Espai grapa

Examen 2019/20-1

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	11/01/2020	12:00

05.566 11 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	11/01/2020	12:00

Enunciats

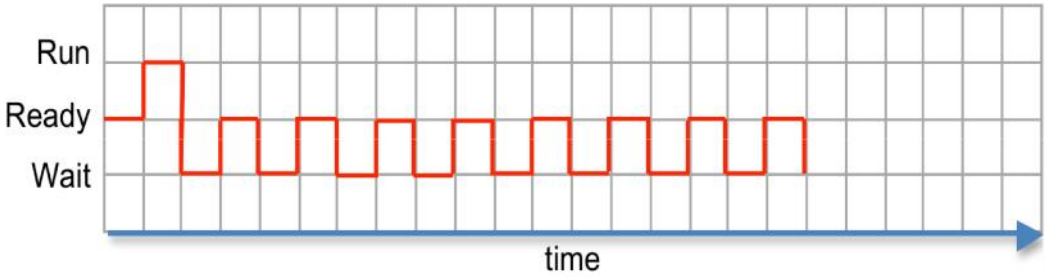
Examen 2019/20-1

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	11/01/2020	12:00

1. Teoria [2.5 punts]

Contestar **justificadament** les següents preguntes:

a) Indicar si el següent cronograma d'evolució dels estats d'un procés és possible en un sistema tipus Unix..



No és possible, ja que implica que un procés pugui passar directament de l'estat de preparat a l'estat d'espera. Això no és possible, ja que perquè un procés es bloqueja per E/S o a l'espera ha d'estar en execució.

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.

Thread1:	Thread2:
x = x+1;	x =x+2;

L'execució d'aquest programa concurrent és indeterminista degut a que tots dos threads poden modificar de forma simultània la variable x, generant condicions de carrera. Aquest problema de indeterminisme es pot solucionar, implementant una exclusió mútua a l'hora de modificar la variable x, que ens garanteixi que no la poden modificar al mateix temps.

c) En un sistema de gestió de memòria basat en paginació sense memòria virtual, és possible executar un programa assignant-li un sol frame de memòria física.

No és possible, ja que si no s'utilitza memòria virtual, es requereix que tota la memòria d'un procés es trobi ubicada a memòria física.

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

sort < file.txt &

El intèrpret d'ordres necessita utilitzar la crida al sistema fork per crear un procés fill. Per redirigir el fitxer "file.txt" a la stdin del procés fill ha d'utilitzar les crides al sistema close per tancar la stdin i l'open per obrir el fitxer de lectura sobre aquest descriptor. A continuació, el procés fill ha de realitzar un recobriment (crida al sistema exec*) perquè el procés executi el programa sort. 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	11/01/2020	12:00

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:

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.

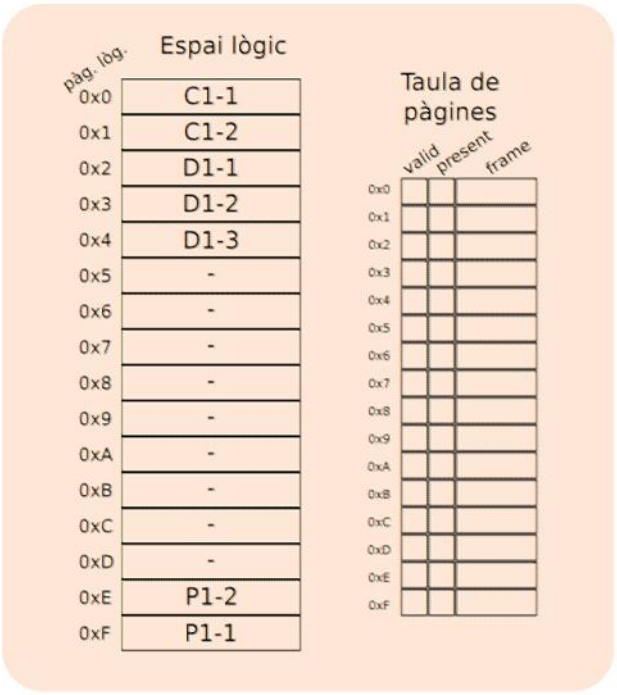
4 Indiqueu dues adreces lògiques vàlides i consecutives del procés 1 que siguin traduïdes a adreces físiques consecutives. Si no és possible, expliqueu.ne el motiu.

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.

