Sistemes Operatius II - Pràctica 2

Octubre del 2014

Aquesta primera pràctica dóna una visió global de les pràctiques a desenvolupar al llarg d'aquesta assignatura. S'indicaran també les tasques a desenvolupar en la primera pràctica. La data d'entrega de la primera pràctica és el diumenge **26 d'octubre del 2014**.

Índex

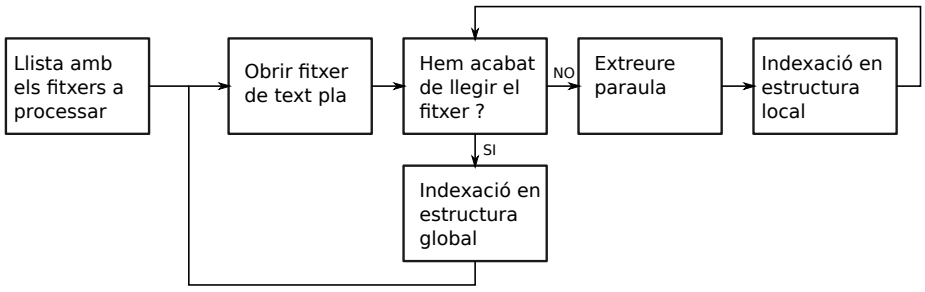

Figura 1: Diagrama de blocs de l'algorisme a implementar en la pràctica 1.

1 Introducció

Aquesta és la segona pràctica de l'assignatura de Sistemes Operatius II. Recordeu que en aquesta assignatura es realitzarà a les pràctiques un únic projecte pràctic al llarg del curs. A la primera pràctica heu après a llegir dades des d'un fitxer de text pla i a manipular cadenes de caràcters. En aquesta segona pràctica ens centrarem en

1. Estendre l'aplicació de la pràctica 1 perquè llegeixi els fitxers de text pla provinents del web Gutenberg (http://www.gutenberg.org/). Aquests fitxers de text són llibres electrònics gratuïts en anglès. L'aplicació haurà de llegir cadascun d'aquests fitxers de text i extreure totes les paraules vàlides que contenen aquests.

2. S'utilitzaran estructures de llistes i d'arbre per tal d'indexar les paraules que s'han estret dels llibres. En aquesta pràctica es proveeix del codi C de les llistes i de l'arbre. Per tant, us haureu de concentrar en utilitzar aquest codi per implementar la funcionalitat demanada.

2 La pràctica

A la figura 1 es mostra un diagrama de blocs (simplificat) de l'algorisme que s'ha d'implementar en aquesta pràctica. L'algorisme, de forma iterativa, obrirà els fitxers de text a processar (la llista amb els noms dels fitxers de text a processar està emmagatzemada en un fitxer de configuració, veure més endavant). Per a cada fitxer de text s'hauran d'extreure les paraules vàlides i indexar-les en una estructura local. Aquesta estructura local és una estructura que estarà buida en iniciar l'extracció de paraules per a un determinat fitxer. Finalitzada l'extracció de paraules d'un fitxer de text es copiaran les dades indexades de l'estructura local a una estructura global. En aquesta estructura global hi hauran indexades totes les paraules de tots els fitxers. Observar que la indexació de les paraules es realitza doncs amb un procediment de dos passos: primer en una estructura local i després en una estructura global.

```
595
./etext00/00ws110.txt
./etext00/1cahe10.txt
./etext00/1vkip11.txt
./etext00/2cahe10.txt
./etext00/2yb4m10.txt
./etext00/8rbaa10.txt
./etext00/8year10.txt
./etext00/andsj10.txt
./etext00/beheb10.txt
./etext00/benhr10.txt
./etext00/bgita10.txt
./etext00/btowe10.txt
./etext00/cbtls12.txt
./etext00/chldh10.txt
./etext00/cptcr11a.txt
./etext00/cstwy11.txt
./etext00/cyrus10.txt
./etext00/dmsnd11.txt
./etext00/dscmn10.txt
```

Figura 2: Exemple de l'estructura del fitxer `llista.cfg`.

