Espai grapa

Examen 2019/20-1

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	22/01/2020	09:00

⊂05.566ℜ22ℜ01ℜ20ℜEΞΘ∈
05.566 22 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	22/01/2020	09:00

Enunciats

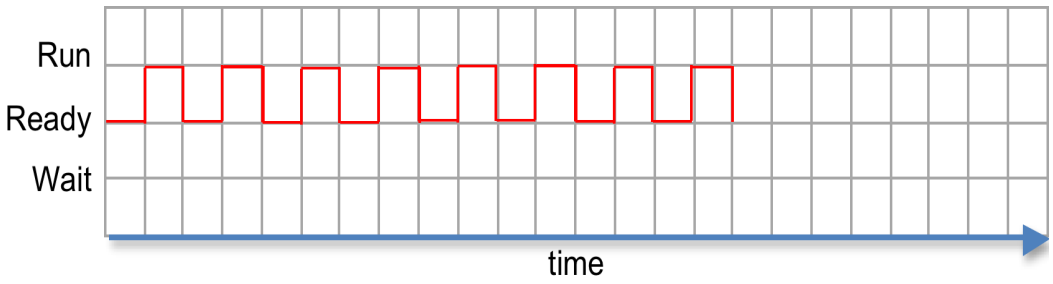
Examen 2019/20-1

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	22/01/2020	09:00

1. Teoria [2.5 punts]

Contestar **justificadament** les següents preguntes:

a) Indiqueu en quina situació (tipus de sistema operatiu, monoprogramat, multiprogramat) es pot obtenir el següent cronograma d'evolució dels estats d'un procés.



Aquest cronograma només és possible en un sistema operatiu multiprogramat en el qual hi hagi diversos processos que s'estiguin executant al mateix temps i per tant, comparteixin la CPU.

b) Tenim dos threads que 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 a que es degut i com es pot solucionar. Assumiu que el semàfor SemA està inicialitzat a 0.

Thread1:	Thread2:
`x = x+1;`	`sem_wait(SemA);`
`sem_signal(SemA);`	`x =x+2;`

L'execució d'aquest programa és determinista, pel fet que el semàfor de sincronització utilitzat per garantir la precedència (que el Thread2 s'executi després que el Thread1 hagi modificat la variable x) evita que es produeixin condicions de carrera i garanteix que el resultat final sempre és el correcte (x + 3).

c) En un sistema de gestió de memòria basat en paginació sense memòria virtual, És possible que dos processos puguin compartir una part de la memòria en direccions lògiques diferents?

Si que és possible, només cal que les dues direccions lògiques de tots dos processos apuntin al mateix frame de memòria física dins de les seves respectives taules de pàgines.

d) Indicar les crides al sistema que necessita utilitzar l'intèrpret d'ordres per executar la següent comanda:

>ls -la | sort> file.txt

Primer, l'intèrpret d'ordres necessita crear una pipe per comunicar les dues comandes, per a això utilitza la crida al sistema pipe. A continuació utilitza la crida al sistema fork per crear dos processos fill (un per al "ls" i un altre per al "sort"). Seguidament ha de redirigir la stdout del procés fill "ls" a l'entrada del pipe i la sortida del pipe a la stdin del procés fill "sort", per a això utilitza les crides al sistema close i dup (o alternativament dup2). Finalment realitza el recobriment en ambdós processos (crida al sistema exec*) per carregar els programes de les

Examen 2019/20-1

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	22/01/2020	09:00

ordres que ha d'executar cada un d'ells. Quan els programes fills finalitzin invocaran a la crida al sistema exit.

Examen 2019/20-1

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	22/01/2020	09:00

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:

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

b) 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 "-".

c) 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.

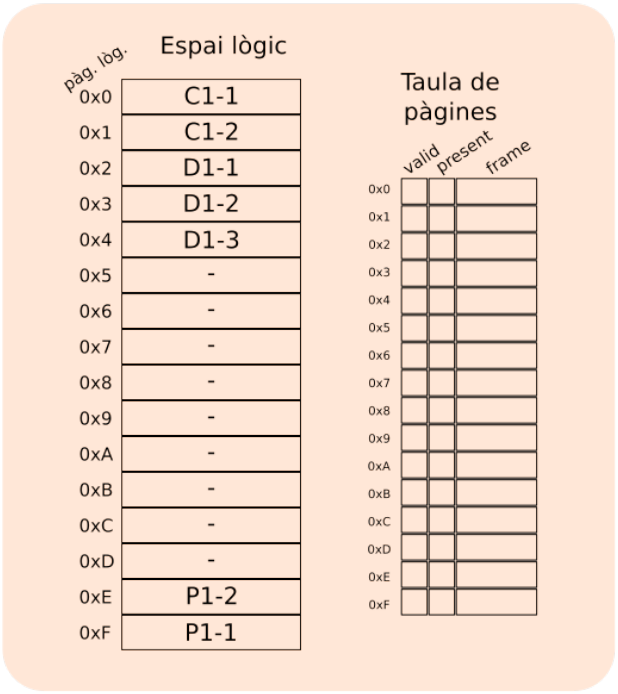
d) Indiqueu dues adreces lògiques vàlides i consecutives del procés 1 que siguin traduïdes a adreces físiques consecutives. Si no és possible, expliqueu.ne el motiu.

e) 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.

