# Esame Sistemi Operativi - Operating Systems Exam
# 2024/02/07

## Ex 01 (2.0 points)

**Italiano**

Si supponga che il disco rigido di un piccolo sistema embedded sia costituito da 32 blocchi di 1 MByte, che tali blocchi siano numerati da 0 a 31, che il sistema operativo tenga traccia dei blocchi liberi (occupati) indicandoli in un vettore con il valore 0 (1), e che la situazione attuale del disco sia rappresentata dal seguente vettore:

`0 1 1 0 0 0 1 0 0 1 1 1 1 1 0 0 1 0 1 0 0 1 0 0 1 0 1 0 1 0 0 1`

Si indichino quali delle seguenti affermazioni sono vere. Si osservi che risposte errate implicano una penalità nel punteggio finale.

**English**

Suppose that the hard disk of a small embedded system is composed of 32 blocks of 1 MByte each, which are numbered from 0 to 31. Suppose that the operating system keeps track of the free (occupied) blocks indicating them in a vector with the value 0 (1), and that the current situation of the disk is represented by the following vector:

`0 1 1 0 0 0 1 0 0 1 1 1 1 1 0 0 1 0 1 0 0 1 0 0 1 0 1 0 1 0 0 1`

Indicate which of the following statements are correct. Note that wrong answers imply a penalty in the final score.

Scegli una o più alternative / Choose one or more options

1. ⬜ Un file di dimensione 2.9 MByte PUO' essere allocato utilizzando una strategia di allocazione contigua. A file with dimension 2.9 MByte CAN be allocated using a contiguous allocation strategy.
2. ⬜ Un file di dimensione 1.9 MByte e uno di dimensione 2.5 MByte possono essere allocati utilizzando una strategia di allocazione contigua. A file with dimension 1.9 MByte and one with dimension 2.5 MByte can be allocated using a contiguous allocation strategy.
3. ⬜ Un file di dimensione 15.5 MByte NON può essere allocato utilizzato una strategia di allocazione concatenata. A file with dimension 15.5 MByte CANNOT be allocated using a linked allocation strategy.
4. ⬜ Con la strategia di allocazione contigua ottenuta mediante l'algoritmo BEST-FIT possono essere allocati nell'ordine i file F1 di 1.6 MByte, F2 di 2.9 MByte e F3 di 1.9 MByte. With the contiguous allocation strategy based on the BEST-FIT algorithm, the following files can be allocated in the following order: F1 of 1.6 MByte, F2 of 2.9 MByte, and F3 of 1.9 MByte.
5. ⬜ Con la strategia di allocazione contigua ottenuta mediante l'algoritmo FIRST-FIT possono essere allocati nell'ordine i file F1 di 1.6 MByte, F2 di 2.9 MByte e F3 di 2.6 MByte. With the contiguous allocation strategy based on the FIRST-FIT algorithm, the following files can be allocated in the following order: F1 of 1.6 MByte, F2 of 2.9 MByte, and F3 of 2.6 MByte.
6. ⬜ Con la strategia di allocazione indexata il file di dimensione maggiori allocabile può occupare 17 MByte. With the indexed allocation strategy the largest file that can be allocated may be stored in 17 MByte.

## Ex 02 (3.5 points)

**Italiano**

Si supponga che il seguente programma venga eseguito mediante l'istruzione `./pgrm 2`.

Si indichino i possibili output del programma. Si osservi che risposte errate implicano una penalità nel punteggio finale.

**English**

Suppose that the following program is run using the command `./pgrm 2`.

Indicate all possible outputs of the program. Note that wrong answers imply a penalty in the final score.

```
#include <sys/types.h>
#include <unistd.h>
```

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

#define N 100

int main (int argc, char **argv) {
  char str[N];
  int n;

  setbuf (stdout, 0);
  printf("M");
  sscanf (argv[1], "%d", &n);
  if (n>0) {
    if (fork () > 0) {
      printf("E");
      sprintf (str, "%d", n-1);
      execlp (argv[0], argv[0], str, NULL);
    } else {
      sprintf (str, "%s %d", argv[0], n-1);
      printf("S");
      system (str);
    }
  }
  return 1;
}
```

Risposta. Answer.

1. ⬜ MEMEMSMSMEMSM
2. ⬜ MMEEMSMSMEMSM
3. ⬜ MSMSMEMEMEMSM
4. ⬜ MEMEMSSMEMMSM
5. ⬜ MESMMMESESMMM
6. ⬜ MEMEMSMSEMMSM
7. ⬜ MESMMESESMMMM

## Ex 03 (5.5 points)

**Italiano**

La seguente funzione calcola ricorsivamente in numero di Fibonacci in posizione `num`:

```
int Fibonacci(int num){
  if (num < 2)
    return 1;
  else
    return Fibonacci(num-1) + Fibonacci(num-2);
}
```

Si scriva utilizzando Pthreads un programma concorrente e completo in C (i.e., completo perché deve contenere la funzione `main()` e tutte le parti utili per eseguirlo) che produce gli stessi risultati. In particolare, implementare una versione con i thread del programma Fibonacci rimpiazzando adeguatamente la chiamata ricorsiva della funzione `Fibonacci` con l'attivazione di nuovi threads.

**English**

The following function computes the Fibonacci number of position `num` recursively:

```
int Fibonacci(int num){
  if (num < 2)
```

```
    return 1;
  else
    return Fibonacci(num-1) + Fibonacci(num-2);
}
```

Write a complete concurrent C program (i.e., complete because it must contain the `main()` function and all the parts of the program needed to execute it) using Pthreads that produces the same result. In particular, implement a threaded Fibonacci program properly replacing the recursive calls to the `Fibonacci` function with new threads activations.

Risposta. Answer.

## Ex 04 (2.0 points)

**Italiano**

Quale tra le seguenti affermazioni sono corrette parlando di Process Control Block (PCB)?

**English**

Which of the following statements are correct talking about Process Control Block (PCB)?

Scegli una o più alternative. Choose one or more options

1. ⬜ Quanto un processo termina il suo PCB viene rilasciato immediatamente. When a process dies the PCB is released immediately.
2. ⬜ L'elenco dei file aperti è memorizzato nel PCB. The list of open files is stored into the PCB.
3. ⬜ Tutti i registri della CPU sono memorizzati nel PCB. All CPU registers are stored into the PCB.
4. ⬜ Il PCB di un processo zombie non è ancora stato completamente deallocato. The PCB of a zombie process has not yet been completely deallocated.
5. ⬜ Con il termine context switching si intende la rimozione definitiva del PCB del processo che viene rimosso dallo stato running in favore del nuovo processo che verrà eseguito. The term context switching refers to the definitive deallocation of the PCB of the process being removed from the running state in favor of the new process being executed.
6. ⬜ La tabella dei segnali è salvata all'interno del PCB. The signal table is stored inside the PCB.

## Ex 05 (2.0 points)

**Italiano**

Si supponga il processo P1 diventi "zombie" e il processo P2 diventi "orfano".
Si indichino quali delle seguenti affermazioni sono vere. Si osservi che risposte errate implicano una penalità nel punteggio finale.

**English**

Suppose that the process P1 becomes a "zombie" and the process P2 becomes an "orphan".
Indicate which of the following statements are correct. Note that wrong answers imply a penalty in the final score.

Scegli una o più alternative. Choose one or more options.

1. ⬜ Il PCB del processo P2 è stato cancellato. The PCB of the P2 process was deleted.
2. ⬜ Nel caso del processo P1, per deallocare completamente le risorse allocate al processo, il processo P1 dovrà effettuare una wait o una waitpid. In the case of process P1, to completely deallocate the resources allocated to the process, process P1 will have to perform a wait or a waitpid.
3. ⬜ Il processo P2 ha o avrà un padre diverso da quello che lo ha generato. The P2 process has or will have a different parent from the one that generated it.
4. ⬜ Un processo zombie può continuare a eseguire del codice. A zombie process can continue to execute code.

5. ⬜ Nel caso di P1 il suo PCB verrà cancellato solo dopo che il padre effettuerà una wait o una waitpid. In the case of P1, its PCB will be deleted only after its parent performs a wait or a waitpid.

## Ex 06 (2.5 points)

**Italiano**

Si indichino quali sono le caratteristiche dell'implementazione del paradigma Reader/Writer riportato in seguito. Si indichino quali delle seguenti affermazioni sono vere. Si osservi che risposte errate implicano una penalità nel punteggio finale.