Es descriuen a continuació els blocs de llista amb els fitxers a processar, extracció de paraules i la indexació local i global de les paraules.

2.1 Fitxers a processar

L'aplicació a desenvolupar ha de permetre especificar (com a argument al programa) un fitxer de configuració amb la llista dels fitxers de text a processar. La figura 2 mostra un exemple de l'estructura d'aquest fitxer: a la primera línia s'emmagatzema el nombre de fitxers a processar (en aquest cas 595, tot i que hi pot haver qualsevol valor). A cada línia del fitxer de configuració hi haurà un nom de fitxer del qual caldrà extreure les paraules i construir l'estructura d'indexació local (primer pas de l'extracció de paraules). Cadascun dels fitxers a processar és un fitxer de text pla. Tots els fitxers (el de configuració i els fitxers a processar) estan disponibles al campus de l'assignatura.

2.2 L'estructura local

Les paraules vàlides, un cop extretes del fitxer, s'indexaran en una estructura local. Aquesta estructura local té per objectiu emmagatzemar totes les paraules que apareixen en un determinat fitxer de text. Per a cada paraula només caldrà emmagatzemar el nombre de vegades que aquesta apareix al text en qüestió. La indexació local es realitzarà mitjançant una estructura de *hash*, veure figura 3. En aquesta figura, l'índex de l'esquerra és el nombre de *hash* mentre que al costat de cada paraula hi ha un nombre que indica el nombre de vegades que apareix al fitxer.

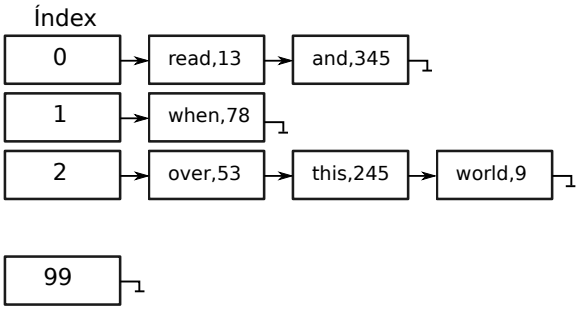

Figura 3: Exemple de taula de *hash* per realitzar la indexació local. En aquesta figura 3 la taula te mida de 100 elements (de 0 a 99). Cada índex de la taula correspon a una llista enllaçada.

Donada una paraula, l'algorisme de *hash* permet calcular per aquesta paraula un nombre sencer (veure secció d'implementació per saber com es pot calcular aquest nombre sencer). El nombre sencer es fa servir com a índex en una taula. Cada índex de la taula correspon a una llista enllaçada. Així, per exemple, el nombre *hash* associat a la paraula "over" és 2. El primer cop que aquesta paraula apareix al fitxer de text s'insereix a la posició 2 de la taula indicant que (fins ara) apareix un sol cop al fitxer. Les següents vegades que aquesta paraula aparegui al fitxer caldrà incrementar només el comptador.

Observar que l'estructura local és una estructura dinàmica: en obrir un fitxer de text estructura és buida (no hi ha cap paraula emmagatzemada a la taula); a mesura que es van afegint paraules l'estructura de *hash* va creixent. En finalitzar l'extracció i indexació local d'un fitxer aquesta estructura emmagatzemarà totes les paraules vàlides que apareixen en el fitxer. Caldrà aleshores copiar les dades de l'estructura local a una estructura global.

Per facilitar la implementació d'aquesta funcionalitat es proporciona el codi d'una llista enllaçada (fitxer `linked-list.c`) així com la funció per calcular el valor de *hash* (fitxer `hash.c`). Veure la secció d'implementació per a més detalls.

