Sistemes Operatius II - Pràctica 5

Novembre del 2017

La darrera pràctica de sistemes operatius 2 se centra en la implementació de l'algorisme del productor-consumidor per a la implementació de l'algorisme multifil que processa els fitxers PDF.

Índex

1 Introducció

A la pràctica 4 s'ha realitzat una implementació multifil de l'algorisme per processar els PDFs. L'esquema a implementar era aquest:

1. En executar el vostre programa només hi ha un fil, el fil principal. Aquest fil és el que imprimeix per pantalla el menú, el que permet 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 demana per teclat el fitxer que conté el llistat de fitxers PDF a processar. El fil principal crea un mínim de $F = 2$ fils secundaris, els quals s'han de repartir entre sí els fitxers PDF a processar per crear l'arbre. Mentrestant, el fil principal es queda esperant que els fils secundaris acabin la seva feina.

3. Un cop finalitzen tots els fils secundaris amb la seva feina, el fil principal es desperta i tornarà a visualitzar el menú.

Aquest esquema us haurà permès augmentar l'eficiència computacional per tal de crear l'arbre. Aquest algorisme, però, té alguns defectes

- Cada fil realitza la mateixa feina: extreu les paraules del fitxer PDF amb una aplicació externa, extreu les paraules del text, les insereix a un arbre local i acaba d'inserir-les a l'arbre compartit. Múltiples fils poden estar executant l'aplicació externa per extreure les paraules a la vegada, per exemple. Això pot produir una reducció del rendiment. Per altra banda, observar que no hi ha realment una especialització de les tasques a realitzar pels fils.

- Cada fil processa un text de mida diferent ja que se li han assignat fitxers de mida diferent. Imagineu-vos el cas extrem en que hi ha un únic fitxer molt gran i d'altres ben petits. El fil que processi el fitxer gran necessitarà molta estona per fer-ho, mentre que els altres fils poden acabar ràpidament de processar tota la resta de fitxers petits. Un cop acabin aquests darrers, no tindran res a fer mentre el primer fil estigui processant el fitxer gran.

En aquesta pràctica es proposa una solució per tal d'abordar aquests dos defectes.

2 Funcionalitat a implementar

Per a la implementació de la solució es demana utilitzar monitors[1]. El fil principal funcionarà de forma molt similar a com ho fa a la pràctica anterior.

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 que conté el llistat de fitxers PDF a processar. El fil principal crearà, amb la funció `pthread_create`, un conjunt de **fils secundaris** els quals hauran de repartir-se entre sí els fitxers PDF a processar i crearan l'arbre. Mentrestant, el fil principal es quedarà esperant, amb la funció `pthread_join`, que els fils secundaris acabin la seva feina.

[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.

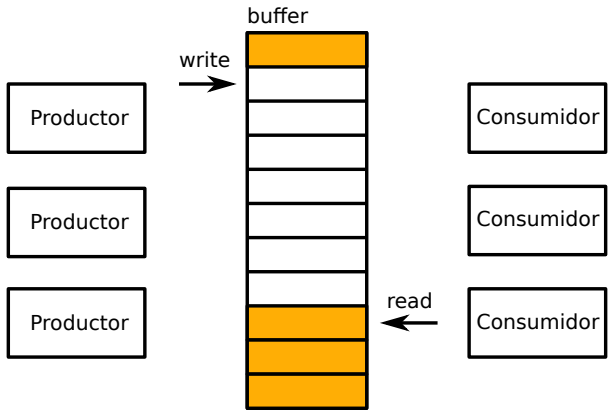

Figura 1: Esquema del productor-consumidor a implementar en aquesta pràctica. Hi ha 1 productor i F consumidors. L'esquema s'ha copiat de les transparències de teoria "Introducció a la concurrència".

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

L'esquema a implementar és un productor-consumidor, veure figura 1. El fil principal crearà $P \geq 1$ productors i $C \geq 1$ consumidors. Els productors i els consumidors es comunicaran entre sí a través del *buffer* compartit entre els fils.

Es descriu a continuació les tasques a fer pels productors i els consumidors:

- Hi haurà P productors. Cada fil productor tindrà com a tasca extreure el text dels PDFs (mitjançant l'aplicació externa `pdftotext`) i inserir-lo al *buffer* compartit. En concret, cadascuna de les cel·les del *buffer* de la figura 1 podrà emmagatzemar N línies de text, on N és un valor constant i prefixat per a totes les cel·les. Per exemple, un valor de $N = 2000$ indica que cada cel·la del *buffer* pot emmagatzemar (fins a) 2000 linies de text.

  Cada productor començarà per extreure el text del primer fitxer PDF (que hagi pogut agafar de la llista compartida). El productor anirà omplint una cel·la amb el text extret. Així que una cel·la s'ompli, el productor començarà per omplir una segona cel·la, i així de forma successiva fins que hagi extret tot el text del primer fitxer PDF. Observar doncs que un fitxer PDF pot ocupar múltiples cel·les del *buffer*. Un cop el productor hagi acabat amb el primer fitxer PDF, continuarà amb el següent fitxer PDF (que hagi pogut agafar de la llista compartida): extraurà el text del segon fitxer i continuarà per omplir la cel·la allà on ho ha deixat amb el primer fitxer que ha processat. Així de forma successiva fins que s'extregui el text de tots els fitxers PDF que hagi agafat.

  Els productors es repartiran entre sí, doncs, els fitxers PDF a processar i aniran transferint al *buffer* les dades llegides. Cada productor serà independent de la resta i omplirà les cel·les del *buffer* de forma independent de la resta de productors. Cal assegurar, però, que els productors no s'encavalquin entre sí i escriguin a una mateixa cel·la.

