Sistemes Operatius 2

Memòria pràctica 4

1. ***Funcionament de l'aplicació***

La implementació d'aquesta pràctica es realitza tota en el punt 1 del nostre menú, és a dir, el de "creació de l'arbre". Per tant:

a. L'usuari selecciona aquesta opció al menú. A continuació, escriu el nom del fitxer que conté els noms dels fitxers *pdf* a processar. Es crida la funció *admin-create-tree*. En ella, creem un vector on guardarem el nom de cada fitxer PDF, després de veure, es clar, quants fitxers PDF s'han de processar.

b. Un cop llegida "tota la feina a fer", es crida la funció *crearFils*, amb aquest vector de noms, el número de PDF's i l'arbre. Aquesta funció té l'objectiu de crear tants fils com s'especifiquen i una estructura per cada un d'ells. Aquesta estructura...:

   i. Té el número de PDF's a gestionar
   ii. Té la llista (vector) de PDF's a gestionar
   iii. Guarda el fil que la conté (per imprimir el seu número només)
   iv. Conté l'arbre global i l'arbre local

c. La funció *treeFils,* la qual es crida al crear cada fil nou, té com a missió gestionar el moviment de fils pels fitxers, és a dir, és on s'organitza quin fitxer agafa cada fil. Per tant, s'estableix un comptador local i un *while*, del qual no es surt fins que no quedin fitxers a processar. Mitjançant aquest comptador cada fil sap quin fitxer agafar. Es fa un *mutex* per evitar que s'iteri el comptador al entrar dos fils "molt a la vegada". Un cop gestionat el fitxer a seleccionar, es crida la funció *create_tree* que retorna l'arbre obtingut de processar el fitxer.

d. Tornant a la funció *treeFils*, ara es crida la funció *copiarArbre*, localitzada a *red-black-tree.c*. Aquesta, de manera recursiva, va copiant cada node al arbre gran. Es realitza la copia entre un *mutex_lock* i un *mutex_unlock,* ja que necessitem evitar que dos fils copiïn el seu arbre a la vegada.

e. Seguidament, un cop alliberades les estructures, tornem a la funció *admin_crearTree*. Ara es fa la crida a la funció *esperarfils()*, la qual realitza "l'ajuntament" dels fils fets servir amb la funció *pthread_join*. A continuació, es destrueix el *mutex.*

Abans de la crida a la funció *crearFils*, s'obté la data del sistema, és a dir, es guarda en una variable el "temps" cronològic en aquell moment. També es guarda el *clock* del sistema. En conseqüència, al acabar la crida de la funció *esperarFils*, es crida un altre cop, en variables diferents, la data cronològica i el *clock* del sistema. Amb la resta d'aquestes informacions, imprimim per pantalla el temps (cronològic i de CPU) que ha trigat la nostra aplicació a processar els fitxers PDF.

- És interessant veure com disminueix el temps d'execució com més fils afegim. És lògic: com mes fils tinguem, més "paral·lelament" es realitzarà el processament de fitxers i per tant acabarà de fer la feina mes d'hora.
- És interessant veure la diferència entre fer-ho amb un fil (com a la pràctica 3) i fer-ho amb varis. La optimització es molt gran!

2. ***Proves realitzades***

Per poder comprovar el correcte funcionament de quin fil agafa quin fitxer, hem realitzat un *printf* principal que mostra quin fil ha agafat quin fitxer. D'aquesta manera, veiem "en temps real" quina és la gestió que estan realitzant els diferents fils amb els diferents fitxers PDF.

En segon lloc, hem comprovat que el processament d'aquests fitxes sigui correcte amb la funció 4 de l'aplicació, la de buscar una paraula en l'arbre. Concretament, creàvem l'arbre amb X fils i en segon lloc buscàvem una paraula concreta, de la qual sabíem quin era el número correcte de repeticions. Després, esborràvem l'arbre, n'afegíem un de nou, tornàvem a inserir el nostre i tornàvem a buscar la o les paraules.

Per culpa d'alguns problemes sorgits també hem anat imprimint per pantalla les adreces de memòria d'algun vector, o anàvem mirant si algun arbre (el local o el global) eren NULL o semblants.

3. ***Problemes obtinguts***

a. Un dels majors problemes obtinguts ha sorgit d'un malentès del enunciat de la pràctica. Vam entendre que podíem passar un vector de fitxers a cada fil, i no tot el vector de fitxers per a que cada fil agafes el que volgués. Així doncs, vam haver de fer modificacions al últim moment per poder-ho adaptar correctament. Finalment, gràcies a un comptador global i local, aconseguim que cada fil que entri a la funció agafi un fitxer diferent, sempre i quant hi hagi de disponibles.
b. Un altre problema important, i de fet, un que no hem solucionat, és si hem o no de destruir l'arbre local després de que un fil processi un fitxer. En una primera instància, fèiem *deleteTree* i després *free*, però llavors el programa no funcionava correctament. En canvi, si només fem *free*, no hi ha cap problema. Una de les possibles raons per les que sorgeix aquest problema, potser, es perquè *deleteTree* esborra tots els punters i nodes del arbre, llavors, com que en primer lloc fem una copia del arbre local al global, potser estem esborrant també els punters que té l'arbre global al local, etc.
c. També vam tenir problemes amb el *Makefile* i la funció *gettimeofday*. Per culpa de les màquines virtuals o el canvi d'ordinadors i sistemes operatius, la data de creació o alguna cosa per l'estil d'aquest fitxer no era la correcte, llavors al compilar llença un *warning* de que la data no es correcte, i per tant pot haver-hi alguna mala interpretació a l'hora de calcular el temps de CPU o el temps cronològic.