UNIVERSITAT POLITÈCNICA DE VALÈNCIA
Departamento de Informática de Sistemas y Computadoras (DISCA)

fSO

EEE1
November, 4th 2013

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

SURNAME		**NAME**		Group
IDN		**Signature**		E

- **Keep the exam sheets stapled.**
- **Write your answer inside the reserved space.**
- **Use clear and understandable writting. Answer in a brief and precise way.**
- **The exam has 9 questions, every one has its score specified.**

---

1. Suppose that you are working in a compay where you have to assist customers that do queries by phone and the reply should be on line, as quick as possible. To do the job you have to perform actions and queries on a computer system. What kind of OS should be installed on the computer: batch monoprogrammed, multiprogrammed or time sharing, explain your answer.

(0,75 points)

1	

---

2. Answer the following questions in relation to an OS:

a) What are the differences between a shell command and a system call? Explain your answer on both the concept and the use.
b) Enumarate the sequence of actions that an OS takes when a user application performs a system call, indicate which are done in kernel mode.

(0,75 points)

2	**a)**
	b)

3. Given the following code in C and POSIX corresponding to a process named Proc:

```
 1 #include …. // as required
 2 #define N 3
 3
 4 main() {
 5   int i = 0;
 6   pid_t pid_a;
 7
 8   while (i<N)
 9   { pid_a = fork();
10     switch (pid_a)
11     { case -1:
12         printf("Error creating child...\n");
13         break;
14       case 0:
15         printf("Message 1: i = %d \n", i);
16         if (i < N-1) break;
17         else exit(0);
18       default:
19         printf("Message 2: i = %d \n", i);
20         while (wait(NULL)!=-1);
21      }
22     i++;
23   }
24   printf("Message 3: i=%d\n",i);
25   exit(0);
26 }
27
```

a) Draw the process tree generated when it is running and indicate for every process at what value of "i" it has been created.

b) Explain if there is a possibility of appearing orphan and/or zombie children.

(1,0 point)

3	**a)**
	b)

4. Given the following code in C and POSIX corresponding to a process named Test, that runs successfully:

```
 1 #include …. // as required
 2
 3 int main() {
 4   pid_t val;
 5   printf("Menssage 1: before exec()\n");
 6   fork();
 7   execl("/bin/ls","ls","-la", NULL);
 8   val = fork();
 9   if (val==0) {
10     execl("/bin/ps","ps","-la", NULL);
11     printf("Menssage 2: after exec()\n");
12     exit(1)
13   }
14   printf("Message 3: before exit()\n);
15   exit(0);
16 }
```

a) Explain the number of processes that are creating when executing Test and the parent/child relationship.
b) Indicate and explain the messages and information shown on the screen when executing Test.

(1,0 point)

4	**a)**
	b)

5. Given the following code corresponding to process Example:

```
 1 #include …. // as required
 2
 3 int main(void) {
 4   int val;
 5   printf("Message 1\n");
 6   val=fork();
 7   /** Write here your changes **/
 8   sleep(5);
 9   printf("Message 2\n");
10   return 0;
11 }
```

a) Indicate what changes are required in the previous code in order to make the child process orphan and to be adopted by `init`. (**Note**: Use sleep() and C senteces).
b) Indicate what chantes are required in the previous code in order to make the child process zombie for a while. (**Note**: Use sleep() and C sentences).
c) Explain in what instructions in the proposed code we can be sure that a CPU context switch will happen and in what intructions the will be an state change in the Example process.

(1,0 points)

5	
	a)
	b)
	c)

---

6. Given the following code that tries to solve the race condition problem, and assuming that there are more than one thread executing concurrently the code of the "add" function, indicate if the following sentences are true or false (**Note**: An error voids a correct answer).

```
1   void *add (void *argument) {
2     long int count;
3     long int aux;
4     while(test_and_set(&key)) ;
5     for (count = 0; count < REPETITIONS; count = count + 1) {
6       V = V + 1;
7     }
8     key =0;
9     printf("-------> End Add (V = %ld)\n", V);
10    pthread_exit(0);
11  }
```

(1,0 points)

6	**T/F**	
		Line 4 implements the input protocol to the critical section.
		The proposed solution doesn't guarantee that it is race condition free
		The proposed solution guarantees that the code is race condition free, because the access to the global variable V in done sequentially
		If we interchange line 4 - 5, and 7 - 8, the resulting code is race condition free
		Function call `test_and_set(&key)` tests and sets the value of key in an atomic way

