Sistemes Operatius II - Pràctica 3

Novembre del 2013

La tercera pràctica de sistemes operatius 2 se centra en l'ús de múltiples fils per incrementar l'eficiència computacional de l'aplicació. La data d'entrega de la tercera pràctica és el diumenge **1 de desembre del 2013**.

Índex

1 Introducció

La primera i la segona pràctica s'han centrat en aprendre a utilitzar diverses estructures per a la manipulació de dades, així com en l'ús de canonades per la comunicació interprocés. En aquesta tercera pràctica ens concentrem en utilitzar múltiples fils per a la creació de l'arbre. El punt de partida d'aquesta pràctica és el codi de la pràctica 2. A les següents seccions es detallen les tasques a realitzar.

2 La pràctica

Recordem a grans trets el procediment per construir un arbre

1. S'obre un fitxer de configuració. Aquest fitxer de configuració conté, a la primera línia, el nombre de fitxers a processar. La resta de les línies contenen els noms dels fitxers a processar.
2. Cada fitxer es processa analitzant les paraules i copiant-les a una estructura local.
3. Un cop finalitzat el processament de les paraules d'un fitxer es copien les paraules de l'estructura local a l'estructura global.

Aquest algorisme s'ha implementat fent servir un sol fil d'execució. L'objectiu d'aquesta pràctica se centra en modificar el codi perquè l'arbre sigui creat amb un mínim de dos fils. Si la implementació es realitza de forma correcta es podrà crear l'arbre amb tants fils com es vulguin.

3 Implementació

Per a la implementació de la solució es demana utilitzar monitors. A continuació es descriu l'esquema a implementar (cada grup de pràctiques tindrà variacions sobre aquesta proposta):

1. En executar el vostre programa, només hi haurà un fil. Anomenarem a aquest fil el fil principal. Aquest fil serà el que imprimirà per pantalla el menú, el que permetrà desar l'arbre a disc, carregar-lo de disc o bé analitzar les paraules de l'arbre.
2. En seleccionar del menú l'opció de creació d'arbre, el fil principal demanarà per teclat el fitxer de configuració. El fil principal obrirà aquest fitxer i llegirà només la primera línia, que conté el nombre de fitxers a processar.
3. En aquest moment s'han de crear els fils (com a mínim dos fils) amb la funció `pthread_create`. Haureu de passar a cada fil la informació necessària perquè pugui accedir als fitxers de la llista de configuració i pugui copiar les dades a l'arbre. Vegeu la fitxa 3 per veure com passar informació als fils.

4. Un cop s'han creat els fils el fil principal es quedarà esperant que els fils creats finalitzin la creació de l'arbre. Ho pot fer amb la funció `pthread_join`.

5. Cada fil creat començarà a executar. Aquests fils hauran de realitzar el següent procediment iteratiu fins que tots els fitxers s'hagin processat

   (a) Agafar el primer element no processat de la llista de configuració. A l'hora d'agafar el primer element no processat heu d'anar amb compte que cap altre fil també agafi el mateix fitxer. Per això utilitzeu les funcions `pthread_mutex_lock` i `pthread_mutex_unlock`. No és bona idea repartir prèviament els fitxers entre fils (per exemple, un fil fa els fitxers senars i l'altre els parells) ja que es pot donar el cas que un fil acabi molt abans que un altre de processar la seva part de les dades degut a la mida de les dades a processar. En aquesta pràctica cada fitxer té una mida diferent i per tant el temps que es necessita per processar cada fitxer és també diferent.

   (b) El fil extraurà les paraules del fitxer i les inserirà en una estructura local. Heu de tenir en compte que cada fil tindrà la seva pròpia estructura local i, per tant, no hi haurà interferència entre fils (encara que utilitzin les mateixes funcions). Assegureu-vos que no hi ha cap variable global compartida entre els fils a l'hora d'inserir les paraules a l'estructura local.

   (c) Un cop creada l'estructura local el fil haurà de copiar les dades de l'estructura local a la global. Tingueu en compte que heu d'evitar deixar l'arbre en un estat inconsistent degut a la manipulació per múltiples fitxers de l'arbre. Per tal d'assegurar això s'han d'utilitzar les funcions `pthread_mutex_lock` i `pthread_mutex_unlock`. Hi ha diverses formes d'utilitzar aquestes funcions per aconseguir consistència de l'arbre: la primera solució, la més senzilla, es basa en utilitzar aquestes dues funcions per protegir (tota) la funció que copia l'estructura local a l'arbre. La segona solució, una mica més complexa, es basa en utilitzar les dues funcions per protegir realment el que cal protegir: les funcions de manipulació d'arbre (inserció d'un element a l'arbre, cercar un element de l'arbre, etc.).

