] - INFO: Waiting for new connection
[Gekko] - INFO: Conexion aceptada para dozer con operador: guilles
[Gekko] - INFO: Waiting for new connection
[Gekko] - INFO: Waiting for new connection
[Gekko] - INFO: Conexion aceptada para dozer con operador: janires
[Gekko] - INFO: Actualizacion de Ibex correcta
~c
Cliente guilles 5
Cliente janires 6
[Gekko] - INFO: Cerrando el programa
ls26023@vela:~/SO$
```

PROBLEMAS ENCONTRADOS

Para esta fase el mayor problema encontrado ha sido hacer que los Dozer y Gekko conecten correctamente, detecten o avisen cuando uno de ellos es cerrado.

Cuando Gekko cierra su programa y los clientes cierran, nos hemos dado cuenta que si hemos abierto más de un cliente con la sesión de LS26023 solo cierra uno, sin embargo si abrimos un Dozer con la sesión LS26023 y otro con la LS26151 (siendo mismo código) si que cierran correctamente todos los Dozer al recibir un mensaje de error.

CONCLUSIONES

En esta fase se tienen que conectar más de un Dozer al Gekko. Para ello utilizamos los threads y las listas para guardar los dozers que se van conectando.

FASE 5

REQUISITOS

En esta fase hemos de implementar los comandos de la Shell de Dozer restantes (show Ibex, sell ACCION QTY, buy ACCION QTY), junto con toda su lógica de ejecución para conseguir el correcto funcionamiento.

El apartado show Ibex realizara una petición a Gekko y este responderá con el listado de los 35 Ibex.

El apartado buy acción qty realizara una petición de compra a Gekko, Gekko buscará si hay acciones disponibles, si es así, notificará a los clientes Dozer de los que hayan restado estas acciones, indicándoles cuánto dinero han ganado y notificara al comprador la cantidad a restar de su dinero y las acciones ganadas.

DISEÑO

Implementaremos un servidor multithread para atender a las diferentes conexiones de los Dozer, de manera que permitamos la conexión multiple de los dozer. De esta forma podremos ir comunicándonos con ellos sin problemas.

También deberemos implementar 1 lista dinámica de clientes conectados, para guardar la información de su nombre de operador y el file descriptor del socket.

Como los dozer consultaran la lista, necesitaremos implementar unos mutex nuevos para implementar el código lectores – escritores.

También necesitaremos proteger la lista de conectados, para cuando varios dozer se conecten o se desconecten.

Por último cuando el Dozer se desconecte avisará a Gekko con una trama de desconexión para que este cierre el thread y el socket asociado a el dozer, además de eliminarlo de la lista de conectados. Si Gekko cierra su proceso, antes de finalizar totalmente su ejecución enviara una trama de desconexión a los Dozer y estos lanzarán una señal SIGINT sobre si mismos para cerrar su proceso.

EXPLICACIÓN DE RECURSOS

Hemos creado una lista de ventas para que cuando los operadores pongan a la venta acciones, estas estén en el listado.

Hemos tenido que implementar los mutex (lector – escritor), y el mutex para la lista de ventas.

PRUEBAS REALIZADAS

Cuando el cliente realice el show Ibex en diferentes momentos ha de poder ver la información de los precios, y si estos han variado, verlos variados.

ABG.P	3.22	300000000	ABG.P	3.14	300000000	ABG.P	3.09	300000000
ABE	15.49	200000000	ABE	15.78	200000000	ABE	15.81	200000000
ANA	53.29	100000000	ANA	53.62	100000000	ANA	54.33	100000000
ACS	29.24	50000000	ACS	28.80	50000000	ACS	28.99	50000000
AMS	26.51	75000000	AMS	26.31	75000000	AMS	26.32	75000000
MTS	9.97	32000000	MTS	9.97	32000000	MTS	10.03	32000000
POP	4.99	150000000	POP	4.99	150000000	POP	5.03	150000000
SAB	2.32	2500000	SAB	2.31	2500000	SAB	2.34	2500000
BKIA	1.42	25000000	BKIA	1.40	25000000	BKIA	1.42	25000000
BKT	6.65	450000000	BKT	6.73	450000000	BKT	6.71	450000000

Cuando un usuario pone acciones a la venta.

