Sistemes Operatius II - Pràctica 1

Setembre del 2016

Índex

1 Introducció

A l'assignatura de Sistemes Operatius II es realitzarà un únic projecte pràctic al llarg del curs. L'objectiu general del projecte pràctic és desenvolupar una aplicació (sense interfície gràfica) que permeti extreure i indexar la informació dels vols d'avió entre ciutats dels Estats Units. Els fitxers de text a processar són proporcionats pel professor tot i que també es poden baixar lliurement d'Internet, veure http://stat-computing.org/dataexpo/2009/the-data.html. Cada fitxer és un fitxer de text pla i conté la informació dels vols d'un determinat any. Cada fila de cadascun d'aquests fitxers conté la informació d'un vol. Entre altres coses, conté informació sbore l'aeroport d'origen, l'aeroport destí, el retard en sortir, el retard en arribar, etc. L'objectiu és extreure part d'aquesta informació i gestionar-la.

Les pràctiques es realitzaran en llenguatge C i estaran centrades en els següents aspectes del llenguatge:

- Pràctica 1. Lectura d'un fitxer de la base de dades i inserció de les dades en un arbre binari a mesura que es llegeixen les dades.
- Pràctica 2. Preparació del codi perquè més endavant es pugui aprofitar la capacitat multiprocessador de l'ordinador.
- Pràctica 3. Emmagatzematge i lectura de l'arbre de disc així com extracció d'informació de les dades emmagatzemades a l'arbre.
- Pràctica 4. Implementació de l'algorisme per aprofitar els múltiples processadors de l'ordinador.

Les pràctiques estan encavalcades entre sí. És a dir, per a realitzar una pràctica és necessari que la pràctica anterior funcioni correctament. Assegureu-vos doncs que les pràctiques estan dissenyades de forma modular perquè puguin ser ampliades de forma fàcil per realitzar la següent pràctica. A cada pràctica es donaran les idees per poder aconseguir-ho.

Les pràctiques es realitzen en parella i podeu discutir amb els vostres companys la solució que implementeu. En tot cas, cada parella ha d'implementar el seu propi codi.

2 La pràctica

L'objectiu de la primera pràctica és aprendre a fer servir els punters en C. Per aconseguir aquest objectiu llegireu el fitxer que conté informació dels vols, extraureu la informació necessària de cada línia i inserireu la informació en un arbre binari a mesura que es llegeix el fitxer.

A les següents subseccions s'explica com està estructurada la informació al fitxer i com ha d'estar estructurat l'arbre binari on s'inserirà la informació. A la secció 3 es realitza una proposta per planificar la feina a realitzar.

2.1 Lectura de la informació del vols

Cada fitxer conté a cada fila la informació d'un vol. Hi ha múltiple informació emmagatzemada a cada fila i està separada per comes (","). És per això que el format de fitxer associat també rep el nom de Comma Separated Values (CSV).

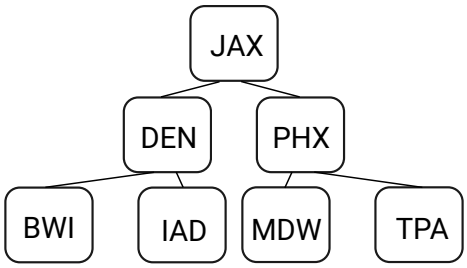

Figura 1: Arbre binari indexats per l'aeroport d'origen.

En aquesta pràctica s'ha inclòs una versió reduïda del fitxer que s'haurà de processar, veure directori `llegir-csv`. El fitxer conté només 10.000 files, mentre el fitxer complet conté més de 7 milions de files. Es molt recomanable que treballeu amb el fitxer reduït fins que us assegureu (amb el `valgrind`) que la vostra aplicació funciona correctament.

Per aquest projecte es demana extreure de cada fila la següent informació

- Columna 4: dia de la setmana del vol, especificat amb un nombre sencer. Un 1 es refereix a dilluns, mentre que un 7 es refereix a diumenge.
- Columna 15: retard d'arribada a l'aeroport destí, en minuts. Aquest valor és un nombre sencer i pot ser positiu o negatiu.
- Columna 17: aeroport d'origen, en un codi en el format de l'International Airport Abbreviation Code (IATA)[1]. El codi té, la gran majoria de les vegades, una longitud de 3 lletres.
- Columna 18: aeroport destí, en el codi de la IATA. El codi té, la gran majoria de les vegades, una longitud de 3 lletres.

Al directori `llegir-csv` s'incou un codi per llegir cada línia del fitxer. Per a més informació al respecte mireu la secció 3.1.

2.2 Inserció de la informació en un arbre binari

Cal inserir la informació extreta de les files en un arbre binari, vegeu la figura 1. En aquest projecte els nodes de l'arbre binari estan indexats pel codi IATA de l'aeroport d'origen. Per a cada node s'emmagatzemen a l'esquerra els codis IATA d'aeroport que lexicogràficament són anteriors al codi IATA del node, mentre que a la dreta es troben els codi IATA d'aeroport lexicogràficament posteriors al codi IATA del node.

