Esame Sistemi Operativi - Operating Systems Exam
2020/06/16

Ex 1 (6.0 points)

Italiano

Si indichi l'output o gli output possibili che possono essere ottenuti eseguendo concorrentemente i processi P , P e P riportato di seguito.

English

Indicate the possible output or outputs that can be obtained by concurrently executing the processes P , P e P as follow.

```
init (S1, 1); init(S2, 0); init(S3, 0);
```

PA
```
wait(S2);
printf("F");
signal(S1);
wait(S2);
printf("G");
```

PB
```
wait(S1);
printf("B");
signal(S3);
wait(S1);
printf("D");
signal(S2);
```

PC
```
printf("A");
wait(S3);
printf("C");
signal(S1);
wait(S1);
printf("E");
signal(S2);
```

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

1. ✅ B A C E F D G
2. ⬜ A C B D F E G
3. ✅ B A C D F E G
4. ✅ A B C D F E G
5. ⬜ A C B E D F G
6. ✅ A B C E F D G
7. ⬜ B A C E D F G
8. ⬜ A C B E F D G

Ex 2 (6.0 points)

Italiano

Si illustrino le caratteristiche dei segnali per inviare informazioni asincrone tra processi. Si riporti un programma che ne illustri l'utilizzo effettuando le seguenti operazioni:

1. Il programma principale crea due processi figlio, P1 e P2 e rimane in attesa di ricevere segnali da tali processi. Ogni volta che riceve un segnale da P1 visualizza su standard output il messaggio "Segnale ricevuto da P1". Analogamente, ogni volta che riceve un segnale da P2 visualizza su standard output il messaggio "Segnale ricevuto da P2". Se però riceve 3 segnali consecutivi dallo stesso processo, uccide i processi P1 e P2, utilizzando il comando di shell "kill" e termina.

2. I processi P1 e P2 eseguono un ciclo infinito, all'interno del quale attendono un tempo casuale e poi inviano un messaggio al processo padre. P1 trasferisce segnali di tipo SIGUSR1; P2 trasferisce segnali di tipo SIGUSR2.

English

Describe the characteristics of the signals to send asynchronous information between processes. Implement the following program:

1. A program generates two processes P1 and P2 and it awaits for receiving signals from them. Every time it receives a message from P1 in displays on standard output the message "Signal received from P1". Analogously, every time it receives a message from process P2 it displays the message "Signal received from P2". If it receives 3 signals from the same process, it terminates processes P1 and P2 using the shell command "kill" and it terminates.

2. Process P1 and P2 run through an infinite cycle. Within the cycle, they await for a random time and then send a signal to the parent. P1 sends signal SIGUSR1; P2 sends signals SIGUSR2.

Soluzione. Solution.

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

int last_sig = -1;
int last_last_sig = -1;
int finish = 0;

void sign_handler(int sig){
	if (sig==SIGUSR1)
		printf("Signal received from P1\n");
	else if (sig==SIGUSR2)
		printf("Signal received from P2\n");

	/* It receives 3 consecutive signals from the same process */
	if (sig == last_sig && last_sig == last_last_sig) {
		finish = 1;
	} else {
		last_last_sig = last_sig;
		last_sig = sig;
	}
}

int main() {
	char cmd[100];
	pid_t pid1, pid2;
	if ( (signal(SIGUSR1, sign_handler) == SIG_ERR) || (signal(SIGUSR2,
sign_handler ) == SIG_ERR) ) {
		printf("Error initializing signal handler");
		exit(-1);
	}
	pid1 = fork();
	if (!pid1) {
		/* P1 */
		while (1) {
			sleep( rand()%2 );
			kill(getppid(), SIGUSR1);
		}
	} else {
		pid2 = fork();
		if (!pid2) {
			/* P2 */
			while (1) {
				sleep( rand()%3 );
				kill(getppid(), SIGUSR2);
			}
		}
	}
	/* Parent */
	while (1) {
		pause();
		if (finish) {
```

```
                    /* Kill P1 with a shell command */
                    sprintf(cmd, "kill -9 %d", pid1);
                    system(cmd);
                    /* Kill P2 with a shell command */
                    sprintf(cmd, "kill -9 %d", pid2);
                    system(cmd);
                    exit(0);
              }
       }
}
```

Ex 3 (3.0 points)

Italiano

Si indichino quali sono le caratteristiche dell'implementazione del paradigma produttore/consumatore riportato alla fine della domanda, ed eseguito concorrentemente da più produttori e consumatori. Segnare tutte le affermazioni vere

English

Indicate which are the characteristics of the implementation of the producer/consumer paradigm reported at the end of the question, and executed concurrently by many producers and consumers. Mark all the true answers.