2.3 L'estructura global

L'estructura global emmagatzema totes les paraules de tots els fitxers que s'han processat. En aquest projecte pràctic es proposa utilitzar una estructura en arbre per tal d'emmagatzemar aquesta informació. A la figura 4 es mostra un exemple de l'estructura d'arbre a utilitzar en cas que es processin un total de 5 fitxers. L'arbre estructura les paraules trobades en els diversos fitxers i els nodes estan ordenats alfabèticament (a l'esquerra el node que alfabèticament va abans, a la dreta el node que alfabèticament va després). En cada node s'emmagatzema la següent informació: la paraula, el nombre de fitxers diferents en que aquesta paraula apareix i un vector indicant el nombre de vegades que apareix en cadascun dels fitxers. Per exemple, al node arrel hi ha emmagatzemada la paraula "great". Aquesta paraula apareix en els 5 fitxers: al primer fitxer de text

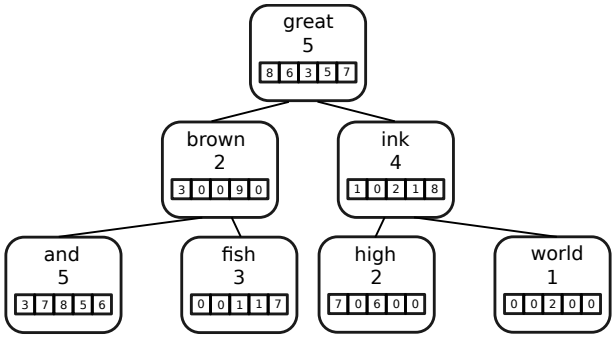

Figura 4: Exemple de l'estructura d'arbre a utilitzar per a realitzar la indexació global.

analitzat apareix 8 vegades, al segon 6 vegades, al tercer 3 vegades, al quart 5 vegades i al cinquè 7 vegades.

Observeu que en copiar les dades de l'estructura local a l'estructura global caldrà actualitzar tots els camps necessaris així com inserir noves paraules en cas que aquestes no existeixin.

Per facilitar la implementació d'aquesta part es proporciona una implementació d'una estructura d'arbre de tipus red-black (fitxer `red-black-tree.c`) que té bones propietats, veure "Red-black tree" a la Wikipedia. A la secció d'implementació es donen més detalls al respecte.

3 Implementació i planificació

Abans de començar a programar, és important que entengueu bé tots els passos a implementar en aquesta pràctica i, en particular, les estructures local i global utilitzades per a la indexació de les paraules. És molt important seguir una estructura modular atès que la resta de pràctiques es basaran en aquesta primera pràctica. Per això:

- Tingueu en compte que a la pràctica 4 s'implementarà l'algorisme fent servir múltiples fils. És per això que es molt recomanable programar el codi de forma que hi hagi una funció (diferent del main) que llegeixi el fitxer de configuració i cridi, per a cada fitxer de text, una altra funció que s'encarregarà d'extreure les paraules i indexar-les en una estructura local.

- Al directori `exemple` de la pràctica se us mostra com utilitzar les estructures local i global per tal d'indexar paraules. Utilitzeu-lo! Tingueu en compte que caldrà modificar el codi `linked-list.c` i `red-black-tree.c` per adaptar-lo a les necessitats particulars d'aquesta pràctica.

- A més, és molt recomanable assegurar-se que un mòdul de l'algorisme funciona correctament abans d'integrar-lo amb la resta de l'aplicació.

```c
#include <stdio.h>
#include <string.h>

#define MAXCHAR 100
#define SIZE    100

int main(void)
{
  int i, len, seed, sum, hash;
  char cadena[MAXCHAR];

  printf("Introdueix la paraula: ");
  fgets(cadena, MAXCHAR, stdin);

  len = strlen(cadena) - 1;
  sum = 0;
  seed = 131;
  for(i = 0; i < len; i++)
    sum = sum * seed + (int)cadena[i];

  hash = sum % SIZE;

  printf("El valor de hash es %d\n", hash);
}

```

Figura 5: Exemple de càlcul de la funció de *hash* per a una paraula o cadena de caràcters.

