

Examen 2018/19-1

| Assignatura | Codi | Data | Hora inici |
|--------------------|--------|------------|------------|
| Sistemes operatius | 05.566 | 09/01/2019 | 12:00 |

05.566 09 01 19 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/DIN-A4 amb anotacions per les dues cares 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 2018/19-1

| Assignatura | Codi | Data | Hora inici |
|--------------------|--------|------------|------------|
| Sistemes operatius | 05.566 | 09/01/2019 | 12:00 |

Enunciats

Examen 2018/19-1

| Assignatura | Codi | Data | Hora inici |
|--------------------|--------|------------|------------|
| Sistemes operatius | 05.566 | 09/01/2019 | 12:00 |

1. Teoria [2.5 punts]

Contesteu **justificadament** les següents preguntes:

- a) Indiqueu les transicions d'estat que pot realitzar un procés que està en l'estat de *Run*. Indiqueu també les transicions que no pot realitzar. Justifiqueu perquè no es poden produir aquestes transicions.

Un procés que està en l'estat de *Run* pot passar als següents estats:

- A l'estat de *End/Finalitzat* si realitza un *exit()* o generar un error (excepció).
- A l'estat de *Wait/Espera* si realitza un operació que impliqui el seu bloqueig (entrada/sortida).
- A l'estat de *Ready/Preparat* si el planificador li treu la CPU (per que se li esgota el quatum o ha arribat un altre procés més prioritari).

L'única transició que no pot fer és la de tornar a l'estat de nou, a causa de que ja està arrencat.

- b) En un sistema de gestió de memòria basat en paginació sota demanda, seria possible que, en el mateix moment, una mateixa direcció lògica pugui estar present en memòria física per a un procés però no present per a un altre procés?

Si que és possible. Els processos tenen diferents espais de memòria lògica. Per tant, una mateixa direcció lògica en diferents processos no té perquè d'estar mapejada a la mateixa direcció física i per tant, pot ser que aquesta direcció estigui present en un procés i no estigui present en un altre.

- c) Què és un enllaç simbòlic (*soft link*)? Un enllaç simbòlic podria generar incoherències dins el sistema d'arxius? En cas afirmatiu, indiqueu com.

Un enllaç simbòlic és un enllaç que permet assignar de forma indirecta diferents noms/rutes a un objecte del sistema d'arxius. Això es realitza indirectament especificant la ruta a l'objecte a la qual apunta.

Els enllaços simbòlics poden generar incoherències en el sistema d'arxius si s'elimina l'objecte original al qual estaven enllaçant. A partir d'aquest moment la resta de noms (enllaços simbòlics) al objecte original esborrat no podran accedir al seu contingut, generant-se un error.

- d) En què consisteix un procés *zombie*? Quines condicions s'han de produir per a la seva aparició?

Un procés zombie és un procés que ha finalitzat però que encara no ha realitzat la sincronització de finalització amb el seu procés pare (*exit-wait*).

Les condicions que s'han de produir és que el procés fill finalitzi i el procés pare encara no hagi realitzat el *wait* o que directament hagi finalitzat sense fer-ho.

Examen 2018/19-1

| Assignatura | Codi | Data | Hora inici |
|--------------------|--------|------------|------------|
| Sistemes operatius | 05.566 | 09/01/2019 | 12:00 |

2. Memòria [2,5 punts]

Sigui un sistema de gestió de memòria basat en paginació sota demanda on les pàgines tenen una mida de 4KBytes, les adreces lògiques són de 16 bits i l'espai físic és de 32 KBytes.

Sobre aquest sistema es creen dos processos.

- Procés 1: el seu fitxer executable determina que l'àrea de codi ocupa dues pàgines, que les dades inicialitzades n'ocupen una, les no inicialitzades una i la pila dues.
- Procés 2: el seu fitxer executable determina que l'àrea de codi ocupa una pàgina, que les dades inicialitzades n'ocupen una, les no inicialitzades cap i la pila una.

Es demana:

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

El fitxer executable conté el codi i el valor inicial de les dades inicialitzades. En aquest cas, tenim dues pàgines de codi i una de dades inicialitzades per tant, aproximadament, l'executable ocuparà la mida corresponent a tres pàgines, és a dir, uns 12KB.

Per a una estimació més precisa caldria afegir la mida de les capçaleres de l'executable i caldria descomptar la fragmentació interna que pugui existir a la darrera pàgina de codi i de dades inicialitzades.