```
ls26023@vela:~/SO> ./gekko.exe
[Gekko] - INFO: Thread para hablar con Tumbling Dice iniciado
[Gekko] - INFO: Reconnect attempt 1/5
[Gekko] - INFO: Conexion realizada correctamente con el host 192.168.50.76:8470
[Gekko] - INFO: Servidor de Dozers - started
[Gekko] - INFO: Listo para conectar con un nuevo Dozer
[Gekko] - INFO: Listo para conectar con un nuevo Dozer
[Gekko] - INFO: Conexion aceptada para dozer con operador: janires
[Gekko] - INFO: Actualizacion de Ibex correcta
[Gekko] - INFO: El operador janires pone a la venta 100 acciones de BBVA

BBVA-800
SAB-990
TEF-3000

janires>sell BBVA 100
[Dozer] - INFO: OK. Accions posades a la venda.

janires>
```

Si luego mira las acciones a la venta el u otro usuario vera que estas han incrementado.

ANA	53.71	100000000		ANA	52.83	100000000
ACS	28.47	50000000		ACS	26.36	50000000
AMS	26.53	75000000	janires>sell BBVA 100	AMS	27.58	75000000
MTS	10.12	32000000	TRAMA: BBVA-100	MTS	10.39	32000000
POP	4.91	150000000	AUX: 100	POP	4.69	150000000
SAB	2.28	2500000		SAB	2.37	2500000
BKIA	1.41	25000000	OK. Accions posades a la venda.	BKIA	1.40	25000000
BKT	6.73	4500000000		BKT	6.84	4500000000
BBVA	9.16	1300000000		BBVA	8.96	13000000100

Si un usuario retira las acciones

```
[Gekko] - INFO: Thread para hablar con Tumbling Dice iniciado
[Gekko] - INFO: Reconnect attempt 1/5
[Gekko] - INFO: Servidor de Dozers - started
[Gekko] - INFO: Conexion realizada correctamente con el host 192.168.50.76:8470
[Gekko] - INFO: Listo para conectar con un nuevo Dozer
[Gekko] - INFO: Listo para conectar con un nuevo Dozer
[Gekko] - INFO: Conexion aceptada para dozer con operador: janires
[Gekko] - INFO: Actualizacion de Ibex correcta
[Gekko] - INFO: El operador janires pone a la venta 100 acciones de BBVA
[Gekko] - INFO: El operador janires ha retirado 100 acciones de BBVA
[Gekko] - INFO: Actualizacion de Ibex correcta

janires>sell BBVA 100
[Dozer] - INFO: OK. Accions posades a la venda.

janires>erase BBVA 100
Retiradas 100 acciones de BBVA
```

Si un usuario que no tiene acciones de un Ibex a la venta e intenta retirar de ese Ibex.

```
ls26023@vela:~/50> ./gekko.exe
[Gekko] - INFO: Thread para hablar con Tumbling Dice iniciado
[Gekko] - INFO: Reconnect attempt 1/5
[Gekko] - INFO: Servidor de Dozers - started
[Gekko] - INFO: Conexion realizada correctamente con el host 192.168.50.76:8470
[Gekko] - INFO: Listo para conectar con un nuevo Dozer
[Gekko] - INFO: Listo para conectar con un nuevo Dozer
[Gekko] - INFO: Conexion aceptada para dozer con operador: janires
[Gekko] - INFO: Actualizacion de Ibex correcta
[Gekko] - INFO: El operador janires pone a la venta 100 acciones de BBVA
[Gekko] - INFO: El operador janires ha retirado 100 acciones de BBVA
[Gekko] - INFO: Actualizacion de Ibex correcta
[Gekko] - INFO: Actualizacion de Ibex correcta

janires>erase BBVA 100
Retiradas 100 acciones de BBVA

janires>erase BBVA 100
[Dozer] - ERROR: No tienes acciones BBVA puestas a la venta

janires>
```

Comprando acciones a los otros usuarios.

