Pràctica 3

Comunicació i sincronització entre processos

Sistemes Operatius 1

Març 2022

Índex

1 Introducció

Aquesta pràctica se centra a utilitzar alguns dels mètodes que ens ofereix el sistema operatiu per comunicar processos entre si. Hi ha múltiples mètodes de comunicació entre processos entre els quals podem esmentar les canonades, els fitxers, la xarxa així com els senyals.

En aquesta pràctica es desenvoluparà un esquema de productor-consumidor. Aquest paradigma és un exemple clàssic de sincronització de processos. De forma general, en aquest paradigma de programació hi ha diversos processos. Els productors "produeixen" informació i l'escriuen en un búffer. Amb això se la "entreguen" als consumidors, els quals llegeixen les dades del búffer i realitzen alguna operació sobre aquestes dades. A l'hora d'implementar aquest esquema cal assegurar que el consumidor no intenti agafar dades si el búffer és buit. De manera similar, el productor no pot introduir dades al búffer fins que el consumidor les hagi agafat.

Aquesta pràctica es basa en aquest paradigma, tot i que se'n modifica una mica l'esquema de funcionament per facilitar la sincronització dels productors i els consumidors. A l'assignatura de Sistemes Operatius 2 es tornarà a analitzar amb més detall aquest esquema al tema de programació concurrent.

2 La pràctica

En aquesta pràctica hi haurà un únic productor i dos consumidors, veure figura 1. El productor extreu de cada línia d'un fitxer d'entrada dues columnes a processar. Hi haurà dos búffers de comunicació; un per cada consumidor. El productor introduirà al primer búffer les dades extretes d'una de les columnes, mentre que introduirà al segon búffer les dades extretes de l'altra columna. Cadascun dels consumidors podrà llegir del seu búffer les dades introduïdes pel productor i les processarà. Observar doncs que cada consumidor processarà dades de columnes diferents.

En aquesta pràctica el búffer de comunicació entre el productor i els dos consumidors es realitzarà mitjançant un fitxer independent per a cada consumidor. A més, cal utilitzar mecanisme perquè el productor pugui notificar als consumidors si ha introduït dades al cadascun dels fitxers. De forma equivalent, també caldrà un mecanisme perquè el consumidor notifiqui el productor que els ha llegit. En aquest cas el mecanisme de notificació (i.e. sincronització) es realitza mitjançant senyals, veure figura 1. A l'assignatura de Sistemes Operatius 2 es veuran altres formes d'implementar aquest paradigma de comunicació.

2.1 El paradigma productor-consumidor

El productor extreu les dades del fitxer `data.csv`. Es tracta d'un fitxer de text que s'ha obtingut de http://publish.illinois.edu/dbwork/open-data. El fitxer original té 14,7 milions línies de text i aquí s'ha retallat a 100.000 línies (99999 línies de dades més la capçalera). El fitxer conté dades sobre viatges en taxi a la ciutat de Nova York. Cada línia del fitxer conté les dades d'un viatge i les dades aquestes està separades entre si per comes (Comman Separated Values, CSV).

El productor, de forma iterativa, llegeix cada línia del fitxer i n'extreu les columnes 8 i 9 que indiquen, respectivament, quants passatgers anaven al taxi i quant ha durat cada viatge, en segons. Els valors de la columna 8 seran escrits al fitxer `col8.bin`, mentre que els valors de la columna 9 seran escrits al fitxer `col9.bin`. Cada vegada que el productor hagi introduït N valors a cadascun dels fitxers realitzarà una notificació, mitjançant un senyal, als consumidors. En particular, el productor utilitzarà el senyal `SIGUSR1` per fer la notificació al consumidor que processa les dades del

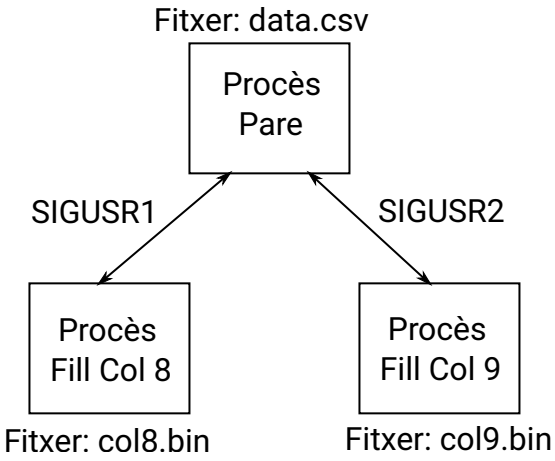

Figura 1: Esquema del productor-consumidor a implementar. Hi ha un productor (procés pare) i dos consumidors (els dos processos fills). La sincronització entre els processos es realitza mitjançant els senyals `SIGUSR1` i `SIGUSR2`. El búffer de comunicació són els fitxers `col8.bin` i `col9.bin`. Veure text per a més detalls.

fitxer `col8.bin`, mentre que utilitzarà `SIGUSR2` per notificar al consumidor que processa les dades del fitxer `SIGUSR2`.

