Sistemes Operatius II - Pràctica 3

Octubre 2019

La tercera pràctica se centra en en la persistència de l'estructura de l'arbre, és a dir, en poder desar i carregar la informació emmagatzemada a/de disc de forma binària.

Índex

1 La pràctica

La pràctica 2 s'ha centrat en la lectura de fitxers en format text. Aquesta pràctica se centrarà en la manipulació de fitxers binaris, formalment coneguts com a fitxers no formatats.

En concret, la pràctica 3 se centra en.

- Utilitzar una interfície de menú senzilla que permeti que l'usuari interactuï amb l'aplicació, veure secció 1.1. En aquesta pràctica es proporciona a l'estudiant del codi C d'aquesta interfície.
- La persistència de l'arbre. Per tal d'evitar haver de crear l'arbre cada cop que s'inicialitza l'aplicació, es proposa poder emmagatzemar i carregar l'estructura d'arbre a/de disc quan l'usuari vulgui, veure secció 1.3.

Es descriuen a continuació cadascun dels anteriors punts.

1.1 Interfície de menú

Per tal de facilitar la interacció de l'usuari amb l'aplicació es proposa desenvolupar una interfície de menú textual. El menú ha d'incloure les següents 5 opcions:

1. Creació de l'arbre. La creació de l'arbre correspon a la pràctica 2. En cas que hi hagi un arbre prèviament creat caldrà alliberar la memòria associada. Veure secció 1.2
2. Emmagatzemament de l'arbre. En seleccionar l'usuari aquesta opció l'aplicació haurà de demanar per teclat el nom del fitxer on es desarà l'arbre. A continuació l'aplicació desarà en aquest fitxer la informació dels nodes l'arbre mitjançant funcions d'entrada/sortida binàries. A la secció 1.3 s'especifica el format de fitxer en què s'emmagatzema l'arbre.
3. Lectura de l'arbre. En seleccionar l'usuari aquesta opció l'aplicació haurà de demanar per teclat el nom del fitxer amb l'arbre a llegir. L'aplicació llegirà del fitxer fent i crearà l'estructura d'arbre, veure secció 1.3. En cas que hi hagi un arbre prèviament creat caldrà alliberar la memòria associada.
4. Permetre fer consultes senzilles respecte la informació emmagatzemada a l'arbre, veure secció 1.4.
5. Sortida. Mitjançant aquesta opció s'allibera tota la memòria reservada i se surt de l'aplicació.

1.2 Creació de l'arbre

En seleccionar l'usuari aquesta opció l'aplicació haurà de demanar per teclat el fitxer de diccionari així com el fitxer amb la base de dades de fitxers a processar. L'aplicació crearà l'arbre i en finalitzar tornarà a mostrar el menú amb totes les opcions.

1.3 Emmagatzemament de l'arbre a disc

L'emmagatzemament de l'arbre a disc evita haver de crear l'arbre cada cop que s'engegui l'aplicació. Mitjançant el menú es podrà decidir si es vol desar o carregar la informació de l'arbre a disc. La informació s'haurà de desar en format binari, fent servir el format de fitxer especificat a la figura 1.

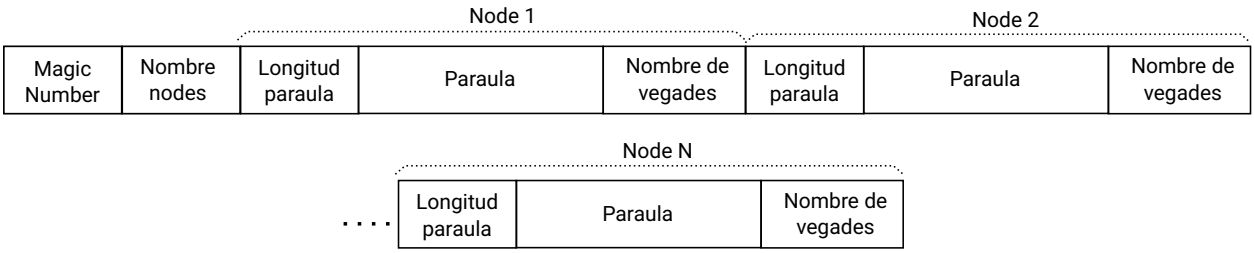

Figura 1: Format del fitxer per emmagatzemar la informació emmagatzemada a l'arbre.

S'especifica el format de fitxer per assegurar interoperatibilitat entre els fitxers generats per totes les aplicacions dels vostres companys.

Aquest és el detall de l'estructura del fitxer. Tots els sencers que s'especifiquen són amb signe[1].

- Nombre màgic, 4 bytes, un sencer. Aquest nombre màgic és un nombre que habitualment tenen tots els fitxers binaris al seu inici i permet identificar el contingut del fitxer. En aquesta pràctica el nombre màgic és 0x01234567.
- Nombre de nodes N de l'arbre, 4 bytes, un sencer.
- Per a cada node de l'arbre s'emmagatzema la següent informació: a) la longitud de la paraula del node, un sencer, b) els bytes de la paraula (sense el byte 0 que indica el final de la cadena), c) el valor del comptador associat a la paraula.

Els nodes es poden emmagatzemnar a disc en l'ordre que es vulgui. Observar doncs que al fitxer només es desa la informació emmagatzemada a cada node. No s'hi desa informació associada a l'estructura de l'arbre (pares, fills o germans de cada node).

