Espai grapa

Examen 2019/20-1

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	18/01/2020	15:30

⊂05.566ℜ18ℜ01ℜ20ℜEΞ℘∈
05.566 18 01 20 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 foli mida DIN-A4/foli amb contingut lliure. En cas de poder fer servir calculadora, de quin tipus?

  PROGRAMABLE
- Si hi ha preguntes tipus test: Descompten les respostes errònies? **NO** Quant?
- Indicacions específiques per a la realització d'aquest examen:

Examen 2019/20-1

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	18/01/2020	15:30

Enunciats

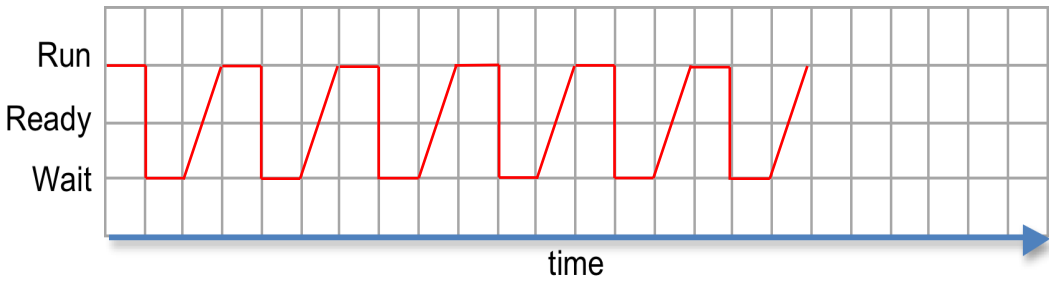
Examen 2019/20-1

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	18/01/2020	15:30

1. Teoria [2.5 punts]

Contestar **justificadament** les següents preguntes:

a) Indicar quinas característiques ha de tenir el sistema operatiu (monoprogramat, multiprogramat) que permeti el següent cronograma d'evolució dels estats d'un procés.



Aquest cronograma implica una transició directa des de l'estat d'espera a l'estat d'execució sense passar per l'estat de preparat. Això només es pot produir en un sistema operatiu monoprogramat a on només hi ha un procés en execució al mateix temps.

b) 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 resultat esperat és de x = 15 i que el semàfor SemA està inicialitzat a 1

```
Thread1:
   sem_wait(SemA);
   x =5;
   sem_signal(SemA);
```

```
Thread2:
   sem_wait(SemA);
   x =3*x;
   sem_signal(SemA);
```

L'execució d'aquest programa concurrent és indeterminista pel fet que el resultat final de la variable x depèn en quin ordre s'executi les modificacions. Si primer s'executa el Thread1 i després el Thread2 el resultat de x és 15. En canvi, si primer s'executa el Thread2 i després el Thread1, el resultat és 5.

El problema d'aquest programa no son les condicions de carrera, ja que tots dos processos implementen l'exclusió mútua en modificar la variable compartida. El problema és de precedència ja que el Thread2 no pot executar la seva modificació fins que el Thread1 hagi realitzat la seva.

Per solucionar el problema, utilitzaríem un segon semàfor de sincronització en el qual el Thread2 s'esperaria fins que el Thread1 hagués realitzat la seva modificació.

c) En un sistema de gestió de memòria basat en paginació amb memòria virtual, indiqueu baix quines circumstàncies un frame s'escriu al swap o es llegeix del swap.

Un frame s'envia al swap quan es necessita la seva memòria per a un altre procés i es necessita salvar el seu contingut en el disc per poder recuperar-la més tard.

Examen 2019/20-1

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	18/01/2020	15:30

Un frame es llegeix del swap quan es produeix una fallada de pàgina i es necessita accedir a la informació d'aquesta pàgina, de manera que cal assignar-li un frame i llegir el seu estat des del swap.

d) Indicar les crides al sistema que necessita utilitzar l'intèrpret d'ordres per executar un programa en mode foreground (primer pla).

Necessita utilitzar la crida al sistema fork per crear un procés fill, a continuació sobre el procés fill ha de realitzar un recobriment (crida al sistema exec*) per carregar el nou programa. Com s'executa en primer pla, l'intèrpret d'ordres no pot continuar amb la seva execució (i demanar una nova ordre) fins que el programa anterior hagi acabat, de manera que necessita invocar la crida al sistema wait. Finalment. quan el programa del procés fill finalitzi invocarà a la crida al sistema exit.

Examen 2019/20-1

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	18/01/2020	15:30

2. Memòria (2,5 punts = 0,5 cada apartat)

Sigui un sistema de gestió de memòria basat en paginació sota demanda on les pàgines tenen una mida de 64KBytes, les adreces lògiques són de 20 bits i l'espai físic és de 512 KBytes.
Sobre aquest sistema es creen dos processos.

