Sistemes Operatius II - Pràctica 2

Octubre del 2015

Aquesta segona pràctica se centra en l'extracció de les paraules d'un conjunt de fitxers de text i en el bolcatge d'aquesta informació a l'arbre que heu creat a la pràctica 1. La data d'entrega de la segona pràctica és dilluns 26 d'octubre pel grup F, i dimecres 28 d'octubre pel grup A i C.

Índex

1 Introducció

A la primera pràctica heu llegit un conjunt de paraules de diccionari i les heu inserit a una estructura d'arbre. En aquesta segona pràctica ens centrarem comptar quantes vegades apareixen aquestes paraules de diccionari en un conjunt de fitxers de text pla. Per això caldrà implementar un algorisme que extregui aquestes paraules dels fitxers. Les paraules extretes s'emmagatzemaran primer de tot en una *estructura local*. Cada cop que es finalitzi l'extracció de les paraules d'un fitxer es bolcarà la informació local a l'arbre, que a partir d'ara anomenarem *estructura global*. Només es bolcarà la informació d'aquelles paraules que ja es troben a l'arbre. S'ignoraran totes paraules que es trobin a l'estructura local però no es trobin a l'estructural global. Els fitxers a analitzar són llibres (de text pla) electrònics gratuïts en anglès i provenen del web Gutenberg.

En aquesta pràctica es proveeix del codi C de l'estructura local i d'altres funcions necessàries per tal de facilitar la implementació.

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 secció 2.1.

Per a cada fitxer de text s'hauran d'extreure les paraules i indexar-les (és a dir, inserir-les) en una estructura local. Aquesta estructura local és una estructura que estarà buida en iniciar l'extracció de paraules per a un fitxer. Finalitzada l'extracció de paraules d'un fitxer de text es bolcaran les dades indexades de l'estructura local a una estructura global (l'arbre). Cal tenir en compte que només es bolcaran a l'estructura global la informació de les paraules que es ja es trobin a l'estructura global. És a dir, en cap cas s'afegiran noves paraules a l'arbre, sinó que només es comptarà el nombre de vegades que cada paraula de l'arbre apareix al fitxer.

Es descriuen a continuació amb més detalls el procediment. En concret, es descriuen els blocs de: llista amb els fitxers a processar, extracció de paraules, la indexació local i el bolcat de la informació a l'estructura global.

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 exemple el valor és 595, tot i que cal programar-ho perquè pugui tenir qualsevol valor. A cada línia del fitxer de configuració hi haurà un