**English**

Indicate which are the characteristics of implementing the Reader/Writer reported below. Indicate which of the following statements are correct. Note that incorrect answers imply a penalty in the final score.

| | | |
|---|---|---|
| `nR = 0;`<br>`init (meR, 1);`<br>`init (meW, 1);`<br>`init (w, 1);` | `wait (meR);`<br>`  nR++;`<br>`  if (nR==1)`<br>`  wait (w);`<br>`signal (meR);`<br>`...`<br>`Critical Section`<br>`...`<br>`wait (meR);`<br>`  nR--;`<br>`  if (nR==0)`<br>`  signal (w);`<br>`signal (meR);` | `wait (meW);`<br>`wait (w);`<br>`...`<br>`Critical Section`<br>`...`<br>`signal (w);`<br>`signal (meW);` |

Scegli una o più alternative / Choose one or more options

1. ⬜ Lo schema è relativo alla precedenza ai Reader. The scheme defines a Reader's precedence.
2. ⬜ A seguito dell'inversione dell'ordine di signal(w); e signal(meW); il programma continua a essere corretto. As a result of the inversion of the order of signal(w); and signal(meW); the program continues to be correct.
3. ⬜ Il mutex meW è inutile perchè duplica semplicemente l'azione effettuata dal mutex w. The mutex meW is useless as it duplicates the mutex w.
4. ⬜ Inizializzando il semaforo meW a 2 si consente a due Writer di accedere alla sezione critica. Initializing the meW semaphore to 2 permits two Writers to access the critical section.
5. ⬜ Il mutex meR potrebbe essere eliminato se i Reader non effettuassero mai il loro prologo o l'epilogo in parallelo. The mutex meR could be eliminated if the Reader never entered their prologue or epilogue in parallel.
6. ⬜ Il semaforo w all'interno dei Reader serve per bloccare tutti i Reader nel caso un Writer sia all'interno della sezione critica. The semaphore w within Readers is used to block all Readers in case a Writer is within the critical section.

## Ex 07 (3.0 points)

**Italiano**

Si chiarisca che cosa si intende per "effetto convoglio" (cioè task brevi in coda a task lunghi attendono molto tempo inutilmente) tramite due esempi basati sull'algoritmo First Come First Served (FCFS): il primo che presenta tale problema e il secondo che non lo presenta. Si faccia vedere con un esempio basato sugli stessi set di task usati in precedenza come tale problema venga risolto utilizzando lo scheduling Shortest Job Firsts (SJF).

**English**

Clarify the meaning of "queue effect" (i.e., short tasks queued after long tasks wait a long time unnecessarily), providing two examples based on the First Come First Served (FCFS) algorithm: the first that presents such a

problem and the second that does not. Provide a further example based on the same set of tasks used above to show how this problem is solved using Shortest Job Firsts (SJF) scheduling.

Risposta. Answer.

## Ex 08 (5.5 points)

**Italiano**

Scrivere un script di shell che riceve come argomenti tramite linea di comando una lista di nomi di file e, dopo aver controllato il corretto passaggio di parametri, per ognuno di essi, se esiste, stampa le seguenti informazioni:

- Se il file è un file regolare, esso stampa il suo nome, la sua dimensione, e se l'utente ha il permesso di lettera e di scrittura.
- Se il file è un link simbolico, esso stampa il nome del file a cui punta (`f1` nell'esempio) e l'utente a cui appartiene.

Si fa notare che l'output del comando `"ls -l foo"`, se il file `foo` esiste è il seguente:

```
-r--r--r-- 1 user user 349 gen 27 2016 foo
```

Invece, l'output del comando `"ls -l lfoo"`, se il file `lfoo` esiste è il seguente:

```
lrwxrwxrwx 1 user user 3 Jan 15 10:12 lfoo -> f1
```

Si supponga che i campi del comando `ls` siano separati da un singolo spazio, e si ottenga esattamente l'output riportato nell'esempio.

Esempio:

```
./prog.sh foo lfoo
```

Output:

```
foo 349 read_perm yes write_perm no
f1 user
```

**English**

Write a shell script that receives as arguments in the command line a list of files and, after checking the parameters, for each one, if it exists, prints the following information:

- If the element is a regular file, it prints its name, its size, and if the user has read and write permissions.
- If the element is a symbolic link, it prints the name of the file it points to (f1 in the example), and the user owner of the file.

Notice that the output of the command `"ls -l foo"`, if the file `foo` exists is like the following:

```
-r--r--r--  1 user user    349 gen 27  2016 foo
```

Instead, the output of the command `"ls -l lfoo"`, if the file `lfoo` exists is the following:

```
lrwxrwxrwx 1 user user 3 Jan 15 10:12 lfoo -> f1
```

Assume that a single space separates the fields of the `ls` command and that the script has to obtain exactly the output provided in the example.

Example:

```
./prog.sh foo lfoo
```

Output:

```
foo 349 read_perm yes write_perm no
f1 user
```

Risposta. Answer.

## Ex 09 (3.0 points)

**Italiano**

Lo stato di un sistema con 5 processi e 3 tipologie di risorse è definito dalle matrici riportate alla fine della domanda. Si supponga il processo P1 effettui una richiesta per le risorse {2, 2, 1}. Si stabilisca se la richiesta può essere soddisfatta permettendo al sistema di rimanere in uno stato sicuro. In caso affermativo si risponda

“YES” e si riporti la sequenza di esecuzione dei processi. Ad esempio, si risponda “YES 5 4 3 2 1” nel caso in cui la sequenza sicura sia P5, P4, P3, P2, P1. In caso contrario, si risponda “NO”.

**English**

The state of a system with 5 processes and 3 types of resources is defined by the matrices reported at the end of the question. Suppose that process P1 requests the resources {2, 2, 1}. Indicate whether or not the request can be granted, allowing the system to remain in a safe state. In the affirmative case, respond with “YES” and report the sequence in which the processes will be executed. For instance, respond “YES 5 4 3 2 1” in the case the safe sequence is P5, P4, P3, P2, P1. Otherwise, respond with “NO”.

| **Processo**<br>**Process** | **Assegnate**<br>**Allocation** | **Massimo**<br>**Max** | **Disponibile**<br>**Available** |
|---|---|---|---|
| **P1** | 0 0 0 | 4 3 5 | 3 3 3 |
| **P2** | 0 0 0 | 4 5 7 | |
| **P3** | 0 2 1 | 5 4 6 | |
| **P4** | 1 1 1 | 2 1 3 | |
| **P5** | 1 2 2 | 3 4 2 | |

Risposta. Answer.

## Ex 10 (2.0 points)

**Italiano**

Sia dato il seguente comando bash:

```
egrep "^1.*[02468]\>" file.txt | egrep "AA(AA)*" | tr -s "A" | cut -d " " -f 1
```

Riportare quali righe vengono riportate in output quando il comando viene eseguito sul file file.txt riportato alla fine della domanda.

Si osservi che risposte errate implicano una penalità nel punteggio finale.

**English**

Given the following bash command:

```
egrep "^1.*[02468]\>" file.txt | egrep "AA(AA)*" | tr -s "A" | cut -d " " -f 1
```

Report which lines are displayed when the previous command is executed on the file file.txt reported at the end of the question.

Note that incorrect answers imply a penalty in the final score.

Contenuto del file file.txt Content of the file file.txt

