PAC2

Presentació

Aquesta PAC planteja un seguit d'activitats amb l'objectiu que l'estudiant es familiaritzi amb la temàtica dels darrers mòduls de l'assignatura.

Competències

Transversals

- Capacitat per a la comunicació escrita en l'àmbit acadèmic i professional

Específiques

- Capacitat per a analitzar un problema en el nivell d'abstracció adequat a cada situació i aplicar les habilitats i coneixements adquirits per a abordar-lo i resoldre'l.

Enunciat

1. Mòduls 5, 6 i 7 **[2.25 punts, 0.75 punts per apartat]** Respondre justificadament a les següents preguntes:

   a) Indiqueu si les següents operacions poden ser suportades o no en les tres estructures d'espai de noms (lineal, arbre i graf dirigit):

   - Crear un arxiu.
   - Crear un directori.
   - Crear un enllaç físic (hard link).
   - Muntar un sistema d'arxius extern.

   La creació dels arxius es pot realitzar en qualsevol de les 3 estructures de noms.

   La creació de directoris, no es pot realitzar en una estructura lineal, ja que només suporta una única dimensió. Les altres dues estructures suporten la creació d'una jerarquia de directoris.

Només la estructura basada en grafs dirigits suporta la creació d'enllaços físics ja que necessiten que un nom pugui apuntar a qualsevol altre objecte existent en l'estructura de sistema d'arxius.

Per poder muntar un sistema d'arxius extern es necessita un directori en el sistema de fitxers arrel que serveixi d'ancoratge, per això l'estructura línia no suporta aquesta operació.

b) Donat el següent codi, explicar justificadament la jerarquia de processos que es crea, quants processos zombi es generen (encara que siguin adoptats pel procés init) i totes les vegades que la crida a sistema wait retorna un error.

```
1 id = fork();
2 if (id ==0)
3 fork();
4 fork();
5 wait(st);
6 exit(0);
```

Es crea una jerarquia de procés formada per un procés pare que té 2 fills (creats en la línia 1 i 4). El primer fill crea també dos processos fills (creats en les línies 3 i 4) i el segon fill no crea descendència. Dels processos néts, el primer nét crea un besnét a la línia 4, mentre que el segon nét no crea cap.

Cada procés executa la crida a sistema wait una única vegada. En el cas que el procés que la invoqui no tingui fills (segon fill, segon nét i el besnét) el wait fallarà. En la resta dels casos funcionarà.

Com que només es realitza un wait, en els casos que un procés creu més d'un fill, un d'ells es convertirà en procés zombi. Això passa en el pare i en el primer fill, de manera que es crearan 2 processos zombi.

c) Tenim dos threads que s'executen de forma concurrent el següent codi que modifica la variable compartida x. Indicar si el resultat d'aquesta execució pot ser indeterminista (que obtingui resultats diferents). En cas afirmatiu indicar que cal i com es pot solucionar. Assumiu que el valor inicial de x és 0, que el resultat final esperat és de 5 i que els semàfors Sema i SemB estan inicialitzats a 1.

Thread1:	Thread2:
`sem_wait(SemA);`	`sem_wait(SemB);`
`x=x + 3;`	`x =x + 2;`
`sem_signal(SemA);`	`sem_signal(SemB);`

L'execució d'aquest programa concurrent és indeterminista degut a que els dos fils utilitzen un semàfor diferents per accedir a la seva secció crítica, pel que són seccions crítiques diferents. Això provoca que no es garanteixi l'exclusió mútua a l'hora de definir la variable compartida x, generant condicions de carrera.

Per solucionar el problema, els dos fils haurien d'utilitzar el mateix semàfor per accedir a la secció crítica on es modifica la variable x. D'aquesta manera, s'impedeix que els dos fils modifiquin la variable al mateix temps i s'evita les condicions de carrera.

