Esame Sistemi Operativi - Operating Systems Exam
2020/09/14

Ex 1 (2.0 points)

Italiano

Quali dei seguenti algoritmi di schedulazione è dotato di prelazione

English

Which of the following scheduling algorithms has preemption

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

1. ⚪ Priority Scheduling (PS)
2. ✅ Round Robin (RR)
3. ⚪ Shortest Job First (SJF)
4. ✅ Shortest Remaining Time First (SRTF)
5. ⚪ First Come First Served (FCFS)

Ex 2 (2.0 points)

Italiano

Spiegare perché:

1. due processi possono o non possono condividere una variabile globale.
2. due thread possono o non possono condividere una variabile globale

English

Explain why:

1. Two processes can or cannot share a global variable.
2. Two threads can or cannot share a global variable.

Soluzione. Solution.

1. Durante la generazione di un nuovo processo, ad esempio con l'uso della system call fork(), l'address space viene duplicato e da lì in poi è disgiunto.
2. I thread condividono lo stesso address space, di conseguenza un'operazione di scrittura su una variabile globale da parte di un thread ha effetto su tutti gli altri thread.

1. During the generation of a new process, for instance by means of the system call fork(), the address space is duplicated and therefore disjoint.
2. Threads share the same address space, as a consequence a write operation on a global variable by means of a thread has effect on all the other threads

Ex 3 (2.0 points)

Italiano

Dati i segmenti di pseudo-codice inseriti al termine del testo in lingua inglese ed eseguiti in concorrenza dai processi P1 e P2 di PID pid_P1 e pid_P2, rispettivamente, indicare quali tra le seguenti affermazioni sono VERE.

English

The following pseudo-code segments, executed concurrently by processes P1 and P2 with PID pid_P1 and pid_P2, respectively, indicate which of the following statements are TRUE.

```
P1
while (1) {
      ...
      kill (pid_P2, SIGUSR1);
      pause ();
      A();
```

```
}

P2
while (1) {
      pause ();
      B();
      ...
      kill (pid_P1, SIGUSR2);
}
```

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

1. ⬜ E' possibile l'esecuzione consecutiva della funzione A() per più di una volta (senza nessuna funzione B() nel mezzo) It is possible to execute function A() more than one time (without the execution of any B() function in between)
2. ⬜ La funzione A() può essere eseguita prima della funzione B() Function A() can be executed before function B()
3. ✅ Sono soggetti a deadlock They are subject to deadlocks
4. ✅ Sono soggetti a starvation They are subject to starvation
5. ⬜ La funzione B() è sicuramente eseguita almeno una volta Function B() is certainly executed at least one time
6. ✅ La funzione B() è sempre eseguita prima della funzione A() Function B() is always executed before function A()

Ex 4 (3.0 points)

Italiano

1) Spiegare l'effetto dell'esecuzione della system call exec da un programma UNIX.
2) Spiegare cosa differenzia le versioni con "l", "v", "p" ed "e".
3) Fare un esempio di come le versioni con "v" e "l" possono essere usate.
4) Perché la system call exec potrebbe non ritornare?

English

1) Explain the effect of running the system call exec from a UNIX program.
2) Explain what differentiate the "l", "v", "p" and "e" versions.
3) Make an example of how the "v" and the "l" version can be used.
4) Why the system call exec should not return?

Soluzione. Solution.

1) La system call exec rimpiazza il programma associato ad un processo (codice, dati, ecc.) con un altro programma specificato come parametro, senza sostituire il PID (Process IDentifier) del processo iniziale.
2) In sostanza cambia il prototipo della funzione e cosa essa riceve in input:
l: list -> la funzione riceve una lista di argomenti.
v: vector -> la funzione riceve un vettore di argomenti.
p: path -> la funzione riceve solo il nome di un file (senza il path). Il file viene cercato esplorando la lista di direttori memorizzata nella variabile d'ambiente di nome PATH.
e: environment -> la funzione riceve un vettore contenente una lista di variabili d'ambiente.
3) Esempio execv:

```
char *cmd[] = {"ls", "-l", (char *)0};
execv("/bin/ls", cmd);
```