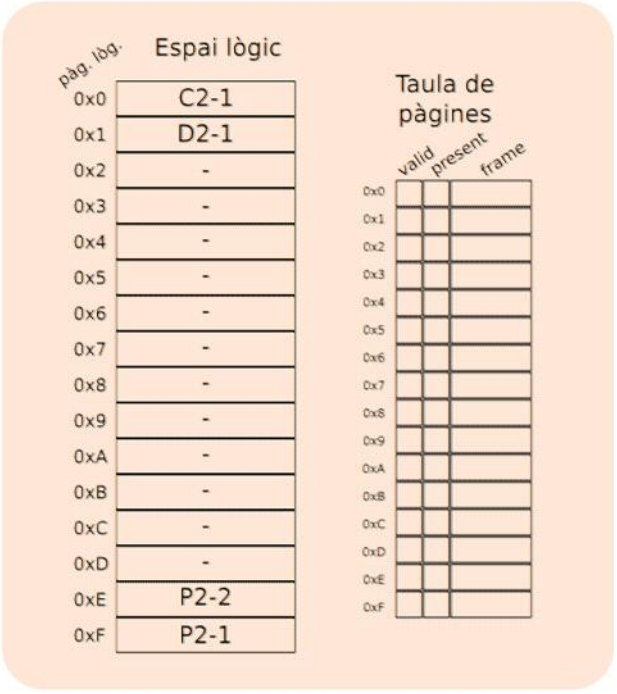
Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	11/01/2020	12:00