2. Mòduls 5 i 6 **[3.75 punts, 1.75 (0.5+0.5+0.75) + 2.0 (0.75+1.25)]**

a) A partir de la informació dels arxius/directoris mostrada en la següent imatge i coneixent que ex3.txt és un enllaç físic (hard link) a ex1.txt, respondre a les següents preguntes:

```
uoc$ ls -la
total 20
drwxr-xr-x  3 nando nando 4096 Dec  4 13:40 .
drwxrwxrwt 14 root  root  4096 Dec  4 16:17 ..
-rwsr-sr-x  1 nando nando    0 Dec  4 13:39 do.sh
-rw-r--r--  2 nando nando    5 Dec  4 13:39 ex1.txt
lrwxrwxrwx  1 nando nando    7 Dec  4 13:39 ex2.txt -> ex1.txt
-rw-r--r--  2 nando nando    5 Dec  4 13:39 ex3.txt
d-w-------  2 nando nando 4096 Dec  4 13:40 noa
uoc$
```

i. Explicar el significat de cada un dels camps d'informació (9 columnes) que es mostra en el llistat. Explicar també el significat dels diferents caràcters que apareixen en el primer camp de la informació mostrada.

El significat dels diferents camps / columnes és el següent:

- **El primer camp conté els permisos d'accés al fitxer juntament amb el tipus de fitxer.**

- **El segon camp conté el nombre d'enllaços físics (hard links) que conté el fitxer.**
- **El tercer camp conté el nom del propietari de l'arxiu.**
- **El quart camp conté el grup d'usuari a el qual pertany el fitxer.**
- **El cinquè camp conté la mida de el fitxer en bytes.**
- **El sisè camp (següents tres columnes) conté la data de creació / modificació del fitxer.**
- **L'últim camp conté el nom del fitxer / directori.**

En el primer camp, el primer caràcter identifica el tipus de fitxer (- fitxer regular, d directori, l enllaç físic, p pipe, etc.). Les següents agrupacions de 3, identifiquen els permisos (r lectura, w escriptura, x execució / accés) per als 3 dominis de protecció (propietari, grup i resta d'usuaris).

El bit s, vol dir que té activat el sticky bit i quan s'executi aquest programa, agafarà els permisos de propietari de l'arxiu independentment de l'usuari que l'executi.

ii. Justificar per que el segon camp té un 3 en la primera entrada un 14 en la segona. A que es deu aquest nombre?

Aquest camp identifica el nombre d'enllaços físics que apunten al directori actual i al directori pare. Està relacionat amb el nombre de directoris existents en el directori pare i en l'actual.

En el mateix sentit, a què es deu el fet que el fitxer ex1.txt i ex3.txt tingui un 2 en el segon camp i el fitxer ex2.txt només un 1.

Es deu al fet que el ext3.txt és un enllaç físic al fitxer ex1.txt i per tant tots dos té dos enllaços físics (dos noms). Mentre que ext2.txt és un enllaç simbòlic.

Seria possible que un fitxer pogués tenir el valor 14 en el segon camp? Com?

Si, si tingués 13 enllaços físics que apuntessin a aquest fitxer.

iii. Indicar les ordres que caldria executar perquè a partir d'un directori buit es pugui crear els fitxers/directoris mostren a la

imatge, amb les mateixes característiques i atributs (la mida i la data pot variar). Les característiques dels directori. i .. no cal replicar.

$ touch ex1.txt
$ ln -s ex1.txt ex2.txt
$ ln ex1.txt ex3.txt
$ touch do.sh
$ chmod +x do.sh
$ chmod +s do.sh
$ mkdir doa
$ chmod -rx noa

b) Imagineu que s'invoca des del shell la següent línia de comandes:

```
$ Ls -la -R / 2> file1.txt | grep "^ l"> file2.txt; wc -l
  file2.txt
```

i. Quines crides al sistema hauria invocar l'intèrpret de comandes quan executi l'ordre descrit?

Les crides que serien necessàries són: fork() per a la creació del procés que ha d'executar la comanda cat; un altre fork() per executar el grep i un últim fork per l'execució de la comanda wc.

