Grau en Enginyeria Informàtica (GEI)

Sistemes Operatius

2023-2024 (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	
Punts	2	3	2	3	10	20
Correcció						

1: Part A: Interbloqueig — 2 Punts

1. Imagineu un sistema amb 1 únic tipus de recurs (A) que disposa de múltiples unitats (U). En aquest sistema s'executen processos on cada procés pot sol · licitar K unitats del recurs A. Quin és el nombre mínim de processos que s'han d'executar per tal que es pugui produir un interbloqueig? **(1 punts)**

Solution:
P = (U-1)/(K-1)

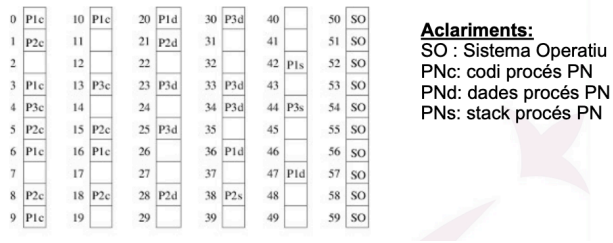
1. Tenim un sistema operatiu que implementa la detecció del deadlock. Hi ha 5 processos $p1, p2, p3, p4, p5$ i 3 tipus de recursos $r1 = 3, r2 = 11, r3 = 12$. Donat l'estat general inicial de Taula respongueu: si el sistema troba en un estat segur i si serviria la petició del procés $Sol \cdot licitud_2(1, 4, 2)$? **(1 punts)**.

	Recurs 1		**Recurs 2**		**Recurs 3**	
	Màxim	*Assignat*	*Màxim*	*Assignat*	*Màxim*	*Assignat*
Procés 1	0	0	1	1	2	1
Procés 2	2	1	8	0	6	0
Procés 3	3	1	6	3	10	6
Procés 4	0	0	6	2	5	4
Procés 5	0	0	7	1	5	0

Solution:
Segons algorisme de seguretat l'estat inicial (taula) és segur. Es possible trobar una seqüència d'execució que no produeixi un interbloqueig. Analitzant la petició per l'algorisme del banquer es pot veure que no es pot concedir la petició.

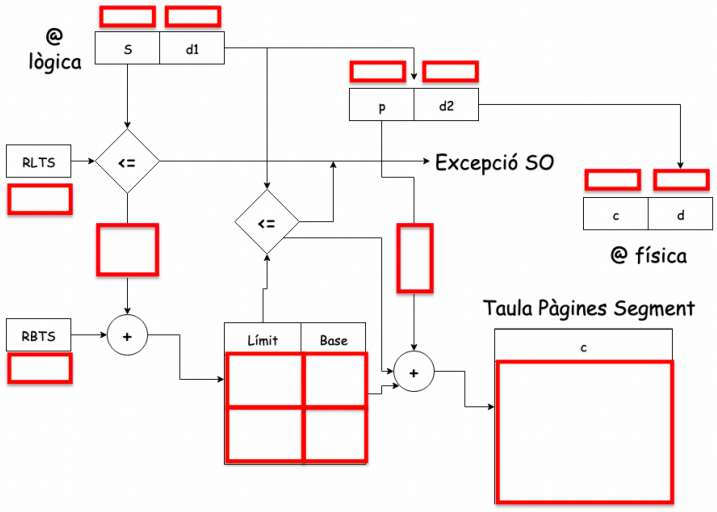
2: Part B: Gestió de Memòria — 3 Punts

1. Disposem d'un sistema de gestió de memòria del tipus segmentació paginada. La mida d'una pàgina és 1Kbyte. La mida d'una paraula és un byte. Un programa consta de com a molt 3 segments. La mida màxima d'un segment és 8Kbytes. Totes les taules H/W s'implementen en memòria principal (MP). A la figura següent es pot veure el contingut de MP:



Aclariments:
SO : Sistema Operatiu
PNc: codi procés PN
PNd: dades procés PN
PNs: stack procés PN

- Ompliu l'esquema de traducció d'adreces (lògiques a físiques) **(0.5 p)**



Solution:
s=2 bits p=3 bits d2=10 bits d1=13 bits c=6 bits RLTS=1B RBTS=2B Limit=2B Base=2B Cel · la=1B

- Com podrien P1 i P2 compartir la cel · la 2 de MP? **(0,5 punts)**

Solution:
Per a compartir el codi/dades cal que estigui en el mateix espai d'adreçes lògiques, per tant,en principi no es poden compartir perquè diu que un procés pot tenir com a màxim 3 segments i per crear una cel · la s'hauria de crear un segment nou.

