Sistemes Operatius II - Pràctica 2

Setembre del 2017

Aquesta segona pràctica se centra en extreure les paraules d'un fitxer PDF i inserir-les en un arbre binari.

Índex

1 Introducció

En aquesta pràctica es comença a desenvolupar el codi del projecte. Com s'ha comentat a la pràctica anterior, l'objectiu general del projecte pràctic és desenvolupar una aplicació (sense interfície gràfica) que permeti extreure i indexar les paraules contingudes en múltiples fitxers PDF. L'aplicació haurà de llegir cadascun d'aquests fitxers PDF, i extreure i indexar totes les paraules que contenen aquests (pràctica 2 i 3). Per fer l'aplicació més ràpida s'aprofitaran els múltiples processadors que ens ofereixen les màquines del laboratori per tal de paral·lelitzar determinades tasques (pràctica 4 i 5).

En aquesta pràctica ens centrem en exteure les paraules d'un fitxer PDF i en indexar aquestes paraules en un arbre binari balancejat.

2 Les eines a utilitzar

En aquesta secció es descriuen les eines a utilitzar per realizar la pràctica. D'una banda, una eina que permet extreure d'un fitxer PDF les paraules que conté i, d'altra banda, una llibreria C que permet indexar dades en un arbre binari.

2.1 Conversió d'un fitxer PDF a text pla

A les aules teniu instal·lada una aplicació per terminal anomenada `pdftotext` que permet convertir un fitxer PDF a un fitxer de text pla. Per exemple, executeu

```
$ pdftotext fitxer.pdf fitxer.txt
```

Aquesta instrucció converteix el fitxer PDF especificat al primer argument a un fitxer de text pla i ho guarda guarda al fitxer especificat al segon argument.

En aquesta pràctica no es guardarà el resultat de la conversió a cap fitxer de disc, sinó que se li demanará a l'aplicació `pdftotext` que enviï el resultat de la conversió a la sortida estàndard (*stdout*). La nostra aplicació, la que desenvoluparem en aquesta pràctica, capturarà el resultat de la sortida estandard. D'aquesta forma s'evita haver obrir el fitxer de text creat a disc.

2.2 Llibreria C d'arbre binari balancejat

Les paraules extretes amb l'aplicació `pdftotext` s'inseriran en un arbre binari balancejat. Observeu la figura 1. És un exemple d'arbre binari que conté paraules (cadenes de caràcters). Per a cada node s'emmagatzemen a l'esquerra les paraules que lexicogràficament són anteriors a la paraula del node, mentre que a la dreta es troben les paraules lexicogràficament posteriors a la paraula del node.

En aquesta pràctica se us proporciona d'un codi en C d'un arbre binari balancejat que està preparat per indexar valors sencers, no pas cadenes. Caldrà doncs modificar el codi per tal d'adaptar-lo a les necessitats de la pràctica, és a dir, que l'arbre permeti indexar paraules.

3 La pràctica

La pràctica se centra en desenvolupar una aplicació C que tingui com un únic argument a la línia de comandes, el fitxer PDF a processar

```
$ practica2 <fitxer.pdf>
```

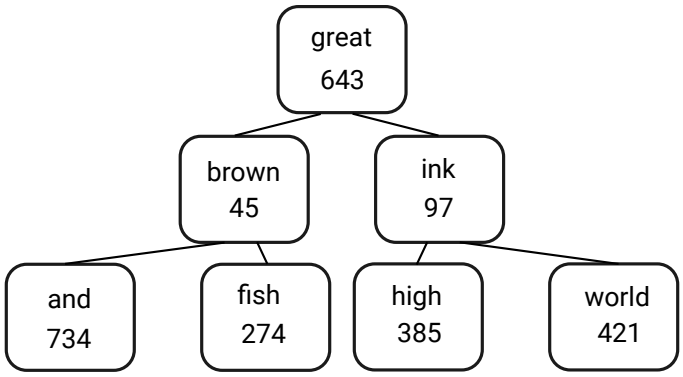

Figura 1: Arbre binari que emmagatzema paraules. Per a cada paraula s'emmagatzema a més el nombre de vegades que ha aparegut al text, veure secció 4.3.

L'aplicació ha de realitzar les següents operacions

1. Extreure les paraules del `fitxer.pdf` especificat a la línia de comandes fent servir l'aplicació `pdftotext`. El resultat de la conversió s'enviarà a la sortida estàndard. L'aplicació `practica2` haurà de fer servir una canonada per capturar-ho, veure secció 6.

2. Extreure les paraules del text capturat, convertir totes les paraules a lletres minúscules, i inserir-les a l'arbre binari. A la secció 3.1 s'indiquen les regles a seguir per extreure les paraules.