Es necessita la crida pipe() per crear el mecanisme de comunicació entre la comanda ls i el grep i les crides close() i dup() per a poder realitzar la redirecció dels canals de la pipe a la stdin i stdout dels dos primers processos.

També es necessita la crida open() per realitzar la redirecció de la stderr del primer procés a l'arxiu file1.txt i la stdout del segon procés a l'arxiu file2.txt.

Finalment, es necessita utilitzar la crida a sistema wait() esperar la finalització dels dos primers processos abans de crear i executar el tercer i per esperar que aquest finalitzi.

ii. Escriure codi C que processi l'execució de la línia d'ordres anterior.

```
intmain()
{
```

```
int p[2];
if ( pipe(p) <0)
    perror("Error pipe creation.");
if ((fork())==0)
{
    /* 1st child -> ls  */
    close(1);
    dup(p[1]);
    close(p[1]);
    close(p[0]);
    close(2);
    if (open("file1.txt",O_CREAT|O_TRUNC|O_WRONLY,
             0600)<0)
        perror("Error opening file.");
    execlp("ls","ls","-la","-R","/",NULL);
    exit(1); /* error execlp */
}
    if ((fork())==0)
{
    /* 2nd child -> grep */
    close(0);
    dup(p[0]);
    close(p[1]);
    close(p[0]);
    close(1);
    if (open("file2.txt",O_CREAT|O_TRUNC|O_WRONLY,
             0600)<0)
        perror("Error opening file");
    execlp("grep","grep","^l",NULL);
    exit(1); /* error execlp */
}

/* Main process */
close(p[0]);
close(p[1]);

wait(NULL);
wait(NULL);

if ((fork())==0)
```

```
    {
        /* 3th child -> wc -l*/
        execlp("wc","wc","-l","file2.txt", NULL);
        exit(1);/* error execlp */
    }

    wait(NULL);

    exit(0);
}
```

3. Mòdul 7**[4 punts, 1 per apartat]**

El Gran Hotel ofereix als seus clients un servei d'aparcacotxes. Els clients arriben a l'Hotel; allà els espera un grum, que els recull el cotxe i se l'emporta al garatge de l'hotel. L'Hotel disposa de diferents grums, indistingibles entre ells.

Per aparcar un cotxe, el grum es dirigeix a l'ascensor de baixada. Aquest ascensor només pot ser usat per un cotxe cada cop. Un cop aparcat el cotxe, torna al seu lloc davant de l'Hotel usant les escales.

Quan un client vol marxar, demana a un grum que li porti el seu cotxe. L'aparcador se'n va al garatge usant les escales, i recupera el cotxe amb l'ascensor de pujada, que també només pot usar un sol cotxe cada cop. Considerem que el client mai marxarà abans de que el seu cotxe hagi estat aparcat.

No és important controlar que un client rebi el seu cotxe, ja que el que es vol estudiar és el comportament de les cues.

Utilitzant les següents operacions de semàfors:

- `sem_init(semaphore s, int v)`. Inicialitza el semàfor *s* amb *v* instàncies inicials (valor inicial).
- `sem_wait(semaphore s)`. Demana una instància del semàfor s. Espera que el valor del semàfor sigui més gran que 0 i quan ho és el decrementa de forma atòmica.
- `sem_signal(semaphore s)`. S'incrementa de forma atòmica el valor del semàfor.

Es demana:

a) Suposant que inicialment no ens importen les places disponibles a l'aparcament (assumim infinites places) ni el número de grums (assumim infinits). Quins serien els processos per al client, tant per aparcar com per recuperar un cotxe?