Els consumidors notificaran al productor, mitjançant els senyals `SIGUSR1` i `SIGUSR2`, la recepció que han rebut el senyal per part del productor. Un cop s'hagi fet la notificació els consumidors podran llegir els N elements que han estat introduïts (pel productor) en el fitxer que processa.

Observar que en aquest cas es treballa amb fitxers i, per tant, el productor podrà afegir elements als fitxers `col8.bin` i `col9.bin` mentre els consumidors van llegint elements del fitxer a mesura que el productor els vagi notificant que han estat introduïts N elements nous al fitxer. El consumidor no sap amb antelació quants blocs d'N dades us enviarà el productor: l'aplicació del consumidor ha de realitzar el processat, doncs, per poder rebre milions de blocs d'N dades.

Arribarà un moment en què el productor ha llegit totes les dades del fitxer i, consegüentment, no hi haurà més dades a llegir. El productor haurà de notificar d'alguna forma als consumidors que no hi ha més dades a llegir. Com es farà això? Es dona resposta a aquesta pregunta a la següent subsecció.

Quan el consumidor sàpiga que no hi ha més dades a processar (i.e. no n'hi ha més dades a llegir) imprimirà per pantalla la mitjana dels valors que ha llegit del fitxer. En concret, el consumidor que llegeix les dades del fitxer `col8.bin` imprimirà per pantalla la mitjana del nombre de passatgers, mentre que el consumidor que llegeix les dades del fitxer `col9.bin` imprimirà la mitjana de durada del viatge.

2.2 L'algorisme a implementar

A la figura 1 es pot veure l'esquema dels processos involucrats en el paradigma del productor-consumidor. El pare és el productor mentre que hi haurà dos fills, els quals faran la funció de consumidor. Se suposa conegut, tant pel pare com pels fills, el valor d'N.

A continuació es detalla l'algorisme a implementar:

1. El productor obre els fitxers `col8.bin` i `col9.bin` per creació i escriptura (en crear-los estaran buits). El productor notificarà als consumidors, mitjançant els senyals `SIGUSR1` i `SIGUSR2`, que els fitxers han estat creats. Els consumidors respondran al pare, respectivament, mitjançant els senyals `SIGUSR1` i `SIGUSR2` que han rebut la notificació. Els consumidors podran llavors obrir el fitxer corresponent per lectura.

2. El productor llegeix de forma iterativa cadascuna de les línies del fitxer de dades d'entrada (ignorant la capçalera) i extreu les columnes 8 i 9, les quals són escrites als fitxers `col8.bin` i `col9.bin` respectivament. Un cop hagi introduït N elements als fitxers envia el senyal `SIGUSR1` i `SIGUSR2` als consumidors corresponents.

3. Cada consumidor rep, respectivament, els senyals `SIGUSR1` i `SIGUSR2`. Aquests notifiquen al productor que han rebut el senyal mitjançant els senyals `SIGUSR1` i `SIGUSR2` respectivament. Cada consumidor podrà, aleshores, llegir les N dades del seu fitxer. A més, el productor, per la seva banda, podrà anar llegint dades del fitxer d'entrada i les podrà anar introduint als fitxers `col8.bin` i `col9.bin` mentre els consumidors llegeixen les dades que els corresponen. El productor podrà enviar una nova notificació d'escriptura d'N elements una vegada hagi rebut la notificació, per part de cada un dels consumidors, que ha estat rebuda la notificació enviada anteriorment. Observar doncs que el productor pot anar escrivint al fitxer a mesura que els consumidors van llegint el fitxer.

