

Universitat de Barcelona

FACULTAT DE MATEMÀTIQUES I INFORMÀTICA

INFORME PRÀCTICA 4: PROGRAMACIÓ MULTIFIL: FUNCIONS DE BLOQUEIG

Sistemes Operatius 2

Junjie Li i Manuel Liu Wang

Novembre 2022

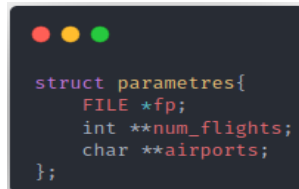
1 Introducció

En aquesta pràctica hem de fer un codi que faci el que fa la pràctica anterior amb diversos fils i amb 2 seccions crítiques, una per lectura i una per escriure.

2 Funcionalitat a implementar

Pregunta 1: Quines variables haurà de passar (per argument) el fil principal al fils secundaris?

Només hi ha 3 paràmetres principals des del fil principal fins al fil secundari, **un fitxer** per accedir i llegir, **una llista d'aeroports** per trobar el número IATA i **una matriu num_flights** per actualitzar i modificar.



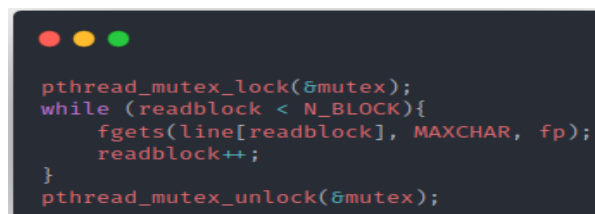
```
struct parametres{
    FILE *fp;
    int **num_flights;
    char **airports;
};
```

Figure 1: variables per argument

Pregunta 2: 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.

Hi ha principalment dues seccions crítiques en el mètode thread, quan els fils volen llegir el fitxer i quan volen actualitzar la variable num_flights.

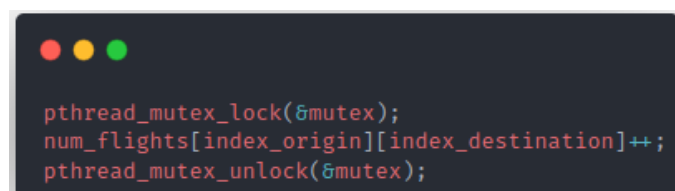
Hem de protegir la lectura de fitxers per evitar que diversos fils llegeixin la mateixa línia de fitxers alhora. Per evitar aquesta situació, podem utilitzar una secció crítica per protegir la lectura de fitxers.



```
pthread_mutex_lock(&mutex);
while (readblock < N_BLOCK){
    fgets(line[readblock], MAXCHAR, fp);
    readblock++;
}
pthread_mutex_unlock(&mutex);
```

Figure 2: secció crítica lectura del fitxer

Protegir l'actualització de la variable num_flights per evitar que diversos fils modifiquin les variables de la mateixa matriu alhora. Per evitar aquesta situació, fem servir una secció crítica per protegir l'actualització de la variable num_flights.



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

Figure 3: secció crítica actualització de la variable num_flights

L'extracció de dades del bloc no requereix protecció, perquè emmagatzemem cada línia llegida del fitxer en una variable dinàmica de **line**, i l'operació d'extracció de dades del bloc es basa en aquesta variable, i com que cada fil té la seva pròpia zona de variable dinàmica, no afectarà altres fils.

```
char** line = (char **) malloc_matrix(N_BLOCK, MAXCHAR, sizeof(char));  
invalid = extract_fields_airport(origin, destination, line[i]);
```

