6 Il Sistema Operativo
Indice

6.1 Generalità
Livello macchina virtuale - 1

Consente l'esistenza di

Livello 5 : Macchina utente
Livello 4 : Macchina programmatore

Livello 3 : Macchina virtuale

Livello 2 : Macchina hardware
Livello 1 : Macchina microprogrammata
Livello 0 : Macchina digitale

Si appoggia su

Offre un insieme di istruzioni più *potenti* di quelle assembler, che agevolano l'accesso e l'uso delle risorse fisiche. Spesso *oscura* la natura ed il funzionamento dei livelli inferiori. Offre un linguaggio a '*chiamate di sistema*'. è prettamente software ed è *indispensabile*.

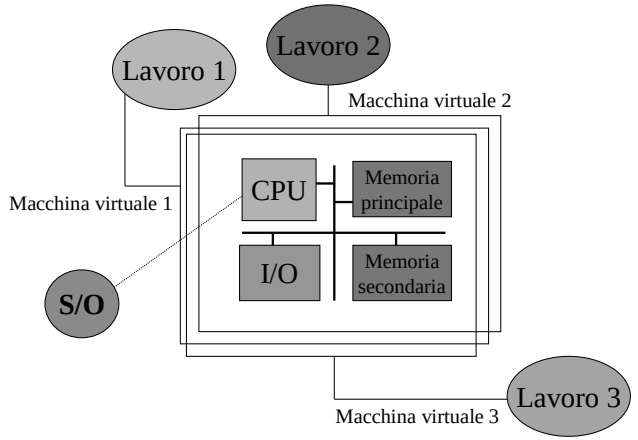
6.1 Generalità
Livello macchina virtuale - 2

- Obiettivi del Sistema Operativo (S/O)
  - Facilitare l'uso dell'elaboratore (macchina hardware)
  - Consentire uso e gestione *ottimali* delle risorse
    - Risorse **fisiche**
      - La CPU, la memoria principale, la memoria secondaria, le unita' di I/O
    - Risorse **logiche**
      - La memoria virtuale, la multi-programmazione, i modelli produttore-consumatore per l'uso delle risorse fisiche
  - Massimizzare il numero possibile di attività simultanee
    - Mantenendo un livello accettabile di prestazioni a livello utente

6.2 Classificazione e definizioni
Modello di riferimento

- Mono-processore multi-programmato
  - La macchina virtuale poggia su **una sola CPU** ed un solo spazio di indirizzamento fisico
  - Il S/O consente multi-programmazione dividendo **nel tempo** l'accesso alle risorse
  - Ciascun lavoro vede ed opera su una macchina virtuale propria

Sistema mono-processore



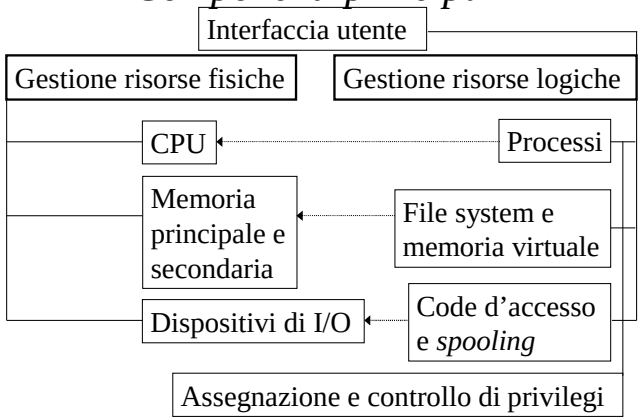
6.2 Classificazione e definizioni

Una definizione di S/O

- Un insieme di programmi progettati per
  - Offrire all'utente una astrazione più semplice e potente della macchina assembler (**macchina virtuale**)
    - Più semplice da usare (p.es., senza bisogno di conoscenze di microprogrammazione ☺)
    - Più potente (p.es., usando la memoria secondaria per realizzare una più ampia memoria principale virtuale)
  - Gestire in maniera **ottimale** le risorse fisiche e logiche dell'elaboratore
    - Ottimalità è la minimizzazione dei tempi di attesa e la massimizzazione dei lavori svolti per unità di tempo

