

Sistemes Operatius II - Pràctica 3

Novembre del 2014

La tercera pràctica de sistemes operatius 2 se centra en en la persistència de l'estructura de l'arbre i en la utilització de les canonades. La data d'entrega de la pràctica és el **dimecres 19 de novembre del 2014**.

Índex

1	Introducció	2
2	La pràctica	2
2.1	Interfície de menú	2
2.2	Emmagatzemament de l'arbre a disc	3
2.3	Gràfica de l'histograma de la longitud de les paraules	3
3	Implementació i planificació	3
4	Entrega	4
5	Gràfiques amb gnuplot	5
5.1	L'aplicació <i>gnuplot</i>	5
5.2	Gràfiques en dues dimensions	6
6	Canonades amb <i>popen</i> i <i>pclose</i>	8

1 Introducció

La segona pràctica s'ha concentrat en utilitzar taules de hash i estructures d'arbre per tal d'indexar les paraules que apareixen en fitxers de text. Aquesta segona pràctica es basa en el codi desenvolupat a la primera pràctica, i se centra en la persistència de l'estructura de l'arbre i la manipulació de l'estructura de dades de l'arbre, la comunicació interprocés mitjançant canonades. A les següents seccions es detallen cadascun dels anteriors punts.

2 La pràctica

La segona pràctica se centra en

- Desenvolupar una interfície de menú senzilla que permeti que l'usuari interactui amb l'aplicació, veure secció 2.1.
- La persistència de l'arbre. Per tal d'evitar haver de crear l'arbre cada cop que s'inicialitza l'aplicació, es proposa poder emmagatzemar i carregar l'estructura d'arbre a/de disc quan l'usuari vulgui, veure secció 2.2.
- La comunicació interprocés mitjançant canonades. L'objectiu és analitzar la longitud de les paraules que hi ha a l'arbre i realitzar una gràfica mitjançant un programari extern, veure secció 2.3.

Es descriuen a continuació cadascun dels anteriors punts.

2.1 Interfície de menú

Per tal de facilitar la interacció de l'usuari amb l'aplicació, es proposa desenvolupar una interfície de menú textual. El menú ha d'incloure les següents 5 opcions:

1. Creació de l'arbre. La creació de l'arbre correspon a la pràctica 2. En seleccionar l'usuari aquesta opció, l'aplicació haurà de demanar per teclat el fitxer de configuració que conté la llista de fitxers a analitzar. Un cop introduïda aquesta informació, l'aplicació crearà l'arbre i en finalitzar tornarà a mostrar el menú amb totes les opcions.
2. Emmagatzemament de l'arbre. En seleccionar l'usuari aquesta opció, l'aplicació haurà de demanar per teclat el nom del fitxer on es desarà l'arbre. A continuació l'aplicació desarà en aquest fitxer la informació dels nodes de l'arbre mitjançant funcions d'entrada/sortida no formatada (per reduir al màxim l'espai de disc necessari). Veure secció 2.2 per entendre com emmagatzemar l'arbre.
3. Lectura de l'arbre. En seleccionar l'usuari aquesta opció, l'aplicació haurà de demanar per teclat el nom del fitxer amb l'arbre a llegir. En cas que hi hagi un arbre carregat a memòria, caldrà alliberar-lo. A continuació

l'aplicació llegirà del fitxer la informació de cada node i la inserirà dintre de l'estructura de l'arbre. Veure secció 2.2.

4. Gràfica de l'histograma de la longitud de les paraules l'arbre. Aquesta gràfica es realitzarà mitjançant una canonada amb una aplicació externa, vegeu 2.3.
5. Sortida. Mitjançant aquesta opció s'allibera tota la memòria reservada i se surt de l'aplicació.

2.2 Emmagatzemament de l'arbre a disc

L'emmagatzemament de l'arbre a disc evita haver de crear l'arbre cada cop que s'engegui l'aplicació. Mitjançant el menú es podrà decidir si es vol desar o carregar la informació de l'arbre a disc.

La informació es desarà mitjançant la tècnica no formatada, vegeu fitxa 3, per reduir així l'espai necessari. Només cal desar-hi la informació emmagatzemada a cada node. No cal desar-hi la informació associada a l'estructura de l'arbre (pares, fills o germans de cada node).

En llegir la informació de l'arbre es llegirà doncs només la informació de cada node. Aquesta informació s'inserirà a l'arbre mitjançant la funció `insertNode`. És possible que l'estructura de l'arbre que es creï en carregar un arbre de disc sigui diferent de l'estructura que hi havia en el moment de desar-lo. Això no importa per la realització d'aquesta pràctica.

