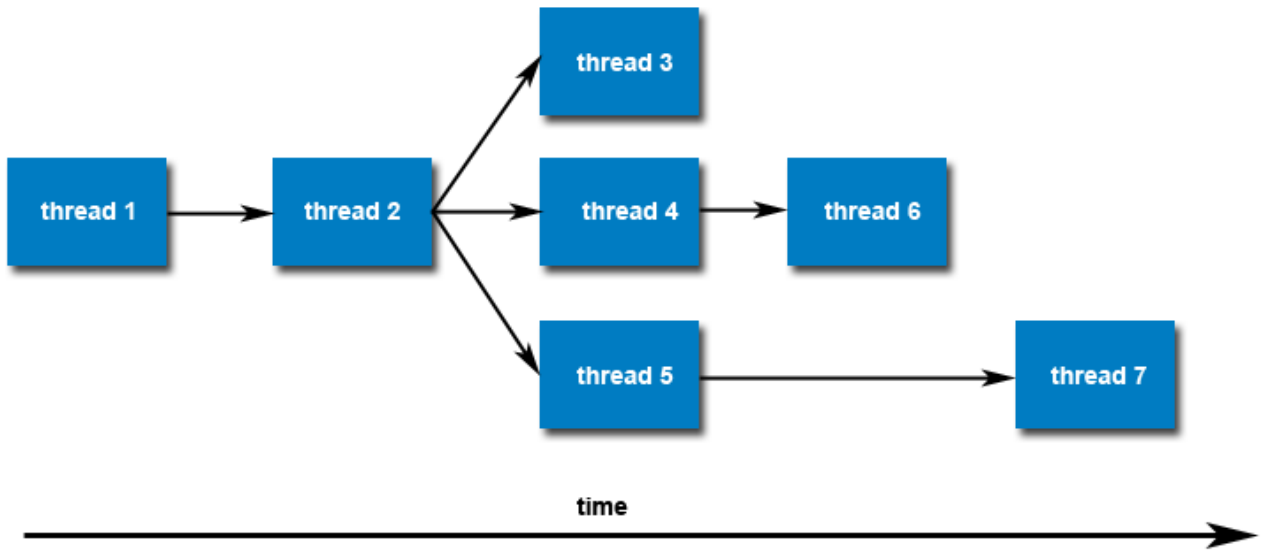
Treballant amb threads

Sistemes Operatius II – Pràctica 4

Xavier de Juan Pulido
David de la Osa Bañales



Introducció:

L'objectiu d'aquesta pràctica ha sigut implementar una solució multifil per a la lectura del fitxer de dades.

En primer lloc, hem realitzat la implementació a nivell d'un únic fil, és a dir, en llegir el fitxer de dades es crea un fil secundari que processa les dades d'aquest fitxer. Per la realització d'aquesta part hem consultat la primera part del fitxer de programació multifil del Campus Virtual per comprendre'n el funcionament.

Feta aquesta part, hem passat a implementar la solució amb múltiples fils, consultant la segona part del fitxer de programació multifil del Campus Virtual entenent d'aquesta manera el funcionament dels monitors, bloqueig i desbloqueig de claus.

Proves Realitzades i Problemes Obtinguts:

En primer lloc, volem comentar que hem definit el nombre de fils amb un *#define* a l'inici del fitxer main.c igual que el bloc de línies a llegir per fil.

El primer problema amb què ens vam trobar el vam tenir a la primera part de la pràctica al haver de crear un fil secundari per llegir i processar les dades del fitxer de dades. Un cop entès com crear el fil ens vam adonar que havíem de passar per paràmetre el punter al fitxer obert i el punter a l'arbre a la funció que executaria el fil. Per això hem creat una estructura *'args_fil'* que conté els dos punters anteriors per passar per paràmetre a la funció que executa el fil.

En posar-nos amb la implementació de la lectura de dades amb múltiples fils hem tingut més problemes.

El primer d'aquests va aparèixer en definir claus per cada node i pel fitxer. Vam haver de diferenciar la forma d'inicialitzar-lo, ja que les claus dels nodes les definim dinàmicament i la clau del fitxer estàticament.

Fet això, havíem d'estructurar el nostre codi per tal d'implementar el processament amb múltiples fils. Per això, vam haver de separar el codi de lectura de dades del processament d'aquestes.

