Espai grapa

Universitat Oberta de Catalunya

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	16/06/2018	15:30

05.566 16 06 18 EX

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	16/06/2018	15:30

Enunciats

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	16/06/2018	15:30

1.- Preguntes breus

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

La primera transició es produeix quan el planificador de la CPU decideix treure l'ús de la CPU al procés que actualment l'està utilitzant. El motiu més raonable és l'expiració del quantum assignat al procés; aquesta expiració provoca una interrupció que es atesa pel planificador i que desencadena que la CPU sigui assignada a algun dels processos a Ready i que el procés que estava a Run passi a Ready.
[Ampliació de la resposta (no era necessària a l'examen):
Aquest no és l´únic motiu. A un planificador basat en prioritats, aquesta transició podria produir-se si apareix un procés a Ready que sigui més prioritari que el que actualment es troba a Run.]

La segona, en principi, no és possible perquè normalment els processos passen de Wait a Ready.

b) Sigui un sistema de gestió de memòria basat en paginació sota demanda on la mida de pàgina és de 4KB. Les adreces lògiques no consecutives 0x1001 i 0x1003 seran sempre traduïdes a adreces físiques no consecutives?
Podeu assumir que les adreces lògiques involucrades a l'exercici corresponen a pàgines vàlides i presents a memòria física.

Aquestes adreces pertanyen a la mateixa pàgina lògica (la 0x1 perquè, com les pàgines són de 4KB=2^12 bytes, és el resultat de descartar els 12 bits baixos de les adreces lògiques).
La traducció de les direccions lògiques serà el resultat de concatenar l'identificador de la pàgina física on està mapejada la pàgina lògica 0x1 (aquest identificador s'obtindrà accedint a la taula de pàgines) i el desplaçament (0x001 i 0x003 respectivament).
Si es fa servir el mateix mapeig de la pàgina lògica 0x1, la traducció d'aquestes adreces lògiques produirà adreces físiques no consecutives perquè els desplaçaments són consecutius (0x001 i 0x003).

c) Els dispositius d'entrada/sortida poden classificar-se com a "físics", "lògics" i "virtuals". En què es diferencien un dispositiu "lògic" 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 virtual ha estat obert (típicament, mitjançant la crida open) i el procés disposa d'un identificador de canal (file descriptor) per accedir-hi mitjançant les crides read/write/...
Un dispositiu lògic correspon a les abstraccions que el SO implementa sobre els dispositius físics. Per exemple, els fitxers, les pipes, /dev/null, ...

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

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	16/06/2018	15:30

d) Què és un deadlock? En què consisteix la condició necessària per a l'existència de deadlock "Espera circular"? Com es podria garantir que mai es satisfarà?

"Espera circular" indica que existeix una cadena de dos o més threads on cada thread de la cadena té algún recurs concedit i està bloquejat esperant obtenir un altre recurs que actualment està concedit a un altre thread de la cadena.
La forma més senzilla de garantir que aquesta condició mai es satisfarà és establint una ordenació entre els recursos i forçant que els threads sol·licitin els recursos que necessiten seguint aquesta ordenació.

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	16/06/2018	15:30

2.- Memòria

Tenim un sistema que té les següents característiques:

- Mida adreça lògica: 24 bits.
- Mida adreça física: 20 bits.
- Mida pàgina / frame: 4 KB

Assumint un sistema de gestió de memòria basat en paginació sota demanda i la següent taula de pàgines per al procés A:

Taula de pàgines (procés A)

	V	P	Frame
00h	1	0	10h
01h	1	0	21h
02h	1	1	03h
03h	1	1	31h
04h	1	0	42h
05h	0	0	44h
06h	0	0	34h
07h	1	0	13h
08h	1	1	02h
09h	1	1	3Ah
0Ah	1	0	2Dh
0Bh	1	0	24h
0Ch	1	1	35h
0Dh	0	0	06h

	V	P	Frame
0Eh	0	0	15h
0Fh	0	0	88h
10h	0	0	66h
11h	1	1	0Eh
12h	1	0	3Eh
13h	1	0	0Bh
14h	1	1	1Fh
15h	0	0	25h
...			
FFBh	0	0	3Fh
FFCh	1	0	40h
FFDh	1	1	11h
FFEh	1	1	03h
FFFh	1	0	08h

Tingueu en compte que totes les pàgines en el rang 16h-FFAh són invàlides.

