ESCOLA POLITÈCNICA SUPERIOR
UNIVERSITAT DE LLEIDA

Curs 2019-20

PRA1. Processos, *pipes* i senyals: Primers

Publicació: Dilluns 07/OCT/2019
Entrega: Dilluns, 28/OCT/2019

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 **17.00 h** del **dilluns 28/OCTUBRE/2019**, dia en el qual **es farà una entrega presencial** al laboratori **responent a les preguntes del professor**.

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: Primers – Pralab 1** i seguiu les instruccions.

Pugeu al Campus virtual un fitxer agrupat i comprimit amb tar que contingui el codi font dels vostres programes DEGUDAMENT COMENTAT i una memòria amb la resposta a la primera pregunta i les consideracions que considereu importants del segon exercici.
Per generar aquest fitxer podeu usar l'ordre `tar -zcvf COGNOM1_1_COGNOM1_2_PRA1_SO.tgz <directori_PRA1>`.

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`, ...).

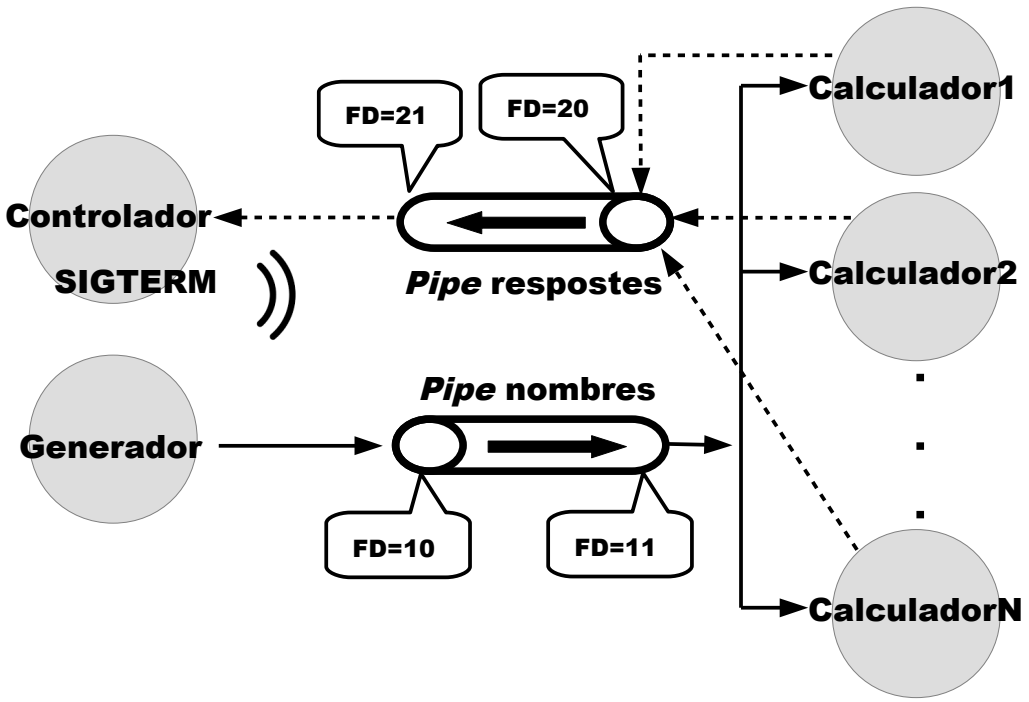
4 Enunciat

A la primera pràctica heu de programar una aplicació multiprocés que determini quins d'una sèrie de nombres són **primers**.

L'aplicació ha de constar de **tres tipus** de processos:

- **Controlador**: procés pare del qual hi haurà una única instància.
- **Generador**: procés fill (de controlador) del qual hi haurà una única instància.
- **Calculador**: procés fill (de controlador) dels quals hi haurà N instàncies.

Per a la comunicació i sincronització dels diferents processos, s'usaran canonades (*pipes*), senyals i la parella de crides *exit*-*wait*, seguint aquesta arquitectura:



Detall de què ha de fer cada procés:

Controlador (**`controlador.c`**):

1. Rep per paràmetre el nombre de processos calculadors a crear.
2. Crea els ***pipes***.
3. Demana a l'usuari un nombre enter N>=2.
4. És el procés pare que crea la resta de processos fill (generador i calculadors segons el punt 1).
5. Després de crear els fills, llegirà del *pipe* de respostes fins que li arribi un EOF (es tanquin els descriptors d'escriptura del *pipe*). Per aquest *pipe* rebrà els resultats de la determinació de quins nombres són primers. Per cada nombre, rebrà una variable de tipus estructura amb la informació:
   - PID del procés que ha fet la comprovació.
   - Nombre comprovat.
   - Resultat: si el nombre és primer o no.

   Quan s'hagi rebut aquesta informació s'haurà de mostrar per pantalla, de manera que quedi clar el PID del procés calculador, el nombre comprovat i si és primer o no.

   A més, també caldrà dur el compte de nombres primers.
6. Quan hagi arribat el final de fitxer (EOF) del *pipe* de respostes, enviarà el senyal SIGTERM als seus fills i esperarà que aquests acabin. Com a codi de finalització dels fills, rebrà quants nombres enters han trobat cadascú d'aquests.
7. Un cop acabin els fills, mostrarà el nombre de primers de la sèrie de nombres fruit de la informació rebuda pel *pipe* de respostes (el que s'especifica al punt 5) i el nombre de primers resultat de la suma del codi d'acabament dels diferents processos calculadors.

