Esame Sistemi Operativi - Operating Systems Exam
2023/09/15

Ex 01 (4.0 points)

Italiano

Si supponga che il seguente programma venga eseguito con il valore 2 sulla linea di comando. Si riporti esattamente l'output generato. Si prega di riportare la risposta su un'unica riga, indicando i vari messaggi e valori di output separati da un unico spazio. Non inserire nessun altro carattere nella risposta.

English

Suppose the following program is executed, passing the value 2 in the command line. Report the exact output generated. Please, report the response in one line, indicating all the messages and the values in the output, which must be separated by a single space. Do not insert any other character in the response.

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

#define N 100

int main (int argc, char *argv[]) {
  int n;
  char str[N];
  n = atoi (argv[1]);
  setbuf(stdout,0);
  while (n>0 && !fork()) {
    fprintf (stdout, "F");
    if (fork()) {
      fprintf (stdout, "E");
      sprintf (str, "%d", n-1);
      execlp (argv[0], argv[0], str, NULL);
    } else {
      sprintf (str, "echo -n S");
      system (str);
    }
    n--;
  }
  return 1;
}
```

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

1. ⬜ FESFESFESFES
2. ✅ FESFESFES
3. ✅ FEFESFESS
4. ⬜ FESFESFFEESS
5. ✅ FESFFEESS

Ex 02 (4.0 points)

Italiano

Si supponga di eseguire il seguente programma. Quali sono le possibili righe di output generate? Si supponga un secondo sia un tempo molto lungo rispetto ai tempi di schedulazione del sistema operativo. Si indichino quali delle seguenti affermazioni sono corrette. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Suppose to run the following program. Which are the possible output lines displayed? Suppose one second is a very long time compared to standard scheduling operations. Indicate which of the following statements are correct. Note that incorrect answers imply a penalty in the final score.

```
#include <stdio.h>
#include <unistd.h>
#include <semaphore.h>
#include <pthread.h>

sem_t sa, sb, scd, sc, sd;
int n;

static void *TA ();
static void *TB ();
static void *TC ();
static void *TD ();

int main (int argc, char **argv) {
  pthread_t th;

  sem_init (&sa, 0, 1);
  sem_init (&sb, 0, 0);
  sem_init (&scd, 0, 0);
  sem_init (&sc, 0, 0);
  sem_init (&sd, 0, 0);

  setbuf(stdout, 0);
  n = 1;
  pthread_create (&th, NULL, TA, NULL);
  pthread_create (&th, NULL, TB, NULL);
  pthread_create (&th, NULL, TC, NULL);
  pthread_create (&th, NULL, TD, NULL);
  pthread_exit(0);
}

static void *TA () {
  pthread_detach (pthread_self ());
  while (1) {
    sem_wait (&sa);
    n++;
    printf ("A");
    if (n<=3) {
      sem_post (&sa);
    } else {
      n = 1;
      sem_post (&sb);
    }
  }
  return 0;
}

static void *TB () {
  pthread_detach (pthread_self ());
  while (1) {
    sem_wait (&sb);
    n++;
    printf ("B");
    if (n<=3) {
      sem_post (&sb);
```

```
    } else {
      n = 1;
      sem_post (&scd);
      sem_post (&sc);
      sem_post (&sd);
    }
  }
  return 0;
}

static void *TC () {
  pthread_detach (pthread_self ());
  while (1) {
    sem_wait (&scd);
    sem_wait (&sc);
    printf ("C");
    n++;
    if (n%2==0) {
      sem_post (&scd);
      sem_wait (&sc);
    } else {
      n = 1;
      printf ("\n");
      sleep (1);
      sem_post (&sd);
      sem_post (&sa);
    }
  }
  return 0;
}

