PROCESSI NON SEQUENZIALI E TIPI DI INTERAZIONE

1

ALGORITMO, PROGRAMMA, PROCESSO

Algoritmo Procedimento logico che deve essere eseguito per risolvere un determinato problema.

Programma Descrizione di un algoritmo mediante un opportuno formalismo (linguaggio di programmazione) che rende possibile l'esecuzione dell'algoritmo da parte di un particolare elaboratore (entità astratta realizzata in hardware e parzialmente in software) in grado di eseguire algoritmi specificati in un linguaggio ad alto livello).

Processo Sequenza di eventi cui dà luogo un elaboratore quando opera sotto il controllo di un programma memorizzato

Evento = esecuzione di un'operazione tra quelle appartenenti all'insieme che l'elaboratore sa riconoscere ed eseguire (elab oratore astratto)

2

PROCESSO SEQUENZIALE

- Sequenza di **stati** attraverso i quali passa l'elaboratore durante l'esecuzione di un programma (storia di un processo o traccia dell'esecuzione del programma)
- ***Esempio***: valutare il massimo comune divisore tra due numeri naturali **x** e **y**
  a) Controllare se i due numeri x e y sono uguali, nel qual caso il loro M.C.D. coincide con il loro valore
  b) Se sono diversi, valutare la loro differenza
  c) Tornare ad a) prendendo in considerazione il più piccolo dei due e la loro differenza
- Un programma descrive non un processo, ma un insieme di processi, ognuno dei quali è relativo all'esecuzione del programma da parte dell'elaboratore per un determinato insieme di dati in ingresso

3

Esempio: M.C.D. di x e y (numeri naturali)

```
{
 a = x; b = y;
 while (a != b)
    if (a > b) a = a – b
               else b = b-a;
}
```

x	**18**	**18**	**18**	**18**	**18**	**18**
y	**24**	**24**	**24**	**24**	**24**	**24**
a	**-**	**18**	**18**	**18**	**12**	**6**
b	**-**	**-**	**24**	**6**	**6**	**6**
	stato iniziale					**stato finale**

4

- Più processi possono essere associati allo stesso programma (**istanze**).
- Ciascuno rappresenta l'esecuzione dello stesso codice con dati di ingresso diversi.

Esempi:

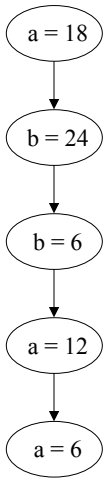
- Il **compilatore** di un linguaggio può dare luogo a più processi, ciascuno relativo alla traduzione di un particolare programma

- Il **software di controllo** può dare luogo a più processi che controllano dispositivi uguali o diversi

5

GRAFO DI PRECEDENZA

- Un processo può essere rappresentato tramite un grafo orientato detto **grafo di precedenza del processo**
- I **nodi** del grafo rappresentano i singoli eventi del processo, mentre gli **archi** identificano le **precedenze temporali** tra tali eventi
- Un **evento** corrisponde all'esecuzione di un'operazione tra quelle appartenenti all'insieme **che l'elaboratore sa riconoscere ed eseguire**
- Essendo il processo **strettamente sequenziale**, il grafo di precedenza è ad **Ordinamento Totale** (ogni nodo ha esattamente un predecessore ed un successore)



6

PROCESSI NON SEQUENZIALI

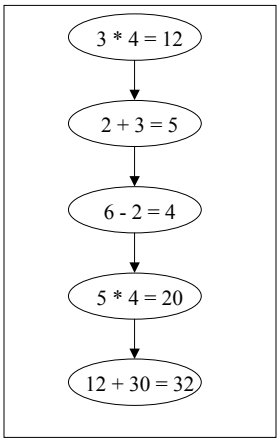
- L'ordinamento totale di un grafo di precedenza deriva dalla **natura sequenziale** del processo (a sua volta imposta dalla natura sequenziale dell'elaboratore).
- In taluni casi l'ordinamento totale è **implicito** nel problema da risolvere; spesso è un'imposizione che deriva dalla natura sequenziale dell'elaboratore.
- Esistono molti esempi di applicazioni che potrebbero più naturalmente essere rappresentate da processi (**non sequenziali**) tra i cui eventi esiste un ordinamento non totale ma solo parziale.