   (d) Un cop copiada l'estructura local a l'arbre cal alliberar l'estructura local i agafar el següent element no processat de la llista de configuració (pas a). En cas que no n'hi hagi cap més fitxer a processar, el fil sortirà de la seva funció d'entrada. D'aquesta forma el fil finalitzarà.

6. Un cop hagin finalitzat tots els fils que processaven els fitxers de text, el fil principal es despertarà (estava esperant a pthread_join) i tornarà a visualitzar el menú.

4 Planificació

Per planificar-vos la feina, és important entendre bé el funcionament dels fils.

1. Llegiu i experimenteu amb la fitxa 3 i 4 penjades al campus que parlen dels fils. Tingueu en compte que per a la realització d'aquesta pràctica no cal utilitzar variables condicionals. Les variables condicionals s'utilitzaran a la darrera pràctica.

2. A l'hora d'utilitzar múltiples fils per construir l'arbre caldrà que cada fil realitzi el procediment iteratiu descrit a la secció anterior. Modifiqueu el vostre codi, en cas necessari, perquè en una una única funció hi hagi el procediment descrit a la secció anterior. Encara no cal que es creïn els fils, només esteu preparant el codi per fer-lo multil.: la funció serà executada pel fil principal.

3. Modifiqueu el codi perquè el fil principal crei només 1 sol fil per crear l'arbre. El fil creat realitzarà la creació de l'arbre mentre el fil principal esperarà que el fil creat finalitzi. Observar que en aquest punt no cal utilitzar les funcions d'exclusió mútua. Simplement permet assegurar-vos que el procediment de creació de l'arbre es realitza correctament si es fa servir 1 fil que no és el fil principal.

4. Si el punt 3 funciona correctament, podeu introduir al codi les funcions `pthread_mutex_lock` i `pthread_mutex_unlock` necessàries per tal d'assegurar exclusió mútua entre els múltiples fils. El fil principal crearà un mínim de dos fils i esperarà que aquests finalitzin.

5 Entrega

El fitxer que entregueu s'ha d'anomenar `P3_Cognom1Cognom2.tar.gz` (o `.zip`, o `.rar`, etc), on `Cognom1` és el cognom del primer component de la parella i `Cognom2` és el cognom del segon component de la parella de pràctiques. El fitxer pot estar comprimit amb qualsevol dels formats usuals (`tar.gz`, `zip`, `rar`, etc). Dintre d'aquest fitxer hi haurà d'haver tres carpetes: `src`, que contindrà el codi font, `proves`, que contindrà resultats d'execució i `doc`, que contindrà la documentació addicional en PDF. Aquí hi ha els detalls per cada directori:

- La carpeta `src` contindrà el codi font. S'hi han d'incloure tots els fitxers necessaris per compilar i generar l'executable. El codi ha de compilar sota Linux amb la instrucció `make`. Editeu el fitxer *Makefile* en cas que necessiteu afegir fitxers C que s'hagin de compilar. El codi font ha d'estar comentat en anglès en un format similar al mostrat als fitxers `red-black-tree.c` i `linked-list.c`. Es necessari comentar com a mínim les funcions que hi ha al codi.

- La carpeta `proves` ha d'incloure qualsevol informació que considereu rellevant. Aquesta informació haurà d'estar comentada a la memòria.

- El directori `doc` ha de contenir un document (dues o tres pàgines, en format PDF) explicant el funcionament de l'aplicació, la discussió de les

proves realitzades i els problemes obtinguts. És particularment interessant que comenteu sobre els següents punts

- Una comparativa sobre el temps d'execució fent servir 1 sol fil o 2 (o més). A la fitxa 4 teniu exemples que mostren com es pot mesurar el temps d'execució d'una funció.
- Quina solució d'exclusió mútua és més eficient ? És millor protegir tota la funció que copia l'estructura local a l'arbre o és millor protegir les funcions que permeten manipular els arbres ? Aquesta pregunta és típica a qualsevol codi que es vol fer multifil: a quin nivell s'han de realitzar la implementació de l'exclusió mútua ?

En aquest document no s'han d'explicar en detall les funcions o variables utilitzades. Sí que es pot explicar (incloent gràfics) les estructures utilitzades.

La data límit d'entrega d'aquesta pràctica és el **1 de desembre del 201**3. El codi tindrà un pes d'un 70% (codi modular i net, ús correcte del llenguatge, bon estil de programació, el programa funciona correctament, tota la memòria és alliberada, sense accessos invàlids a memòria, etc.) i el document i les proves el 30% restant.