Generador (**`generador.c`**):

1. Rep per paràmetre, enviat pel controlador, el nombre enter N que aquest ha demanat.
2. Escriu al *pipe* de nombres la seqüència de nombres de 2 a N.
3. Tanca el descriptor d'escriptura del *pipe* de nombres.
4. Es queda a l'espera de rebre el senyal SIGTERM.
5. Quan rebi el SIGTERM acabarà fent un *exit*.

Calculadors (**`calculador.c`**):

1. Llegeixen números del *pipe* de nombres fins que llegeixin un EOF.
2. Comproven si el nombre llegit és primer.
3. Un cop determinat si el nombre és primer, escriuran al *pipe* de respostes la informació relativa a la determinació de si el nombre és primer. Per a fer-ho usaran una variable de tipus estructura amb la informació:
   - PID del mateix procés.
   - Nombre comprovat.
   - Resultat: si el nombre és primer o no.
4. Un cop rebin el EOF del *pipe* de nombres, tancaran els descriptors dels *pipes* que tinguin oberts, es quedaran a l'espera de rebre el senyal SIGTERM.
5. Quan rebin el SIGTERM acabaran fent un *exit*, retornant com a codi d'estat el nombre de primers que han calculat del total de nombres llegits.

```
# Exemple de sortida de l'execució
$ ./controlador 3 # 3 processos calculador
Calcular nombres primers de 2 a? 7 # Usuari introdueix 7
Controlador: rebut del Calculador PID 2113: nombre 3 es primer? S
Controlador: rebut del Calculador PID 2111: nombre 2 es primer? S
Controlador: rebut del Calculador PID 2112: nombre 6 es primer? N
Controlador: rebut del Calculador PID 2113: nombre 4 es primer? N
Controlador: rebut del Calculador PID 2112: nombre 5 es primer? S
Controlador: rebut del Calculador PID 2113: nombre 7 es primer? S
Controlador: nombrePrimersPipe=4 nombrePrimersExit=4
# Sempre s'ha de complir que nombrePrimersPipe=nombrePrimersExit
$
```

Requeriments d'ús de descriptors de fitxer dels *pipes*: tal com podeu veure a l'esquema de l'arquitectura, caldrà usar els següents descriptors de fitxer en els dos *pipes*:

- Descriptor escriptura *pipe* de nombres: 10.
- Descriptor lectura *pipe* de nombres: 11.
- Descriptor escriptura *pipe* de respostes: 20.
- Descriptor lectura *pipe* de respostes: 21.

MOLT IMPORTANT: Heu de fer la pràctica en tres executables (pels tres programes .c especificats a sobre), i emprant recobriment (funcions *execl*).

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.**

Concretament:

1. Funcionament (*pipes*, senyals, gestió d'errors, disseny del codi, etc.): **70%**
2. Entrega (codi comentat, memòria, preguntes professor, etc.): **30%**

6 Annexos

Annex A: Crides al Sistema involucrades

A la pràctica podeu utilitzar les següents crides a Sistema:

- `int open(const char *pathname, int flags);`
- `int close(int fd);`
- `off_t lseek(int fildes, off_t offset, int whence);`
- `ssize_t read(int fd, void *buf, size_t nbytes);`

- `ssize_t write(int fd, const void *buf, size_t num);`
- `sighandler_t signal(int signum, sighandler_t handler);`
- `int kill(pid_t pid, int sig);`
- `int pause(void);`
- `unsigned int alarm(unsigned int seconds);`
- `pid_t fork(void);`
- `int execlp( const char *file, const char *arg, ...);`
- `pid_t wait(int *status);`
- `void exit(int);`

Annex B: Estructures de dades

Les estructures de dades permeten crear nous tipus de variables complexes a partir dels tipus bàsics i nous tipus fruit de crear noves estructures:

- Una estructura és un conjunt d'elements heterogenis (anomenats camps) unificats sota un mateix nom, tipus i espai de memòria.
- Els camps d'una estructura seran dels tipus bàsics de C i/o dels nous tipus d'estructures creades.
- La sintaxi de declaració d'una estructura és:

```
struct nom_estructura {
      tipus_1 camp_1;
      tipus_2 camp_2;
      ...
      tipus_n camp_n;
} ;
```

- La creació d'una nova variable del nou tipus:
  `struct nom_estructura nom_variable;`
- Tot i això, el més habitual és crear a la vegada la nova estructura i un àlies a aquesta afegint la comanda *typedef* amb el nou àlies:

```
typedef struct [nom_opcional]{
      tipus_1 camp_1;
      tipus_2 camp_2;
      ...
      tipus_n camp_n;
} nom_nou_tipus_de_dada;
```

- La creació d'una nova variable del nou tipus:
  `nom_nou_tipus_de_dada nom_variable;`
- Així, en el cas d'aquesta pràctica, es pot crear la següent estructura:

```
typedef struct
{
 int  pid;
 int  nombre;
```

```
    char primer;
} t_infoNombre;
```

- I la creació d'una variable:
  ```
  t_infoNombre     infoNombre;
  ```
- Per accedir a cadascun dels camps individualment, s'usa el nom de la variable, seguit d'un punt i el nom del camp: `infoNombre.nombre = 35`.
- En el cas de punters:
  ```
  t_infoNombre     *pinfoNombre;
  infoNombre->nombre = 35.
  ```

Annex C: Algunes funcions que us poden ajudar

Altres funcions que us poden ser d'utilitat:

Mida informació a llegir/escriure:

- sizeof().

Annex D: Fòrum per a dubtes al Campus Virtual

A l'espai de l'assignatura al Campus Virtual, a l'apartat de debats, hi ha disponible el fòrum **PRA1. Processos, *pipes* i senyals: Primers** on podeu participar exposant els vostres dubtes i suggerint respostes.