Grau en Tècniques d'Interacció Digital i Computació (GTIDIC)

Sistemes Operatius

2024-2025 (1er Semestre)

Examen Demo Gener 2024

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	2	1	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.

V	F	
■	□	… En un sistema amb un algorisme de planificació **Shortest-Job-First (SJF)**, els processos més llargs poden patir inanició si arriben molts processos curts.
□	■	… La política de substitució de pàgines **Last Recently Used (LRU)** garanteix la taxa mínima de fallades de pàgina en qualsevol escenari.
□	■	… La **segmentació** divideix l'espai de memòria en blocs de mida fixa.
■	□	… El planificador per **Loteria (Lottery Scheduling)** assegura que cada procés té una probabilitat no nul · la de rebre temps de CPU si té almenys un bitllet assignat.
□	■	… Si l'algorisme del **banquer** rebutja una petició de recurs, vol dir que el sistema entrarà inevitablement en interbloqueig si aquesta petició s'accepta.
□	■	… La segmentació (*sense paginació*) indueix a fragmentació interna
□	■	… Una pàgina física pertany a un únic proces.
■	□	… Una màquina amb *48-bits* d'adreces virtuals i *32-bits* d'adreces físiques. Si les pàgines són de *8KB*. En una taula de pàgines puc tenir 2^{35} entrades.
□	■	… El nombre de fallades de pàgina en un sistema de memòria virtual es pot reduir afegint més memòria en qualsevol cas.
□	■	… Imagineu una situació amb un procés molt prioritari **(H)** i un procés menys prioritari **(L)** en un sistema Round Robin pot patir inanició el procés **L**.

valor 0.2

Aquesta part no requereix justificació, però si consideres que has de justificar alguna resposta, fes-ho a continuació.

Solution:

1. Cert: L'SJF prioritza els processos més curts. Si els processos curts arriben contínuament, els processos llargs poden no executar-se mai (inanició).
2. Fals: Es poden dissenyar algorismes més eficients que LRU per a certes càrregues de treball.
3. Fals: La segmentació divideix l'espai de memòria en segments de mida variable, depenent de la mida de cada programa o dades.
4. Cert: El planificador per Loteria selecciona processos aleatòriament basant-se en bitllets. Si un procés té almenys un bitllet, té probabilitats no nul · les de ser escollit.
5. Fals: L'algorisme del banquer garanteix que el sistema es manté en un estat segur. Si l'estat no és segur, l'interbloqueig pot o no pot ocórrer.
6. Fals: La segmentació indueix a fragmentació externa, no interna.
7. Fals: Una pàgina física pot ser compartida per diversos processos, per exemple, en memòria compartida.
8. Cert: Si les pàgines són de 8KB i tenim 48-bits d'espai d'adreçament, el nombre de pàgines virtuals és de $\frac{2^{48}}{2^{13}} = 2^{35}$.
9. Fals: Afegir més memòria pot augmentar el nombre de fallades de pàgina en certes situacions, com l'anomalia de Belady.
10. Fals: Amb Round Robin, tots els processos reben un temps de CPU regularment, evitant la inanició.

2: Part B: Planificació de processos — 2 Punts

Realitza la planificació dels següents processos utilitzant els algorismes de planificació **FCFS** amb *2 processadors* i **Round Robin amb Quantum 2** amb *1 processador*. En cas d'empat on 2 processos arribin a la cua de preparats al mateix temps, s'executarà el procés amb prioritat més elevada en aquest cas (*D*,*A*,*B*,*C*).

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

Es demana:

1. Mostrar la planificació dels processos en un diagrama de Gantt. **(1 punt)**
2. Calcular totes les mètriques de planificació per a cada algorisme. **(0.5 punts)**
3. Raona i justifica quin és l'impacte del valor del quantum. **(0.5 punts)**.

