ESCOLA POLITÈCNICA SUPERIOR
UNIVERSITAT DE LLEIDA

Curs 2020-21

PRA1. Processos, *pipes* i senyals: El 7,5

Publicació: Dimecres 07/OCT./2020
Entrega: Dimecres, 04/NOV./2020

1. Objectius

Els objectius de la pràctica són:

- Comprensió crides a Sistema UNIX.
- Conèixer les funcions de creació de processos *fork* i *exec*. Entendre com funcionen, com es criden, que retornen.
- Conèixer els mètodes de sincronització de processos *exit* i *wait* i la seva relació.
- Veure els *pipes* com a mecanismes de comunicació entre processos.
- Treball amb els senyals entre processos.

2. Presentació

MOLT IMPORTANT: L'enviament de codi que no compili correctament, suposarà suspendre TOTA la pràctica.

No s'acceptaran pràctiques entregades fora del termini.

La presentació d'aquests exercicis és **obligatòria** i pot realitzar-se per parelles.

Cal presentar els fitxers amb el codi font realitzat. Tota la codificació es farà exclusivament en llenguatge C.

La data límit serà les **17:10 h** del **dimecres 04/NOVEMBRE/2020**, dia en el qual **es farà una prova de validació INDIVIDUAL, de manera presencial o virtual** al laboratori.

Per presentar la pràctica adreceu-vos a l'apartat Activitats del Campus Virtual a l'assignatura de Sistemes Operatius, aneu a l'activitat **PRA1. Processos, *pipes* i senyals: El 7,5 – Pralab 2** i seguiu les instruccions.

Tots els autors de la pràctica heu de pujar al Campus Virtual un fitxer agrupat i comprimit amb tar, que contingui el codi font dels vostres programes DEGUDAMENT COMENTAT i una memòria amb els aspectes que considereu importants. Per generar aquest fitxer podeu usar l'ordre `tar -zcvf COGNOM1_1_COGNOM1_2_PRA1_SO.tgz <llista_fitxers>`.

El nom del fitxer comprimit ha de ser: COGNOM1_1_COGNOM1_2_PRA1_SO.tgz on:

- COGNOM1_1: el 1r cognom d'un membre del grup.
- COGNOM1_2: el 1r cognom de l'altre membre del grup.

Comenceu els vostres programes amb els comentaris:

```
/* ------------------------------------------------------------------
 PRA1 : [TODO]
 Codi font : <nom>.c

 Nom complet autor1.
 Nom complet autor2.
 ------------------------------------------------------------------ */
```

3 Restriccions en l'ús de funcions C

Per realitzar la pràctica heu d'utilitzar les crides a sistema d'Unix. **No es podran utilitzar funcions de C d'entrada/sortida** (`getc`, `scanf`, `printf`, ...), en el seu lloc s'utilitzaran les crides a sistema `read` i `write`. Sí que podeu utilitzar funcions de C per al formatatge de cadenes (`sprintf`, `sscanf`, ...).

4 Enunciat

Es vol simular el joc de cartes del set i mig entre un seguit de jugadors i un crupier.

Per facilitar el desenvolupament de la pràctica us proposem un seguit de passos incrementals fins a la solució final.

També us facilitem el fitxer PRA1_src.tgz que conté els següents fitxers:

- El codi font del procés jugador del primer pas (`jugador.c`).
- Una primera versió del procés crupier (`crupier.c`)
- Fitxers executables (`jugador2ok, croupier1ok, ...`) que es comporten com hauria de fer-ho la vostra aplicació a cada pas. Utilitzeu-los per comprovar la correctesa de la vostra solució.

Per desagrupar i descomprimir aquest fitxer cal executar la comanda `tar -zxvf PRA1_src.tgz`.

Pas 0: Familiarització amb els programes facilitats

Aquest pas té com a objectiu familiaritzar-se amb els programes facilitats:

- `jugador.c` : Simula una partida del set i mig per part d'un únic jugador. Aleatòriament, genera les cartes i decideix si es planta. Per compilar-lo i executar-lo podeu fer servir les comandes:

  ```
  $gcc -Wall jugador.c -o jugador
  $./jugador
  ```

  Observeu que la darrera línia escrita pel programa es fa en color vermell, groc o verd.