Esempio execl:

```
execl("/bin/ls", "ls", "-l", NULL);
```

4) La system call exec non ritorna al programma chiamante se viene eseguita con successo. In caso di errore, exec (e tutte le sue varianti) ritorna il valore -1.
1) The exec system call replaces the program associated with a process (code, data, etc.) with another one specified as a parameter, without replacing the PID (Process IDentifier) of the initial process.
2) It basically changes the prototype of the function and what it receives as input:
l: list -> the function receives a list of arguments.

v: vector -> the function receives a vector of arguments.
p: path -> the function receives only the name of the file (without the path). The file is searched by browsing the list of directories stored in the environment variable with name PATH.
e: environment -> the function receives a vector containing a list of environment variables.
3) Example execv:

```
char *cmd[] = {"ls", "-l", (char *)0};
execv("/bin/ls", cmd);
```

Example execl:

```
execl("/bin/ls", "ls", "-l", NULL);
```

4) The system call exec does not return to the caller program if executed with success. In the case of error, exec (in all its variants) returns -1.

Ex 5 (3.0 points)

Italiano

Un programma definisce 4 semafori (sa, sc, sb, and sac) e una variable globale n, e li inizializza nel modo seguente:
n = 0;
sem_init (sa,0,1);
sem_init (sc,0,1);
sem_init (sac,0,1);
sem_init (sb,0,0);
Poi esegue 3 thread TA, TB and TC, con il seguente codice (vedi fine domanda).
Specificare quali possono essere i possibili ordini di esecuzione (anche più di uno) tra le seguenti possibilità

English

A program defines 4 semaphores (sa, sc, sb, and sac) and a global variable n, and it initializes them as follows:
n = 0;
sem_init (sa,0,1);
sem_init (sc,0,1);
sem_init (sac,0,1);
sem_init (sb,0,0);
Then it runs 3 threads TA, TB and TC, with the following code. Specify which are the possible execution orders (possibly more than one) among the following ones

```
static void *TA () {
 sem_wait (sa);
 sem_wait (sac);
 printf ( "A ");
 n++;
 if (n==1) {
  sem_post (sb);
 } else {
  n=0;
  sem_post(sa);
  sem_post(sc);
  sem_post(sac);
 }
 pthread_exit();
}
```

```
static void *TC () {
 sem_wait(sc);
 sem_wait(sac);
 printf ( "C ");
 n++;
 if(n == 1) {
  sem_post(sb);
 } else {
  n=0;
  sem_post(sa);
  sem_post(sc);
  sem_post(sac);
 }
 pthread_exit();
}
```

```
static void *TB () {
 sem_wait(sb);
 printf ("B ");
 sem_post(sac);
}
```

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

1. ✅ *A*B*C*
2. ⬜ *B*A*C*
3. ⬜ *C*A*B*
4. ✅ *C*B*A*
5. ⬜ *B*C*A*

6. ⬜ *A*C*B*

Ex 6 (4.0 points)

Italiano

Un programma inizializza un semaforo di nome sem al valore N, e poi permette al massimo a N thread di entrare nella sezione critica CS, utilizzando in tutti i thread il seguente codice di prologo e di epilogo:

```
sem_wait (&sem);
CS
sem_post (&sem);
```

Re-implementare lo stesso prologo ed epilogo utilizzando come strategia di sincronizzazione solo 2 mutex.
Si ricordi di usare solo le seguenti system call di sincronizzazione.

```
int pthread_mutex_lock (pthread_mutex_t *mutex);
int pthread_mutex_unlock (pthread_mutex_t *mutex);
```

English

A program initializes a semaphore named sem to N, and then it allows at most N threads entering the critical section CS using in all threads the following prologue and epilogue:

```
sem_wait (&sem);
CS
sem_post (&sem);
```

Re-implement the same prologue and epilogue using only 2 mutexes as synchronization strategies.
Please, remind and use only the following synchronization system calls.

```
int pthread_mutex_lock (pthread_mutex_t *mutex);
int pthread_mutex_unlock (pthread_mutex_t *mutex);
```

Soluzione. Solution.