```
gues>
Compra realizada. Coste: 869.70 €
```

Cuando otro operador compra nuestras acciones

```
gues>show me the money

10317.47€

gues>sell TEF 50

[Dozer] - INFO: OK. Accions posades a la venda.

gues>
[GEKKO]: Venta realizada. TEF 50 acciones. 632.75€.
gues>show me the money

10950.22€

gues>
```



```
ls26151@vela:~/SO$ ./dozer.exe
Benvingut al Dozer!
IP-SERVER: 192.168.50.76:8471

aires>
[GEKKO]: Venta realizada. SAB 20 acciones. 50.54€.
aires>
```

```
ls26151@vela:~/SO2$ ./dozer.exe
Benvingut al Dozer!
IP-SERVER: 192.168.50.76:8471

guilles>
[GEKKO]: Venta realizada. SAB 20 acciones. 50.54€.
guilles>
```

```
vela.salleurl.edu - PuTTY
TRE 39.38 369850000
TEF 11.27 445200000
VIS 44.79 361255000

aires>exit
Sayonara
ls26151@vela:~/SO$ ./dozer.exe
Benvingut al Dozer!
IP-SERVER: 192.168.50.76:8471

aires>
[GEKKO]: Venta realizada. BBVA 30 acciones. 277.41€.
aires>exit
Sayonara
ls26151@vela:~/SO$
```

```
vela.salleurl.edu - PuTTY
[Tumbling Dice] - INFO: Tratando peticion...
[Tumbling Dice] - INFO: Tratando peticion...
[Tumbling Dice] - INFO: Tratando peticion...
[Tumbling Dice] - INFO: Cerrando conexion
[Tumbling Dice] - INFO: Nueva conexion...
[Tumbling Dice] - INFO: Se ha recibido la trama de conexion
[Tumbling Dice] - INFO: Tratando peticion...
[Tumbling Dice] - INFO: Tratando peticion...
[Tumbling Dice] - INFO: Tratando peticion...
[Tumbling Dice] - INFO: Tratando peticion...
[Tumbling Dice] - INFO: Tratando peticion...
[Tumbling Dice] - INFO: Tratando peticion...
[Tumbling Dice] - INFO: Cerrando conexion
[Tumbling Dice] - INFO: Nueva conexion...
[Tumbling Dice] - INFO: Se ha recibido la trama de conexion
[Tumbling Dice] - INFO: Tratando peticion...
[Tumbling Dice] - INFO: Tratando peticion...
[Tumbling Dice] - INFO: Tratando peticion...
```

```
vela.salleurl.edu - PuTTY
[GeKKo] - INFO: El operador guilles pone a la venta 10 acciones de BBVA
[GeKKo] - INFO: El operador aires pone a la venta 30 acciones de BBVA
[GeKKo] - INFO: Listo para conectar con un nuevo Dozer
[GeKKo] - INFO: Conexion aceptada para dozer con operador: janires - socket: 7
[GeKKo] - INFO: Recibida peticion de show Ibex del operador guilles
[GeKKo] - INFO: Recibida peticion de show Ibex del operador aires
[GeKKo] - INFO: Actualizacion de Ibex correcta
[GeKKo] - INFO: Recibida peticion de show Ibex del operador guilles
[GeKKo] - INFO: Conexion cerrada del operador: guilles
[GeKKo] - INFO: Conexion cerrada del operador: aires
[GeKKo] - INFO: Listo para conectar con un nuevo Dozer
[GeKKo] - INFO: Conexion aceptada para dozer con operador: aires - socket: 8
[GeKKo] - INFO: Listo para conectar con un nuevo Dozer
[GeKKo] - INFO: Conexion aceptada para dozer con operador: guilles - socket: 5
[GeKKo] - INFO: El operador guilles pone a la venta 20 acciones de VIS
[GeKKo] - INFO: Actualizacion de Ibex correcta
[GeKKo] - INFO: El operador guilles ha retirado 20 acciones de VIS
[GeKKo] - INFO: Conexion cerrada del operador: guilles
[GeKKo] - INFO: Conexion cerrada del operador: aires
```

