Práctica 2: Gestión de suscripciones

Programación 2

Curso 2022-2023

En esta segunda práctica de la asignatura desarrollarás un gestor de suscriptores para una plataforma de *streaming*. Los conceptos necesarios para desarrollar esta práctica se trabajan en el *Tema 1*, *2* y *3* de teoría.

Condiciones de entrega

- La fecha límite de entrega para esta práctica es el **viernes 31 de marzo**, hasta las **23:59**
- Debes entregar un único fichero llamado `prac2.cc` con el código de todas las funciones

Código de honor

Si se detecta copia (total o parcial) en tu práctica, tendrás un **0** en la entrega y se informará a la dirección de la Escuela Politécnica Superior para que adopte medidas disciplinarias

Está bien discutir con tus compañeros posibles soluciones a las prácticas
Está bien apuntarte a una academia si sirve para obligarte a estudiar y hacer las prácticas

Está mal copiar código de otros compañeros para resolver tus problemas
Está mal apuntarte a una academia para que te hagan las prácticas

Si necesitas ayuda acude a tu profesor/a
No copies

Normas generales

- Debes entregar la práctica exclusivamente a través del servidor de prácticas del Departamento de Lenguajes y Sistemas Informáticos (DLSI). Se puede acceder a él de dos maneras:
  - Página principal del DLSI (`https://www.dlsi.ua.es`), opción "ENTREGA DE PRÁCTICAS"
  - Directamente en la dirección `https://pracdlsi.dlsi.ua.es`
- Cuestiones que debes tener en cuenta al hacer la entrega:
  - El usuario y la contraseña para entregar prácticas son los mismos que utilizas en UACloud
  - Puedes entregar la práctica varias veces, pero sólo se corregirá la última entrega
  - No se admitirán entregas por otros medios, como el correo electrónico o UACloud
  - No se admitirán entregas fuera de plazo
- Tu práctica debe poder ser compilada sin errores con el compilador de C++ existente en la distribución de Linux de los laboratorios de prácticas

- Si tu práctica no se puede compilar su calificación será 0
- El 70 % de la nota de la práctica dependerá de la corrección automática, por lo que es imprescindible que respetes estrictamente los textos y los formatos de salida que se indican en este enunciado. El otro 30 % dependerá de la revisión manual del código que haga tu profesor de prácticas, por lo que debes también ajustarte a la guía de estilo de la asignatura. Además se tendrá en cuenta que hayas actualizado tu código en *GitHub* todas las semanas
- Al comienzo de todos los ficheros fuente entregados debes incluir un comentario con tu NIF (o equivalente) y tu nombre. Por ejemplo:

prac2.cc

```
// DNI 12345678X GARCIA GARCIA, JUAN MANUEL
...
```

- El cálculo de la nota de la práctica y su relevancia en la nota final de la asignatura se detallan en las transparencias de presentación de la asignatura (*Tema 0*)

1. Descripción de la práctica

La plataforma Streamflix quiere detectar cuál es la dirección IP principal de cada uno de sus suscriptores, de forma que pueda impedir el acceso desde cualquier otra dirección. Tu labor para lograr tan noble propósito es implementar un programa que gestione los suscriptores, y que permita detectar cuál es la dirección IP principal desde la que se conectan. El programa permitirá añadir y borrar suscriptores, y añadir direcciones IP a los suscriptores para determinar cuál es la principal. También habrá opciones para importar y exportar los datos de la plataforma y sus suscriptores a distintos formatos usando ficheros de texto y binarios.

2. Detalles de implementación

En el Moodle de la asignatura se publicarán varios ficheros que necesitarás para la correcta realización de la práctica:

- `prac2.cc`. Este fichero contiene un esqueleto de programa sobre el que realizar tu práctica. Descárgalo y añade tu código en él. Este fichero contiene la siguiente información:
  - Los registros (`struct`) necesarios para hacer la práctica
  - Un tipo enumerado `Error` que contiene todas las posibles clases de error que se pueden dar en esta práctica (por ejemplo, `ERR_OPTION`)
  - Una función `error` que se encarga de mostrar el correspondiente mensaje de error por pantalla en función del parámetro que se le pase. Por ejemplo, cuando la función recibe el parámetro `ERR_OPTION`, mostrará por pantalla el mensaje `ERROR: wrong menu option`
  - Una función `showMainMenu` que muestra por pantalla el menú principal
  - Los prototipos de las funciones principales que se deben implementar obligatoriamente para gestionar las opciones del programa. **Puedes modificar los parámetros de entrada y valor devuelto de estas funciones**. Lo único que no puedes hacer es cambiarles el nombre. Además de éstas, deberás incluir las funciones que consideres oportunas para que tu código sea más sencillo y fácil de leer e interpretar por un humano (más concretamente, por el profesor que tiene que corregir tu práctica).
  - Una función `main()` que implementa la gestión del menú principal y llama a las funciones correspondientes dependiendo de la opción elegida por el usuario

- `autocorrector-prac2.tgz`. Contiene los ficheros del autocorrector para probar la práctica con algunas pruebas de entrada. La corrección automática de la práctica se realizará con un programa similar, con estas pruebas y otras más definidas por el profesorado de la asignatura
- `prac2`. Fichero ejecutable de la práctica (compilado para máquinas Linux de 64 bits) desarrollado por el profesorado de la asignatura, para que puedas probarlo con las entradas que quieras y ver la salida correcta esperada

- En la corrección de la práctica se introducirán siempre datos del tipo correcto, aunque con valores que pueden ser incorrectos. Por ejemplo, si se pide el identificador de un suscriptor, se probará siempre con un valor entero, que podría ser `-1237` o `0`, pero nunca se introducirá un valor de otro tipo (carácter, `string`, número real, ...)
- Se recomienda crear una función similar a `error()` pero para mostrar los distintos mensajes de solicitud de datos que aparecen durante el programa
- El resto de decisiones en la implementación de la práctica quedan a tu criterio, pero ten en cuenta que el código fuente será revisado por tu profesor de prácticas siguiendo la guía de estilo publicada en el Moodle de la asignatura. Parte de la nota de la práctica depende de dicha revisión

3. Componentes

El programa a desarrollar debe poder gestionar dos elementos fundamentales: los *suscriptores*, que almacenarán la información sobre todos los clientes de la empresa, y la *plataforma*, que almacenará los suscriptores en una lista. Los siguientes apartados describen la estructura de cada uno de estos componentes en detalle.

Además de los registros `Subscriber` (suscriptor) y `Platform` (plataforma), hay dos registros adicionales `BinSubscriber` y `BinPlatform` para poder almacenar información y trabajar con ficheros binarios. Esto último es necesario porque los dos primeros registros contienen campos de tipo `string` y `vector` que, como sabes, no pueden utilizarse para leer o escribir información de ficheros binarios al no tener un tamaño fijo (si no sabes de qué estoy hablando, consulta la transparencia 33 del *Tema 3* de teoría).

Los registros `BinSubscriber` y `BinPlatform` sólo se utilizarán cuando haya que leer o escribir en un fichero binario. Para el resto de funcionalidades de la práctica se usarán los registros `Subscriber` y `Platform`.

3.1. Subscriber

La estructura `Subscriber` almacena información de un suscriptor dentro de la plataforma. Cada suscriptor contendrá un identificador único (`id`), nombre (`name`), correo electrónico (`email`), su dirección IP principal (`mainIp`) y la lista de direcciones IP desde la que se conecta cada uno de sus dispositivos (`ips`).

```
struct Subscriber {
  unsigned int id;
  string name;
  string email;
  string mainIp;
  vector<string> ips;
};
```

3.2. Platform

Este registro tiene un nombre (`name`) y un vector (`subscribers`) que almacenará todos los suscriptores que se vayan añadiendo a la plataforma. También contiene un campo entero `nextId` que almacenará el identificador que deberá asignarse al campo `id` del siguiente suscriptor que se cree. Este campo tendrá el valor `1` al inicio del programa y se incrementará de uno en uno para cada nuevo suscriptor. Es decir, el primer suscriptor que se cree tendrá como identificador el `1`, el segundo tendrá el `2` y así sucesivamente.

```
struct Platform {
  string name;
  vector<Subscriber> subscribers;
  unsigned int nextId;
};
```

En el programa principal (`main`) deberás declarar una variable de tipo `Platform` y asignar a su campo `name` el valor `Streamflix` y a su campo `nextId` el valor `1`. El vector `subscribers` estará inicialmente vacío e irá creciendo conforme se añadan nuevos suscriptores al programa.

