Sistemes Operatius II - Pràctica 5

Novembre del 2018

Aquesta darrera pràctica se centrarà en la utilització de variables condicionals per tal d'implementar l'esquema de creació de l'arbre fent servir el paradigma del productor-consumidor.

Índex

1 Introducció

La pràctica 4 s'ha centrat en la creació de l'arbre fent servir múltiples fils. En particular, el fil principal gestionava el menú i la creació dels fils secundaris que processaven les dades dels vols d'avions. Els fils secundaris realitzaven dues tasques ben diferenciades: la lectura de les dades de disc (en blocs de N línies) i el processament d'aquestes dades introduint la informació a l'arbre.

En aquesta pràctica es re-estructura el codi de processament de les dades de forma que hi hagi un únic fil secundari, el productor, que únicament s'encarregarà de la lectura de dades de disc (en blocs de N línies) i F fils secundaris més, els consumidors, que processaran aquestes dades i les introduiran a l'arbre. El productor haurà de passar als consumidors els blocs de les línies llegides. Es proposa que aquesta comunicació entre productor i consumidor es realitzi mitjançant el paradigma de productor-consumidor.

A les següents seccions es detallen les tasques a realitzar. Per a la implementació de la solució es demana utilitzar monitors[1]. Les funcions que es mencionen en aquest document es trobareu descrites a les dues fitxes del campus.

2 Funcionalitat a implementar

A continuació es descriu l'esquema general a implementar.

1. En executar el vostre programa només hi haurà un fil. Anomenarem 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 imprimir el nombre de vegades que una paraula apareix a l'arbre.

2. En seleccionar del menú l'opció de creació d'arbre, el fil principal demanarà per teclat el fitxer dels aeroports així com el fitxer que conté les dades dels vols d'avions. A continuació el fil principal crearà l'estructura d'arbre a partir del fitxer d'aeroports.

3. Per processar les dades dels vols es faran servir múltiples fils secundaris. El fil principal crearà, amb la funció *pthread_create*, **un únic fil productor** i F **fils consumidors**. Atès que el productor i els consumidors es comunicaran entre sí a través d'un buffer (circular), farà falta que el fil principal passi per argument, tant al fil productor com als fils consumidors, el buffer de comunicació. El fil principal haurà de passar al productor i als consumidors la resta d'informació necessària com, per exemple, el fitxer de dades i l'arbre respectivament.

4. Mentre el productor i els consumidors llegeixen i processen, respectivament, les dades, el fil principal es quedarà esperant, amb la funció *pthread_join* que els fils secundaris acabin la seva feina.

5. Un cop hagin finalitzat tots els fils secundaris amb la seva feina, el fil principal es despertarà i tornarà a visualitzar el menú.

El fil productor i consumidor seguiran el paradigma del productor-consumidor. En concret,

1. El productor és l'únic fil encarregat de llegir el fitxer de dades. Llegirà el fitxer de dades en blocs de N línies i transferirà cada bloc al buffer que comparteixen productor i consumidors.

[1] A data d'inici d'aquesta pràctica possiblement encara no s'han impartit a teoria. La pràctica es pot iniciar, però, sense aquest coneixement. Veure secció 3.

El buffer tindrà una mida per poder emmagatzemar B blocs, i es recomana que B sigui igual o superior al nombre F de fils secundaris.

2. Els consumidors agafaran del buffer els blocs de N línies, processaran les línies i inseriran la informació extreta a l'arbre compartit tal com s'ha fet a la pràctica 4. És a dir, cada node de l'arbre tindrà la seva pròpia clau i el fil consumidor haurà d'agafar la clau per poder accedir a un node. Observar que només els nodes consumidors tenen accés a l'arbre.

3 Planificació i implementació

Per planificar-vos la feina, és important entendre bé el funcionament dels fils. Es proposen els següents punts de treball:

1. Modifiqueu el vostre codi perquè, en seleccionar el menú de creació de l'arbre, el fil principal crei dos fils: un fil productor i un consumidor. El fil productor és qui llegeix el fitxer de dades mentre que el fil consumidor és qui processa les dades. Dissenyeu l'estructura del buffer de comunicació perquè el consumidor pugui hi pugui col·locar molt ràpidament un bloc de N línies, i que el productor en pugui agafar un bloc (veure final d'aquesta secció).

2. Llegiu i experimenteu amb la fitxa del campus que parla dels monitors i les variables condicionals (document de programació amb fils, 2a part). Aquesta és la que us permetrà implementar l'esquema del productor-consumidor.

3. Implementeu l'esquema del productor-consumidor fent servir un productor i un consumidor. El productor i el consumidor hauran de finalitzar de forma "neta", és a dir, hauran de sortir de la seva funció d'entrada executant el "return". No està permès fer servir funcions que "matin" els fils per finalitzar-los (hi ha una funció en C que permet fer-ho).