FCFS amb *2 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																										
D																										

Round Robin amb Quantum 2 amb *1 processador*

	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																										
D																										

Mètriques de planificació

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

Solution:

FCFS amb 2 CPU

	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		E	E	E	E	E	W	W	E	E	E	E	F														9	2
B				P	P	E	W	E	W	P	P	E	F														3	2
C		P	P	E	E	W	E	E	E	E	W	W	E	F													7	3
D	E	E	E	W	P	P	E	W	W	P	E	F															5	3
	1	2	2	2	2	2	2	2	2	2	2	2	1	0														

Round Robin Q=2 amb 1 CPU

	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	E	W
A		P	E	E	P	P	P	P	E	E	P	P	P	P	E	W	W	P	P	E	E	P	E	E	F		9	2
B				P	P	P	P	E	W	P	P	P	P	E	W	P	P	P	E	F							3	2
C		P	P	P	E	E	W	P	P	P	E	E	P	P	P	E	E	W	W	P	P	E	F				7	3
D	E	E	P	P	P	P	E	W	P	P	P	P	E	W	W	P	P	E	F								5	3
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1				

FCFS 2 CPU		RR 1 CPU	
% ús CPU	92%	% ús CPU	100%
Productivitat	0,31	Productivitat	0,17
Temps Espera	2,25	Temps Espera	11
A	0	A	12
B	4	B	11
C	2	C	11
D	3	D	10
Temps Resposta	1	Temps Resposta	2
A	0	A	1
B	2	B	4
C	2	C	3
D	0	D	0
Temps Retorn Normalitzat	1,34	Temps Retorn Normalitzat	2,41
A	1,00	A	2,09
B	1,80	B	3,20
C	1,20	C	2,10
D	1,38	D	2,25

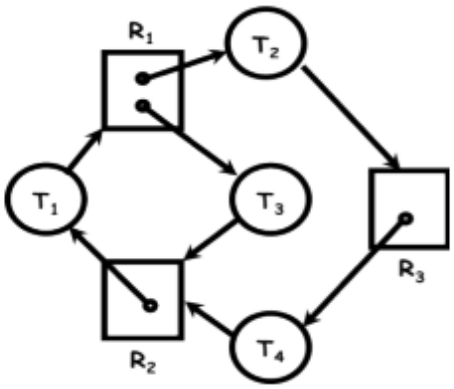
Si el valor del quàntum és molt petit, significa que els processos es commuten amb més freqüència, la qual cosa pot augmentar el cost dels canvis de context. Això es deu al fet que canviar d'un procés a un altre implica certes operacions, com ara desar i restaurar l'estat del procés, que tenen un cost associat.

No obstant això, és important destacar que un valor de quàntum molt gran tampoc és òptim, ja que podria conduir a una manca de resposta en processos interactius. Per tant, la elecció del valor del quàntum ha de ser equilibrada per minimitzar el cost dels canvis de context i garantir una bona experiència d'usuari.

En resum, la elecció del valor del quàntum és un compromís entre minimitzar el cost dels canvis de context i garantir una experiència d'usuari òptima.

3: Part C: Interbloqueig — 1 Punt

- Considera el següent sistema amb 3 recursos (R1,R2,R3) i 4 tasques (T1,T2,T3,T4). On *R1 té dos instàncies*, mentre que els altres dos recursos *(R2,R3) tenen una instància cadascun.* Assumeix:
  - Els recursos no es poden preemtir.
  - Els processos no alliberen voluntàriament els recursos abans de finalitzar.
- Analitza si els processos estan en interbloqueig. En cas afirmatiu, explica com es compleixen les quatre condicions d'interbloqueig. En cas negatiu, dona una possible seqüència en què els processos s'executen. **(0,5 punts)**



Solution:
Els processos estan en interbloqueig. Les quatre condicions d'interbloqueig es compleixen de la següent manera:

1. **Mutual exclusion**: els recursos són exclusius, ja que cada recurs només pot ser utilitzat per un sol procés.
2. **Hold and wait**: Segons l'enunciats els recursos no s'alliberen fins que el procés finalitza.
3. **No preemption**: els recursos no es poden preemtir, per tant, els processos no poden alliberar els recursos abans de finalitzar.
4. **Circular wait**: hi ha un cicle de dependències entre els processos. P1->R1->P3->R2->P1 o P1->R1->P2->R3->P4->R2->P1.