static void *TD () {
  pthread_detach (pthread_self ());
  while (1) {
    sem_wait (&scd);
    sem_wait (&sd);
    printf ("D");
    n++;
    if (n%2==0) {
      sem_post (&scd);
      sem_wait (&sd);
    } else {
      n = 1;
      sleep (1);
      printf ("\n");
      sem_post (&sc);
      sem_post (&sa);
    }
  }
  return 0;
}
```

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

1. ⬜ AAABBBCC
2. ✅ AAABBBDC
3. ⬜ AAABBBDD
4. ✅ AAABBBCD

5. ⬜ AAABBBCB
6. ⬜ AABBCCDD
7. ⬜ ABCDABCD
8. ⬜ AABBCDCD

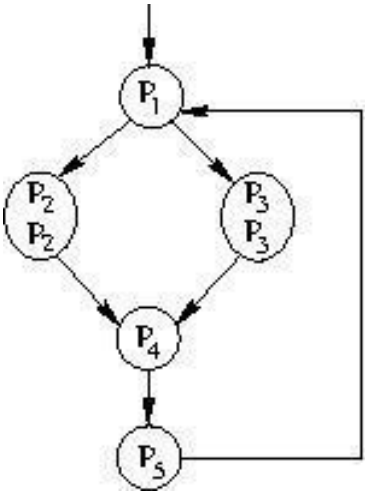
Ex 03 (5.0 points)

Italiano

Dato il seguente grafo di precedenza, realizzarlo utilizzando il minimo numero possibile 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 (P_1, ..., P_5) e l'inizializzazione dei semafori. Si noti che P_2 e P_3 devono effettuare due iterazioni in sequenza del proprio ciclo while per ogni iterazione globale sull'intero grafo di precedenza.

English

Given the following precedence graph, implement it using the minimum possible number of semaphores. The processes represented must be cyclical processes (with the body of type `while(1)`). Use the primitives `init`, `signal`, `wait` and `destroy`. Indicate any superfluous arcs and report the body of the processes (P_1, ..., P_5) and the initialization of the semaphores. Note that P_2 and P_3 must perform two sequential iterations of their while loop for each global iteration on the entire precedence graph.



Risposta: Answer:

```
// Solution 1
init(s1,1);
init(s2,0);
init(s3,0);
init(s4,0);
init(s5,0);
int n2=0;
int n3=0

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

P2
while(1){
 wait(s2);
 n2++;
 printf("P2\n");
```

```
 if(n2==1){
  signal(s2);
 }
 else{
  n2=0;
  signal(s4);
 }
}

P3
while(1){
 wait(s3);
 n3++;
 printf("P3\n");
 if(n3==1){
  signal(s3);
 }
 else{
  n3=0;
  signal(s4);
 }
}

P4
while(1){
 wait(s4);
 wait(s4);
 printf("P4\n");
 signal(s5);
}

P5
while(1){
 wait(s5);
 printf("P5\n");
 signal(s1);
}
destroy(s1);
destroy(s2);
destroy(s3);
destroy(s4);
destroy(s5);

// Solution 2
init(s1,1)
init(s2,0)
init(s3,0)
init(s4,0)
init(s5,0)

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

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

P3:
while(1){
  wait(s3);
  printf("P3\n");
  signal(s4);
}

P4:
while(1){
  signal(s4);
  signal(s4);
  signal(s4);
  signal(s4);
  printf("P4\n");
  signal(s5);
}

P5:
while(1){
  wait(s5);
  printf("P5\n");
  signal(s1);
}

destroy(s1)
destroy(s2)
destroy(s3)
destroy(s4)
destroy(s5)
```

Ex 04 (5.0 points)

Italiano

Realizzare uno script di shell BASH che riceva un parametro singolo sulla riga di comando e sia in grado di:

- Processare tutti i file contenuti nella directory corrente aventi come estensione quella data come parametro di ingresso (es., "`txt`"), cambiando tutte le stringhe "`printf`" contenute all'interno di ciascun file in "`write`".
- Rinominare i file con lo stesso nome, ma con estensione "`.new`".

English

Implement a shell script using BASH that receives a single parameter on the command line and is able to:

- Process all the files contained in the current directory with the extension provided as the input parameter of the script (e.g., "txt"). On each selected files, change its content to replace each "`printf`" strings into "`write`".
- Rename all the files with the same name but with the extension "`.new`".

Risposta: Answer:

```
#!/bin/bash

#######################################################################
```

```
# Version 1: find, string replace in pure bash rudimentary way
##########################################################################

# Check correct number of command line parameter is passed
if [ $# -lt 1 ]; then
	echo "Usage: ex.sh <FILE_EXTENSION>"
	exit
fi

# Find all files in the current directory with the given extension
find . -maxdepth 1 -type f -name "*.$1" > "list.tmp"
while read file; do

