Threads

Exam 2020/06/16 - 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

Exam 2021/02/12 - Ex 2 (2.5 points)

Italiano

Si supponga che il seguente programma venga eseguito con il valore 4 sulla riga di comando. Si riporti l'output generato. Si prega di riportare la risposta su un'unica riga, indicando i vari messaggi e valori in output separati da un unico spazio. Non inserire nessun altro carattere nella risposta. Errori di formato verranno considerati alla stregua degli altri errori. Questo è un esempio di risposta corretta: 2 5 0 3

English

Suppose to execute the following program with the value 4 passed on the command line. Report the output generated by the program. Please, report the response on a single line, indicating the various messages and output values separated by a single space. Do not insert any other character into the answer. Format errors will be treated in the same way as other errors. This is an example of a correct answer: 2 5 0 3

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

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

pthread_t thread;
int i;

void *t1 (void *a){
	int *p;
	p = (int *) a;
	i = *p;
	pthread_detach (pthread_self ());
	printf ("%d ", i);
	i--;
	if (i>0)
		pthread_create (&thread, NULL, t1, (void *) &i);
	return NULL;
}

int main(int argc, char **argv) {
	i = atoi (argv[1]);
	if (fork())
		pthread_create (&thread, NULL, t1, (void *) &i);
	sleep (1);
	printf ("%d ", -i);
}
```

Risposta: Answer:

```
4 3 2 1 0 -4
```

Exam 2021/02/12 - Ex 3 (2.0 points)

Italiano

Si analizzi il seguente tratto di codice.

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

English

Analyze the following piece of code.

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

```
void *thread_main(void *p) {
	int x, *y;
	y = (int *)p;
	x = *y;
	x += x;
	*p = x;
	return NULL;
}

int main() {
	int data = 1;
	pthread_t one, two;
	pthread_create(&one, NULL, thread_main, &data);
	pthread_create(&two, NULL, thread_main, &data);
	pthread_join(one, NULL);
```

```
        pthread_join(two, NULL);
        printf("%d\n", data);
        return 0;
}
```

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

1. ✅ Il codice contiene una race-condition The code contains a race condition
2. ⬜ Il codice non contiene una race-condition The code does not contain a race condition
3. ⬜ Può essere stampato il valore 1 The value 1 can be printed
4. ⬜ Può essere stampato il valore 0 The value 0 can be printed
5. ✅ Può essere stampato il valore 4 The value 4 can be printed
6. ✅ Può essere stampato il valore 2 The value 2 can be printed

Exam 2021/09/07 - Ex 7 (1.5 points)

Italiano

Si consideri la seguente affermazione: “Se un thread finisce, tutti gli altri thread del processo necessariamente terminano”. Si indichi in quali delle seguenti circostanze quanto affermato si verifica. Si osservi che risposte errate implicano una penalità nel punteggio finale.

English

Consider the following statement: "If one thread terminates, all other threads in the process will necessarily terminate." Indicate in which circumstances among the following ones the previous statement is correct. Note that incorrect answers imply a penalty in the final score

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

1. ⬜ Quando il thread effettua una return dalla sua funzione iniziale. When the thread performs a return from its start function.
2. ✅ Quando il thread effettua una exit. When the thread performs and exit.
3. ✅ Quando il thread effettua una return dal main. When the thread performs a return from the main.
4. ⬜ Quando il thread riceve una pthread_cancel da un altro thread. When the thread receives a pthread_cancel from another thread.
5. ⬜ Quando il thread effettua una pthread_exit. When the thread performs a pthread_exit.

Exam 2020/09/14 - 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.

Risposta: Answer:

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.

Exam 2022/06/14 - Ex 11 (3.5 points)

Italiano

Si indichino le principali differenze tra processi e thread. Si descriva la gestione di tali entità da parte del sistema operativo (struttura in memoria, ecc.). Se ne indichino le caratteristiche principali e i relativi vantaggi e svantaggi.

Si indichino inoltre le principali differenze tra thread a livello utente e thread a livello kernel, descrivendo i vari modelli di thread normalmente disponibili.

English

Indicate the main differences between processes and threads. Describe how these entities are handled by the operating system (data structures, etc.). State the main characteristics of both along with the advantages and disadvantages of using each of them. In addition, indicate the main differences between thread at user level and threads at kernel level describing the multi thread programming models usually available.

Risposta: Answer:

- Processo: Process:
  - creazione lenta slow creation
  - spazio di indirizzamento duplicato duplicate address space
  - gerarchia padre - figlio hierarchy parent - child
  - gestione attraverso PCB (Process Control Block) e diagramma stati dei processi. Managed through PCB (Process Control Block) and process states diagram.
- Thread: Thread:
  - creazione più veloce fast creation
  - spazio di indirizzamento condiviso shared address space
  - dati privati: program counter, registri, stack private data: program counter, registers, stack
  - maggiore scalabilità greater scalability
  - richiedono maggiori sforzi di sincronizzazione. require more synchronization efforts.
- Kernel thread: Kernel thread
  - thread gestiti dal kernel tramite system call threads managed by the kernel through system call
  - esiste una thread table in ogni processo (TCB simile alla PCB) exists a thread table for any process (TCP similar to PCB)
  - la schedulazione è gestita dal kernel scheduling is managed by the kernel
  - problemi su thread bloccanti problems on blocked threads
  - effettivo parallelismo. effective parallelism.
- User thread: User thread:
  - il kernel gestisce solo i processi; i thread sono gestiti da una libreria utente the kernel manage only processes; threads are managed by a user library and the kernel manages only processes;
  - le informazioni sui thread sono memorizzate all'interno dei processi information about threads are stored within processes
  - i kernel thread sono mappati su user thread. kernel threads are mapped on user threads.
- Implementazione ibrida: hybrid implementation
  - intermedia e comunemente adottata dai moderni SO. intermediate and commonly adopted by modern SOs.