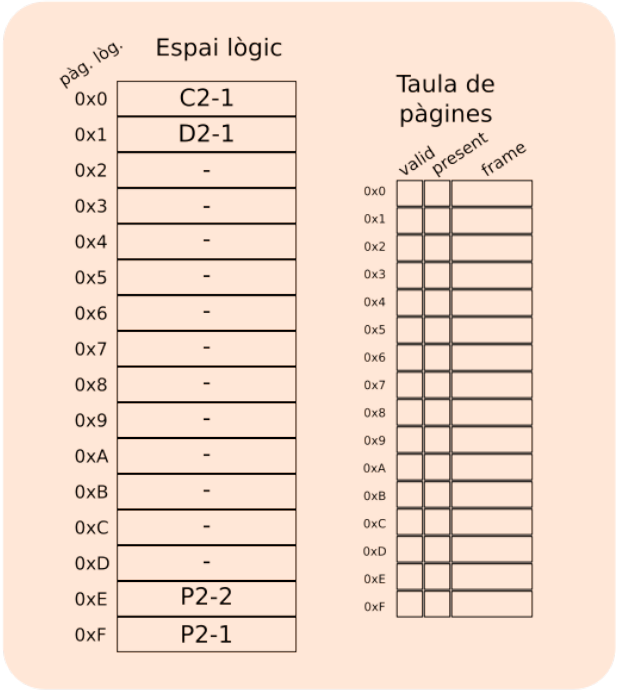
Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	22/01/2020	09:00