Sono stati usati 2 mutex: m1 e m2. m1 è inizializzato come sbloccato (cioè 1), mentre m2 è inizializzato come bloccato (cioè 0).
Two mutexes were used: m1 and m2. m1 was initialized as unlocked (i.e., 1), while m2 was initialized as locked (i.e., 0).

```
int n = 0;

pthread_mutex_lock(&m1);
n++;
if (n > N) {
      pthread_mutex_unlock(&m1);
      pthread_mutex_lock(&m2);
} else {
      pthread_mutex_unlock(&m1);
}

CS

pthread_mutex_lock(&m1);
n--;
if (n >= N)
      pthread_mutex_unlock(&m2);
pthread_mutex_unlock(&m1);
```

Ex 7 (3.0 points)

Italiano

Specificare che cosa succede se un thread cerca di: 1. leggere da una pipe, ma tutti i descrittori di file riferiti all'estremo di scrittura della pipe sono stati chiusi. 2. scrivere su una pipe, ma tutti i descrittori di file riferiti all'estremo di lettura sono stati chiusi

English

Specify what happens if a thread attempts to: 1. read from a pipe, but all file descriptors referring to the write end of a pipe have been closed. 2. write to a pipe, but all file descriptors referring to the read end of a pipe have been closed.

Soluzione. Solution.

1. Se tutti i file descriptor che si riferiscono alla parte write della pipe vengono chiusi, allora un tentativo di lettura dalla pipe invierà un end-of-file (la read ritorna).
2. Se tutti i file descriptor relativi alla parte read della pipe vengono chiusi, allora una write sulla pipe causerà la generazione di un segnale SIGPIPE per il processo chiamante. L'azione di default quando SIGPIPE viene ricevuto è la terminazione del processo.

1. If all file descriptors referring to the write-end of a pipe have been closed, then an attempt to read from the pipe will see end-of-file (read returns 0).
2. If all file descriptors referring to the read-end of a pipe have been closed, then a write to the pipe will cause a SIGPIPE signal to be generated for the calling process. The default action of SIGPIPE is the termination of the process.

Ex 8 (4.0 points)

Italiano

Un programma esegue un numero di thread **T** molto elevato, ma per non eccedere le risorse hardware globalmente disponibili sul server di calcolo, forza uno specifico tratto di codice (denominato SC) ad essere eseguito contemporaneamente da un massimo di N thread (con **N<<T**).
Per realizzare tale comportamento, il programmatore definisce un semaforo **sem** inizializzato al valore **N** e quindi permette un accesso a SC solo attraverso il seguente tratto di codice:

```
sem_wait (sem);
SC
sem_post (sem);
```

Si supponga che in un secondo momento, il programmatore desideri eseguire alcuni altri thread che occupano il doppio delle risorse hardware dei thread precedenti e quindi desideri fare in modo che ognuno di essi acceda a SC al posto di due thread precedenti. L'idea del programmatore è quella di forzare tali thread ad accedere a SC attraverso il tratto di codice successivo:

```
sem_wait (sem);
sem_wait (sem);
SC
sem_post (sem);
sem_post (sem);
```

Dopo aver scritto il precedente tratto di codice, il programmatore realizza però che esso presenta un problema fondamentale in un sistema in cui alcuni thread originali e alcuni di quest'ultimo tipo sono contemporaneamente in esecuzione.
Indicare tale problema e come sia possibile correggerlo

English

A program executes a very high number of threads **T**, but in order not to exceed the hardware resources globally available on the server, it forces a specific piece of code (called SC) to be executed simultaneously by a maximum of N threads (with **N << T**). To achieve this behavior, the programmer defines a semaphore **sem** initialized to the value **N**, and then he allows access to SC only through the following section of code:

```
sem_wait (sem);
SC
sem_post (sem);
```

Suppose that later on, the programmer wants to run some other threads that consume twice the hardware resources of the former threads, thus he wants to make each thread access to SC in place of one thread of the first type. The programmer's idea is to manage the access to SC of these threads through the following piece of code:

```
sem_wait (sem);
```

```
sem_wait (sem);
SC
sem_post (sem);
sem_post (sem);
```

Anyhow, after writing the previous piece of code, the programmer realizes that it presents a fundamental problem in a system in which some original and latter threads run simultaneously.
Indicate this problem and how it can be corrected.