- Procés 1: el seu fitxer executable determina que el codi ocuparà dues pàgines, que les dades inicialitzades n'ocupen dues, les no inicialitzades una i la pila dues-.
- Procés 2: el seu fitxer executable determina que el codi ocuparà una pàgina, no hi ha dades inicialitzades, les dades no inicialitzades ocuparan una pàgina i que la pila ocuparà dues pàgines.

Es demana:

1 Estimeu la mida del fitxer executable corresponent al procés 1.

2 Suposant que les pàgines es carreguen a memòria física tal i com indica el diagrama següent, indiqueu quin serà el contingut de les taules de pàgines de tots dos processos (podeu contestar sobre el diagrama de l'enunciat). Considereu com a invàlides les entrades marcades amb el símbol "-".

3 Suposant que el procés en execució és el procés 1, indiqueu quines seran les adreces físiques corresponents a les següents adreces lògiques: 0x4C122 i 0xE4228. Variaria la resposta si el procés en execució fos el procés 2? En cas afirmatiu, indiqueu el motiu i com canviaria.

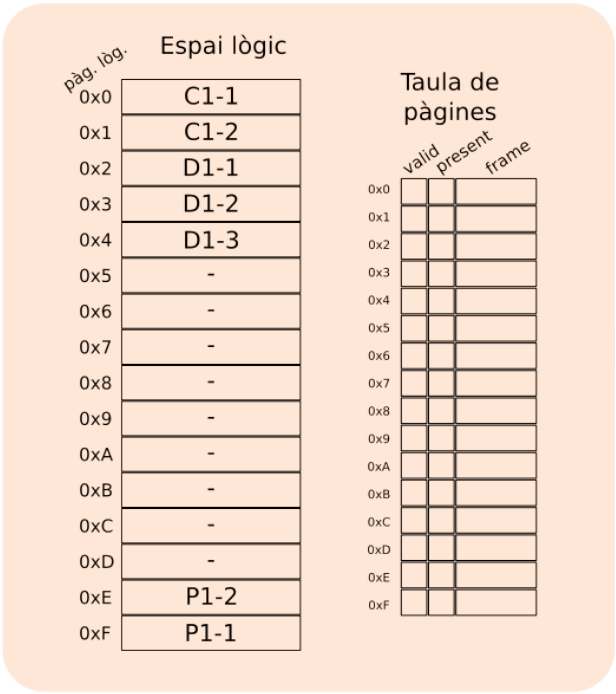
4 Indiqueu dues adreces lògiques vàlides i consecutives del procés 1 que siguin traduïdes a adreces físiques consecutives. Si no és possible, expliqueu.ne el motiu.

5 Indiqueu dues adreces lògiques vàlides i consecutives del procés 2 que **no** siguin traduïdes a adreces físiques consecutives. Si no és possible, expliqueu-ne el motiu.

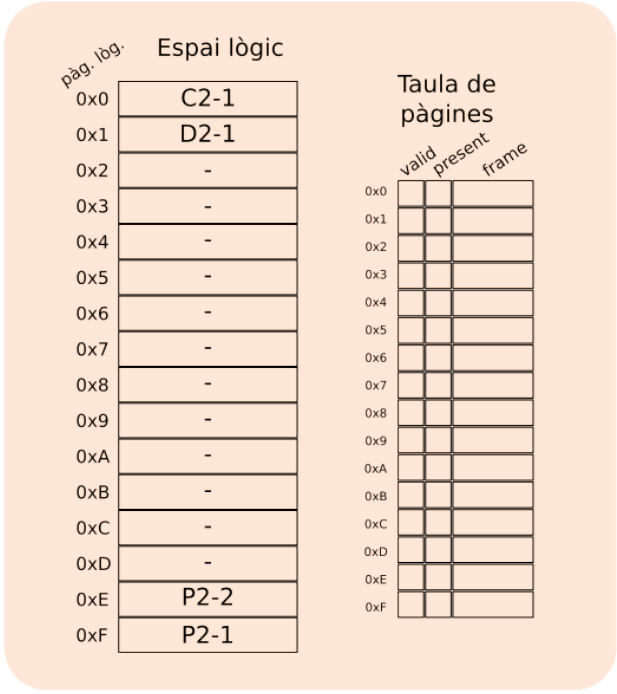
Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	18/01/2020	15:30