4. Aquest procés anterior s'itera fins que el productor ja hagi llegit totes les línies del fitxer d'entrada. Un cop el productor ja no tingui més dades a introduir als fitxers `col8.bin` i `col9.bin` procedirà a tancar els fitxers i enviarà als consumidors, de nou, els senyals `SIGUSR1` i `SIGUSR2`. Observeu que aquest senyal s'envia encara que no s'hagin escrit realment N elements a cadascun dels fitxers. Els consumidors, per la seva part, rebran la notificació (i confirmaran la recepció al productor) i llegiran les dades. Els consumidors, en intentar llegir N dades, detectaran que han arribat a final de fitxer. Els consumidors sabran en conseqüència que no hi ha més dades a llegir.

5. Un cop no hi hagi més dades a llegir, els consumidors imprimiran per pantalla la mitjana del nombre de passatgers (fitxer `col8.bin`) i la mitjana del temps de viatge (fitxer `col9.bin`). És important recordar que els consumidors no saben amb antelació quants blocs d'N elements enviarà el productor de manera que no és adequat que els consumidors emmagatzemin totes les dades que reben, sinó que cada cop que un consumidor llegeixi un bloc d'N dades ha de processar-lo i descartar-lo.

6. Quan els consumidors hagin finalitzat la seva execució el productor esborrarà de disc els fitxers `col8.bin` i `col9.bin` i finalitzarà.

3 Planificació i implementació

Juntament amb aquesta pràctica es lliura una plantilla a partir de la qual es pot començar a treballar. Aquesta plantilla inclou el codi per extreure la columna 8 i 9 d'un fitxer. Es recomana utilitzar aquesta plantilla ja que aquesta forma únicament caldrà concentrar-se en els punts primordials i bàsics de aquesta pràctica: la sincronització de processos mitjançant senyals i la comunicació entre processos mitjançant els fitxers.

Al codi plantilla que es lliura es llegeixen totes les línies del fitxer de dades i es calcula la mitjana de passatgers així com la mitjana del temps de durada del viatge.

Executeu el codi i comproveu el resultat que us dóna quan es llegeix tot el fitxer `data.csv` proporcionat amb la pràctica. Si es vol, també es pot provar amb el fitxer de 14,7 milions de línies que es pot descarregar de l'enllaç proporcionat anteriorment (el fitxer correspon a `trip_data_1.csv`).

A continuació es proposa reescriure el codi perquè utilitzi el paradigma del productor-consumidor tal com s'ha descrit anteriorment, vegeu figura 1. Com s'ha comentat a la secció 2.2 en aquesta pràctica el productor serà el pare mentre que hi haurà dos consumidors. Un dels consumidors processarà els elements de la columna 8 i l'altre els de la columna 9. El programa serà el mateix, la diferència rau en el fet que pare i fills executen funcions diferents. El pare executarà, per exemple, una funció anomenada `productor` mentre que els fills executaran, per exemple, les funcions `consumidor_col8` i `consumidor_col9`.

S'indiquen a continuació una sèrie de punts a tenir en compte a l'hora de implementar la pràctica, així com uns passos a seguir per implementar la pràctica.

3.1 Sincronització mitjançant senyals

La sincronització entre el productor i els dos consumidors es farà mitjançant els senyals `SIGUSR1` i `SIGUSR2`. Quan el productor enviï aquesta senyal als consumidors aquests hauran de respondre per notificar que han rebut el senyal, veure figura 1. El productor no podrà tornar a enviar els senyals `SIGUSR1` i `SIGUSR2` fins que no hagi rebut la notificació per part dels consumidors que l'anterior senyal ha sigut rebut.

La raó de procedir d'aquesta forma rau en el fet que el mecanisme que el sistema operatiu utilitza per gestionar els senyals és binari. Internament, el sistema operatiu emmagatzema si un procés ha rebut un determinat un senyal. No hi ha cap comptador que emmagatzemi el nombre de vegades que un procés hagi rebut un senyal. Això implica que si s'envia múltiples vegades el mateix senyal a un procés sense que aquest darrer tingui temps per gestionar-los es podrien "perdre" un munt de senyals atès que, com s'ha comentat, el sistema operatiu utilitza una variable binària (i.e. un *flag*) per saber si s'ha rebut un senyal o no. És per això que en aquesta pràctica es proposa que els consumidors confirmin al productor que han rebut el senyal abans que el productor torni a enviar un nou senyal. A l'assignatura de Sistemes Operatius 2 es veuran altres mecanismes de senyalització més avançats en què es fa servir una cua associada als senyals que s'envien.

