Sistemes Operatius II - Pràctica 4

Novembre 2019

La quarta pràctica se centra en la creació de l'arbre fent servir tècniques de concurrència. En particular, es faran servir múltiples processos amb relació pare-fill que es comunicaran entre sí fent servir fitxers anònims mapats a memòria, també coneguda com a memòria compartida.

Índex

1 La pràctica

La pràctica 2 i 3 s'han centrat en la creació de l'arbre i la persistència de l'arbre a disc. Aquesta pràctica se centra únicament en la part de la creació de l'arbre, que es va fer a la pràctica 2. No caldrà modificar el codi associat a la persistència de l'arbre, implementat a la pràctica 3.

En aquesta pràctica es crearà l'arbre fent servir tècniques concurrents. En aquesta pràctica 4 la concurrència serà a nivell de procès i a la pràctica 5 serà a nivell de fil. Es proposa que en aquesta pràctica la sincronització es realitzi a nivell de semàfor i a la pràctica 5 amb monitors (mutex).

Atès que en aquesta pràctica es faran servir múltiples processos per crear l'arbre de forma concurrent, caldrà fer servir alguna eina de comunicació interprocés que permeti que els processos es comuniquin entre sí per actualitzar les dades dels nodes de l'arbre. Hi ha múltiples eines de comunicació interprocés que es poden fer servir per realitzar aquesta tasca. Una que és habitual són els *sockets* (comunicació via xarxa). És la que fa servir el navegador Chrome perquè els processos es comuniquin entre sí.

Aquí es faran servir els fitxers anònims mapats a memòria, tècnica coneguda també com a memòria compartida, que és la tècnica de comunicació interprocés més ràpida entre processos d'un mateix d'ordinador. Les dades del node (l'estructura `node_data`) estaran emmagatzemades a un fitxer anònim mapat a memòria, permetent que diversos processos puguin accedir a aquestes dades per accedir i actualitzar els membres. Els fitxers anònims mapats a memòria són equivalents als fitxers mapats a memòria que s'han vist a la pràctica 1, amb la diferència que al fitxer mapat no se li dóna cap nom explícit (la implementació particular del mapat anònim depèn del sistema operatiu Unix).

A grans trets, l'algorisme que es proposa implementar és el següent:

1. El procés pare llegeix el diccionari de paraules i crea l'estructura de l'arbre, tal com es fa actualment.

2. El procés pare mapa les dades del nodes de l'arbre a un únic fitxer anònim mapat a memòria. Per fer-ho es fa servir una tècnica anomenada serialització. De forma senzilla, podeu interpretar la serialització com el que heu fet a la pràctica 3 a l'hora de desar i llegir les dades de l'arbre. La serialització serà una tècnica que es farà servir a altres assignatures (com Software Distribuït, per exemple). Els llenguatges com Java inclouen mètodes que permeten serialitzar objectes.

   Per aquesta pràctica se us proporciona el codi necessari per fer la serialització de les dades dels nodes a un fitxer anònim mapat a memòria mantenint l'estructura de l'arbre.

3. Un cop feta la serialització, el procés pare crearà un conjunt de processos fills. Aquests processos fills seran els encarregats de llegir els fitxers de la base dades, extreure les paraules, i actualitzar la informació de les dades dels nodes (que estan compartides entre tots els processos gràcies al fitxer anònim mapat a memòria).

4. Un cop els processos fills han processat tots els fitxers de la base de dades, el procés pare procedeix a la de-serialització. La idea és agafar el fluxe de bytes que hi ha al fitxer anònim mapat a memòria i transformar-lo a nodes de dades. D'aquesta forma l'arbre resultant fa servir les mateixes estructures que heu fet servir a la pràctica 2 i 3. Es proporciona el codi per fer aquesta de-serialització.

Consulteu la secció d'implementació per tenir una proposta de com implementar el codi.

2 Implementació

Aquesta secció dóna alguns consells per tal d'implementar correctament aquesta pràctica. Tingueu en compte que només caldrà modificar el codi de creació de l'arbre (i.e. l'opció 1 del menú). Es proposa seguir el següent ordre a l'hora d'implementar la pràctica. Sou lliures de fer altres implementacions i modificar el codi que se us proporciona a les vostres necessitats.

2.1 Serialització i deserialització de les dades de l'arbre

Començarem la creació de l'arbre fent servir només el procés pare (no es creen processos fills). L'objectiu és comprovar si podeu crear l'arbre fent servir la serialització i deserialització. Teniu totes les funcions que us fan falta al fitxer `tree-to-mmap.c`. La creació de l'arbre es fa de la següent forma

1. El procés llegeix el diccionari de paraules i crea l'estructura de l'arbre, tal com es fa actualment.
2. El procés serialitza les dades de l'arbre fent servir una funció que ja es proporciona, en particular `serialize_node_data_to_mmap`.
3. Procediu a processar els fitxers de la base de dades i comptar quantes vegades apareix cada paraula.
4. El procés deserialitza les dades de l'arbre fent servir `deserialize_node_data_from_mmap`. Aquesta funció agafa les dades serialitzades construeix l'arbre fent servir l'estructura utilitzada a la pràctica 2 i 3. A més, també allibera la memòria associada al mapat de fitxer.

Amb aquest procediment es crearà l'arbre amb un únic procés. Comproveu que l'arbre es crea correctament fent, per exemple, consultes als nodes de l'arbre. Assegureu-vos que valgrind no dóna cap mena d'error.