7

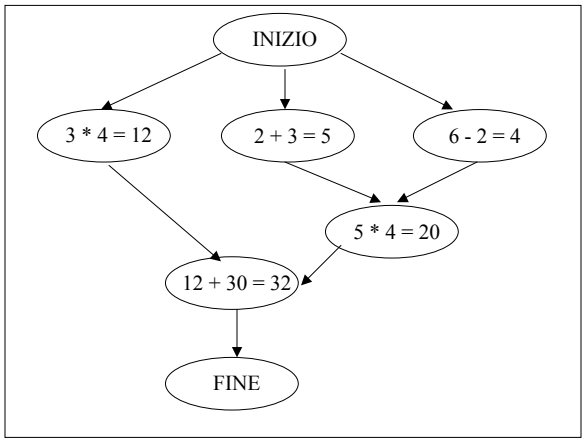
Esempio: valutazione dell'espressione

(3 * 4) + (2 + 3) * (6 - 2)

Grafo di precedenza ad ordinamento totale:



Grafo di precedenza ad ordinamento parziale:



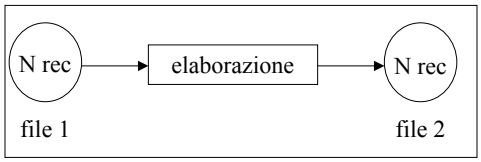
8

Esempio – segue:

- La logica del problema **non impone** un ordinamento totale fra le operazioni da eseguire; ad esempio è indifferente che venga eseguito (2 + 3) prima di eseguire (6 - 2) o viceversa.
- Entrambe le operazioni precedenti devono invece essere eseguite **prima** del prodotto dei loro risultati
- Certi eventi del processo sono tra loro **scorrelati da qualunque relazione di precedenza temporale**; il risultato dell'elaborazione è indipendente dall'ordine con cui gli eventi avvengono
- Molti settori applicativi possono essere rappresentati da **processi non sequenziali**: sistemi in tempo reale, sistemi operativi, sistemi di simulazione, etc…

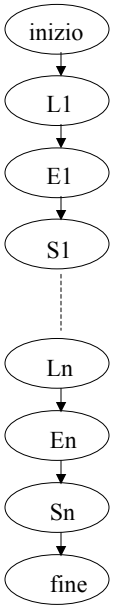
9

Esempio: Elaborazione di dati su un file sequenziale




```
Var    buffer : T;
       i : 1..N;
Begin
      for i := 1 to N do
            lettura (buffer);          (L)
            elaborazione(buffer);      (E)
            scrittura (buffer);        (S)
end
```

Grafo di precedenza ad ordinamento totale:



10

Esempio – segue:

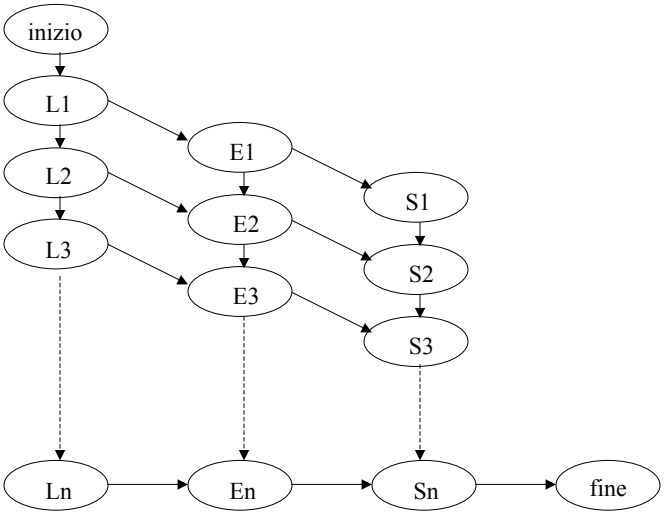
- La logica del problema impone due soli vincoli
  1. Le operazioni di lettura (o elaborazione o scrittura) dovranno essere eseguite **in sequenza** sugli N record
  2. Le operazioni di lettura, elaborazione e scrittura di un record dovranno essere eseguite **in quest'ordine**

