Esame Sistemi Operativi - Operating Systems Exam
2021/01/29

Ex 1 (3.0 points)

Italiano

Un programma concorrente è costituito da un unico processo di nome P1 (e funzione P1()), di tipo ciclico, di cui sono presenti 2 istanze.

Il comportamento del programma è il seguente:

- All'inizio le 2 istanze del processo P1 sono eseguite in parallelo.
- Quando entrambe le istanze di P1 hanno finito di eseguire il proprio codice (quello presente nella funzione P1()), entrambe le istanze di P1 sono eseguite nuovamente.

Si indichino quali dei seguenti codici sono corretti. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

A concurrent program is composed of a single process called P1 (and a function P1()). The process, and consequently the function, are cyclic. There are 2 instances of process P1.

The behavior of the program is as follows:

- At the beginning, the 2 instances of process P1 are executed in parallel.
- When both instances of P1 have finished executing their code (i.e., the code that is present in function P1()), both instances of P1 are executed again.

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

Scegli UNA SOLA alternativa. Choose JUST ONE option.

1. ⬜
```
int n=0;
init(s, 2);
init(m, 1);
init(b, 0);
while(1) {
	wait(s);
	P1();
	wait(m);
	n++;
	if (n==2) {
		signal(s);
		signal(s);
		n=0;
		signal(b);
	} else {
		signal(m);
		wait(b);
	}
}
```
2. ✅
```
int n=0;
init(s, 2);
init(m, 1);
init(b, 0);
while(1) {
	wait(s);
	P1();
	wait(m);
	n++;
	if (n==2) {
		signal(s);
		signal(s);
```

```
                        n=0;
                        signal(b);
                        signal(m);
                  } else {
                        signal(m);
                        wait(b);
                  }
            }
```

3. ⬜

```
            int n=0;
            init(s, 2);
            init(m, 1);
            while(1) {
                  wait(s);
                  P1();
                  wait(m);
                  n++;
                  if (n==2) {
                        signal(s);
                        signal(s);
                        n=0;
                  }
                  signal(m);
            }
```

Ex 2 (3.0 points)

Italiano

Si supponga di eseguire il seguente programma. Si riporti l'output da esso generato rispettando esattamente il formato prodotto.

Si supponga che il programma venga eseguito su un sistema operativo nel quale l'attesa di 1 secondo sia sufficientemente lunga per completare tutti gli altri task in esecuzione in quel momento.

Si prega di riportare la risposta su un'unica riga, indicando i vari messaggi/valori in output separati da un unico spazio. Non inserire nessun altro carattere nella risposta.

English

Suppose you are running the following program. Report the output it generates, remembering to respect exactly the output format produced.

Assume that the program is running on an operating system where a sleep of 1 second is long enough to complete all the other tasks currently running.

Please, write the answer on a single line, indicating the various output messages/values separated by a single space. Do not enter any other character in the response.

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

int main(){
      int x;
      x=0;
      while (x<2 && fork()){
            if (!fork())
                  execlp ("echo", "x++", "x", NULL);
            x++;
            sleep (1);
            system("echo x+x");
      }
}
```

Risposta. Answer.

```
x x+x x x+x
```

Ex 3 (2.0 points)

Italiano

Si indichino quali delle seguenti affermazioni sono vere, relativamente alla sequenza di comandi di shell sotto riportata. Si osservi che risposte errate implicano una penalità nel punteggio finale.

```
find / -name "a?b.c" -exec cat \{} \; | tail -80 | head > /tmp/x
cat /tmp/x | sort -r | tr "e-g" "E-G"
```

English

Indicate which statements related to the following sequence of shell commands are correct. Note that wrong answers imply a penalty in the final score.