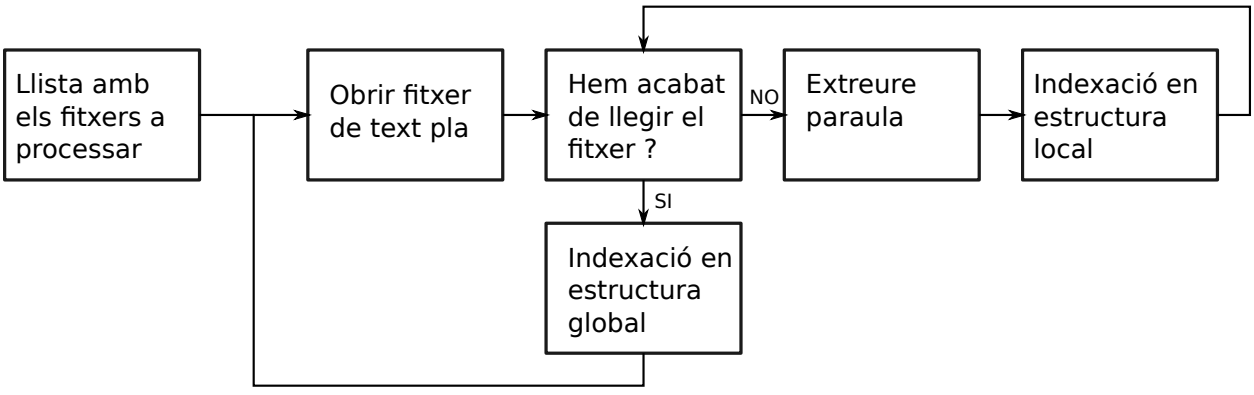

Figura 1: Diagrama de blocs de l'algorisme a implementar en la pràctica 1.

```
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`.

nom de fitxer del qual caldrà extreure les paraules i construir l'estructura d'indexació local. 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 Lectura de dades d'un fitxer

Hi ha múltiples funcions per llegir un fitxer de text pla. Depenent del tipus de dades d'entrada, una funció pot ser millor que una altra per implementar la funcionalitat requerida. En aquesta pràctica s'analitzaran les diverses formes amb què es poden llegir dades d'un fitxer. Recordem que en aquesta pràctica les dades a llegir són fitxers de text pla. Hi ha múltiples formes de llegir les dades un fitxer: byte a byte (mitjançant la funció *fgetc*), línia a línia (mitjançant *fgets* o *fscanf*), o carregar tot el fitxer de cop a memòria amb una única instrucció (fent servir *fread*).

L'objectiu és que comenceu per experimentar quina funció us pot ser més útil per implementar la funcionalitat requerida. A la secció 3.2 se us proposen una sèrie d'experiments per decidir-vos per una determinada funció. A més, a la fitxa 2 teniu una descripció de les funcions que s'acaben d'esmentar.

2.3 Extracció de paraules d'un fitxer

Observeu a la figura 3 un exemple d'un fitxer 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 contingudes en els fitxers de text excloent els signes d'admiració així com espais o tabuladors. Així de la primera línia del text de la figura 3 "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". Les paraules que continguin apòstrofs, com per exemple "wit's", són una única paraula vàlida.

```
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 3: Exemple de fitxer 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. Vegeu secció 3 per a més informació al respecte.

3. Paraules que continguin números o altres símbols s'ignoraran. Així, per exemple, una paraula com "hello123" o "##continue" s'ignoraran.

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 la secció 3.3 per a les funcions disponibles per fer-ho.

2.4 Indexació a una estructura local

Tota paraula vàlida extreta s'indexarà la paraula en una estructura local. Aquesta estructura local emmagatzemarà totes les paraules que apareixen al fitxer de text. L'estructura local és una taula de *hash*, veure figura 4. En altres paraules, la taula de *hash* és un vector de llistes enllaçades: es pot accedir a cada llista enllaçada mitjançant un índex, també anomenat *hash*. Per a cada paraula només caldrà emmagatzemar el nombre de vegades que aquesta apareix al text en qüestió. En la figura 4, 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.

Donada una paraula, l'algorisme de *hash* permet calcular per aquesta paraula un nombre sencer (veure secció 3.4 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 l'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 bolcar 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

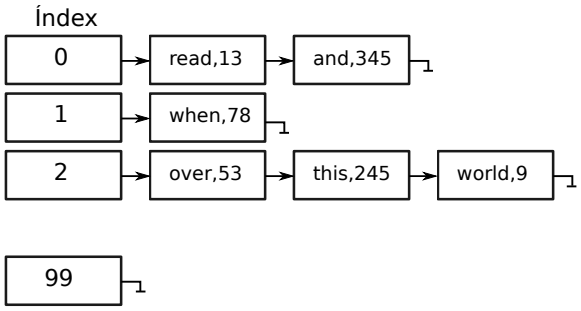

Figura 4: Exemple de taula de *hash* per realitzar la indexació local. En aquesta figura la taula te mida de 100 elements (de 0 a 99). Cada índex de la taula correspon a una llista enllaçada.

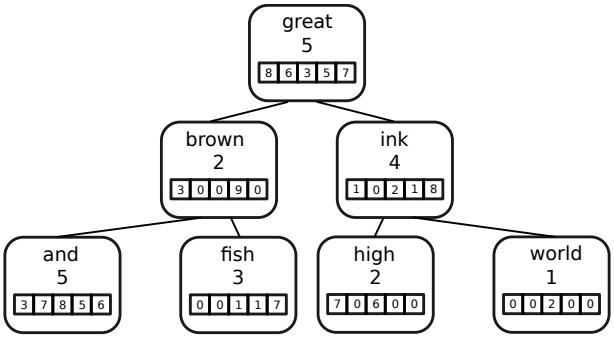

Figura 5: Exemple de l'estructura d'arbre a utilitzar per a realitzar la indexació global.

`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.

2.5 Bolcat a una estructura global

L'estructura local, un cop s'ha processat un fitxer de text, contindrà totes les paraules vàlides extretes del fitxer. Aquesta informació és aleshores bolcada a l'estructura d'arbre. La informació emmagatzemada a l'arbre s'ampliarà respecte la pràctica 1. En aquesta segona pràctica emmagatzemarà 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 analitzat apareix 8 vegades, al segon 6 vegades, al tercer 3 vegades, al quart 5 vegades i al cinquè 7 vegades.

En bolcar la informació de l'estructura local a la global caldrà actualitzar doncs tots els camps necessaris. Cal tenir en compte que només es bolca la informació d'aquelles paraules que prèviament es troben a l'arbre. En cap cas s'inserirà una nova paraula a l'arbre.