- Assumeix que es permet afegir una instància addicional de R1 o R2. Estan els processos en interbloqueig en aquest cas? En cas afirmatiu, explica com es compleixen les quatre condicions d'interbloqueig. En cas negatiu, dona una possible seqüència en què els processos s'executen. **(0,5 punts)**

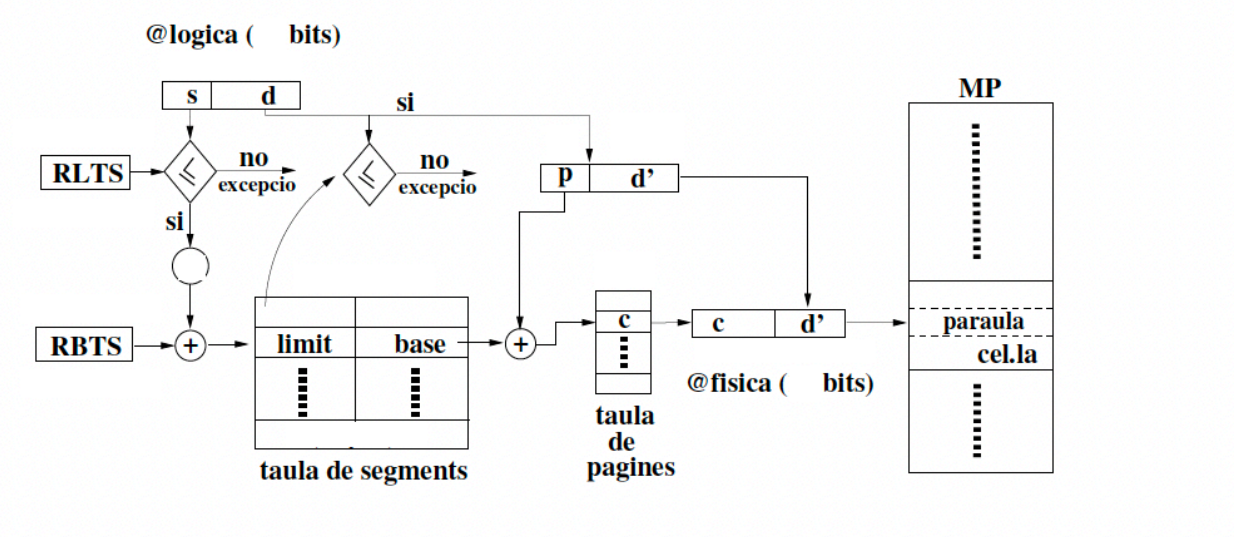
Solution:
No hi ha interbloqueig. Suposem que R2 té una instància addicional.

- P3 adquireix la instància addicional de R2 i finalitza, alliberant les instàncies de R1 i R2.
- P1 adquireix una instància de R1 i finalitza, alliberant les instàncies de R1 i R2.
- P4 adquireix una instància de R2 i finalitza, alliberant les instàncies de R2 i R3.
- P2 adquireix la instància de R3 i finalitza.

