UNIVERSIDAD DE GRANADA

Práctica LEX

Doble Grado en Ingeniería Informática y Matemáticas

Autor:

Noura Lachhab Bouhmadi

Quintín Mesa Romero

1. Introducción

Cuando subimos a un autobús de línea en Granada, tenemos tres opciones: o ticamos con un bonobus de determinada clase, pagamos directamente con monedas el precio de un billete ordinario, o bien le pedimos al conductor que nos proporcione una tarjeta nueva, la cual podemos recargar con 5, 10 o 20 euros.
El objetivo de este trabajo se centra en la elaboración de un escáner de bonobuses, el cual, dado un bonobus cualquiera (a continuación procederemos a la explicación de la estructura de un bonobus), identifica la clase de bonobus del que se trata, el identificador del bonobus, el saldo en euros que tiene en el instante en que se escanea, el precio al que se le está cobrando el billete (que dependerá de la clase de bonobus), el nombre del usuario, si procede, la linea de autobus en la que se viaja y la fecha de la última utilización. De esta forma, en función de la clase a la que pertenezca se le cobrará un determinado precio por el billete, que se restará al saldo que tenga la tarjeta y, si se determina que no tiene dinero suficiente en la misma, se le solicitará que se recargue. Todas las modificaciones que se realicen sobre el bonobus se actualizarán al ticar o si se solicita un credibus nuevo al conductor, se creará un bonobus con las características propias de una tarjeta ordinaria, con el saldo que se le especifique.

Para nosotros, un bonobus va a ser un fichero de texto con la siguiente estructura:

Bonobus

ID: numero de 8 dígitos
SALDO: número decimal con 3 cifras decimales
PRECIO: número decimal con 2 cifras decimales
CLASE: ∈{Joven,Universitario,Ordinario,PersonasConDiscapacidad,Credibus,Pensionista}
NOMBRE: si procede: Joven, Universitario, PersonasConDiscapacidad, Pensionista
LINEA: Mayúscula+Digito o Dígito o Tres dígitos
FECHA: Dia Mes nºdia hora:min:seg año (en este formato)

2. Desarrollo del programa

Para desarrollar este "escáner" de bonobuses se ha elaborado un programa en C++ cuya funcionalidad está descrita según la estructura propia de un fichero Flex:

```
Declaraciones
%%
Reglas
%%
Procedimientos de Usuario
```

En la sección de declaraciones se han distinguido dos partes:

- Un primer bloque en el que se incluyen las librerías necesarias para implementar correctamente el programa, y donde se declaran las variables globales necesarias y las cabeceras de las funciones que nos hagan falta (ver código del programa).
- Un segundo bloque en el que se da nombre a las expresiones regulares que hemos utilizado:

```
DIGITO          [0-9]
MAY             [A-Z]
MIN             [a-z]
DECIMAL         {DIGITO}+"."{DIGITO}{DIGITO}
SALDODECIMAL    {DIGITO}+"."{DIGITO}{DIGITO}{DIGITO}
ESPACIO         [[:blank:]]
ALPHA           [A-Za-z]
NAME            ({ALPHA}*{ESPACIO}{ALPHA}*)
FECHA           ({ALPHA}*{ESPACIO}{ALPHA}*{ESPACIO}{DIGITO}*{ESPACIO}{DIGITO}*"\:"{DIGITO}*"\:"{DIGITO}*{ESPACIO}{DIGITO}*)
LINE            ({MAY}{DIGITO}*|{DIGITO}|{DIGITO}{DIGITO}{DIGITO})
IDI             ({DIGITO}{DIGITO}{DIGITO}{DIGITO}{DIGITO}{DIGITO}{DIGITO}{DIGITO})
end             [ \n\t]
```

Nota

Se han definido dos expresiones regulares que representan números decimales: DECIMAL y SALDODECIMAL porque había conflicto, de ahí el tener que considerar el SALDO como un número decimal con tres decimales y el PRECIO del billete como uno con dos cifras decimales.