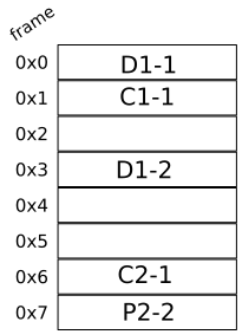
Procés 1

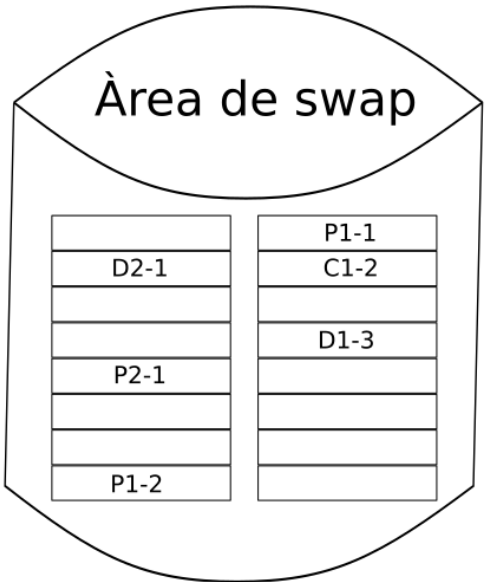


Procés 2



Espai físic





Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	18/01/2020	15:30

1 – A l'executable tindrem una capçalera i les zones de codi i dades inicialitzades. Si ometem la mida de la capçalera i considerem que no hi ha fragmentació interna a aquestes zones, la mida de l'executable serà l'equivalent a 4 pàgines, és a dir, 256 KB. Si hi hagués fragmentació interna a les zones de codi o de dades inicialitzades, la mida de l'executable seria inferior.

2-

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	18/01/2020	15:30

3-

@L	@F (procés 1)	@F (procés 2)
0x4C122	Excepció fallada de pàgina	Excepció adreça invàlida
0xE4228	Excepció fallada de pàgina	0x74228

La traducció és diferent perquè cada procés té una taula de pàgines diferent.

4- Les adreces lògiques 0x00000 i 0x00001 són consecutives i seran traduïdes a adreces físiques consecutives perquè corresponen a una mateixa pàgina lògica (la 0x0) present a memòria física. Concretament seran traduïdes a les adreces físiques 0x10000 i 0x10001.

5- No és possible perquè necessitaríem que dues pàgines lògiques consecutives del procés 2 fossin presents. Analitzant la taula de pàgines del procés 2, veiem que aquesta condició no es compleix.

Examen 2019/20-1

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	18/01/2020	15:30

3. Processos (2,5 punts = 1,0 + 1,5)

a) Indiqueu quin serà el resultat d'executar els següents programes (jerarquia de processos creada, informació mostrada per cada procés per la sortida estàndar i en quin ordre, paràmetres esperats, ...) . Podeu assumir que cap crida al sistema retornarà error.

i)

```
main(int argc, char *argv[])
{
   int i;

   for (i=1; i < argc; i++) {
      execlp(argv[i], argv[i], NULL);
   }
   exit(0);
}
```

ii)

```
main()
{
   int fd[2], p;
   char c='b';

   pipe(fd);
   p = fork();
   if (p>0) {
      write(fd[1], "a",1);
      wait(NULL);
   }
   else read(fd[0], &c, 1);

   write(1, &c, 1);
   exit(0);
}
```

b) Escriviu un programa tal que, utilitzant les crides al sistema vistes a l'assignatura, creï cinc processos fills que executin concurrentment el programa "child". Quan algun procés fill mori, el procés pare ho mostrarà per la sortida estàndar i haurà de crear un nou procés fill que el remplaci i passi a executar "child". Quan el pare hagi creat un total de 20 fills, deixarà de crear-ne més; quan tots els fills hagin mort, el procés pare finalitzarà.

S'adjunta un possible exemple de l'execució (tots els missatges els escriu el procés pare). Les 5 primeres línies sempre hauran d'aparèixer en aquest ordre. Les següents dependran de l'ordre en que els fills vagin morint.

```
promtp$ ./father
Father starts
Child number 0 has been created
Child number 1 has been created
Child number 2 has been created
Child number 3 has been created
Child number 4 has been created
Child number 2 has finished
Child number 5 has been created
Child number 0 has finished
Child number 6 has been created
...
Child number 13 has finished
Child number 19 has finished
Child number 12 has finished
Child number 5 has finished
Child number 18 has finished
Father finishes
prompt$
```

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	18/01/2020	15:30

No és precís que indiqueu el tractament d'errors a les crides al sistema ni els includes. Sí que és precís que les crides al sistema estiguin correctament parametritzades.

a-i)
El programa entra un bucle on executa la crida execlp assumint que el paràmetre argv[1] serà el nom d'un fitxer executable. Ara bé, com la crida execlp substitueix el codi del procés, no s'executarà cap altra iteració del bucle. Per tant, el codi executa el programa indicat com a paràmetre argv[1] i no crea cap procés.