Declaració variables i semàfors

```
Semaphore SemClientArriba, SemClientMarxa, SemBaixa,
          SemPuja, SemEspCotxe;
```

Inicialització

```
sem_init(SemClientArriba,0);
sem_init(SemClientMarxa,0);
sem_init(SemBaixa,1);
sem_init(SemPuxa,1);
sem_init(SemEspCotxe,0);
```

Aparcar

```
{
    sem_wait(SemClientArriba);
    Agafar_cotxe;
    sem_wait(SemBaixa);
    Surt_ascensor_B;
    sem_signal(SemBaixa);
    Pujar_a_Recepció;
}
```

Recuperar

```
{
    sem_wait(SemClientMarxa);
    Baixar_a_Parking;
    sem_wait(SemPuja);
    Surt_ascensor_P;
    sem_signal(SemPuja);
    sem_signal(SemEspCotxe);
}
```

Client

```
{
        sem_signal(SemClientArriba);
        Donar_cotxe;
        Anar_a_habitació;
        Tornar_d'habitació;
        sem_signal(SemClientMarxa);
        sem_wait(SemEspCotxe);
        Agafar_cotxe;
}
```

b) Suposem ara que el nombre de places de aparcament és limitat (tenim P places), però seguim tenint grums il·limitats. Si quan un grum vol entrar a l'aparcament no hi ha places lliures s'espera fins que hi hagin (sense deixar el cotxe). Suposem que el temps que un client triga en tornar a demanar el cotxe és sempre més gran que el que ha necessitat el grum per aparcar. Com queden ara el procediments aparcar i recuperar?

Declaració variables i semàfors

```
SemaphoreSemClientArriba, SemClientMarxa, SemBaixa,
          SemPuja, SemEspCotxe, SemPlacesLliures;
```

Inicialització

```
sem_init(SemClientArriba,0);
sem_init(SemClientMarxa,0);
sem_init(SemBaixa,1);
sem_init(SemPuxa,1);
sem_init(SemEspCotxe,0);
sem_init(SemPlacesLliures,P);
```

Aparcar	**Recuperar**

{
sem_wait(SemClientArriba);
Agafar_cotxe;
sem_wait(SemPlacesLliures);
sem_wait(SemBaixa);
Surt_ascensor_B;
sem_signal(SemBaixa);
Pujar_a_Recepció;
}
	{
	sem_wait(SemClientMarxa)
	Baixar_a_Parking;
	sem_wait(SemPuja);
	Surt_ascensor_P;
	sem_signal(SemPuja);
	sem_signal(SemPlacesLliures);
	sem_signal(SemEspCotxe);
	}

c) Ara suposar que en comptes d'un ascensor tenim una rampa que permet accedir al pàrquing, però només té un carril. Quan estan baixant cotxes al pàrquing, no poden pujar i viceversa. Poden haver-hi múltiples cotxes circulant en un mateix sentit per la rampa al mateix temps.Com queden ara el procediments aparcar i recuperar?

Declaració variables i semàfors

```
SemaphoreSemClientArriba, SemClientMarxa, SemEspCotxe,
          SemPlacesLliures, SemMutexA, SemMutexB,
          SemRampa;

int CotxesBaixant=0,CotxesPujant=0;
```

Inicialització

```
sem_init(SemClientArriba,0);
sem_init(SemClientMarxa,0);
sem_init(SemEspCotxe,0);
sem_init(SemPlacesLliures,P);
sem_init(SemMutexA,1);
sem_init(SemMutexB,1);
sem_init(SemRampa,1);
```

Aparcar	**Recuperar**
{	{

sem_wait(SemClientArriba);
Agafar_cotxe;
sem_wait(SemPlacesLliures);
sem_wati(SemMutexA);
CotxesBaixant++;
If (CotxesBaixant==1)
sem_wait(SemRampa);
sem_signal(SemMutexa);
Baixar_Rampa_Parking;
sem_wait(SemMutexA);
CotxesBaixant--;
If (CotxesBaixant==0)
sem_signal(SemRampa);
sem_signal(SemMutexA);
Pujar_a_Recepció;
}
	sem_wait(SemClientMarxa)
	Baixar_a_Parking;
	sem_wati(SemMutexB);
	CotxesPujant++;
	If (CotxesPujant==1)
	sem_wait(SemRampa);
	sem_signal(SemMutexB);
	Pujar_Rampa_Parking;
	sem_wait(SemMutexB);
	CotxesPujant--;
	If (CotxesPujant==0)
	sem_signal(SemRampa);
	sem_signal(SemMutexB);
	sem_signal(SemPlacesLliures);
	/* Donar cotxer client */
	sem_signal(SemEspCotxe);
	}

