

## Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	20/06/2018	09:00



Enganxeu en aquest espai una etiqueta  
identificativa  
amb el vostre codi personal  
Examen

### Fitxa tècnica de l'examen

- Comprova que el codi i el nom de l'assignatura corresponen a l'assignatura matriculada.
- Només has d'enganxar una etiqueta d'estudiant a l'espai corresponent d'aquest full.
- No es poden adjuntar fulls addicionals, ni realitzar l'examen en llapis o retolador gruixut.
- Temps total: **2 hores** Valor de cada pregunta: **2.5 punts**
- En cas que els estudiants puguin consultar algun material durant l'examen, quins són?  
**Un full mida foli/A4 amb anotacions per les dues cares.** En cas de poder fer servir calculadora, de quin tipus? **NO PROGRAMABLE**
- Si hi ha preguntes tipus test: Descompten les respostes errònies? **NO** Quant?
- Indicacions específiques per a la realització d'aquest examen:

## Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	20/06/2018	09:00

---

### Enunciats

---

# Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	20/06/2018	09:00

## 1. Preguntes breus

a) En quines circumstàncies un procés passa d'estat Blocked(Wait) a estat Ready? I de Run a Blocked(Wait)? Si és possible, indiqueu una crida al sistema que provoqui en canvi a cada cas.

Típicament, un procés abandona l'estat de Wait i passa a Ready quan es produeix l'esdeveniment que el procés estava esperant (per exemple, la finalització d'una operació de lectura, l'expiració d'un temporitzador,...).

[No és necessari] També es pot provocar aquesta transició amb algun signal o si el SO detecta algun error mentre s'espera l'esdeveniment.

La transició de Run a Wait es produirà quan el procés invoqui una crida al sistema bloquejant com ara un a lectura (read) sobre el teclat, sigsuspend,...

b) Sigui un sistema de gestió de memòria basat en paginació sota demanda on la mida de pàgina és de 4KB. És possible que la traducció de l'adreça lògica 0x1001 no provoqui una excepció per fallada de pàgina però que la traducció de l'adreça lògica 0x1002 sí la generi?

Podeu assumir que les adreces lògiques involucrades a l'exercici corresponen a pàgines vàlides i que el contingut de la taula de pàgines del procés és el mateix a les dues traduccions; les adreces estan expressades en hexadecimal.

Totes dues direccions lògiques corresponen a la mateixa pàgina lògica (la 0x1, el resultat de descartar els 12 bits baixos perquè les pàgines són de  $2^{12}$  bytes). Per tant, al traduir les dues direccions lògiques estarem accedint a la mateixa entrada de la taula de pàgines. Per tant, com ens diuen que el contingut de la taula de pàgines no varia, o les dues traduccions no generen fallada de pàgina o les dues traduccions el generen.

c) Els dispositius d'entrada/sortida poden classificar-se com a "físics", "lògics" i "virtuals". En què es diferencien un dispositiu "físic" i un "virtual"? Indiqueu quins tipus de dispositius apareixen al següent fragment de codi:

```
...
int fd, p[2];
char c;

fd = open("file.txt", O_RDONLY);
pipe(p);

while (read(fd, &c, 1) > 0)
  write(p[1], &c, 1);
...
```

Un dispositiu físic correspon al hardware nu, sense cap suport del SO.

Un dispositiu virtual és la visió que tenen els processos dels dispositius d'entrada/sortida; és una visió oferta pel SO i independent del dispositiu físic al que s'està accedint.

Al codi apareixen dos dispositius lògics ("file.txt" i una pipe), i tres dispositius virtuals (fd, p[0], p[1]).

d) Què és un deadlock? En què consisteix la condició necessària per a l'existència de deadlock "No-expropiació"? Com es podria garantir que mai es satisfarà?

## Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	20/06/2018	09:00

Consisteix en que els recursos que han estat assignats a un thread només poden ser lliberats pel propi thread. La forma d'evitar aquesta condició seria permetre que els threads poguessin lliberar recursos que actualment estàn assignats a altres threads.

## Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	20/06/2018	09:00

### 2.- Memòria

Tenim un sistema de gestió de memòria basat en paginació que té les següents característiques:

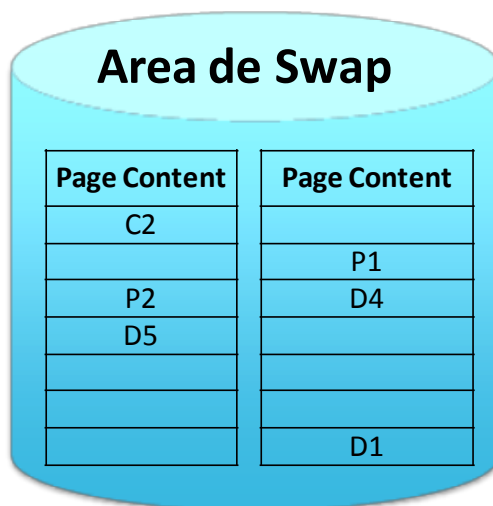
- Mida adreça lògica: 16 bits.
- Mida adreça física: 12 bits.
- Mida pàgina / frame: 16 Bytes

#### Espai Lògic (procés A)

Pàg. Lògica	Content
0x0	C1
0x1	C2
0x2	C3
0x3	C4
0x4	C5
0x5	
0x6	
...	
0x0F	
0x1A	D1
0x1B	D2
0x1C	D3
0x1D	D4
0x1E	D5
0x1F	
...	
0xFFFC	P4
0xFFFD	P3
0xFFFE	P2
0xFFFF	P1

#### Espai físic

Frame	Content	Frame	Content
0	C1	8	
1		9	P3
2	D2	0A	D3
3	P4	0B	
4		0C	C5
5	C3	0D	
6	C4	0E	
7		0F	



Assumiu que les pàgines sense contingut són invàlides.

Contestar, justificant les respostes, a les següents preguntes:

## Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	20/06/2018	09:00

a) Assumint un sistema de paginació sota demanda, cal indicar que efecte produeix un accés a una entrada a la taula de pàgines per a cadascuna de les següents combinacions dels bits V i P. Si alguna combinació no és possible indicar-ho.

- V = 0, P = 0

Pàgina no vàlida i no present. Un accés a aquesta pàgina generarà una excepció de direcció invàlida, ja que la pàgina no pertany a l'espai de memòria lògica del procés.

- V = 0, P = 1

Pàgina no vàlida i present. Combinació no vàlida, ja que no pot ser que una pàgina que no pertanyi a la memòria lògica del procés, es trobi present.

- V = 1, P = 0

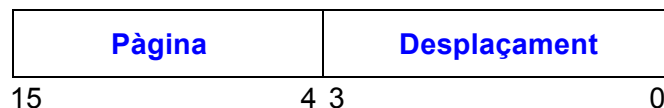
Pàgina vàlida i no present. Un accés a aquesta pàgina generarà una fallada de pàgina, ja que la pàgina és vàlida però no es troba en memòria física.

- V = 1, P = 1

Pàgina vàlida i present. S'accedirà a la posició de memòria, sense cap problema.

b) Calcular la mida de cada un dels camps de l'adreça lògica i de l'adreça física.

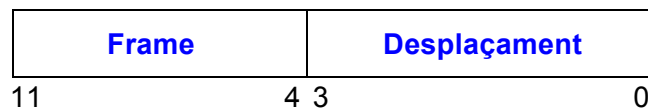
L'adreça lògica té 20 bits que es descomponen de la següent manera:



Desplaçament pàgina =  $\log_2(\text{Mida\_pag}) = \log_2(16) = 4$  bits

Pàgina =  $\log_2(\text{Nombre\_pags}) = \log_2(2^{16}/16) = 12$  bits

Per accedir a tota la memòria física tenim 12 bits:



Desplaçament frame =  $\log_2(\text{Mida\_frame}) = \log_2(16) = 4$  bits

Frame =  $\log_2(\text{Nombre\_frames}) = \log_2(2^{12}/16) = 8$  bits

c) A partir de l'assignació de memòria física i el mapa de memòria lògica del procés omplir els camps de la seva taula de pàgines:

## Taula de Pàgines

## Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	20/06/2018	09:00

	V	P	Frame
0x0	1	1	0x0
0x1	1	0	
0x2	1	1	0x5
0x3	1	1	0x6
0x4	1	1	0xC
0x5	0	0	
0x6	0	0	
...	0	0	
0x0F	0	0	
0x1A	1	0	
0x1B	1	1	0x2
0x1C	1	1	0xA
0x1D	1	0	
0x1E	1	0	
0x1F	0	0	
...	0	0	
0xFFFC	1	1	0x3
0xFFFD	1	1	0x9
0xFFFE	1	0	
0xFFFF	1	0	

d) Quina adreça física es correspon amb l'adreça lògica 01C6H del procés A?

- **Adr. Lògica 001C6H -> Adr. física: 0A6h**

**Pàgina: 01Ch Despl: 6h**

**Frame: TaulaPàgines[Pàgina] = TaulaPàgines[01Ch] = 0Ah**

**Adr. Física: 0Ah & 06h = 0A6h**

```

for (a=8;a<argc;a++) {
    ids[a-8] = concurrent_evaluate_tickets(argv[a], prizes_file,
    argv[1], argv[2], argv[3], argv[4], argv[5], argv[6], argv[7]);
}

```

```

for (a=8;a<argc;a++)

```

## Examen 2017/18-2

```

    tpid, status;

```

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	20/06/2018	09:00

```

    error("[concur2ok::main] Error waitingchildprocess.");
    if (!WIFEXITED(status))
        error("[concur2ok::main] Error cal_prizeschildprocess anormal
        termination.");
    if (!WEXITSTATUS(status))

```

La segona pràctica de l'assignatura consistia en implementar una aplicació concurrent per calcular els bitllets premiats d'una la loteria 6-49. L'aplicació rebrà els bitllets que cal avaluar en diferents fitxers (provinents de diferents administracions de loteria). L'usuari introduirà com a paràmetres els 6 nombres premiats, més el complementari. L'aplicació avaluarà cadascun dels bitllets i calcularà si han estat premiats i la categoria del premi. Com a resultat de l'execució, l'aplicació crea un fitxer amb els bitllets premiats. També mostrarà per pantalla el total de bitllets de loteria avaluats i el nombre de bitllets premiats.

```

    use
    winner_tickets += WEXITSTATUS(status);
}

```

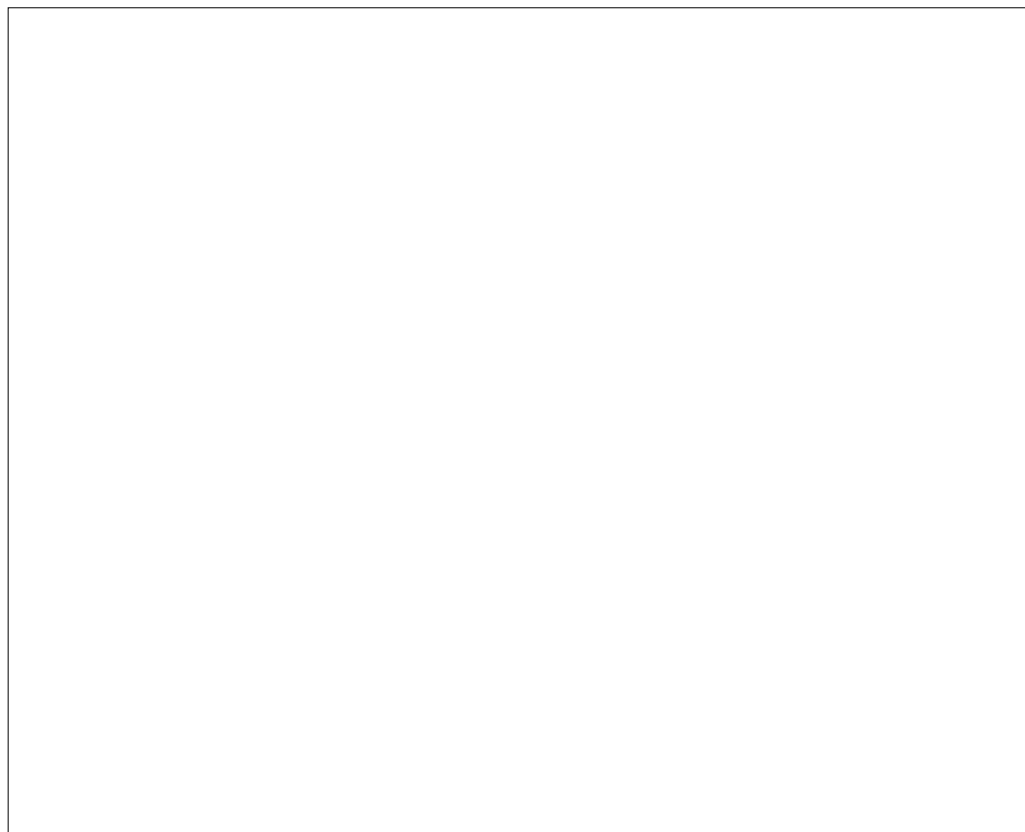
a) L'enunciat del pas2, "*Processament concurrent dels bitllets de loteria.*", deia:

Implementar l'aplicació (programa concur2.c) de manera que es puguin avaluar els bitllets de loteria de forma concurrent. Per aconseguir això, cada fitxer de bitllets d'entrada serà processat per un procés diferent.

El programa principal (concur2.c) haurà de crear un procés fill que invoqui al programa calc\_prizes per cadascun dels fitxers d'entrada. Cada procés tindrà redirigida el fitxer a processar a la seva stdin.

Així mateix, el programa principal controlarà el codi d'acabament dels processos fills i en cas d'error ho notificarà a l'usuari. El codi de finalització dels processos fills també s'utilitzarà per calcular el nombre total de butlletes de loteria premiats. Per això cada procés fill tornarà un 0 si s'ha produït un error o el nombre de butlletes premiats més 1 si el procés fill s'ha executat correctament.

S'adjunta un fragment de la solució proposada per a aquest pas:





## Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	20/06/2018	09:00

Es demana:

- Identificar quina informació rep el procés pare en la crida al sistema wait (tant el paràmetre, com en el valor de retorn) i per a què serveixen les macros WIFEXITED i WEXITSTATUS.

En el paràmetre status es rep el codi de finalització del procés fill que ha acabat, en aquest cas es rep el nombre de butlletes premiades + 1. Com a valor de retorn de la crida al sistema wait es rep el pid del procés finalitzat o el codi de error en el cas que la crida doni un error.

La macro WIFEXITED retorna cert si el procés fill ha finalitzat correctament mitjançant un exit.

La macro WEXITSTATUS retorna el codi de finalització del procés fill.

- En la solució proposada pot passar que el nombre total de bitllets premiats calculat pel procés principal no sigui el correcte a causa que el codi d'acabament té una mida limitada (8 bits) i es pot desbordar si el nombre retornat per fills supera els 254 bitllets premiats.

Proposeu una solució a aquest problema. No cal que implementeu el codi, només cal explicar que mecanismes es utilitzarien i com es portaria a terme.

En aquest cas, es pot utilitzar la comunicació mitjançant pipes per passar el nombre de butlletes premiades del procés fill (calc\_prizes) al procés pare (concurs). Per a això, el procés pare ha de crear una pipe de comunicació amb cada un dels processos calc\_prizes i redirigir el descriptor d'escriptura de la pipe a la stdout del procés fill. El procés fill, escriuria el nombre de butlletes premiades a la stdout i la resta de missatges per la stderr.

- L'enunciat del pas4, "Processament concurrent i balancejat", deia:

L'objectiu d'aquest pas, és assolir una execució més balancejada dels processos calc\_prizes. Per això, ajuntarem totes les butlletes premiades en un mateix fitxer i aquest fitxer serà compartit/llegit per tots els processos calc\_prizes. De la mateixa manera, per crear un únic fitxer de sortida, tots els processos escriuran en el mateix fitxer de sortida.