3 Implementació i planificació

En aquesta secció es descriuen amb més detall les funcionalitats a implementar. També es comenten tota una sèrie d'experiments a realitzar per entendré bé com s'ha d'implementar l'aplicació. És important seguir una estructura modular atès que la resta de pràctiques es basaran en aquesta segona pràctica. Les seccions que hi ha a continuació són llargues, però el codi a implementar no ho és tant.

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

int main(void)
{
  int c;
  FILE *fp;

  fp = fopen("fitxer.txt", "r");
  if (!fp) {
    printf("No he pogut obrir fitxer.\n");
    exit(1);
  }

  c = fgetc(fp);
  while (!feof(fp)) {
     printf("Caracter: %c (byte %d)\n", c, c);
     c = fgetc(fp);
  }

  fclose(fp);
}

```

Figura 6: Exemple de codi que llegeix un fitxer byte a byte amb *fgetc* (codi `exemple_fgetc.c`).

3.1 Execució de l'aplicació

La segona pràctica s'executarà passat dos arguments per línia de comandes.

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

El primer argument és el fitxer amb les paraules de diccionari (pràctica 1) i el segon argument és el fitxer de configuració. Per facilitar la implementació suposeu que el fitxer de configuració així com tots els fitxers de text pla estan situats en el mateix directori en què s'executa l'aplicació. A l'hora d'entregar no cal que entregueu ni el fitxer de configuració ni els fitxers de text pla.

Tingueu en compte doncs que els passos a realitzar són:

1. L'aplicació haurà de crear primer de tot l'arbre amb les paraules de diccionari words. Correspon a la pràctica 1.

2. Un cop creat l'arbre procediu a llegir els fitxers de text pla i a bolcar la informació a l'arbre, tal com es descriu en aquesta pràctica. No cal imprimir res per pantalla al llarg de tot el processament.

3. En acabar l'aplicació assegureu-vos d'alliberar tota la memòria.

3.2 Lectura de dades d'un fitxer

A la primera pràctica hem vist que una forma de llegir un fitxer de text és mitjançant la funció *fscanf*. Aquesta funció és suficient pels propòsits de la primera pràctica, però pot no ser útil en cas que cada línia del fitxer de text pla contingui un nombre variable de paraules que poden tenir un nombre variable de signes de puntuació. L'objectiu d'aquesta secció és que investigueu i analitzeu altres funcions que us permeten llegir fitxers de text amb l'objectiu d'extreure-ne les paraules.

Anem a veure-les! Per a la implementació d'aquesta pràctica no hi ha realment un procediment que vagi molt millor que altre. Sí que és convenient que sigueu conscients dels possibles problemes amb què us podeu trobar per decidir-vos per una o altra funció.

1. Lectura byte a byte (mitjançant la funció *fgetc*). A l'exemple de la figura 6 se us mostra un exemple d'ús. Teniu un exemple de fitxer de text pla juntament amb el codi font. Compileu i executeu aquest codi. Observeu què és el que surt per pantalla. Observeu que el procediment per llegir el fitxer és: a) llegir el caràcter, b) comprovar si s'ha arribat a final de fitxer i c) imprimir el caràcter per pantalla. Per què cal seguir aquest ordre i no podem fer la comprovació b) abans de fer a)? Per a la aplicació prevista (extreure les paraules), penseu que es tracta d'una bona d'implementar la funcionalitat demanada en

aquesta pràctica? No es convenient que ho penseu en termes d'eficiència ja que això ho veurem a teoria. Penseu-ho simplement a nivell de programació.

2. Línia a línia (mitjançant la funció *fgets*). A l'exemple de la figura 7 se us mostra l'exemple que llegeix línia a línia fent servir *fgets*. Observeu de nou que el procediment per llegir el fitxer és: a) llegir la línia, b) comprovar si s'ha arribat a final de fitxer i c) imprimir la línia per pantalla. En executar el codi, observeu que no apareixen totes les línies del fitxer. Per què? Hi ha alguna cosa al fitxer que fa que la darrera línia no aparegui? Podeu modificar el codi perquè apareguin totes les línies, independentment de com estigui format el fitxer?

   Tal i com es comenta a la fitxa 2, la funció *fgets* llegeix del fitxer fins arribar al caràcter final de línia ('\n') tenint en compte que a tot estirar ha de llegir un determinat nombre de bytes especificat a l'argument de *fgets*, que en aquest exemple és 100. Modifiqueu el codi perquè es llegeixin un màxim de 10 bytes i comproveu el que surt per pantalla en executar el codi.

   Què penseu d'utilitzar *fgets* per implementar la pràctica? Tingueu en compte que, en aquesta pràctica, totes les línies dels fitxers tindran menys de 100 caràcters i que *fgets* inclou el caràcter '\n' a la la cadena de caràcters llegida (ja que cada línia del fitxer d'exemple té una longitud inferior a 100 caràcters). En cas que féssiu servir la funció *fgets* per llegir els fitxers, quines instruccions C faríeu servir per tal "d'eliminar" el caràcter '\n' del final de la cadena? Proveu-ho!

3. Mitjançant la funció *fscanf*. Una forma típica de llegir un fitxer formatat és mitjançant la funció *fscanf*. A la figura 8 se us mostra la forma de procedir. Sembla que *fscanf* està separant "automàticament" el fitxer d'entrada en paraules. Com ho està fent exactament? Per respondre aquesta paraula mireu la documentació d'aquesta funció (`man fscanf`). L'ús de *fscanf* us facilitarà la implementació de l'algorisme d'extracció de paraules respecte les altres instruccions proposades? Comenteu la vostra resposta.

   A l'exemple que se us dóna s'ha suposat que la longitud màxima de la paraula és de 20 caràcters. Què passarà si al fitxer hi ha una paraula amb una longitud superior a 20 caràcters?

4. Finalment, es proposa una solució que determina, abans d'obrir el fitxer, la longitud en bytes d'aquest i llegeix després d'un cop tot el fitxer amb *fread*. La solució és la que es proposa a la figura 9. Què en penseu d'aquesta solució? En el cas hipotètic que el fitxer a llegir sigui molt gran és útil aquesta solució? (Per a la realització d'aquesta pràctica heu de saber que no tindreu fitxers gaire grans a processar)

3.3 Extracció de paraules d'un fitxer

En aquesta pràctica s'ha d'implementar un algorisme que extregui les paraules que hi ha en un fitxer. Per abordar el problema, se us aconsella que comenceu per implementar un algorisme, a ser possible independent del codi que esteu implementant, que permeti imprimir per pantalla totes les paraules que hi ha en una cadena de caràcters introduïda per teclat. Utilitzeu com a punt de partida el codi `exemple-extraccio.c` de què disposeu amb l'enunciat. Se us recomana utilitzar les funcions següents de la llibreria estàndard per processar la cadena: *isalpha* permet saber si un caràcter és una lletra (a-z, A-Z), *isdigit* permet saber si el caràcter és de tipus numèric, *ispunct* permet saber si un caràcter és un signe de puntuació (punt, coma, punt i coma, dos punts, guionet, cometes, signe d'admiració, signe d'exclamació, parèntesi, claudàtors, etc), *isspace* permet saber si un caràcter és un espai, tabulador o retorn de carro. En cas que un caràcter no pertanyi a cap de les anteriors classes podeu suposar que la cadena a la qual pertany el caràcter no és una paraula (això pot passar, per exemple, si el caràcter té un accent). Vegeu la figura 10 que correspon al codi del directori `extraccio-paraules`. Finalment, cal comentar que 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.

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

int main(void)
{
  char str[100];
  FILE *fp;

  fp = fopen("fitxer.txt", "r");
  if (!fp) {
    printf("No he pogut obrir fitxer.\n");
    exit(1);
  }

  fgets(str, 100, fp);
  while (!feof(fp)) {
     printf("Linia: %s\n", str);
     fgets(str, 100, fp);
  }

  fclose(fp);
}