En llegir la informació de l'arbre es llegirà el fitxer en el format especificat anteriorment. Atès que només es llegeix la informació de cada node és possible que l'estructura de l'arbre resultant sigui diferent de l'estructura que hi havia en el moment de desar-lo. Això no importa pas ja que l'estructura de l'arbre no té importància per a aquestes pràctiques.

En llegir l'arbre, comproveu que el nombre màgic és correcte. Si no ho és, imprimiu un missatge d'error i sortiu de l'aplicació. En cas contrari, podeu llegir el fitxer suposant que totes les dades estan en el format correcte.

1.4 Consulta d'informació emmagatzemada a l'arbre

Un cop creat l'arbre, l'usuari haurà de poder realitzar dues consultes (s'hi poden afegir altres consultes si així es vol),

1. Donada una paraula, imprimiu per pantalla el valor del comptador associat.
2. Cercar la paraula que apareix més vegades.

[1] No cal assegurar compatibilitat entre arquitectures big i little-edian.

2 Implementació

Aquesta secció dóna alguns consells per tal d'implementar correctament aquesta pràctica.

Es proporciona a l'estudiant del codi en C del menú a utilitzar. No s'ha inclòs el codi associat a cadascuna de les opcions possibles. Tingueu en compte que l'usuari pot escollir les opcions del menú en qualsevol ordre i, per tant, caldrà tenir en compte les restriccions associades. Per exemple, a l'hora de desar l'arbre cal assegurar que hi ha un arbre a memòria; a l'hora de llegir un arbre de disc cal alliberar l'arbre en cas que n'hi hagi algun carregat a memòria, etc.

Es proposa seguir el següent ordre a l'hora d'implementar la pràctica

1. Implementeu la part del menú que permet crear l'arbre. El codi ha de funcionar amb la base de dades completa.
2. Implementació de les funcions per desar i carregar la informació de l'arbre a disc en format binari tal com s'ha comentat a la secció 1.3. Comproveu que en llegir un fitxer de disc s'imprimeixen per pantalla els mateixos valors que s'imprimeixen si es crea l'arbre directament amb l'opció 1.
3. Un cop arribat aquí, assegureu-vos que tot funciona bé amb el `valgrind`.
4. La consulta de la informació associada a l'arbre.
5. Assegureu-vos que tot funciona bé amb el `valgrind`.

Es proporciona amb aquesta pràctica un fitxer `tree.so2` que ha estat creat amb el format de la figura 1. La vostra aplicació hauria de poder llegir aquest fitxer. En cas que hi feu una consulta, la paraula "the" és la que apareix més vegades – en concret, 17833 vegades – i la paraula "and" apareix 15485 vegades.

3 Entrega

El fitxer que entregueu s'ha d'anomenar `P3_NomCognom1NomCognom2.tar.gz` (o `.zip`, o `.rar`, etc), on `NomCognom1` és el nom i cognom del primer component de la parella i `NomCognom2` és el nom i cognom del segon component de la parella de pràctiques. El fitxer pot estar comprimit amb qualsevol dels formats usuals (`tar.gz`, `zip`, `rar`, etc). Dintre d'aquest fitxer hi haurà d'haver dues carpetes: `src`, que contindrà el codi font, i `doc`, que contindrà la documentació addicional en PDF. Aquí hi ha els detalls per cada directori:

- La carpeta `src` contindrà el codi font. S'hi han d'incloure tots els fitxers necessaris per compilar i generar l'executable. El codi ha de compilar sota Linux amb la instrucció `make`. És recomanable que poseu les funcions en fitxers diferents agrupats per la seva funcionalitat (per exemple, un fitxer per la creació de l'arbre, un altre per llegir i desar l'arbre, ...). El codi font ha d'estar comentat (com a mínim les funcions).

  Per simplificar, es pot suposar que el codi s'executa dins del directori en què està situada la base de dades. El fitxer de diccionari (que pot tenir qualsevol nom) també estarà situat en aquest directori. El codi no tindrà cap paràmetre, atès que tot el control es farà des del menú de l'aplicació

  ```
  practica3
  ```

- El directori `doc` ha de contenir un document (màxim tres pàgines, en format PDF, sense incloure la portada). Observeu que la secció 1.3 es comenta, a peu de pàgina, que "No cal assegurar compatibilitat entre arquitectures big i little-edian." Expliqueu què vol dir que una arquitectura pugui ser big-endian o little-endian. Comenteu també quines implicacions té a l'hora de emmagatzemar o carregar dades de disc. Quines implicacions té per a cadascuna de les dades que vosaltres emmagatzemeu a disc? Què cal fer per assegurar que l'arbre que emmagatzemeu a disc sigui compatible entre arquitectures big-endian i little-endian? Doneu resposta a aquestes preguntes donant exemples explícits perquè un lector que no conegui el tema pugui saber com procedir i fer el vostre codi compatible entre arquitectures big-endian i little-endian.

La data límit d'entrega està indicat al document de planificació. El codi té un pes d'un **80%** (codi amb funcions comentades, codi modular i net, ús correcte del llenguatge, bon estil de programació, el programa funciona correctament, tota la memòria és alliberada, sense accessos invàlids a memòria, etc.). Tingueu en compte que es comprovarà el bon funcionament del vostre codi fent servir el `valgrind`. El document té un pes del **20%** restant.