Perquè l'accés compartit a tots dos fitxers funcioni, cal que les operacions de lectura d'un bitllet o l'escriptura d'un premi es realitzi de forma atòmica. No obstant això, el format actual del fitxer d'entrada dificulta això. Analitzar el format d'entrada i identificar quin és el problema que pot impedir les lectures atòmiques d'una butlleta. Un cop identificat, crear un programa auxiliar que permeti convertir el format del fitxer d'entrada de bitllets en el format adequat per ser compartit pels diferents processos calc\_prizes.

Es demana:

- ¿Indiqueu que impacte tindria que tots els processos calc\_prizes, fossin els responsables d'obrir el fitxer d'entrada?

Si tots els processos calc\_prizes obren el fitxer d'entrada, llavors cada un d'ells tindran el seu propi descriptor de lectura del fitxer i tots ells llegiran i processaran totes les butlletes. Amb la qual cosa, tots els processos replicaran el còmput dels altres i la versió concurrent no tindrà cap benefici.

## Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	20/06/2018	09:00

ii. ¿Com afectaria al resultat si els premis no es escrivissin de manera atòmica?

En compartir tots els processos `calc_prizes4ok` el mateix fitxer de sortida, si no s'escriuen els premis de forma atòmica, llavors es pot barrejar la informació de diferents premis, generant-se un resultat incorrecte.

## Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	20/06/2018	09:00

### 4.- Concurrencia

A una fàbrica es construeixen els productes P i Q a partir dels components A, B i C.

Per controlar les existències de cadascun dels components A, B i C, s'utilitzen tres semàfors de recursos (`semA`, `semB` i `semC` respectivament). Si ens fixem en el component A, `semA` estarà inicialitzat al nombre de components A disponibles inicialment; cada cop que algú necessiti un component A haurà d'executar `sem_wait(semA)`; cada cop que un nou component A estigui disponible caldrà executar `sem_signal(semA)`. Els components B i C seran gestionats de forma anàloga.

a) Supposeu que inicialment, només es produeix el producte P i que ens garanteixen que només hi ha un thread produint aquest producte.

El producte P necessita dos components A, un component B i un component C. El codi que executa el thread productor de P és:

```
void produce_P()
{
    sem_wait(semA);
    sem_wait(semB);
    sem_wait(semC);
    sem_wait(semA);

    /* produce P */
    ...
}
```

En aquest escenari, es podria arribar a un deadlock? En cas negatiu indiqueu quina(es) condició(ns) necessària(ies) per provocar deadlock no es satisfà(n). En cas positiu, indiqueu com es podria evitar el deadlock.

No es pot produir deadlock perquè únicament hi ha un thread involucrat. Per tant, no es compleix ni "Mutual exclusion" ni "Circular wait".

## Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	20/06/2018	09:00

b) Suposeu que el producte Q necessita dos components B, un component C i un component A. Completeu el codi de `produce_P()` i de `produce_Q()` de forma que esperin a disposar de tots els components necessaris per a produir P i Q.

A més, per restriccions de la fàbrica, no es poden produir dos o més productes simultàniament (P i P, P i Q, Q i Q).

Tingueu també present que ara es permet que hi hagi diversos threads executant `produce_P()` i `produce_Q()`.

Afegiu el codi necessari a `produce_P()` i `produce_Q()` i indiqueu les variables globals (i la seva inicialització) per obtenir el comportament especificat.

```
/* Variables globals i inicialitzacions */
sem_t mutex; /* init to 1 */
```

```
void produce_P()
{
    sem_wait(mutex);

    sem_wait(semA); sem_wait(semA);
    sem_wait(semB);
    sem_wait(semC);

    /* Produce P */
    ...

    sem_signal(mutex);
}
```

```
void produce_Q()
{
    sem_wait(mutex);

    sem_wait(semA);
    sem_wait(semB); sem_wait(semB);
    sem_wait(semC);

    /* Produce Q */
    ...

    sem_signal(mutex);
}
```