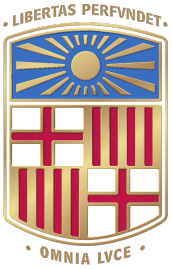

UNIVERSITAT DE BARCELONA

Pràctica 4. Programació multifil

Javier González Béjar
Alejandro Guzmán Requena

15 novembre 2022

ÍNDEX

1 INTRODUCCIÓ

La quarta pràctica de L'assignatura es centra en l'ús de múltiples fils per incrementar l'eficiència computacional del codi proporcionat a la pràctica anterior. Per tant, la importància residirà en la correcta utilització de les funcions de creació i bloqueig de fils, per evitar que aquests executin les denominades seccions crítiques del programa a la vegada.

Amb la utilització de les funcions de bloqueig comprovarem en quina mesura es pot agilitzar l'execució d'un codi en concret amb un paradigma de programació diferent, i quin avantatge en termes d'eficiència suposa aquest.

2 TREBALL REALITZAT AL LABORATORI

1. Qüestions plantejades

Pregunta: Quines variables haurà de passar (per argument) el fil principal al fils secundaris?
Per tal que els fils secundaris disposin de tota la informació necessària durant la seva execució, el principal en crear-los haurà de passar per paràmetre (o argument) l'arxiu a llegir, el nombre de línies N (nombre de línies que es llegeixen cada cop), la matriu a modificar *num_flights* i el llistat d'aeroports *airports* amb el seu codi IATA .

Pregunta: Els fils secundaris accedeixen a recursos (variables) compartits entre ells. Quines seran les seccions crítiques? Quines parts del codi són les que s'han de protegir? Cal protegir la lectura del fitxer? Cal protegir l'extracció de dades del bloc? Cal protegir l'actualització de la variable *num_flights*? Comenteu la vostra resposta
La finalitat del programa és llegir dades del fitxer i escriure-les a una matriu. Per això, creem dos fils que s'encarregaran d'aquestes dues funcions en blocs de N dades a cada crida. Per a evitar que els dos fils llegeixin o escriguin simultàniament en recursos compartits com el fitxer o la matriu on es desen les dades, hem de protegir aquestes seccions crítiques.
Aquestes dues seccions crítiques del programa són, per tant:

- La zona on es llegeixen dades de l'arxiu.
- La zona on escrivim a la matriu *num_flights*.

Així, hem de protegir la secció crítica de lectura o extracció de dades del fitxer per a què no llegeixin alhora ambdós fils tal com s'observa en la figura 2.1.

```
pthread_mutex_lock(&mutex);
while ((its < info->N) && fgets(data[its], MAXCHAR, info->fp) != NULL){
  its++;
}
pthread_mutex_unlock(&mutex);
```

Figura 2.1: Zona crítica de lectura

I també em de protegir la secció crítica d'escriptura a la matriu *num_flights* tal com s'observa en la figura 2.2.

```
pthread_mutex_lock(&mutex);
info->num_flights[index_origin][index_destination]++;
pthread_mutex_unlock(&mutex);
```

Figura 2.2: Zona crítica d'escriptura

2. Experiments amb diferents fils i blocs de línies

En primer lloc fem proves amb N_HILOS = 1 i N_HILOS = 2 i N = 1000

N_HILOS	1	2
2007.csv	235.29	123.90
2008.csv	229.46	116.38

Table 2.1: Execució amb diferent nombre de fils

Com podem comprovar, quan el nombre de fils és 2 triga aproximadament la meitat en executar el programa que quan el nombre de fils és només 1, això té sentit si ho pensem des del punt de vista de que dos fils fan el doble de feina que un.

Ara provarem a anar canviant el nombre de línies a processar per a cada fil a cada lectura i escriptura.

N_línies	1	10	100	1000	10000
2007.csv	124.80	124.28	120.08	123.90	124.57
2008.csv	119.32	117.04	114.44	116.38	118.42

Table 2.2: Execució amb diferent valor d'N

Com es pot comprovar, no existeix una gran diferència en el temps d'execució quan modifiquem el nombre de línies a escriure i llegir pels fils. Tot i així, es pot observar que quan el nombre de línies és més petit, el temps d'execució és major, el que ve provocat a causa del nombre de crides a lock i unlock que es fan (per a cada línia es fa lock i unlock tant per a escriptura com per a lectura) ja que a cada crida n'hi a un cert temps dedicat a cridar a la funció, per tant com més cops es cridi, més gran serà aquest temps.

També podem comprovar que a nombres N molt majors el temps d'execució també augmenta, el que ve provocat perquè tot el temps que s'està llegint dades un fil, l'altra està esperant que aquest acabi, i viceversa a l'hora d'escriure.

3 CONCLUSIONS

Ens agradaria concloure que aquesta pràctica ens ha resultat principalment útil per aprendre la importància del control de les seccions crítiques dels programes, en un paradigma de programació multifil, mitjançant les funcions de bloqueig, ja que quan emprem més d'un fil d'execució tenim el risc de patir solapaments que derivin en resultats erronis.

A nivell personal pensem que la dificultat ha estat elevada en comparació a l'anterior, però l'hem aconseguit assolir sense gaires problemes. També agraïm haver fet la tercera pràctica com a preàmbul d'aquesta ja que això ens ha facilitat la comprensió del codi. Estem per tant satisfets per tant amb el nostre treball.