2.2 Mapat dels noms de fitxer de la base de dades a memòria

Farà falta mapar els noms de fitxer a processar a memòria atès que els processos fills hauran d'accedir als noms de fitxer de la base de dades a processar. Modifiqueu el codi de la creació de l'arbre per comprovar que, fent servir només el procés pare, es pot crear l'arbre correctament.El codi del mapat es proporciona al codi `dbfnames-mmap.c`. Les funcions que es proporcionen són:

1. `dbfnames_to_mmap`: funció que mapa els noms de fitxers a memòria.
2. `get_dbfname_from_mmap`: funció que retorna un punter cap al nom de fitxer i-éssim. La funció retorna NULL si s'intenta accedir fora de la taula.
3. `dbfnames_munmap`: funció que permet alliberar el mapat a memòria.

2.3 Creació de l'arbre fent sevir un únic fill

L'objectiu ara és crear l'arbre fent servir un únic fill. Observeu que encara no hi haurà problemes de sincronització. La proposta és:

1. El procés pare llegeix el diccionari de paraules i crea l'estructura de l'arbre, tal com es fa actualment.

2. El procés pare llegeix els fitxers de la base de dades i els emmagatzema en un fitxer anònim mapat a memòria.

3. El procés pare crea un únic procés fill. Aquest procés fill processarà els fitxers de la base de dades i actualitzarà els camps de les dades dels nodes. Observar que el procés fill podrà accedir al fitxer anònim mapat a memòria atès que el procés fill "hereta", entre altres coses, els fitxers que ha obert el pare.

4. Mentre el procés fill processa els fitxers, el procés pare s'esperarà. Mireu sistemes operatius 1 per recordar com es feia...

5. Un cop el procés fill acaba, el procés pare fa la deserialització i alliberament de la memòria mapada.

Assegureu-vos, de nou, que l'arbre es crea correctament. Comproveu que en fer una consulta

2.4 Creació de l'arbre fent servir múltiples fills

A continuació procedim a crear l'arbre amb múltiples fills. Aquí caldrà tenir en compte que els processos fills hauran d'accedir a "recursos" compartits. Quins són aquests recursos compartits?

La idea és fer una ampliació del punt 2.3 fent servir múltiples fills. En particular

1. El procés pare fa la inicialització de les dades: serialització de l'arbre i mapat dels noms de fitxers.

2. El procés pare crea els processos fills. Els processos fills hauran d'accedir als noms de fitxer assegurant que no es produeixin *race conditions*. El fills extrauran les paraules del fitxer i hauran d'assegurar que tampoc es produeixin *race conditions* a l'hora d'actualitzar les dades dels nodes de l'arbre. Quina solució se us acut? Per inspirar-vos llegiu tot aquest document... en particular, el document que haureu d'entregar. No fa falta implementar la solució més eficient.

3. Mentre els processos fills processen els fitxers, el procés pare esperarà.

4. Un cop els processos fills acaben, el procés pare fa la deserialització i alliberament de la memòria mapada.

Tingueu en compte que l'objectiu en aquesta pràctica és fer un codi multiprocés per crear l'arbre fent que la sincronització es realitzi de forma correcta. Això implica que el resultat que obteniu a l'hora de fer consultes és el mateix que el que s'obté quan l'arbre es crea fent servir només el procés pare. L'objectiu no es fer un codi eficient. Si us sobra temps, us hi podeu dedicar...

3 Entrega

El fitxer que entregueu s'ha d'anomenar `P3_NomCognom1NomCognom2.tar.gz` (o `.zip`, o `.rar`, etc), on `NomCognom1` és el nom i cognom del primer component de la parella i `NomCognom2` és el nom i 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 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. No cal incloure la base de dades ni el fitxer de diccionari. El codi ha de compilar sota Linux amb la instrucció `make`. És recomanable que poseu les funcions en fitxers diferents agrupats per la seva funcionalitat (per exemple, un fitxer per la creació de l'arbre, un altre per llegir i desar l'arbre, ...). El codi font ha d'estar comentat (com a mínim les funcions).

  Per simplificar, es pot suposar que el codi s'executa dins del directori en què està situada la base de dades. El fitxer de diccionari (que pot tenir qualsevol nom) també estarà situat en aquest directori. El codi no tindrà cap paràmetre, atès que tot el control es farà des del menú de l'aplicació

  `practica4`

- El directori `doc` ha de contenir un document (màxim tres pàgines, en format PDF, sense incloure la portada). Es demana respondre a dues preguntes

  1. Un procés fill, a l'hora de processar i actualitzar les dades dels nodes té diverses formes de procedir. Comenteu quina avantatge o problemes pot ocasionar cadascuna de les solucions proposades: a) fer una única secció critica en què estigui inclòs agafar el fitxer a processar, llegir i extreure les paraules, i actualitzar les dades dels nodes, b) fer una secció crítica que, en extreure una paraula, inclogui comprovar si la paraula és a l'arbre i actualitzar el comptador si és necessari, c) fer una secció crítica que, únicament en cas que calgui actualitzar un comptador, bloquegi (tot) l'arbre i actualitzi el comptador corresponent, d) fer una secció crítica que bloquegi cada node de forma independent cada cop que calgui actualitzar el comptador.
  2. Quina és la solució que vosaltres heu implementat? Per què? Heu obtingut alguna millora en l'execució del codi?

La data límit d'entrega està indicat al document de planificació. El codi té un pes d'un **80%** (codi amb funcions comentades, 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.). Tingueu en compte que es comprovarà el bon funcionament del vostre codi fent servir el `valgrind`. El document té un pes del **20%** restant.