```
init (full, 0); init (empty, SIZE); init (mutex, 1);

Producer () {
       int val;
       while (TRUE) {
              produce (&val);
              wait (empty);
              wait (mutex);
              enqueue (val);
              signal (mutex);
              signal (full);
       }
}

Consumer () {
       int val;
       while (TRUE) {
              wait (mutex);
              wait (full);
              dequeue (&val);
              signal (mutex);
              signal (empty);
              consume (val);
       }
}
```

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

1. ✅ E' soggetta a starvation. It is subject to starvation.
2. ✅ Limita la concorrenza dei processi. It limits processes concurrency.
3. ✅ E' soggetta a deadlock. It is subject to deadlock.

Ex 4 (3.0 points)

Italiano

Dato un file di nome "file.txt" contenente delle parole (cioè delle sequenze di caratteri alfabetici separati da una spazio, anche su più righe) con un contenuto simile al seguente:

```
one two three
four five six
seven
```

Realizzare uno script bash che stampi la parola con la lunghezza maggiore.
Dato l'esempio precedente, lo script dovrà produrre in output la parola "three" o la parola "seven" perché entrambe hanno 5 caratteri.

English

Given a file named "file.txt" containing some words (i.e., sequences of alphabetic characters separated by spaces, even on multiple lines) with content similar to the following:

```
one two three
four five six
seven
```

Implement a bash script that prints the word with a longer length.
Given the previous example, the script must print the word "three" or the word "seven" in the output because both have 5 characters.

Soluzione. Solution.

```
#!/bin/bash
# Exam 2020/06/16 - Exercise 4

longest_word=""
length_longest=0
# Scan the file word by word
for word in $(cat file.txt) do
      # Detect the length of $word
      length=$(echo $word|wc -c)
      #length=${#word} # Other possibility
      if [ $length -gt $length_longest ] then
            length_longest=$length
            longest_word=$word
      fi
done
echo $longest_word
```

Ex 5 (2.0 points)

Italiano

Il direttorio /home/user contiene i file riportati di seguito, il cui elenco è stato derivato mediante il comando ls -l.

```
-rw-r--r-- 1 scanzio root 0 mag 17 11:31 abXcx.txt
-rw-r--r-- 1 root scanzio 0 mag 17 11:38 a.py
-rw-r--r-- 1 scanzio root 0 mag 17 11:33 axcXab.txt
-rw-r-xr-- 1 scanzio root 0 mag 17 11:36 w.py
-rw-r--r-- 1 scanzio root 0 mag 17 11:33 Xcxab.txt
-rw-r--r-- 1 scanzio root 0 mag 17 11:33 x.py
-rw-r--r-- 1 scanzio root 3464324 mag 17 11:34 y.py
drwxr-xr-x 2 scanzio root 4096 mag 17 11:35 z.py
```

Quali sono i file selezione del comando find che segue:

```
find /home/user -type f \ ( -name "*.cpp" -o -name "*.py" -o -name "X?x*.txt" \)
-group root -size -1M ! -perm 644
```

English

The directory /home/user contains the following files, which were derived from the output of the command ls -l.

```
-rw-r--r-- 1 scanzio root 0 mag 17 11:31 abXcx.txt
-rw-r--r-- 1 root scanzio 0 mag 17 11:38 a.py
-rw-r--r-- 1 scanzio root 0 mag 17 11:33 axcXab.txt
-rw-r-xr-- 1 scanzio root 0 mag 17 11:36 w.py
-rw-r--r-- 1 scanzio root 0 mag 17 11:33 Xcxab.txt
-rw-r--r-- 1 scanzio root 0 mag 17 11:33 x.py
```

```
-rw-r--r-- 1 scanzio root 3464324 mag 17 11:34 y.py
drwxr-xr-x 2 scanzio root 4096 mag 17 11:35 z.py
```

What are the files selected by the find command that follows:

```
find /home/user -type f \ ( -name "*.cpp" -o -name "*.py" -o -name "X?x*.txt" \)
-group root -size -1M ! -perm 644
```

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

1. ⬜ z.py
2. ⬜ y.py
3. ✅ x.py
4. ✅ Xcxab.txt
5. ⬜ w.py
6. ⬜ abXcx.txt
7. ⬜ a.py
8. ⬜ axcXab.txt

Ex 6 (2.0 points)

Italiano

Data la seguente riga di comando bash:

```
egrep -e "^[1-9]\ " -e "^1[0-9]\ |^2[0-1]\ " input.txt | egrep -e
"[1-9][0-9]*\.[0-9]" | egrep "X$"| c ut -d " " -f 2,4
```

Riportare quali righe sono stampate in output dal precedente comando, quando esso è eseguito sul file input.txt riportato alla fine della domanda.

English

Given the following bash command:

```
egrep -e "^[1-9]\ " -e "^1[0-9]\ |^2[0-1]\ " input.txt | egrep -e
"[1-9][0-9]*\.[0-9]" | egrep "X$"| c ut -d " " -f 2,4
```

Report which lines are printed in output from the previous command, when it is executed on the file input.txt, which is reported at the end of the question.

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

```
20 Stefano 12.5 X
22 Gabriele 13.9 Y
20 Giulia 7.4 X
15 Lodovica 17.3 A
16 Saverio 24.2 Y
21 Anna 12.3 X
2 Giorgio 14 X
```

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

1. ⬜ Lodovica 17.3 A
2. ✅ Giulia X
3. ⬜ Gabriele 13.9 Y
4. ⬜ Gabriele Y
5. ⬜ Lodovica A
6. ✅ Anna X
7. ⬜ Anna 12.3 X
8. ✅ Stefano X
9. ⬜ Stefano 12.5 X

Ex 7 (6.0 points)

Italiano

Una funzione è costituita da due sezioni distinte A e B eseguite un'unica volta in sequenza:

```
... f (...) {
      A
```

```
      B
}
```

Dato che la funzione è eseguita da N thread in parallelo (con N intero, definito ma ignoto) si vogliono sincronizzare gli N thread in modo che la sezione B del codice venga eseguita da un thread solo dopo che **tutti** i thread hanno terminato l'esecuzione della sezione A. In altre parole, gli N thread si devono sincronizare dopo aver eseguito A e prima di eseguire B, in modo che nessuno inizi ad eseguire B se anche solo un thread non ha ancora terminato A. Si noti che la funzione f() può essere chiamata da ogni processo una sola volta.
Si riporti il tratto di codice che è necessario inserire tra A e B per sincronizzare gli N thread.

English

A function is formed by two distinct sessions A and B, executed only once in sequence:

```
... f (...) {
      A
      B
}
```

The function is executed by N threads in parallel (with N integer value, defined but unknown), thus the user would like to synchronize the N threads such that section B is executed by a thread only after **all** threads have terminated the execution of section A. In other words, the N threads must synchronize after running on A and before one of them is running on B. Note that function f() can be executed by each process only once.
Write the required C code to implement such a behavior.

oluzione. Solution.

```
#define N ....
sem_t m;
sem_t b;
sem_t sync;
int nwait = 0;
void f(void) {
 A();
 // T1
 sem_wait(&b);
 sem_post(&b);
 sem_wait(&m);
 nwait++;
 if(nwait==N){
 sem_wait(&b);
 for(c=0; c<N; c++) sem_post(&sync);
 }
 sem_post(&m);
 sem_wait(&sync);
 sem_wait(&m);
 nwait--;
 if (nwait==0) sem_post(&b);
 sem_post(&m);

 B();
}
void thread(void *arg) {
 while (1) {
 f();
 }

}
int main() {
 sem_init(&m, 0, 1);
 sem_init(&b, 0, 1);
 sem_init(&sync, 0, 0);
```

```
 /* Threads generation */
 pthread_create(...);

 return 0;
}
```

Ex 8 (2.0 points)

Italiano

In un sistema multiprocessore relativamente scarico (in cui i tempi di risposta sono veloci), cosa produce su video il frammento di codice che segue?

English

In a relatively unloaded multiprocessor system (where response times are fast), what does the following fragment of code produce on standard output?

```
int i=0; pthread_t thread, thread2;
void *t1(void *a){
      pthread_detach (pthread_self ());
      printf ("%d", ++i);
      return NULL;
}

void *t2(void *a){
      sleep(1);
      printf ("%d", ++i);
      return NULL;
}

int main() {
      setbuf(stdout, 0);
      if(fork())
            pthread_create (&thread, NULL, t1, NULL);
      sleep(1);
      if(fork())
            pthread_create (&thread2, NULL, t2, NULL);
      printf("A");
}
```

Scegli UNA SOLA alternativa. Choose JUST ONE option.

1. ⬜ AA
2. ✅ 1AAAA
3. ⬜ 12AAAA
4. ⬜ AAAA
5. ⬜ 1AA
6. ⬜ 12AA

Ex 9 (3.0 points)

Italiano

Lo stato di un sistema con 3 processi e 4 differenti risorse è il seguente:

- Il processo P1 detiene la risorsa {R1} ed è in attesa della risorsa {R2}
- Il processo P2 detiene la risorsa {R4} ed è in attesa delle risorse {R2, R3}
- Il processo P3 detiene la risorsa {R2} ed è in attesa delle risorse {R1, R3, R4}

Quale arco può essere rimosso per eliminare il deadlock?

English

The state of a system with 3 processes and 4 different resources is the following:

- Process P1 holds resource {R1} and waits resource {R2}

- Process P2 holds resource {R4} and waits resources {R2, R3}
- Process P3 holds resource {R2} and waits resources {R1, R3, R4}

Which edge can you remove to eliminate the deadlock?

Scegli UNA SOLA alternativa. Choose JUST ONE option.

1. ⬜ R4→P2
2. ⬜ R1→P1
3. ✅ R2→P3
4. ⬜ P3→R4
5. ⬜ P3→R1
6. ⬜ R3→P3

Ex 10 (3.0 points)

Italiano

Si consideri l'insieme di processi riportato di seguito. Calcolare il tempo di attesa per il processo P1 utilizzando l'algoritmo di scheduling RR (Round Robin).
Si consideri un quantum temporale di 10 unità di tempo.

English

Consider the following set of processes. Compute the waiting time for the process P1 using the RR (Round Robin) scheduling algorithm.
Consider a temporal quantum of 10 units.

Processo Process	TempoArrivo ArrivalTime	BurstTime
P0	0	35
P1	12	25
P2	23	30

Risposta. Answer.

43