```

Figura 7: Exemple de codi que llegeix un fitxer línia a línia amb *fgets* (codi `exemple_fgets.c`).

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

int main(void)
{
  char str[20];
  FILE *fp;

  fp = fopen("fitxer.txt", "r");
  if (!fp) {
    printf("No he pogut obrir fitxer.\n");
    exit(1);
  }

  fscanf(fp, "%s", str);
  while (!feof(fp)) {
     printf("Dades llegides: %s\n", str);
     fscanf(fp, "%s", str);
  }

  fclose(fp);
}

```

Figura 8: Exemple de codi que llegeix un fitxer amb *fscanf* (codi `exemple_fscanf.c`).

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>

int main(void)
{
  int size;
  char *str;
  FILE *fp;

  struct stat st;
  stat("fitxer.txt", &st);
  size = st.st_size;

  printf("El fitxer te %d bytes!\n", size);

  fp = fopen("fitxer.txt", "r");
  if (!fp) {
    printf("No he pogut obrir fitxer.\n");
    exit(1);
  }

  str =  malloc(sizeof(char) * size);
  fread(str, sizeof(char), size, fp);
  fclose(fp);

  printf("Contingut del fitxer:\n");
  printf("%s", str);

  free(str);
}

```

Figura 9: Exemple de codi que determina la longitud del fitxer i llegeix d'un cop tot el fitxer amb *fread* (codi `exemple_fread.c`).

