Esame Sistemi Operativi - Operating Systems Exam
2025/09/17

Ex 01 (2.0 points)

Italiano

Si indichino quali delle seguenti affermazioni sono corrette relativamente ai file system. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Considering a file system, select all the correct statements from the following. Note that incorrect answers imply a penalty in the final score.

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

1. ⬜ Il file system a grafo di tipo ciclico non permette la condivisione di elementi. ----- The filesystem of type cyclic graph does not allow sharing of entries.
2. ✅ Le system call read() e write() possono essere utilizzate per leggere e scrivere testo, caratteri e sequenze di byte. ----- The system calls read() and write() can be used to read and write text, characters, and sequences of bytes.
3. ⬜ La system call open() ha esattamente tre parametri. ----- The system call open() has exactly three parameters.
4. ⬜ La system call read() ritorna il numero di oggetti letti, come nel caso della funzione fread(). ----- The system call read() returns the number of read objects, as in the case of the fread() function.
5. ✅ Quando un file viene creato attraverso la funzione open(), contestualmente si possono impostare i permessi del file creato. ----- When a file is created using the open() system call, the permissions of the created file can be set at the same time.

Ex 02 (3.5 points)

Italiano

Si supponga che il seguente programma venga eseguito mediante l'istruzione `./pgrm 3`.

Si indichino quante volte viene visualizzato il carattere "A" e quante volte il carattere "B" in quest'ordine (ovvero si riportino due interi separati da un singolo spazio). Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Suppose the following program is executed by means of the instruction `./pgrm 3`.

Report how many times the character "A" and the character "B" are displayed in that order (i.e., report two integers separated by a single space). Note that incorrect answers imply a penalty in the final score.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>

int main (int argc, char *argv[]) {
  int i;
  char str[50];

  i = atoi (argv[1]);
  fflush (stdout);

  if (i<=0) exit (0);

  if (fork () > 0) {
    sprintf (str, "echo A");
    system (str);
```

```
    sprintf (str, "%d", i-1);
    execlp (argv[0], argv[0], str, NULL);
  } else {
    sprintf (str, "echo B");
    system (str);
    sprintf (str, "%d", i-2);
    execlp (argv[0], argv[0], str, NULL);
  }

  exit (0);
}
```

Soluzione. Solution.
4 4

Ex 03 (2.0 points)

Italiano

Si supponga un processo diventi "orfano".
Si indichi quali delle seguenti affermazioni sono corrette. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Suppose a process becomes an "orphan."
Please select all the correct statements from the following. Note that incorrect answers imply a penalty in the final score.

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

1. ⬜ Il processo sta attendendo che il padre effettui una wait. ----- The process is waiting that the parent performs a wait.
2. ⬜ Il processo è diventato orfano in quanto non ha effettuato una wait. ----- The process becomes an orphan because it did not perform a wait.
3. ⬜ Il processo diventerà "zombie" alla sua terminazione. ----- The process will become "zombie" at its termination.
4. ✅ Il processo viene ereditato dal processo "init". ----- The process is inherited by the "init" process.
5. ✅ Un processo orfano continua la sua esecuzione normalmente dopo essere stato adottato. ----- An orphaned process continues its execution normally after being adopted.
6. ⬜ Un processo orfano non può più creare processi figli. ----- An orphaned process can no longer create child processes.
7. ✅ Il processo non diventerà "zombie" alla sua terminazione perché il processo "init" lo erediterà. ----- The process will not become "zombie" at its termination because the "init" process will inherit it.

Ex 04 (2.0 points)

Italiano

Nell'utilizzo di una pipe di nome fd, lo scrittore effettua la seguente operazione

```
write (fd[1], "Hello world!", 12);
```

mentre il lettore effettua la lettura con il codice successivo:

```
char c, s[100];
int i=0;
while ((n = read (fd[0], &c, 1)) > 0) {
  s[i] = c;
  i++;
}
s[i] = '\n';
print ("%s", s);
```

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

English

Using a pipe named fd, the writer does the following operation:

```
write (fd[1], "Hello world!", 12);
```

while the reader reads with the following code:

```
char c, s[100];
int i=0;
while ((n = read (fd[0], &c, 1)) > 0) {
  s[i] = c;
  i++;
}
s[i] = '\n';
print ("%s", s);
```

Please identify all correct statements below. Note that incorrect answers imply a penalty in the final score.

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

1. ⬜ Affinché l'operazione sia corretta e la stampa venga effettuata, lo scrittore deve inserire nella pipe il carattere '\n'. ----- To have a correct execution and printing, the writer must insert the character '\n' in the pipe.
2. ⬜ Dato che la lettura avviene a caratteri anche la scrittura deve avvenire a caratteri. ----- Since read is performed character-by-character, the write operation must also be performed character-by-character.
3. ⬜ Il lettore stampa correttamente la stringa s = "Hello world!". ----- The reader correctly prints the string s = "Hello world!".
4. ✅ Il lettore rimane bloccato sulla read a tempo indefinito una volta letto tutto il messaggio. ----- The reader remains blocked on the read indefinitely after the entire message has been read.
5. ⬜ Affinché la stampa avvenga correttamente la lettura deve avvenire con formato while (read (fd[0], str, 12)>0). ----- To have a correct printing, the read operation must be performed with the format while (read (fd[0], str, 12)>0).
6. ✅ Il lettore non stampa nulla. ----- The reader does not print anything.

Ex 05 (2.0 points)

Italiano

Dato il seguente elenco di comandi, si indichino quali delle seguenti affermazioni sono corrette dopo averli eseguiti. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Given the following list of commands, indicate which of the following statements are correct after executing them. Note that incorrect answers imply a penalty in the final score.

```
echo -n "hello_world" > f1
ln f1 f2
ln f1 f3
ln -s f1 f4
rm f1
echo "hello" > f4
echo "_world" >>  f2
```

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

1. ⬜ Il file f1 è stato rimosso e quindi non è presente nella directory. ----- File f1 was removed, and therefore it is not present in the directory.
2. ✅ I file f2 e f3 hanno lo stesso contenuto. ----- Files f2 and f3 have the same content.
3. ⬜ Dopo il comando "`ln f1 f3`" l'i-node del file f1 ha un counter uguale a 2. ----- After the command "`ln f1 f3`" the file f1 has a counter equal to 2.
4. ✅ La dimensione del file f4 è di 2 bytes. ----- The dimension of file f4 is 2 bytes.
5. ✅ Al termine della sequenza di istruzioni, il file f3 contiene "hello_world_world". ----- After all instructions, file f3 stores "hello_world_world".

6. ⬜ Al termine della sequenza di istruzioni, il file f4 contiene un puntatore al file f1 che però non esiste più. ----- After all instructions, file f4 stores the pointer to f1 that does not exist anymore..
7. ✅ Il numero di hard link del file f2 è pari a 2. ----- The number of hard links of file f2 is 2.
8. ⬜ Il file f2 contiene la stringa “hello_world”. ----- File f2 contains the string “hello_world”.

Ex 06 (2.0 points)

Italiano

Dire quali delle seguenti affermazioni relative ai threads sono corrette. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Considering threads, identify all correct statements from the following. Note that incorrect answers imply a penalty in the final score.

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

1. ✅ Per comunicare i thread possono usare le variabili globali. ----- Threads can use global variables to communicate.
2. ✅ La funzione pthread_join() attende la terminazione di uno specifico thread il cui identificatore è passato come parametro alla funzione. ----- The function pthread_join() waits the termination of a specific thread whose identifier is passed as a parameter to the function.
3. ⬜ Ogni thread con l’eccezione del thread con tid=0 ha un padre. ----- Each thread with the exception of the thread with tid=0 has a parent.
4. ⬜ L’esecuzione parallela di un algoritmo per mezzo di più di un thread porta SICURAMENTE alla riduzione del suo tempo di esecuzione. ----- The parallel execution of an algorithm by means of more than one thread leads IN ANY CASE to a reduction on its execution time.
5. ✅ La funzione pthread_exit() eseguita da un thread porta alla terminazione del solo thread che la esegue. ----- The function pthread_exit() executed by one thread leads to the termination of the only thread that executes it.
6. ⬜ L’istruzione return eseguita da un thread porta SICURAMENTE alla terminazione dell’intero processo. ----- The instruction return executed by one thread leads IN ANY CASE to the termination of the whole process.

Ex 07 (4.0 points)

Italiano

Si descrivano il significato e l’utilizzo delle system call signal() e kill().
Che cosa è una corsa critica (race condition)?
Quali sono le funzioni principali della system call signal(SIGCHLD, SIG_IGN)? Che relazione ha con la system call wait()?

English

Describe the meaning and use of the signal() and kill() system calls.
What is a race condition?
What are the main functions of the signal(SIGCHLD, SIG_IGN) system call? How does it relate to the wait() system call?

Soluzione. Solution.

Un segnale è un interrupt software, i.e., un evento di sistema inviato a un processo. I segnali permettono di gestire eventi asincroni. Non sono interrupt perché gli interrupt sono inviati dall’hardware al sistema operativo. I segnali sono inviati a processi da altri processi.
Sono utilizzati per notificare il verificarsi di eventi particolari: condizioni di errore, violazioni di segmentazione di memoria, errori floating point o istruzioni illegali, etc.
Per esempio ctrl-C invia un segnale INT (SIGINT) e ctrl-Z invia un segnale `TSTP` (`SIGTSTP`).
Una race condition avviene quando il comportamento di più processi che lavorano su dati comuni dipende dall’ordine di esecuzione. La segnalazione tra processi può generare race conditions che portano il programma a non funzionare nel modo desiderato.