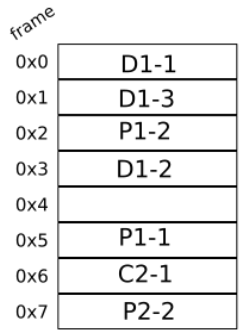
Procés 1

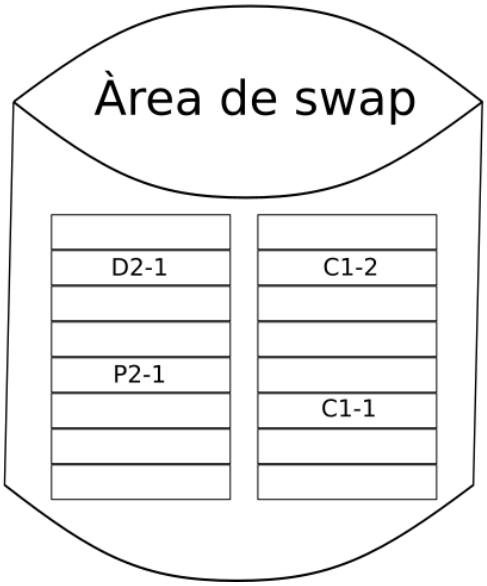


Procés 2



Espai físic



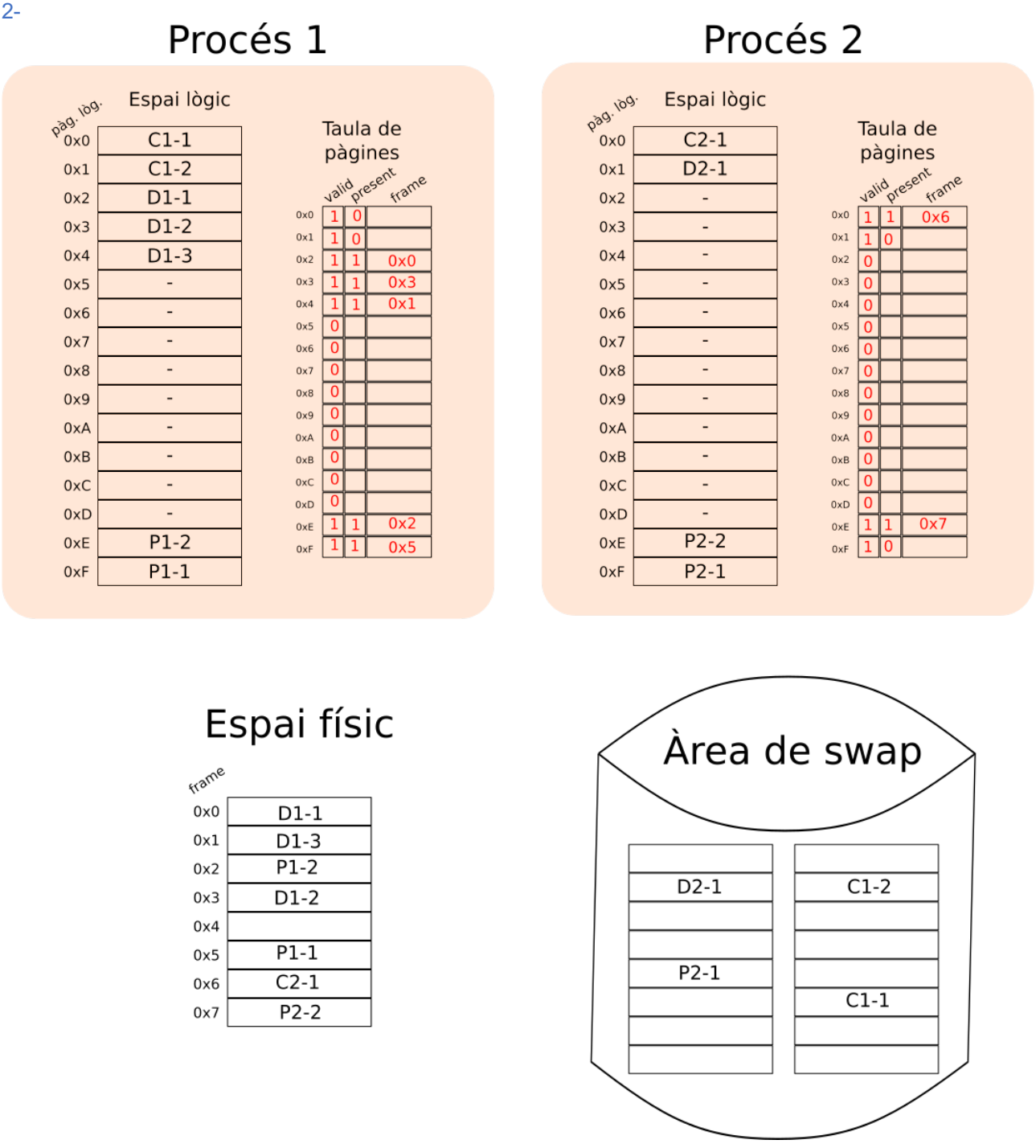


Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	22/01/2020	09:00

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	22/01/2020	09:00

3-

@L	@F (procés 1)	@F (procés 2)
0x4C122	0x1C122	Excepció adreça invàlida
0xE4228	0x24228	0x74228

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

4- Les adreces lògiques 0x40000 i 0x40001 són consecutives i seran traduïdes a adreces físiques consecutives perquè corresponen a una mateixa pàgina lògica (la 0x4) 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	22/01/2020	09:00

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++) {
      if (fork() > 0) {
         execlp(argv[i], argv[i], NULL);
      }
   }
   exit(0);
}
```

ii)

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

   pipe(fd);
   p = fork();
   if (p>0) {
      wait(NULL);
      write(fd[1], "a",1);
   }
   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 tres processos fills que executin concurrentment el programa "child". Quan el pare detecti que han mort tots els fills, el procés pare tornarà a crear tres nous processos fills que executin concurrentment "child". Quan aquests morin, el procés pare finalitzarà.

S'adjunta un possible exemple de l'execució (tots els missatges els escriu el procés pare). Les 3 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 1 has finished
Child number 0 has finished
Child number 2 has finished
Child number 3 has been created
Child number 4 has been created
Child number 5 has been created
Child number 5 has finished
Child number 4 has finished
Child number 3 has finished
Father finishes
prompt$
```

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.

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	22/01/2020	09:00

a-i)

El programa entra un bucle que recorre el vector d'arguments i on a cada iteració es crea un procés: el procés pare executa la crida execlp assumint que el paràmetre argv[i] serà el nom d'un fitxer executable i el procés fill passa a executar la següent iteració del bucle. Per tant es creen tants processos com arguments hagi rebut i els processos executen concurrentment els programes indicats pels arguments. La jerarquia creada és pare-fill-net-besnet-...

a-ii)

El programa crea una pipe i a continuació un procés fill. El pare espera la mort del fill. El fill llegeix un caràcter de la pipe, però com la pipe és buida i existeix algun procés escriptor sobre la pipe (tant el pare com el fill), el fill es queda bloquejat a la lectura. Per tant, tots dos processos queden bloquejats i la resta de codi no s'executarà.

b)