3.4 Indexació a una estructura local

A la figura 4 es mostra l'estructura de la taula de *hash*. A l'enunciat d'aquesta pràctica disposeu d'una implementació de la taula de *hash* preparada per treballar amb sencers. És dir, en comptes d'inserir paraules a la taula (tal com es mostra a la figura 4) s'insereixen sencers i es compta el nombre de vegades que s'insereix cada sencer la taula. Caldrà doncs modificar aquesta taula de *hash* per adaptar-lo a les necessitats d'aquesta pràctica.

Cada cop que vulgueu inserir una nova paraula a la taula, caldrà calcular el seu valor de *hash*. Aquest valor és un nombre sencer i indica l'índex de la taula en què s'ha d'inserir la paraula. Per exemple, el nombre *hash* associat a la paraula "over" és 2. Amb l'enunciat aquesta pràctica disposeu d'un codi exemple per calcular el valor de *hash* de la paraula, vegeu figura 11.

3.5 Bolcat a una estructura global

Per a la implementació d'aquesta part tingueu en compte les indicacions de la secció 2.5. És important tenir en compte que a l'hora de bolcar la informació no és creen nous nodes a l'arbre. Només es transfereix aquella informació de les paraules que ja es troben presents a l'arbre: el nombre de fitxers diferents en que aquesta paraula apareix i un vector indicant el nombre de vegades que apareix en cadascun dels fitxers. S'ignorarà tota informació de paraules que es troba a l'estructura local però no a la global.

3.6 Proves a realitzar

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. Els professors, a l'hora d'avaluar la pràctica faran servir la base de

```c
#include <stdio.h>
#include <string.h> // per la funcio strlen
#include <ctype.h>  // per les funcions isalpha, isdigit, ...

#define MAXCHAR 100

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

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

  len = strlen(cadena);
  for(i = 0; i < len; i++)
  {
    printf("Posicio: %02i, caracter: %c, es de tipus: ", i, (char) cadena[i]);

    if (isalpha(cadena[i]))
      printf("lletra");
    else if (isdigit(cadena[i]))
      printf("nombre");
    else if (ispunct(cadena[i]))
      printf("signe de puntuacio");
    else if (isspace(cadena[i]))
      printf("espai o retorn de carro");
    else
      printf("no se sap");

    printf("\n");
  }
}
```

Figura 10: Exemple de funcionament de les funcions C isalpha, isdigit, ispunct i isspace.

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

#define MAXCHAR 100
#define SIZE    100

int main(void)
{
  unsigned 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 11: Exemple de càlcul de la funció de *hash* per a una paraula o cadena de caràcters.

dades de Gutenberg fent servir un valor diferent de 595 i amb el valgrind. És important doncs que el codi funcioni posant un valor diferent de 595 a la primera línia del fitxer de configuració.

- 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 words llista.cfg
  ```

4 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 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`. 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.

- 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. Vegeu en particular les seccions 3.2 i 3.6. Utilitzeu aquestes preguntes com a fil per tal de realitzar el vostre document. Intenteu no respondre a les qüestions directament, com si fos un examen. En aquest document no s'han d'explicar en detall les funcions o variables utilitzades.

La data límit d'entrega d'aquesta pràctica és dilluns 26 d'octubre pel grup F, i dimecres 28 d'octubre pel grup A i C. 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.

5 Alguns detalls sobre la pràctica 3 i 4

Esmentarem a continuació alguns detalls sobre la pràctica 3 i 4 per tal que pugueu preveure què fareu a les properes pràctiques:

- La pràctica 3 se centrarà principalment en emmagatzemar tota la informacó de l'arbre en un fitxer. És important doncs que la pràctica 2 us funcioni correctament.

- La pràctica 4 se centra en la implementació multifil de la funcionalitat implementada a la pràctica 2. Per tal de facilitar la implementació de la pràctica 4 és important, però, que el codi estigui modularitzat de forma que a) hi hagi una funció que llegeixi les paraules de diccionari i les insereixi a l'arbre, b) hi hagi una funció que, per a un fitxer de text determinat, extregui les paraules i les insereixi a l'estructura local, c) hi hagi una funció que bolqui la informació a l'estructura local, d) hi hagi una funció principal que cridi a les funciona i faci les iteracions sobre b i c.