```
#include <signal.h>
```

```
void (*signal (int sig, void (*func)(int)))(int);
```
Il file signal.h definisce i nomi dei segnali (i.e., `SIGABORT` Process abort, `SIGALARM` Alarm clock, etc.).
La signal consente di settare un signal handler. Ha due parametri: sig che specifica il segnale da intercettare e func che indica la funzione da invocare. func ha un solo parametro di tipo int (il segnale ricevuto).
Quando un segnale si presenta è possibile: (a) Ignorare esplicitamente il segnale (tale comportamento è impossibile per i segnali `SIGKILL` e `SIGSTOP` che non possono essere ignorati), (b) Lasciare che si verifichi il comportamento di default, (c) Catturare il segnale.

```
#include <signal.h>
int kill (pid_t pid, int sig);
```
Invia un segnale a un processo o a un gruppo di processi.
E' possibile spedire segnali solo a processi con lo stesso UID.

La system call `signal (SIGCHLD, SIG_IGN)` permette a un processo di ignorare (costante `SIG_IGN`) i segnali `SIGCHLD` inviati dai figli che terminano. Questo è anche il comportamento di default. Se un processo vuole essere notificato della terminazione dei figli (ed evitare che diventino "zombie") deve utilizzare la system call wait().

A signal is a software interrupt, i.e., a system event sent to a process. Signals allow the management of asynchronous events. They are not interrupts because interrupts are sent from hardware to the operating system. Signals are sent to processes by other processes.
They are used to notify the occurrence of specific events: error conditions, memory segmentation violations, floating point errors, or illegal instructions, etc.
For example, ctrl-C sends an `INT (SIGINT)` signal, and ctrl-Z sends a `TSTP (SIGTSTP)` signal.
A race condition occurs when the behavior of multiple processes working on common/shared data depends on the order of execution. Signaling between processes can generate race conditions that cause the program to not function as intended.

```
#include <signal.h>
void (*signal (int sig, void (*func)(int)))(int);
```
The signal.h file defines the signal names (e.g., `SIGABORT` Process abort, `SIGALARM` Alarm clock, etc.).
The signal allows the setting of a signal handler. It has two parameters: sig, which specifies the signal to capture, and func, which indicates the function to invoke. func has a single int parameter (the received signal).
When a signal occurs, you can: (a) Explicitly ignore the signal (this behavior is impossible for the `SIGKILL` and `SIGSTOP` signals, which cannot be ignored), (b) Allow the default behavior to occur, or (c) Catch the signal.

```
#include <signal.h>
int kill (pid_t pid, int sig);
```
Sends a signal to a process or group of processes.
Signals can only be sent to processes with the same UID.

The `signal (SIGCHLD, SIG_IGN)` system call allows a process to ignore (constant `SIG_IGN`) `SIGCHLD` signals sent by terminating children. This is also the default behavior. If a process wants to be notified of the termination of its children (and prevent them from becoming "zombies"), it must use the wait() system call.

Ex 08 (5.0 points)

Italiano

Realizzare uno script bash che riceva come unico argomento un file di testo. Lo script deve:

- effettuare una copia del file in un file con lo stesso nome ma con estensione xyx (es. "hello.txt" diventa "hello.xyx")
- modificare il file originario come segue:
    - all'inizio di ogni riga aggiungere il numero di parole della riga e il numero di righe totali del file
    - ordinare le righe in ordine crescente in base al numero di parole.

English

Create a bash script that receives a text file as its only argument. The script must:

- make a copy of the file with the same name but with the extension xyx (i.e. "hello.txt" becomes "hello.xyx")
- modify the original file as follows:
  - at the beginning of each line add the number of words in the line and the total number of lines in the file
  - sort the lines in ascending order based on the number of words.

Soluzione. Solution.