Figure 4: L'extracció

3 Experiments amb diferents N_FILS i N_BLOCK

En primer lloc, comparem el temps d'execució d'un **sol fil** i de **dos fils** amb 1000 block executant el mateix fitxer.

```
junjieli@junjieli-Aspire-A514-53: /media/junjieli/DE9C2D959...  
junjieli@junjieli-Aspire-A514-53:/media/junjieli/DE9C2D959C2D68EB/UB/3/Tardo/S02  
/Practica/Practica_4/codi/codi$ make; \  
./analisi aeroportos.csv fitxer_petit.csv > test_petit.txt; \  
./analisi_original aeroportos.csv fitxer_petit.csv > original_petit.txt; \  
diff test_petit.txt original_petit.txt \  
\  
./analisi aeroportos.csv 2007.csv > test_2007.txt; \  
./analisi_original aeroportos.csv 2007.csv > original_2007.txt; \  
diff test_2007.txt original_2007.txt \  
\  
./analisi aeroportos.csv 2008.csv > test_2008.txt; \  
./analisi_original aeroportos.csv 2008.csv > original_2008.txt; \  
diff test_2008.txt original_2008.txt  
make: Nothing to be done for 'all'.  
304c304  
< Tiempo para procesar el fichero: 0.015712 segundos  
---  
> Tiempo para procesar el fichero: 0.015815 segundos  
304c304  
< Tiempo para procesar el fichero: 5.038674 segundos  
---  
> Tiempo para procesar el fichero: 8.695462 segundos  
304c304  
< Tiempo para procesar el fichero: 4.577762 segundos  
---  
> Tiempo para procesar el fichero: 10.059184 segundos  
junjieli@junjieli-Aspire-A514-53:/media/junjieli/DE9C2D959C2D68EB/UB/3/Tardo/S02
```

Figure 5: comparació de temps d'execució

La figura 5 ens ofereix un temps d'execució molt clar per a cada fitxer. Podem veure clarament que la diferència d'execució entre un fil únic i un fil doble és gairebé el doble.

A continuació provem diferents temps d'execució amb diferents nombres de fils en cas de $N_BLOCK = 1000$:

N_FILS	1	2	3	4	8
petit.csv	0.025090	0.019362	0.020200	0.019374	0.018312
2007.csv	8.641193	4.860380	4.407401	4.253601	4.297149
2008.csv	8.371836	4.565511	4.178817	4.220294	4.050525

Table 1: Comparació de temps d'execució amb diferents nombre de fils.

A la taula anterior, podem veure clarament que quan els fils del programa són superiors a 2, el seu temps d'execució no canvia significativament. Per què és això?

```
junjieli@junjieli-Aspire-A514-53:/media/junjieli/DE9C2D959C2D68EB/UB/3/Tardo/SO2  
/Practica/Practica_4/codi/codi$ cat /proc/cpuinfo | grep 'cores' | uniq  
cpu cores : 2
```

Figure 6: Nombre nuclis de CPU.

Amb la comanda anterior, podem veure que la CPU del nostre ordinador, només té dos nuclis, el que significa que podem utilitzar com a màxim fins a dos nuclis per executar el nostre programa, cada nucli per cada fil, per tant, dos fils per executar el nostre programa. Els fils addicionals es convertiran en processos en el nucli.

A partir de l'experiment anterior, podem saber que el nostre ordinador només pot executar un programa de doble fil com a màxim, de manera que en l'experiment següent, utilitzarem un programa de doble fil i provarem diferents N_BLOCK per comparar el seu temps d'execució.

N_BLOCK	10	100	1000	10000
petit.csv	0.018170	0.018002	0.008621	0.018284
2007.csv	5.036442	5.246535	5.004558	4.983163
2008.csv	4.914276	5.396218	4.780812	4.965698

Table 2: Comparació de temps d'execució amb diferents nombre de blocks.

A l'experiment anterior, podem veure que els diferents N_BLOCK tenen poc efecte en els resultats experimentals i no hi ha molta diferència en el seu temps d'execució.

4 Extra

```
//printids("nou fil: ");  
int readblock = 0;  
pthread_mutex_lock(&mutex);  
while (readblock < N_BLOCK){  
    fgets(line[readblock], MAXCHAR, fp);  
    //printf("%s\n", line[readblock]);  
    readblock++;  
}
```

Figure 7: N_BLOCK continua.

Per aprovar la continuïtat de N_BLOCK, deixem 2 línies de codi comentades al codi, si el descommentem, trobarem que cada fil llegirà el fitxer amb N_BLOCK amb seguretat.

La següent figura és una representació similar:

Podem veure que quan establim N_BLOCK igual a 10, cada fil llegirà els fitxers N_BLOCK continus que li pertanyen.

```
1  nou fil:  pid 14649 tid 2968376896 (0xb0edd640)
2  2008,1,3,4,2003,1955,2211,2225,WN,335,N712SW,128,150,116,-14,8,IAD,TPA,810,4,8,0,,0,NA,NA,NA,NA,NA
3  2008,1,3,4,754,735,1002,1000,WN,3231,N772SW,128,145,113,2,19,IAD,TPA,810,5,10,0,,0,NA,NA,NA,NA,NA
4  2008,1,3,4,628,620,804,750,WN,448,N428WN,96,90,76,14,8,IND,BWI,515,3,17,0,,0,NA,NA,NA,NA,NA
5  2008,1,3,4,926,930,1054,1100,WN,1746,N612SW,88,90,78,-6,-4,IND,BWI,515,3,7,0,,0,NA,NA,NA,NA,NA
6  2008,1,3,4,1829,1755,1959,1925,WN,3920,N464WN,90,90,77,34,34,IND,BWI,515,3,10,0,,0,2,0,0,0,32
7  2008,1,3,4,1940,1915,2121,2110,WN,378,N726SW,101,115,87,11,25,IND,JAX,688,4,10,0,,0,NA,NA,NA,NA,NA
8  2008,1,3,4,1937,1830,2037,1940,WN,509,N763SW,240,250,230,57,67,IND,LAS,1591,3,7,0,,0,10,0,0,0,47
9  2008,1,3,4,1039,1040,1132,1150,WN,535,N428WN,233,250,219,-18,-1,IND,LAS,1591,7,7,0,,0,NA,NA,NA,NA,NA
10 2008,1,3,4,617,615,652,650,WN,11,N689SW,95,95,70,2,2,IND,MCI,451,6,19,0,,0,NA,NA,NA,NA,NA
11 2008,1,3,4,1620,1620,1639,1655,WN,810,N648SW,79,95,70,-16,0,IND,MCI,451,3,6,0,,0,NA,NA,NA,NA,NA
12 nou fil:  pid 14649 tid 2976769600 (0xb16de640)
13 2008,1,3,4,706,700,916,915,WN,100,N690SW,130,135,106,1,6,IND,MCO,828,5,19,0,,0,NA,NA,NA,NA,NA
14 2008,1,3,4,1644,1510,1845,1725,WN,1333,N334SW,121,135,107,80,94,IND,MCO,828,6,8,0,,0,8,0,0,0,72
15 2008,1,3,4,1426,1430,1426,1425,WN,829,N476WN,60,55,39,1,-4,IND,MDW,162,9,12,0,,0,NA,NA,NA,NA,NA
16 2008,1,3,4,715,715,720,710,WN,1016,N765SW,65,55,37,10,0,IND,MDW,162,7,21,0,,0,NA,NA,NA,NA,NA
17 2008,1,3,4,1702,1700,1651,1655,WN,1827,N420WN,49,55,35,-4,2,IND,MDW,162,4,10,0,,0,NA,NA,NA,NA,NA
18 2008,1,3,4,1029,1020,1021,1010,WN,2272,N263WN,52,50,37,11,9,IND,MDW,162,6,9,0,,0,NA,NA,NA,NA,NA
19 2008,1,3,4,1452,1425,1640,1625,WN,675,N286WN,228,240,213,15,27,IND,PHX,1489,7,8,0,,0,3,0,0,0,12
20 2008,1,3,4,754,745,940,955,WN,1144,N778SW,226,250,205,-15,9,IND,PHX,1489,5,16,0,,0,NA,NA,NA,NA,NA
21 2008,1,3,4,1323,1255,1526,1510,WN,4,N674AA,123,135,110,16,28,IND,TPA,838,4,9,0,,0,0,0,0,0,16
22 2008,1,3,4,1416,1325,1512,1435,WN,54,N643SW,56,70,49,37,51,ISP,BWI,220,2,5,0,,0,12,0,0,0,25
23 nou fil:  pid 14649 tid 2968376896 (0xb0edd640)
24 2008,1,3,4,706,705,807,810,WN,68,N497WN,61,65,51,-3,1,ISP,BWI,220,3,7,0,,0,NA,NA,NA,NA,NA
25 2008,1,3,4,1657,1625,1754,1735,WN,623,N724SW,57,70,47,19,32,ISP,BWI,220,5,5,0,,0,7,0,0,0,12
26 2008,1,3,4,1900,1840,1956,1950,WN,717,N786SW,56,70,49,6,20,ISP,BWI,220,2,5,0,,0,NA,NA,NA,NA,NA
27 2008,1,3,4,1039,1030,1133,1140,WN,1244,N714CB,54,70,47,-7,9,ISP,BWI,220,2,5,0,,0,NA,NA,NA,NA,NA
```

Figure 8: N_BLOCK continua.

5 Conclusió

En aquesta pràctica hem après el perquè de les dues seccions crítiques, perquè si no hi haurà solapament, l'ús dels fils i la separació de blocs per cada fil.