Cada node de l'arbre haurà d'emmagatzemar els aeroports destins d'aquell node així com la informació dels retards associats, vegeu figura 2. En aquesta figura es mostra, per a l'aeroport origen DEN, una llista dels aeroports destí. Els aeroports destí són, en aquest exemple, JFK, PHX, BWI i JAX. Aquesta informació està emmagatzemada en una llista enllaçada. La llista enllaçada només ha de contenir els aeroports destí de l'aeroport origen; no pot contenir cap aeroport que no sigui destí de l'origen. Per a cada aeroport destí caldrà emmagatzemar la informació necessària per poder calcular el *retard mig* per a cada dia de la setmana. Així, per exemple, per a la segona posició

[1]Els codis IATA dels aeroports dels Esats Units es poden obtenir aquí: http://stat-computing.org/dataexpo/2009/supplemental-data.html.

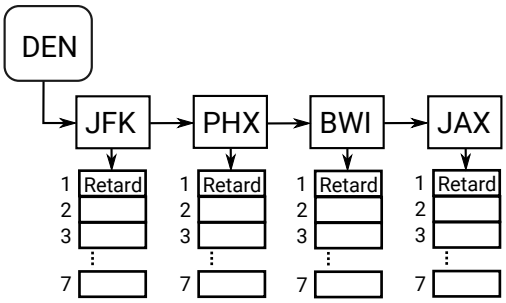

Figura 2: En aquesta figura es mostra la informació emmagatzemada al node DEN de l'arbre. Aquí es mostra una llista enllaçada amb els aeroports destí per a DEN.

del vector l'aeroport destí PHX s'emmagatzema el retard mig acumulat per a tots els vols que s'han realitzat els dimarts entre DEN i PHX.

En aquest projecte no caldrà implementar l'arbre binari ni tampoc la llista enllaçada. Se us proporcionen de les implementacions necessàries perquè pugueu implementar la funcionalitat demanada. A la secció 3.2 se us dóna més informació al respecte.

3 Implementació i planificació

Per tal d'assolir amb èxit aquesta pràctica es recomana revisar abans de tot la Fitxa 1 (recordatori bàsic del llenguatge C). És particularment important la secció "Manipulació de cadenes caràcters". A l'hora de programar és important seguir una estructura modular atès que la resta de pràctiques d'aquesta assignatura es basaran en aquesta primera pràctica. Es proposa a continuació una forma de procedir per a la implementació d'aquesta pràctica.

3.1 Lectura i extracció de dades del fitxer

Al directori `llegir-csv` s'inclou un codi per llegir un fitxer línia a línia. Hi ha moltes funcions que permeten llegir un fitxer i aquí, en particular, es proposa utilitzar la funció *fgets*. Aquesta funció permet llegir tota la línia d'un fitxer. En aquest projecte és molt important utilitzar aquesta funció (o una de similar ja que a la pràctica 4 caldrà separar clarament la lectura del fitxer de la del processament de cada fila.

Es proposa seguir el següent esquema

1. Compileu i executeu el codi. Comproveu que el codi funciona correctament amb `valgrind`.
2. Modifiqueu el valor de MAXCHAR a 10. Torneu a compilar i executeu. Sembla que haver modificat el valor de MAXCHAR no té efecte sobre la sortida que s'imprimeix a pantalla. Comproveu si ara `valgrind` dóna algun error. El comportament que observeu és l'esperat?
3. Torneu a deixar MAXCHAR a 100. Modifiqueu el codi per extreure de cada fila les columnes demanades a la secció 2.1. Llegiu-vos la fitxa 1 per tenir idees sobre com fer-ho. Un cop extreta la informació, imprimiu-la per pantalla si voleu. Observeu que les columnes 4 i 15 contenen una cadena de caràcters que haureu de convertir en un sencer (ja que haureu d'operar aritmèticament sobre els sencers): per fer aquesta conversió disposeu de la funció *atoi*, vegeu manual amb `man`. Les columnes 17 i 18 contenen els codis dels aeroports d'origen i destí.

4. Comproveu que el codi funciona correctament passant el `valgrind`.

3.2 Inserció de la informació dels aeroports d'origen a l'arbre binari

Els nodes de l'arbre hauran d'indexar els codis IATA dels aeroports d'origen, veure figura 1. En aquesta secció ens concentrem només en la inserció dels codis IATA d'aeroports d'origen a l'arbre binari. A la subsecció següent es comenta respecte la llista enllaçada que es farà servir per emmagatzemar informació sobre els aeroports destí i el retard mig.

Tenir en compte que tant l'arbre binari com la llista enllaçada són estructures dinàmiques. En iniciar la lectura del fitxer l'arbre és buit (no conté cap element). A mesura que es va llegint informació del fitxer s'anirà afegint informació als nodes de l'arbre. De la mateixa forma, també s'aniran afegint aeroports destí i, per tant, les llistes enllaçades anirant creixent de mida.