```
find / -name "a?b.c" -exec cat \{} \; | tail -80 | head > /tmp/x
cat /tmp/x | sort -r | tr "e-g" "E-G"
```

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

1. ✅ Il comando "sort -r" prende in input il contenuto del file /tmp/x e genera in output lo stesso contenuto ordinato in ordine alfabetico invertito The command "sort -r" takes in input the content of the file /tmp/x, and it generates in output the same content but ordered in reverse alphabetical order
2. ⬜ Il file /tmp/x contiene le prime 10 righe dei file selezionati dal comando find The file /tmp/x contains the first 10 rows of the files selected by the find command
3. ⬜ Il comando find cerca tutti i file che hanno un nome che inizia con la lettera "a", finisce con le lettere "b.c" ed è composto da un numero qualsiasi di caratteri The find command searches all the files which name starts with the letter "a", finishes with letters "b.c", and it is composed of any number of characters
4. ⬜ Il comando tr "e-g" "E-G" sostituisce le lettere e, g minuscole con le corrispettive lettere maiuscole (cioè E, G) The command tr "e-g" "E-G" substitutes the lowercase letters e, g with the corresponding uppercase letters (i.e., E, G)
5. ⬜(also ✅) Il comando find cerca i file nella home directory The find command searches the files in the home directory
6. ✅ Il comando find cerca tutti i file che hanno un nome che inizia con la lettera "a", finisce con le lettere "b.c" ed è composto da 5 caratteri The find command searches all the files which name starts with the letter "a", finishes with letters "b.c", and it is composed of 5 characters

Ex 4 (3.5 points)

Italiano

Lo stato di un sistema con 4 processi e 3 tipologie di risorse è definito dalle matrici riportate alla fine della domanda.

Si esegua l'algoritmo del banchiere e si riporti il vettore "Available" che si ottiene a seguito dell'esecuzione dell'algoritmo.

Riportare la soluzione come tre numeri separati da spazi.

English

The state of a system with 4 processes and 3 types of resources is defined by the matrices reported at the end of the question.

Execute the Banker's Algorithm, and report the vector "Available" that is obtained after the executing of the algorithm.

Report the solution as three numbers separated by spaces.

Assegnate **Allocation**	**Massimo** **Max**	**Disponibile** **Available**
0 0 1	0 3 1	2 1 1
0 1 1	4 2 2	

0 0 1	1 1 1	
0 3 0	2 3 2	

Risposta. Answer.

2 4 3

Ex 5 (2.0 points)

Italiano

Si supponga che il disco rigido di un piccolo sistema embedded sia costituito da 24 blocchi di 1 MByte, che tali blocchi siano numerati da 0 a 23, che il sistema operativo mantenga 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

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 24 blocks of 1 MByte each, which are numbered from 0 to 23. 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

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 3.5 MByte NON può essere allocato utilizzato una strategia di allocazione contigua A file with dimension 3.5 MByte CANNOT be allocated using a contiguous allocation strategy
2. ⬜ Un file di dimensione 3.5 MByte NON può essere allocato utilizzato una strategia di allocazione concatenata A file with dimension 3.5 MByte CANNOT be allocated using a linked allocation strategy
3. ✅ 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 1.9 MByte e F3 di 2.6 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 1.9 MByte, and F3 of 2.6 MByte
4. ⬜ 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 1.9 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 1.9 MByte, and F3 of 2.6 MByte

Ex 6 (1.5 points)

Italiano

Si supponga un processo diventi "zombie".

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

English

Suppose that a process becomes "zombie".

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 è stato cancellato The PCB of the process was deleted
2. ⬜ Il processo ha effettuato una wait The process performed a wait
3. ⬜ Il processo non ha processo padre The process has not a parent process
4. ✅ Il processo è terminato The process is terminated
5. ⬜ Il processo prima di terminare aveva quale processo padre il processo "init" The process before terminating had the "init" process as parent process

6. ⬜ Il processo è stato ereditato dal processo "init" The process was inherited by the process “init”
7. ✅ Il suo PCB verrà cancellato solo dopo che il padre effettuerà una wait o una waitpid Its PCB will be deleted only after its parent performs a wait or a waitpid

Ex 7 (1.5 points)

Italiano

Si faccia riferimento alla system call e al comando di shell kill.
Si indichino quali delle seguenti affermazioni sono vere. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Refer to the system call kill and to the kill shell command.
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 segnale inviato può essere mascherato solo con la system call The sent signal can be masked only in the case of the system call
2. ✅ Tanto il comando di shell quando la system call possono uccidere un processo Both the shell command and the system call can kill a process
3. ⬜ La system call uccide un un processo, il comando di shell no The system call kills a process, the shell command does not
4. ✅ Entrambe inviano un segnale a un processo Both send a signal to a process
5. ⬜ Il comando di shell uccide un processo, la system call no The shell command kills a process, the system call does not

Ex 8 (1.5 points)

Italiano

Un programma multi-thread è costituito da diversi thread. Il thread A prima di terminare effettua una exit(), il thread B una pthread_exit() e il thread C una return.
Si indichino quali delle seguenti affermazioni sono vere. Si osservi che risposte errate implicano una penalità nel punteggio finale

English

A multi-threaded program consists of several threads. Thread A executes an exit() before terminating, thread B a pthread_exit(), and thread C a return.
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. ⬜ Per non far terminare gli altri thread la exit() deve essere effettuata solo dalla funzione iniziale del thread (ad esempio main()) In order not to terminate the other threads, the exit() must be performed only by the initial function of the thread (e.g., main())
2. ⬜ Tutti gli altri thread terminano sicuramente insieme al thread B All other threads end with thread B
3. ⬜ Tutti gli altri thread terminano sicuramente insieme al thread C All other threads end with thread C
4. ✅ Per poter terminare il thread C deve effettuare la return dalla sua funzione iniziale del thread (ad esempio main()) In order to terminate, thread C must perform the return from its initial function of the thread (e.g., main())
5. ✅ Tutti gli altri thread terminano sicuramente insieme al thread A All other threads end with thread A

Ex 9 (1.5 points)

Italiano

Relativamente all’implementazione delle pipe in un sistema operativo Unix, si indichino quali delle seguenti affermazioni sono vere. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Regarding the implementation of pipes in a Unix operating system, 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. ✅ La scrittura su una pipe in cui tutti gli estremi di lettura sono stati chiusi provoca la generazione di un segnale di tipo SIGPIPE The write operation on a pipe, in which all reading extremes have been closed, causes the generation of a SIGPIPE signal
2. ⚪ Il tentativo di scrittura su una pipe piena restituisce errore The write attempt on a full pipe returns an error
3. ✅ Il tentativo di scrittura su una pipe piena è normalmente bloccante The write attempt on a full pipe is normally blocking
4. ✅ Una costante indica quanti byte possono essere scritti in una pipe in modo atomico A constant indicates how many bytes can be atomically written on a pipe
5. ⚪ L'operazione di scrittura su una pipe è sempre eseguita in modo atomico The write operation on a pipe is always performed atomically
6. ⚪ La lettura su una pipe in cui tutti gli estremi di scrittura sono stati chiusi provoca la generazione di un segnale di tipo SIGPIPE The read operation on a pipe, in which all writing extremes have been closed, causes the generation of a SIGPIPE signal

Ex 10 (1.5 points)

Italiano

Si indichino quali delle seguenti affermazioni sono vere relativamente alla legge di Amdhal. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

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

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

1. ✅ Lo speedup ottenibile aumentando il numero di core per eseguire parallelamente un algoritmo è limitato asintoticamente The speedup that can be obtained by increasing the number of cores for the execution of an algorithm in a parallel way is limited asymptotically
2. ⚪ La legge di Amdhal assume che sia possibile uno speedup lineare rispetto al numero di core Amdhal's law assumes that a linear speedup with respect to the number of cores is possible
3. ✅ Come conseguenza alla legge di Amdhal, incrementare la parte parallelizzabile di un algoritmo è molto più importante che aumentare il numero di core As a consequence of Amdhal's law, increasing the parallelizable part of an algorithm is much more important than increasing the number of cores
4. ⚪ Raddoppiando il numero di core lo speedup ottenibile per eseguire parallelamente un algoritmo raddoppia With twice the number of cores for the execution of an algorithm in a parallel way, also the speedup is twice
5. ✅ La legge di Amdhal assume che la dimensione del problema da parallelizzare rimanga costante Amdhal's law assumes that the dimension of the problem to be parallelized remains constant

Ex 11 (2.5 points)

Italiano

Si consideri il seguente insieme di processi schedulati con un quantum temporale di 10 unità di tempo.
Rappresentare mediante diagramma di Gantt l'esecuzione di tali processi utilizzando l'algoritmo Round Robin (RR), al fine di calcolare il tempo di terminazione di ciascun processo e il tempo di attesa medio.
Si prega di riportare la risposta su un'unica riga, indicando i tempi di terminazione di P1, P2, P3, P4 e P5 seguiti dal tempo di attesa medio. Separare i numeri con un unico spazio. Riportare il tempo di attesa medio con 1 sola cifra decimale. Non inserire nessun altro carattere nella risposta.
Esempio di risposta corretta: 20 23 11 45 67 30.5

English

Consider the following set of processes, which are scheduled with a temporal quantum of 10 units.
Represent using a Gantt diagram the execution of these processes using the Round Robin (RR) scheduling algorithm, in order to compute the termination time of each process, and the average waiting time.
Please, write your answer on a single line, indicating the termination times of P1, P2, P3, P4 and P5 followed by the average waiting time. Separate the numbers with a single space. Report the average waiting time with a

single decimal digit. Do not enter any other character in the response. Example of correct answer: 20 23 11 45 67 30.5

Processo Process	TempoArrivo ArrivalTime	BurstTime	Priorità Priority
P1	0	23	5
P2	5	17	1
P3	10	15	4
P4	11	21	3
P5	13	19	2

Fasi operative. Workflow.