- 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).
- Suposant que el procés en execució és el procés 1, indiqueu quines seran les adreces físiques corresponents a les següents adreces lògiques: 0xE9AB i 0xF452. Variaria la resposta si el procés en execució fos el procés 2? En cas afirmatiu, indiqueu el motiu i com canviaria.

Primer cal descompondre les adreces lògiques en identificador de pàgina i desplaçament.

Com la mida de pàgina és de 4KB (2^{12} bytes), calen 12 bits per a codificar el desplaçament dins de la pàgina, és a dir, tres dígit hexadecimals. La resta de bits (4) seran l'identificador de pàgina. Per tant, el primer dígit hexadecimal ens indica l'identificador de pàgina lògica i la resta de dígit indiquen el desplaçament dins la pàgina.

Les traduccions serien:

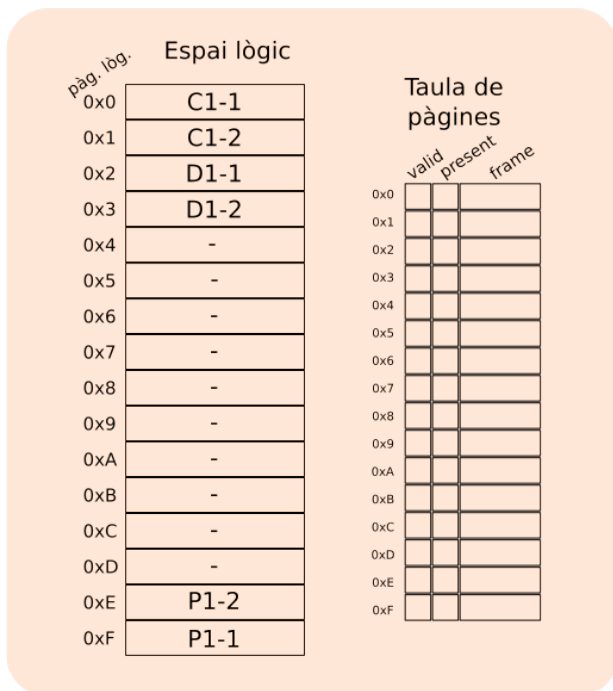
| @L | @F Procés 1 | @F Procés 2 |
|--------|-------------|-----------------------------|
| 0xE9AB | 0x39AB | Excepció adreça invàlida |
| 0xF452 | 0x7452 | Excepció: fallada de pàgina |

Són diferents a cada procés perquè cada procés té una taula de pàgines pròpia.

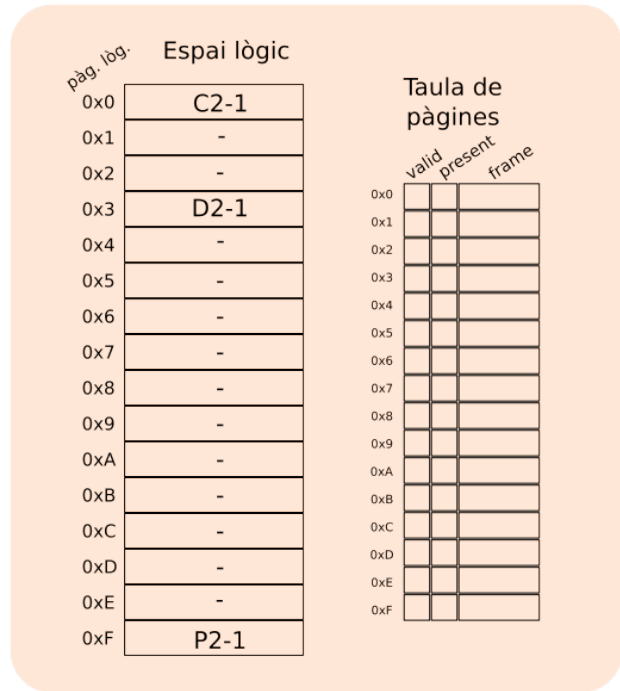
Examen 2018/19-1

| Assignatura | Codi | Data | Hora inici |
|--------------------|--------|------------|------------|
| Sistemes operatius | 05.566 | 09/01/2019 | 12:00 |