Contestar, justificant les respostes, a les següents preguntes:

a) Assumint un sistema de paginació sota demanda, indicar el significat en la taula de pàgines 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. Vol dir que aquesta pàgina no pertany a l'espai de memòria lògica del procés.

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	16/06/2018	15:30

- 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

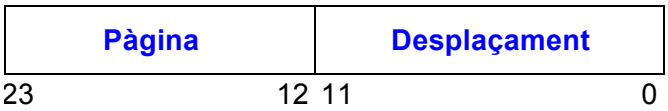
  Pàgina vàlida i no present. Vol dir que està pàgina forma part de la memòria lògica del processos, però que ara no es troba en memòria física, sinó en l'àrea de swap.

- V = 1, P = 1

  Pàgina vàlida i present. Vol dir que està pàgina forma part de la memòria lògica del procés i que està en memòria física.

b) Calcular la mida de cada un dels camps de la direcció lògica i de la direcció física.

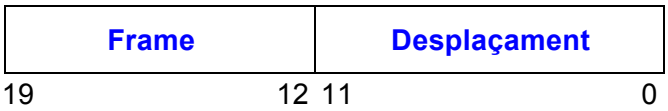
La direcció lògica té 20 bits que es descomponen de la següent manera:



Desplaçament pàgina = $\log_2$(Mida_pag) = $\log_2$(4096) = 12 bits

Pàgina = $\log_2$(Nombre_pags) = $\log_2$(2^24/4096) = 12 bits

Per accedir a tota la memòria física tenim 20 bits:



Desplaçament pàgina = $\log_2$(Mida_frame) = $\log_2$(4096) = 12 bits

Frame = $\log_2$(Nombre_frames) = $\log_2$(2^20/4096) = 8 bits

c) Quin seria el rang d'adreces lògiques vàlides del procés? Es correspon amb el rang d'adreces físiques? Per què?

000000h-003FFFh

007000h-00CFFFh

011000h-014FFFh

FFC000h-FFFFFFh

Aquest rang d'adreces lògiques no es correspon amb el rang d'adreces físiques per així permetre la reubicació dinàmica de la memòria del procés a qualsevol zona de memòria lliure que sigui prou gran. D'aquesta manera, no es produeix fragmentació externa.

d) Quina adreça física es correspon amb la direcció lògica 011100H del procés A?

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	16/06/2018	15:30

- **Adr. Lògica 011100H -> Adr. física: 0E100h**

  Pàgina: 11h Despl: 100h

  Frame: TaulaPàgines[Pàgina] = TaulaPàgines[11h] = 0Eh

  Adr. Física: 0Eh & 0100h = 0E100h

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	16/06/2018	15:30

3.- Gestió de processos

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.

a) L'enunciat del pas3, "*Versió concurrent amb fitxer de bitllets premiats ordenat*", deia:

L'objectiu d'aquest pas, és aconseguir que l'aplicació concurrent generi un fitxer amb els bitllets premiats ordenats en funció de la seva categoria. El programa, un cop processats els fitxers d'entrada concurrentment, haurà d'implementar les accions oportunes per executar les següents ordres:

```
> cat Winners.txt | sort -nr> Winners_sorted.txt
> cp Winners_sorted.txt Winners.txt
> rm Winners_sorted.txt
```

Per executar la primera comanda s'haurà de crear 2 processos nous perquè executin les comandes cat i sort respectivament. Perquè la sortida de la comanda/procés cat la processi la comanda/procés sort caldrà redirigir mitjançant pipes la sortida estàndard del procés cat a l'entrada estàndard procés sort.