```
10102 AA
10125 A
13194 AAAAAA
13148 A
15792 AAAA
10123 AAA
20210 AAA
20122 AAAA
```

Scegli una o più alternative: Choose one or more options:

1. ⬜ `10102 AA`
2. ⬜ `10102`
3. ⬜ `13194`
4. ⬜ `13194 AAAAAA`
5. ⬜ `15792`
6. ⬜ `20122 AAAA`

7. ⬜ 20122
8. ⬜ 15792 AAAA
9. ⬜ 13148 A
10. ⬜ 13148

## Ex 11 (3.0 points)

**Italiano**

Sia dato un sistema con 3 processi (P1, P2, P3) e 3 tipi di risorse (R1, R2, R3), in cui vi sia 1 singola istanza di risorse di tipo R1 e 2 istanze per le risorse di tipo R2 e R3. Lo stato del sistema è il seguente:

- Il processo P1 detiene entrambe le due istanze di risorse di tipo {R2}, detiene un'istanza della risorsa di tipo {R3} ed è in attesa della risorsa {R1}
- Il processo P2 detiene la risorsa {R1}, è in attesa di un'istanza della risorsa di tipo {R2} ed è in attesa di entrambe le due istanze di risorse di tipo {R3}
- Il processo P3 detiene un'istanza della risorsa di tipo {R3} ed è in attesa di un'istanza della risorsa di tipo {R2}

Si effettui un'analisi della situazione dal punto di vista del deadlock basandosi ESCLUSIVAMENTE sulla presenza di cicli nel grafo di allocazione delle risorse.
Si indichi quali delle seguenti affermazioni sono corrette. Si osservi che risposte errate implicano una penalità nel punteggio finale.

**English**

A system has 3 processes (P1, P2, P3) and 3 types of resources (R1, R2, R3), in which the number of instances of resources of type R1 is 1, and the number of instances for each resource of type R2 and R3 is 2. The state of the system is the following:

- Process P1 holds both instances of resources of type {R2}, holds an instance of resource of type {R3} and it waits resource {R1}
- Process P2 holds resource {R1}, it waits an instance of resource of type {R2}, and it waits both instances of resources of type {R3}
- Process P3 holds an instance of resource of type {R3} and it waits an instance of resource of type {R2}

Analyze the situation from the point of view of the deadlock, focusing the analysis ONLY on the presence of cycles in the resource allocation graph.
Please, indicate which of the following statements are correct. Note that incorrect answers imply a penalty in the final score.

Scegli una o più alternative: Choose one or more options:

1. ⬜ Il sistema SICURAMENTE NON è in una condizione di deadlock The system SURELY is NOT in a deadlock condition.
2. ⬜ Eliminando l'arco P1->R1 si ha la CERTEZZA di NON essere in una condizione di deadlock Eliminating the edge P1->R1, SURELY the system is NOT in a deadlock condition.
3. ⬜ Eliminando l'arco R2->P1 si ha la CERTEZZA di NON essere in una condizione di deadlock Eliminating the edge R2->P1, SURELY the system is NOT in a deadlock condition.
4. ⬜ Il deadlock è POSSIBILE The deadlock is POSSIBLE.
5. ⬜ Eliminando l'arco R1->P2 si ha la CERTEZZA di NON essere in una condizione di deadlock Eliminating the edge R1->P2, SURELY the system is NOT in a deadlock condition.
6. ⬜ Eliminando l'arco P2->R3 si ha la CERTEZZA di NON essere in una condizione di deadlock Eliminating the edge P2->R3, SURELY the system is NOT in a deadlock condition.
7. ⬜ Il sistema SICURAMENTE è in una condizione di deadlock The system is SURELY in a deadlock condition.

## Ex 12 (2.0 points)

**Italiano**

Quali delle seguenti affermazioni sulle pipe sono corrette. Si osservi che risposte errate implicano una penalità nel punteggio finale.

**English**

Which of the following pipe statements are correct. Note that incorrect answers imply a penalty in the final score.

Scegli una o più alternative: Choose one or more options:

1. ⬜ Un'operazione di lettura su una pipe è sempre bloccante. A read operation on a pipe is always blocking.
2. ⬜ Su una pipe non si possono effettuare operazioni di lettura/scrittura di dati strutturati (e.g., una struct) se non utilizzando una sequenza di operazioni di read/write successive. On a pipe it is not possible to execute read/write operations on a composite data structure (such as a struct) if not using a sequence of read/write operations.
3. ⬜ Una pipe si riempie quando lo scrittore scrive troppo sulla pipe senza che il lettore legga nulla. A pipe gets filled up when the writer writes too much to the pipe without the reader reading any of it.
4. ⬜ Un'operazione di scrittura è tipicamente bloccante su una pipe piena. A write operation is typically blocking on a full pipe.
5. ⬜ Le operazioni di scrittura su una pipe sono atomiche fino alla dimensione della costante PIPE_BUF. Write operations on a pipe are atomic up to the size of the constant PIPE_BUF.
6. ⬜ E' corretto avere più processi che leggono contemporaneamente da una pipe. It is correct to have more than one process reading from a pipe at the same time.
7. ⬜ Si può scrivere da ambo il lati della pipe contemporaneamente. It is possible to write on both sides of the pipe at the same time.