

# Elementi di Architettura e Sistemi Operativi - Modulo II

Bioinformatica - Tiziano Villa

29 Settembre 2020

Nome e Cognome:

Matricola:

Posta elettronica:

problema	punti massimi	i tuoi punti
problema 1	10	
problema 2	10	
problema 3	10	
totale	30	

1. In molti sistemi di calcolo si possono produrre corse critiche (in inglese: race conditions).

(a) Si descriva che cosa e' una corsa critica.

Traccia di soluzione.

E' una situazione in cui piu' processi accedono e modificano i dati in modo concorrente e i risultati dipendono dall'ordine degli accessi.

(b) Si consideri un sistema bancario con le due funzioni seguenti:

`deposita(importo)` e `preleva(importo)`.

A queste due funzioni si passa l'importo da depositare o prelevare da un conto bancario. Si supponga che ci sia un conto bancario con 2500 euro condiviso da una moglie e un marito e che concorrentemente il marito chiami la funzione `preleva(500 euro)` e la moglie chiami la funzione `deposita(1000 euro)`.

Si descriva come può verificarsi una condizione di corsa critica nell'esempio precedente e che cosa si dovrebbe fare per prevenirla.

Traccia di soluzione.

Alla fine delle operazioni l'importo corretto sul conto dovrebbe essere di 3000 euro. Sia l'operazione di deposito che quella di prelievo si traducono in terne di operazioni elementari: lettura dell'importo sul conto dalla memoria condivisa a un registro, aggiornamento dell'importo in un registro, scrittura dell'importo da un registro alla memoria condivisa.

Dato che le tali operazioni elementari saranno eseguite sequenzialmente intercalate in un ordine qualsiasi potrebbe succedere che sia il marito che la moglie leggano l'importo di 2500 euro, poi che il marito con il suo prelievo lo diminuisca localmente a 2000 euro, e la moglie lo aumenti localmente a 3500 euro; infine che prima la moglie aggiorni a 3500 euro l'importo nella memoria condivisa, e poi il marito aggiorni a 2000 euro l'importo nella memoria condivisa. L'importo finale risultante di 2000 euro sarebbe sbagliato (e lo sarebbe anche nel caso simmetrico che fosse la moglie l'ultima a scrivere 3500 euro).

Per prevenire tale situazione bisogna istituire una sezione critica del blocco lettura-aggiornamento-scrittura e farvi accedere un solo utente per volta, in modo che se un utente inizia un'operazione a partire da un certo importo sul conto, nessun altro utente che condivide il conto possa iniziare un'altra operazione prima del termine della precedente.

2. Si consideri il seguente programma che crea processi invocando la chiamata di sistema fork.

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

int main()
{
    pid_t pid;

    /* crea mediante fork un processo figlio */
    pid = fork();

    if (pid < 0) { /* si e' verificato un errore */
        fprintf(stderr, "Fork fallita");
        return 1;
    }
    else if (pid == 0) /* processo figlio */
        execlp("bin ls", "ls", NULL);
        printf("LINE J");
    }
    else { /* processo padre */
        /* il padre attende che il figlio termini */
        wait(NULL);
        printf("Figlio completato"));
    }

    return 0;
}
```

(a) Si spieghi la semantica della chiamata di sistema `fork`.

Si analizzi il codice precedente spiegandone il funzionamento.

Traccia.

La famiglia di funzioni `exec` rimpiazza l'immagine del processo corrente con un nuovo processo specificato in argomento.

Traccia di soluzione.

La chiamata di sistema `fork()` crea un nuovo processo figlio che avrà una copia dello spazio degli indirizzi del processo genitore. Entrambi i processi genitore e figlio continuano l'esecuzione dall'istruzione successiva alla chiamata di sistema `fork()`, con una differenza: la chiamata di sistema `fork()` restituisce il valore 0 nel processo figlio, ma restituisce l'identificatore del processo figlio (il PID diverso da 0) nel processo genitore. Il processo genitore invocando la chiamata di sistema `wait()` si autorimuove dalla coda dei processi pronti fino alla terminazione del figlio. Quando il processo figlio termina, il processo genitore chiude la chiamata di sistema `wait()` e continua con il resto del codice.

(b) In quali circostanze si eseguirà la linea di codice `printf("LINE J")` ?

Si argomenta la risposta.

Traccia di soluzione.

La famiglia di funzioni `exec` rimpiazza l'immagine del processo corrente con un nuovo processo specificato in argomento.

Il processo figlio eseguirà `printf("LINE J")` solo se la chiamata precedente alla funzione `exec` fallisce. Se la `exec` va a buon fine, essa sovrascrive l'immagine in memoria del processo figlio con quella del programma `ls`, e quindi la `printf("LINE J")` non sarà eseguita, in quanto non si tornerà ad eseguire il codice del processo figlio.

3. Si consideri la seguente tabella delle pagine per un sistema con indirizzi virtuali e fisici a 12 cifre binarie con pagine di 256 byte. La lista delle pagine fisiche libere in memoria e'  $D$ ,  $E$  e  $F$  ( $D$  e' in testa alla lista,  $E$  al secondo posto ed  $F$  e' in coda).

Pagina logica	Pagina fisica
0	—
1	2
2	C
3	A
4	—
5	4
6	3
7	—
8	B
9	0

Si convertano i seguente indirizzi logici nei corrispondenti indirizzi fisici, in esadecimale. Il trattino nella colonna della pagina fisica indica che la pagina non e' in memoria.

- (a) 9EF
- (b) 111
- (c) 700
- (d) 0FF

Traccia di soluzione.

Gl'indirizzi logici e fisici sono costituiti da 12 cifre binarie, di cui le prime 4 indicano le pagine, e le ultime 8 indicano le parole in ogni pagina. In pratica, per convertire un indirizzo logico in uno fisico si sostituisce la cifra esadecimale piu' significativa con quella indicata dalla tabella; se la tabella non specifica la traduzione (segno —), allora si alloca la prima pagina fisica libera.

- (a)  $9EF \rightarrow 0EF$
- (b)  $111 \rightarrow 211$
- (c)  $700 \rightarrow D00$
- (d)  $0FF \rightarrow EFF$