Se us recomana seguir la següent planificació de treball:

1. Comenceu per analitzar l'exemple d'ús de les estructures locals (una llista) i globals (un arbre), vegeu directori `exemple`. Tingueu en compte que l'exemple ha estat preparat per tal d'utilitzar un valor sencer per tal d'indexar dades en la llista i l'arbre. Discutiu amb el vostre company com heu de modificar aquests fitxers per tal d'adaptar-lo a les necessitats de l'aplicació que heu d'implementar. No ho feu encara, ja que aquests detalls s'expliquen més endavant!

2. Programeu la lectura del fitxer de configuració. Per això l'executable ha de permetre especificar un únic argument per línia de comandes, que es un fitxer de text que conté una llista de fitxers a processar:

   ```
   $ ./practica2 llista.cfg
   ```

   Es proposa doncs implementar la lectura del fitxer de configuració. Quina funció fareu servir per llegir aquest fitxer ? La funció *fgetc*, *fgets*, *fscanf*, o *fread* ? La figura 2 mostra un exemple de l'estructura d'aquest fitxer: a la primera línia s'emmagatzema el nombre de fitxers a processar. En aquest exemple és 595 (per passar una cadena de caràcters a un sencer podeu fer servir la funció *atoi* de C). Després, a cada línia, hi haurà el nom de fitxer del qual caldrà extreure les paraules i construir l'estructura d'indexació local. El primer fitxer tindrà associat el nombre 0, el segon fitxer el nombre 1, i així successivament fins al 594. Aquest nombre s'utilitzarà per indexar les vegades que apareix cada paraula als vectors de l'estructura global, vegeu 4.

3. Concentreu-vos a continuació en la implementació de la indexació local. Feu servir el codi de la pràctica 1 per tal d'extreure les paraules vàlides que conté el fitxer de text. Cada cop que s'obri un nou fitxer de text s'inicialitzarà una estructura local buida i en acabar s'alliberarà (és millor alliberar que buidar en previsió de la pràctica 4) tota la memòria ubicada per aquest. Per tal d'inserir la paraula a l'estructura local cal calcular el seu valor de *hash* (l'objectiu d'aquesta pràctica no és implementar la millor funció de *hash* ja que les funcions de *hash* són un tema de recerca en sí, sinó implementar-ne una que sigui prou bona). Per ajudar-vos a implementar aquesta part se us dóna el següent codi

   (a) Al fitxer `hash.c` (copiat a la figura 5) es mostra una forma senzilla de calcular el hash per a una paraula. És important mencionar que a la variable SIZE de la línia 5 es defineix la mida de la taula de *hash* (vegeu la figura 3). És a dir, un valor de N indica que els valors de sortida de la funció de *hash* aniran de 0 a $N - 1$.

   (b) Al fitxer `linked-list.c` s'implementa una llista enllaçada que us ajudarà a implementar l'estructura local. Heu de modificar el codi per adaptar-lo a les vostres necessitats. A més, la implementació actual pot no incloure totes les funcions que fan falta per implementar la pràctica. Us serà molt útil la funció *strcmp*, una funció de la llibreria de C que permet comparar dues cadenes de caràcters. L'estructura local ha de ser dinàmica, és a dir, s'introduiran elements nous (tupla paraula i nombre d'aparicions al text) cada cop que es necessitin. No heu de suposar cap mida màxima del nombre de paraules que hi pot haver en un text.

