Grau en Enginyeria Informàtica (GEI)

Sistemes Operatius

2024-2025 (1er Semestre)

Examen Segon Parcial Gener 2025

Solució

Jordi Mateo jordi.mateo@udl.cat

— Informació important —

Temps: 2 h

- Aquest examen avalua els coneixements teòrics i pràctics de l'assignatura.
- Només es permet l'ús d'una fulla d'apunts en format A4; preparada prèviament per l'estudiant. Altres materials de consulta no estan permesos.
- L'examen s'ha de respondre directament en el mateix document proporcionat i amb bolígraf; les respostes escrites amb llapis no seran vàlides.
- Qualsevol sospita de plagi o còpia comportarà sancions d'acord amb la normativa acadèmica.

Happy coding!!

— Omplir per l'estudiant —

Nom: .....

Cognoms: .....

DNI: .....

- Declaro que jo he entès les instruccions.

Signatura: .....

— Omplir pel professor —

Exercisis	1	2	3	4	5	6	
Punts	2	1	2	2	1	2	10
Correcció							

1: Part A: V/F — 2 Punts

Indica si les següents afirmacions són certes o falses. No requereix justificar les respostes. Cada encert suma **0,2 punts**. Cada error resta **0,1 punts**. La resposta en blanc no suma ni resta punts. Si consideres que has de justificar alguna resposta, fes-ho.

V	F	
■	□	… Les taules de pàgines tenen un avantatge important sobre les taules de segmentació simples (és a dir, utilitzant base i límit) per a la traducció d'adreces virtuals, ja que eliminen la fragmentació externa en l'assignador de memòria física.
□	■	… En un sistema que utilitza planificació Round Robin, augmentar el quantum de temps **Q** sempre augmenta el **Throughput** indefinidament, independentment del cost del canvi de context.
□	■	… En un sistema que utilitza la planificació **Shortest-Job-First (SJF)** la justícia (*fairness*) del sistema sempre està garantida, ja que s'executen primer els processos més curts.
□	■	… Quan un procés sol · licita al nucli recursos que no es poden concedir sense risc de bloqueig (segons l'algoritme del banquer), el nucli ha d'enviar un senyal **SIGKILL** al procés sol · licitant per evitar el bloqueig.
□	■	… L'algoritme de reemplaçament de pàgines LRU (Least Recently Used) sempre ofereix un rendiment equivalent o millor que **FIFO (First-In, First-Out)**, independentment del patró d'accés a la memòria.
■	□	… Suposem que dos programes d'escacs s'estan executant l'un contra l'altre en la mateixa CPU, i cada programa té un límit de temps real per al temps total que passen prenent decisions. Realitzar moltes operacions d'E/S innecessàries pot fer que un dels programes sigui considerat interactiu pel planificador, aconseguint així més prioritat i més temps efectiu de CPU.
■	□	… L'algorisme **Short Remaining Time (SRT)** esmorteix l'efecte de *convoy*.
□	■	… Amb l'algoritme **FIFO**, augmentar el nombre de marcs de pàgina disponibles sempre redueix el nombre de fallades de pàgina.
■	□	… En un sistema amb **hiperpaginació**, el rendiment de les aplicacions que no requereixen accés freqüent a la memòria es veu poc afectat.
■	□	… Assumeix un sistema amb paginació de 2 nivells (*ambdós implementats a memòria*), on l'adreça virtual es compon de la entrada a la pàgina de nivell 1, la entrada a la pàgina de nivell 2, i el desplaçament dins de la pàgina. Quan es produeix una fallada de pàgina en una pàgina en mode de lectura, són necessaris 5 accessos a memòria per gestionar la fallada.

****valor 0.2****

Solution:

2: Part B: Planificació de processos — 1 Punt

Realitza la planificació dels següents processos utilitzant la planificació **SJF**. Assumeix una únic processador i el següent ordre de prioritats *C,B,A* (on C és la prioritat més elevada). Mostra el diagrama de Gantt i calcula les mètriques de planificació.