3. Un cop creat l'arbre es demana alliberar tota la memòria ubicada i sortir de l'aplicació.

A les següents pràctiques s'ampliarà a la funcionalitat d'aquesta aplicació per permetre a) processar múltiples fitxers PDF, b) emmagatzemar l'arbre creat a disc, c) utilitzar els múltiples processadors de l'ordinador per fer paral·lelisme.

3.1 Extracció de paraules

Observeu a la figura 2 un exemple de text pla a processar. Com es pot veure, el fitxer conté paraules incloent els corresponents signes de puntuació (punt, coma, punt i coma, dos punts, guionet, cometes, signe d'admiració, signe d'exclamació, parèntesi, claudàtors, etc).

L'objectiu al pas d'extracció de paraules és extreure totes les paraules capturades per la canonada excloent els signes d'admiració així com espais o tabuladors. Així de la primera línia del text de la figura 2 "To sing a song that old was sung," l'aplicació ha d'extreure les paraules "To", "sing", "a", "song", "that", "old", "was", "sung". S'hauran de tenir en compte les següents regles per extreure les paraules:

1. Les paraules poden estar separades per espais, tabuladors o altres signes de puntuació. Es considerarà que aquests símbols no formen part de la paraula. És a dir, en trobar la cadena "why?" s'extraurà la paraula vàlida "why" ja que "?" és un signe puntuació. Les paraules unides per guions, com per exemple "taper-light", són paraules vàlides separades "taper" i "light" ja que el guió es considera un signe de puntuació.

```
To sing a song that old was sung,
From ashes ancient Gower is come;
Assuming man's infirmities,
To glad your ear, and please your eyes.
It hath been sung at festivals,
On ember-eves and holy-ales;
And lords and ladies in their lives
Have read it for restoratives:
The purchase is to make men glorious;
Et bonum quo antiquius, eo melius.
If you, born in these latter times,
When wit's more ripe, accept my rhymes,
And that to hear an old man sing
May to your wishes pleasure bring,
I life would wish, and that I might
Waste it for you, like taper-light.
This Antioch, then, Antiochus the Great
Built up, this city, for his chiefest seat;
The fairest in all Syria,
I tell you what mine authors say:
```

Figura 2: Exemple de text pla a processar.

2. L'aplicació no té perquè ser capaç de tractar paraules amb accents. Així, paraules com "Wäts" s'ignoraran. Es recomana no intentar tractar aquests casos ja que les lletres amb aquests tipus de símbols s'emmagatzemen de forma molt particular en un fitxer de text pla i són complicats de processar.

3. Paraules que continguin números o altres símbols que no siguin de puntuació s'ignoraran. Així, per exemple, una paraula com "hello123" s'ignorarà.

Les llibreries C ens proporcionen una sèrie de funcions per saber si és un signe de puntuació, un número, una lletra o altre, vegeu secció 4.2.

Caldrà convertir, de totes les paraules vàlides que s'extreguin, les lletres majúscules a lletres minúscules. Així, la paraula "To" caldrà convertir-la a "to". Vegeu també la secció 4.2 per a les funcions disponibles per fer-ho.

4 Implementació i planificació

Es proposa a continuació una forma de procedir per a la implementació d'aquesta pràctica. Seguiu els passos descrits a les seccions següents.

4.1 Captura la sortida estàndard de l'aplicació `pdftotext`

L'aplicació `pdftotext` és l'aplicació que es farà servir per convertir un fitxer PDF en un fitxer de text pla. Atès que aquí es demana fer servir una canonada, es proposa

1. Consultar el manual de l'aplicació `pdftotext` per saber quin(s) és el paràmetre que s'ha especificar a aquesta aplicació perquè s'imprimeixi el resultat de la conversió a la sortida estàndard (i no pas a un fitxer, com es fa per defecte).

2. Llegiu la secció 6 i modifiqueu el codi d'exemple (figura 3) per executar l'aplicació `pdftotext` amb un fitxer PDF qualsevol i capturar la seva sortida estàndard. Es recomana que aquesta

captura es faci amb la funció *fgets*, funció que heu fet servir a la primera pràctica (no ho feu amb *fscanf*, ja que us pot portar a dificultats a l'hora d'extreure les paraules). Un cop capturades les dades, imprimiu-les (per exemple) a la sortida estàndard.

