DJILALI BOUNAAMA KHEMIS MILIANA UNIVERSITY
Faculty of Material Sciences and Computer Science
DEPARTMENT OF COMPUTER SCIENCE

Jan 16th ,2025
3rd year L3-CS
T : 1h & 30mn

FINAL EXAME ()

OPERATING SYSTEMS *(B)*

ID: .....G:.....

NAME: .....

Exercice 1 : (6 pts)

Etant donné le programme ci-dessous en supposant que le PID Shell est **1750**, le PID correspondant à ce programme est **1990** et que le système affecte des identificateurs séquentiels aux nouveaux processus.

```
/* Programme Exam_1.c  */
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
int main ()
{
        int i= 3 ;
        int j=3;
        int p ;
        p = fork();   i ++ ;

        if (p == 0) {
        int i = 1 ;
        p = fork () ; i += 2 ;
                        if (p == 0) {
                                p = fork () ; j += 4 ;

                                printf("processus=%d,i=%d; j=%d ;Proc.père=%d\n",getpid(),i,j,getppid());
                        return 0 ;
                        }
                }
                else    {
                        p = fork () ; i += 2 ;
                }

                p = fork () ; i *= 2 ; j *= 2 ;

                printf("processus=%d,i=%d;j=%d;Proc.père=%d\n",getpid(),i,j,getppid()) ;
                return 0 ;
        }
```

1. Dérouler le programme et compléter les informations suivantes. :

Le nombre de processus générés dans le programme est : ……8……

	Min	Max
PID	1990	1997
i	3	12
j	6	7

2processus=1990,i=12; j=6;Proc.père=1750
2processus=1991,i=8; j=6;Proc.père=1990
2processus=1992,i=12; j=6;Proc.père=1990
2processus=1993,i=12; j=6;Proc.père=1990
1processus=1994,i=3; j=7 ;Proc.père=1991
2processus=1995,i=12; j=6;Proc.père=1992
2processus=1996,i=8; j=6;Proc.père=1991
1processus=1997,i=3; j=7 ;Proc.père=1994

Exercice 2 :

Consider N processes Pi and an independent process Pr, with the following schema: :

Process Pi:	**Independent Process Pr:**
Begin	**Begin**
PA ;	IA;
PB ;	IB;
End	**End**

(PA, PB, IA, and IB are blocks of instructions)

- The N processes Pi and the independent process Pr execute in parallel.
- Each process Pi executes the instruction block PA and then blocks.
- After completing the instruction block IA, the independent process Pr waits for all processes Pi to finish their respective PA blocks; it then proceeds to execute the IB block.
- Once the IB block is completed, the independent process Pr releases all blocked processes Pi, allowing them to continue their execution.

Q: Propose a synchronization schema for the processes Pi and the independent process Pr using semaphores.

Declarations :
SMaitre : semaphore (init to 0);
SP : semaphore (init to 0);
mutex : semaphore (init to 1);
i : entier (init to 0);
N : constant representing the number of processes « Pi ».

Processus Pi	**Processus Maître**
Begin	Begin
PA ;	MA ;
P(mutex) i := i + 1; V(mutex)	P(SP);
Si i == N alors V(SP) Finsi	MB ;
P(SMaitre);	V(SMaitre)
PB ;	End.
V(SMaitre)	
End.	

Exercise:

Consider two categories of activities: authors and readers.

- **Authors** write books and place them in a shared bookshelf called: **Library Shelf**.
- **Readers** borrow and read the books placed on the shelf.
- The library shelf has a limited capacity of **N** books.

The operation of these two categories of activities must satisfy the following constraints:

1. **Authors** do not place more books when the library shelf is full.
2. **Readers** do not take books from the library shelf when it is empty.
3. **Only one person** (either an author or a reader) can access the library shelf at a time.
4. Books must not be lost or borrowed twice.

Q: Give the synchronization solution using monitors.

```
Program ProducersConsumers;
Const  N=…;
Type   object=….;

Monitor ProdCons;
Const  N=…;

Var    Buffer : Array [0…N-1] of object;
       nonEmpty , nonFull  : condition;
       in,out : integer
       Counter:0….N-1;

Procedure deposit (ob:object);
Begin
        If Counter=N then nonFull.wait;
        Buffer[in]:=ob;
        In:=in+1modN;
        Counter:=Counter+1;
        nonEmpty.signal;
End;

Procedure withdraw (var ob:object);
Begin
        If Counter=0 then nonEmpty.wait;
        ob:= Buffer[out];
        out:=out+1modN;
        Counter:=Counter-1;
        nonFull.signal;
End;

Begin
        Counter:=0;
        In:=0;
        Out:=0;
End;
```

```
Process Author-I;
Var  objectproduce:object;
Begin
  Repeat
        Produce (objectproduce);
Call ProdCons. deposit (objectproduce);

  Until End= true;
```

```
Process Reader-j;
Var objectconsume: object;
Begin
Repeat
Call ProdCons.withdraw (objectconsume);

consume (objectconsume);
 Until Fin= true;
```

End ;	**End ;**

Begin
ParBegin

Author-1; Author-2; Author-3; …….; Author-I;
Reader-1; Reader-2; Reader-3;……; Reader-j;

ParEnd;
End;