d) Finalment, suposem la versió de l'ascensor (apartat b) i que ara el nombre de grums també és limitat (tenim G grums). Si un client arriba i no hi ha grums disponibles s'espera sempre que hi hagin menys de N cotxes davant, sinó marxa. Com queden ara els procediments que simulen els clients, l'aparcament i la recuperació?

Declaració variables i semàfors

```
int NumEspera=0;
SemaphoreSemClientArriba, SemClientMarxa, SemBaixa,
          SemPuja, SemEspCotxe, SemPlacesLliures,
          SemMutex, SemGrums;
```

Inicialització

```
sem_init(SemClientArriba,0);
sem_init(SemClientMarxa,0);
sem_init(SemBaixa,1);
```

```
sem_init(SemPuxa,1);
sem_init(SemEspCotxe,0);
sem_init(SemPlacesLliures,P);
sem_init(SemMutex,1);
sem_init(SemGrums,G);
```

Aparcar

```
{
    sem_wait(SemClientArriba );
    Agafar_cotxe;
    sem_wait(SemPlacesLliures);
    sem_wait(SemBaixa );
    Surt_ascensor_B;
    sem_signal(SemBaixa );
    Pujar_a_Recepció;
    sem_signal(Grums)
}
```

Recuperar

```
{
    sem_wait(SemClientMarxa )
    Baixar_a_Parking;
    sem_wait(SemPuja );
    Surt_ascensor_P;
    sem_signal(SemPuja );
    sem_signal(SemPlacesLliures
    );
    tornar_cotxe;
    sem_signal(Grums);
    sem_signal(SemEspCotxe)
}
```

Client

```
{
    sem_wait(SemMutex);
    if (NumEspera==N){
        sem_signal(SemMutex);
        return;
    }
    NumEspera++;
    sem_signal(SemMutex);
    sem_wait(Grums);
    sem_signal(SemClientArriba );
    Donar_cotxe;
    sem_wait(SemMutex);
```

```
    NumEspera--;
    sem_signal(SemMutex);
    Anar_a_habitació;
    Tornar_d'_habitació;
    sem_signal(SemClientMarxa );
    sem_wait(SemEspCotxe );
    Agafar_cotxe;
}
```

Recursos

Bàsics:

- Mòduls5, 6,i 7 de l'assignatura.
- Document "Introducció a la programació de UNIX" (disponible a l'aula) o qualsevol altre manual similar.
- L'aula "Laboratori de Sistemes Operatius" (podeu plantejar els vostres dubtes relatius a l'entorn Unix, programació,...).

Complementaris:

- La bibliografia de l'assignatura.

Criteris de valoració

Es valorarà la justificació de les respostes presentades. S'agrairan les respostes breus i concises. El pes de cada pregunta està indicat a l'enunciat.

En la correcció es tindran en compte els següents aspectes:

- Les respostes hauran d'estar articulades a partir dels conceptes estudiats en teoria i en la guies de l'assignatura.
- S'agrairà la claredat i la capacitat de síntesis en les respostes.
- Es valorarà essencialment la correcta justificació de les respostes, recolzada pels fonaments teòrics.

Format i data de lliurament

La solució es lliurarà en un fitxer text (format .txt o .pdf)

El nom del fitxer tindrà el format següent: "Cognom1Cognom2PAC2.txt". Els cognoms s'escriuran sense accents. Per exemple, l'estudiant Marta Vallès i Marfany utilitzarà el nom de **fitxer següent: VallesMarfanyPAC2.txt**

La data límit per al lliurament de la pac és el **dijous26 de desembre 2019**.