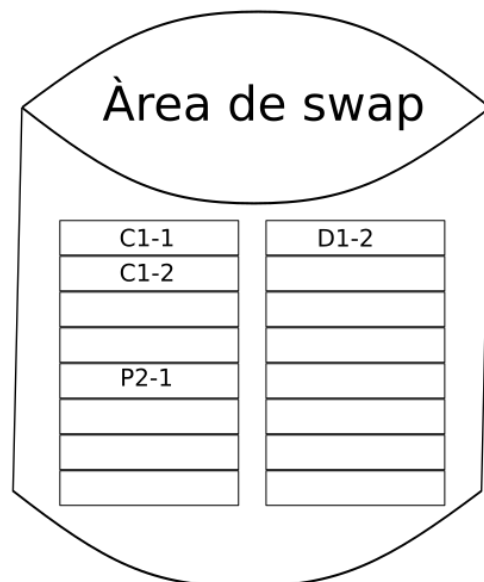
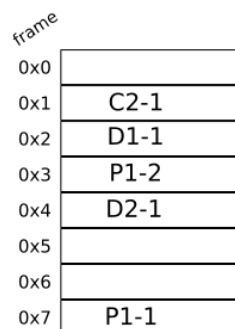
Procés 1



Procés 2



Espai físic



Examen 2018/19-1

| Assignatura | Codi | Data | Hora inici |
|--------------------|--------|------------|------------|
| Sistemes operatius | 05.566 | 09/01/2019 | 12:00 |

Procés 1

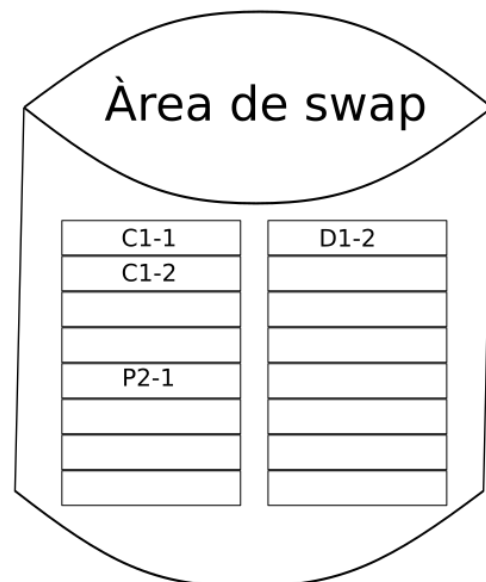
| pàg. lòg. | Espai lògic | Taula de pàgines | | |
|-----------|-------------|------------------|---------|-------|
| | | valid | present | frame |
| 0x0 | C1-1 | 1 | 0 | |
| 0x1 | C1-2 | 1 | 0 | |
| 0x2 | D1-1 | 1 | 1 | 0x2 |
| 0x3 | D1-2 | 1 | 0 | |
| 0x4 | - | 0 | | |
| 0x5 | - | 0 | | |
| 0x6 | - | 0 | | |
| 0x7 | - | 0 | | |
| 0x8 | - | 0 | | |
| 0x9 | - | 0 | | |
| 0xA | - | 0 | | |
| 0xB | - | 0 | | |
| 0xC | - | 0 | | |
| 0xD | - | 0 | | |
| 0xE | P1-2 | 1 | 1 | 0x3 |
| 0xF | P1-1 | 1 | 1 | 0x7 |

Procés 2

| pàg. lòg. | Espai lògic | Taula de pàgines | | |
|-----------|-------------|------------------|---------|-------|
| | | valid | present | frame |
| 0x0 | C2-1 | 1 | 1 | 0x1 |
| 0x1 | - | 0 | | |
| 0x2 | - | 0 | | |
| 0x3 | D2-1 | 1 | 1 | 0x4 |
| 0x4 | - | 0 | | |
| 0x5 | - | 0 | | |
| 0x6 | - | 0 | | |
| 0x7 | - | 0 | | |
| 0x8 | - | 0 | | |
| 0x9 | - | 0 | | |
| 0xA | - | 0 | | |
| 0xB | - | 0 | | |
| 0xC | - | 0 | | |
| 0xD | - | 0 | | |
| 0xE | - | 0 | | |
| 0xF | P2-1 | 1 | 0 | |

Espai físic

| frame | |
|-------|------|
| 0x0 | |
| 0x1 | C2-1 |
| 0x2 | D1-1 |
| 0x3 | P1-2 |
| 0x4 | D2-1 |
| 0x5 | |
| 0x6 | |
| 0x7 | P1-1 |