Per executa la segona comanda s'haurà d'esperar que finalitzi la primera comanda i a continuació crear un nou procés perquè executi la comanda cp.

Per executa la tercera comanda s'haurà d'esperar que finalitzi la segona comanda i a continuació crear un nou procés perquè executi la comanda rm.

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

Es demana:

a) Contesteu les següents preguntes referents al codi:

i. Quin seria l'efecte sobre l'execució de l'aplicació si llevéssim tots els waits de la solució proposada.

Si traguéssim els waits en el codi anterior, llavors no es esperaria a que es completés el procés anterior. Per tant es podria copiar el fitxer abans que estigués ordenat o eliminar abans que tingui tots els resultats.

ii. En el primer procés fill, perquè serveix tancar els descriptors dep1? Que passaria si no es tanquessin?

El tancar els descriptors repetits, permet aconseguir que només hi hagi una còpia de cada un d'ells, el descriptor utilitzat per cada un dels processos de la comunicació. Això permet controlar la finalització de l'escriptura o la finalització de la lectura, que es produirà quan no hi hagi cap descriptor obert d'aquest extrem de la comunicació.

Si no es tanquen, llavors el procés sort no finalitzaria mai i el programa es quedaria bloquejat en el segon dels dos waits seguits, esperant indefinidament.

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	16/06/2018	15:30

iii. ¿Que impacte té en el programa si canvi el segon paràmetre del primer *execlp*, per un "rm"?

Cap impacte, ja que només estem canviant el nom del procés i això no afecta al seu comportament.

b) Ampliar la solució proposada per únicament es guardin els premis amb 5 números encertats. Podeu utilitzar la comanda grep per realitzar aquesta modificació. Prohibit utilitzar la funció system.

Recordeu que el format del fitxer de premis és el següent:

5+1 734439603 1 2 3 5 11 13*
5+0 199283327 2 7 11 13 9
4+0 1735933903 2 3 7 11
3+0 199283329 1 3 7
3+0 199283332 5 7 11
3+0 1572999123 1 2 5
3+0 356876030 1 5 7

```
void sort_prizes_file()
{
	int pid1, pid2, pid3, pid4, pid5, cod_cat, cod_sort, cod_cp, cod_rm, cod_grep;
	int prizes_file;
	int p1[2], p2[2];

	if (pipe(p1)<0) error("[concur2ok::sort_prizes_file] Error creating pipe.");

	if (pipe(p2)<0) error("[concur2ok::sort_prizes_file] Error creating pipe2.");

	// First Child --> cat Winners.txt
	if ((pid1=fork())==0)
	{
		// redirect stdout -> pipe in (p1[1])
		close(1);
		if (dup(p1[1])<0)
			error("[Child_cat::sort_prizes_file] Error duplicating pipe descriptor.");

		// close not used pipe descriptors.
		close(p1[0]);
		close(p1[1]);
		close(p2[0]);
		close(p2[1]);

		// Executing cat command.
		execlp("cat", "cat", "./Winners.txt",NULL);
		error("[Child_cat::sort_prizes_file] Error exec cat.");
	}

	// Second  Child --> sort -nr > grep "5+0"
	if ((pid2=fork())==0)
```

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	16/06/2018	15:30

```
{
        // redirect stdin -> output pipe (p1[0])
        close(0);
        if (dup(p1[0])<0)
                error("[Child_cat::main] Error duplicating pipe descriptor.");

        // redirect stdout -> Winners.txt file.
        close(1);
        if (dup(p2[1])<0)
                error("[Child_cat::main] Error duplicating pipe descriptor.");

        // close not used pipe descriptors.
        close(p1[0]);
        close(p1[1]);
        close(p2[0]);
        close(p2[1]);

        execlp("sort","sort", "-nr", NULL);
        error("[Child_sort::sort_prizes_file] Error exec sort.");
}

// Third  Child --> grep "5+0" > Winners_sorted.txt
if ((pid5=fork())==0)
{
        // redirect stdin -> output pipe (p2[0])
        close(0);
        if (dup(p2[0])<0)
                error("[Child_cat::main] Error duplicating pipe descriptor.");

        // redirect stdout -> Winners.txt file.
        close(1);
        prizes_file = open("./Winners_sorted.txt", O_WRONLY|O_CREAT|O_TRUNC, 0640);
        if (prizes_file<0)
                error("[Child_sort::main] Error open output prizes file.");

        // close not used pipe descriptors.
        close(p1[0]);
        close(p1[1]);
        close(p2[0]);
        close(p2[1]);

        execlp("grep","grep", "5+0", NULL);
        error("[Child_grep::sort_prizes_file] Error exec grep.");
}

// close not used pipe descriptors.
close(p1[0]);
close(p1[1]);
close(p2[0]);
close(p2[1]);
```