- Si es crea un nou procés (P4), amb una mida de codi = 5 Kbytes, dades = 5 Kbytes i stack= 1 Kbyte. En cas que aquest procés càpiga en memòria, doneu un exemple de com quedarien les seves taules de segments i de pàgines (el SO les posa en la cel · la 49). **(1 punts)**

Solution:

Cel·la 49			
@	Contingut		
49·1K + 0	RLTS [1B] → 2		
49·1K + 1	RBTS [2B] → 49K+3		
49·1K + 3	L [2B] → 5K	B [2B] → 49K+15	
49·1K + 7	L [2B] → 5K	B [2B] → 49K+20	
49·1K + 11	L [2B] → 1K	B [2B] → 49K+25	
49·1K + 15	C [1B] → 2		TP seg. codi
49·1K + 16	C [1B] → 7		
49·1K + 17	C [1B] → 11		
49·1K + 18	C [1B] → 12		
49·1K + 19	C [1B] → 14		
49·1K + 20	C [1B] → 17		TP seg. dades
49·1K + 21	C [1B] → 19		
49·1K + 22	C [1B] → 22		
49·1K + 23	C [1B] → 24		
49·1K + 24	C [1B] → 26		
49·1K + 25	C [1B] → 27		TP seg. pila

1. En un nucli de Linux de 32 bits, el nucli suporta un nombre màxim de 32768 processos en execució. El kernel utilitza 1073741824 bytes d'espai d'adreces virtual. L'espai d'adreces virtual es divideix equitativament entre tots els processos. En el cas que tots els processos estiguin en execució, quina és la mida de l'espai d'adreces virtual que es designa a cada procés? **(1 punt)**

Solution:
El total d'espai d'adreces virtual és 2^32 bytes = 4294967296. Per tant, si el kernel utilitza 1073741824 bytes, queden 3221225472 bytes per als processos. Per tant, cada procés té 3221225472 / 32768 = 1048576 bytes = 1 MB d'espai d'adreces virtual.

3: Part C: Memòria Virtual — 2 Punts

1. Disposem d'un sistema amb memòria virtual del tipus paginació sota demanada. On la mida de la pàgina = mida de cel · la = 200 paraules. Disposem d'assignació local de cel · les igualitària (3 cel · les per procés). Utilitzem un algorisme de reemplaçament de pàgina basada en LRU. La mida d'una paraula i d'un enter és 1 Byte. *Quantes fallades de pàgina es produiran en el programa següent, dona la formula en funció de* ***m*** *i* ***n****?*

```
1  define m 100
2  define n 100
3  main() {
4      int i =0;
5      int j =0;
6      int **A;
7      A = (int **) calloc (n * sizeof(int *));
8      for (i = 0; i < n; i++) {
9          A[i] = (int *) calloc (m * sizeof(int));
10     }
11     for (i = 0; i < n; i++) {
12         for (j = 0; j < m; j++) {
13             A[i][j] = i + j;
14         }
15     }
16 }
```

Assumeix que les variables i,j,m,n no ocupen *Memòria Principal*. S'implementen en registres de la CPU. En el disc la matriu A està emmagatzemada per files. **(1,5 punts)**

Solution:
En total es produeixen 101 fallades de pàgina.

