UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Departamento de Informática de Sistemas y Computadores (DISCA)

fSO

EEE1
November, 11th 2016

etsinf Escola Tècnica Superior d'Enginyeria Informàtica

SURNAME		NAME		Group
ID		Signature		

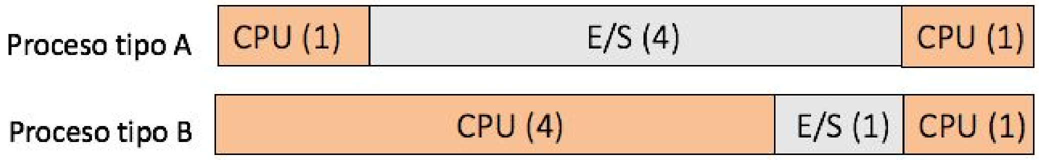
- **Keep the exam sheets stapled.**
- **Write your answer inside the reserved space.**
- **Use clear and understandable writing. Answer briefly and precisely.**
- **The exam has 7 questions, everyone has its score specified.**

1. Considering the following data stored on a process PCB, explain for everyone if it changes or not from the process start to its end and what is the change cause.

(1,2 points = 0,3+0,3+0,3+0,3)

1a	**PID**
1b	**PPID**
1c	**Copy of program counter and CPU registers**
1d	**State**

2. Let's consider a system with only one CPU and I/O made by one hard disk, that allow a **maximum concurrency of 2 processes**. Two type of processes can arrive to this system, they alternate CPU and I/O bursts as described on the following picture (burst length is indicated on parenthesis):



Ask the following questions as much concisely as possible:

(1,7 points = 0,6+0,6+0,5)

2a	Describe what is a CPU bound process and what is an I/O bound process, and tell to what kind correspond type A processes and type B processes
2b	Indicate what process combination and arrival order gives the maximum performance (throughput) and the maximum CPU utilization. Justify your answer graphically and compute the throughput and CPU utilization. **Note**. Running processes can be of the same type A or B
2c	If the hard disk is replaced by an SSD reducing I/O burst time in 1/4, indicate what mix of process types will provide the maximum throughput.

3. Given the following programs:

F.c code:

```c
#include <stdlib.h>
#include <stdio.h>
int main(int argc, char * argv[]){
  int pid,i,status;
  for(i=0;i<3;i++){
      pid=fork();
      if (pid==0){
      printf("exit\n");
      exit(i);
      }
  }
  i=0;
  while(wait(&status)>0) i++;
  printf("wait(%d)\n",i);
}
```

E.c code:

```c
#include <unistd.h>
int main(int argc, char * argv[]){
     execl("./F","F",NULL);
     execl("./F","F",NULL);
}
```

Suppose that both are compiled into E and F on the default folder. **(1,6 points = 0,4+0,4+0,4+0,4)**

3a	In all, how many processes creates the command ./F? Draw the process tree.
3b	What is the command ./F output on the terminal?

3c	From processes created with command ./F, how many become orphan? Explain your answer.
3d	How many processes creates command ./E? Explain your answer

4. We intend to analyze short term schedulers for the operating system running on a computer. The selected schedulers are: SJF (Shortest-Job-First), SRTF (Shortest-Remaining-Time-First), y RR (Round-Robin) with 1 ut *quantum* (q = 1). The analysis relies on the mean turnaround time, mean waiting time and CPU utilization of the following processes:

Process	Arrival time	CPU and I/O bursts
A	0	4 CPU + 1 I/O + 1 CPU
B	1	2 CPU + 2 I/O + 1 CPU
C	4	1 CPU

Fill up the following execution tables for every scheduler and compute the mean turnaround time, the mean waiting time and the CPU utilization. In case of simultaneous events consider the following order: 1) new job arrival, 2) job ending, 3) leaving suspended state y 4) quantum ending. Which one will be the best scheduler considering independently the mean turnaround time, the mean waiting time and the CPU utilization.

(1,7 points = 0,5 + 0,5 + 0,5+ 0,2)

4a) SJF (Shortest-Job-First)

T	Ready	CPU	I/O Queue	I/O	Event
0					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					

	A	B	C	*Mean*
Turnaround time				
Waiting time				
CPU utilization				

4b) SRTF (Shortest-Remaining-Time-First)

T	Ready	CPU	I/O Queue	I/O	Event
0					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					

	A	B	C	*Mean*
Turnaround time				
Waiting time				
CPU utilization				

4c) RR (Round-Robin) q = 1 ut

T	Ready	CPU	I/O Queue	I/O	Event
0					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					

	A	B	C	*Mean*
Turnaround time				
Waiting time				
CPU utilization				

4d) Scheduler selection

Relying index	Best scheduler	Best value
Turnaround time		
Waiting time		
CPU utilization		

