Informatica B
Esercitazione 12 (Soluzioni)

20 dicembre 2021

Processi

Risolvere i seguenti problemi :

12.1 Un calcolatore dotato di un sistema operativo multiutente e multiprogrammato ha in esecuzione 3 processi, ciascuno da un utente diverso. come di seguito:

- Il primo utente sta utilizzando un programma Matlab per eseguire dei calcoli scientifici che leggono un file e di seguito eseguiscono dei calcoli che richiedono un'elevato tempo di esecuzione ma non richiede piu input dall'utente. Dato il tipo di programma, l'utente esegue lo script Matlab e si allontana dalla sua postazione per dare tempo allo script di finire i calcoli.
- Il secondo utente sta eseguendo un programma in linguaggio C. Il programma legge dei dati da un file e esegisce dei calcoli. In un caso normale di uso, il programma esegue dei calcoli abbastanza velocemente, pero dipendente dallo stato dei calcoli, in meta esecuzione potrebbe richiedere dei nuovi dati all'utente. Il secondo utente, lancia il programma in esecuzione e lascia la sua postazione per consentire al programma di finire l'esecuzione (senza fornire i secondi possibili dati).
- Il terzo utente, lancia un programma che esegue dei calcoli e stampa un messaggio in terminale, senza richiedere dati all'utente.

Descrivere possibili esecuzioni e i stati in cui passano i diversi processi e gli stati dei processi quando gli utenti ritornerano nelle propie postazioni, considerando un sistema di scheduling round-robin.

Il primo utente ha lanciato un programma attivo, che non rimane in'attesa di input dall'utente. Il programma parte da uno stato pronto, per poi essere schedulato per uno o piu quanti di esecuzione durante i quali passa tra gli stati di esecuzione e pronto. Il programma accede un file, il che richede di eseguire istruzioni di natura I/O, che sono protette quindi eseguira una chiamata al supervisore causando una interruzione interna, e passando quindi in stato di

attesa. Quando la lettura del file sara finita, il processo passero in stato pronto, e poi finira l'esecuzione in una o piu quanti di scheduling, alternandosi tra i stati di esecuzione e pronto. Di conseguenza, quando l'utente torna, potrebbe trovare il programma in due possibili stati. O il programma non ha ancora finito il suo lavoro quindi e ancora in esecuzione, o il programma ha finito la sua esecuzione.

Il secondo utente ha lanciato un programma di tipo reattivo. Il programma parte da uno stato pronto, per poi passare ad'uno stato di esecuzione per uno o piu quanti di esecuzione alternandosi con stati pronto. Quanto si richiede la lettura del file , il programma passa ad'uno stato di attesa, aspettando la lettura del file, tramite una chiamata al supervisor. Dopo la lettura, il processo passa nuovamente allo stato pronto. In seguito il programma continua l'esecuzione alternandosi tra gli stati in esecuzione e pronto. In questo caso, abbiamo anche una componente di incerteza, siccome, in funzione dei dati del file, il programma potrebbe continuare la sua esecuzione fino a quando termina il suo lavoro, oppure passare in stato d'attesa nel caso servino ulteriori dati. Siccome l'utente ha lasciato la sua postazione, il secondo processo potrebbe rimanere in uno stato di attesa fino a quanto l'utente non torna alla sua postazione. Di conseguenza, il secondo utente potrebbe trovare il programma in 3 possibili situazioni: il programma potrebbe essere ancora in esecuzione, potrebbe aver terminato l'esecuzione o potrebbe essere ancora in'attesa di un'input dall'utente.

Il terzo utente ha lanciato un programma di tipo attivo, il quale parte da uno stato pronto, passa uno o piu stati di esecuzione, alternandosi con stati pronto dipendente dalla quantita dei calcoli, per poi eseguire una chiamata al supervisor per chiedere la stampa su terminale, passando quindi ad'uno stato di attesa. Di seguito, dopo la stampa, il processo passa allo stato pronto, e attende il suo ultimo stato di esecuzione per terminare la sua attivita'. Di conseguenza, l'utente, anche in questo caso, puo trovare il programma in due possibili stati. O il programma non ha ancora finito il suo lavoro quindi e ancora in esecuzione, o il programma ha finito la sua esecuzione.

12.2 Dato il seguente script in linguaggio Matlab:

```
ris = [];
y = input("Fornisci un valore intero positivo");
for i = 1:y
    a = input("Inserisci un primo carattere");
    b = input("Inserisci un secondo carattere");
    ris = ris + [a b];
    disp(ris);
end
```

considerando il lasso di tempo tra il lancio dello script e la sua terminazione, dire quante volte il processo associato a tale script passa:

1. dallo stato di pronto allo stato di esecuzione,
2. dallo stato di esecuzione allo stato di attesa,
3. dallo stato di attesa allo stato di pronto.

Si assume che:

- ogni input() di un carattere provochi un solo passaggio da "esecuzione" ad "attesa" (e poi a "pronto" e "in esecuzione")
- similmente per ogni disp().

Per ognuno dei punti sopra elencati, il processo transita:

1. 1 [all'inizio] + 1 + 2 * y [dopo le input()] + 1 * y [dopo la disp()] = 2 + 3 * y volte da pronto a esecuzione. Per quanto riguarda il passaggio da pronto ad esecuzione infatti questo dipende non solo dalle interruzioni interne che sono determinate dalla struttura del programma, ma anche da eventi esterni su cui a priori non abbiamo informazioni e che riguardano interruzioni esterne causate da altri processi e la fine del quanto di tempo assegnato per l'esecuzione del nostro processo.
2. 1 + 2 * y [per le input()] + 1 * y [per la disp()] = 1 + 3 * y volte da esecuzione ad attesa
3. 1 + 2 * y [dopo le input()] + 1 * y [dopo la disp()] = 1 + 3 * y volte da attesa a pronto.

