UNIVERSITAT POLITÈCNICA DE VALÈNCIA
Departamento de Informática de Sistemas y Computadores (DISCA)
etsinf Escola Tècnica Superior d'Enginyeria Informàtica

EEE1
October, 31th 2019

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. Answer the following questions in a short but clear way:

(1.4 points = 4 x 0.35)

1	a) ¿What is understood by CPU and I/O concurrency?.
	b) ¿Why processors have two execution modes?¿which are they?
	c) ¿Which of the following elements belong to the operating system kernel and which don't? ***process manager*** / ***shell*** / ***memory manager*** / ***device handler*** / ***ls command*** / ***system calls interface***/ ***internet navigator*** ***Belong to OS kernel****:* ***Don't belong to OS kernel****:*
	d) ¿What is CPU utilization? write the formula that computes it

2. Consider that there is a file "hello.txt" in the current working directory, containing the text "hello \n", so that command "cat hello.txt" prints a line with the word hello. Assume that the following program is compiled and executed, for every value of X = 1, 2, 3 and 4. Expose for each of 4 cases (values of X), what is displayed on the terminal when the program is executed and explain your answers.

```
 1 #include <...all headers...>
 2 #define X 1    //1,2,3,4
 3
 4 int main(int argc, char *argv[]) {
 5     int val = 0;
 6     int parent_pid = getpid();
 7
 8     if (X >= 3) val = fork();
 9     if (val == 0) {
10         if (X%2 == 1) // X odd
11             execl("/bin/cat", "cat", "hello.txt", NULL);
12         else
13             execl("/cat/bin", "cat", "hello.txt", NULL);
14     }
15
16     if (getpid() == parent_pid)
17         printf("parent\n");
18     else
19         printf("child\n");
20     return 0;
   }
```

Nota: ***a % b*** returns the remainder of the integer division ***a / b***.

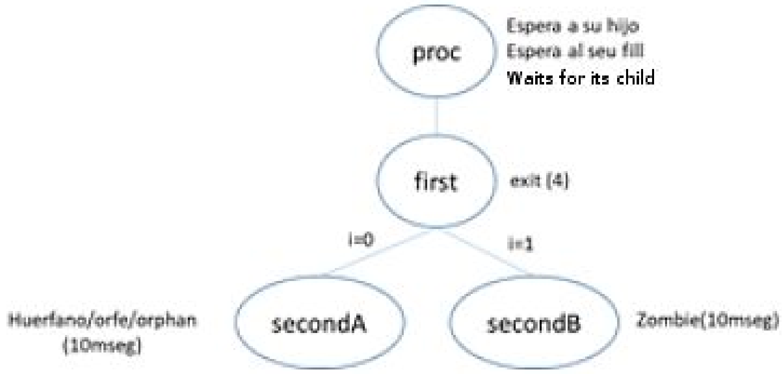
(1.4 points = 0,35+0,35+0,35+0,35)

	a) Case X=1 (#define X 1)
	b) Case X=2 (#define X 2)
	c) Case X=3 (#define X 3)
	d) Case X=4 (#define X 4)

3. Complete the following code for *proc.c* which executable file will be named ***proc,*** in such a way that:

- When executing ***proc*** a new process ***first*** will be created, which in turn will create two children ***secondA*** and ***secondB***.
- Process ***proc*** has to wait for its child ***first*** and ***first*** has to finish with ***exit(4)***.
- Processes ***secondA*** will endup orphan and ***secondB*** zombie along, at least, 10 miliseconds. To achieve this it is suggested to use ***sleep(),*** with ***sleep(10)***, ***sleep(20)*** y ***sleep(30),*** when considered appropriate.

The following diagram shows the parent-child relations and processes actions.



(1.1 points)

3.

```
//Program proc.c
#include <all required.h>
int main(int argc, char *argv[]) {
    int val1, val2, status;
    int i, pid;
    printf("Process proc\n");
    val1 = fork();






    while ((pid=wait(&status)) > 0)
        printf("hijo esperado %d, estado %d\n", pid, status/256);
    exit(0);
}
```

4. In a timesharing system there is a single I/O device that is managed by a FCFS queue. To this system 3 processes arrive: A, B and C. Their arrival instants, priority (being 1 the highest priority, 3 the lowest) and standing alone processing sequences are on the following table:

Process	Arrival	Priority	Standing alone processing sequence
A	0	3 (-)	4 CPU + 1 I/O + 3 CPU + 1 I/O + 4 CPU
B	1	2	2 CPU + 2 I/O + 3 CPU + 1 I/O + 3 CPU
C	2	1 (+)	1 CPU + 5 I/O + 1 CPU + 5 I/O + 1 CPU

(**2.1 points = 1.1+ 0.3+0.3+0.4)**

4 a) Fill the following table with the processing evolution considering the ready queue scheduled by **preemptive priorities**. In every visit to CPU and I/O indicate the remaining time for that burst.