- Hi haurà C fils consumidors. El fils consumidors s'encarregaran d'agafar les línies de text que hi ha a les cel·les del *buffer*, extreure'n les paraules i inserir-les a l'arbre local. Cada cop que un fil acabi de processar una cel·la, bolcarà la informació a l'arbre global, de forma similar a com

s'ha fet a la pràctica anterior. A continuació agafarà el text d'un altra cel·la per processar-la, de forma similar a com ho ha fet amb l'anterior cel·la.

Aquest esquema proposa una solució als defectes de la pràctica anterior. En concret, observar que hi ha múltiples fils productors encarregats d'extreure el text dels fitxers PDF mentre que els fils consumidors processen el text i l'insereixen a l'arbre. Hi ha doncs una especialització de les tasques a fer per cada fil. Per altra banda, la feina es pot repartir de forma més uniforme entre els fils productors i consumidors. En cas que hi hagi un fitxer PDF molt gran, la feina de processament es repartirà entre els fils consumidors. No serà pas un únic fil el que s'encarregui de processar tot el fitxer gran.

Arran d'aquest esquema pot sorgir a més la pregunta: per obtenir un bon rendiment, faran falta molts productors i pocs consumidors? O és suficient amb un productor i múltiples consumidors? Quina solució creieu que permetrà obtenir més eficiència? En altres paraules, on és el coll d'ampolla? Es troba a l'hora de fer la canonada per extreure les línies de text del PDF o a l'hora d'extreure les paraules i inserir-les a l'arbre? Podreu obtenir la resposta aviat!

3 Planificació i implementació

A continuació es descriu l'esquema a implementar. Cada grup de pràctiques pot tenir variacions sobre aquesta proposta, però és convenient mencionar-les al document en cas que es decideixi fer-ho.

1. Llegiu i experimenteu amb la fitxa de programació de fils, 2a part, que és la que explica les funcions per implementar l'algorisme de productor-consumidor. En aquesta pràctica caldrà fer servir les funcions d'exclusió mútua i les funcions de sincronització condicional.

2. En seleccionar del menú l'opció de creació d'arbre el fil principal demanarà per teclat el fitxer que conté els fitxers PDF a processar. Es recomana que el fil principal creï el *buffer* de comunicació, veure figura 1. Cada cel·la ha de poder contenir N línies de text. El *buffer* tindrà B cel·les. Aquest valor hauria de ser més gran que el nombre de productors. D'aquesta forma els productors poden anar transferint les dades llegides al *buffer* sense esperar que els consumidors les agafin.

3. Per començar a programar l'aplicació es recomana que l'aplicació tingui un productor, un consumidor i un *buffer* de mica B. 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 processat tots els fitxers i hagi transferit totes les dades a les cel·les del *buffer*. 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.

4. A continuació caldrà ampliar l'aplicació per a P productors i C 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 els productors han acabat? Quan tots els productors hagin transferir totes les dades de tots els fitxers a les cel·les del *buffer*. Com saben els consumidors si han acabat? Quan tots els productors hagin transferit totes les dades al *buffer* i el *buffer* sigui buit.

5. El fil principal es despertarà així que el fil productors i consumidor hagin acabat. Aleshores tornarà a mostrar el menú.

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. En aquesta pràctica això implica, per exemple, evitar que el productor llegeixi de la canonada i copiï les dades al *buffer* mentre és a l'interior de la secció crítica. Caldrà doncs llegir les dades de la canonada abans de transferir-les al *buffer*. Per al consumidor cal evitar que extregui les paraules de la cel·la mentre és a l'interior de la secció crítica. Ho ha de fer a l'exterior de la cel·la. A l'interior de la secció crítica el productor i consumidor només haurien de transferir les dades del *buffer* a/de variables locals del fil.

Es recomana doncs que el productor treballi amb variables locals mentre llegeixi les dades de la canonada. 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 al *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. Idealment, a més, en cas que es vulgui aconseguir realment una bona eficiència, això implica "operar" amb punters del llenguatge C per evitar copiar text (amb la funció `strcpy`, per exemple) a l'hora de transferir dades de/al *buffer*.

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 dos 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 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. Es necessari comentar com a mínim les funcions que hi ha al codi.
- El directori `doc` ha de contenir un document màxim 5 pàgines, en format PDF) explicant el funcionament de l'aplicació, 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, però, que feu una comparativa fent els següents experiments
  - Sigui U el nombre de processadors de l'ordinador. Agafeu $P = 1$ i $C = U$. Quan de temps triga en executar l'aplicació? Proveu ara amb $P = U$ i $C = 1$. Quina de les dues configuracions té una major eficiència? Quines conclusions podeu extreure d'aquests dos experiments? Podeu agafar, per exemple, una mida de *buffer* amb $B = 50$ cel·les per assegurar que el *buffer* no limita.

- Del punt anterior, agafeu la combinació que P i C que us doni millor resultat. Feu aleshores experiments sobre el valor de N, el nombre de línies de text que emmagatzema cada cel·la, per avaluar l'eficiència de l'aplicació. Proveu amb valors de $N = 500$, $N = 1000$, $N = 1500$, i així successivament fins $N = 5000$, per exemple. Hi ha algun valor de N que sigui "òptim" a nivell d'eficiència? Té algun efecte notable el valor d'N? Què en podeu concloure?
- Compareu la solució proposada en aquesta pràctica amb la implementada a la pràctica anterior. A la pràctica anterior executeu l'aplicació amb P fils secundaris (tants fils secundaris com processadors té la màquina). Què és el que obteniu?

El codi tindrà un pes d'un 80% i el document i les proves el 20% restant.