12	12	12	12	12
12	13	13	13	12
12	13	14	13	12
12	13	13	13	12
12	12	12	12	12

Figura 1: Matrice esempio.

Esercizio d'esame : Matlab

Esercizio 2, Tema di Esame B, 29 Gennaio 2018

Si consideri il seguente problema: si vuole creare una matrice quadrata che sia organizzata come quella in Figura 1.

1. Si scriva in linguaggio **MATLAB** una funzione iterativa **cornici** che, data la dimensione di una matrice **N** e un numero di partenza **P**, restituisca al chiamante una matrice quadrata **NxN** definita: la matrice contiene nella cornice piu esterna il numero **P** e numeri crescenti nelle cornici piu interne.

2. Si scriva inoltre uno script **MATLAB** che acquisisca da tastiera la dimensione desiderata **N** e il numero di partenza **P**, invochi la funzione **cornici** con gli opportuni parametri e infine stampi a video la matrice risultante.

1. Primo Punto

```
function M = cornici(N, P)
    % La funzione prende come parametro la dimensione
        della matrice
    % N e un numero P e restituisce una matrice quadrata
        della dimensione
    % richiesta composta da 'cornici' di numeri partendo
        da P e nella
    %  prima e ultima colonna e la prima e ultima righa
        per poi continuare
    %  con 'cornici' con numeri crescenti
    M = P + zeros(N);
     ii = 1;
     while ii < N/2
         M(ii+1:N-ii,ii+1:N-ii) = P + ii;
          ii = ii + 1;
     end
```

2. Secondo Punto

```
N = input("Inserire la dimensione della matrice:");
P = input("Inserire il numero di partenza:");
M = cornici(N,P)
```

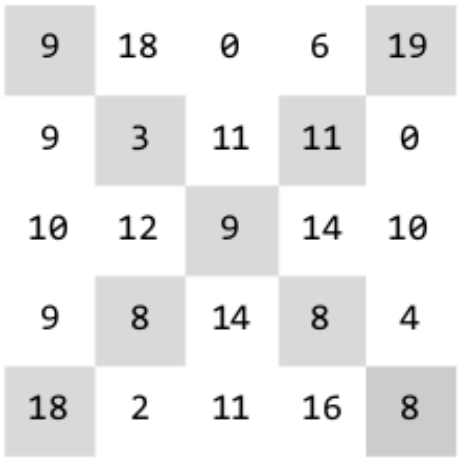

Figura 2: Matrice esempio(Esercizio 2).

Esercizio d'esame : Matlab

Esercizio 2

1. In linguaggio Matlab, si definisca la funzione con la seguente intestazione:

$$[val, freq] = frequenteSuDiagonali(M)$$

tale che:

- riceva una matrice quadrata di interi
- determini qual e' il valore intero che e' piu presente sulle due diagonali della matrice M.
- restituisca tale valore in val mentre in $freq$ restituisca il numero di volte che tale elemento e' presente sulle diagonali della matrice M. Per semplicita', si supponga che non esista un'altro elemento che e' presente lo stesso numero di volte di val sulle diagonali della matrice M.

  ad esempio, si consideri la matrice M come in Figura 1. La funzione restituira in tal caso i valori: $val = 8$ e $freq = 3$.

2. In linguaggio Matlab, si scriva uno script che:

- chieda all'utente di inserire una matrice quadrata di dimensione qualsiasi, che contenga solo numeri interi positivi;
- verifichi che la matrice inserita rispetti le condizioni indicate, e in caso contrario chieda all'utente di inserire la matrice finche' tali condizioni non sono rispettate;

- invochi la funzione frequenteSuDiagonali definita al punto precedente, passando come parametro la matrice appena acquisita, e stampi a schermo il valore piu' frequente e la frequenza corrispondente.

1. Primo Punto

```
function [val, freq] = frequenteSuDiagonali(M)
% La funzione determina il valore che e' maggiormente presente
% sulle due diagonali di una matrice.
% Input:
% M: Matrice quadrata di interi
% Output:
% val: valore intero maggiormente presente sulle due diagonali di M
% freq: Numero di volte che il valore val e' presente sulle due diagonali
% della matrice M
% Creo un filtro per selezionare gli elementi su entrambe le diagonali di M,
% combinando i filtri per selezionare ciascuna delle due diagonali
filter_diag = eye(size(M));
filter_antidiag = filter_diag(:,end:-1:1);
filter = filter_diag | filter_antidiag;
% Seleziono gli elementi (unici) e ne conto le rispettive occorrenze
all_vals = M(filter);
unique_vals = unique(all_vals');
unique_freqs = [];
for current_val = unique_vals
current_freq = sum(all_vals == current_val);
unique_freqs = [unique_freqs, current_freq];
end
% Trovo la frequenza piu' alta e l'elemento corrispondente
[freq, pos] = max(unique_freqs);
val = unique_vals(pos);
```

2. Secondo Punto

```
clear
clc
close all
% Setto a false, in modo da eseguire il ciclo almeno una volta
square = false;
numeric = false;
integers = false;
positive = false;
while (not(square) | not(numeric) | not(integers) | not(positive))
    M = input("Inserire una matrice quadrata di dimensione qualsiasi,"
"che contenga solo numeri interi positivi");
    % In alternativa: issquare(M)
    [rows, cols] = size(M);
    square = rows == cols;
    numeric = isnumeric(M);
    % In alternativa: isinteger(M)
    integers = all(M(:) == round(M(:)));
    positive = all(M(:) > 0);
end
[v, f] = frequenteSuDiagonali(M)
```