Sistemes Operatius II - Pràctica III

Gabriel Lluís i Pau Soler

27 de Novembre de 2019

1 Introducció

En aquesta pràctica hem d'implementar un menú (que es dona fet) amb les següents opcions:

1. Crear un arbre a partir de dos fitxers, un *diccionari* i una *llista de documents de Gutenberg*
2. Emmagatzemar l'arbre creat en un fitxer tal i com es diu a l'enunciat de la pràctica.
3. Lectura del fitxer que hem generat.
4. Consultes del top1 del arbre i quantes vegades una paraula és a l'arbre.
5. Soritr del programa.

Repartirem cada implementació en un apartat a continuació.

2 Crear l'arbre

És la implementació de la pràctica dos copiada a una funció. L'únic canvi que hem introduït és el lloc on obrim els fitxers.
Abans els obriem dins de la funció corresponent a on s'usaven; ara ho hem canviat tot a la funció *practica2*, així si l'usuari s'equivoca no hi ha una violació de segment directa i ens retorna al main, on hi ha el menú.

3 Emmagatzemar a disc

Per emmagatzemar l'arbre a disc obrim (o creem) un fitxer, guardem amb *fwrite* totes les dades, que són les següents:

- **Nombre màgic**: Hem de guardar primer el define en una variable, sinó *C* no sap el tipatge de la variable i no es pot assignar.
- **Nombre d'elements:** Per guardar el nombre d'elements l'hem hagut de guardar. A la funció *diccionariarbre* tenim un contador que augmenta sempre que

s'inserti un node i guarda el valor al tree. Evidentment hem modificat l'arbre per poder guardar l'enter, apart de modificar les funcions *init_tree* i *delete_tree* per no perdre memòria al allibrerar-la incorrectament.

- **Nodes:** Per cada node escrivim la longitud de l'string (hem modificat la classe node pertinent per a que funcioni), l'string i el nombre de vegades que surt. Amb un sol for i els *fwrite* corresponents a cada variable llegim tots els nodes que hi ha a l'arbre.

Cal mencionar que a main hem colocat una condició per si no hi ha un arbre creat. En cas de que no hi sigui (inicialitzat a NULL) no et deixa guardar-lo.

4 Recuperar l'Arbre

Fem el procés d'emmagatzemar a l'inversa. Llegim i comprovem el màgic amb l'*fread*, llegim el nombre d'elements i el coloquem al for per saber quants nodes hi ha. Per cada un recuperem el nombre de chars, reservem memòria per l'string, el nombre de cops i ajuntem el node a l'arbre.

Cal mencionar que a main hem col·locat una condició per si no hi ha un arbre creat. Si hi ha un arbre a memòria, primer borrem l'arbre existent i en cree un de nou.

5 Consultar Informació

Per solventar aquest apartat de la pràctica hem reutilitzat l'script per trobar el top 10.

Guardem l'arbre en un fitxer amb format paraula-nombrevegades. Llavors amb la comanda *system* cridem l'script al nom del fitxer i impirmeix el top 1.

Per retornar la paraula demanada creem un node amb aquesta key i en fem una cerca.

6 Big Endian i Little Endian

Abans de definir el que implica que una arquitectura sigui Big Endian o Little Endian, cal que definim els conceptes de byte més significatiu (MSB) i de byte menys significatiu (LSB). Definirem, doncs, aquests bytes com el que més modifica les dades si el canviem i el que menys les modifica, respectivament. Així doncs, una arquitectura Big Endian serà aquella que ordena els bytes de MSB a LSB i, per contra, una arquitectura Little Endian serà aquella que els ordena de LSB a MSB.

Tant a l'hora de carregar dades de disc com a l'hora d'emmagatzemar-les, el problema que trobem si no comprovem amb quin tipus d'arquitectura estem tractant és que si hem guardat a disc les dades en una arquitectura Little Endian, si les intentem carregar en una arquitectura Big Endian les dades canviaran, ja que aquestes dues arquitectures llegeixen les dades en ordre invers. Per posar un exemple més clar,

guardar dades en Little Endian i carregar-les en Big Endian (també a la inversa) seria el mateix que escriure una frase d'esquerra a dreta i després intentar llegir-la de dreta a esquerra.

En el cas del nostre arbre, per assegurar la compatibilitat entre aquestes dues arquitectures podem fer els següents passos:

1. Crear un mètode que comprovi amb quina de les arquitectures estem tractant. Amb escriure un nombre de 2 bytes i comprovar quin és el primer byte que s'ha escrit (més o menys significatiu) n'hauriem de tenir prou. També hauriem de saber per a quina arquitectura ha estat emmagatzemat l'arbre.

2. En cas que l'arquitectura en la que ens trobem i per la qual hem emmagatzemat l'arbre siguin la mateixa, llegim normalment. En cas contrari, haurem d'invertir l'ordre de lectura de cada una de les variables byte a byte, ja que el més significatiu en una arquitectura és el que ho és menys en l'altra i viceversa.

Cal dir que si alguna de les variables només ocupa un byte no hi ha cap problema de compatibilitat, aquest problema apareix quan escrivim variables que ocupen més d'un byte.