6.2 Classificazione e definizioni

Componenti principali



6.2 Classificazione e definizioni

La nozione di processo

- Un processo è un programma in esecuzione
  - Corrisponde a
    - L'insieme **ordinato** di **stati** assunti dal **sistema** (il programma sulla macchina virtuale) nel corso dell'esecuzione
      - Visione di processo come 'automa a stati'
    - L'insieme **ordinato** delle **azioni** (operazioni del programma sulla macchina virtuale) effettuate nel corso dell'esecuzione
      - Visione di processo come 'attore' (operatore di azioni)

6.2 Classificazione e definizioni
Implementazione di processo

- Spazio di indirizzamento logico
  - La memoria della macchina virtuale che il processo può leggere e scrivere (*core*)
    - Memoria virtuale organizzata a pagine e/o segmenti
    - Programma eseguibile
    - Dati del programma
      - Organizzazione dell'informazione in forma di *file*
    - Aree di lavoro
    - Definizione del **contesto di esecuzione** ed area di salvataggio

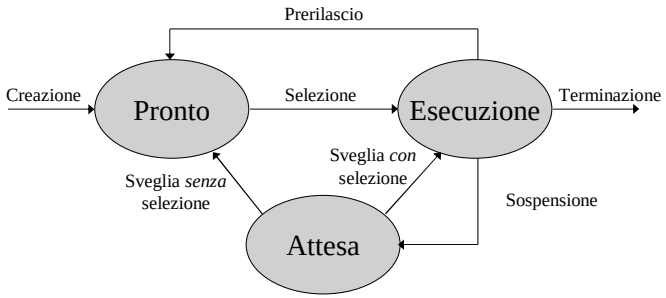
6.2 Classificazione e definizioni
Caratteristiche di processi - 1

- In un sistema *coesistono* processi utente e processi di S/O
  - Possono cooperare ma hanno privilegi diversi
- I processi avanzano concorrentemente
  - Il S/O assegna loro le risorse necessarie secondo diverse politiche di ordinamento
    - A divisione di tempo; a livello di priorità
- I processi possono dover comunicare e sincronizzarsi tra loro
  - Il S/O deve fornire i meccanismi ed i servizi necessari

6.2 Classificazione e definizioni
Caratteristiche di processi - 2

- Un processo può creare processi 'figli'
  - Esempio: un processo interprete di comandi (*shell*) lancia un processo figlio per eseguire il comando inviato da un utente
- I processi vengono
  - **Creati** per eseguire un lavoro
  - **Sospesi** per consentire l'esecuzione di altri processi
  - **Terminati** al compimento del lavoro assegnato
    - Un processo figlio che sopravvive alla terminazione del processo padre è detto 'orfano' ed è molto dannoso

6.2 Classificazione e definizioni
Stati di avanzamento di processo



6.2 Classificazione e definizioni
Gestore dei processi - 1

- Costituisce il cuore o nucleo del S/O (*kernel*)
  - Gestisce ed assicura l'avanzamento dei processi
    - Stato di avanzamento
      - In esecuzione, pronto per l'esecuzione, sospeso in attesa di un evento (una comunicazione, la disponibilità di una risorsa, …)
    - La scelta del processo da eseguire ad un dato istante si chiama ordinamento (*scheduling*)
    - Il gestore decide il cambio di stato dei processi
      - Per divisione di tempo
      - Per trattamento di eventi (p.es., risorsa libera / occupata)

6.2 Classificazione e definizioni
Gestore dei processi - 2

- La politica di ordinamento deve essere 'giusta' (*fairness*)
  - Processi pronti per eseguire devono avere l'opportunità di farlo
  - Processi in attesa di risorse devono avere l'opportunità di accederle
- I meccanismi e servizi di comunicazione e sincronizzazione devono essere efficaci
  - Il dato (o segnale) inviato da un processo mittente deve raggiungere il destinatario in un tempo breve ed in modo sicuro

6 Il Sistema Operativo
Indice (segue)

6.2 Classificazione e definizioni
Definizione di risorsa

- **Risorsa** è qualsiasi elemento fisico (hardware) o logico (a gestione software) necessario alla creazione, esecuzione ed avanzamento di processi
- Le risorse possono essere
  - Durevoli (p.es., CPU) o consumabili (p.es., memoria fisica)
  - Ad accesso divisibile od indivisibile
    - Divisibile se tollera alternanza con accessi di altri processi
    - Indivisibili se *non* tollera alternanza
  - Ad accesso individuale o molteplice
    - Molteplicità fisica o logica (virtualizzata)