1. Indica el mapa de memòria del procés les seves variables principals. No cal tenir en compte les crides a funcions de llibreries externes. Assumeix que totes les variables es troben a MP. **(0,5 punts)**

Solution:

```
1  Adreça Baixa
2      +-------------------------+
3      | Codi                    |  El codi del programa
4      +-------------------------+
5      | Dades Inicialitzades    |  n,m
6      +-------------------------+
7      | Dades No Inicialitzades|  res
8      +-------------------------+
9      | Heap                    |  A
10     +-------------------------+
11     | Stack                   |  i,j
12     +-------------------------+
13 Adreça Alta
```

4: Part D: Bash Scripting — 3 Punts

1. Completa el codi següent seguint les especificacions que es donen a continuació. **(1,5 punts)**

```bash
#!/bin/bash

check_password() {
    local password=$1

    # Check if the password has at least 8 characters
    if ___________; then
        # Check if the password has at least one capital letter
        if [[ $password =~ ___________ ]]; then
            # Check if the password has at least one number
            if [[ $password =~ ___________ ]]; then
                # Check if the password has at least one special character
                if [[ $password =~ ___________ ]]; then
                    echo "Password is valid."
                    return 0
                else
                    echo "Password must contain at least one special character."
                fi
            else
                echo "Password must contain at least one number."
            fi
        else
            echo "Password must contain at least one capital letter."
        fi
    else
        echo "Password must be at least 8 characters long."
    fi

    return 1
}

read -sp "Enter your password: " user_password
echo

check_password "$user_password"
```

Solution:

```bash

#!/bin/bash

check_password() {
    local password=$1

    # Check if the password has at least 8 characters
    if [ ${#password} -ge 8 ]; then
        # Check if the password has at least one capital letter
        if [[ $password =~ [A-Z] ]]; then
            # Check if the password has at least one number
            if [[ $password =~ [0-9] ]]; then
                # Check if the password has at least one special character
                if [[ $password =~ [^a-zA-Z0-9] ]]; then
                    echo "Password is valid."
                    return 0
                else
                    echo "Password must contain at least one special character."
                fi
            else
                echo "Password must contain at least one number."
            fi
        else
            echo "Password must contain at least one capital letter."
        fi
    else
        echo "Password must be at least 8 characters long."
    fi

    return 1
}

check_password "$1"
```

2. Analitza el següent codi amb C i respon les preguntes següents:

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

- Reescriu en Bash el codi anterior per tal que faci el mateix. **(0.5 punts)**

Solution:

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

- Indica quin problema pot causar aquest codi. **(0.25 punts)**

Solution:
Aquest script causa un fork bomb, és a dir, genera un nombre infinit de processos. Això pot provocar que el sistema operatiu no pugui crear més processos i que el sistema operatiu es quedi bloquejat.

- Escriu un script en Bash que sigui capaç de detectar aquesta situació i l'aturi.**(0.25 punts)**

Solution:
Puc assumir que un script tingui més de 100 processos identics (fills) en execució es un escenari molt extrany en un sistema operatiu. Per tant, podem assumir que aquest script té un compartament anòmal semblant a una fork bomb.

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

5: Part Pràctica — 10 Punts

Per realitzar aquesta pràctica, heu de seguir les instruccions que trobareu a l'enllaç següent: https://classroom.github.com/a/E41OXCU-. Les preguntes les heu de respondre al README.md del vostre repositori. Un cop finalitzat, heu de fer un commit i un push de tots els canvis al vostre repositori de GitHub.

Analitza el següent codi i respon les preguntes següents:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_LINE 100
#define MAX_NAME 20

typedef struct {
    char name[MAX_NAME];
    float mark;
} student;

int main(int argc, char *argv[]) {
    student *s;
    int n;
    printf("Enter the number of students: ");
    scanf("%d", &n);
    s = (student *) malloc(n * sizeof(student));
    for (int i = 0; i < n; i++) {
        printf("Enter the name of student %d: ", i + 1);
        scanf("%s", s[i].name);
        printf("Enter the mark of student %d: ", i + 1);
        scanf("%f", &s[i].mark);
    }
    FILE *f = fopen("notes.txt", "w");
    fprintf(f, "Name\tMark\n");
    fprintf(f, "----\t----\n");
    for (int i = 0; i < n; i++) {
        fprintf(f, "%s\t%.2f\n", s[i].name, s[i].mark);
    }
    fclose(f);
    free(s);
    return 0;
}
```

- Explica què fa el programa i descriu si pot tenir problemes relacionats amb la memòria. Justifica la resposta amb un exemple. **(1 punt)**

Solution:
Aquest programa demana a l'usuari el nombre d'estudiants i les seves dades (nom i nota), i després imprimeix aquestes dades per pantalla. Pot tenir problemes de memòria si l'usuari introdueix un nombre negatiu d'estudiants o si un nom és més llarg que la mida màxima de name.

- Escriu una comanda amb bash que permeti executar el programa anterior i eliminar les 2 primeres línies de la sortida. També ha de guardar la sortida a un fitxer _notes.txt. **(3 punt)**

Solution:

```
./program && cat notes.txt | tail -n +3  > _notes.txt
```

- Escriu un programa amb bash que llegeixi el resultat anterior i ens retorni una acta amb el nom de l'estudiant i la seva qualificació categòrica segons els següents llindars. Assumiu que l'usuari us passarà els arguments correctes. No cal fer cap mena de verificació. **[5 punt]**.
  - < 5 retornarà la cadena **SUSP**.
  - $>= 5$ i < 7 retornarà la cadena **APRV**.
  - $>= 7$ i < 9 retornarà la cadena **NOTB**.
  - $>= 9$ En aquest últim cas, es demanarà a l'usuari si la nota ha de ser excel · lent (**EXEL.**) o matrícula d'honor (**EXMH.**).

Per exemple, si l'entrada és:

```
cat _notes.txt
john 4
mary 5
peter 7
```

i l'execució del programa és:

```
./acta.sh _notes.txt
```

La sortida serà:

```
Name    Mark    Qualification
----    ----    -------------
John    4.00    SUSP
Mary    5.00    APRV
Peter   7.00    NOTB
```

Solution:

```
#!/bin/bash

declare -A marks

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

for name in "${!marks[@]}"; do
    mark=${marks[$name]}
    if [ $(echo "$mark < 5" | bc) -eq 1 ]; then
        qual="SUSP"
    elif [ $(echo "$mark >= 5 && $mark < 7" | bc) -eq 1 ]; then
        qual="APRV"
    elif [ $(echo "$mark >= 7 && $mark < 9" | bc) -eq 1 ]; then
        qual="NOTB"
    elif [ $(echo "$mark >= 9" | bc) -eq 1 ]; 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
```

- Afegeix al programa anterior la possibilitat de guardar les estadístiques (mitjana númerica de cada categoria i percentatge d'aprovats). [**1 punt**].

```
STATS=TRUE ./acta.sh _notes.txt
```

La sortida serà:

```
Name    Mark    Qualification
----    ----    -------------
John    4.00    SUSP
Mary    5.00    APRV
Peter   7.00    NOTB

Statistics
----------
Average mark: 5.33
Percentage of passed students: 66.67%
```

Solution:

```bash
#!/bin/bash

declare -A marks

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

for name in "${!marks[@]}"; do
    mark=${marks[$name]}
    if [ $(echo "$mark < 5" | bc) -eq 1 ]; then
        qual="SUSP"
    elif [ $(echo "$mark >= 5 && $mark < 7" | bc) -eq 1 ]; then
        qual="APRV"
    elif [ $(echo "$mark >= 7 && $mark < 9" | bc) -eq 1 ]; then
        qual="NOTB"
    elif [ $(echo "$mark >= 9" | bc) -eq 1 ]; 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

if [ "$STATS" = "TRUE" ]; then
    total=0
    passed=0
    for name in "${!marks[@]}"; do
        mark=${marks[$name]}
        total=$(echo "$total + $mark" | bc)
        if [ $(echo "$mark >= 5" | bc) -eq 1 ]; then
            passed=$(echo "$passed + 1" | bc)
        fi
    done
    average=$(echo "scale=2; $total / ${#marks[@]}" | bc)
    percentage=$(echo "scale=2; $passed / ${#marks[@]} * 100" | bc)
    echo
    echo "Statistics"
    echo "----------"
    echo "Average mark: $average"
    echo "Percentage of passed students: $percentage%"
fi
```