	# Compose new file name
	new_name=${file%.$1}.new

	# Replace each occurrence of strinf "printf" with "write". Mind
that this is a rudimentary way to do it (as it does not preserve the
original whitespaces)
	# see ex2.sh for a more elegant solution.
	while read line; do
	for word in $line; do
		if [ $word == "printf" ]; then
			echo -n "write"
		else
			echo -n $word
		fi
	done
	echo ""
	done < $file > $new_name

	# Remove original file
	rm $file

done < "list.tmp"

# Delete temporary file
rm "list.tmp"

#!/bin/bash

##########################################################################
# Version 2: glob expansion, string replace in pure bash more elegant way
##########################################################################

# Check correct number of command line parameter is passed
if [ $# -lt 1 ]; then
	echo "Usage: ex2.sh <FILE_EXTENSION>"
	exit
fi

# Find all files in the current directory with the given extension
for file in $(ls *.$1); do

	# Compose new file name
	new_name=${file%.$1}.new

	# Replace all instances on line of printf with write
	while read line; do
```

```
	echo ${line//printf/write}
	done < $file > $new_name

	# Remove original file
	rm $file
done

#!/bin/bash

##########################################################################
# Version 3: find, sed
##########################################################################

# Check correct number of command line parameter is passed
if [ $# -lt 1 ]; then
	echo "Usage: ex3.sh <FILE_EXTENSION>"
	exit
fi

# Find all files in the current directory with the given extension
find . -maxdepth 1 -type f -name "*.$1" > "list.tmp"
while read file; do

	# Replace all instances on line of printf with write
	sed -i '' s/printf/write/g $file

	# Rename file
	mv "$file" "${file%.*}.new"

done < "list.tmp"

#!/bin/bash

##########################################################################
# Version 4: find, exec one-liner (note the use of subshell)
##########################################################################

# Check correct number of command line parameter is passed
if [ $# -lt 1 ]; then
	echo "Usage: ex4.sh <FILE_EXTENSION>"
	exit
fi