A continuación tenemos la sección de reglas, en la que hemos definido una serie de expresiones regulares, usando las del bloque de declaraciones, asociadas a unas reglas escritas en C++ y encerradas entre llaves. Estas reglas son la base del escáner, pues son las que permiten seleccionar del fichero de texto que representa el bonobus, la información que se quiere almacenar y manipular.

```
{IDI}                                                       {ID= yytext;}
{SALDODECIMAL}                                              {SALDO_A = stof(yytext);}
{DECIMAL}                                                   {PRECIO = stof(yytext);}
(Joven|Universitario|Ordinario|PersonasConDiscapacidad|Credibus)    {CLASE = yytext;}
{NAME}                                                      {NOMBRE= yytext;}
{LINE}                                                      {LINEA = yytext;}
{FECHA}                                                     {DATE= yytext;}
```

Las reglas que se han definido están dirigidas simplemente al reconocimiento y almacenamiento de los datos de los que consta un bonobus. De esta forma, se han definido expresiones regulares que representan a cada uno de los elementos que forman parte de la estructura del mismo. Para captar el texto de un token reconocido en el fichero, se utiliza la variable de tipo char *, **yytext**, cuyo valor se almacena en la variable global correspondiente.

Por último, llegamos a la sección de procedimientos de usuario, donde nos hemos dejado llevar programando con libertad todas las funciones que en su momento declaramos en la sección de declaraciones.

El programa consta de una función main en la que se presenta un MENÚ con dos opciones: Ticar (1) o Pedir Credibus al conductor (2).

- Si se elige **Ticar**, LEX escanea el bonobus, almacenando en las variables globales los tokens que vaya reconociendo y a continuación, se llama a la función **ticar()**, que simula la acción de ticar con el bonobus. Esta función muestra por pantalla los datos del bonobus utilizado en la linea de autobús LINEA: FECHA, ID, CLASE, PRECIO del billete, NOMBRE, y SALDO actual (saldo del que dispone la tarjeta al cobrarse el precio del billete). Como el precio del billete depende de la clase a la que pertenezca el bonobus, se ha implementado el cobro del billete mediante una estructura condicional anidada sencilla.
  En caso de que el SALDO actual no llegue a cubrir el PRECIO de un billete de su CLASE, se le solicita al cliente que recargue su bonobus con 5, 10 o 20 euro. Para ello, se han implementado dos funciones: **solicitud_recarga**, que solicita al cliente que recargue su bonobus, y que introduzca la cantidad deseada, y **recargar**, que lleva a cabo el proceso de recarga del bonobus y obliga al cliente a volver a ticar.

- Si se elige **Pedir Credibus al conductor**, con la función **crear_bonobus** se crea un nuevo bonobus con ID y LINEA aleatorios, FECHA la actual del sistema y CLASE "Credibus". Una vez creado el bonobus se solicita al cliente que recargue con una cantidad permitida y se le obliga a ticar.

Nota

Todos los datos del bonobus que sufren modificaciones durante la ejecución del programa, son actualizados en el propio fichero txt que representa al bonobus. Para ello, se ha creado una función, **actualizar_bonobus** que recibe como argumento un puntero FILE (el que se genera al abrir el fichero y que apunta al principio del mismo) y se sobrescriben todos los datos modificados.

3. Proceso de Ejecución

El proceso de compilación y ejecución del escáner de bonobuses creado es tal y como se indica en los apuntes:

- Invocamos a flex con

  flex++ escaner_bonobus.l

- Compilamos con g++ el archivo generado con la orden anteior: lex.yy.cc

  g++ lex.yy.cc -o prog

- Ejecutamos el ejecutable prog sobre un bonobus cualquiera:

  ./prog bonobus_i.txt

Para probar el funcionamiento del programa, se proporciona un directorio con 100 bonobuses, en ficheros con nombre bonobus_i.txt con $i \in \{1, ..., 100\}$, cada uno generado aleatoriamente mediante un programa que funciona de la misma forma que la función crear_bonobus, pero tomando los datos de forma aleatoria. El campo nombre y apellidos de los bonobuses que lo requieran se han tomado de forma aleatoria de un fichero txt con 100 nombres y apellidos todos distintos. Para generar los 100 ficheros se ha creado un script de bash (todos los programas usados se proporcionan en la carpeta).

Recomendación

Se recomienda probar la ejecución del programa con el editor de códigos **Visual Studio Code**, que permite tener el programa en una ventana, el bonobus en cuestión en otra, y la terminal en la parte baja, siendo así más fácil poder visualizar los cambios que se van produciendo en los bonobuses conforme se va ejecutando el programa.