UNIVERSITAT DE BARCELONA

# Pràctica 3. Comunicació i sincronització entre processos

Noah Márquez Vara
Alejandro Guzman Requena

6 maig 2022

# 1 INTRODUCCIÓ

La present pràctica, tal i com indica el seu títol, es centra en la comunicació i sincronització entre processos del sistema operatiu.

Hem vist a les classes de teoria i teoricopràctica que hi ha múltiples mètodes de comunicació entre processos: les canonades, els fitxers, la xarxa i els senyals.

En aquesta pràctica hem desenvolupat un esquema productor-consumidor (certament, eren dos consumidors). Aquest esquema és bàsic pel que fa a la sincronització de processos.

Els productors "produeixen" informació (en el nostre cas extreuen les dades de les columnes 8 i 9 del fitxer donat) i l'escriuen en un búffer (dos fitxers en el nostre cas: *col8.bin* i *col9.bin*). Amb això se l' "entreguen" als consumidors, els quals llegeixen les dades del búffer i realitzen una operació sobre aquestes (en el nostre cas, van sumant els valors de les corresponents columnes).

És important a l'hora d'implementar aquest paradigma tenir clar que el consumidor no pot intentar agafar dades si el búffer és buit i que el productor no pot introduir dades al búffer fins que el consumidor les hagi agafat.
Per això, necessitem un mecanisme perquè el consumidor notifiqui el productor de que ha llegit les dades. En aquest cas el mecanisme de notificació (i.e. sincronització) es realitza mitjançant senyals.

Adjuntem la figura de l'enunciat de la pràctica per entendre tot l'esmentat anteriorment.

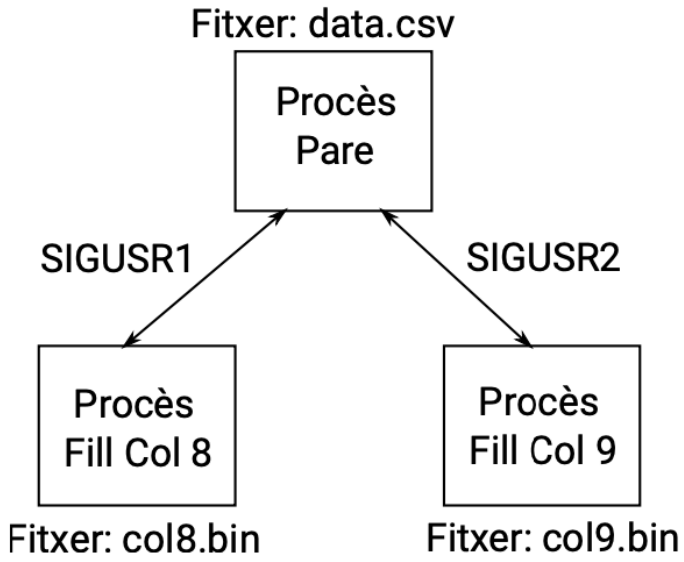


Figura 1.1: Esquema del productor-consumidor a implementar

# 2 PROVES REALITZADES

Per tal de comprovar el correcte funcionament del codi, hem fet diverses proves amb el fitxer *data.csv* que ens era proporcionat en aquesta pràctica.
A continuació mostrarem algunes de les proves realitzades, ja que en totes el resultat era l'esperat i no es vol estendre el document.

```
$ ./practica3 data.csv 400
Col9: Aplication read 99999 elements
Col9: Mean of trip time: 650.639771 secs
Col8: Aplication read 99999 elements
Col8: Mean of passengers: 2.163002
Pare finalitza!

$ ./practica3 data.csv 8000
Col9: Aplication read 99999 elements
Col9: Mean of trip time: 650.639771 secs
Col8: Aplication read 99999 elements
Col8: Mean of passengers: 2.163002
Pare finalitza!

$ ./practica3 data.csv 1500
Col9: Aplication read 99999 elements
Col9: Mean of trip time: 650.639771 secs
Col8: Aplication read 99999 elements
Col8: Mean of passengers: 2.163002
Pare finalitza!

$ ./practica3 data.csv 2
Col9: Aplication read 99999 elements
Col9: Mean of trip time: 650.639771 secs
Col8: Aplication read 99999 elements
Col8: Mean of passengers: 2.163002
Pare finalitza!
```

Amb les proves realitzades hem pogut comprovar que els resultats coincideixen amb el que havíem d'esperar.

## 3 PREGUNTA SECCIÓ 3.1

S'han de fer servir les crides a sistema davant les crides a la llibreria d'usuari ja que són més adequades (tot i que es podria realitzar perfectament amb aquestes també). Això és degut a que al cap i a la fi, les crides a sistema *fwrite* i *fread* acaben invocant a les crides a sistema *write* i *read*. A més, tal i com hem vist a teoria, les crides a sistemes són molt més adequades per a la comunicació interprocés (propòsit d'aquesta pràctica), ja que les crides a la llibreria d'usuari utilitzen el búffer propi.

Atès el context especificat a la pràctica no ens farà falta que cada procés tingui la seva pròpia variable, ja que perdria les possibilitats que ens ofereix l'espera activa per gestionar la sincronització entre el productors i els dos consumidors.

Amb dues variables globals (en el nostre cas, **sigusr1** i **sigusr2**) ja podem fer que en nostre codi esperi de forma activa, consumint CPU i sense bloquejar-se (evitem els problemes que donaria la funció **pause** en cas de rebre un senyal abans d'executar la instrucció), per tal de rebre els senyals **SIGUSR1** i **SIGUSR2** en aquest cas.

A més, sabem que la solució de l'espera activa és molt adequada en el nostre cas donat que

sabem que els consumidors no hauran d'esperar gaire per rebre el senyal per part del productor. Evitem així malgastar gaire temps de CPU de manera innecessària.

## 4 PREGUNTA SECCIÓ 3.2

En escriure o llegir les dades en blocs d'N elements és molt recomanable 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). Això és degut a que, ja pel simple fet d'estar fent servir les crides a la llibreria d'usuari (***fwrite*** i ***fread***) ens estariem estalviant moltes crides a sistema, ja que aquestes crides a la llibreria d'usuari utilitzen un búffer propi de procés que no es visible a cap altre, d'aquesta manera es permet realitzar un nombre menor de crides a sistema i, d'aquesta manera, reduir el temps d'execució.

Però aquest no és el nostre cas, nosaltres hem de fer ús de les crides a sistema ***write*** i ***read***, així que haurem de inserir les dades en un búffer i escriure aquest búffer en els fitxers, per tal d'estalviar-nos un gran nombre de crides a sistema, millorant així l'eficiència del nostre programa.

## 5 CONCLUSIONS

Durant el transcurs de la pràctica ens hem anat trobant amb diversos problemes/errors, sobretot centrats en la part de la comunicació mitjançant les senyals, ja que hi havia alguns moments en que o bé el productor o bé algun dels consumidors no rebia correctament la informació. També vam tenir problemes a l'hora d'executar dos *forks* seguits creant només dos fills d'un mateix pare. Un cop solucionats aquests problemes però, la pràctica va ser molt profitosa.

Gràcies a aquesta pràctica hem pogut veure i treballar una de les vàries maneres de comunicació interprocés i a més hem pogut conèixer el paradigma del productor-consumidor.

La mateixa pràctica es podria haver fet amb ús de canonades com a búffer i no amb fitxers com hem realitzat nosaltres.

Ha estat de molta ajuda per tal d'acabar d'entendre de manera completa com es poden comunicar els processos dins del nostre sistema operatiu.