```
vela.salleurl.edu - PuTTY
guilles>show stock

BBVA-70
SAB-970
TEF-1350
VIS-10

guilles>erase VIS 20

Retiradas 20 acciones de VIS

guilles>exit
Sayonara
ls26151@vela:~/SO2$
```

Por último, al compilar vemos que se cumple la normativa de no warnings.

```
vela.salleurl.edu - PuTTY
ls26151@vela:~/SO$ make
gcc -Wall -Wextra -c gekko.c
gcc -Wall -Wextra -c Frame.c
gcc -Wall -Wextra -c Ibex.c
gcc -Wall -Wextra -c Config.c
gcc -Wall -Wextra -c lista_dozer.c
gcc -Wall -Wextra -c lista_a_la_venta.c
gcc -lpthread gekko.o Frame.o Ibex.o Config.o lista_dozer.o lista_a_la_venta.o -Wall -Wextra -o gekko.exe
gcc -Wall -Wextra -c dozer.c
gcc -Wall -Wextra -c lista.c
gcc -Wall -Wextra -c Share.c
gcc -lpthread dozer.o Frame.o Config.o lista.o Share.o -Wall -Wextra -o dozer.exe
rm -rf *.o
ls26151@vela:~/SO$
```

CONCLUSIONES

En esta fase se han utilizado todos los recursos vistos en clase. Para implementar las funciones se han introducido también los mutex para proteger las listas de ventas, dozers...

FASE 6

REQUISITOS

En esta fase hemos de implementar el comando de la Shell de Dozer erase, junto con toda su lógica de ejecución para conseguir el correcto funcionamiento.

El apartado erase acción qty realizará una petición de borrar a Gekko, Gekko buscará en la lista de acciones a la venta, si están ahí, notificará al cliente Dozer que las acciones que había puesto a la venta han sido canceladas y se las devolverá.

DISEÑO

Se utiliza la lista de acciones a la venta para verificar que el Dozer que quiere cancelar sus acciones, las tiene añadidas a la lista de ventas. Si es así, modificará o borrará sus acciones, dependiendo de la cantidad que quiera borrar.

EXPLICACIÓN DE RECURSOS

Se reaprovecha la lista de acciones a la venta creada en la fase anterior.

PRUEBAS REALIZADAS

Al poner a la venta 100 acciones de BBVA, si queremos cancelar 50 de ellas hay que hacer: erase BBVA 100. Aparecerá un mensaje que diga: Retiradas 50 acciones de BBVA.

Además, permanecerán las otras 50 que había puesto a la venta en la lista de ventas.

```
Benvingut al Dozer!  
IP-SERVER: 192.168.50.76:8471  
  
janires>sell bbva 100  
  
[Dozer] - INFO: OK. Accions posades a la venda.  
  
janires>erase bbva 50  
  
Retiradas 50 acciones de BBVA  
  
janires>
```

CONCLUSIONES

En esta práctica hay que tener cuidado de que no borre acciones que no hemos puesto a la venta previamente.

CONCLUSIONES GENERALES

Esta práctica nos ha ayudado a poder utilizar de forma gradual, todos los conocimientos que hemos visto en clase.

Al estar dividida la práctica en diferentes fases nos ha servido para comprobar que cada fase funcionara correctamente para poder seguir adelante. De esta manera nos ha ayudado a cerciorarnos que todo funciona, y no pasar a la siguiente fase hasta que no lo haga correctamente y evitar arrastrar errores a otras fases.

Además nos hemos dado cuenta de la potencia que tiene el lenguaje C en comparación con los conocimientos que teníamos de primero y a codificar de manera eficiente. En la asignatura de Sistemas Operativos no sólo nos conformamos con que funcione la práctica, sino que lo tiene que hacer eficientemente.