```
RR:
  P1: from=  0 to  =  9 end = 10 waitingTime =  0
  P2: from= 10 to  = 19 end = 20 waitingTime =  5
  P1: from= 20 to  = 29 end = 30 waitingTime = 10
  P3: from= 30 to  = 39 end = 40 waitingTime = 20
  P4: from= 40 to  = 49 end = 50 waitingTime = 29
  P5: from= 50 to  = 59 end = 60 waitingTime = 37
  P2: from= 60 to  = 66 end = 67 waitingTime = 40
  P1: from= 67 to  = 69 end = 70 waitingTime = 37
  P3: from= 70 to  = 74 end = 75 waitingTime = 30
  P4: from= 75 to  = 84 end = 85 waitingTime = 25
  P5: from= 85 to  = 93 end = 94 waitingTime = 25
  P4: from= 94 to  = 94 end = 95 waitingTime =  9
  EndTime: 70 67 75 95 94
  AverageWaitingTime: [(  0 + 10 + 37 ) + (  5 + 40 ) + ( 20 + 30 ) + ( 29 + 25 +  9
) + ( 37 + 25 ) ] / 5 = 53.40
  AverageTournaroudTime: ( 70 + 62 + 65 + 84 + 81 ) / 5 = 72.40
  GanttChart:
11111111112222222221111111111333333333344444444445555555555222222211133333444444444
45555555554
```

Risposta. Answer.