4. Un cop us funcioni el pas 2 ens concentrarem en implementar la indexació en una estructura global. Observeu de nou la figura 4: a cada node s'emmagatzema una paraula, el nombre de fitxers en què apareix i un vector que indica el nombre de vegades que la paraula apareix en cadascun dels fitxers. En particular, per a l'exemple de la figura 2 el vector tindrà una mida de 595 posicions, i la posició 0 del vector estarà associat al fitxer 0 (primer fitxer de la llista), la posició 1 del vector estarà associat al fitxer 1 (segon fitxer de la llista), i així successivament. Per ajudar-vos a implementar aquesta part se us dóna una implementació d'un arbre binari balancejat, vegeu el fitxer `red-black-tree.c`. Heu d'adaptar aquest codi a les vostres necessitats.

Per assegurar el correcte funcionament de l'aplicació, se us proposa realitzar les següent proves:

- Per fer proves curtes modifiqueu la primera línia del fitxer de configuració i poseu-hi un valor més petit que el que hi ha actualment. Proveu amb un valor de 5, 10 o 20. Per les proves finals caldrà posar-hi el valor original, 595.

- En finalitzar el processament local per cadascun dels fitxers, imprimiu per pantalla el nombre de vegades que apareix cada paraula al text (vegeu funció *dumpList* al fitxer `linked-list.c`).
- En finalitzar el processament global, imprimiu per pantalla cadascuna de les paraules que s'han trobat i el nombre de vegades que s'ha trobat la paraula en total.
- Com afecta el valor de SIZE a `hash.c` al rendiment del programa ? Per què ? Per tal de mesurar el temps d'execució d'un programa podeu fer servir la següent instrucció des de terminal

  ```
  $ time ./practica2 llista.cfg
  ```

- Quina és la paraula més llarga que apareix en els fitxers de text ? Quina paraula és ? En quin fitxer apareix ?

4 Entrega

El fitxer que entregueu s'ha d'anomenar `P2_Cognom1Cognom2.tar.gz` (o `.zip`, o `.rar`, etc), on `Cognom1` és el cognom del primer component de la parella i `Cognom2` 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 tres carpetes: `src`, que contindrà el codi font, `proves`, que contindrà resultats d'execució 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`. Editeu el fitxer *Makefile* en cas que necessiteu afegir fitxers C que s'hagin de compilar. El codi font ha d'estar comentat: és necessari comentar com a mínim les funcions que hi ha al codi. No cal comentar cada línia de codi.
- La carpeta `proves` ha d'incloure en el fitxer de text `log.txt` el resultat de l'execució amb el fitxer `llista_prova.cfg` (tal com es comenta al final de la secció 3). Per fer-ho executeu la vostra aplicació així fent servir els 595 fitxers que conté la base de dades:

  ```
  $ ./practica2 llista.cfg 2> log.txt
  ```

  L'aplicació ha d'imprimir per pantalla la tupla `(vegades, paraula)` que correspon a cadascuna de les paraules que s'han trobat i el nombre de vegades total que apareix la paraula a la base de dades. Així:

  ```
  (25, hello)
  (2343, at)
  ...
  ```

A més, també es demana incloure un fitxer anomenat `valgrind.txt` que inclogui el resultat de l'execució amb l'aplicació `valgrind` fent servir només els primers 10 fitxers de la llista de configuració.

```
$ valgrind ./practica2 llista.cfg 2> valgrind.txt
```

- El directori `doc` ha de contenir un document (dues o tres pàgines, en format PDF) explicant el funcionament de l'aplicació, la discussió de les proves realitzades i els problemes obtinguts. En aquest document no s'han d'explicar en detall les funcions o variables utilitzades. Sí que es pot explicar (incloent gràfics) les estructures utilitzades. També podeu incloure resultats sobre les vostres proves. En particular, és interessant que proveu el següent: com afecta el valor de SIZE a `hash.c` al rendiment del programa ? Per què ? Quina és la paraula més llarga que apareix en els fitxers de text ? Quina paraula és ? En quin fitxer apareix ? Quantes paraules diferents hi ha en total ?

La data límit d'entrega d'aquesta pràctica és el **26 d'octubre del 2014**. El codi tindrà un pes d'un 70% (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.) i el document i les proves el 30% restant.