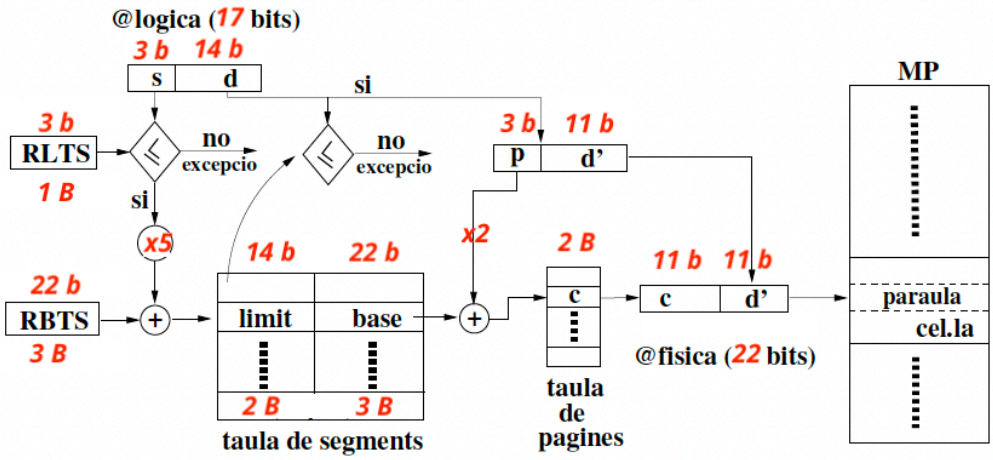
4: Part D: Memòria — 2 Punts

Disposem d'un sistema de gestió de la Memòria principal del tipus segmentació paginada. La mida de Memòria Principal és 4MBytes. Un procés consta de com a molt 8 segments. La mida màxima d'un segment és 16KBytes. La mida d'una pàgina (i una cel · la) és de 2KBytes. La mida d'una paraula és 1Byte. Totes les taules s'implementen en Memòria Principal. Es demana:

- Esquema de traducció d'adreces lògiques a físiques. Amb totes les mides de les estructures (o registres que hi apareguin). Poseu-hi també el(s) multiplicador(s) que cregueu convenient(s). **0.5 punts**



Solution:



- Quina és l'adreça lògica corresponent a l'adreça física 2048 (decimal) d'aquest procés? Su-

posant que les taules de segments i de pàgines d'un procés són (tots els nombres estan en decimal): **0.5 punts**

- Taula de segments:

Base	Limit
1024	4KBytes
1110	6KBytes
1200	2KBytes

- @1024:

Pàgina	Cel · la
0	20
1	40

- @1110:

Pàgina	Cel · la
0	1
1	10
2	12

- @1200:

Pàgina	Cel · la
0	60

Solution:
- @ Lògica: 3 bits (segment) + 3 bits (pàgina) + 11 bits (desplaçament) = 17 bits.
- @ Física: 11 bits (cel.la) + 11 bits (desplaçament) = 22 bits.

Si tenim l'adreça física 2048, aquesta es pot descompondre en 11 bits de cel · la i 11 bits de desplaçament. Per tant, 2048 es pot representar com 10000000000 en binari. Aquesta adreça correspon a la cel.la 1. La cel · la 1 es troba a la taula de pàgines del segment 1, a la pàgina 0. Per tant, l'adreça lògica corresponent és 001 000 00000000000 o sigui $2^{15} queendecimals 32768$.

- Imagina que tenim un sistema amb processos molt grans. Recomanaries aquest sistema de gestió de la memòria? Justifica la teva resposta. **1 punt**

Solution:
Si els processos són molt grans, el sistema de gestió de la memòria actual no és el més adequat, ja que la segmentació paginada no és eficient per a processos grans. La segmentació paginada és més eficient per a processos petits i mitjans. Per a processos grans, seria més eficient utilitzar la paginació pura, ja que permet gestionar millor la memòria i no requereix tantes taules com la segmentació paginada. Un altra opció seria utilitzar la paginació multinivell, que permet gestionar millor la memòria i és més eficient per a processos grans. Finalment, també es podria utilitzar la paginació segmentada.

5: Part E: Memòria Virtual — 1 Punt

Suposeu que tenim un sistema amb paginació sota demanda i paginació (amb la taula implementada en Hardware). El sistema opera amb els següents paràmetres:

- El temps d'accés a memòria principal (MP) és de 2 ms.
- Les pàgines tenen 1.000 paraules cadascuna.
- El dispositiu de paginació és un disc dur amb:
  - Temps de cerca mitjà: 8 ms.
  - Velocitat de gir: 3.000 revolucions per minut (RPM).
  - Velocitat de transferència: 1.000.000 paraules/segon.
- El temps de paginació per hardware i actualitzacio de les taules és menyspreable.