6.2 Classificazione e definizioni
La risorsa processore

- Risorsa *indispensabile* per l'avanzamento di tutti i processi
- A livello *fisico* (hardware) corrisponde alla CPU
- A livello *logico* (gestione software) corrisponde ad una macchina virtuale offerta dal S/O

6.2 Classificazione e definizioni
La risorsa memoria

- Risorsa ad accesso individuale se in scrittura, ad accesso molteplice se in lettura
- La gestione software la *virtualizza* (usando la memoria secondaria) attribuendone l'accesso ai vari processi secondo particolari politiche
- Se virtualizzata, diventa riutilizzabile e pre-rilasciabile
  - Altrimenti consumabile ed indivisibile
- Gestione velocizzata con l'utilizzo di supporto hardware

6.2 Classificazione e definizioni
Le risorse di I/O

- Risorse generalmente riutilizzabili, non pre-rilasciabili, ad accesso individuale
- La gestione software ne facilita' l'impiego nascondendone le caratteristiche hardware e uniformandone il trattamento
- L'accesso fisico ha bisogno di utilizzare programmi proprietari e specifici (*BIOS*)

6.2 Classificazione e definizioni
File system

- Risorsa logica introdotta per virtualizzare e facilitare la visione, l'uso e la gestione della memoria
- Un processo vede la memoria (ed i dispositivi di I/O) come un *file system*
  - I *file* risiedono in memoria secondaria
  - Il programma e' compilato in un *file* che, chiamato in esecuzione, diventa un processo
- La gestione software consente la creazione, la cancellazione e la modifica delle proprietà dei *file*

6.2 Classificazione e definizioni
L'interfaccia utente - 1

- L'insieme delle modalità con le quali l'utente può accedere ai servizi offerti dal S/O
- Accesso basato su un interprete di comandi (*shell*) che li acquisisce, li decodifica e ne attiva l'esecuzione
  - Accesso mediante chiamate di sistema (*system call*)
  - Attivazione mediante caricamento del programma richiesto e creazione del processo corrispondente

6.2 Classificazione e definizioni
L'interfaccia utente - 2

- Definizione di un interfaccia standard di sistema (*application programming interface, API*)
  - Consente l'implementazione di S/O *indipendente* dal linguaggio applicativo
- Linguaggi applicativi *concorrenti* (p.es., Ada, Java)
  - Possiedono un supporto a tempo di esecuzione *indipendente* dal S/O

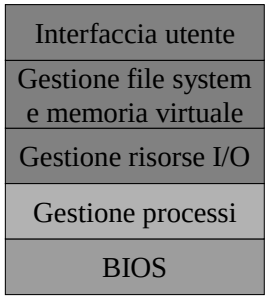
6.2 Classificazione e definizioni
Sicurezza e protezione

- Sicurezza come affidabilità
  - Capacità di svolgere correttamente le proprie funzioni
  - Anche in presenza di guasti
    - *Tolleranza ai guasti*
- Sicurezza come protezione
  - Capacita' di prevenire accessi ed operazioni non autorizzate
    - *Protezione e controllo degli accessi*

6.3 Struttura generale
Esempio architettura: S/O a livelli - 1

- Funzionalità suddivise a *livelli gerarchici*
  - Ogni componente di un livello può utilizzare soltanto i servizi offerti dal livello sottostante
- Esibiscono maggior modularità e maggior facilità di controllo
  - Ogni livello superiore e' un S/O più potente del livello inferiore
  - La struttura a livelli facilita il controllo degli accessi
- Una buona suddivisione a livelli non è facile da definire

6.3 Struttura generale
S/O a livelli - 2



6.3 Struttura generale
Modelli di comunicazione tra processi

- Ad ambiente globale
  - Comunicazione tramite *variabili condivise*
  - Necessita di meccanismi di *protezione*
    - P.es., semafori
- Ad ambiente locale
  - Comunicazione mediante *scambio di messaggi*
    - Modello *sincrono*
    - Modello *asincrono*