   El fet que només hi hagi un productor i un consumidor fa que sigui més senzilla la condició de finalització, la condició que permet saber que no hi ha més feina a fer. Com sabrà el productor si ha acabat? Ho sabrà quan hagi llegit tot el fitxer i hagi transferit els blocs a les cel·les del buffer. Aleshores, com sabrà si el consumidor ha acabat? Ho sabrà quan el productor hagi transferit totes les dades a les cel·les i el buffer sigui buit. Heu de tenir en compte aquest fet a l'hora de sincronitzar el productor i el consumidor.

4. Llegiu i experimenteu amb les les funcions de bloqueig que s'expliquen també a la següent fitxa penjada al campus. Tingueu en compte que per a la realització d'aquesta pràctica no cal utilitzar variables condicionals. Les variables condicionals es faran servir a la següent pràctica.

5. A continuació caldrà ampliar l'aplicació per a un productor i F consumidors. Haureu d'implementar l'algorisme de comunicació per a múltiples consumidors i productors. Una de les complicacions es troba en el fet de saber quan acaben els productors i consumidors ja que, igual que pel punt anterior, els productors i consumidors han d'acabar de forma "neta".

   Quina és la condició que permet saber que el productor han acabat? Igual que abans, ho sabrà quan hagi llegit tot el fitxer i hagi transferit els blocs a les cel·les del buffer. Com saben els consumidors si han acabat? Igual que abans, quan el productor hagi transferit totes les dades a les cel·les i el buffer sigui buit. Heu de tenir en compte aquest fet a l'hora de sincronitzar el productor i els consumidors.

6. El fil principal tornarà a mostrar el menú un cop el productor i consumidors hagin acabat.

Un dels elements essencials per aconseguir una bona eficiència a l'aplicació és fer que el productor i el consumidor realitzin poques operacions costoses a l'interior de la secció crítica del buffer compartit. Es recomana doncs que el productor treballi amb variables locals mentre llegeixi les dades del fitxer. És a dir, el productor hauria de tenir una cel·la local on emmagatzemi les dades llegides. Un cop la cel·la s'ompli, pot transferir les dades a una cel·la del buffer. De la mateixa forma, el consumidor haurà d'agafar les dades del buffer. Per processar-les es recomana que ho faci amb una variable associada a una cel·la local.

A més es demana que la transferència de dades entre el productor cap al buffer així com del buffer cap al consumidor es faci de forma eficient. Això implica "operar" amb punters del llenguatge C per evitar copiar text (amb la funció strcpy, per exemple) a l'hora de transferir dades al/de buffer. Es proporciona un exemple amb aquest pràctica de com procedir.

Per acabar, es comenta que a l'hora de realitzar el document d'aquesta pràctica s'ha implementat també una solució en què el consumidor utilitza una còpia local de l'arbre de forma que no cal bloquejar els nodes de l'arbre en inserir-hi la informació extreta de les línies. En acabar de processar tota la informació el consumidor bolca la informació del seu arbre local a l'arbre compartit. S'ha observat que aquesta solució és una mica més eficient que la solució proposada en aquesta pràctica (es passa d'uns 17 segons a 16 segons d'execució als ordinadors en què s'han fet les proves). Recordar que en aquesta pràctica es proposa que els fils consumidors insereixen la informació extreta directament a l'arbre compartit bloquejant la clau del node.

4 Entrega

El fitxer que entregueu s'ha d'anomenar `P5_NomCognom1NomCognom2.tar.gz` (o `.zip`, o `.rar`, etc), on `NomCognom1` és el cognom del primer component de la parella i `NomCognom2` 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 dues carpetes: `src`, que contindrà el codi font, i `doc`, que contindrà la documentació addicional en PDF. Aquí hi ha els detalls:

- La carpeta `src` contindrà el codi font de la pràctica. 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 directori `doc` ha de contenir un document (màxim 5 pàgines, en format PDF) explicant la discussió de les proves realitzades i els problemes obtinguts. En aquest document no s'han d'explicar en detall les funcions o variables utilitzades. És particularment interessant que feu una anàlisi (no cal que sigui exhaustiu) del temps d'execució fent servir diversos valors d'F, diferents valors d'N, la mida del bloc, i B, el nombre de cel·les del buffer. Feu totes les proves en un únic ordinador; els resultats obtinguts dependran molt del tipus de disc (magnètic o SSD) així com el tipus de processador i altres característiques de l'ordinador. Es recomana fer les proves pel fitxer de 100.000.000 de línies (disponible al campus), atès que pel fitxer de 7.000.000 de línies el temps d'execució és relativament petit per a un sol fil (és inferior a 10 segons).

És important que no feu servir màquines virtuals per fer aquestes proves ja que una màquina virtual acostuma a executar-se en un sol processador. A més, també és convenient que proveu el codi compilat amb opcions d'optimització: no feu servir l'opció -g en compilar sinó que feu servir l'opció -O.

A la qualificació d'aquesta pràctica, el codi tindrà un pes d'un 80% i el document i les proves el 20% restant.