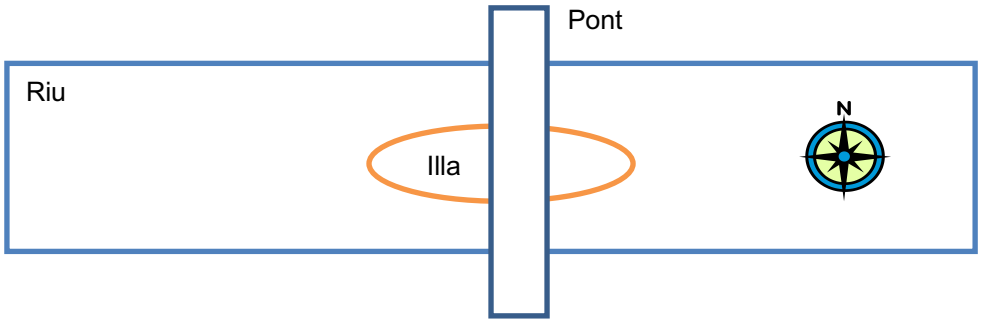
```
main()
{
	int i, j, k, pids[3], p;

	printf("Father starts\n");
	for (k=0; k<2; k++) {
		for (i=0; i<3; i++) {
			if ((pids[i] = fork()) == 0)
				execlp("child", "child", NULL);
			printf("Child number %d has been created\n", i+3*k);
		}

		for (i=0; i<3; i++) {
			p = wait(NULL);
			for (j=0; pids[j] != p; j++);
			printf("Child number %d has finished\n", j+3*k);
		}
	}
	printf("Father finishes\n");
}
```

Examen 2019/20-1

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	22/01/2020	09:00

Concurrencia [2.5 puntos]

Suposeu una carretera molt secundària en la que ens trobem amb una secció com la que es mostra a la figura:



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) El pont de la figura és molt estret i només permet que hi passi un cotxe alhora. Escriviu el codi que descriu el pas d'un cotxe de nord a sud i el de sud a nord.

Declaració variables i semàfors

```
Semaphore SemPont;
```

Inicialització

```
sem_init(SemPont,1);
```

Creuar_NS()	**Creuar_SN()**
{ sem_wait(SemPont);	{ sem_wait(SemPont);
Creuar_Pont;	Creuar_Pont;
sem_signal(SemPont); }	sem_signal(SemPont); }

Examen 2019/20-1

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	22/01/2020	09:00

b) La Conselleria de Foment decideix fer una actuació per millorar la circulació en aquest tram de manera que ara poden creuar fins a 10 cotxes pel pont al mateix temps, però només en la mateixa direcció. Escriviu el codi que descriu el pas d'un cotxe de nord a sud i el de sud a nord.

Declaració variables i semàfors

```
Semaphore SemMutexNS, SemMutexSN, SemPont,
          SemLliures;
int CotxesCreuantNS=0, CotxesCreuantNS=0;
```

Inicialització

```
sem_init(SemMutexNS,1);
sem_init(SemMutexSN,1);
sem_init(SemPont,1);
sem_init(SemLliures,10);
```

Creuar_NS()	**Creuar_SN()**

{
/* Si el pont està lliure, el bloquegem perquè no puguin passar de NS. Si no està lliure, ens esperem fins que ho estigui */
sem_wait(SemMutexNS);
CotxesCreuantNS++;
if (CotxesCreuantNS ==1)
sem_wait(SemPont)
sem_wait(SemLliures);
/* Si hi ha espai en el pont, creuem, si no ens esperem */
sem_signal(SemMutexNS);
	{
	/* Si el pont està lliure, el bloquegem perquè no puguin passar de SN. Si no està lliure, ens esperem fins que ho estigui */
	sem_wait(SemMutexSN);
	CotxesCreuantSN++;
	if (CotxesCreuantSN==1)
	sem_wait(SemPont)
	sem_wait(SemLliures);
	/* Si hi ha espai en el pont, creuem, si no ens esperem */
	sem_signal(SemMutexSN);

Creuar_Pont;	Creuar_Pont;

/* Sortim del pont, alliberem espai al pont */
sem_wait(SemMutexNS);
/* Quan no hi ha més cotxes creuant de NS, alliberem el pont. */
CotxesCreuantNS--;
if (CotxesCreuantNS==0)
sem_signal (SemPont)
sem_signal (SemLliures);
sem_signal(SemMutexNS);
}
	/* Sortim del pont, alliberem espai al pont */
	sem_wait(SemMutexSN);
	/* Quan no hi ha més cotxes creuant de SN, alliberem el pont. */
	CotxesCreuantSN--;
	if (CotxesCreuantSN==0)
	sem_signal (SemPont)
	sem_signal(SemLliures);
	sem_signal(SemMutexSN);
	}

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	22/01/2020	09:00

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	22/01/2020	09:00

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	22/01/2020	09:00

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	22/01/2020	09:00

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	22/01/2020	09:00