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	16/06/2018	15:30

```
        // Waiting for cat and sort child processes.
        wait(&cod_cat);
        wait(&cod_sort);
        wait(&cod_grep);
        if ( !WIFEXITED(cod_cat) || WEXITSTATUS(cod_cat) || !WIFEXITED(cod_sort) || WEXITSTATUS(cod_sort) ||
!WIFEXITED(cod_grep) || WEXITSTATUS(cod_grep))
                error("[sort_prizes_file] Error in sorting comands.");
```

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	16/06/2018	15:30

1.

```
...
voidsort_prizes_file()
{
    int pid1, pid2, pid3, pid4, cod_cat, cod_sort, cod_cp, cod_rm;
    intprizes_file;
    int p1[2];

    if (pipe(p1)<0)
        error("[concur2ok::sort_prizes_file] Error creatingpipe.");
    if ((pid1=fork())==0)
    {
        close(1);
        if (dup(p1[1])<0)
            error("[Child_cat::sort_prizes_file] Error duplicatingpipe
descriptor.");
        close(p1[0]);
        close(p1[1]);

        execlp("cat", "cat", "./Winners.txt",NULL);
        error("[Child_cat::sort_prizes_file] Error exec cat.");
    }

    if ((pid2=fork())==0)
    {
        close(0);
        if (dup(p1[0])<0)
            error("[Child_cat::main] Error duplicatingpipe descriptor.");
        close(1);
        prizes_file = open("./Winners_sorted.txt", O_WRONLY|O_CREAT|O_TRUNC,
0640);
        if (prizes_file<0)
            error("[Child_sort::main] Error open output prizesfile.");
        close(p1[0]);
        close(p1[1]);

        execlp("sort","sort", "-nr", NULL);
        error("[Child_sort::sort_prizes_file] Error exec sort.");
    }

    // closenotusedpipe descriptors.
    close(p1[0]);
    close(p1[1]);

    wait(&cod_cat);
    wait(&cod_sort);

    if ((pid3=fork())==0)
    {
        execlp("cp", "cp", "Winners_sorted.txt", "Winners.txt", NULL);
        error("[Child_cp::sort_prizes_file] Error exec cat.");
    }
    wait(&cod_cp);

    if ((pid4=fork())==0)
    {
        execlp("rm","rm", "Winners_sorted.txt", NULL);
        error("[Child_rm::sort_prizes_file] Error exec cat.");
    }
    wait(&cod_rm);
}

...
```

Text 1: Fragment de la solució de concur3ok

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	16/06/2018	15:30

4.- Concurrència [2.5 punts]

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) Suposeu que inicialment, només es produeix el producte P i que ens garanteixen que només hi ha un thread produïnt aquest producte.

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

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

    /* 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	16/06/2018	15:30

b) Suposeu que el producte Q necessita un components B, dos components A i un component C. 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 P simultàneament ni dos o més productes Q simultàniament. Ara bé, sí que es podrà produir simultàniament un producte P i un producte 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 mutexP; /* init to 1 */
sem_t mutexQ; /* init to 1 */
```

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

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

/* Produce P */
...

sem_signal(mutexP);
}
```

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

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

/* Produce Q */
...

sem_signal(mutexQ);
}
```