- `crupier.c` : Espera com a paràmetre el nombre de processos jugadors que ha de crear. El programa executa els jugadors de forma seqüencial, és a dir, crea el jugador i-èssim únicament quan el jugador (i-1)-èssim ha finalitzat la seva partida. Un cop compilat, podeu provar el programa fent, per exemple:

```
$./crupier 5
```

Observeu que les línies mostrades per un jugador no es barregen amb les línies mostrades pels altres jugadors i que crupier escriu algunes línies en color blau.

Pas 1: Execució concurrent dels processos jugadors [15%]

Copieu `crupier.c` sobre `crupier1.c` i modifiqueu `crupier1.c` de forma que el jugador i-èssim es creï encara que el jugador (i-1)-èssim no hagi acabat l'execució (per tant, es podran barrejar línies impreses per diferents jugadors).

Observacions:

- El crupier ha de finalitzar la seva execució quan tots els jugadors hagin acabat.
- El comportament d'aquesta aplicació hauria de ser com el de l'executable `crupier1ok` (recordeu que disposeu d'un executable de referència per a cada pas de la pràctica).
- No heu de modificar el programa `jugador.c`.

Pas 2: Determinació del guanyador utilitzant les crides `exit`/`wait` (*exit code*)[25%]

Fins ara, el crupier no sap com han acabat els processos jugadors. Heu de fer que els jugadors informin el crupier de la seva puntuació final de forma que aquest pugui decidir qui ha guanyat (en cas d'empat assumiu que guanya el jugador de *pid* inferior). Per fer aquesta comunicació, els processos jugadors utilitzaran el codi d'acabament (paràmetre de la crida al sistema `exit`). Copieu `crupier1.c` i `jugador.c` sobre `crupier2.c` i `jugador2.c` respectivament, i feu els canvis necessaris sobre els nous fitxers.

Pas 3: Determinació del guanyador utilitzant *pipes* [25%]

Ara utilitzareu *pipes* per tal que els processos jugadors que no s'han passat informin el procés crupier amb quina puntuació han acabat. Quan el crupier hagi recopilat la informació de tots els processos jugadors que no s'han passat, el crupier mostrarà qui ha guanyat.

Copieu `crupier2.c` i `jugador2.c` sobre `crupier3.c` i `jugador3.c` respectivament, i feu els canvis necessaris sobre els nous fitxers.

Observacions:

- Aquest problema es pot resoldre utilitzant una única *pipe* o utilitzant tantes *pipes* com a processos jugadors. Totes dues opcions són vàlides, escolliu la que us sembli millor.
- Per comprovar que la vostra solució és correcta, mostreu també qui és el guanyador utilitzant el mecanisme que heu implementat al pas anterior; el guanyador ha de ser el mateix en tots dos casos.

Pas 4: Enviament de les cartes per part del procés crupier [30%]

Fins ara, cada procés jugador genera aleatòriament les cartes. Ara farem que sigui el procés crupier qui enviï les cartes als jugadors per torns (a cada torn es repartirà una carta a cada jugador que encara continuï jugant). Per fer-ho, el crupier farà servir tantes *pipes* com a processos jugadors (a més de les que heu fet servir al pas anterior).

Per enviar una carta a un jugador, el crupier l'escriurà a la *pipe* corresponent i esperarà rebre un senyal enviat pel jugador que ha rebut la carta; si rep un `SIGUSR1` indica que el jugador voldrà més cartes, si rep un `SIGUSR2` indicarà que el jugador no en vol rebre més. Un cop rebut el senyal, el crupier podrà enviar la carta al següent jugador.

Copieu `crupier3.c` i `jugador3.c` sobre `crupier4.c` i `jugador4.c` respectivament, i feu els canvis necessaris sobre els nous fitxers.

Observacions:

- És necessari que el crupier inicialitzi el generador de nombres aleatoris de forma anàloga a com ho fa el jugador.

5 Avaluació

Es valorarà l'ús correcte de les crides a sistema, el control d'errors en la seva utilització i la correcta programació, estructura i funcionament de l'aplicació.

Cal que els programes estiguin correctament tabulats diferenciant els diferents blocs de codi.

6 Annexos

Annex A: Fòrum per a dubtes al Campus Virtual

Fora de l'aula, l'única via per a plantejar qualsevol dubte és l'apartat **Fòrums** a l'espai de l'assignatura al Campus Virtual, on hi ha disponible el fòrum **Dubtes Pràctiques → PRA1. Processos, *pipes* i senyals: El 7,5**, on podeu participar exposant els vostres dubtes i suggerint respostes.