Pedro Pizarro Huertas

David Martín Vilar

SIstemas Operativos 2

Practica 3

# Desarrollo de la entrega

Hemos tomado mucho tiempo en las reparaciones del código de la practica 2 para poder llevar a cabo el código de la practica 3, después de muchos problemas no llegamos a arreglar los problemas con la lectura que nos surgieron, por eso no hemos podido llegar a implementar bien el código de la practica 3.

Frente a la practica 3, comentaré lo que hemos llevado a cabo para las distintas funciones que propone el menú:

El menú presenta 5 opciones:

1. **Creación del árbol.**

En la creación del árbol hemos utilizado el siguiente código:

printf("Introdueix fitxer que conte llistat d'aeroports: ");

fgets(str1, MAXLINE, stdin);

str1[strlen(str1)-1]=0;

printf("Introdueix fitxer de dades: ");

fgets(str2, MAXLINE, stdin);

str2[strlen(str2)-1]=0;

creacio\_arbre(str1, str2);

break;

En el cual en los dos printf pedimos al usuario que introduzca los datos de los ficheros a leer para rellenar el árbol y llamamos a la función “creacio\_arbre”, el cual contiene todo el código de lectura implementado en la practica 2, el cual genera el árbol a partir de los datos IATA de los aeropuertos origen y va añadiendo los distintos destinos de cada uno en una lista tal y como indicaba la entrega anterior.

1. **Guardar árbol en el disco.**

Nuestra idea de código era llevar a cabo lo siguiente:

void guardar\_llista(list \*l, FILE \*fp){

int n\_vols, retras;

list\_item \*item = malloc(sizeof(list\_item));

item = l->first;

while(item != NULL){

// Guardem el destí

fwrite(item->data->key, sizeof(char), strlen(item->data->key), fp);

// Guardem el nombre de vols

n\_vols = item->data->num\_times;

fwrite(&(n\_vols), sizeof(int), 1, fp);

// Guardem el retràs

retras = item->data->retraso;

fwrite(&(retras), sizeof(int), 1, fp);

item = item->next;

}

}

void guardar\_node(node \*x, FILE \*fp){

int n\_elements;

// Guardem l'origen

fwrite(x->data->key, sizeof(char), strlen(x->data->key), fp);

// Calculem i guardem el número d'elements

n\_elements = calcular\_lista\_num(x->data->link);

fwrite(&n\_elements, sizeof(int), 1, fp);

// Guardem els elements de la llista

guardar\_llista(x->data->link, fp);

}

void guardar\_arbre(node \*x, FILE \*fp){

if (x->left != NIL){

guardar\_arbre(x->left,fp);

}

guardar\_node(x, fp);

if (x->right != NIL){

guardar\_arbre(x->right,fp);

}

}

Desde la opción del menú hacemos tal que así después de pedir el nombre del archivo:

* Comprobamos si existe un árbol para llevarlo a cabo
* Si no existe pedimos el nombre del fichero que tendrá
* Creamos el magic number y lo escribimos, utilizamos otra función la cual no voy a poner llamada “calcular\_numero\_nodes” que cuenta el número de nodo que tenemos.
* Llamamos a la función guardar\_arbre que empieza el recorrido:

Siempre mirará la rama de la derecha antes de ejecutar la función de escribir el nodo, después a al de la izquierda, la función que escribe el nodo, la cual como indica en la práctica guarda el key del nodo en cuestión, el número de destinos que tiene (con la función “calcular\_lista\_num” y en la lista guardamos la key , el numero de vuelos y el retraso medio.

Los char\* (los cuales son las keys) se escriben tal que:

Fwrite(&variable, sizeof(char),strlen(variable),fp);

Y los ints (que son el resto de datos):

Fwrite(&variable, sizeof(int),1,fp);

1. **Leer árbol de disco**

Antes de leer tendríamos que ver si el magic number leído sería el correspondiente con un if(magic == MAGIC\_NUMBER).

Entonces hacemos la llamada a la función “carregar\_arbre”:

void carregar\_llista(list \*l, int n, FILE \*fp){

list\_data \*data;

int n\_vols, retras;

for(int i = 0; i < n; i++){

data = malloc(sizeof(list\_data));

data->key = malloc(sizeof(char)\*4);

fread(data->key, sizeof(char), 3, fp);

data->key[3] = '\0';

fread(&n\_vols, sizeof(int), 1, fp);

data->num\_times = n\_vols;

fread(&retras, sizeof(int), 1, fp);

data->retraso = retras;

insert\_list(l, data);