```
70 67 75 95 94 53.4
5 67 72 95 94 53.0
75 67 72 95 94 53.8
```

Ex 12 (2.5 points)

Italiano

Sia dato il seguente script e lo si supponga incluso nel file di nome script.sh:

```
#!/bin/bash
echo "Number: $#"
for x in $* do
      echo -n $x" "
done
```

Si supponga inoltre esso venga invocato con il seguente comando:

```
./script.sh "questa frase - $0 :" ha w parole e '$1' x
```

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

English

Assume that the following script is included in the file named script.sh:

```
#!/bin/bash echo "Number: $#" for x in $* do echo -n $x" " done
```

Also suppose it is invoked with the following command:

```
./script.sh "questa frase - $0 :" ha w parole e '$1' x
```

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. ⬜ L'output contiene la stringa "$0" The output contains the string "$0"
2. ⬜ L'output occupa più di 2 righe The output has a length of more than 2 lines
3. ✅ L'output contiene la stringa "$1" The output contains the string "$1"
4. ⬜ L'output contiene la stringa "Number: 6" The output contains the string "Number: 6"
5. ✅ L'output contiene la stringa "Number: 7" The output contains the string "Number: 7"
6. ✅ L'output occupa 2 righe The output has a length of 2 lines
7. ⬜ Il valore $1 viene sostituito con 0 The value $1 is substituted with 0

Ex 13 (5.0 points)

Italiano

Scrivere uno script bash capace di mostrare il nome del processo che contiene il massimo numero di caratteri. Si ricordi che l'output del comando

```
ps -e
```

è simile al seguente:

```
1564 ? 00:00:00 update-notifier
1673 ? 00:00:11 gnome-terminal
1681 pts/0 00:00:00 bash
1696 pts/1 00:00:00 bash
1710 pts/2 00:00:00 bash
1724 pts/3 00:00:00 bash
1923 pts/1 00:00:05 emacs
2004 ? 00:00:00 kworker/0:1-events
2005 ? 00:00:00 kworker/u2:0-events_unbound
2044 pts/2 00:00:00 less
```

In questo caso il nome di processo più lungo è "kworker/u2:0-events_unbound" perché è composto da 27 caratteri.

English

Write a bash script able to show the process name that contains the maximum number of characters. Please, remind that the output of the command

