Espai grapa

Universitat Oberta de Catalunya

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	09/06/2018	12:00

05.566 09 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	09/06/2018	12:00

Enunciats

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	09/06/2018	12:00

1.- Preguntes breus

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

Un procés passa de Ready a Run quan el procés actualment a Run perd la CPU i el planificador de la CPU tria, entre els processos que estan a Ready, a aquest procés per executar-se.

Un procés passarà de Ready a Wait si rep un signal d'un altre procés que el bloquegi mitjançant, per exemple, la crida al sistema kill(pid, SISTOP).

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 0x0FFF i 0x2000 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.

Tot i que les adreces lògiques no són consecutives, corresponen a pàgines lògiques diferents no consecutives (la 0x0 i la 0x2 respectivament, resultat de descartar els 12 bits baixos de les direccions lògiques perquè les pàgines són de 2^12 bytes). Per tant, aquestes adreces lògiques estaran mapejades a pàgines físiques diferents. Gràcies a paginació, aquestes pàgines físiques poden ser dos pàgines físiques qualsevol, amb el que podrien ser consecutives.

c) Els dispositius d'entrada/sortida poden classificar-se com a "físics", "lògics" i "virtuals". En què es diferencien un dispositiu "físic" i un "lògic"? 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);
...
```

El dispositiu físic és el que ens ofereix el hardware. El dispositiu lògic és una abstracció que ofereix el SO. Per exemple, el disc dur és un dispositiu físic però els fitxers són els dispositius lògics que el SO abstreu sobre aquest dispositiu físic.

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

d) Què és un dealock? En què consisteix la condició necessària per a 'existència de deadlock "Retenció i espera"? Com es podria garantir que mai es satisfarà?

"Retenció i espera" consisteix en que dos o més threads tenen assignat algun recurs i, a més a més, es troben bloquejats esperant la concesió d'algun altre recurs. Aquesta condició podria evitar-se si la sol·licitud dels recursos es realitzés de forma atòmica: o el thread rep tots els recursos o el thread es bloqueja, sense cap recurs assignat, fins que tots els recursos estiguin disponibles.

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	09/06/2018	12:00

2.- Memòria

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

- Mida adreça física/lògica: 20 bits.
- Mida memòria física: 32 KB.
- Mida pàgina/frame: 4 KB.

Tenim la següent informació de la ubicació d'un procés en memòria:

Regió	Adreça lògica d'inici	Mida
Codi	001000h	14 KB
Dades	008000h	24 KB
Pila	0FE000h	7 KB

Taula de pàgines

	V	P	frame
0h	0	0	8h
1h	1	1	1h
2h	1	1	7h
3h	1	1	3h
4h	1	0	3h
5h	0	0	1h
6h	0	0	6h
7h	0	0	4h
8h	1	1	0h
9h	1	0	6h

	V	P	frame
Ah	1	1	6h
Bh	1	1	4h
Ch	1	0	4h
Dh	1	0	6h
FBh	0	0	
FCh	0	0	35h
FDh	0	0	2Dh
FEh	1	1	5h
FFh	1	0	2h

Tingueu en compte que totes les pàgines en el rang 0xE-0xFA són invàlides.

Contestar, justificant les respostes, a les següents preguntes:

a) En el cas que es tractés d'un sistema de paginació pura, quins canvis es produirien en els camps de la taula de pàgines del procés? Per què?

Seria possible la creació del procés A en un sistema de paginació pura? Per que?

En el cas de paginació pura, no es necessita el camp P (Pàgina Present) ja que no hi ha memòria virtual i totes les pàgines es troben en memòria física.

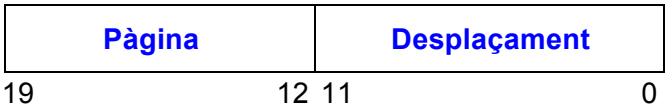
La paginació pura obliga que totes les pàgines del procés es trobin assignades a un frame de memòria física. Per tant, no seria possible crear el procés A, degut a que necessita 45 KB de memòria física i el sistema només disposa de 32 KB.

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	09/06/2018	12:00

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

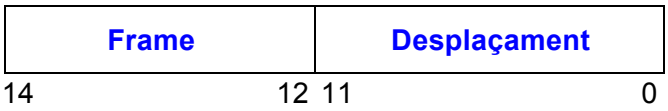
L'adreça lògica té 20 bits que es descomponen de la següent manera:



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

Pàgina = $\log_2$(Nombre_pags) = $\log_2$(2^20/4096) = 8 bits

Per accedir a tota la memòria física tenim 15 bits:



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

Frame = $\log_2$(Nombre_frames) = $\log_2$(2^15/4096) = 3 bits

c) Indicar en la següent taula la memòria física assigna al procés, indicant si la pàgina pertany a la regió de codi (C), dades (D) o a la pila (P) i el nombre de pàgina dins d'aquesta regió (per exemple, C1, C2, D1, P2).

Dir. Física	Content
0x00000	**D1**
0x01000	**C1**
0x02000	
0x03000	**C3**
0x04000	**D4**
0x05000	**P2**
0x06000	**D2**
0x07000	**C2**
0x08000	
0x00000	
0x0A000	
0x0B000	
0x0C000	

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	09/06/2018	12:00

0x0D000	
0x0E000	
0x0F000	

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

- **Adr. Lògica 01C6H -> Fallada de pàgina**

  Pàgina: 00h Despl: 01C6h

  Frame: TaulaPàgines[Pàgina] = TaulaPàgines[0h] = No vàlida

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	09/06/2018	12:00

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.

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.

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