4.2 Extracció de les paraules

A l'enunciat de la pràctica disposeu, al directori `extraccio-paraules`, d'un exemple que permet analitzar els tipus de caràcter de cada paraula. Es proposa

1. Executeu el codi per veure com es comporta. Proveu amb paraules com "hola123", "Why?" o "Wäts".

2. Modifiqueu el codi perquè es puguin extreure les paraules d'una cadena segons les regles especificades a la secció 3.1.

3. Abans d'inserir una paraula a l'arbre heu de passar tots els seus caràcters a minúscula. Les funcions *isupper* i *islower* permeten saber si una lletra és majúscula o minúscula i les funcions *toupper* i *tolower* permeten convertir una lletra a majúscula o minúscula respectivament.

4. Un cop extretes les paraules podeu, per exemple, imprimir només les paraules vàlides en lletres minúscules per pantalla.

4.3 Inserció de les paraules en un arbre binari

Al directori `arbre-binari` disposeu del codi que implementa un arbre binari balancejat. Els fitxers `red-black-tree.c` i `red-black-tree.h` contenen la implementació de l'arbre, i el fitxer `main.c` conté un exemple d'ús.

La implementació que se us ha proporcionat és una en què l'arbre binari s'indexa per valors sencers i no pas cadenes. Caldrà doncs modificar el codi perquè indexi la informació per cadenes de caràcters. Es proposa doncs

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. Modifiqueu el codi per adaptar-lo a les necessitats d'aquesta pràctica: en comptes d'inserir sencers a l'arbre caldrà inserir paraules (cadenes de caràcters). Per això cal editar els fitxers `red-black-tree.h` i `red-black-tree.c`, modificar les estructures corresponents així com algunes de les funcions associades.

4. Un cop modificat el codi integreu-lo amb els codis desenvolupats a les seccions 4.1 i 4.2. Es demana que cada node de l'arbre emmagatzemi la cadena amb el nombre mínim necessari de bytes per fer-ho. Cada node haurà d'emmagatzemar a més el nombre de vegades que la paraula ha aparegut al text analitzat, veure figura 1.

4.4 Alliberament de la memòria

Un cop creat l'arbre, allibereu tota la memòria associada. Assegureu-vos a més que tot funciona correctament fent servir el `valgrind`. A la primera fitxa del campus teniu informació sobre el funcionament d'aquesta aplicació.

5 Entrega

El fitxer que entregueu s'ha d'anomenar `P2_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 en PDF. Aquí hi ha els detalls per cada directori:

- El directori `doc` ha de contenir un document (tres o quatre pàgines, en format PDF, sense incloure la portada) en què s'inclogui:
  - El codi de la canonada implementat. Comenteu el seu funcionament i com heu aconseguit que l'aplicació pdftotext enviï les dades a la sortida estàndard.
  - Comenteu cadascun dels camps membres de l'estructura del node de l'arbre. Indiqueu quines funcions heu modificat (i com les heu modificat) per tal de permetre que l'arbre binari indexi per cadenes de caràcters.
  - Comenteu quines proves heu fet per assegurar el bon funcionament de l'aplicació. Una forma de fer proves es crear fitxers PDF manualment i comprovar que l'aplicació compta de forma correcta el nombre de vegades que apareixen algunes de les paraules del text.
- La carpeta `src` contindrà el codi font comentat (només fa falta que ho estiguin les funcions). 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 té un pes d'un **70%** (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.). El document i les proves tenen un pes del **30%** restant.

6 Canonades amb *popen* i *pclose*

Suposem la següent instrucció de terminal

```
$ cat fitxer | wc
```

Aquesta instrucció realitza una canonada entre dos processos, en concret entre el procés que executa "`cat fitxer`" i el procés que executa "`wc`". La canonada és una de les formes de comunicació interprocés més senzilles entre dos processos i al terminal s'identifica amb la barra vertical que veieu. La instrucció "`cat fitxer`", per si sola, llegeix el fitxer i l'imprimeix per pantalla (*stdout*, o sortida

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

#define MAXLINE 100

int main(void)
{
  char  line[MAXLINE];
  FILE  *fpin, *fpout;

  fpin = fopen("fitxer", "r");
  if (!fpin) {
    printf("ERROR: no puc obrir fitxer d'entrada.\n");
    exit(EXIT_FAILURE);
  }

  fpout = popen("wc", "w");
  if (!fpout)
  {
    printf("ERROR: no puc crear canonada.\n");
    exit(EXIT_FAILURE);
  }

  while (fgets(line, MAXLINE, fpin) != NULL) {
    if (fputs(line, fpout) == EOF) {
      printf("ERROR: no puc escriure a la canonada.\n");
      exit(EXIT_FAILURE);
    }
  }

  if (pclose(fpout) == -1)
  {
    printf("ERROR: pclose.\n");
    exit(EXIT_FAILURE);
  }