Procés	Temps arribada	Ràfegues
A	0	$4_{CPU}, 2_{E/S}, 3_{CPU}, 2_{E/S}, 1_{CPU}$
B	1	$1_{CPU}, 4_{E/S}, 1_{CPU}$
C	0	$5_{CPU}, 2_{E/S}, 2_{CPU}, 2_{E/S}, 2_{CPU}$

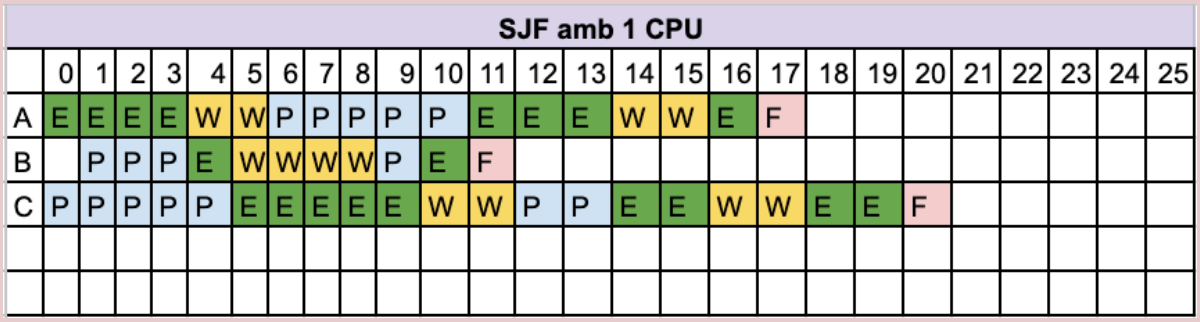
SJF amb *1 processadors*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
A																										
B																										
C																										

Mètriques de planificació

	SJF
%CPU	
Productivitat	
Temps Espera	
A	
B	
C	
D	
Temps Resposta	
A	
B	
C	
D	
Temps Retorn Normalitzat	
A	
B	
C	
D	

Solution:



- %CPU = 95%
- Productivitat = 0.14
- Temps Espera:
  - A = 5
  - B = 4
  - C = 7
- Temps Resposta:
  - A = 0
  - B = 3
  - C = 5
- Temps Retorn Normalitzat:
  - A = 1.42
  - B = 1.67
  - C = 1.62

3: Part C: Interbloqueig — 2 Punts

- Suposem un sistema amb un únic tipus de recurs **R1** que disposa de *U unitats disponibles*. En aquest sistema s'estan executant diversos processos, cadascun dels quals pot requerir fins a *K unitats d'aquest recurs*. Quin és el nombre màxim de processos **P** que poden existir en el sistema assegurant que no es produeixi cap situació de bloqueig (**deadlock**)? **(1 punts)**

Solution:
- U = P*(K-1) + 1
- P = (U-1)/(K-1)
- Com la divisió pot no ser exacta, el resultat ha de truncar-se.

- Donat un sistema amb 4 processos i 5 recursos diferents, on cada recurs té 5 unitats disponibles. A més, cada procés necessita els següents recursos:

Procés	R1 (Necessari / Màxim)	R2 (Necessari / Màxim)	R3 (Necessari / Màxim)	R4 (Necessari / Màxim)	R5 (Necessari / Màxim)
P1	1 / 2	3 / 4	0 / 2	0 / 0	1 / 1
P2	1 / 2	2 / 4	2 / 2	1 / 2	2 / 2
P3	0 / 2	1 / 1	1 / 2	0 / 0	2 / 3
P4	2 / 2	0 / 1	2 / 2	0 / 1	2 / 4

Determina si el sistema es troba en un estat segur. En cas afirmatiu, indica l'ordre d'execució dels processos. En cas negatiu, indica quins processos podrien provocar un interbloqueig. **(1 punt)**

Solution:
El primer pas és construir la taula d'assignació que s'obté restant els recursos necessaris als recursos màxims:

- P1: (1,1,2,0,0)
- P2: (1,2,0,1,0)
- P3: (2,0,1,0,1)
- P4: (0,1,0,1,2)

El segon pas és construir la taula de recursos disponibles que s'obté restant els recursos assignats als recursos inicials:

- Disponibles: (1,1,2,3,2)

A continuació, s'executa l'algoritme del banquer per determinar si el sistema es troba en un estat segur:

- P1 no es pot executar per que necessitar 3 unitats de R2 i només en queden 1.
- P2 no es pot executar per que necessitar 2 unitats de R2 i només en queden 1.
- P3 necessita (0,1,1,0,2) per tant es pot executar. Després de la seva execució, els recursos disponibles seran (3,1,3,3,3).
- P4 necessita (2,0,2,0,2) per tant es pot executar. Després de la seva execució, els recursos disponibles seran (3,2,3,4,5).
- P2 necessita (1,2,0,1,0) per tant es pot executar. Després de la seva execució, els recursos disponibles seran (4,4,3,5,5).
- P1 necessita (1,3,0,0,1) per tant es pot executar. Després de la seva execució, els recursos disponibles seran (5,5,5,5,5).