El directori `src/arbre-binari` disposa del codi que fareu servir en aquestes pràctiques. En concret, els fitxers `red-black-tree.c` i `red-black-tree.h` contenen la implementació d'un arbre binari balancejat (Sabeu què és un arbre balancejat? Busqueu-ho!). El codi de l'arbre està preparat, però, per indexar sencers als nodes de l'arbre en comptes de cadenes de caràcters. El fitxer `main.c` conté un exemple que insereix sencers a l'arbre.

Es proposa seguir els següents passos en aquesta pràctica:

1. Executeu el codi d'exemple. Per compilar-lo cal executar `make` dins del directori en què es troba el fitxer `Makefile`.

2. Editeu i analitzeu el funcionament de l'exemple, en concret de la funció `main`. Observeu com s'insereixen nous nodes a l'arbre i com es tracta el cas en què un node ja existeix a l'arbre.

3. Executeu el codi amb el `valgrind`. Observeu que valgrind dona un missatge d'error de memòria no alliberada. Sou capaços d'arreglar el problema?

4. Modifiqueu el codi per adaptar-lo a les necessitats d'aquesta pràctica: en comptes d'indexar l'arbre per sencers cal indexar-lo per cadenes de caràcters: els codis IATA d'aeroports. Els codis IATA dels aeroports tenen la gran majoria de les vegades una longitud de 3 caràcters. Implementeu el codi de forma que s'ignorin (i.e. no s'insereixin a l'arbre) tots aquells codis IATA que tinguin una longitud superior a 3 caràcters. En altres paraules, al fitxer hi pot haver codis IATA amb una longitud superior a 3 caràcters però només s'han s'han d'inserir a l'arbre aquells que tinguin una longitud de 3 caràcters (o inferior).

   Per poder indexar l'arbre pels codis IATA d'aeroport caldrà editar els fitxers `red-black-tree.h` i `red-black-tree.c`, modificar les estructures corresponents, així com algunes de les funcions. Us serà particularment útil la funció *strcmp*, que permet comparar dues cadenes de caràcters. Llegiu atentament al manual d'aquesta funció (la trobareu a Internet, per exemple) per saber com funciona.

5. Un cop modificat el codi integreu-lo amb la lectura del fitxer CSV de la subsecció anterior. Modifieu el codi només per indexar l'arbre per l'aeroport d'origen.

6. No cal imprimir res per pantalla. Assegureu-vos a més que tot funciona correctament fent servir el `valgrind`.

3.3 Inserció de la informació de destí a la llista enllaçada

En aquesta secció es comenta la implementació de la llista enllaçada que conté la informació destí. Recordar que la llista enllaçada és dinàmica: a mesura que es vagi llegint informació va creixent de mida. No feu cap mena de suposició sobre la mida d'aquesta llista.

El directori `src/linked-list` conté la implementació d'una llista enllaçada. La implementació disponible és una llista preparada per emmagatzemar sencers, igual que l'arbre que heu fet servir a la secció anterior.

Es proposa seguir els següents pasos:

1. Compileu i executeu el codi. Per compilar-lo cal executar `make` dins del directori en què es troba el fitxer `Makefile`.

2. Comproveu que `valgrind` no dóna errors d'execució.

3. Modifiqueu el codi per adaptar-lo a les necessitats d'aquesta pràctica: en comptes d'indexar la llista per sencers caldrà indexar-la per cadenes de caràcters (els codis IATA dels aeroports, no suposeu pas que tenen una longitud de 3 caràcters!). Per això caldrà editar els fitxers `linked-list.h` i `linked-list.c`, modificar les estructures corresponents, així com algunes de les funcions. No és necessari modificar el codi perquè la llista contingui els noms dels aeroports destí ordenats alfabèticament. Tingueu en compte que a cada element de la llista caldrà guardar la informació necessària per poder calcular el retard mig per a cada dia de la setmana. Quina és la informació que us farà falta emmagatzemar? Es demana utilitzar una implementació que faci servir vectors i/o matrius dinàmiques. D'aquesta forma `valgrind` pot trobar fàcilment problemes en cas que hi hagi accessos invàlids.

4. Integreu el vostre codi amb la resta del codi. Comproveu que tot funciona correctament fent servir el `valgrind`.

4 Entrega

El fitxer que entregueu s'ha d'anomenar `P1_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 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:

- El directori `doc` ha de contenir un document (tres o quatre pàgines, màxim cinc pàgines, en format PDF, sense incloure la portada) explicant breument aquells detalls o decisions interessants de la vostra aplicació que no estiguin comentats a l'enunciat d'aquesta pràctica. No repetiu fil per randa el que es comenta en aquest enunciat i no expliqueu o incloeu els detalls del vostre codi ja que aquests es poden veure en llegir el codi.

- La carpeta `src` contindrà el codi font comentat (com a mínim les funcions). Els comentaris poden estar en anglès, català o castellà. 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`. Editeu el fitxer `Makefile` en cas que necessiteu afegir fitxers C que s'hagin de compilar.

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 els professors comprovaran el bon funcionament del vostre codi fent servir el `valgrind` amb servir el `fitxer.csv` (o un subconjunt d'aquest). El document tenen un pes del **20%** restant.