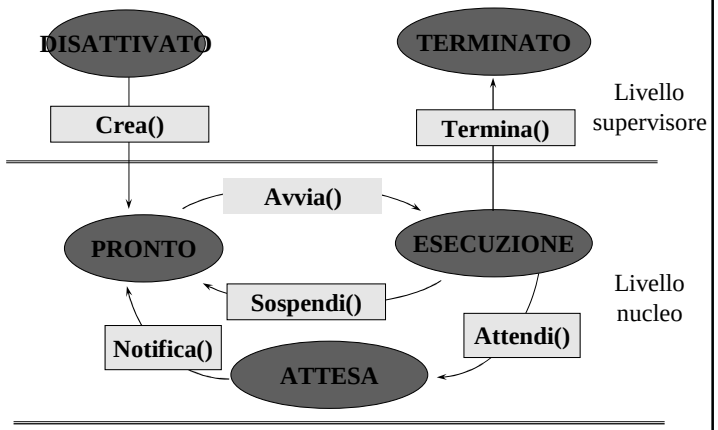
6.3 Struttura generale
Caricamento del S/O

Il S/O può risiedere

- Permanentemente in ROM
  - Soluzione tipica di sistemi di controllo industriale e di sistemi *dedicati*
- In memoria secondaria ed essere caricato (tutto o in parte) in memoria principale all'attivazione di sistema (*bootstrap*)
  - Adatto a sistemi di elevata complessità oppure adibiti al controllo (alternativo) da parte di più S/O
  - In ROM risiede il caricatore di sistema (*bootstrap loader*)

6.3 Struttura generale
S/O elementare - 1

- Struttura concettuale di un S/O mono-processore multi-programmato a suddivisione di tempo (*time sharing*), organizzato a livelli
  - **Supervisore**
    - Verifica dei diritti ed assegnazione delle risorse
  - **Gestore delle risorse**
    - Fisiche e logiche, con esclusione del processore
  - **Nucleo** (*kernel*)
    - per la gestione dei processi



6.3 Struttura generale
S/O elementare - 2

DISATTIVATO

Il programma e' in memoria secondaria. Il supervisore lo carica in memoria mediante una chiamata di sistema che crea una struttura di controllo di processo (*Process Control Block, PCB*)

PRONTO

Il processo, pronto per l'esecuzione, rimane in attesa del suo turno

ESECUZIONE

Il processore e' stato attribuito al processo selezionato, la cui esecuzione avanza

6.3 Struttura generale
S/O elementare - 3

ATTESA

Il processo è sospeso in attesa di una risorsa attualmente non disponibile o di un evento non ancora verificatosi

TERMINATO

Il processo ha concluso regolarmente le sue operazioni e si predispone ad abbandonare la sua macchina virtuale

6.3 Struttura generale
S/O elementare - 4

Crea()

Assegna una macchina virtuale ad un nuovo processo, aggiornando lista dei processi pronti (*ready list*)

Avvia()

Manda in esecuzione il primo processo della lista dei pronti

Sospendi()

Il processo in esecuzione ha esaurito il suo quanto di tempo e torna in fondo alla lista dei pronti

6.3 Struttura generale
S/O elementare - 5

Attendi()

Il processo richiede l'uso di una risorsa o l'arrivo di un evento e viene sospeso se la risorsa è occupata o se l'evento non si e' ancora verificato

Notifica()

La risorsa richiesta dal processo bloccato e' di nuovo libera o l'evento atteso si e' verificato. Il processo ritorna nella lista dei pronti

Termina()

Il processo in esecuzione termina il suo lavoro e rilascia la macchina virtuale (ed il *PCB* ad essa associato)

6.3 Struttura generale
S/O elementare - 6

I compiti del livello nucleo sono

- Gestire le transizioni di stato di attivazione dei processi
- Gestire le interruzioni esterne causate da
  - Eventi di I/O
  - Situazioni anomale rilevate da altri processi o componenti del S/O
- Consentire ai processi di accedere a risorse e di attendere eventi

6 Il Sistema Operativo
Indice (segue)

6.4 La gestione dei processi
Strutture di rappresentazione

- Ogni processo è rappresentato da un **descrittore** (*Process Control Block*) contenente
  - Identificatore del processo
  - Contesto di esecuzione (stato interno) del processo
  - Stato di avanzamento del processo
  - Priorità (iniziale ed attuale)
  - Diritti di accesso alle risorse e privilegi
  - Puntatore al PCB del processo padre e degli eventuali processi figli
  - Puntatore alla lista delle risorse assegnate alla macchina virtuale del processo

6.4 La gestione dei processi
Ordinamento di processi

- Diversi metodi sono utili per determinare quando porre un processo in stato di esecuzione in sostituzione di un altro (*switch*)
  - **Scambio cooperativo** (*cooperative* o *non pre-emptive switching*): il processo in esecuzione decide quando passare il controllo al processo successivo → Windows 3.1 ☹
  - **Scambio a prerilascio**: il processo in esecuzione viene rimpiazzata da
    - In processo pronto ed a priorità maggiore (*priority-based pre-emptive switching*) → Sistemi detti 'a tempo reale'
    - All'esaurimento del suo quanto di tempo (*time-sharing pre-emptive switching*) → Unix, Windows NT (misti!)