Soluzione. Solution.

Il problema della soluzione proposta è la possibilità di deadlock. Se tutti gli N thread eseguono la prima sem_wait() prima che la seconda sem_wait() venga eseguita da almeno uno di loro, essi saranno bloccati poiché nessun thread potrebbe entrare nella sezione critica (CS), e di conseguenza eseguire la sem_post(), che permette agli altri thread di entrare nella sezione critica.
Una possibile soluzione è di rendere entrambe le sem_wait() atomiche, in sostanza usando la mutua esclusione:

```
sem_wait(ME);
sem_wait(sem);
sem_wait(sem);
sem_post(ME);
CS
sem_post(sem);
sem_post(sem);

/* con ME inizializzato ad 1 */
```

The problem of the proposed solution is the possibility of deadlock. If all the N threads execute the first sem_wait() before the second sem_wait() is executed by at least one of them, they would be blocked because no thread could enter the critical section (CS), and consequently execute the sem_post(), which allows other threads to enter the critical section.
A possible solution is to make both sem_wait() “atomic”, i.e. by using mutual exclusion:

```
sem_wait(ME);
sem_wait(sem);
sem_wait(sem);
sem_post(ME);
CS
sem_post(sem);
sem_post(sem);

/* with ME initialized to 1 */
```

Ex 9 (2.0 points)

Italiano

Quale tra le seguenti informazioni NON è salvata all'interno del Process Control Block (PCB)?

English

Which of the following pieces of information are NOT stored inside the Process Control Block (PCB)?

Scegli UNA SOLA alternativa. Choose JUST ONE option.

1. ⚪ Stato del processo / Process state
2. ⚪ Lista dei file aperti / List of open files
3. ⚪ Registri della CPU / CPU registers
4. ⚪ Signal handlers
5. ⚪ Identificativo del processo (PID) / Process Identifier (PID)
6. ✅ Sono tutte salvate all'interno del PCB / They are all stored within the PCB
7. ⚪ Dati amministrativi ad esempio sull'uso della CPU / Administrative data, for instace related to CPU usage

8. ⬜ Program Counter

Ex 10 (4.0 points)

Italiano

Realizzare uno script bash che richieda di introdurre il nome di un file contenente caratteri alfabetici, e fornisca errore nel caso in cui lo script sia lanciato con un numero non corretto di parametri. Il programma dovrà trasformare in lettere maiuscole le righe con un numero pari di parole ed in lettere minuscole le righe con un numero dispari di parole.

Ad esempio, se il file "file.txt" ha il seguente contenuto:

```
Nel mezzo del
Cammin di
Nostra vita mi ritrovai per una
Selva oscura
```

L'output del comando

```
> ./prog.sh file.txt
```

dovrà essere il seguente:

```
nel mezzo del
CAMMIN DI
NOSTRA VITA MI RITROVAI PER UNA
SELVA OSCURA
```

In particolare la prima riga è stata trasformata in minuscolo perché composta da 3 parole, mentre le restanti righe sono state trasformate in maiuscolo perché composte da 2, 6 e 2 parole, rispettivamente.

English

Produce a bash script that requires in input the name of a file containing alphabetic characters, and raises an error if the script is run with an incorrect number of parameters. The program must transform lines with an even number of words into capital letters, and lines with an odd number of words into lowercase letters.

For instance, if the file "file.txt" has the following content:

```
Nel mezzo del
Cammin di
Nostra vita mi ritrovai per una
Selva oscura
```

The output of the command:

```
> ./prog.sh file.txt
```

must be the following:

```
nel mezzo del
CAMMIN DI
NOSTRA VITA MI RITROVAI PER UNA
SELVA OSCURA
```

In particular, the first line has been transformed into lowercase letters because it consists of 3 words, while the remaining lines have been transformed into uppercase letters because they consist of 2, 6 and 2 words, respectively.

