Arquitectura de Sistemas II
Lab 4. Aplicación distribuida multihebra y segura

Prof.: Marisol García Valls (mvalls@it.uc3m.es)
Arquitectura de Sistemas II
Grado Ingeniería Telemática, 3^o
Universidad Carlos III de Madrid

Abril 2018

Objetivos y normas de realización

Objetivos:

En esta práctica los alumnos aprenderán a crear una aplicación distribuida con Ice y con lenguage C++ que será multihebra y ofrecerá concurrencia de forma segura. Los objetivos de esta práctica son aprender:

- (*i*) diseñar e implementar una aplicación distribuida haciendo uso de las bibliotecas proporcionadas por un software de intermediación (como Ice),
- (*ii*) utilizar programación con hebras para la sincronización de la ejecución,
- (*iii*) programación en C++ de forma básica, es decir, basándose en patrones proporcionados en la práctica y en teoría. El alumno deberá también investigar de forma autónoma cuestiones sencillas sobre programación C++ [1] [2]

Normas:

- Deberá realizar una **utilización intensiva de la documentación en línea de Linux** (`man` pages) y documentación online sobre APIs y funciones y de la bibliografía del curso, especialmente el libro de *Robert Love*.
  Es imprescindible que consulte el manual de ayuda de Linux preferiblemente en línea[3].
- **Importante:** Es imprescindible que el alumno consulte la bibliografía recomendada para esta sesión que comprende capítulos seleccionados del manual de Ice[4].
- *Deberá seguir las convenciones de nomenclatura que se dan en el enunciado.*
- *La práctica deberá ser ejecutada sobre un entorno Linux no virtualizado.*
- **Indente de forma clara su código**. Si no es así, se considerará **ilegible**.
- **Fecha de entrega parcial**: 23 de abril de 2018 23:59
- **Ficheros a entregar**: `controlclientes.cpp`, y `memoria.pdf`.

[1] B. Stroustrup. "The C++ programming Language". 4^{th} Edition. Addison-Wesley. 2013.
[2] S. B. Lipmann, J. Lajoie, B. E. Moo. "C++ primer". 5^{th} Edition. Addison-Wesley. 2013.
[3] http://man7.org/linux/man-pages/man7/
[4] http://download.zeroc.com/Ice/3.5/Ice-3.5.1.pdf

- **Fecha de entrega final**: 8 de mayo de 2018 23:59
- **Ficheros a entregar**: `cliente.cpp`, `servidor.cpp`, `interfaz.ice` y `memoria.pdf`.

Sistema distribuido para la gestión de clientes de una operadora de servicios de red

1 Gestión centralizada

Se desea construir una aplicación de gestión de clientes de una operadora de servicios de Internet. Este sistema se codificará en C++ y se ejecuta sobre un nodo con un proceso principal al que nos referiremos como *Control de Clientes* que ofrece la siguiente interfaz.

```
[1] Imprimir datos de clientes
[2] Alta de usuario
[3] Baja de usuario
[4] Cambio de tarifa
[5] Actualizar descuentos
[6] Terminar

Opcion:
```

En la figura 1 se muestra una vista esquemática de las funciones que realiza el nodo de *control de clientes*. Éste, además, se apoya en la existencia de dos hebras `h_desc` y `h_factura` para realizar operaciones adicionales sobre los datos de clientes.

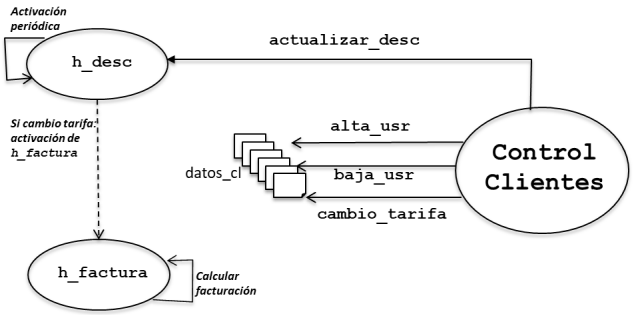

Figure 1: *Nodo que implementa el control de los clientes. Los datos de clientes están almacenados en este nodo en un array* `datos_cl`.

Almacenamiento de clientes en `datos_cl`:

El servidor guarda toda la información de los clientes en un array `datos_cl`. La operadora tendrá un tamaño máximo de clientes definidos por la constante `MAX_CLIENTES`. Cada posición almacena la información de un cliente, de forma que cada posición es de un tipo `info_cliente` como sigue:

```
struct info_cliente{
  unsigned dni;
  string nombre;
  char tarifa;
  unsigned alta;
  unsigned descuento;
};
```

La figura 2 muestra un ejemplo de array con tres clientes.