6.4 La gestione dei processi
Dispatcher - 1

- L'avviatore di processi all'esecuzione viene chiamato *Dispatcher*
  - Deve essere estremamente efficiente perché gestisce *ogni* scambio
  - Deve salvare il contesto del processo in uscita, installare quello del processo in entrata (*context switching*) ed affidargli il controllo della CPU
- L'efficienza del *dispatcher* si misura in
  - Percentuale di utilizzo della CPU
  - Numero di processi avviati all'esecuzione per unità di tempo
  - Durata di permanenza di un processo in stato di pronto

6.4 La gestione dei processi
Dispatcher - 2

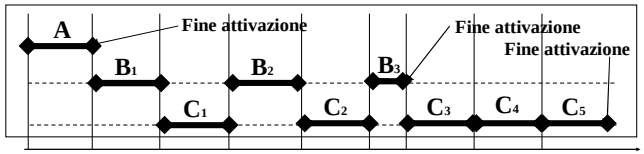
- I processi in stato di pronto sono accodati in una struttura detta 'lista dei pronti' (*ready list*)
- La più semplice gestione della lista è con tecnica a coda (*First-Come-First-Served*, *FCFS*)
  - Il primo processo ad entrare in coda sarà anche il primo avviato all'esecuzione
  - Facile da implementare e da gestire
  - La garanzia di esecuzione di altri processi (*fairness*) dipende dalla politica di scambio
    - Lo scambio cooperativo *non* offre garanzie

6.4 La gestione dei processi
Dispatcher - 3

- Le attività di un processo tipicamente comprendono sequenze di azioni eseguibili dalla_CPU intervallate da sequenze di azioni di_I/O
- I processi si possono dunque classificare in
  - *CPU-bound*
    - Comprendenti *poche* attività sulla CPU e di durata *molto lunga*
  - *I/O- bound*
    - Comprendenti molte attività, di breve durata, sulla CPU, intervallate da attività_di I/O molto lunghe
- La tecnica FCFS penalizza i processi della classe *I/O-bound*

6.4 La gestione dei processi
Dispatcher - 4

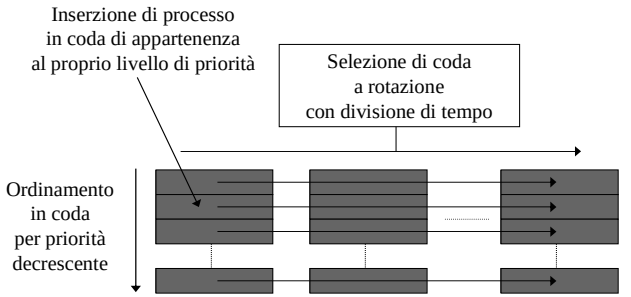
- Imponendo la suddivisione di tempo (*time-sharing*) sulla politica *FCFS* si deriva una tecnica di rotazione detta *round-robin*
- Vediamone l'applicazione su tre processi A, B e C con tempi di esecuzione 2, 5 e 10 ms. e quanto di tempo 2 ms.



6.4 La gestione dei processi
Dispatcher - 5

- Ad ogni processo possiamo attribuire una priorita' individuale, che denota il suo livello di privilegio
- Processi diversi possono poi essere categorizzati per attributi (p.es.: *CPU- bound*, *I/O- bound*)
- Possiamo allora istituire una coda per ciascuna categoria di processo, ed ordinarla a priorità
- Stabiliamo poi una politica di ordinamento *tra* code (p.es.: *round-robin*)
- Otteniamo una politica di ordinamento a livelli (p.es.: rotazione con priorità)