A mesura que realitzàvem la pràctica l'executàvem amb *'valgrind'* per comprovar que tot funcionava correctament i no cometíem errors de memòria. La veritat és que havent entès com bloquejar i desbloquejar claus, juntament amb el flux d'execució, no ha sigut massa difícil dur a terme la implementació.

Finalment, un cop feta la implementació, hem tingut problemes amb la quantitat de dades llegides. Com hem dit inicialment, hem definit el nombre de fils i el bloc de línies a llegir per cada fil. El problema que teníem era que volíem llegir el fitxer de 10.000 línies, amb 4 fils i blocs de 2.000 línies, amb una única execució de cada fil, obtenim 2.000 * 4 = 8.000 línies. Per solucionar-ho hem fet que s'executi en un bucle fins que l'última línia llegida és la del final del fitxer, sortint així d'aquest bucle.

Per provar aquesta implementació s'han dut a terme múltiples proves, amb múltiples valors de fils i de blocs de línies imprimint per pantalla el nombre de línies llegides juntament amb l'identificador del fil. Finalment, s'ha aconseguit realitzar aquesta implementació satisfactòriament.
Tot i això, cal dir que al executar el programa amb *'valgrind'*, el programa es penja i no s'executa tal i com s'espera quan s'intenta llegir el fitxer de dades big_file.csv (100.000.000 de línies), això no passa amb el fitxer de dades de la pràctica anterior (10.000 línies).

Anàlisi del temps d'execució:

Totes les proves han sigut realitzades sobre el fitxer big_file.csv de 100.000.000 de línies.
Hem realitzat les proves amb 2, 4 i 8 fils per comparar el rendiment. Les proves han sigut realitzades amb un portàtil de 8 nuclis.

1) En aquestes proves el bloc de línies a llegir l'hem establert a 1.000 i hem realitzat les proves sobre el fitxer de de dades big_file.csv de 100.000.000 de línies.

Aquests han sigut els resultats de les proves:

- 2 fils: creació arbre amb fitxer de dades big_file.csv.
    - Temps CPU: 140.182313 segons
    - Temps cronològic: 70.633625 segons
- 4 fils: creació arbre amb fitxer de dades big_file.csv.
    - Temps CPU: 152.534110 segons
    - Temps cronològic: 38.686871 segons
- 8 fils: creació arbre amb fitxer de dades big_file.csv.
    - Temps CPU: 239.430459 segons
    - Temps cronològic: 31.363966 segons

2) En aquestes proves el bloc de línies a llegir l'hem establert a 10.000 i hem realitzat les proves sobre el fitxer de de dades big_file.csv de 100.000.000 de línies.

Aquests han sigut els resultats de les proves:

- 2 fils: creació arbre amb fitxer de dades big_file.csv.
    - Temps CPU: 137.581549 segons
    - Temps cronològic: 70.141862 segons
- 4 fils: creació arbre amb fitxer de dades big_file.csv.
    - Temps CPU: 151.054508 segons
    - Temps cronològic: 38.967598 segons
- 8 fils: creació arbre amb fitxer de dades big_file.csv.
    - Temps CPU: 237.816926 segons
    - Temps cronològic: 30.962796 segons

3) En aquestes proves el bloc de línies a llegir l'hem establert a 100.000 i hem realitzat les proves sobre el fitxer de de dades big_file.csv de 100.000.000 de línies.

Aquests han sigut els resultats de les proves:

- 2 fils: creació arbre amb fitxer de dades big_file.csv.

    - Temps CPU: 135.397080 segons
    - Temps cronològic: 70.236104 segons
- 4 fils: creació arbre amb fitxer de dades big_file.csv.
    - Temps CPU: 143.924135 segons
    - Temps cronològic: 36.983115 segons
- 8 fils: creació arbre amb fitxer de dades big_file.csv.
    - Temps CPU: 238.568728 segons
    - Temps cronològic: 30.721873 segons

Com podem veure, obtenim els millors resultats, en termes de temps cronològic, amb 8 fils. Pel que fa al nombre de línies per bloc, veiem que no influeix massa en el rendiment, tot i això, els millors resultats veiem que han sigut obtinguts amb blocs de 100.000 línies.

Finalment hem deixat el nostre codi amb un nombre de 4 fils i 1.000 línies per bloc.