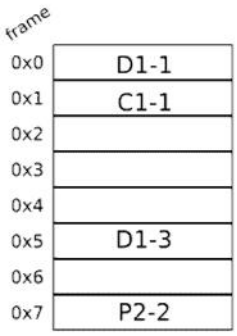
Procés 1

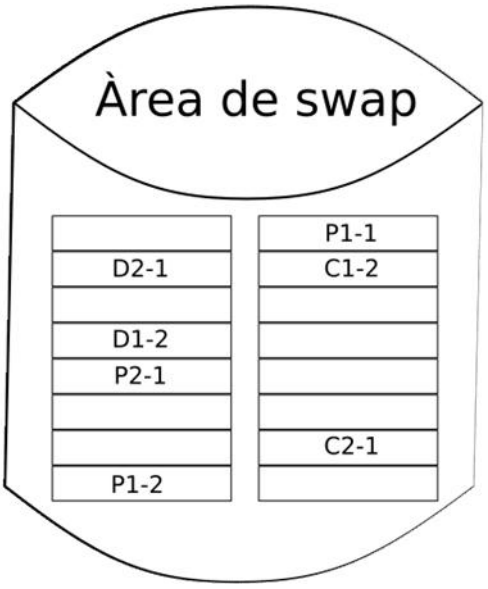


Procés 2



Espai físic



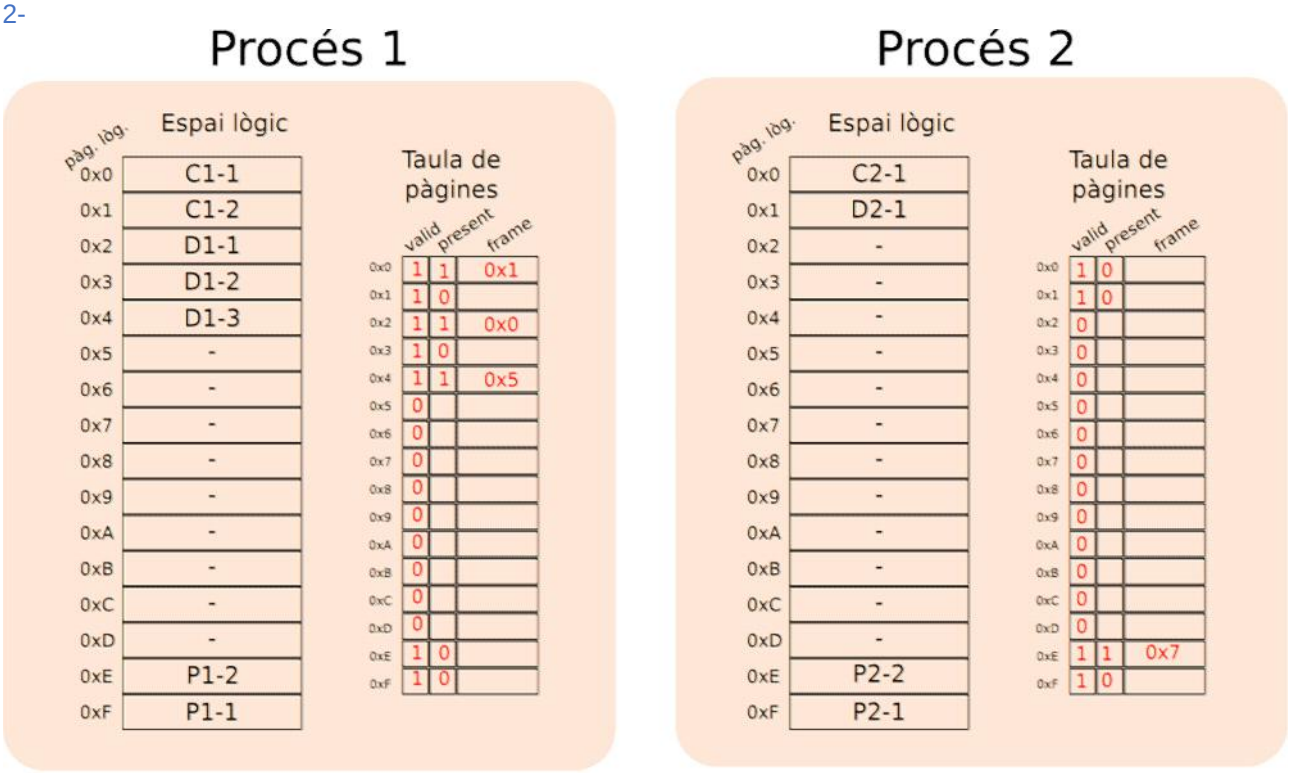


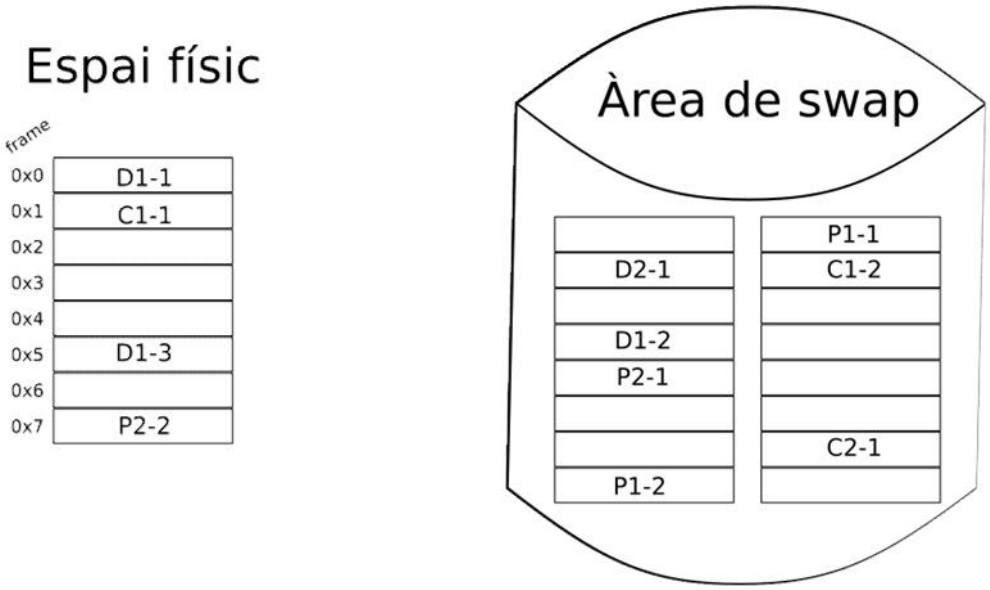
Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	11/01/2020	12:00

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	11/01/2020	12:00

3-

@L	@F (procés 1)	@F (procés 2)
0x4C122	0x5C122	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	11/01/2020	12:00

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[])
{
   char *p = argv[argc-1];

   write(1, p, strlen(p));
   write(1, "\n", 1);

   if (argc > 1) {
      argv[argc-1] = NULL;
      execvp(argv[0], argv);
   }
   exit(0);
}
```

ii)

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

   p = fork();
   pipe(fd);
   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". El procés pare ha d'esperar la mort dels fills i ha de mostrar per la sortida estàndar en quin ordre han acabat (el primer fill creat serà el número 0, el darrer serà el número 4).

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 5 darreres poden aparèixer en qualsevol ordre.