Recordar que el productor envia als consumidors el senyal `SIGUSR1` i `SIGUSR2` quan a) el productor obre el fitxers de comunicació per escriptura per indicar als consumidors que el poden obrir per lectura, b) el productor ha escrit N elements als fitxers per indicar als consumidors que tenen noves dades al fitxer.

En fer proves amb la implementació del codi proposat s'han detectat problemes en utilitzar la funció `pause`, que bloqueja un procés fins que rep un senyal. S'han detectat casos en què el senyal s'envia abans que s'executi `pause`. La funció `pause` bloqueja el procés i espera a rebre un senyal. Si un procés rep el senyal abans d'executar `pause` no el té en compte i, en conseqüència, el aplicació queda bloquejada (això es pot veure ja que deixa de consumir CPU). Per això es recomana utilitzar l'anomenada *espera activa* per gestionar la sincronització entre el productor i el consumidor en comptes d'utilitzar la funció `pause`. L'espera activa implica un codi similar al que es mostra a continuació

```
while (!sigusr1) {};
segusr1 = 0;
```

A l'exemple anterior el codi espera de forma activa, consumint CPU i sense bloquejar-se, per rebre el senyal `SIGUSR1`. Suposem que `sigusr1` es una variable global inicialitzada a 0, i que en rebre el senyal la funció que gestiona els senyals posa `sigusr1` a 1. En el codi proposat no es fa cap `pause`: en rebre el senyal es posa la variable `sigusr1` a 1 i el codi surt del bucle `while` per tornar a posar de forma immediata la variable `sigusr1` a 0. Observar que el codi funcionarà encara que es rebi (i es processi) el senyal abans d'entrar a l'espera activa.

La solució proposada és adequada en aquest cas atès que sabem que els consumidors no hauran d'esperar gaire per rebre el senyal per part del productor. No es malgastarà doncs gaire temps de CPU de forma innecessària.

En aquesta pràctica caldrà fer servir aquesta tècnica d'espera activa als dos consumidors i al productor. Recordar que el productor és el pare dels dos consumidors. *Atès el context especificat, es poden fer servir les mateixes variables globals (per exemple, `sigusr1` i `sigursr2`) als tres processos per gestionar la recepció dels senyals? O farà falta que cada procés tingui la seva pròpia variable global (per exemple, `sigursr_pare`, `sigusr1_consumidor1`, `sigusr1_consumidor2` i l'equivalent per a `sigusr2`)?*

Per començar aquesta pràctica es proposa començar per la part de sincronització a l'hora d'obrir els dos fitxers que s'utilitzaran per transferir les dades del productor al consumidor. Com s'ha comentat abans, es proposa que el productor comenci per obrir el fitxer per escriptura, enviï a continuació el senyal de sincronització de forma que els consumidors sàpiguen que poden obrir el fitxer per lectura.

L'obertura dels fitxers per part del productor i els consumidors s'ha de fer fent servir la crida a sistema `open`. Observar que no té sentit, en el context d'aquesta pràctica, fer servir les crides a la llibreria d'usuari per fer la comunicació interprocés. *Podeu raonar per què cal fer servir les crides a sistema i no es poden fer servir les crides a la llibreria d'usuari?*

3.2 Transferència en blocs de mida N

En aquesta secció es descriu amb detall la implementació del algorisme descrit a la secció 2.2.

El productor llegeix del fitxer d'entrada i extreu la informació de les columnes 8 i 9. Aquestes dues dades han de ser emmagatzemades als fitxers `col8.bin` i `col9.bin`.

L'escriptura i lectura de les dades al fitxer s'ha de realitzar de forma binària fent servir les crides a sistema `write` i `read`. En escriure o llegir les dades en blocs d'N elements es recomana fer-ho amb una única crida a sistema en comptes de fer la crida per cadascun dels elements (i.e. fer N crides a sistema). *Podeu imaginar per què és més convenient procedir de la forma proposada?*

Tal com s'ha comentat abans es proposa que els consumidors notifiquin que han rebut el senyal abans de començar a llegir les dades de disc. Això permet que el productor vagi llegint les dades del fitxer d'entrada (i també podrà escriure a `col8.bin` i `col9.bin`) mentre els consumidors llegeixen el bloc les dades del seu fitxer.

Observar que el darrer bloc que s'escrigui a `col8.bin` i `col9.bin` no tindrà, necessàriament, una mida d'N dades. És molt important que el productor tanqui el seu extrem de comunicació (i.e. tancar els fitxers `col8.bin` i `col9.bin` amb la crida a sistema `close`). D'aquesta forma, quan els consumidors arribin a final de fitxer sabran que no hi ha més dades a llegir. Un cop hagin arribat a final de fitxer els consumidors podran imprimir les mitjanes.

