Sistemes Operatius II - Pràctica 1

Setembre 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 **5 d'octubre del 2014**.

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 les paraules contingudes en múltiples fitxers de text pla. Els fitxers de text són proporcionats pel professor tot i que també es poden baixar lliurement d'Internet. Aquests fitxers de text són llibres electrònics gratuïts en anglès i provenen del web Gutenberg (http://www.gutenberg.org/). Tingueu en compte que hi ha centenars d'aquests fitxers. L'aplicació haurà de llegir cadascun d'aquests fitxers de text, extreure i indexar totes les paraules que contenen aquests (pràctica 1 i 2). Un cop s'ha obtingut aquesta informació s'haurà d'analitzar i visualitzar determinades parts de la informació extreta (pràctica 3). Finalment, 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).

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

- Pràctica 1: manipulació de cadenes de caràcters i lectura de dades de fitxers (data límit d'entrega: 5 d'octubre)
- Pràctica 2: punters i estructures de dades (data límit d'entrega: 26 d'octubre).
- Pràctica 3: comunicació interprocés mitjançant canonades (data límit d'entrega: 16 de novembre).
- Pràctica 4: programació multifil (data límit d'entrega: 21 de desembre)

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 que puguin ser ampliades de forma fàcil (més endavant es donaran algunes idees per aconseguir-ho).

2 La pràctica

L'objectiu de la primera pràctica és aprendre a manipular cadenes de caràcters i a llegir dades de fitxers. Específicament, l'objectiu d'aquesta primera pràctica és:

1. Aprendre les diverses tècniques amb què es poden llegir dades d'un fitxer. Es pretén que analitzeu les avantatges i desavantatges de cadascuna de les tècniques per tal d'escollir la que millor us convingui per a la implementació de la pràctica.

2. Implementar un algorisme que permeti extreure les paraules que hi ha en un únic fitxer de text pla (el problema d'analitzar múltiples fitxers de text es deixarà per a la segona pràctica).

Els dos punts anteriors es descriuen a la següents subseccions

2.1 Lectura de dades d'un fitxer

La lectura de dades d'un fitxer és un tema delicat ja que hi ha múltiples funcions que permeten fer-ho. Depenent del tipus de dades d'entrada, una funció pot ser millor que una altra per implementar la funcionalitat requerida (en aquesta pràctica, extreure les paraules d'un fitxer). 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 doncs 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*). A la fitxa 2 teniu una descripció de les funcions que s'acaben d'esmentar i a la secció 3 es detalla la forma de procedir en cadascun dels casos.

