PRACTICA – 1

SISTEMES OPERATIUS II

Marcos Buenacasa i Alberto Leiva

Octubre 2016

INTRODUCCIÓN

En esta primera práctica de Sistemas operativos nos encargamos básicamente de leer un fichero descargado de internet e implementar un parser para que el resultante sea un árbol binario en el cual dentro contiene una lista enlazada de elementos. En este documento se intenta explicar de forma breve todo aquello que no tiene que ver con el código en sí de la práctica tales como la decisión a la hora de hacer cada uno de los apartados de la práctica. La práctica trata de leer la información de vuelos.

REALIZACIÓN DE LA PRÁCTICA

La realización de la práctica viene indicada por el guión de esta el cual está bastante guiada. Por ello en este documento seguiremos las pautas a orden de índice.

1 Lectura de la información de los vuelos.

El enunciado básicamente nos pide implementar un *parser*, dado que el fichero *raw* que se nos ofrece es uno el cual toda la información esta separada por comas. Así que nosotros hemos implementado un parser el cual recorre todas las lineas del fichero objetivo recordando en todo momento el numero de comas recorridas y una vez llegado a la coma deseada, es decir, aquella que precede a la información que nos interesa posteriormente la extraeremos. La mayoría del código nos viene dado en el esqueleto de la práctica, así que la implementación de este apartado de la práctica no supone ningún problema.

2 Inserción de la información de los vuelos en un árbol binario.

Ahora se nos pide que la información que hemos obtenido con el *parser* del apartado anterior la insertemos en un árbol binario balanceado. El esqueleto de la práctica nos da ya la estructura del árbol binario así que debemos modificar la implementación dada por una nuestra que haga bien las funciones, además de cambiar el tipo de dato que debemos guardar, hay que corregir la forma en la que el árbol se administra, es decir, cómo el árbol compara los nodos para discernir si un nodo ya se encuentra en éste o no. Una primera implementación de este código nos produjo resultados érroneos, ya que comparábamos los apuntadores y esto solo comparaba el primer carácter. La correcta solución era usar el comando de C *strcomp*. Después de este error, insertar los la información de cada uno de los aeropuertos de salida como nodo no tuvo ninguna complicación adicional.

Como dato adicional, el problema que encontraba *valgrind* a la hora de ejecutarse erradicaba en que la función *deletetree()* eliminaba todo el árbol a excepción de la raíz, así que la solución era liberar esta raíz del árbol de la memoria y *valgrind* no mostró mas el error.

3 Inserción de la información de destino a la lista enlazada.

Una vez tenemos el árbol balanceado creado con la información de los vuelos de origen se nos pide almacenar dentro de cada uno de estos nodos una lista enlazada con cada uno de los aeropuertos de destino de estos vuelo y a la vez dentro de cada uno de los elementos de la lista enlazada almacenar el día de la semana en que ocurrió el vuelo y el retardo que tuvo el vuelo.

Para la realización de esta parte de la práctica procedimos de forma similar a la que lo hicimos en el apartado de insertar los datos del vuelo donde. Primero modificamos el código esqueleto ya

implementado, para que almacene la lista enlazada. Aquí tambien tuvimos el mismo problema explicado referente a la comparacion sobre si un elemento de la lista ya estaba implementado, el uso de la función *strcomp()*, fue la clave para que funcionara. Modificado el código de la lista enlazada sólo nos queda obtener del *parser* implementado anteriormente los datos necesarios, formatearlos e añadirlos a la lista enlazada.
Para la parte del día de la semana y el tiempo de retardo, lo que realizamos fue crear dentro de cada elemento de la lista enlazada una lista de 14 elementos, dos veces 7, referente a cada uno de los días de la semana las primeras 7 para acumular el restraso y las otras 7 para tener un contador de vuelos, así si en un futuro se nos pide hacer estadísticas con los datos, es fácil obtener tanto el día como el retraso de forma trivial.