```
ps -e
```

is similar to the following one:

```
1564 ? 00:00:00 update-notifier
1673 ? 00:00:11 gnome-terminal
1681 pts/0 00:00:00 bash
1696 pts/1 00:00:00 bash
1710 pts/2 00:00:00 bash
1724 pts/3 00:00:00 bash
1923 pts/1 00:00:05 emacs
2004 ? 00:00:00 kworker/0:1-events
2005 ? 00:00:00 kworker/u2:0-events_unbound
2044 pts/2 00:00:00 less
```

In this case the longest process name is "kworker/u2:0-events_unbound" because it consists of 27 characters.

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

```
#!/bin/bash
# Version A
longest_name=""
max=0
for name in $(ps -e | tr -s " " | cut -d " " -f 4); do
      len=$(echo "$name" | wc -c)
      if [ $len -gt $max ]; then
            max=$len
```

```
            longest_name=$name
      fi
done
echo "The process with the longest name is $longest_name"

#!/bin/bash
# Version B
longest_name=""
max=0
ps -e > "tmp.txt"
while read line; do
      name=$(echo $line | tr -s " " | cut -d " " -f 4)
      len=$(echo "$name" | wc -c)
      if [ $len -gt $max ]; then
            max=$len
            longest_name=$name
      fi
done < "tmp.txt"
rm "tmp.txt"
echo "The process with the longest name is $longest_name"
```

Ex 14 (5.0 points)

Italiano

Un numero imprecisato di thread compete per l'utilizzo di un insieme di m risorse equivalenti fra loro e il cui numero è noto a priori.

Ogni thread esegue la funzione

```
int request(void)
```

per richiedere una risorsa. La funzione deve restituire l'indice i della risorsa assegnata al thread chiamante (dove i è un numero incluso tra 0 a m-1). La politica di assegnazione deve essere decisa dalla funzione request, la quale deve anche assicurare che ogni risorsa sia assegnata a un solo thread in ogni istante. La funzione request deve porre il chiamante in attesa nel caso in cui non vi sia nessuna risorsa immediatamente disponibile e risvegliarlo non appena una risorsa risulta assegnabile.

Ogni thread esegue la funzione

```
void release(int i)
```

per rilasciare la risorsa di indice i allocata in precedenza.

Si realizzino le funzioni request e release usando i semafori POSIX, definendone le relative variabili condivise. Si ipotizzi che i thread non tentino di rilasciare risorse non ad essi allocate e che nessun thread (considerato individualmente) tenti di allocare più di m risorse senza rilasciarne alcuna.

English

An unspecified number of threads compete for the use of a set of m equivalent resources, whose number is known a priori.

Each thread executes the function

```
int request(void)
```

to request a resource. The function must return the index i of the resource assigned to the calling thread (where i is a number between 0 to m-1). The resources allocation policy must be decided by the request function, which must also ensure that each resource is assigned to only one thread at a time. The request function must block the caller if there is no resource immediately available, and wake it up as soon as a resource becomes available again. Each thread executes the function

```
void release(int i)
```

to release a previously allocated resource with index i.

Implement the request and release functions using POSIX semaphores, defining the related shared variables. Assume that threads do not attempt to release resources not allocated to them and that no thread (considered individually) attempts to allocate more than m resources without releasing any.

Soluzione. Solution.

```
#define M ...
```

```
int resources[M] = {0}; /* Vector of resources: 1 occupied, 0 free. Initially all
zeros. */

sem_t empty, mutex;
sem_init(&empty, M); /* Initially M resources available */
sem_init(&mutex, 1); /* Mutex to access resources shared vector */

int request(void){
      int i;
      sem_wait(&empty); /* If no resource available, it block the thread */
      sem_wait(&mutex);
      for(i=0; i<M; i++){ /* Search for a free resource */
            if (resources[i] == 0) {
                  resources[i] = 1;
                  break;
            }
      }
      sem_post(&mutex);
      return i;
}

void release(int i){
      sem_wait(&mutex);
      resources[i] = 0; /* Set the resource with index i as free */
      sem_post(&mutex);
      sem_post(&empty); /* Release the resource */
}
```