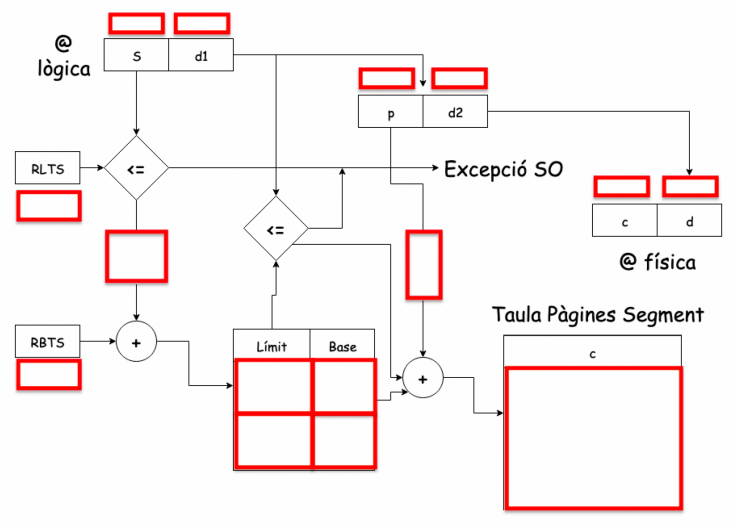
4: Part D: Memòria — 2 Punts

Disposem d'un sistema de gestió de Memòria del tipus segmentació paginada. La mida d'una paraula és igual a 1Byte. Mida pàgina = Mida cel · la = 512 paraules. La mida de MP és 8MBytes. La mida màxima d'un segment és 16KBytes. Un procés té de mida 2MBytes com a molt. En MP tenim carregats els dos processos següent:

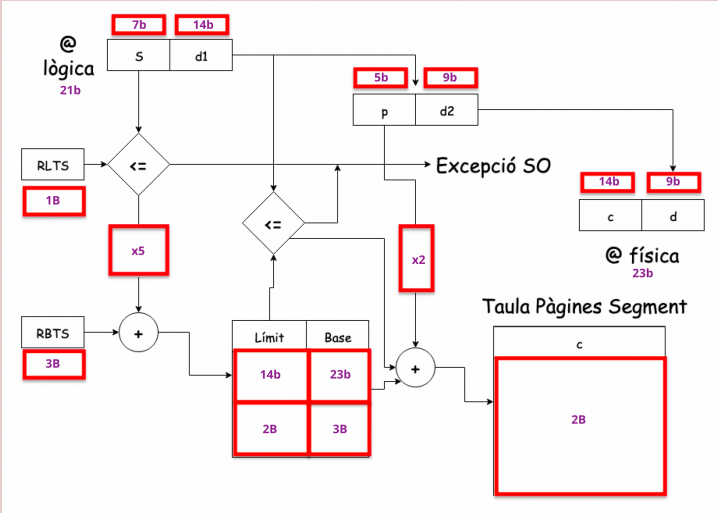
	P1	P2
Procediment 1 (512 Bytes)		X
Procediment 2 (2500 Bytes)	X	
Dades (4000 Bytes)	X	X

Les **X's** en la taula indiquen quins procediments i dades estan assignats a cada procés. Una fila amb dos X's indica que el procediment o dades estan compartits. A més, cada procés té una pila de 3KBytes.

- Ompliu l'esquema de traducció d'adreces (lògiques a físiques). **(0.75 punts)**



Solution:



- Doneu un exemple de la informació que ha de guardar el sistema operatiu per gestionar la memòria dels processos Procés1 i Procés2. Suposeu també que tota aquesta informació s'ha de posar a la cel · la 6 i ha d'ocupar el mínim espai possible. Podeu suposar que totes les cel · les posteriors a la cel · la 6 estan disponibles. **(0.75 punt)**

Solution:

Cel·la 6				
6*2^9 + 0	RLTS	2	1B	
6*2^9 + 1	RBTS	6*2^9 + 4	3B	
6*2^9 + 4	2500B (L)	6*2^9 + 38 (B)	5B	T.S Codi P1
6*2^9 + 9	4000B (L)	6*2^9 + 48 (B)	5B	T.S Dades P1
6*2^9 + 14	3000B (L)	6*2^9 + 64 (B)	5B	T.S Stack P1
6*2^9 + 19	RLTS	2	1B	
6*2^9 + 20	RBTS	6*2^9 + 23	3B	
6*2^9 + 23	512B (L)	6*2^9 + 76 (B)	5B	T.S Codi P2
6*2^9 + 28	4000B (L)	6*2^9 + 48 (B)	5B	T.S Dades P2
6*2^9 + 33	3000B (L)	6*2^9 + 78 (B)	5B	T.S Stack P2
6*2^9 + 38	7		2B	Procediment 2
6*2^9 + 40	8		2B	
6*2^9 + 42	9		2B	
6*2^9 + 44	10		2B	
6*2^9 + 46	11		2B	
6*2^9 + 48	12		2B	Dades
6*2^9 + 50	13		2B	
6*2^9 + 52	14		2B	
6*2^9 + 54	15		2B	
6*2^9 + 56	16		2B	
6*2^9 + 58	17		2B	
6*2^9 + 60	18		2B	
6*2^9 + 62	19		2B	
6*2^9 + 64	20		2B	Stack P1
6*2^9 + 66	21		2B	
6*2^9 + 68	22		2B	
6*2^9 + 70	23		2B	
6*2^9 + 72	24		2B	
6*2^9 + 74	26		2B	
6*2^9 + 76	27		2B	Procediment 1
6*2^9 + 78	28		2B	Stack P2
6*2^9 + 80	29		2B	
6*2^9 + 82	30		2B	
6*2^9 + 84	31		2B	
6*2^9 + 86	32		2B	
6*2^9 + 88	33		2B	

- Suposant que en Mèmoria principal sol tenim carregats els 2 processos i el SO no es té en compte, digueu quina serà la fragmentació interna total. **(0.5 punts)**

Solution:
P1 té 3 segments:
- Procediment 2: 2500 Bytes (5 pàgines)
- Dades: 4000 Bytes (8 pàgines)
- Stack: 3KBytes (6 pàgines)

El primer segment té 5*512-2500=60 Bytes de fragmentació interna. El segon segment 8512-4000=112 Bytes de fragmentació interna. El tercer segment 6*512-3000=312 Bytes de fragmentació interna.
P2 té 3 segments:
- Procediment 1: 512 Bytes (1 pàgina)
- Dades: 4000 Bytes (8 pàgines)
- Stack: 3KBytes (6 pàgines)

El primer segment no té fragmentació interna. El segon segment 8*512-4000=112 Bytes de fragmentació interna. El tercer segment 6512-3000=312 Bytes de fragmentació interna.

5: Part E: Memòria Virtual — 1 Punt

Disposem d'un sistema amb memòria virtual de tipus paginació sota demanda, on la mida de la pàgina és igual a la mida de cel · la: 400 paraules. Cada paraula i enter ocupen 1 Byte. Es fa una assignació local de cel · les igualitària, amb 3 cel · les per procés, i s'utilitza un algorisme de reemplaçament de pàgina basat en LRU (Least Recently Used). Centra't únicament en el codi de la multiplicació de matrius i no consideris els accessos a les variables (N,M,i,j,k,n,m) ni tampoc a la càrrega del codi. Indica quantes falles de pàgina es produiran durant la multiplicació i justifica el càlcul per a cadascuna de les matrius.

```
1  define N 100
2  define M 100
3  int main(){
4      int i, j;
5      int n, m;
6      int A[N][M];
7      int B[N][M];
8      int C[N][M];
9
10     // ...
11
12     // Multiplicació de matrius
13     for(i = 0; i < N; i++){
14         for(j = 0; j < M; j++){
15             for(k = 0; k < N; k++){
16                 C[i][j] += A[i][k] * B[k][j];
17             }
18         }
19     }
20 }
```

Solution:
El total de fallades serà de 250050.