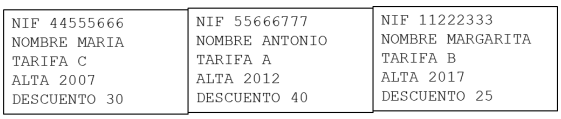

Figure 2: *Ejemplo de datos de clientes: el array* `datos_cl`.

Descuentos:

Las tarifas y los descuentos deben estar asignados de forma coherente de la siguiente manera:

- Los clientes dados de alta antes de 2008 con tarifa `A` deben tener un descuento del 30%.
- Los clientes dados de alta entre los años 2009 y 2012 con tarifa `A` deben tener un descuento del 40%.
- Los clientes dados de alta con posterioridad a 2012 con tarifa `A` tendrán un descuento del 25%.

de lo contrario su descuento será 0%. En el momento de dar de alta a un usuario se puede introducir cualquier descuento (para captar al cliente).

Tarifas:

Existen tres tipos de tarifas: `A` (que incluye el mayor número de servicios), `B` (incluye sólo parte del conjunto total de servicios posibles), y `C` (que incluye únicamente servicios muy básicos). En cualquier momento, puede solicitarse un cambio de tarifa.

Paso 1 – Implementación de las operaciones principales

Las siguientes operaciones principales se implementan en una función cada una con los nombres:

- `alta_usr`: dar de alta a un cliente en el sistema proporcionado su `dni` que es una variable de tipo *entero sin signo* y el resto de campos de la estructura `info_cliente`. Si el cliente no figura en el sistema se le da de alta, si no, se devuelve una indicación de que ya figura en el sistema.
- `baja_usr`: se proporciona el **dni** de un cliente y si éste existe, se le da de baja.
- `cambiar_tarifa`: se proporciona el **dni** de un cliente y la nueva **tarifa**. Independientemente de la tarifa que tuviera anteriormente dicho cliente, ésta se cambia a la nueva tarifa.
- `imprimir_datos_cl`: muestra por pantalla los datos de los clientes.
- `terminar`: acabar el programa.

Paso 2 – Implementación de las operaciones adicionales en configuración multihebra

En la aplicación de control de clientes se realizan dos operaciones adicionales:

- *Actualización de descuentos*: Función realizada por la hebra `h_desc` que se ejecuta cuando el administrador selecciona la opción [5] y consiste en recorrer uno a uno el vector de clientes de forma que, para cada cliente:
  1. se comprueba si su descuento es correcto según se indica en la subseción “Tarifas”,

2. si es correcto, no se modifica nada y se pasa al siguiente cliente,
3. si no es correcto, se actualiza el valor del descuento según se indica en la subsección "Tarifas",

Esta operación se realizará en bucle de forma periódica con una periodicidad programada al inicio (en el momento de lanzamiento del programa). Una vez creada la hebra `h_desc`, ésta no muere hasta el final del programa.

- *Cálculo de la facturación anual estimada*: Función realizada por la hebra `h_factura` y consiste en recorrer el array de clientes para calcular la facturación anual que suponen los clientes actuales. La facturación anual se calcula: tarifa `A` supone 800€al año, tarifa `B` supone 600€y tarifa `C` supone 300€. Cada vez que calcule el nuevo importe estimado de facturación anual, ésta lo imprimirá por pantalla. Una vez creada, la hebra `h_factura` no muere hasta el final del programa.

Importante: En cada periodo de activación de la hebra `h_desc`, en cuanto ésta detecte que hay que realizar un cambio de tarifa en algún cliente, deberá avisar a la hebra `h_factura` para que recalcule la facturación estimada. Para este aviso, deberá utilizarse una **variable de condición**.