  printf("L'aplicació ha acabat.\n");

  return 0;
}

```

Figura 3: Codi `exemple_popen.c`.

estàndard, segons els conceptes de unix). La instrucció "`wc`", per si sola, és una aplicació que permet comptar el nombre de paraules que s'introdueixen per teclat (*stdin*, o entrada estàndard, segons els conceptes de Unix). Proveu vosaltres mateixos d'executar l'aplicació "`wc`" des de terminal: introduïu-hi algunes paraules que poden estar separades per línies i per acabar polseu Ctrl+D (és equivalent a enviar un end-of-file). L'aplicació "`wc`" us mostrarà per pantalla el nombre de línies, el nombre de paraules i el nombre de caràcters que heu introduït per teclat.

Què fa, doncs, la instrucció "`cat fitxer | wc`" ? Aquesta instrucció connecta (és a dir, comunica) la sortida estàndard de la aplicació de l'esquerra amb l'entrada estàndard de l'aplicació de la dreta. De forma senzilla, és equivalent a obrir l'aplicació "`wc`" i introduir a mà tot el text que hi ha al fitxer. La canonada ens fa la feina per nosaltres.

Les canonades no només es poden fer servir des de terminal, sinó que també es pot programar en C (i altres llenguatges) una canonada per interconnectar dos processos qualssevol. En concret, a la figura 3 se us mostra la implementació de la instrucció que hem executat des de línia de comandes. En aquest programa s'obre el fitxer a llegir (línies 11–25), s'obre la canonada o via de comunicació amb l'aplicació "`wc`" (línies 17—22), i a continuació llegeix el fitxer d'entrada i s'envia

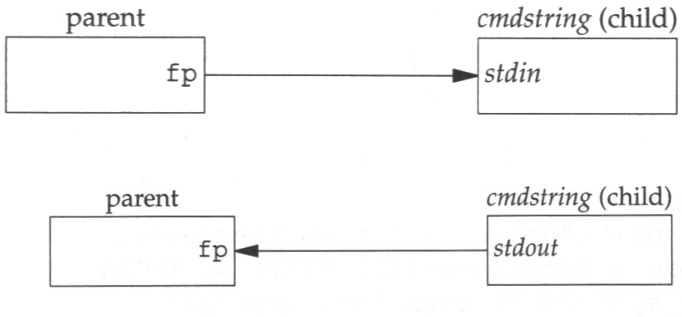

Figura 4: Part superior: resultat d'executar *fp = popen(cmdstring, "w")*. Part inferior: Resultat d'executar *fp = popen(cmdstring, "r")*.

per la canonada al procés "`wc`" (línies 24—29). Un cop acabat, es tanca la canonada (línies 31–35). En tancar la canonada estem enviant un senyal de final de fitxer a "`wc`" cosa que fa que aquest darrer imprimeixi per pantalla el nombre de línies, el nombre de paraules i el nombre de caràcters del fitxer que li ha enviat el primer procés.

En executar l'aplicació surt això per pantalla

```
$ ./exemple_popen
    7    21    166
L'aplicació ha acabat.
```

Fàcil, no ? Observeu que en tot moment fem servir instruccions associades a la lectura i escriptura de fitxers de disc.

Hi ha dues instruccions específiques per manipular canonades: aquestes dues funcions són *popen* i *pclose*. La funció *popen* té aquesta declaració

```
#include <stdio.h>

FILE *popen(char *cmdstring, char *type);
int pclose(FILE *fp);
```

La funció *popen* executa la comanda especificada a *cmdstring* i retorna un punter a fitxer. Anomenarem *procés pare* al procés que executa la instrucció popen i *procés fill* al procés indicat com a argument a la funció popen. En cas que *type* sigui "`w`", tot el que el procés pare escrigui en el fitxer serà enviat a l'entrada estàndard del fill (veure figura 4). En cas que *type* sigui "`r`", la direcció de la canonada va del procés fill al pare. Tot el que el fill escrigui a la sortida estàndard es podrà llegir pel pare mitjançant aquest fitxer (veure figura 4). Una forma senzilla de recordar l'ús de l'argument *type* és que funciona igual que *fopen*: si volem obrir el fitxer per lectura fem servir "`r`", mentre que si volem obrir el fitxer per escriptura fem servir "`w`". Fixeu-vos que a l'exemple de la figura 3 el procés pare obre el procés fill amb mode d'escriptura (per enviar-hi dades).

La funció *pclose* tanca la canonada i espera que el procés especificat a *cmdstring* finalitzi. En cas que hi hagi algun error la funció retorna -1.