- Non esiste alcuna relazione logica di precedenza tra la lettura dell'i-esimo e la scrittura dell'(i-1)-esimo

- Il grafo degli eventi più naturale risulta essere quello ad **ordinamento parziale**

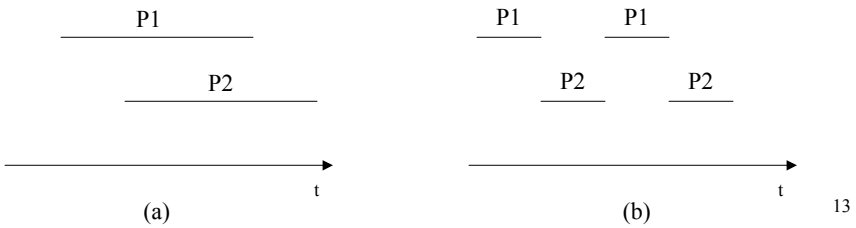
11

Esempio – segue: GRAFO AD ORDINAMENTO PARZIALE



12

- Un arco che collega un nodo di un processo con il nodo di un altro rappresenta il vincolo di precedenza tra i corrispondenti eventi
- Vincolo di sincronizzazione: **ordinamento di eventi**
- L'esecuzione di un processo non sequenziale richiede
  - elaboratore non sequenziale
  - linguaggio di programmazione non sequenziale
- Elaboratore non sequenziale (in grado di eseguire più operazioni contemporaneamente):
  - architettura parallela
  - sistemi multielaboratori (a)
  - sistemi monoelaboratori (b)



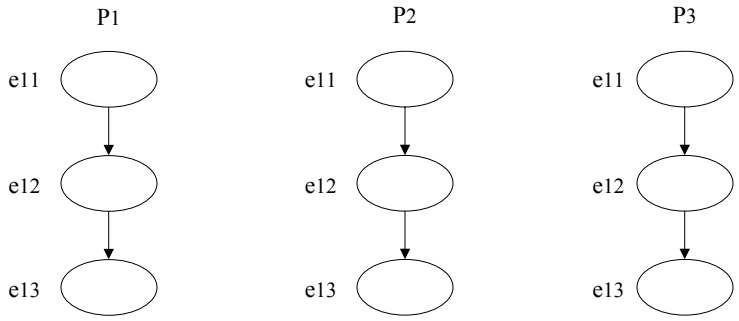
13

LINGUAGGI DI PROGRAMMAZIONE NON SEQUENZIALI

- Caratteristica comune: consentire la descrizione di un insieme di attività concorrenti tramite moduli sequenziali che possono essere eseguiti in parallelo (processi sequenziali)
- Scomposizione di un processo non sequenziale in un insieme di processi sequenziali, eseguiti "contemporaneamente", ma analizzati e programmati **separatamente**
- Consente di dominare la complessità di un algoritmo non sequenziale
- Le attività rappresentate dai processi possono essere
  - completamente indipendenti
  - interagenti

14

SCOMPOSIZIONE DI UN ALGORITMO NON SEQUENZIALE IN PROCESSI SEQUENZIALI ESEGUITI CONCORRENTEMENTE



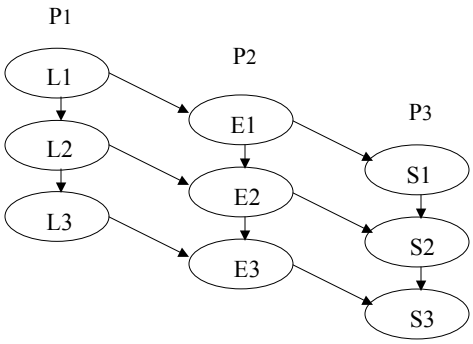
PROCESSI INDIPENDENTI

(asincroni)

- L'evoluzione di un processo **non influenza quella dell'altro**

15

- Nel caso di grafi connessi ad ordinamento parziale, la scomposizione del processo globale in processi sequenziali consiste nell'individuare sul grafo un insieme P1….Pn di sequenze di nodi (insiemi di nodi totalmente ordinati)