- Tu programa sólo gestionará un único Platform, que contendrá todos los suscriptores de la empresa

3.3. BinSubscriber

Este tipo de registro se utilizará para leer y escribir información de un suscriptor en el fichero binario:

```
struct BinSubscriber {
  unsigned int id;
  char name[KMAXSTRING];
  char email[KMAXSTRING];
  char mainIp[KMAXIP];
};
```

En este registro se han introducido dos cambios con respecto al registro `Subscriber`:

- Los `string` se sustituyen por cadenas de caracteres (p.ej. el campo `char name[KMAXSTRING]`), donde `KMAXSTRING` y `KMAXIP` son constantes de tipo entero cuyo valor es `50` y `16` respectivamente
- Ha desaparecido la lista de direcciones IP (más detalles en la sección 5.5.3)

3.4. BinPlatform

Este tipo de registro se utilizará para leer y escribir información de la plataforma en el fichero binario:

```
struct BinPlatform {
  char name[KMAXSTRING];
  unsigned int nextId;
};
```

Igual que sucedía con `BinSubscriber`, el campo `string name` se sustituye por el campo `char name[KMAXSTRING]`. También ha desaparecido el vector de suscriptores.

- Recuerda que puedes encontrar más detalles sobre los motivos de estos cambios en la transparencia 33 del *Tema 3* de teoría

4. Menú

Al ejecutar la práctica se mostrará de inicio por pantalla el menú principal del programa, quedando a la espera de que el usuario elija una opción:

Terminal

```
[Options]
1- Show subscribers
2- Add subscriber
3- Add subscriber IP
4- Delete subscriber
5- Import/export
q- Quit
Option:
```

Este menú será el que aparezca por pantalla cuando el usuario inicie el programa y por lo tanto deberá de gestionarse en la función `main`.

Las opciones válidas que puede introducir el usuario son los números del `1` al `5` y la letra `q`. Si la opción elegida no es ninguna de éstas (por ejemplo un `7`, una `x` o un `;`) se emitirá el error `ERR_OPTION` llamando a la función `error` con dicho parámetro. Cuando el usuario elige una opción correcta, se debe ejecutar el código asociado a dicha opción. Al finalizarla, volverá a mostrar el menú principal y a pedir otra opción, hasta que el usuario decida salir del programa utilizando la opción `q`.

5. Opciones

En los siguientes apartados se describe el funcionamiento que deberá tener cada una de las opciones que se muestran en el menú principal del programa.

5.1. Show subscribers

Esta funcionalidad se activa con la opción `1` del menú y mostrará un listado con la información de los suscriptores. Su código se deberá incluir en la función `showSubscribers()` de `prac2.cc`. Cada suscriptor se mostrará en una línea con el siguiente formato:

Terminal

```
id:name:email:mainIp:ips
```

El último campo (`ips`) deberá mostrarse como una lista de direcciones IP separadas por el carácter '|'. El siguiente ejemplo muestra una salida válida con dos suscriptores, con identificadores 1 y 2 respectivamente. El orden en que se muestran es el mismo en el que se crearon.

Terminal

```
1:Teddy Sarao:teddy@streamflix.com:111.111.111.111:111.111.111.111|222.222.
222.222|111.111.111.111
2:Fez Berzos:fez@shopping.com::
```

En este ejemplo, el suscriptor con identificador 1 tiene una lista de 3 direcciones IP, que corresponden a cada uno de los dispositivos desde los que se conecta. De estas 3 direcciones se ha determinado que su dirección principal es la 111.111.111.111 (ver sección 5.3). El suscriptor 2, por el contrario, todavía no tiene ninguna dirección IP asociada, por lo que el campo `mainIp` también está vacío.