# Replace all instances of printf with write in all files with the
# given extension in the current directory and then rename them
find . -maxdepth 1 -type f -name "*.$1" -exec sed -i '' s/printf/write/g
{} \; -exec sh -c 'file="{}"; mv "$file" "${file%.*}.new"' \;
```

Ex 05 (4.5 points)

Italiano

Si consideri il seguente insieme di processi schedulati con un quanto temporale di 10 unità di tempo. Rappresentare mediante diagramma di Gantt l'esecuzione di tali processi utilizzando gli algoritmi First Come First Served (FCFS), Shortest Job First (SJF) e Priority Scheduling (PS), al fine di calcolare il tempo di terminazione di ciascun processo e il tempo di attesa medio. Tra i tre algoritmi, si riporti lo scheduling **meno** vantaggioso dal punto di vista del 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 processes, scheduled with a temporal quantum of 10 units. Represent using a Gantt diagram the execution of these processes using the First Come First Served (FCFS), Shortest Job First (SJF), and Priority Scheduling (PS) algorithms, to compute the termination time of each process, and the average waiting time. Compare the three schedulings, and report only the **worst** one regarding 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	8	4
P2	3	12	1
P3	5	17	3
P4	10	13	5
P5	13	20	2

Solution

FCFS:

P1: start=0, end=7, waitingTime=0
P2: start=8, end=19, waitingTime=5
P3: start=20, end=36, waitingTime=15
P4: start=37, end=49, waitingTime=27
P5: start=50, end=69, waitingTime=37
AverageWaitingTime: 16.80
GanttChart: 1111111122222222222233333333333333333344444444444445555555555555555555

SJF:

P1: start=0, end=7, waitingTime=0
P2: start=8, end=19, waitingTime=5
P4: start=20, end=32, waitingTime=10
P3: start=33, end=49, waitingTime=28
P5: start=50, end=69, waitingTime=37
AverageWaitingTime: 16.00
GanttChart: 1111111122222222222244444444444443333333333333333335555555555555555555

PS:

P1: start=0, end=7, waitingTime=0
P2: start=8, end=19, waitingTime=5
P5: start=20, end=39, waitingTime=7
P3: start=40, end=56, waitingTime=35
P4: start=57, end=69, waitingTime=47
AverageWaitingTime: 18.80
GanttChart: 1111111122222222222255555555555555555555533333333333333333344444444444444

Risposta: Answer:
8 20 57 70 40 18.8

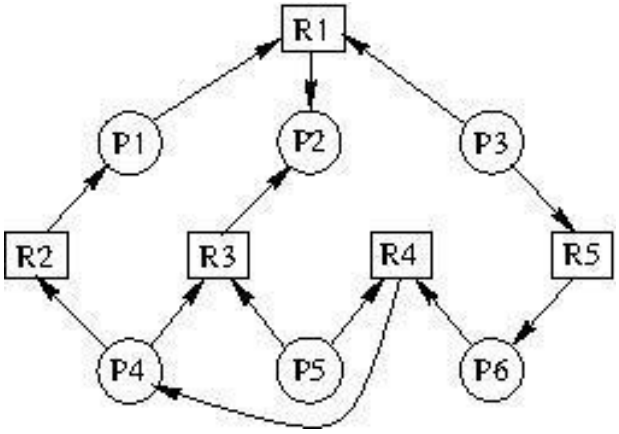
Ex 06 (3.5 points)

Italiano

Si consideri il seguente grafo di allocazione delle risorse. Se il sistema è in deadlock si risponda D. Se il sistema non è in deadlock si riportino i processi nell'ordine in cui possono terminare. Si osservi che risposte errate, anche solo a livello sintattico, implicano una penalità nel punteggio finale. La seguente è un esempio di risposta corretta: P1 P2 P3 P4 P5 P6.

English

Consider the following resource allocation graph. If the system is in deadlock, answer D. If the system is not in deadlock, report the processes in the order in which they can terminate. Note that incorrect answers, even if only syntactically, imply a penalty in the final score. The following is an example of a correct answer: P1 P2 P3 P4 P5 P6.



Risposta: Answer:

P2 P1 P4 P5 P6 P3

Ex 07 (2.5 points)

Italiano

Relativamente all'implementazione delle pipe in un sistema operativo UNIX/Linux, 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/Linux 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. ✅ Una pipe può essere condivisa solo tra processi con un parente comune. A pipe can only be shared between processes with a common parent.
2. ⬜ Essendo uno "pseudo-file", una pipe, una volta creata, viene individuata da un oggetto di tipo "`FILE *`". Being a "pseudo-file", a pipe, once created, is identified by an object of type "`FILE *`".
3. ⬜ Essendo uno "pseudo-file", una pipe, una volta creata, deve essere aperta con la system call `open`. Being a "pseudo-file", a pipe, once created, must be opened with the system call `open`.
4. ⬜ Le system call `read` e `write` non sono mai bloccanti su una pipe. The `read` and `write` system calls are never blocking on a pipe.
5. ⬜ Le funzioni `fscanf` e `fprintf` possono essere utilizzate per trasferire dati con una pipe. The functions `fscanf` and `fprintf` can be used to transfer data with a pipe.
6. ⬜ Nel flusso standard di utilizzo prima si effettua una `exec` quindi la system call `pipe`. In the standard utilization flow, first an `exec` is performed, then the system call `pipe`.
7. ✅ Nel flusso standard di utilizzo prima si effettua una `pipe` quindi la system call `fork`. In the standard utilization flow, first a `pipe` is performed, then the system call `fork`.
8. ✅ La dimensione di una pipe è limitata. The dimension of a pipe is limited.

Ex 08 (2.5 points)

Italiano

Dire quali delle seguenti risposte relative al deadlock sono vere. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Indicate which of the following statements referred to deadlock are correct. Note that incorrect answers imply a penalty in the final score.

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

1. ⬜ Una risorsa con possibilità di prelazione, quale ad esempio la CPU, può portare a deadlock. A resource with the possibility of preemption, such as the CPU, can lead to deadlock.