- Presenza di **vincoli di precedenza** tra le operazioni dei processi (espresse dagli archi) o **vincoli di sincronizzazione**

16

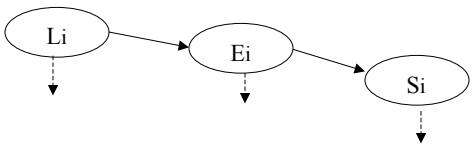
- In questo caso i tre processi **non sono fra loro indipendenti** (processi interagenti)
- Le interazioni tra i processi di lettura, elaborazione e scrittura sono relative ad uno **scambio di informazioni**. I dati su cui opera il processo di elaborazione sono forniti dal processo di lettura
- **Vincolo di sincronizzazione**: vincolo imposto da ogni arco del grafo di precedenza che collega nodi di processi diversi
- I due processi, quando arrivano ad un punto di interazione corrispondente ad uno scambio di informazioni, **devono sincronizzarsi**, cioè ordinare i loro eventi come specificato dal grafo di precedenza

17

VARI TIPI DI DECOMPOSIZIONE

La decomposizione di un grafo di precedenza ad ordinamento parziale in un insieme di sequenze di nodi (processi) può essere fatta in vari modi

ESEMPIO: Lettura, elaborazione e scrittura dell'i-esimo record (i=1,2,…,n)



- I vincoli di sincronizzazione sono gli archi verticali del grafo
- La scelta più idonea del tipo di decomposizione in processi sequenziali di un'elaborazione non sequenziale è quella per la quale le interazioni tra processi sono **poco frequenti** così da agevolare **l'analisi separata** della singole attività

18

INTERAZIONE TRA PROCESSI

- **COOPERAZIONE** – comprende tutte le interazioni "prevedibili e desiderate", insite cioè **nella logica dei programmi** (archi, nel grafo di precedenza ad ordinamento parziale).

  Prevede: **scambio di informazioni**

  - segnale temporale (senza trasferimento di dati)
  - messaggi (dati) (**comunicazione**)

  In entrambi i casi esiste un vincolo di precedenza (**sincronizzazione**) tra gli eventi di processi diversi

- **COMPETIZIONE** – "La macchina concorrente" su cui i processi sono eseguiti mette a disposizione **un numero limitato di risorse**

  Competizione per l'uso di risorse comuni che **non possono essere usate contemporaneamente**

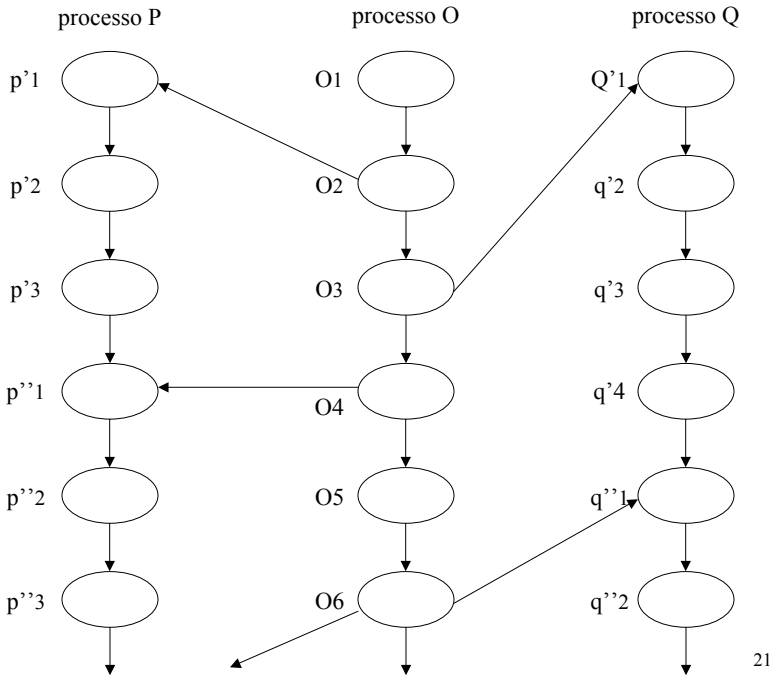
- "Interazione **prevedibile** e non **desiderata** ma **necessaria**"