Assumeix que un 10% de les instruccions accedeixen a una pàgina diferent i que el 80% d'aquestes pàgines ja es troben a memòria. Si cal reemplaçar una pàgina, el 60% de les vegades la pàgina a reemplaçar ha estat modificada. Es demana calcular el temps Efecitu d'Accés (TEA).

Solution:
El temps d'accés efectiu es pot calcular com:
TEA = 0.9 x (Temps NO fallada) + 0.1 x [0.2 x (0.6 x (Reemplaçament amb canvis) + 0.4 x (Reemplaçament sense canvis)) + 0.8 x (Temps sense reemplaçament)]
On:

- Temps NO fallada: 2 ms.
- Temps rotació mitjà: $\frac{60}{3000x2} = 10ms$.
- Temps de transferència: $\frac{1000}{1000000} = 1ms$.
- Temps accés al disc: 8 + 10 + 1 = 19 ms.
- Reemplaçament amb canvis: 19 + 19 + 2 = 40 ms.
- Reemplaçament sense canvis: 19 + 2 = 21 ms.
- Temps sense reemplaçament: 2 ms.
- TEA = 2.6 ms.

6: Part E: Shell Scripting — 2 Punts

- Completa el següent codi i indica com s'utilitza. **0.5 punts**

```
#!/bin/bash

declare -A marks

while read -r name mark; do
    marks[$name]=$mark
done < "$1"

for name in ________(1); do
    mark=${marks[$name]}
    if [ ________________________________________(2)]; then
        qual="SUSP"
    elif [ ________________________________________(3)]; then
        qual="APRV"
    elif [ ________________________________________(4)]; then
        qual="NOTB"
    elif [ ________________________________________(5)]; then
        read -p "Is $name's mark excellent (EXEL) or outstanding (EXMH)? "
            user_qual
        if [ "$user_qual" != "EXEL" ] && [ "$user_qual" != "EXMH" ]; then
            echo "Invalid qualification: $user_qual"
            exit 1
        fi
        qual=$user_qual
    else
        echo "Invalid mark for $name: $mark"
        exit 1
    fi
    printf "%s\t%s\t%s\n" "$name" "$mark" "$qual"
done
```

Solution:

```
# (1) ${!marks[@]} retorna els índexs de l'array associatiu marks.
# (2) $mark -lt 50
# (3) [ $mark -ge 50 ] && [ $mark -lt 70 ]
# (4) [ $mark -ge 70 ] && [ $mark -lt 90 ]
# (5) $mark -ge 90

Per executar el script, cal passar-li un fitxer com a argument amb el següent
    format:
nom nota

on nota pot ser un nombre enter entre 0 i 100.
```

- Escriu un script de bash que et permeti crear un programa en bash i que es pugui executar des de qualsevol ubicació del sistema sense necessitat de referenciar la ruta completa i sense utilitzar la comanda bash. **0.5 punts**

Solution:

```
#!/bin/bash

echo "#!/bin/bash\n" > "/usr/local/bin/$1"
chmod +x "/usr/local/bin/$1"
```

- Analitza el següent codi amb C i reescriu en Bash el codi anterior per tal que faci el mateix. **0.5 punts**

```
#include <unistd.h>
int main(void) {
    fork();
    main();
}
```

Solution:

```
#!/bin/bash
f(){
    f &
}
f
```

- Explica quin és l'objectiu del següent codi executat en el context del programa anterior. **0.5 punts**

```
#!/bin/bash
threashold=100
if [ $(pidof $(basename $0) | wc -w) -gt $threashold ]; then
    for pid in $(pidof $(basename $0)); do
        kill -9 $pid
     done
fi
```

Hint:

- `pidof` retorna els PID dels processos amb el mateix nom.
- `basename` retorna el nom del fitxer sense la ruta.
- `wc -w` compta el nombre de paraules.

Solution:
El codi mata tots els processos amb el mateix nom que el programa si hi ha més de 100 instàncies en execució. En el context del programa anterior, el codi mata tots els processos que es creen recursivament i evita la fork bomb.