}

}

void carregar\_arbre(int n, FILE \*fp){

tree = (rb\_tree \*) malloc(sizeof(rb\_tree));

node\_data \*n\_data;;

int n\_elem;

init\_tree(tree);

for(int i = 0; i < n; i++){

n\_data = malloc(sizeof(node\_data));

n\_data->key = malloc(sizeof(char) \* 4);

n\_data->link = malloc(sizeof(list));

init\_list(n\_data->link);

fread(n\_data->key, sizeof(char), 3, fp);

n\_data->key[3] = '\0';

fread(&n\_elem, sizeof(int), 1, fp);

carregar\_llista(n\_data->link, n\_elem, fp);

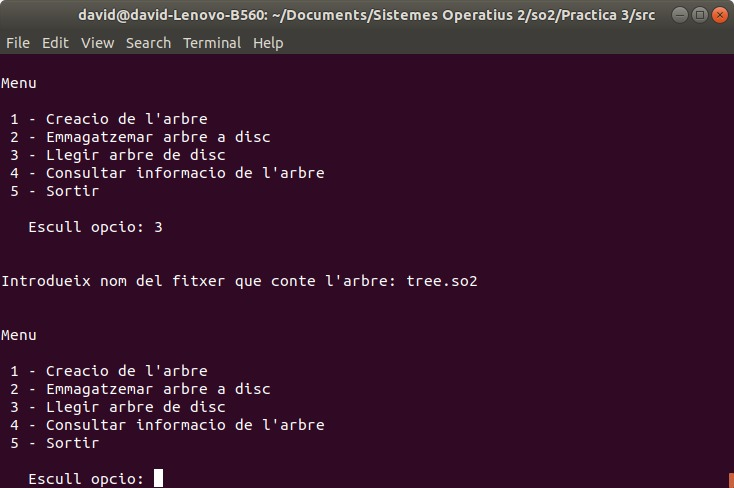
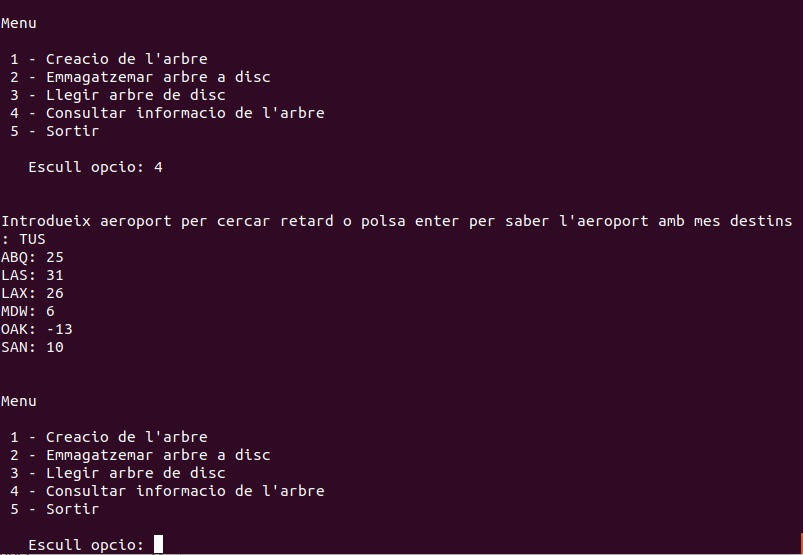
insert\_node(tree, n\_data);

}

}

El cual, dependiendo del número leído de nodos en el árbol, hace un fread del tamaño del código IATA, también lee el número de elementos y los añade al nodo iniciado anteriormente, llama al método que hace lo mismo, pero con los datos de cada elemento de la lista y finalmente se añade la lista al nodo iniciado y se inserta en el árbol.

Prácticamente como hacíamos en la practica 2.

En las imágenes se puede ver como se hace con éxito la lectura con el main de la solución ejecutable.

En la primera imagen se ve como se selecciona la opción, y en la segunda se pueden ver con éxito (con la función de la opción 4 del menú) los retrasos correctos de los vuelos leídos.

1. **Consultar información del árbol.**

Primero pedimos al usuario sobre que vuelo quiere buscar el retraso (o no introducir ninguno y buscar el aeropuerto con más destinaciones. Pero este no ha podido llevarse a implementación).

Una vez tenemos el dato introducido llamamos a la función siguiente con el IATA introducido por el usuario.

void calcular\_retard(char \*vol){

// Calcular retards

node\_data \*node = malloc(sizeof(node\_data));

list\_item \*item = malloc(sizeof(list\_item));

int media;

node = find\_node(tree, vol);

item = node->link->first;

while(item != NULL){

media = item->data->retraso / item->data->num\_times;

printf("%s: %d\n", item->data->key, media);

item = item->next;

}

free(node);

free(item);

}

Que simplemente se hace servir de un find\_node para encontrar el nodo que el usuario busca y imprime la media ya previamente calculada.

1. **Salir.**

Se llama a las funciones free correspondientes que liberan todos los datos de nuestro árbol y usados.