- Ten en cuenta que en el ejemplo los datos del suscriptor 1 se han mostrado en dos líneas para que quepan en la página, pero en tu programa se deberán mostrar en una única línea
- La lista de direcciones IP puede tener direcciones repetidas, ya que cada una corresponde a un dispositivo distinto y puede haber varios dispositivos en la misma ubicación (todos los dispositivos de tu casa comparten la misma IP)

5.2. Add subscriber

Esta opción permite añadir un suscriptor nuevo al programa. Se activa cuando el usuario elige la opción `2` del menú principal. Su código se deberá incluir en la función `addSubscriber()` de `prac2.cc`. En ella se pedirá al usuario que introduzca el nombre del suscriptor mostrando el siguiente mensaje:

Terminal
```
Enter name:
```

El valor introducido por el usuario deberá revisarse para comprobar que sea un nombre válido. Se considerarán válidos los nombres que tengan una longitud mínima de 3 caracteres y que no contengan ningún carácter ':' (dos puntos). No es necesario realizar ninguna otra comprobación.

Si se introduce un nombre inválido se emitirá el error `ERR_NAME` y se volverá a pedir al usuario que introduzca el nombre mostrando el mismo mensaje.

Si el nombre es correcto se asignará al suscriptor y se pedirá a continuación al usuario que introduzca el email mostrando el siguiente mensaje:

Terminal
```
Enter email:
```

El campo `email` deberá validarse para comprobar que tenga un formato correcto. Las reglas para validar un email son las siguientes:

- Deberá contener un carácter '@', y sólo podrá aparecer una vez
- Antes y después del carácter '@' deberá haber dos secuencias de caracteres válidas (parte izquierda y parte derecha)
- Ninguna de las dos partes izquierda o derecha podrá ser una cadena vacía
- Las dos partes izquierda y derecha sólo podrán contener los caracteres de la 'a' a la 'z' (en minúsculas y/o mayúsculas), los dígitos del '0' al '9', el punto '.' y la barra baja '_'
- Las dos partes izquierda y derecha no podrán comenzar ni terminar por el carácter punto '.'
- La parte derecha debe contener al menos un punto '.'

- En la corrección no se introducirán caracteres acentuados, ni cualquier otro que no pertenezca al alfabeto inglés

Si se introduce una cadena inválida o vacía se emitirá el error `ERR_EMAIL` y se volverá a pedir al usuario que se introduzca el email.

Una vez leídos el nombre y el email correctos se creará el suscriptor con los campos `mainIp` y `ips` vacíos, ya que estos se rellenarán con la siguiente opción del menú. Se asignará de manera automática al nuevo suscriptor un identificador único que se almacenará en el campo `id` de `Subscriber`. Este identificador vendrá dado por el valor que haya en ese momento almacenado en el campo `nextId` de la estructura `Platform` que contiene toda la información del programa. Como se indicó en la Sección 3.2, este identificador único comenzará con el valor `1` y se incrementará de uno en uno para cada nuevo suscriptor. Por lo tanto, cada vez que añadas un suscriptor, deberás de incrementar el campo `nextId` para que al siguiente suscriptor se le asigne un nuevo identificador correcto. Tras asignar el identificador, se incorporará el nuevo suscriptor al vector `subscribers` del registro `Platform` del programa y se volverá al menú principal.

- Si se borra un suscriptor no se reutilizará su identificador, ni aunque sea el último suscriptor creado. Es decir, si tienes tres suscriptores, con identificadores `1`, `2` y `3`, y se borra el `3`, al crear un nuevo suscriptor se le asignará el identificador `4` aunque ya no exista un suscriptor con el identificador `3`

5.3. Add subscriber IP

Esta opción permite añadir una dirección IP de un dispositivo que pertenezca a un suscriptor ya existente. Se activa cuando el usuario elige la opción 3 del menú principal. Su código se deberá incluir en la función `addSubscriberIp()` de `prac2.cc`. En ella se pedirá el identificador del suscriptor con el mensaje:

```
Terminal
Enter subscriber id:
```

Si no existe un suscriptor con ese identificador, o se introduce la cadena vacía, se emitirá el error `ERR_ID` y se volverá al menú principal. Si el identificador es correcto, se pedirá la dirección IP a añadir con el mensaje:

```
Terminal
Enter IP:
```

El valor introducido por el usuario deberá revisarse para comprobar que sea válido. Una dirección IP válida debe tener una secuencia de 4 números separados por el carácter punto '.', y los números deben estar entre los valores 0 y 255 (ambos incluidos). Si se introduce como dirección una cadena vacía o inválida se emitirá el error `ERR_IP` y se volverá a pedir al usuario que introduzca la dirección mostrando el mismo mensaje.