T	Ready	CPU	I/O queue	I/O	Comment
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					

4 b) Indicate the waiting time (CPU queue) and the turnaround time for every process (**Table PRIO_1**)

PRIO_1	Waiting time	Turnaround time
A		
B		
C		

4 c) Using the same preemptive priority policy on the ready queue scheduler, on the same workload, the following waiting times and turnaround times are obtained. Determine what priorities have been assigned to the processes to obtain these values. . (**Table PRIO_2**)

PRIO_2	T.espera	T.retorno
A	0	13
B	7	26
C	14	28

4 d) The following table shows the results obtained for the same workload using a Round-Robin policy with quantum q = 2ut. **(Table RR**)

RR	T.espera	T.retorno
A	8	22
B	8	19
C	6	21

Relying on tables PRIO_1, PRIO_2 and RR, compute the throughput obtained on every one of the three scheduling policies.

From the results on tables PRIO_1, PRIO_2 and RR, what type of processes have benefited on every policy? Explain your answer.

5. Given the following program ***Threads.c*** whose executable file is ***Threads***, answer the following items:

(1.5 points = 0.6 +0.3+0.3+0.3)

```
#include <... all headers...>
#define NTHREADS 3
pthread_t Th[NTHREADS];
pthread_attr_t atrib;
int N=0;

void *Func(void *arg)
{ int i= (int) arg;
  N=N+i;
  printf("Hilo %d,N = %d\n",i ,N);
  pthread_exit(0);
}
```

```
int main (int argc, char *argv[]) {
  pid_t pid;
  int i;
  printf("START MAIN \n");
  pthread_attr_init(&atrib);

  pid=fork();
  for (i=0;i<NTHREADS;i++)
   {pthread_create(&Th[i],&atrib, Func,(void *)i);
    pthread_join(Th[i],NULL); }

  if (pid == 0) pthread_exit(0);
  else exit(0);
  printf("END MAIN \n");
}
```

5	**a)** Indicate the sequence (one of the possible ones) that the program prints on the Terminal when executing it. Explain your answer.
	b) Indicate the maximum number of threads that could be running concurrently when executing the ***Threads*** program is executed. Explain your answer.
	c) Indicate if there is a risk of race condition when running the ***Threads*** program. Explain your answer.
	d) Assume that the threads on the ***Threads*** process are user level (runtime) threads, what the system's scheduler should manage in that case?. Explain your answer.

6. On a computer with a single CPU running an operating system with a Round-Robin scheduler, it is desired to solve the problem of accessing to a critical section with ONLY TWO THREADS INVOLVED, relying on **variants of the test_and_set()** approach. Explain for every case proposed if the limited waiting condition is met and if it can be considered as active waiting or non-active waiting:

(1.2 points = 0.4+0.4+0.4)

6

a)

Input protocol:	Output protocol:
`while (test_and_set(&key))` `  /*bucle vacio*/ ;`	`key = 0;`

Does it always verify limited waiting?

Is it active waiting?

b)

Input protocol:	Output protocol:
`while (test_and_set(&key))` `  usleep(100) ;`	`key = 0;`

Does it always verify limited waiting?

Is it active waiting?

c)

Input protocol:	Output protocol:
`while (test_and_set(&key))` `  usleep(100);`	`Key = 0;` `usleep(105);`

Does it always verify limited waiting?

Is it active waiting?

7. Describe a possible sequence of the concurrent execution of threads ThA, ThB and ThC.

```
// Initial values
 int x=0, y=0;
 Semaphore: S1=0, S2=1, S3=0;
```

ThA	**ThB**	**ThC**
P(S3); P(S2); x = x + 1; y = 2*x + y; V(S2); V(S1);	P(S1); P(S2); x = 2*x; y = x + y; V(S2);	P(S2); x = x - 2; y = x - y: V(S2); V(S3);

Use the following table to keep track of the threads operations, as well as the values of variables and semaphores.

(1.3 points)

7

T		S1=0	S2=1	S3=0	X=0	Y=0
1						
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
Final values						