Examen 2018/19-1

| Assignatura | Codi | Data | Hora inici |
|--------------------|--------|------------|------------|
| Sistemes operatius | 05.566 | 09/01/2019 | 12:00 |

3. Processos [2,5 punts]

- a) Indiqueu quin és el comportament dels següents programes (jerarquia de processos creada, missatges escrits per sortida estàndard/pipes/fitxers i en quin ordre,...). Podeu assumir que cap crida al sistema retornarà error.

| | |
|---|--|
| <pre> /* A */ main() { int i; write(1, "Hello\n", 6); for(i=0; i<3; i++) fork(); write(1, "Bye\n", 4); exit(0); } </pre> | <pre> /* B */ main() { int p[2], c; write(1, "F", 1); pipe(p); j = fork(); if (j == 0) write(1, write(p[1], </pre> |
| <pre> /* C */ main() </pre> | <pre> } else { </pre> |

- a) El procés inicial escriu "Hello\n". A continuació, el bucle crea set nous processos (tres –F1, F2 i F3- són fills del procés inicial, dos –F11 i F12- són fills de F1, un –F21- és fill de F2 i un –F111- és fill de F11). Finalment, tots 8 processos escriuen "Bye\n".
- b) El procés inicial escriu "Hello\n". A continuació crea una pipe i un procés fill. El procés fill escriu "Bye1" per la sortida estàndard, escriu "abc" a la pipe i mor. El procés pare entra en un bucle de lectura de la pipe. Les tres primeres iteracions es sincronitzen amb l'escriptura del procés fill amb el que s'escriu tres vegades per la sortida estàndard el missatge "Looping\n". A la quarta iteració el procés pare queda bloquejat llegint de la pipe perquè la pipe té oberta un canal per escriptura (el del propi procés inicial). Per tant, el procés fill queda en estat zombie..
- c) El procés carrega l'executable "ls" i l'executa amb el que es mostra per la sortida estàndard la llista de fitxers del directori actual. El write posterior mai arriba a executar-se perquè, assumint que la crida `execvp` ha funcionat correctament, el codi original del procés ha estat substituït pel codi de l'executable `ls`.

Examen 2018/19-1

| Assignatura | Codi | Data | Hora inici |
|--------------------|--------|------------|------------|
| Sistemes operatius | 05.566 | 09/01/2019 | 12:00 |

b) Escriviu un programa que creï tres processos fills F1, F2 i F3 on:

- F1 passarà a executar el programa `prog1` havent-li redireccionat l'entrada estàndard al fitxer `file_in.txt` i la sortida estàndard al fitxer `file_out1.txt`
- F2 passarà a executar el programa `prog2` havent-li redireccionat l'entrada estàndard al fitxer `file_in.txt` i la sortida estàndard al fitxer `file_out2.txt`
- F1 i F2 s'executaran concurrentment.
- Si el procés F1 acaba abans que el procés F2, F3 executarà el programa `prog3` havent-li redireccionat l'entrada estàndard al fitxer `file_out1.txt`.
- Altrament, si F2 acaba abans que F1, F3 executarà el programa `prog3` havent-li redireccionat l'entrada estàndard al fitxer `file_out2.txt`.

Cal que indiqueu tots els paràmetres a les crides al sistema i que tracteu el valor de retorn coherentment. No cal indicar els includes ni fer el tractament d'errors a les crides al sistema. Podeu assumir que els fitxers d'entrada ja existeixen i que els de sortida no existeixen.

```
main()
{
    int F1, F2, first;

    F1 = fork();
    if (F1 == 0) {
        close(0);
        open("file_in.txt", O_RDONLY);
        close(1);
        open("file_out1.txt", O_WRONLY);
        execl("prog1", "prog1", NULL);
    }

    F2 = fork();
    if (F2 == 0) {
        close(0);
        open("file_in.txt", O_RDONLY);
        close(1);
        open("file_out2.txt", O_WRONLY);
        execl("prog2", "prog2", NULL);
    }

    first = wait(NULL);
    if (fork() == 0) {
        close(0);
        if (first == F1) open("file_out1.txt", O_RDONLY);
        else open("file_out2.txt", O_RDONLY);
        execl("prog3", "prog3", NULL);
    }
    wait(NULL);
    wait(NULL);
}
```