Si la dirección IP es correcta se añadirá al vector `ips`. El siguiente paso será determinar qué dirección IP es la principal para el suscriptor, eligiendo la que aparezca un mayor número de veces en el vector. Si hay varias direcciones que aparecen el mismo número de veces se elegirá la primera que se haya encontrado en el vector contando desde la primera posición. Una vez se haya elegido la dirección principal deberá almacenarse en el campo `mainIp` y se volverá al menú principal.

- Los números de las direcciones IP no tendrán ceros a la izquierda (salvo en el caso particular de que el número sea el 0), y podrán tener menos de 3 cifras. Por ejemplo, podría darse como entrada la dirección correcta 84.0.127.255, mientras que direcciones como 88.216.00001.255 no se usarán para probar el programa. También deberías comprobar que no se introduzcan direcciones inválidas como 57.1024.33.200 (es inválida porque tiene un número mayor que 255)
- La dirección principal deberá recalcularse cada vez que se añada una nueva ip al suscriptor

5.4. Delete subscriber

Esta opción permite borrar un suscriptor existente en la plataforma. Se activa cuando el usuario elige la opción 4 del menú principal. Su código se deberá incluir en la función `deleteSubscriber()` de `prac2.cc`. En ella se pedirá el identificador del suscriptor con el mensaje:

```
Terminal
Enter subscriber id:
```

Si no existe un suscriptor con ese identificador, o se introduce la cadena vacía, se emitirá el error `ERR_ID` y se volverá al menú principal. Si el identificador es correcto, se eliminará el suscriptor del vector `subscribers` del registro `Platform` que almacena todos los suscriptores del programa y se volverá al menú principal.

5.5. Import/export

Esta opción permite leer la información de los suscriptores desde ficheros en distintos formatos, así como almacenarla en estos mismos formatos. Su código se deberá incluir en la función `importExportMenu()` de `prac2.cc`. Esta función se encargará de mostrar el menú de manejo de ficheros y de llamar a las distintas funciones para cada una de las opciones, de manera análoga a como se hace en el `main` con el menú principal. Esta función se activa cuando el usuario elige la opción `5` del menú principal, tras lo cual se deberá mostrar el menú de manejo de ficheros:

Terminal

```
[Import/export options]
1- Import from CSV
2- Export to CSV
3- Load data
4- Save data
b- Back to main menu
Option:
```

Las opciones válidas que puede introducir el usuario son los números del `1` al `4` y la letra `b`. Si la opción elegida no es ninguna de éstas (por ejemplo un `7`, una `x` o un `;`) se emitirá el error `ERR_OPTION` llamando a la función `error` y se volverá a mostrar este mismo menú. Cuando el usuario elige una opción correcta, se debe ejecutar el código asociado a dicha opción. Al finalizarla, volverá a mostrar este menú y a pedir otra opción, hasta que el usuario decida volver al menú principal utilizando la opción `b`.

5.5.1. Import from CSV

Esta opción permite importar información de suscriptores almacenados en un fichero de texto en formato CSV (*Comma Separated Values*, o valores separados por comas). Estos ficheros contienen distintos campos separados por un carácter. Algunos caracteres comunes empleados como separadores son la coma ‘,’, punto y coma ‘;’ o dos puntos ‘:’. En esta práctica usaremos el separador dos puntos ‘:’.

