Sistemes Operatius II - Pràctica 2

Octubre del 2013

La segona pràctica de sistemes operatius 2 se centra en la comunicació interprocés mitjançant canonades i en la persistència de l'estructura de l'arbre. La data d'entrega de la primera pràctica és el diumenge **10 de novembre del 2013**.

Índex

1 Introducció

La primera 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 manipulació de l'estructura de dades de l'arbre, la comunicació interprocés mitjançant canonades i la persistència de l'estructura de l'arbre. 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 interactuï amb l'aplicació, veure secció 2.1.
- La comunicació interprocés mitjançant canonades. L'objectiu és analitzar les paraules que hi ha a l'arbre i realitzar gràfiques mitjançant un programari extern a l'aplicació que es desenvolupa en aquesta pràctica, veure secció 2.2.
- 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 2.3.

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 1. 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 l'arbre mitjançant funcions d'entrada/sortida no formatada (per reduir al màxim l'espai de disc necessari).
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.

4. Anàlisi de les paraules de l'arbre. Aquest punt es descriu a la secció 2.2.

5. Sortida. Mitjançant aquesta opció s'allibera tota la memòria reservada i se surt de l'aplicació.

2.2 Anàlisi de les paraules de l'arbre

L'objectiu de la part d'anàlisi de paraules és analitzar la distribució de les paraules dels textos que hi ha emmagatzemades a l'estructura de l'arbre. En particular, es proposen aquí dos anàlisi diferents. Cada anàlisi dóna lloc a unes dades que es dibuixaran mitjançant un programa extern a l'aplicació (veure detalls més endavant en aquesta secció). Es proposa:

1. Dibuixar la probabilitat d'aparició d'una determinada paraula segons el fitxer. Donada una paraula, buscar la paraula a l'arbre i calcular la probabilitat d'aparició de la paraula en cada fitxer. Per a fer la gràfica independent de l'ordre en què el fitxers estan indicats al fitxer de configuració, es proposa ordenar la probabilitat d'aparició de major a menor abans de dibuixar-la.

2. Dibuixar la probabilitat d'aparició de les paraules als textos segons el nombre de lletres que tenen. L'objectiu és 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.

En aquesta pràctica sou lliures d'implementar altres algorismes d'anàlisi però cal implementar-ne com a mínim dos semblants als proposats aquí.

Les gràfiques es realitzaran mitjançant l'aplicació *gnuplot* (veure secció 5). El control de l'aplicació *gnuplot* es realitzarà mitjançant una canonada (veure secció 6). Per tal de dibuixar una gràfica a pantalla, primer 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.

2.3 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.

En aquesta secció només es vol incidir en el fet que en desar l'arbre no cal desar la informació de pares, fills o germans de cada node. Només cal desar-hi la informació emmagatzemada a cada node. En carregar l'arbre a memòria es llegirà la informació de cada node i s'inserirà a l'arbre mitjançant la funció que hi ha disponible per aquesta tasca. És possible doncs que l'estructura de l'arbre

que es crea en carregar un arbre sigui diferent de l'estructura que hi havia en el moment de desar-lo. Això no importa per la realització d'aquesta pràctica.

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. Abans de començar la pràctica, 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.

2. 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*.

3. 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 l'estructura de l'arbre (relació de pares, fills o germans). Només cal desar la informació associada a cada node. Per fer-ho es recomana fer servir la instrucció *fwrite* (veure fitxa 2), que permet escriure dades no formatades a un fitxer. 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*.

4. Implementació de l'anàlisi de l'arbre. Recordar que hi ha d'haver dues sub-opcions disponibles "Dibuixar la probabilitat d'aparició d'una determinada paraula segons el fitxer" i "Dibuixar la probabilitat d'aparició de les paraules als textos segons el nombre de lletres que tenen". Podeu fer ús de les funcions que hi ha disponibles per a la manipulació de l'arbre. Per a la primera sub-opció es demana ordenar les dades: utilitzeu l'exemple `qsort.c` disponible amb aquest enunciat per inspirar-vos. Per a la segona sub-opció 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 `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 en anglès en un format similar al mostrat als fitxers `red-black-tree.c` i `linked-list.c`. Es necessari comentar com a mínim les funcions que hi ha al codi.

