2018-19

Sistemes Operatius II

PRÀCTICA 4

FRANCISCO DÍAZ RUIZ NIUB: 16828405

JOSÉ BLANCO FERNANDEZ NIUB: 18088022

- **OBJECTIUS:**

Per aquesta pràctica havíem de modificar el codi de la pràctica 3 de tal manera de que fes el mateix però utilitzant fils de processos. Primer havíem d'aplicar un fil secundari a l'hora de llegir el fitxer de dades que després utilitzaríem per omplir els diferents nodes del arbre.

Una vegada ja tenim les dades llegides, ara hem d'utilitzar N fils per introduir les dades llegides dintre dels nodes del arbre, per tal de fer-ho hem hagut d'utilitzar les diferents funcions de *pthread*.

- **IMPLEMENTACIÓ:**

Per tal de realitzar amb èxit les diferents tasques plantejades a resoldre en aquesta pràctica primer de tot havíem d'incloure la llibreria de fils al fitxer <phtread.h>.

Després de incloure la llibreria dels fils, en el fitxer *ficheros-csv.c*, definim el *mutex* que utilitzarem més endavant i l'inicialitzem amb *PTHREAD_MUTEX_INITIALIZER.* En el fitxer *ficheros-csv.h* definim el nombre de threads que el nostre programa utilitzarà (*N_THREADS*) així com el nombre de línies que llegirà per bloc (*NUM_LINES*).

Com demanava la pràctica, el fil principal s'encarrega de crear l'arbre i posar els nodes dels aeroports que llegeix de dintre del fitxer aeroports.csv. Una vegada l'arbre ja està creat ara toca omplir-lo amb les llistes de vols per a cada node/aeroport. En aquest cas ara si que hem d'implementar fils. Mentre es va omplint l'arbre amb els nodes, aquests també tenen un propi *mutex* i cal inicialitzar-ho per a cadascun abans d'inserir-lo dins l'arbre.

Per guardar l'arbre i el fitxer amb el que estem treballant, creem un struct anomenat *param* i llavors, al llarg de la pràctica utilitzarem l'struct creat per accedir al arbre amb el que estem treballant i el fitxer corresponent.

Primer de tot en la funció *read_airports_data*, obrim diferents fils secundaris que llegiran tot el fitxer de dades (ja sigui el dades.csv, el 2008.csv o el big_file.csv) i posarà les línies guardades dintre d'una llista de char (*char fi_lists[NUM_LINES][MAXCHAR]).* Mentre es llegeix el fitxer i s'omple la llista, amb el *mutex* bloquegem qualsevol accés al fitxer o la llista que estem utilitzant mitjançant el *pthread_mutex_lock(&mutex)* i el *pthread_mutex_unlock(&mutex).*

Una vegada ja ha acabat el procés de llegir les dades del fitxer, es comprova que la línia llegida sigui vàlida i posteriorment es crida a una funció que hem creat que s'encarrega d'inserir les llistes de vols dintre de cada node. Primer de tot comprova que l'aeroport/node existeix. En cas afirmatiu, mitjançant el *mutex* del node amb el que treballarem, bloquegem amb el

pthread_mutex_lock(&node_data->mutex) per tal de que altres processos/fils pugin accedir a aquest node i, a continuació, s'omplirà la llista de vol per a aquest aeroport. Una vegada hagi finalitzar això, es desbloquejarà el procés amb el *pthread_mutex_unlock(&node_data->mutex)* i la funció acabarà el seu funcionament.

Per tal de testejar el nostre codi, hem utilitzat diferents fitxers que ens han sigut proporcionats amb l'objectiu de veure quant de temps trigarà el programa en realitzar les lectures i les operacions corresponents depenent del nombre de threads que utilitzem. Primer de tot vam provar amb un sol thread i, depenent del fitxer de dades que volguéssim utilitzar, trigava entre un únic segon o diversos, arribant a un minut pel big_file.csv.

Després vam fer diferents proves amb altres threads i els temps no canviava molt respecte a un únic fil. També vam provar de canviar el nombre de línies que el programa llegirà per a cada bloc i el temps incrementava un mica en cas de que el nombre de línies llegides fos petit, encara que aquest canvi és quasi inapreciable.

- **COMPLICACIONS I OBSERVACIONS:**

Respecte a complicacions, hem trobat moltes complicacions a l'hora de crear els diferents threads que anàvem a utilitzar, així com on els havíem d'utilitzar i com funcionaven exactament.

Com no podíem veure si realment funcionava tot codi que estàvem creant, això va suposar tenir que entregar més tard la pràctica així com no saber com arreglar uns quants errors que tenim respecte al *valgrind*, on aquest en indica que hi han diferents lectures i escriptures de línies amb *size* incorrecte, però quan revisem el codi no aconseguim trobar on succeeix això exactament.

Sospitem que a causa dels errors mencionats hi ha un petit increment en el temps que triga el programa a realitzar la tasca, aquest increment hem vist que ronda aproximadament 1 segon. Encara així, l'arbre es genera adequadament i els nodes tenen tota la informació necessària per realitzar les altres opcions del menú. També cal dir que respecte a l'alliberament de memòria passa net els tests del *valgrind*.