2.2 Extracció de paraules d'un fitxer

Observeu a la figura 1 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 es 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 1 "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

```
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 1: Exemple de fitxer de text pla a processar.

de la paraula. Les paraules unides per guions, com per exemple "taper-light", són paraules separades: "taper" i "light". Les paraules que continguin apòstrofs, com per exemple "wit's", són una única paraula.

2. Si la paraula conté nombres o altres símbols no es considerarà la paraula. Així, ni el nombre "414" ni tampoc la cadena "FN212" es consideraran una paraula.

3. 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.

4. Queda a la selecció de l'alumne decidir què passa amb cadenes que contenen adreces de correu, enllaços web, o altres tipus de cadenes similars. S'aconsella optar per la solució que es cregui més senzilla: es pot optar per decidir que es tracta d'una única paraula, descartar-la completament, o bé extreure algunes de les subparaules. Per exemple, amb "lluis.garrido@ub.edu" podem considerar la cadena com una única paraula, descartar la cadena completament o bé considerar per exemple les cadenes "lluis", "garrido", "ub" i "edu" com a paraules).

5. En cas que la paraula contingui alguna lletra en majúscula es convertirà a minúscula.

A la secció d'implementació trobareu alguns detalls per tal de procedir en la implementació d'aquesta part.

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 i manipulació de cadenes de caràcters). És particularment important la secció "Manipulació de cadenes caràcters". Llegiu també la Fitxa 2 (funcions bàsiques per llegir i escriure dades de fitxers de disc).

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 la implementació d'aquesta pràctica.

3.1 Lectura de dades d'un fitxer

A la secció anterior s'han comentat les diverses formes que hi ha per llegir un fitxer de text pla. A continuació es detallen i per a cadascuna d'aquestes es realitzen algunes preguntes a investigar:

1. Lectura byte a byte (mitjançant la funció *fgetc*). A l'exemple de la figura 2 se us mostra un exemple d'ús. Compileu i executeu aquest codi. Teniu un exemple de fitxer de text pla juntament amb el codi font. Observeu què és el que surt per pantalla. En executar apareix, de tant en tant, el caràcter amb codi ASCII 10. A quin caràcter correspon ? 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 forma de llegir el fitxer ? És una forma eficient d'abordar el problema ?

2. Línia a línia (mitjançant la funció *fgets*). A l'exemple de la figura 3 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

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

del fitxer. Sou capaços de modificar el codi font perquè apareguin totes les línies però sense modificar el fitxer ?

Tingueu en compte que, 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. El que surt per pantalla és el que esperàveu ?

Què penseu d'utilitzar *fgets* per llegir el fitxer ? En el cas de llegir 100 bytes cada cop, tingueu en compte que *fgets* inclou el caràcter '\n' a la la cadena de caràcters (ja que cada línia del fitxer d'exemple té una longitud inferior a 100 caràcters). Quines instruccions C faríeu servir per tal "d'eliminar" el càracter '\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 4 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 solucions proposades ? Raoneu la vostra

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

resposta.

A l'exemple 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 5. 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)

3.2 Extracció de paraules

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 que permeti imprimir per pantalla totes les paraules que hi ha en una cadena de caràcters introduïda per teclat. Utilitzeu 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, claudà-

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

tors, 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 6 que correspon al codi `exemple1.c`. 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.

4 Entrega

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

- El directori `doc` ha de contenir un document (tres o quatre pàgines, en format PDF, sense incloure la portada) explicant:

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

- Les proves realitzades a la secció 3.1. Es demana escriure un discurs coherent: plantegeu cadascuna de les solucions que heu provat (com si no s'haguessin proposat en aquest enunciat), comenteu les proves que heu realitzat i incloeu la resposta a les preguntes que es fan a la secció 3.1. Comenteu sobre les avantatges i/o desavantatges de fer servir cadascuna de les solucions dintre del context de la pràctica. Sou lliures de comentar altres aspectes que cregueu rellevants. Finalment, comenteu per quina solució heu optat per implementar la pràctica. Tingueu en compte que no hi ha realment una solució molt millor que altra. Es tracta que els comentaris que escriviu siguin raonats.
- Comenteu també el funcionament de l'aplicació d'extracció de paraules, 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. Expliqueu només el funcionament a grans trets. Incloeu, per exemple, resultats d'execució de la vostra aplicació indicant quines cadenes han estat validades com a

```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 6: Exemple de funcionament de les funcions C isalpha, isdigit, ispunct i isspace.

paraules i quines no.

- La carpeta `src` contindrà el codi font comentat (com a mínim 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` (vegeu directori `makefile` per a un exemple). Editeu el fitxer `Makefile` en cas que necessiteu afegir fitxers C que s'hagin de compilar.

  L'aplicació haurà de permetre indicar a la línia de comandes el fitxer a processar. La sortida de l'aplicació hauria de ser similar a la mostrada a sota (observeu que, tal i com es demana a la secció 2.2, les paraules vàlides apareguin amb les lletres en minúscula).

```
$ ./programa fitxer.txt
Paraula valida: hola
Paraula valida: good
Paraula valida: good
Paraula valida: good
Paraula valida: good's
...
Paraula no valida: good@ub.edu
Paraula no valida: ästrid
...
```

- La carpeta `proves` ha d'incloure els següents dos fitxers: el fitxer `log.txt` ha de contenir el resultat de l'execució de l'aplicació amb el `fitxer.txt` inclòs a la pràctica.

  ```
  $ practica1 fitxer.txt 2> log.txt
  ```

  El fitxer de text `valgrind.txt` ha de contenir el resultat de l'execució amb el `valgrind`:

  ```
  $ valgrind practica1 fitxer.txt 2> valgrind.txt
  ```

La data límit d'entrega d'aquesta pràctica és el **5 d'octubre del 2014**. El codi té un pes d'un **50%** (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 **50%** restant.