ERRORRES Y PROBLEMAS SURGIDOS EN LA REALIZACIÓN DE LA PRÁCTICA

El problema más grave ya ha sido comentado durante la redacción del transcurso de la realización de la práctica, el referente a la comparación de llaves de los nodos del árbol/lista enlazada. Además de este, otro de los problemas que nos surgió fue a la hora de obtener los datos de fichero objetivo, este consistía en que intentando reutilizar al máximo las variables con tal de obtener una eficiencia y el menor consumo de memoria nos pisábamos variables necesarias para el correcto funcionamiento. Otro error que tuvimos fué a la hora de crear el make, no usamos el que se nos daba en el enunciado y dado esto no podíamos debugar normalmente la práctica, tuvimos que bucear en internet para encontrar como realizar un make con los marcadores necesarios para hacer el debug.

Algún error más hemos tenido, pero sin la importancia necesaria como para estar incluido en la memoria.

CONCLUSIONES

Tras la realización de esta primera práctica tenemos como resultado un árbol binario balanceado, cuyos nodos son los códigos de todos los aeropuertos americanos en los que en el año 2008 despegó un avión. Dentro de estos nodos tenemos una lista enlazada con todos los aeropuertos destino de los que despegó un avión y para cada uno de los aeropuertos de destino una lista que almacena los datos del día de la semana y el retraso acumulativo referente a cada día. Dato a añadir, hemos acabado utilizando el debugger que viene instalado de forma nativa en el ide propietario de JetBrains, Clion debido a la tosquedad que nos ha parecido usar los que se nos ofrecían en las practicas.

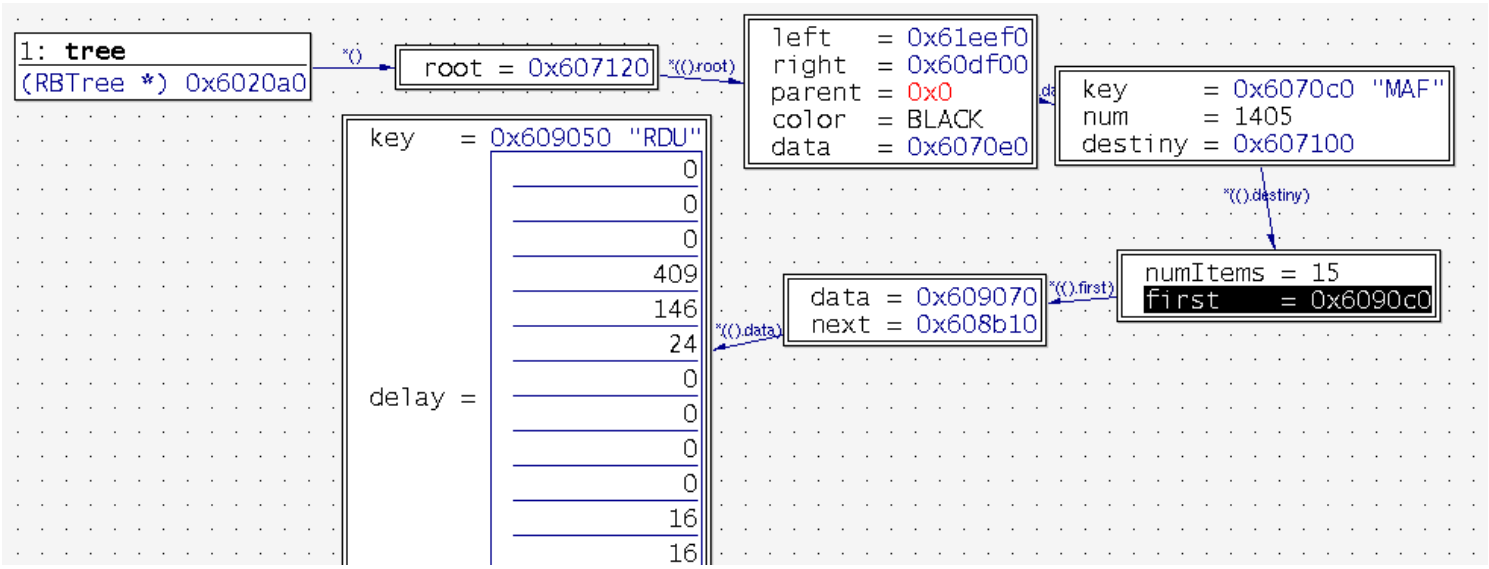

Figura 1: captura de pantalla de la application ddd que muestra el correcto funcionamiento de nuestro código al tener como raíz el aeropuerto con código "MAF" y uno de los destinos "RDU" el cual contiene una lista en la que vemos el delay y el numero de vuelos.