```
#!/bin/bash
if [ $# -ne 1 ]
then
  echo "usage $0 file.txt"
  exit 1;
fi
newfilename=$(basename $1 ".txt")
newfilename=$newfilename".xyx"
cat $1 > $newfilename # or cp $1 $newfilename
nlines=`cat $1|wc -l`
rm -f tmp1.txt
while read line
do
  nwords=`echo $line | wc -w`
  echo $nwords $nlines $line >> tmp1.txt
done < $1
cat tmp1.txt | sort -k 1 -n > $1
rm tmp1.txt
```

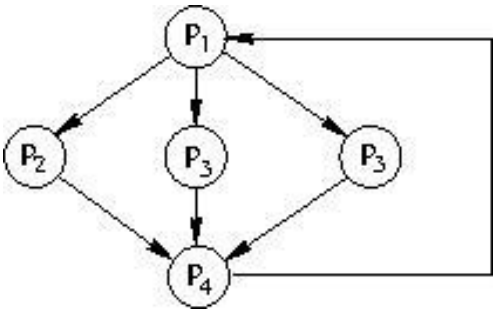
Ex 09 (5.5 points)

Italiano

Realizzare il seguente grafo di precedenza, utilizzando il minimo numero di semafori. I processi rappresentati devono essere processi ciclici (con corpo del tipo while(1)). Utilizzare le primitive init, signal, wait e destroy. Indicare gli eventuali archi superflui e riportare il corpo dei processi (P1, ..., P4) e l'inizializzazione dei semafori. Si osservi che il grafo prevede l'esecuzione di 5 processi tra i quali ci sono **due istanze identiche** dello stesso processo P3.

English

Implement the following precedence graph, using the minimum number of semaphores. The processes represented must be cyclical (with a body of type while(1)). Use the primitives init, signal, wait, and destroy. Indicate any unnecessary arcs and report the body of the processes (P1, ..., P4), and the initialization of the semaphores. Observe that the graph involves the execution of 5 processes, among which there **are two identical instances** of the same process P3.



Soluzione. Solution.

```
init (s1, 1);
init (s2, 0); init (s3, 0); init (s4, 0);
```

```
init (s, 0); n = 0;

P1
while (1) {
  wait (s1);
  printf ("P1\n");
  signal (s2);
  signal (s3);
}

P2
while (1) {
  wait (s2);
  printf ("P1\n");
  signal (s4);
}

P3
while (1) {
  wait (s3);
  n++;
  if (n==1)
    signal (s3);
  else {
    n = 0;
    signal (s);
    signal (s);
  }
  wait (s);
  printf ("P3\n");
  signal (s4);
}

P4
while (1) {
  for (i=0; i<3; i++)
    wait (s4);
  printf ("P4\n");
  signal (s1);
}


destroy(s1);
destroy(s2);
destroy(s3);
destroy(s4);
```

Ex 10 (2.0 points)

Italiano

Dato il seguente codice, si indichino quali delle seguenti affermazioni sono corrette. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Given the following code, indicate which of the following statements are correct. Note that incorrect answers imply a penalty in the final score.