2.3 Gràfica de l'histograma de la longitud de les paraules

L'objectiu és dibuixar a una gràfica la probabilitat d'aparició de les paraules als textos segons el nombre de lletres que tenen. Es tracta d'analitzar globalment l'arbre i calcular la probabilitat d'aparició de les paraules d'una lletra, de dues lletres, de tres lletres, i així successivament fins a la paraula que més lletres té. Un cop realitzat aquest càlcul, es dibuixarà la gràfica a pantalla mitjançant l'aplicació *gnuplot* (llegeix 3 i 5 per a més informació al respecte).

El control de l'aplicació *gnuplot* es realitzarà des de la vostra aplicació mitjançant una canonada (veure secció 6). Per tal de dibuixar una gràfica a pantalla cal emmagatzemar a disc les dades a dibuixar i després enviar al *gnuplot*, mitjançant la canonada, la instrucció per dibuixar la gràfica. Veure secció 5 per una explicació del format en què han d'estar les dades i la instrucció que permet dibuixar la gràfica.

3 Implementació i planificació

Aquesta secció dóna alguns consells per tal d'implementar correctament aquesta pràctica. Es recomana implementar la pràctica seguint el següent ordre:

1. Implementació del menú textual. Comenceu per implementar el menú textual amb les opcions comentades a 2.1. Atès que en aquest moment

disposeu del codi de la pràctica 1, només podreu activar les opcions de menú 1 i 5. La resta de les opcions les implementareu al llarg d'aquesta pràctica. Per tal de capturar una lletra del teclat podeu utilitzar la funció *fgetc*.

2. Implementació de les funcions per desar i carregar la informació de l'arbre a disc. L'arbre es desarà en un únic fitxer i tal com s'ha comentat abans no cal desar-hi l'estructura de l'arbre (relació de pares, fills o germans). Només cal desar la informació associada a cada node. Per la implementació es recomana fer servir la instrucció *fwrite* (veure fitxa 2), que permet escriure dades no formatades a un fitxer. Caldrà que implementeu una funció que recorri tot l'arbre: baseu-vos en la funció que esborra l'arbre per implementar-ho. De la mateixa forma, a l'hora de llegir es recomana fer servir la instrucció *fread* (veure fitxa 2), que permet llegir dades no formatades de disc.

Observeu que la paraula associada al node té una mida variable: per tal de desar una cadena a disc es recomana desar primer la longitud de la cadena (funció *strlen*) com a un sencer seguit de la cadena en sí. En llegir una cadena llegireu primer la longitud de la cadena (un sencer) seguit de la cadena en sí. Això facilita la gestió de la lectura i escriptura de la cadena. Un cop hagueu implementat l'escriptura i lectura de l'arbre de disc, assegureu-vos que funciona correctament fent servir el *valgrind*.

3. Feu-vos un petit exemple C completament independent de la pràctica en què un procés pare es comunica amb un procés fill, el *gnuplot*, per dibuixar una gràfica qualsevol (per exemple *grafica2d.data* esmentat a la secció 5.2). L'objectiu aquí es aprendre a fer servir canonades i assegurar-se que el codi implementat funciona abans d'introduir-lo al codi de la pràctica.
4. Implementació de la funció que permet dibuixar l'histograma de la longitud de les paraules de l'arbre. Caldrà que implementeu una funció que recorri tot l'arbre: baseu-vos en la funció que esborra l'arbre per implementar-ho. Un cop calculades les dades a dibuixar caldrà guardar aquestes dades a disc i enviar a *gnuplot* la instrucció per dibuixar-les.

4 Entrega

El fitxer que entregueu s'ha d'anomenar **P3_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.

- La carpeta **proves** ha d'incloure l'arbre que creat per la vostra aplicació així com qualsevol altra informació que considereu rellevant.
- 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.

La data límit d'entrega d'aquesta pràctica és el **dimecres 19 de novembre 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.

5 Gràfiques amb gnuplot

L'objectiu d'aquesta secció és donar la mínima informació necessària de l'aplicació *gnuplot* perquè es pugui realitzar la pràctica.

5.1 L'aplicació *gnuplot*

L'aplicació *gnuplot* és una aplicació, controlada mitjançant línia de comandes, que permet dibuixar gràfiques de molts diversos tipus. Aquesta aplicació va ser creada originalment perquè investigadors i estudiants poguessin visualitzar dades i funcions matemàtiques de forma interactiva, però ha crescut i ara suporta una sèrie d'aplicacions no interactives com per exemple scripts web.

Gnuplot suporta diversos tipus de gràfiques en 2D o 3D. Pot dibuixar amb línies, punts, contorns, camps de vectors, superfícies així com altres tipus de gràfiques. A més, es capaç d'exportar la gràfica dibuixada en pantalla en diversos formats (bmp, jpg, png, svg, etc).