Ejemplo de ejecución

A continuación se muestra el interfaz mostrado por el proceso principal y un ejemplo de ejecución:

```
[1] Imprimir datos de clientes
[2] Alta de usuario
[3] Baja de usuario
[4] Cambio de tarifa
[5] Actualizar descuentos
[6] Terminar

Opcion: 1

Listado de clientes:
1: 44555666 | MARIA | C | 2007 | 30
2: 55666777 | ANTONIO | A | 2012 | 40
3: 11222333 | MARGARITA | B | 2017 | 25

[1] Imprimir datos de clientes
[2] Alta de usuario
[3] Baja de usuario
[4] Cambio de tarifa
[5] Actualizar descuentos
[6] Terminar

Opcion: 2

DNI del usuario: 12345678
Nombre del usuario: CARMEN
Tarifa inicial: C
Fecha de alta: 2018
Descuento inicial: 25

Solicitud de alta al servidor central en curso...
Resultado: Usuario dado de alta.

[1] Imprimir datos de clientes
```

```
[2] Alta de usuario
[3] Baja de usuario
[4] Cambio de tarifa
[5] Actualizar descuentos
[6] Terminar

Opcion: 4

DNI del usuario: 44555666
Nueva tarifa: C

Solicitud de cambio de tarifa al servidor central en curso...
Resultado: Cambio de tarifa efectuado.

[1] Imprimir datos de clientes
[2] Alta de usuario
[3] Baja de usuario
[4] Cambio de tarifa
[5] Reseteo de descuentos
[6] Terminar

Opcion: 5

Solicitud de activacion de actualizacion de tarifas al servidor central en curso...
Resultado: Actualizacion automatica de tarifas activado en el servidor.
Nueva facturacion estimada: 1700 euros

[1] Imprimir datos de clientes
[2] Alta de usuario
[3] Baja de usuario
[4] Cambio de tarifa
[5] Reseteo de descuentos
[6] Terminar

Opcion: 6

Avisando a cliente de la terminacion!
Servidor terminado!
```

Plazo de entrega de versión centralizada: 23 de abril de 2018: 23:59

Memoria. Deberá ocupar una extensión máxima de 3 páginas (11 pt. times new roman) y deberá contener las siguientes explicaciones:

1. *Instrucciones de compilación.* Describa (a modo de manual) las instrucciones exactas para la compilación de su programa en C++ en las máquinas de los laboratorios de Telemática.
2. Prototipos de las funciones implementadas. Indique de forma completa los prototipos de cada función implementada y describa las principales decisiones de diseño en cada una de ellas (si las hay). Un posible ejemplo de prototipo es: `int alta_usr(unsigned int dni, string nombre, char tarifa, unsigned int fecha, unsigned int descuento);`
3. *Esquema de hebras* y el *esquema de sincronización y comunicación* entre ellas. Explique qué mecanismos de sincronización ha utilizado para llevar a cabo los comportamientos solicitados entre las hebras `h_desc` y `h_factura`. Explique cómo ha incluido las librerías necesarias para su utilización desde código C++.
4. *Explique cómo ha conseguido resolver algunas de las principales cuestiones en C++)¡*: creación y gestión de arrays, strings (o `char *`), entrada/salida, etc.

2 Versión distribuida

Ahora se desea distribuir la aplicación anterior de forma que las funciones que se describen anteriormente puedan ser invocadas por diferentes administradores de forma simultánea en nodos remotos (nodos "cliente") como muestra la figura 3.