- 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 **10 de novembre del 2013**. 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

        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*, *kwrite*, *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

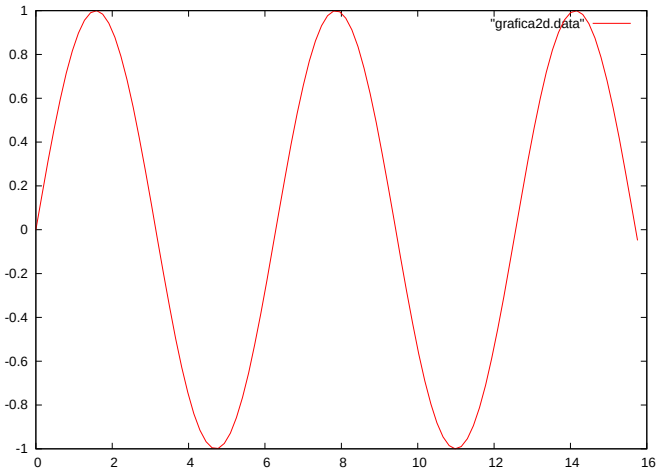

Figura 1: Gràfica 2D generada mitjançant el *gnuplot*.

```
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):

Figura 2: Resultat d'executar *fp = popen(cmdstring, "r")*.

```
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*

Ja que la creació de canonades entre processos (pare i fill) és una operació comuna, la llibreria estàndard I/O ofereix dues funcions que fan tota la feina bruta que s'ha vist a classe de teoria: crear la canonada amb la instrucció *pipe*, crear un fill amb *fork*, tancar els descriptors de fitxers necessaris i executar el programa que desitgem al fill. Aquestes dues funcions són *popen* i *pclose*.

```
#include <stdio.h>

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

La funció *popen* realitza un *fork* i un *exec* per executar la comanda especificada a *cmdstring* (que serà el procés fill) i retorna un punter a fitxer. 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 2). Per altra banda, 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 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`".

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

Figura 3: Resultat d'executar *fp = popen(cmdstring, "w")*.

A la figura 4 mostrem el codi `exemple_popen.c` que fa servir les funcions *popen* i *pclose*. Aquest codi implementa el que per línia de comandes (a un terminal) és

```
$ cat fitxer | less
```

En concret, en aquest programa el pare obre el fitxer (línies 17–21), obre la canonada (línies 23—28), i a continuació llegeix el fitxer d'entrada i ho envia per la canonada (línies 32—37). Un cop acabat, tanca la canonada (línies 39–43).

El codi és molt més curt i senzill que el que cal fent servir les funcions pipe, fork i exec. En concret, aquí tot el procés de creació d'una canonada entre pare i fill es redueix a una única instrucció: *popen*. En el moment de tancar la canonada només cal cridar a *pclose*. Fàcil, no ?

Observeu 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. Per defecte, no s'estableix cap comunicació en sentit invers, és a dir, del segon procés al primer. Si volguéssim establir una comunicació bidireccional caldria implementar-ho manualment fent servir les instruccions pipe i fork.

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 `practica2`, 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, una memòria intermitja al procés d'usuari (un *buffer*). En utilitzar les instruccions *fprintf*, *fputs*, etc per escriure a la canonada tingueu en compte que les dades a escriure es guarden primer a la memòria intermitja. Quan la memòria intermitja és plena es realitza realment l'operació d'escriptura a la canonada. Si el procés pare escriu a la canonada alguna informació no us hauria d'estranyar si aquesta informació no es rebi de forma immediata pel procés fil. 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 la memòria intermitja i s'escriguin les dades al fitxer.

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

#define MAXLINE 100

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

  if (argc != 2)
  {
    printf("us: %s fitxer\n", argv[0]);
    exit(EXIT_FAILURE);
  }

  fpin = fopen(argv[1], "r");
  if (!fpin) {
    printf("ERROR: no puc obrir fitxer d'entrada.\n");
    exit(EXIT_FAILURE);
  }

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

  /*  copiem el contingut de argv[1]
      a la canonada */
  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 4: Codi `exemple_popen.c`.