Esta opción se activa cuando el usuario elige la opción `1` del menú de manejo de ficheros. Su código se deberá incluir en la función `importFromCsv()` de `prac2.cc`. En primer lugar se pedirá al usuario el nombre del fichero donde está almacenada la información mostrando el mensaje:

Terminal

```
Enter filename:
```

El nombre del fichero podrá contener espacios en blanco, pero no será necesario realizar ninguna otra comprobación. A continuación se abrirá el fichero de texto y se leerá, añadiendo al programa los suscriptores almacenados en el fichero. Si el nombre de fichero está vacío o no se puede abrir el fichero, se emitirá el error `ERR_FILE` y se volverá al menú de manejo de ficheros. El formato del fichero será como el descrito en la Sección 5.1, pero sin incluir el identificador en el primer campo de cada línea:

data.csv

```
Teddy Sarao:teddy@streamflix.com:111.111.111.111:111.111.111.111|222.222.
222.222|111.111.111.111
Fez Berzos:fez@shopping.com::
```

Al importar un fichero no se perderán los datos que ya hay en el registro `Platform`, sino que los suscriptores que se lean se irán añadiendo al final del vector `subscribers` en el mismo orden en el que aparezcan en el fichero, asignándoles un identificador único a cada uno (como si se hubieran creado mediante la opción `Add subscriber`).

Al añadir los suscriptores deberán realizarse las validaciones de los campos `name` y `email`, tal como se describe en la sección 5.2. También deberá comprobarse que la dirección `mainIp` y las direcciones de

la lista `ips` sean válidas según se describe en la sección 5.3, aunque en este caso estos dos campos sí podrían estar vacíos. Si algún campo tiene un valor inválido el suscriptor no se añadirá a la plataforma, se mostrará el mensaje de error correspondiente y se pasará al siguiente suscriptor del fichero sin realizar las comprobaciones de los campos restantes. Sin embargo, no se deberá comprobar si la dirección `mainIp` corresponde a la más frecuente en la lista `ips`, ni se recalculará a partir de esta lista.

- El formato de las líneas del fichero será siempre correcto, pero los valores de los campos podrán ser inválidos según las reglas descritas en la Sección 5.2
- Los campos `name` y `email` no contendrán nunca dos puntos (`:`), así que puedes asumir que siempre que aparezca este carácter en el fichero se trata de un separador
- El fichero de entrada puede estar vacío. En ese caso, no se importará nada, pero tampoco se mostrará ningún tipo de error
- Recuerda que los ficheros CSV son en realidad ficheros de texto, puedes crear tus propios ficheros de ejemplo con cualquier editor de textos

5.5.2. Export to CSV

Esta funcionalidad, que se activa con la opción 2 del menú de manejo de ficheros, guardará en un fichero de texto la información sobre los suscriptores. Su código se deberá incluir en la función `exportToCsv()` de `prac2.cc`. El formato en el que se guardará la información será el mismo que se ha descrito en la sección anterior para la importación, guardando cada suscriptor en una línea distinta, sin incluir sus identificadores.

En primer lugar se pedirá al usuario el nombre del fichero donde desea almacenar la información, mostrando el siguiente mensaje:

Terminal

```
Enter filename:
```

Se abrirá el fichero de texto para escritura y se guardará en él la información correspondiente, volviendo a continuación al menú de manejo de ficheros. Si el nombre de fichero está vacío o no se puede abrir el fichero se emitirá el error `ERR_FILE` y se volverá al menú de manejo de ficheros.

- Si el fichero de salida ya existe, se sobreescribirá su contenido borrando lo que hubiera antes almacenado en él

5.5.3. Load data

Esta opción permite cargar en el programa todos los datos de la plataforma a partir de una copia de seguridad en un fichero binario. Su código se deberá incluir en la función `loadData()` de `prac2.cc`. Los datos que hubiera en ese momento en la estructura `Platform` del programa se borrarán completamente y se sustituirán por la información leída del fichero. Esta funcionalidad se activa con la opción 3 del menú de manejo de ficheros.

En primer lugar se preguntará al usuario si desea continuar, advirtiéndole de que todos los datos del programa se borrarán:

Terminal

```
All data will be erased. Continue? [y/n]:
```

Si el usuario introduce `Y` o `y`, se procederá a pedir el nombre del fichero a cargar. Si el usuario introduce `N` o `n`, se volverá al menú de manejo de ficheros sin llevar a cabo ninguna acción. En caso de que el usuario introduzca cualquier otro carácter, se volverá a mostrar el mismo mensaje solicitando la confirmación.