Examen 2018/19-1

| Assignatura | Codi | Data | Hora inici |
|--------------------|--------|------------|------------|
| Sistemes operatius | 05.566 | 09/01/2019 | 12:00 |

4. Concurrencia [2.5 punts]

Tenim una aplicació concurrent formada per N fils d'execució. Cada un d'aquests fils d'execució rep com a paràmetre un identificador que identifica l'ordre lògic en què s'ha creat el procés (des d'1 fins a N). De moment, els fils, únicament imprimeixen el seu identificador per pantalla:

```
Thread(int id)
{
    char msg;

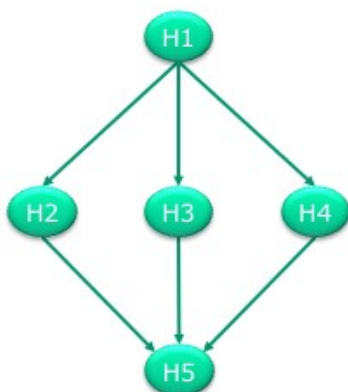
    sprintf(msg, "Thread %d.\n", id);
    write(1, msg, strlen(msg));
}
```

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 decremента de forma atòmica.
- `sem_wait_mult(semaphore s, int n)`. Demana n instàncies del semàfor s . Espera que el valor del semàfor sigui més gran que $n-1$ i quan ho és el decremента en n de forma atòmica.
- `sem_signal(semaphore s)`. S'incrementa de forma atòmica el valor del semàfor.
- `sem_signal_mult(semaphore s, int n)`. Augmenta de forma atòmica el valor del semàfor en n .

Es demana:

- a) Assumint que es creen 5 fils, garantir amb semàfors que els missatges dels fils s'imprimeix en funció del següent graf de precedència:



És a dir, fins que el fil 1 no imprimeixi el seu missatge, els fils 2, 3 i 4 no poden imprimir els seus (en qualsevol ordre). Finalment el fil 5 no pot imprimir el seu missatge fins que els fils 2, 3 i 4 han imprès el seu. Només cal escriure el codi dels fils 1, 2 i 5.

Examen 2018/19-1

| Assignatura | Codi | Data | Hora inici |
|--------------------|--------|------------|------------|
| Sistemes operatius | 05.566 | 09/01/2019 | 12:00 |

```
Semaphore SemH1, SemH5;

sem_init(&SemH1,0);
sem_init(&SemH5,0);
```

```
Thread1(int id)
{
    char msg;

    sprintf(msg,"Thread %d.\n",id);
    write(1,msg,strlen(msg));

    sem_signal_mult(&SemH1,3);
}
```

```
Thread2(int id)
{
    char msg;

    sem_wait(&SemH1);

    sprintf(msg,"Thread %d.\n",id);
    write(1,msg,strlen(msg));

    sem_signal(&SemH5);
}
```

```
Thread5(int id)
{
    char msg;

    sem_wait_mult(&SemH5,3);

    sprintf(msg,"Thread %d.\n",id);
    write(1,msg,strlen(msg));

}
```

b) Assumiu que els fils parells volen llegir un fitxer i que els fils senars volen escriure'l. Implementeu el codi que han d'utilitzar els dos tipus de fils (ThreadParell, Thread Senar) per garantir les següents condicions:

- Podem tenir múltiples fils parells llegint el fitxer de forma simultània
- Quan un fil imparell està escrivint el fitxer no pot haver un altre fil treballant amb el fitxer (ni parell ni senar).

```
Semaphore SemMutex, SemWriter;
Shared intreaders=0;

sem_init(&SemMutex, 1);
sem_init(&SemWriter, 1);
```

Examen 2018/19-1

| Assignatura | Codi | Data | Hora inici |
|--------------------|--------|------------|------------|
| Sistemes operatius | 05.566 | 09/01/2019 | 12:00 |

```
ThreadPar(int id) // Reader
{
    sem_wait(SemMutex);
    readers++;
    if (readers==1)
        sem_wait(SemWrite);
    sem_signal(SemMutex);

    readfile();

    sem_wait(SemMutex);
    readers--;
    if (readers==0)
        sem_signal(SemWrite);
    sem_signal(SemMutex);
}
```

```
ThreadImpar(int id) // Writer.
{
    sem_wait(SemWrite);
    write_file();
    sem_signal(SemWrite);
}
```