a-ii)
El programa crea una pipe i a continuació un procés fill. El pare escriu a la pipe el caràcter 'a' i espera la mort del fill. El fill llegeix un caràcter de la pipe, amb el que llegirà la 'a' i la guardarà a la variable c, l'escriu per stdout i mor. Ara el pare es desbloquejarà i escriurà per stdout el valor de la variable c (que el pare no ha modificat) i per tant escriurà 'b'. En conseqüència, el programa escriu per stdout "ab" i crea un procés fill.

b)

```
main()
{
	int n=0, j, dead=0, pids[20], p;

	printf("Father starts\n");
	while (dead<20) {
		if (n>4) {
			p = wait(NULL);
			for (j=0; pids[j] != p; j++);
			printf("Child number %d has finished\n", j);
			dead++;
		}
		if (n<19) {
			if ((pids[n] = fork()) == 0)
				execlp("child", "child", NULL);
			printf("Child number %d has been created\n", n);
			n++;
		}
	}
	printf("Father finishes\n");
}
```

Examen 2019/20-1

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	18/01/2020	15:30

4. Concurrencia [2.5 puntos]

S'ha inaugurat un nou bar de tapes en un local amb una entrada molt estreta per on només hi pot passar una persona.

Utilitzant les següents operacions de semàfors:

- `sem_init(semaphore s, int valor)`. Inicialitza el semàfor *s* amb *valor* 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) Després d'una inspecció de la Conselleria d'Indústria, l'amo del bar es veu obligat a limitar l'aforament del bar a 100 persones. Assumiu que els clients que no poden entrar es queda esperant fins que hi hagi lloc lliure. Escriviu el codi que descriu, per a un client, el procediment d'entrar i el de sortir del bar.

Declaració variables i semàfors

```
Semaphore SemMutex, SemAforament;
int aforament=0;
```

Inicialització

```
sem_init(SemMutex,1);
sem_init(SemAforament, 100);
```

Entrar()	**Sortir()**

{
sem_wait(SemAforament);
sem_wait(SemEntrada);
	{
	sem_wait(SemEntrada);

Entra;	Surt;

sem_signal(SemEntrada);
}
 	
	sem_signal(SemEntrada);
	sem_signal(SemAforament);
	}

b) Després d'una inspecció de la Conselleria de la conselleria d'igualtat, l'amo del bar es veu obligat a garantir una accés equilibrat entre nois i noies, així que decideix que han d'anar entrant al bar de forma alterna, primer una noia, després un noi, després una noia, etc. A més, assumiu que ara els

Examen 2019/20-1

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	18/01/2020	15:30

clients que no poden entrar es van a un altre local. Escriviu el codi que descriu, per a un client (noi i noia), el procediment que fa servir per entrar al bar.

Declaració variables i semàfors

```
Semaphore SemNoia, SemNoi, SemEntrada,
          SemMutex;
int Aforament=0;
```

Inicialització

```
sem_init(SemNoia,1);
sem_init(SemNoi,0);
sem_init(SemEntrada,1);
sem_init(SemMutex,1);
```

EntrarNoia()

```
{
    /* Comprovem si hi ha espai lliure al local, si no ens
       anem a un altre lloc */
    sem_wait(SemMutex);
    if (NAforament==100) {
        sem_signal(SemMutex);
        retorn;
    }
    NAforament++;
    sem_signal(SemMutex);

    /* Esperem que li toqui el torn a una noia */
    sem_wait(SemNoia );
    /* Entrem d'1 en 1*/
    sem_wait(SemEntrada);
```

Entra;

```
    sem_signal( SemEntrada );

    /* Li toca el torn a un noi */
    sem_signal(SemNoi );

}
```

EntrarNoi()

```
{
    /* Comprovem si hi ha espai lliure al local, si no ens
       anem a un altre lloc */
    sem_wait(SemMutex);
    if (NAforament==100) {
        sem_signal(SemMutex);
        retorn;
    }
    NAforament++;
    sem_signal(SemMutex);

    /* Esperem que li toqui el torn a un noi */
    sem_wait(SemNoi);
    /* Entrem d'1 en 1*/
    sem_wait(SemEntrada);
```

Entra;

```
    sem_signal(SemEntrada );

    /* Li toca el torn a una noia */
    sem_signal(SemNoia );

}
```

Sortir ()

```
{

    sem_wait(SemEntrada );
```

Surt;

```
    sem_signal( SemEntrada );
```

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	18/01/2020	15:30

```
    /* Augmentem l'espai lliure al local */
    sem_wait(SemMutex);
    aforo++;
    sem_signal(SemMutex);

}
```

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	18/01/2020	15:30

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	18/01/2020	15:30

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	18/01/2020	15:30

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	18/01/2020	15:30

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	18/01/2020	15:30

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	18/01/2020	15:30