7. Given the following code that has generated an executable file named "Threads1".

```
 1 #include … // as required
 2 #define TwoSeconds 2000000
 3 pthread_t H1,H2;
 4 pthread_attr_t atr;
 5
 6 void *WorkB (void *P)
 7 { char * text =(char *) P;
 8   usleep(TwoSeconds);
 9   printf("%s\n",text);
10 }
11
12 void *WorkA (void *T)
13 { printf("Text:\n");
14   pthread_create(&H2, &atr, WorkB, T);
15   usleep(TwoSeconds);
16   pthread_join(H2, NULL);
17 }
18
19 int main() {
20   pthread_attr_init(&atr);
21   pthread_create(&H1, &atr, WorkA, "FSO exam");
22   usleep(TwoSeconds);
23   pthread_join(H1, NULL);
24   pthread_join(H2, NULL); /* Thread created inside WorkA*/
25   return(0);
   }
```

a) Indicate and explain the strings that are printed in the terminal after Threads1 execution.
b) Indicate and explain the approximate time that Threads1 will take for execution.
c) Explain if in main() it will be correct not to wait for thread H2 that is the one created inside WorkA.
d) During the execution of Thread1, how many active threads are there when "Text:" is written in the terminal, and how many when it is written "FSO exam"? Explain your answer.

(1,0 points)

7	**a)**
	b)
	c)

d)

8. In a computer system there are three process classes: OS processes (QOS), user processes (QU) and background processes (QB). The short term scheduler in the system has three queues, one for every class of processes. The queue OS uses preemtive priorities (PP), the queue U uses STRF and the queue QB uses RR with q=1. **The inter queue scheduling is preemtive pritorities**, being QOS the one with higher priority and QB the one with lower priority. I/O operations are performed on the same device with FCFS scheduling.
Two processes of every class arrive to the system at the instants specified in the following table with the corresponding execution profiles:

Process	Execution profile	Arrival instant	Process class
A (-priority)	2 CPU	2	Sistem (QOS)
B (+priority)	3 CPU	3	Sistem (QOS)
C	1 CPU + 1 E/S + 1 CPU	6	User (QU)
D	2 CPU + 1 E/S + 1 CPU	17	User (QU)
E	4 CPU + 3 E/S + 2 CPU	0	Background (QB)
F	2 CPU + 1 E/S + 2 CPU	5	Background (QB)

a) Fill the following table with the execution time line.
b) Obtain the mean waiting time, the mean turnaround time and the CPU utilization.

2.0 points (1.25+0.75)

8a

T	QOS PP	QU SRTF	QB RR q=1	CPU	I/O Queue	I/O	Comments
0							E arrives
1							
2							A arrives
3							B arrives
4							
5							F arrives
6							C arrives
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							D arrives
18							
19							
20							
21							
22							
23							
24							

8b	Mean waiting time = Mean turnaround time = CPU utilization =

9. Given the following program, where there are three semaphores, with the threads "add" and "sub". Indicate and explain for every one of the values of "x" and "y" proposed, if it could happen or not a race condition, the state at what the threads will end and the final value of V that is written at line 12:

```
#include … // as required
int V = 100;
sem_t sem, add, sub;
```

```
void *Add (void *argument) {
  int count;
  for (count=0; count<100; count++) {
    sem_wait(&add);
    sem_wait(&sem);
    V = V + 1;
    sem_post(&sem);
    sem_post(&sub);
  }
}
```

```
void *Sub (void *argument) {
  int count;
  for (count=0; count<100; count++) {
    sem_wait(&sub);
    sem_wait(&sem);
    V = V - 1;
    sem_post(&sem);
  }
}
```

```
1.   int main (void) {
2.     pthread_t threadAdd, threadSub, threadInspec;
3.     pthread_attr_t attr;
4.     int x, y;
5.     pthread_attr_init(&attr);
6.     sem_init(&sem,0,1);
7.     sem_init(&add,0,x);
8.     sem_init(&sub,0,y);
9.     pthread_create(&threadAdd, &attr, Add, NULL);
10.    pthread_create(&threadSub, &attr, Sub, NULL);
11.    usleep(80000000); // enough to execute Add and Sub
12.    fprintf(stderr, "-------> FINAL VALUE: V = %d\n", V);
13.    exit(0);
14.  }
```

a) Suposse x = 5 and y = 1.
b) Suposse x = 500 and y = 1.
c) Suposse x = 20 and y = 5. (1.5 points)

9	a)
	b)

	c)