Al continuar se pedirá al usuario el nombre del fichero binario donde está almacenada la información mostrando el mensaje:

```
Terminal
Enter filename:
```

Si el nombre de fichero está vacío o el fichero no se pudiera abrir se emitirá el error `ERR_FILE` y se volverá al menú sin borrar los datos de la plataforma. Si el fichero se abre correctamente se procederá con el borrado de los datos actuales de la plataforma y con la lectura de los nuevos datos desde el fichero.

La información almacenada en el fichero binario tendrá la siguiente estructura:

- En primer lugar aparecerá un registro de tipo `BinPlatform` con el nombre (`name`) seguido del siguiente identificador (`nextId`)
- A continuación aparecerán cero o más registros de tipo `BinSubscriber`, hasta el final del fichero. Cada registro contendrá el identificador (`id`), el nombre (`name`), el correo electrónico (`email`) y la dirección IP principal (`mainIp`)

Para cada suscriptor leído del fichero, si el valor de la dirección principal (`mainIp`) no está vacío, deberá añadirse esta dirección como único elemento de su vector de direcciones (`ips`).

La siguiente imagen muestra un ejemplo de cómo podría estar estructurado un fichero binario (es sólo un dibujo para aclarar la estructura; recuerda que el fichero en realidad sólo contiene ceros y unos):

BinPlatform	BinSubscriber	BinSubscriber
name	id	id
Streamflix	1	2
nextId	name	name
3	Teddy Sarao	Fez Berzos
	email	email
	teddy@streamflix.com	fez@shopping.com
	mainIp	mainIp
	111.111.111.111	

En primer lugar aparecería en el fichero un registro de tipo `BinPlatform`. A continuación aparece un registro `BinSubscriber` representando el primer suscriptor, cuyo `id` es 1 y su nombre (`name`) es `Teddy Sarao` (además del resto de campos). A continuación aparece un nuevo registro de tipo `BinSubscriber` que representa el segundo suscriptor almacenado en el fichero (con `id` 2 y nombre `Fez Berzos`). Con este último registro se completa toda la información almacenada en el fichero.

- Se asume que todos los datos almacenados en el fichero binario son correctos, por lo que no es necesario hacer ninguna comprobación adicional (p.ej. no habrá suscriptores con el nombre ni el email incorrectos)

5.5.4. Save data

Esta opción permite hacer una copia de seguridad de toda la plataforma en un fichero binario. Se activa con la opción `4` del menú de manejo de ficheros. Su código se deberá incluir en la función `saveData()` de `prac2.cc`. En primer lugar, se pedirá al usuario el nombre del fichero binario donde se almacenará la información:

Terminal

```
Enter filename:
```

Si el nombre de fichero está vacío o el fichero no se puede abrir, se emitirá el error `ERR_FILE` y se volverá al menú de manejo de ficheros. Si el fichero ya existe, se sobrescribirá todo su contenido borrando lo que hubiera antes almacenado en él. El formato de salida será exactamente el mismo que se ha descrito en el apartado anterior para la opción `Load data`, por lo que se almacenará toda la información de los registros menos las listas de direcciones IP.

- Ten en cuenta que puede que tengas que recortar los campos de texto (`name` de `Platform`, y `name` y `email` de `Subscriber`) al guardarlos en el fichero binario. Por ejemplo, podrías tener un registro `Subscriber` con una cadena de 60 caracteres almacenada en el campo `name`. Sin embargo, en el fichero binario tienes que guardar una estructura de tipo `BinSubscriber` cuyo campo `name` tiene un tamaño máximo de `50`. En este caso, tendrías que recortar el nombre original y almacenar sólo los primeros 49 caracteres en fichero (el máximo admitido, ya que tienes que dejar un espacio para incluir el `'\0'` final). Deberás asumir que, al recortar estos campos, sus valores seguirán siendo válidos

6. Argumentos del programa

En esta práctica el programa debe permitir al usuario indicar algunas acciones a realizar mediante el paso de argumentos por línea de comando. Deberás utilizar los parámetros `int argc` y `char *argv[]` de la función `main` para poder gestionarlos. El programa admitirá los siguientes argumentos por línea de comando:

- `-i <fichero texto>`. Permite importar suscriptores de un fichero de texto, cuyo nombre deberá indicarse tras el argumento `-i`, comportándose igual que si el usuario seleccionara la opción `Import from CSV` del menú de manejo de ficheros, también a la hora de tratar los errores si alguno de los datos leídos es incorrecto. Por ejemplo, si se lee del fichero un suscriptor cuyo nombre contiene caracteres no válidos, se emitirá el error `ERR_NAME`, se ignorarán los datos de ese suscriptor y se seguirá procesando el resto del fichero. Si no se puede abrir el fichero se emitirá el error `ERR_FILE` y se finalizará el programa sin llegar a mostrar el menú principal. El siguiente ejemplo importará los datos almacenados en el fichero de texto `data.csv`:

  Terminal

  ```
  $ prac2 -i data.csv
  ```

- `-l <fichero binario>`. Permite cargar información de la plataforma desde un fichero binario cuyo nombre deberá indicarse después del argumento `-l`, comportándose igual que si el usuario seleccionara la opción `Load data` del menú de manejo de ficheros. En este caso no deberá pedirse confirmación al usuario para cargar los datos desde el fichero. Si no se puede abrir el fichero, se emitirá el error `ERR_FILE` y se finalizará el programa sin llegar a mostrar el menú principal. El siguiente ejemplo cargará los datos almacenados en el fichero binario `data.bin`:

  Terminal

  ```
  $ prac2 -l data.bin
  ```

- Los dos argumentos son opcionales y pueden no aparecer en la llamada al programa
- Un determinado argumento sólo puede aparecer una vez en la llamada, de lo contrario se considerará que los argumentos son erróneos
- Los ficheros binarios no tienen por qué tener necesariamente la extensión `.bin` y los de texto no tienen por qué tener la extensión `.csv`. Utilizamos estas extensiones en los ejemplos para que quede más claro con qué tipo de fichero estamos trabajando

Los dos argumentos pueden aparecer al mismo tiempo en una llamada al programa y además en cualquier orden. Independientemente del orden en el que aparezcan en la llamada, el orden en el que se procesarán los argumentos será el siguiente:

1. En primer lugar se procesará la opción `-l` para cargar datos de fichero binario
2. En segundo lugar se ejecutará la opción `-i` para importar datos desde un fichero de texto

En el siguiente ejemplo, en primer lugar se cargarán los datos del programa almacenados en el fichero `data.bin` y a continuación se incorporarán los suscriptores almacenados en el fichero `data.csv`:

Terminal
```
$ prac2 -i data.csv -l data.bin
```

Otro ejemplo, cuyos parámetros son válidos, sería el siguiente:

Terminal
```
$ prac2 -l -l -i -i
```

Aunque a primera vista los parámetros pudieran parecer incorrectos, en realidad no lo son. El programa recibe un primer parámetro `-l` que va seguido de un fichero que se llama `-l`. A continuación recibe un parámetro `-i` que va seguido de un fichero que se llama `-i`. Esta secuencia de parámetros es, por tanto, correcta.

En caso de que se produzca un error en los argumentos de entrada (por ejemplo, que se introduzca una opción que no existe), se deberá emitir el error `ERR_ARGS` y se finalizará el programa sin llegar a mostrar el menú principal. Si todos los argumentos son correctos, se procesarán las opciones introducidas por el usuario y después se mostrará el menú principal de programa de la forma habitual. Los argumentos pueden ser erróneos si se da alguna de las siguientes circunstancias:

- Alguna de las opciones aparece repetida. Por ejemplo:

  Terminal
  ```
  $ prac2 -l data.bin -l moredata.bin
  ```

- A las opciones `-l` o `-i` les falta el argumento de detrás con el nombre del fichero. Por ejemplo:

  Terminal
  ```
  $ prac2 -l data.bin -i
  ```

- Aparece un argumento que no es ninguno de los permitidos. Por ejemplo:

  Terminal
  ```
  $ prac2 -n data.csv
  ```