En aquesta pràctica ens centrarem en la funcionalitat bàsica de què disposa aquesta aplicació. L'objectiu és, tal com s'ha dit anteriorment, fer servir *gnuplot* com a eina per dibuixar gràfiques en dues dimensions.

Per executar *gnuplot* des del terminal només hem d'introduir la instrucció *gnuplot*.

```
$ gnuplot
G N U P L O T
Version 4.4 patchlevel 2
last modified Wed Sep 22 12:10:34 PDT 2010
System: Linux 3.0.4-43-desktop
```

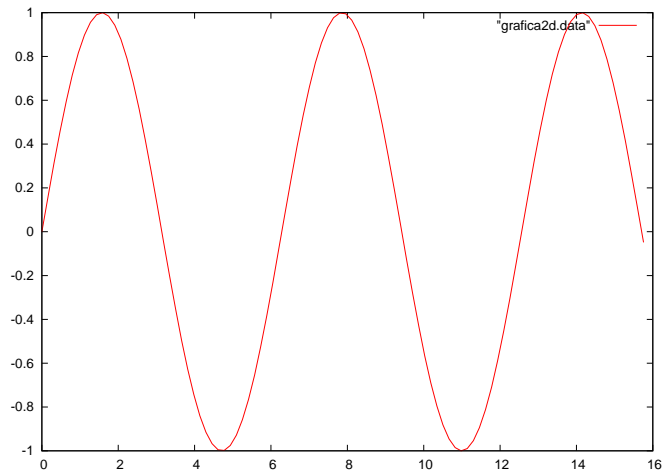


Figura 1: Gràfica 2D generada mitjançant el *gnuplot*.

Copyright (C) 1986-1993, 1998, 2004, 2007-2010
Thomas Williams, Colin Kelley and many others

```
gnuplot home:      http://www.gnuplot.info
faq, bugs, etc:    type "help seeking-assistance"
immediate help:    type "help"
plot window:       hit 'h'
```

```
Terminal type set to 'wxt'
gnuplot>
```

Observeu que aquesta aplicació es controla des de teclat; no hi ha cap interfície gràfica.

5.2 Gràfiques en dues dimensions

Una forma senzilla de realitzar gràfiques amb *gnuplot* (i que serà la que s'utilitzarà en aquesta pràctica) és mitjançant fitxers formatats que contenen les dades a dibuixar. Dintre del ZIP de l'enunciat de pràctica 2 hi ha un directori anomenat *gnuplot* que conté fitxers de text amb dades per dibuixar. Observeu que podeu visualitzar qualsevol dels fitxers mitjançant un editor de text qualsevol (*kate*, *kurite*, *vi*, etc). Centrem-nos en el fitxer **grafica2d.data**: aquest fitxer conté els valors de la funció $f(x) = \sin(x)$ avaluats per a diversos valors d' x . A la primera columna hi apareix el valor d' x , mentre que a la segona hi apareix

el valor d' $f(x)$ avaluat per al valor corresponent d' x que apareix a la primera columna.

Per dibuixar la gràfica fent servir el *gnuplot* es pot fer servir aquesta instrucció

```
gnuplot> plot "grafica2d.data" with lines
```

A la Figura 1 podeu veure el resultat esperat. Podem també dibuixar la gràfica amb punts o impulsos.

```
gnuplot> plot "grafica2d.data" with dots
gnuplot> plot "grafica2d.data" with points
gnuplot> plot "grafica2d.data" with impulses
```

Inclús també podem superposar una gràfica sobre una altra. Veiem un exemple on dibuixem la funció $f(x) = \sin(x)$ (fitxer *grafica2d.data*) superposat amb la funció $f(x) = \sin(x^2)$ (fitxer *grafica2d_2.data*).

```
gnuplot> plot "grafica2d.data" with lines, "grafica2d_2.data" with lines
```

Observeu que els dos fitxers tenen mides diferents. Això és degut a que al segon fitxer hi ha més punts avaluats que al primer, però no implica cap problema per a *gnuplot*.

Per sortir de l'aplicació podem fer servir la instrucció *quit*.

```
gnuplot> quit
$
```

Finalment, en cas que vulguem exportar la gràfica a un fitxer també tenim les eines necessàries per fer-ho. Hem d'introduir aquestes instruccions per guardar el fitxer en format SVG (Scalable Vector Graphics):

```
gnuplot> set term svg
gnuplot> set out "file.svg"
gnuplot> plot "grafica2d.data" with lines
gnuplot> quit
$ ls -l file.svg
-rw-r--r-- 1 lluis users 89550 28 set 12:34 file.svg
```

A la figura 1 s'ha inserit el fitxer generat per *gnuplot*.