2. ⬜ Quando si verifica starvation si verifica anche un deadlock. When starvation occurs, a deadlock also occurs.
3. ✅ Quando si verifica un deadlock si verifica anche starvation. When a deadlock occurs, starvation also occurs.
4. ✅ Il grafo di attesa contiene solo nodi che indicano processi non nodi che indicano risorse. The wait-for graph contains only nodes indicating processes, not nodes indicating resources.
5. ⬜ Il grafo di rivendicazione contiene solo nodi che indicano processi non nodi che indicano risorse. The claim graph contains only nodes indicating processes, not nodes indicating resources.
6. ⬜ Dato un grafo di assegnazione delle risorse la presenza di un ciclo indica la presenza di un deadlock. Given a resource allocation graph, the presence of a loop indicates the presence of a deadlock.
7. ⬜ Gli stati non sicuri sono sicuramente degli stati di stallo e per essi non esiste una sequenza sicura. Unsafe states certainly lead to deadlock states and there is not a safe sequence for them.
8. ✅ L'algoritmo del Banchiere può essere applicato tanto con risorse ad istanze singole che ad istanze multiple. The Banker's algorithm can be applied with both single and multiple instances of resources.
9. ⬜ Le condizioni necessarie per il deadlock sono mutua esclusione, possesso e attesa, prelazione e l'assenza di attesa circolare. The necessary conditions to have a deadlock are mutual exclusion, hold and wait, preemption, and no circular wait.

Ex 09 (2.5 points)

Italiano

In riferimento all'uso dei segnali in un ambiente UNIX/Linux. Dire quali delle seguenti risposte sono vere. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Refer to the use of signals in a UNIX/Linux environment. 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. ⬜ Usando la system call `signal` è possibile ignorare ogni tipo di segnale. By using the system call `signal` it is possible to ignore any type of signal.
2. ✅ All'interno di un signal handler dovrebbero essere utilizzate solo funzioni rientranti. Inside a signal handler, only reentrant functions should be used.
3. ⬜ La ricezione di un segnale da parte di un processo causa sempre la terminazione del processo. The reception of a signal by a process always causes the termination of the process.
4. ⬜ L'uso della coppia di funzioni `kill()` e `pause()` o delle primitive semaforiche `sem_post()` e `sem_wait()` sono equivalenti per la sincronizzazione dei processi. The use of the pair of functions `kill()` and `pause()` or of the semaphore primitives `sem_post()` and `sem_wait()` are equivalent for process synchronization.
5. ✅ La ricezione di alcuni segnali non può essere ignorata. The reception of some signals cannot be ignored.

6. ✅ L'esecuzione di un signal hadler dopo la ricezione di un segnale da parte di un processo può portare a race conditions. The execution of a signal handler after the reception of a signal by a process can lead to race conditions.
7. ✅ Entrambi il comando di shell `kill` e la system call `kill()` possono terminare un processo. Both the shell command `kill` and the system call `kill()` can terminate a process.
8. ⬜ La system call `kill()` termina un processo, invece il comando di shell `kill` non lo termina. The system call `kill()` terminates a process, while the shell command `kill` does not.
9. ⬜ La system call `signal()` spedisce un segnale a un processo. The system call `signal()` sends a signal to a process.

Ex 10 (2.5 points)

Italiano

Si supponga che un processo diventi "zombie" e che un altro diventi "orfano". Dire quali delle seguenti risposte sono vere. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Suppose that a process becomes "zombie" and another one 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. ⬜ La PCB dello zombie è ancora presente in memoria, quella del processo orfano non lo è più. The PCB of the zombie is still present in the memory, the one of the orphan process is not.
2. ⬜ La PCB dell'orfano è ancora presente in memoria, quella del processo zombie non lo è più. The PCB of the orphan is still present in the memory, the one of the zombie process is not.
3. ⬜ Un processo orfano diventerà un processo zombie prima o poi. The orphan process will become a zombie process sooner or later.
4. ✅ Il padre di un processo zombie non ha ancora eseguito una `wait` (o una `waitpid`). The parent of a zombie process has not made a `wait` (or a `waitpid`) yet.
5. ⬜ Il processo orfano non ha un processo padre. The orphan process does not have a parent process.
6. ⬜ La PCB di un processo padre sarà rimossa solo dopo che il processo figlio eseguirà una `wait` o una `waitpid`. The PCB of the parent process will be deleted only after the child process performs a `wait` o una `waitpid`.
7. ✅ La PCB del processo zombie sarà rimossa solo dopo che il suo padre effettuerà una `wait` o una `waitpid`. The PCB of the zombie process will be deleted only after its parent performs a `wait` or a `waitpid`.
8. ⬜ La PCB del processo orfano non sarà mai rimossa. The PCB of the orphan process will never be deleted.