6: Part E: Shell Scripting — 2 Punts

- Completa el següent codi: **0.5 punts**

```bash
#!/bin/bash

declare -A file_types
declare -A file_patterns

file_patterns["regular"]='^.*$'          # Qualsevol fitxer regular
file_patterns["log"]='\.log$'            # Fitxers amb extensió '.log'


while read -r filename; do

    # Comprovar si el fitxer existeix
    if [ ________(1) ]; then
        echo "$filename exists."

        # Comprovar si el fitxer no es regular
        if [ ________(2) ]; then
            file_types["$filename"]="non-regular"
            echo "$filename exists but is not a regular file."

        # Comprovar si el fitxer és un fitxer de log
        elif [[ "$filename" =~ ________(3) ]]; then
            file_types["$filename"]="log"
            echo "$filename matches pattern for log files."
        # Es un fitxer regular
        else
            file_types["$filename"]="regular"
            echo "$filename exists and is a regular file."
        fi

        # Obtenir el nom del fitxer: /path/to/filename.ext -> filename.ext
        # TIP: Elimineu la subcadena més llarga fins al caràcter '/'
        basename="________"(4)
        echo "Base name: $basename"

    else
        echo "$filename does not exist."
    fi
done < "$1"
```

Solution:

```bash
# (1) -e "$filename"
# (2) ! -f "$filename"
# (3) ${file_patterns["log"]}
# (4) ${filename##*/}$
```

- Indica dos formes per executar un script de bash anomenat `script.sh` que es troba al directori actual. **0.2 punts**

Solution:
Per executar el codi es pot fer amb la comanda `bash script.sh` o bé `./script.sh` si s'ha donat permisos d'execució al fitxer (`chmod +x ./script.sh`).

- Analitza el següent codi i respon a les preguntes següents:

```
1  #!/bin/bash
2  # Executa el script amb la següent comanda: ./script.sh
3  str1="operating"
4  str2="system"
5  str3="final"
6
7  str="${str1:3:1}${str2:5:1} -${str1:3:1}${str3:0:1} ."
8
9  # echo "$str"  # (1)
10 # echo `$str`  # (2)
```

a. Explica quin serà el comportament de l'execució de `./script.sh` si descomentes únicament la línia (1). **0.2 punts**

Solution:
La sortida serà la cadena **rm -f .**, ja que la variable str conté aquesta cadena de text. Això es deu a que el codi extreu certs caràcters de les variables str1, str2 i str3 per formar una cadena que s'assigna a str.

b. Indica quin serà el valor de `$?` després de l'execució de `./script.sh` si descomentes la línia (1). **0.2 punts**

Solution:
El valor de `$?` serà 0, ja que l'última comanda (`echo "$str"`) s'executa correctament, mostrant la cadena formada.

c. Explica quin serà el resultat de l'execució de `./script.sh` si descomentes únicament la línia (2), enlloc de la línia (1). **0.2 punts**

Solution:
En aquest cas, la variable str es converteix en una comanda i s'executa com si fos una instrucció. La comanda `rm -f .` intenta eliminar el directori actual. Si l'usuari té permisos per fer-ho, el directori serà eliminat sense mostrar cap missatge. Si no té permisos, es mostrarà un missatge d'error.

d. Indica quin serà el valor de `$?` després de l'execució de `./script.sh` si descomentes únicament la línia (2). **0.2 punts**

Solution:
El valor de `$?` serà 0 si l'usuari té permisos per eliminar el directori actual. En cas contrari, el valor de `$?` serà diferent de 0.

- Assumeix que un usuari utilitza sempre la comanda `ls` amb els arguments `-l` i `-a`. Per tant, decideixes crear un **alias** per a la comanda `ls` que sempre inclogui aquests dos arguments. Escriu un script de Bash que creï aquest alias de manera que estigui disponible cada cop que l'usuari obri una sessió de Bash, sense necessitat de referenciar la ruta completa. El script ha de garantir que l'alias es creï només una vegada i que després de crear-lo, l'alias estigui disponible en la sessió actual. **0.5 punts**

  Notes:

  La comanda `alias` permet crear un alias. Per exemple, `alias ll='ls -l'` crea un alias `ll` per la comanda `ls -l`.

  La comanda `source` permet executar un script de bash en l'entorn actual. Per exemple, `source example.sh` executa el script `example.sh` en l'entorn actual.

  La comanda `grep -q "text" file` retorna 0 si "text" es troba al fitxer file, i 1 en cas contrari.

Solution:

```
#!/bin/bash
cmd="alias ls='ls -l -a'"
if ! grep -q "$cmd" ~/.bashrc; then
    echo "$cmd" >> ~/.bashrc
    source ~/.bashrc
fi
```