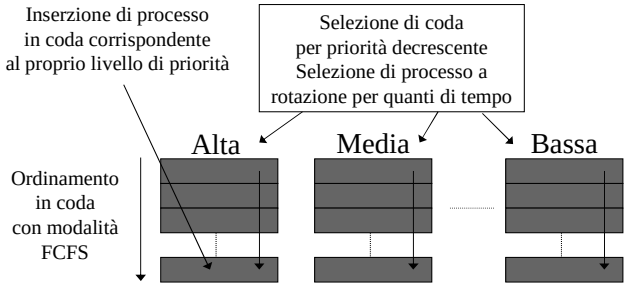
Politica a rotazione con priorità



6.4 La gestione dei processi
Dispatcher - 6

- Possiamo anche facilmente (ed utilmente) definire una politica *duale* alla precedente
  - Istituiamo una coda per ogni livello di priorità attribuita ai processi
  - Selezioniamo la coda a priorità più elevata
  - Applichiamo la politica a rotazione (*round-robin*) sul processo selezionato
  - Otteniamo la politica a priorità con rotazione

Politica a priorità con rotazione



6.4 La gestione dei processi
Creazione di processi - 1

- In ambiente UNIX il programmatore crea un processo utilizzando la chiamata di sistema *fork()*
- Tale funzione crea ed attiva un processo figlio, in tutto simile al processo padre (un clone), ma con proprio identificatore (*Process Identifier*, *PID*)
- La funzione ritorna un valore che ne indica l'esito
  - -1 creazione non riuscita
  - 0 creazione con successo, viene restituito al processo figlio
  - >0 creazione con successo, viene restituito al padre, e denota il *PID* del figlio

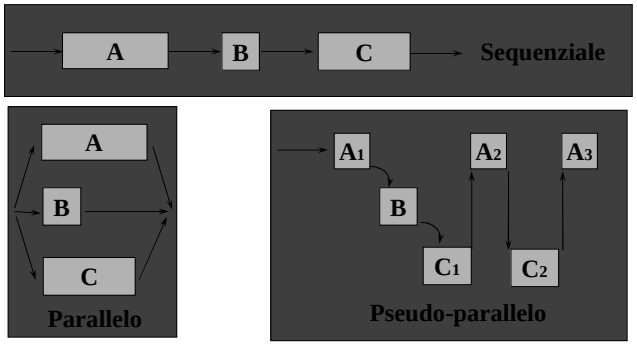
6.4 La gestione dei processi
Creazione di un processo - 2

```
#include <stdio.h>
#include <sys/types.h>
int main(void) {
  pid_t pid;
  pid = fork();
  if (pid == -1) {
    fprintf(stderr, "fork() non riuscita.\n");
    exit(-1);
  }
  if (pid == 0) {
    printf("Sono il processo figlio.\n");
    exit(0);
  }
  if (pid > 0) {
    printf("Sono il padre del figlio nr %d.\n", pid);
    exit(0);
  }
} /* I messaggi appariranno in ordine casuale */
```


6.4 La gestione dei processi
Avanzamento dei processi - 1

- I processi possono avanzare in maniera
  - **Sequenziale**, nei sistemi mono-processore con S/O in modalità 'a lotti' (*batch*)
  - **Parallela**, nei sistemi multi-processore
  - **Pseudo-parallela**, nei sistemi mono-processore con S/O capaci di supportare più macchine virtuali

6.4 La gestione dei processi
Avanzamento dei processi - 2



6.4 La gestione dei processi
Avanzamento dei processi - 3

- La modalità di avanzamento (pseudo-) parallelo consente l'esecuzione concorrente (reale o simulato) dei processi
- Ciò consente un più efficace utilizzo delle risorse, un aumento della velocità di esecuzione globale ed una strutturazione più modulare dei programmi

6.4 La gestione dei processi
Avanzamento dei processi - 4

L'avanzamento concorrente incauto di più processi può portare a risultati indesiderati:

Il processo P_1 esegue 2 azioni sequenziali, A e B:
(*p.es.:* A: x=0 *e* B: y=x+2)
Il processo P_2 esegue 2 azioni sequenziali, C e D
(*p.es.:* C: x=10 *e* D: y= x-1)
Diverse sequenze di avanzamento con diversi effetti:

A B C D *(x=10, y=9)*	C A B D *(x=0, y=-1)*
A C D B *(x=10, y=12)*	C A D B *(x=0, y=2)*
A C B D *(x=10, y=9)*	C D A B *(x=0, y=2)*