```
1 : init (full, 0); init (empty, SIZE); init (mutex, 1); int in=0, out=0,
buff[SIZE];
2 : void push (int val) {
3 :   wait (empty);
4 :   wait (mutex);
5 :   buff[in]=val;
6 :   in=(in+1)%SIZE;
7 :   signal (mutex);
8 :   signal (full);
9 : }
10: int pop () {
11:   int res;
12:   wait (full);
13:   wait (mutex);
14:   res=buff[out];
15:   out=(out+1)%SIZE;
16:   signal (mutex);
17:   signal (empty);
18:   return(res);
19: }
```

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

1. ✅ L'inversione dell'ordine delle due istruzioni signal (signal(mutex) and signal(empty)) in linee 16 e 17 non modifica sostanzialmente il comportamento di questo codice. ----- Inverting the order of the two signal instructions (signal(mutex) and signal(empty)) in lines 16 and 17 does not substantially modify the behavior of this code.
2. ✅ L'uso di due mutex diversi per le due funzioni (usando cioè wait(mutex1) e signal(mutex1) per push(), e wait(mutex2) e signal(mutex2) per pop()) incrementa il livello di concorrenza. ----- The use of two different mutexes for the two functions (i.e., using wait(mutex1) and signal(mutex1) for push(), and wait(mutex2) and signal(mutex2) for pop()) increases the level of concurrency.
3. ⬜ Questo codice risolve il problema dei lettori e degli scrittori. ----- This code solves the problem of readers and writers.
4. ⬜ L'istruzione wait(empty); è utilizzata per bloccare i processi quando il buffer è vuoto. ----- The instruction sem_wait(empty); is used to block processes when the buffer is empty.
5. ✅ L'inversione dell'ordine delle due istruzioni wait (wait(empty) and wait(mutex)) in linee 3 e 4 può generare stallo/deadlock. ----- Inverting the order of the two wait instructions (wait(empty) and wait(mutex)) in lines 3 and 4 may lead to deadlock.

Ex 11 (3.0 points)

Italiano

Si consideri il seguente insieme di processi schedulati con un quantum temporale di 15 unità di tempo. Rappresentare, mediante diagramma di Gantt, l'esecuzione di tali processi utilizzando l'algoritmo Shortest Remaining Time First (SRTF). Si calcolino i tempi di terminazione di ciascun processo e il tempo di attesa medio. Si riporti 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 numeri e stringhe 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 processes, scheduled with a temporal quantum of 15 units. Using a Gantt diagram, represent the execution of these processes using the Shortest Remaining Time First (SRTF) scheduling algorithm. Compute the termination time of each process and the average waiting time. Report your answer on a single line, indicating P1, P2, P3, P4, P5, and P6 termination times, 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	15	4
P2	0	13	3
P3	3	12	5
P4	3	11	2
P5	4	6	1

Fasi operative. Workflow.

```
SRTF:
  P2: from=  0 to  =  3 end =  4 waitingTime =  0
  P5: from=  4 to  =  9 end = 10 waitingTime =  0
  P2: from= 10 to  = 18 end = 19 waitingTime =  6
  P4: from= 19 to  = 29 end = 30 waitingTime = 16
  P3: from= 30 to  = 41 end = 42 waitingTime = 27
  P1: from= 42 to  = 56 end = 57 waitingTime = 42
  EndTime: 57 19 42 30 10
  AverageWaitingTime: [( 42 ) + (  6 ) + ( 27 ) + ( 16 ) + (  ) ] / 5 = 18.20
  AverageTournaroudTime: ( 57 + 19 + 39 + 27 +  6 ) / 5 = 29.60
  GanttChart: 22225555552222222224444444444333333333333111111111111111
```

Risposta. Answer.

```
57 19 42 30 10 29.60
```

Ex 12 (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 {1, 1, 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 {1, 1, 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 Process	Assegnate Allocation	Massimo Max	Disponibile Available
P1	0 0 1	4 4 2	3 2 3
P2	0 2 0	3 4 2	
P3	0 1 0	3 2 2	
P4	0 0 0	4 3 5	
P5	1 0 1	2 5 5	

Fasi operative. Workflow.

```
---------------------------------------------------------
Check Request (process 3 - row 2)
Richiesta soddisfacibile; nuovo stato
  nProc  5
  nRis  3
  Processo Fine Assegnate Massimo Necessita' Disponibili
```

```
  |  1  |  0  |  0  0  1  |  4  4  2  |  4  4  1  |  1  1  2  |
  |  2  |  0  |  0  2  0  |  3  4  2  |  3  2  2  |
  |  3  |  0  |  2  2  1  |  3  2  2  |  1  0  1  |
  |  4  |  0  |  0  0  0  |  4  3  5  |  4  3  5  |
  |  5  |  0  |  1  0  1  |  2  5  5  |  1  5  4  |
  RequestByProcP  3 (row  2)
  RequestOf  2  1  1
------------------------------------------------------------
Soluzione Iterativa (una soluzione)
  Step  0;                                              Disponibili:  1  1  2
  Step:  1   RunProc:  3   Necessita':  1  0  1    -->    Disponibili:  3  3  3
  Step:  2   RunProc:  2   Necessita':  3  2  2    -->    Disponibili:  3  5  3
    Stato NON sicuro
```

Risposta. Answer.

NO