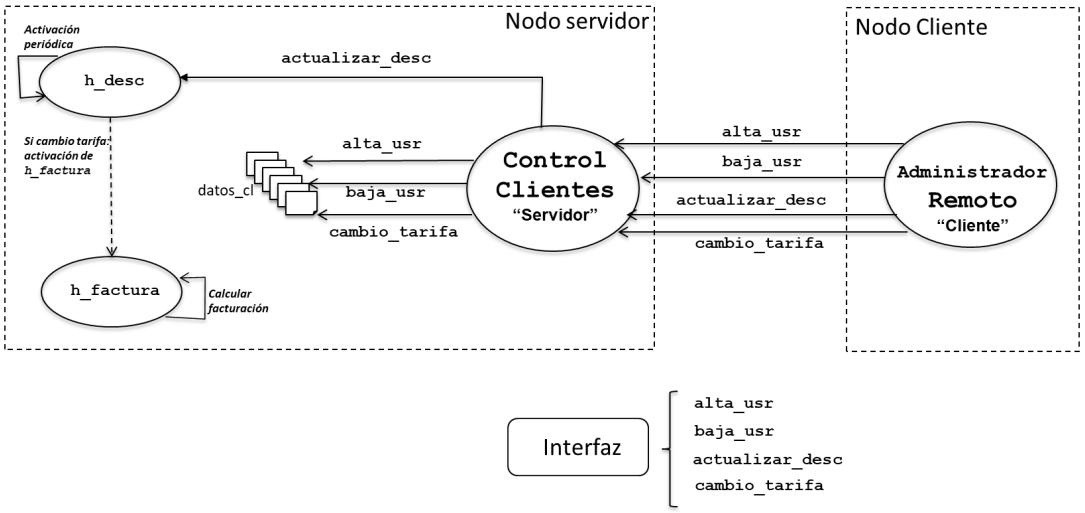

Figure 3: *Aplicación distribuida para la gestión de clientes.*

Por lo tanto, la interfaz de gestión en la que se muestran las opciones [1] a [6] tendrá que ser presentada por el nodo cliente. Cada una de estas opciones generará que el cliente invoque la correspondiente función del servidor. Además, deberá resolverse la posible **concurrencia sobre el vector** `datos_cl` que puede darse al invocar varios administradores funciones que acceden a él de forma remota.

En la opción [5], una vez arrancada la opción de actualizar descuentos, cada vez que se recalcule la facturación del proveedor, deberá enviarse esta información a los clientes (se considerará correcto si sólo se consigue para un cliente) y éstos deberán presentar la información por pantalla como un aviso tal y como se muestra a continuación.

La impresión que se conseguirá en el cliente es:

```
[1] Imprimir datos de clientes
[2] Alta de usuario
[3] Baja de usuario
[4] Cambio de tarifa
[5] Actualizar descuentos
[6] Terminar

Opcion: 5

Solicitud de activacion de actualizacion de tarifas al servidor central en curso...
Resultado: Actualizacion automatica de tarifas activado en el servidor.

[1] Imprimir datos de clientes
[2] Alta de usuario
[3] Baja de usuario
[4] Cambio de tarifa
```

```
[5] Actualizar descuentos
[6] Terminar

Opcion:


Aviso de facturacion recibido del servidor: 1700 euros
```

La impresión que se conseguirá en el servidor es:

```
$ ./servidor &

Recibida solicitud de alta de cliente.

Recibida solicitud de baja de cliente.

Recibida solicitud de actualizacion de tarifas.
Nueva facturacion estimada: 1700 euros --> Aviso enviado a los clientes.

Recibida solicitud de terminacion.
Servidor terminado!
```

Es decir, cada vez que el servidor recibe una petición remota desde un cliente, imprimirá en pantalla esta petición. Además si recibe una solicitud de actualización de tarifas, imprimirá el recálculo de la facturación cuando ésta se produzca.

Parte opcional:

Impresión del listado de clientes ***en el cliente****. Ésta es una función compleja ya que implica recibir datos en un array.*

Importante: Deberá probar el cliente y servidor en remoto, es decir, en dos máquinas físicas diferentes.

Plazo de entrega de versión distribuida: 8 de mayo de 2018: 23:59

Memoria. Deberá ocupar una extensión máxima de 3 hojas (11 pt. Times New Roman) y deberá contener las siguientes explicaciones:

1. *Instrucciones de compilación.* Describa (a modo de manual) las instrucciones exactas para la compilación de su programa en las máquinas de los laboratorios de Telemática.
2. *Interfaz* (`interfaz.ice`). Muestre la interfaz que ha programado. Si ha programado varias interfaces nómbrelas de forma clara y explíquelas indicando su contenido. Indique los tipos de Slice que ha tenido que utilizar para la creación de la interfaz. Muestre la correspondencia de los tipos Slice usados a los de C++.
3. *Esquema de sincronización y comunicación.* Explique qué mecanismos de sincronización ha utilizado para llevar a cabo los comportamientos solicitados y la protección de los datos comules.
4. Explique cómo ha conseguido distribuir físicamente su aplicación: qué objetos remotos hay, adaptadores, qué interfaces, qué módulos, y todo lo necesario en cuanto a las facilidades de Ice que haya utilizado.
5. *Comunicación de servidor a cliente.* Explique cómo ha conseguido que el cliente reciba y muestre el aviso enviado por el servidor cada vez que se produce un recálculo de una tarifa.

Nota:

*La **implementación completa y correcta de todas las funciones de la parte distribuida** (excepto la parte opcional de impresión de clientes en el cliente) permite obtener hasta 1 punto extra en la evaluación continua y, además, supondrá la exención de contestar en el examen las preguntas relativas a Ice. La profesora podrá llamar a grupos completos o a alumnos individuales de algunos grupos que realicen esta parte para defender proyecto de forma oral.*

Tras la corrección de la parte opcional, se indicará en las calificaciones: la nota (0 a 1) y si el alumno está exento de la parte de Ice en el examen.