3.3 Depuració del codi

És habitual utilitzar les crides a `printf` per tal de depurar el codi. Tot i que pot ser útil en alguns casos, es convenient mencionar que la introducció d'aquestes instruccions pot fer que canviïn els punts en què els processos productor i consumidors facin un canvi de context. Per tant, és possible que posant `printf` el codi us funcioni correctament mentre que en treure'ls no ho faci. Això és indicació, molt possiblement, que la sincronització no s'està realitzant correctament. Caldrà analitzar doncs amb detall el codi per tal de trobar la font del problema: fer proves amb diferents valors d'N, provar d'enviar un bloc, etc.

4 Entrega

El lliurament consisteix en dues parts: **el codi font** (80% de puntuació) i **l'informe** (20% de puntuació). Les dues parts s'han de comprimir en un **únic fitxer ZIP** i lliurar tal com s'indica a la tasca del campus.

4.1 Codi font

Es demana lliurar els següents fitxers:

- El codi font que implementi la funcionalitat descrita a la secció 2.2. Tot el codi es pot implementar en un únic fitxer C si així es vol.
- Un `Makefile` que permeti compilar el codi. L'estructura d'aquests fitxers està descrita en un document del campus, a la secció de teorico-pràctica.

El codi font tindrà dos arguments d'execució: el fitxer a processar i el valor d'N.

```
$ ./practica3 <fitxer_a_processar> <valor_N>
```

El codi ha d'estar adaptat per poder processar qualsevol mida de fitxer, sigui el que s'adjunta amb aquesta pràctica com el fitxer amb més de 14,7 milions de línies.

A la implementació es pot suposar que el processos productor i consumidors únicament utilitzaran els senyals `SIGUSR1` i `SIGUSR2` per a realitzar la sincronització descrita a les seccions anteriors.

Es demana que la implementació utilitzi els mecanismes l'espera activa així com una escriptura i lectura amb una única crida a sistema (és a dir, que els N elements s'escriguin o es llegeixin amb una única crida a sistema). Reviseu la secció 3 per a més informació al respecte. No es permet utilitzar funcions `sleep` o similars per a la implementació del codi.

Amb l'enunciat de la pràctica disposeu del codi compilat de la solució. D'aquesta forma podreu veure el que s'espera de vosaltres. Observar que el codi no imprimeix cap missatge fins al final del processament del fitxer.

4.2 Informe

L'informe a lliurar ha d'estar en **format PDF o equivalent (no s'admeten formats com odt, docx,...)**. Una bona manera d'escriure un informe tècnic és dividir-ho en tres parts: **introducció, informe/proves realitzades i conclusions.**

A la part d'introducció es descriu breument el problema solucionat (i.e. un resum del que proposa aquesta pràctica). A la part de l'informe/proves es demana mostrar:

- Les proves que s'han fet per assegurar el bon funcionament del codi. És suficient mostrar proves amb el fitxer `data.csv` proporcionat en aquesta pràctica. Es proposa mostrar que el codi implementat dóna els mateixos valors mitjans que el codi inclòs amb aquesta pràctica per a diferents valors d'N.

- Doneu resposta a les preguntes que es realitzen a les seccions 3.1 i 3.2. El text associat a les preguntes està emfatitzat.

Sigueu breus i clars en els comentaris i experiments realitzats, no cal que us estengueu al text escrit. No es necessari que expliqueu el codi font llevat que hi hagi alguns detalls que vulgueu esmentar respecte la vostra implementació.

Finalment, a la part de conclusions es descriuen unes conclusions tècniques de les proves realitzades. Per acabar, es poden incloure conclusions personals.

En cas que vulgueu incloure captures de pantalla en comptes d'incloure els resultats dels experiments en format text, assegureu-vos que el text de la captura es pot llegir bé (és a dir, que tingui una mida similar a la resta del text del document) i que totes les captures siguin uniformes (és a dir, que totes les captures tinguin la mateixa mida de text).

El document ha de tenir una longitud màxima de 4 pàgines (sense incloure la portada). El document s'avaluarà amb els pesos següents: proves realitzades i comentaris associats, un 60%; escriptura sense faltes d'ortografia i/o expressió, un 20%; paginació del document feta de forma neta i uniforme, 20%.