19

SCAMBIO DI SEGNALI TEMPORALI

- Processo P, costituito da tre operazioni p1, p2, p3, deve essere eseguito **ogni due secondi**
- Processo Q, costituito da quattro operazioni q1, q2, q3, q4, deve essere eseguito **ogni tre secondi**
- Processo O (orologio) ha il compito di **registrare il passare del tempo** e di **attivare periodicamente** i processi P e Q inviando segnali temporali
- I nodi O1 , O2, … rappresentano le azioni del processo O e **denotano i secondi scanditi** dall'orologio
- Gli archi che collegano i nodi di O con i nodi di P (Q) rappresentano i **vincoli di precedenza** dovuti al fatto che P (Q) deve essere riattivato ogni due (tre) secondi

20



21

- Conclusa una esecuzione P (Q) **deve attendere** un nuovo *segnale di attivazione* prima di ricominciare.
- **Relazione di causa ed effetto** tra l'esecuzione dell'operazione di *invio* da parte del processo mittente e l'esecuzione dell'operazione di *ricezione* da parte del processo ricevente.
- Vincolo di precedenza tra questi eventi (*sincronizzazione dei due processi*).
- *Comunicazione*: è previsto uno scambio di dati.
- Linguaggio di programmazione deve fornire costrutti linguistici atti a specificare la **sincronizzazione e la eventuale comunicazione** tra i processi

22

COMPETIZIONE

- Due processi P e Q devono usare in certi istanti una **comune** stampante

  ps1, ps2,…, psn e qs1, qs2,…,qsn

  sono le **istruzioni** che P e Q devono rispettivamente eseguire per produrre un messaggio sulla stampante

- Le due sequenze devono essere eseguite in modo **mutuamente esclusivo**

processo P

processo Q

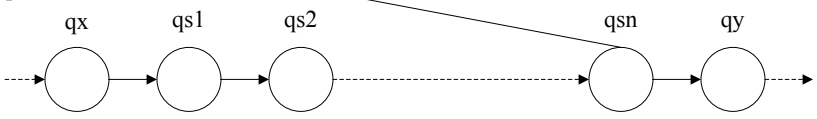
px e py (qx e qy) denotano rispettivamente l'ultima operazione eseguita da P (Q) prima della stampa e la prima eseguita dopo la stampa

23

- L'interazione si estrinseca in un **vincolo di precedenza** tra eventi di processi diversi (psn deve precedere qs1), cioè in un **vincolo di sincronizzazione**.
- Non è (come nel caso della comunicazione) un vincolo di causa ed effetto, cioè l'ordine con cui devono avvenire due eventi **non è sempre lo stesso**. Basta che sia verificata la proprietà di mutua esclusione

processo P

processo Q



24

Cooperazione → sincronizzazione diretta o espllicita

Competizione → sincronizzazione indiretta o implicita

Interferenza provocata da **errori di programmazione**

- Inserimento nel programma di interazioni tra processi non richieste dalla natura del problema
- Erronea soluzione a problemi di interazione (cooperazione e competizione) necessarie per il corretto funzionamento del programma

Interazione "non prevista non desiderata"

- dipende dalla **velocità relativa** dei processi
- gli errori possono manifestarsi nel corso dell'esecuzione del programma a seconda delle diverse condizioni di velocità di esecuzione dei processi

(ERRORI DIPENDENTI DAL TEMPO)

25

- Esempio di *interferenza del primo tipo*:

  Solo P deve operare su una risorsa R. Per errore di programmazione viene inserita in Q un'istruzione che modifica lo stato di R. La condizione di errore si presenta solo per particolari velocità relative dei processi.

- Esempio di *interferenza del secondo tipo*:

  P e Q competono per una stampante. Si garantisce la mutua esclusione solo per la stampa della prima linea. La condizione di errore si presenta solo per particolari velocità relative dei processi.

Obiettivo fondamentale della programmazione concorrente è **l'eliminazione delle interferenze**.

26