```
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 3 has finished
Child number 0 has finished
Child number 1 has finished
Child number 4 has finished
Child number 2 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.

a-i) El programa escriu per la stdout el darrer argument rebut al vector argv. Si el programa ha rebut més d'un argument, invalida l'últim i el programa es torna a invocar a ell mateix (però amb un argument menys). Per tant, el programa escriu els arguments rebuts en ordre invers. No es crea cap procés.

a-ii) El programa crea un procés fill i tots dos processos creen una pipe ,El pare escriu un caràcter a la seva pipe i espera la mort del fill; el fill intenta llegir de la seva pipe però es bloqueja perquè la pipe és buida i

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	11/01/2020	12:00

existeix un escriptor sobre la pipe (ell mateix). Per tant, els dos processos queden bloquejats i la resta de codi no s'executarà..

b)

```
main()
{
      int i, j, pids[5], p;

      printf("Father starts\n");
      for (i=0; i<5; i++) {
            if ((pids[i] = fork()) == 0)
                  execlp("child", "child", NULL);
            printf("Child number %d has been created\n", i);
      }

      for (i=0; i<5; i++) {
            p = wait(NULL);
            for (j=0; pids[j] != p; j++);
            printf("Child number %d has finished\n", j);
      }
      printf("Father finishes\n");
}
```

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	11/01/2020	12:00

Examen 2019/20-1

Assignatura	Codi	Data	Hora inici
Sistemes operatius	05.566	11/01/2020	12:00

Concurrencia [2.5 puntos]

Suposeu que tenim un aeroport amb una sola pista.

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) Escriviu el codi que descriu la seqüència de l'enlairament i l'aterratge dels avions en aquest aeroport si no hi ha cap mena de prioritat establerta, garantint naturalment que no hi haurà cap coincidència que pugui provocar accidents.

Declaració variables i semàfors

```
SemaphoreSemPista;
```

Inicialització

```
sem_init(SemPista,1);
```

Aterrar ()	**Enlairar ()**
{ sem_wait(SemPista);	{ sem_wait(SemPista);
Aterra;	Enlaira;
sem_signal(SemPista); }	sem_signal(SemPista); }

b) Suposem ara que els avions que aterren tenen prioritat sobre els que s'enlairen, la qual cosa vol dir que mentre hi hagi avions aterrant no se'n pot enlairar cap. A més, no hi pot haver més de 20 avions esperant per aterrar, els avions que arribin superant aquest nombre han de marxar a un altre aeroport. Escriviu el codi que descriu la seqüència de l'enlairament i de l'aterratge dels avions en aquest aeroport.

Declaració variables i semàfors

```
Semaphore SemPista, SemAterra;
Int NAterratges=0;
```

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	11/01/2020	12:00

Inicialització

```
sem_init(SemPista,1);
sem_init(SemAterra,1);
sem_init(SemMutex,1);
```

Aterrar()

```
{
    sem_wait(SemMutex);

    /* Si hi ha 20 avions esperant aterrar, em vaig a un
       altre aeroport */
    if (NAterratges ==20 ) {
          sem_signal(SemMutex);
          return;
    }
    NAterratges++;

    / * Si hi ha avions enlairant-se, esperar. Si no hi ha
       avions enlairant-se, bloquejar enlairament i aterrar * /
    if ( NAterratges == 1 )
          sem_wait( SemPista );
    sem_signal(SemMutex);

    /* Només un avió pot aterrar al mateix temps */
    sem_wait(SemAterra);

    Aterra;

    sem_wait( SemAterra );

    /* Quan tots els avions han aterrat, desbloquegem
       l'enlairament */
    sem_wait(SemMutex);
    NAterratges--;
    if(NAterratges== 0 )
          sem_signal( SemPista );
    sem_signal(SemMutex);
}
```

Enlairar()

```
{

    /* Només un avió pot aterrar al
       mateix temps */
    sem_wait(SemPista);

    Enlaira;

    sem_signal(SemPista);

}
```

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	11/01/2020	12:00

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	11/01/2020	12:00

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	11/01/2020	12:00

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	11/01/2020	12:00

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	11/01/2020	12:00

Examen 2019/20-1

Assignatura	**Codi**	**Data**	**Hora inici**
Sistemes operatius	05.566	11/01/2020	12:00