5. Program ThreadsAdd.c from the threads lab session is shown below. It does the addition of a bidimensional matrix.

```
#include <stdio.h>
#include <pthread.h>

#define DIMROW 1000000
#define NUMROWS 20

typedef struct row{
  int vector[DIMROW];
  long addition;
} row;

struct row matrix[NUMROWS];
long total_addition=0;

void *AddRow( void *ptr )
{
  int k;
  row *fi;
  fi = (row *)ptr;

  fi->addition=0;
  for(k=0;k<DIMROW;k++)
    fi->addition +=
    exp((k*(fi->vector[k])+(k+1)*
    (fi->vector[k]))/
    (fi->vector[k]+2*k))/2;
  //APDO.D
}
```

```
int main()
{
  int i,j;
  pthread_t  threads[NUMROWS];
  pthread_attr_t atrib;

  // Vector elements are initialized to 1
  for(i=0;i<NUMROWS;i++)
    for(j=0;j<DIMROW;j++)
      matrix[i].vector[j]=1;

  // Thread attributes initialization
  pthread_attr_init( &atrib );

  for(i=0;i<NUMROWS;i++){
    pthread_create(&threads[i],&atrib,
            AddRow,(void *)&matrix[i]);
    //APDO.C
  }

  for(i=0;i<NUMROWS;i++)
    pthread_join(threads[i],NULL);

  for(i=0;i<NUMROWS;i++)
    total_addition += matrix[i].addition;
  printf("Total addition is: %ld \n",
           total_addition);
  pthread_exit(0);
}
```

Answer the following questions about program ThreadsAdd.c. In all questions the changes are done on the initial code and we suppose that the execution is done on a multicore processor.

(1,2 points = 0,3 + 0, 3 + 0, 3 + 0,3)

5a	What will be the effect on the final result got if lines 45 and 46 are removed?
5b	What will be the final result and the execution time got if lines 45, 46 and 51 are removed?
5c	What will be the final result and the execution time got if lines 45 and 46 are removed and the sentence "`pthread_join(threads[i],NULL);`" is appended at line 42?
5d	If lines 48 and 49 are removed and the sentence "`total_addition += fi->addition;`" is appended at line 24, will be the result got correct?

6. Given functions `add` and `sub` seen on lab sessions, answer the following questions:
NOTE. Consider that before declaring the former functions the following constants and variables are declared:

```
#define REPEAT  20000000
long int V = 100;
int Key = 0;
```

```
1   void *add (void *argument){
2    long int count;
3    long int aux;
4
5    for (count=0; count<REPEAT; count++) {
6
7      V = V + 1;
8
9    }
10
11   printf("--> End ADD (V=%ld) \n", V);
12   pthread_exit(0);
13  }
14
```

```
15  void *sub (void *argument) {
16   long int count;
17   long int aux;
18
19   for (count=0; count<REPEAT; count++) {
20
21     V = V - 1;
22
23   }
24
25   printf("--> End SUB (V=%ld)\n", V);
26   pthread_exit(0);
27  }
28
```

```
29  int main (void) {
30   pthread_t threadAdd, threadSub,
31   pthread_attr_t attr;
32
33   pthread_attr_init(&attr);
34   pthread_create(&threadAdd, &attr, add, NULL);
35   pthread_create(&threadSub, &attr, sub, NULL);
36
37   pthread_join(threadAdd, NULL);
38   pthread_join(threadSub, NULL);
39
40   fprintf(stderr, "-------> FINAL VALUE: V = %ld\n\n", V);
41   exit(0);
42  }
```

(1,6 points = 0,4 + 0,4 + 0,4 + 0,4)

6a	What is a critical section? If there are critical sections in the former code, tell on **what line numbers** they are and on what lines the input protocol and the output protocol to protect them have to be added.

6b	If we protect the critical sections using basic Test&Set hardware solution, write the code that implements the input protocol and the output protocol.
6c	Explain if the basic Test&Set hardware solution, the one used before, comply with the three conditions of the critical section access protocols.
6d	Explain the relation between basic Test&Set hardware solution and active (busy) waiting. Under what conditions is recommended to use active waiting on the critical section access protocols?

7. Given three semaphores: A, B and C initialized as: A=0, B=1, C=0; there are 3 processes that run concurrently and perform the following accesses to the semaphores:

Process 1	**Process 2**	**Process 3**
. . **P(A);** . . . **V(C);**	. . **P(B);** . . . **V(A);** **V(B);**	. . **P(B);** **P(C);** . . **V(B);**

(1 point = 0,5 + 0,5)

7a Indicate one sequence of P and V operations of the three processes that allow all of them to end.

PROCESS	**SEMAPHORE OPERATION**

7b Indicate one sequence of P and V operations of the three processes that end with a deadlock.

PROCESS	**SEMAPHORE OPERATION**