

Pràctica 4

SISTEMES OPERATIUS II

DAVID DE LA PAZ / ORIOL RABASEDA

ÍNDIX

Funcionament del programa	3
Proves realitzades.....	4

Funcionament del programa

En aquesta quarta pràctica tornem a treballar sobre les pràctiques anteriors, centrant-nos en la utilització de fils per tal de processar les dades d'aeroports de forma més ràpida. El main és el mateix menú de la pràctica 3, tot i que ara hem modificat la funció 1 de creació de l'arbre (**database.c**) per tal de crear l'arbre binari utilitzant fils. D'aquesta manera, tindrem un fil secundari per inicialitzar l'arbre d'aeroports i un nombre variable de fils addicionals per processar la informació del fitxer de vols i inserir-la a l'arbre. Haurem d'anar amb compte per tal de bloquejar els fils a l'hora de llegir informació del fitxer principal i per inserir dades als nodes. Anem a veure amb més detall el funcionament de la pràctica:

Per tal d'evitar repetir explicar el funcionament de la pràctica anterior, només ens centrarem en explicar les noves instruccions relacionades amb l'ús dels fils. A l'hora de crear un arbre, comencem per cridar a la funció **airport_tree(*fp)** per tal d'inicialitzar-lo amb el fitxer d'aeroports. La única diferència amb l'altra pràctica és que hem modificat l'estructura dels nodes per afegir un *mutex* individual per cadascun per tal de poder-los bloquejar al modificar els seus vols. Abans d'inserir un node a l'arbre, l'inicialitzem a NULL amb la funció **pthread_mutex_init(&(n_data->mutex))** (com es pot veure, el mutex per cada node està a la struct que defineix el node).

Un cop tenim l'arbre, el següent pas és crear una llista dels threads que utilitzarem per omplir l'arbre amb els diferents vols. També inicialitzem una estructura *thread_data* per poder enviar informació als threads, la qual conté el nom del fitxer de vols i l'arbre en qüestió. A continuació utilitzem un *for* per crear tots els threads a utilitzar amb la funció **pthread_create(&ntid[i], NULL, thread_ini, (void *)t_data)**, la qual va creant els diferents threads de la llista (amb atributs per defecte) utilitzant la funció *thread_ini*, que veurem a continuació amb més detall. També enviem el struct *t_data* per tal de poder enviar informació a cada thread. Un cop creats, utilitzem un altre *for* per cridar per a cada fil la funció **pthread_join(ntid[i], NULL)**, la qual espera a que aquests acabin la seva execució.

Què fa exactament la funció **thread_ini(*args)**? Cada fil utilitzarà un bloc de mida donada per anar llegint grups de N línies del fitxer de vols per posteriorment introduir les dades a l'arbre. Els arguments de la funció són el struct que conté el arbre i el nom del fitxer de vols. Mentre encara quedin línies per llegir, fem malloc per un crear un bloc de mida «BLOCK_SIZE» (variable). A continuació fem un **lock** de *f_mutex*, el qual hem declarat a dalt de tot de manera estàtica amb **PTHREAD_MUTEX_INITIALIZER**. Això ens permet bloquejar el file de vols per lectura, de manera que un bloc pugui llegir N línies seguides. Utilitzant un bucle *while* anem llegint les N línies amb *fgets*, fem el seu malloc corresponent i les anem guardant al bloc amb un **memcpy**. Com que l'últim bloc pot ser de mida menor a la resta, si passa això activem un flag 'eof' per saber que ja hem acabat. Un cop omplert el bloc, desbloquegem *f_mutex* fent **unlock** i introduïm a l'arbre la informació del bloc. D'aquesta manera, cada fil va introduint informació a l'arbre en blocs de N línies fins acabar de llegir el fitxer (Quan acabem d'utilitzar un bloc l'alliberem amb el seu *free* corresponent). El flag de fi de fitxer, també para l'execució de lectura de la resta de fils perquè està declarat de forma global dins de la zona del mutex de file (cada fil executa lectura més escriptura de forma iterativa).

Finalment, a l'hora d'inserir la informació dels vols a l'arbre també hem de fer **lock** per a cada node per tal d'evitar que 2 o més fils escriguin al mateix node alhora. D'aquesta manera, quan la funció *flight_list()* troba el node a modificar, si aquest existeix el programa **bloqueja** el mutex propi del node (*n_data->mutex*) per poder introduir les dades corresponents a un vol. Al finalitzar, es fa **unlock** del mutex pertanyent al node per tal de que altres fils puguin introduir també informació sobre aquest node. D'aquesta manera aconseguim evitar qualsevol tipus de conflicte relacionat amb els fils. Un cop aquest acabin d'introduir totes les dades corresponents, el fil principal despertarà i el programa tornarà al menú principal (s'espera a l'execució dels fils amb **pthread_join(ntid[i], NULL)**).

Proves realitzades

Hem realitzat la creació de l'arbre amb diferents nombres de fils i diferents mides de blocs, concretament per 2, 4, i 8 fils (ja que l'ordinador no en té més) i per blocs de 10, 100, 1.000, 10.000, 100.000, 1.000.000 i 10.000.000 línies. Hem fet les pràctiques sobre el fitxer de 100 milions de línies. Els resultats són els següents (en segons d'execució):

```
Introdueix fitxer que conte llistat d'aeroports: data/aeroports.csv
Introdueix fitxer de dades: data/big_file.csv
Number of blocks: 10      Number of threads: 2      Time: 159.285513
Number of blocks: 10      Number of threads: 4      Time: 139.810528
Number of blocks: 10      Number of threads: 8      Time: 135.539584
Number of blocks: 100     Number of threads: 2      Time: 137.719929
Number of blocks: 100     Number of threads: 4      Time: 132.748663
Number of blocks: 100     Number of threads: 8      Time: 144.541514
Number of blocks: 1000    Number of threads: 2      Time: 128.594523
Number of blocks: 1000    Number of threads: 4      Time: 129.929398
Number of blocks: 1000    Number of threads: 8      Time: 128.581112
Number of blocks: 10000   Number of threads: 2      Time: 131.790870
Number of blocks: 10000   Number of threads: 4      Time: 130.192257
Number of blocks: 10000   Number of threads: 8      Time: 125.209941
Number of blocks: 100000  Number of threads: 2      Time: 138.786797
Number of blocks: 100000  Number of threads: 4      Time: 125.015290
Number of blocks: 100000  Number of threads: 8      Time: 125.889029
Number of blocks: 1000000 Number of threads: 2      Time: 138.912201
Number of blocks: 1000000 Number of threads: 4      Time: 133.969300
Number of blocks: 1000000 Number of threads: 8      Time: 127.278094
Number of blocks: 10000000 Number of threads: 2      Time: 248.808124
Number of blocks: 10000000 Number of threads: 4      Time: 1794.728027
Number of blocks: 10000000 Number of threads: 8      Time: 1281.392253
```

D'aquests resultats podem extreure que a major número de fils millor en general. Però el que també és molt important és la mida del bloc, un bloc massa gran implicarà el bloqueig de la lectura del fitxer massa temps i no es guanyarà gaire respecte l'execució sense fils secundaris. Per altra banda, un bloc massa petit implicarà no aprofitar suficientment la llesca de temps en la que se't deixa llegir del fitxer.

Conseqüentment, el millor es establir una mida de bloc d'entre 10.000 i 100.000 usant el màxim nombre de fils dels que disposi l'ordinador

Adicionalment, els fitxers estan compilats amb les comandes d'execució optimitzada.