Soluzione. Solution.

```
#!/bin/bash
# Exam 2020/09/14 - Exercise 10

# Check correct number of parameters
if [ $# -lt 1 ]; then
      echo "Usage: ./prog.sh "
      exit 1
fi
# Read file line by line
while read line; do
      # Compute number of words in line
      n=$(echo $line | wc -w)
```

```
        # If line has an odd number of words convert it to lower case
        if [ $[$n%2] -eq 1 ]; then
                echo $line | tr [A-Z] [a-z]
        fi
        # If line has an even number of words convert it to upper case
        if [ $[$n%2] -eq 0 ]; then
                echo $line | tr [a-z] [A-Z]
        fi
done < $1
```

Ex 11 (4.0 points)

Italiano

Lo stato di un sistema, costituito dai processi (P1, ..., P6) e dalle risorse (R1, R2, R3), sia quello descritto in tabella. Si supponga inoltre che la disponibilità iniziale di risorse sia uguale a (3, 2, 3). Si indichi se lo stato indicato è sicuro oppure non sicuro. Nel primo caso si riporti l'ordine in cui i processi possono essere completati (per esempio 1 2 3 4 5 6), nel secondo caso si riporti come risposta -1.

English

The state of a system, composed of processes (P1, ..., P6) and resources (R1, R2, R3), is described in the following table. Suppose the initially available resources is equal to (3, 2, 3). Determine whether the given state is safe or not. In the first case write the order in which processes should be completed (e.g., 1 2 3 4 5 6), in the second case write -1 as a response.

Processo Process	Assegnate Assigned	Massimo Max	Fine Finish
P1	0 0 1	4 4 6	F
P2	1 0 1	4 4 4	F
P3	0 1 0	1 4 4	F
P4	0 0 0	5 5 6	F
P5	0 1 1	2 3 3	F
P6	1 1 0	4 4 6	F

Risposta. Answer.

5 3 2 1 6 4

Ex 12 (3.0 points)

Italiano

Data l'esecuzione in sequenza dei seguenti comandi:

```
touch f1
ln -s f1 f2
ln f1 f3
ln f1 f4
echo "hello" > f3
rm f1
```

Si indichi quale (o quali) dei seguenti output è (o sono) possibili

English

Given the execution in sequence of the following commands:

```
touch f1
ln -s f1 f2
ln f1 f3
ln f1 f4
echo "hello" > f3
rm f1
```

Indicate which (one or more) of the following outputs is (are) possible.

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

1. ⬜ A seguito del seguente comando After the command:
   ```
   # cat f4
   ```
   non si ottiene nessun output no output is obtained
2. ✅ A seguito del seguente comando After the command:
   ```
   # ls -l
   ```
   si ottiene il seguente output the following output is obtained:
   ```
   lrwxrwxrwx 1 scanzio scanzio 2 ago 13 16:03 f2 -> f1
   -rw-r--r-- 2 scanzio scanzio 6 ago 13 16:04 f3
   -rw-r--r-- 2 scanzio scanzio 6 ago 13 16:04 f4
   ```
3. ✅ A seguito del seguente comando After the command:
   ```
   # cat f4
   ```
   si ottiene il seguente output the following output is obtained:
   ```
   hello
   ```
4. ⬜ A seguito del seguente comando After the command:
   ```
   # cat f2
   ```
   si ottiene il seguente output the following output is obtained:
   ```
   hello
   ```
5. ✅ A seguito del seguente comando After the command:
   ```
   # echo "pippo" > f2
   # ls -l
   ```
   si ottiene il seguente output the following output is obtained:
   ```
   -rw-r--r-- 1 scanzio scanzio 6 ago 13 16:19 f1
   lrwxrwxrwx 1 scanzio scanzio 2 ago 13 16:03 f2 -> f1
   -rw-r--r-- 2 scanzio scanzio 6 ago 13 16:04 f3
   -rw-r--r-- 2 scanzio scanzio 6 ago 13 16:04 f4
   ```