```
...
intconcurrent_evaluate_tickets(char *infilename, intoutfile, char *n1, char *n2,
char *n3, char *n4, char *n5, char *n6, char *cmp)
{
     intpid, tickets_file;

     pid = fork();
     if (pid< 0)
          error("[concur2ok::concurrent_evaluate_tickets] Error
creatingchildprocess.");

     if (pid==0)
     {
          close(0);
          tickets_file = open(infilename, O_RDONLY, 0640);
          if (tickets_file<0)
               error("[concur2ok::concurrent_evaluate_tickets] Error open
ticketsfile.");

          close(1);
          if (dup(outfile)<0)
               error("[concur2ok::concurrent_evaluate_tickets] Error dupstdout.");
          close(outfile);

          execlp("./calc_prizes2ok","calc_prizes2ok", n1, n2, n3, n4, n5, n6, cmp,
NULL);
          error("[concur2ok::concurrent_evaluate_tickets] Error in execlp on
childprocess.");
     }
     return(pid);
}
...
```

Text 1: Fragment de la solució de concur2ok

Es demana:

a) Contesteu les següents preguntes referents al codi:

   i. ¿Perquè no cal tancar el descriptor tickets_file i si es necessita tancar el descriptor outfile?

   El descriptor tickets_ile no es necessita tancar a causa de que no s'ha duplicat i només hi ha una còpia.

Mentre que el descriptor outfile si que es necessita tancar perquè ha estat duplicat i tenim dues còpies.

ii. Després de crear correctament el procés fill, indicar el valor de la variable pid en el procés pare i al fill.

En el procés pare, la variable pid conté el pid del nou procés creat. Al fill, la variable pid conté un 0.

iii. Indicar una situació en la que s'executarà el codi que hi ha a continuació del execlp.

Aquest codi només s'executa si es produeix un error en la crida al sistema *execlp* i no es pot produir el recobriment. Si la crida s'executa correctament, es canvia el codi del procés per la del nou programa i es comença a executar des del *main*.

b) Modificar la solució proposada per incloure les següents modificacions:

i. Que els processos calc_prizes2ok llegeixin directament el fitxer de tiquets, en comptes de redirigir-ho a la seva stdin. No cal implementar la lectura en el procés *calc_prizes2ok*, només teniu que modificar la forma en què s'invoca al programa des del procés *concur2ok*.

ii. Que els resultats en el fitxer de sortida estiguin ordenats de major a menor en funció dels nombres encertats. Assumiu que no coneixeu el nom del fitxer de sortida associat amb el descriptor outfile.

Implementeu la solució per a un únic procés *calc_prizes2ok*, dins de la funció *concurrent_evaluate_tiquets()* i explicar com es generalitzaria per a la resta de processos.

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

```
int concurrent_evaluate_tickets(char *infilename, int outfile, int pipe_sort, char
*n1, char *n2, char *n3, char *n4, char *n5, char *n6, char *cmp)
{
    int pid, tickets_file;

    if (pipe_sort==0) // Reading descriptor for calc_prizes -> sort pipe
    {
        // Sorting process --> sort -nr > outfile
        if ((pid2=fork())==0)
        {
            // redirect stdin -> output pipe (pipe_sort)
            close(0);
            if (dup(pipe_sort)<0)
                error("[Child_cat::main] Error duplicating pipe descriptor.");
```

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	09/06/2018	12:00

```
            close(pipe_sort);

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

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

    pid = fork();
    if (pid < 0)
        error("[concur2ok::concurrent_evaluate_tickets] Error creating child
process.");

    if (pid==0)
    {
        // Redirect stdout to outfile.
        close(1);
        if (dup(outfile)<0)
            error("[concur2ok::concurrent_evaluate_tickets] Error dup stdout.");
        close(outfile);

        // Child process.
        execlp("./calc_prizes2ok","calc_prizes2ok", n1, n2, n3, n4, n5, n6, cmp,
infilename, NULL);
        error("[concur2ok::concurrent_evaluate_tickets] Error in execlp on child
process.");
    }

    return(pid);
}
```

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	09/06/2018	12:00

Examen 2017/18-2

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	09/06/2018	12:00

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

```
void produce_P()
{
    sem_wait(semC);
    sem_wait(semB);
    sem_wait(semA);
    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	09/06/2018	12:00

b) Suposeu que el producte Q necessita dos components C, un component B 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, en tot moment només pot haver un producte en producció i que després de produir un producte P s'ha de produir un producte Q, que després d'un producte Q s'ha de produir un producte P i així successivament. El primer producte que es produirà serà de tipus P.
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 syncP; /* init to 1 */
sem_t syncQ; /* init to 0 */
```

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

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

/* Produce P */
...

sem_signal(syncQ);
}
```

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

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

/* Produce Q */
...

sem_signal(syncP);
}
```