El fitxer SVG es pot editar després amb alguna aplicació de dibuix (en el cas de Linux, una aplicació molt bona és *inkscape*). També es pot exportar directament en un format d'imatge tipus PNG o JPEG però la qualitat de la imatge resultant no serà mai tan bona com amb el SVG.

Per a més informació es recomana veure la secció de demos de la pàgina web de l'aplicació, <http://www.gnuplot.info>.

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 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 2 se us mostra la implementació de la instrucció que hem vist a l’inici d’aquesta secció. 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 per la canonada al procés “*wc*” (línies 24–29). Un cop acabat, 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. Però hi ha dues instruccions específiques per manipular canonades: aquestes dues funcions són *popen* i *pclose*. La funció *popen* té aquesta declaració

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



```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define MAXLINE 100
5
6 int main(void)
7 {
8     char line[MAXLINE];
9     FILE *fpin, *fpout;
10
11     fpin = fopen("fitxer", "r");
12     if (!fpin) {
13         printf("ERROR: no puc obrir fitxer d'entrada.\n");
14         exit(EXIT_FAILURE);
15     }
16
17     fpout = popen("wc", "w");
18     if (!fpout)
19     {
20         printf("ERROR: no puc crear canonada.\n");
21         exit(EXIT_FAILURE);
22     }
23
24     while (fgets(line, MAXLINE, fpin) != NULL) {
25         if (fputs(line, fpout) == EOF) {
26             printf("ERROR: no puc escriure a la canonada.\n");
27             exit(EXIT_FAILURE);
28         }
29     }
30
31     if (pclose(fpout) == -1)
32     {
33         printf("ERROR: pclose.\n");
34         exit(EXIT_FAILURE);
35     }
36
37     printf("L'aplicació ha acabat.\n");
38
39     return 0;
40 }
41

```

Figura 2: Codi exemple_popen.c.

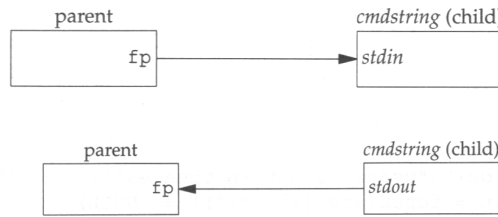


Figura 3: Part superior: resultat d'executar $fp = popen(cmdstring, "w")$. Part inferior: Resultat d'executar $fp = popen(cmdstring, "r")$.

```
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 (no entrarem en detalls aquí del per què) *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 escriu en el fitxer serà enviat a l'entrada estàndard del fill (veure figura 3). En cas que *type* sigui “r”, la direcció de la canonada va del procés fill al pare. Tot el que el fill escriu a la sortida estàndard es podrà llegir pel pare mitjançant aquest fitxer (veure figura 3). 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 2 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 .

Internament, les funcions *popen* i *pclose* fan moltes coses. Si esteu interessats en saber què fan mireu la fitxa 3 penjada al campus. Observeu a més les comandes *popen* i *pclose* serveixen per establir i tancar, respectivament, una **comunicació unidireccional** entre dos processos. És a dir, que podem enviar informació d'un primer procés a un segon procés. No s'estableix cap comunicació bidireccional, és a dir, en el dos sentits. Si volguéssim establir una comunicació bidireccional caldria implementar-ho manualment fent servir les instruccions *pipe* i *fork*, vegeu la fitxa 3.

En el context de la segona pràctica haureu de realitzar la connexió amb l'aplicació *gnuplot* amb una canonada. Per dibuixar una gràfica des de la vostra aplicació haureu de guardar a disc les dades a dibuixar (vegeu secció 5 per informació sobre el format a generar). Un cop heu generat les dades, heu d'enviar al *gnuplot* (mitjançant la canonada) la comanda necessària perquè aquest dibuixi la gràfica.

Hi ha un detall important a tenir en compte a l'hora d'implementar aquesta funcionalitat a la vostra pràctica. Tal com s'ha comentat a teoria, recordeu

que l'estructura `FILE` utilitzada per la canonada inclou, entre altres membres, un buffer d'usuari. En utilitzar les instruccions *fprintf*, *fputs*, etc per escriure a la canonada tingueu en compte que les dades a escriure es guarden primer a buffer d'usuari. Quan el buffer d'usuari és ple es realitza realment l'operació d'escriptura a la canonada. És a dir, si el procés pare escriu a la canonada alguna informació no us hauria d'estranyar si aquesta informació no es rebí de forma immediata pel procés fill. Recordeu que el llenguatge C ens ofereix una solució en cas que vulgueu escriure (o transmetre) la informació de forma immediata: la instrucció *fflush* (mireu el manual per veure com funciona). Aquesta instrucció fa que es buidi al buffer d'usuari i s'escriguin les dades al fitxer.