UNIVERSITÀ
DEGLI STUDI DI TRIESTE

dipartimento
di ingegneria
e architettura

Corso di Laurea in Tecniche di Radiologia Medica per immagini e Radioterapia

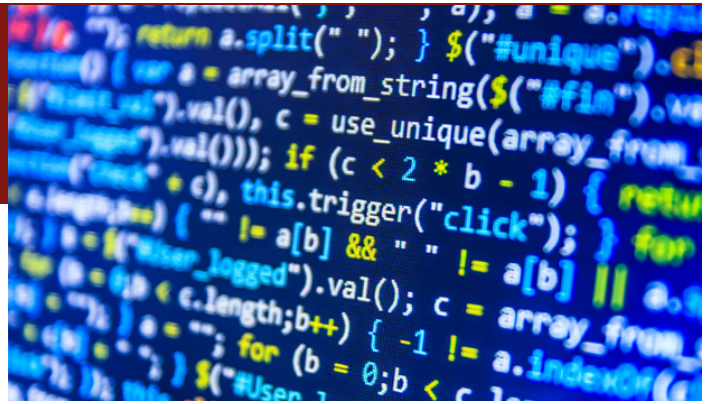
Sistemi Elettronici e informatici in ambito di Imaging I

1CFU – 10 ore

SOFTWARE: PROGRAMMI E SISTEMI OPERATIVI

Prof. Sara Renata Francesca Marceglia

Software



Users

Application software

Operating and systems software

Hardware

Interfaccia utente/macchina

Application Software (programmi)

- Algoritmo
  - descrizione della soluzione di problema scritta in modo da poter essere eseguita da un esecutore (eventualmente diverso dall'autore dell'algoritmo)
  - sequenza di istruzioni che operano su dati.
- Programma
  - algoritmo scritto in modo da poter essere eseguito da un calcolatore (esecutore automatico)

- Per scrivere un programma è necessario rappresentare istruzioni e dati in un formato tale che l'esecutore automatico sia capace di memorizzare e manipolare.

Il percorso di programmazione

Identificazione del problema

- Problema: classe di domande omogenee (concetto astratto)
- Richiesta: può essere l'istanza di un problema (caso specifico)
- Esempio: sommare I numeri 2 e 3 è l'istanza del problema generale di somma di due numeri (A+B)
- Per trovare la soluzione: dall'istanza devo risalire al problema e risolvero a livello astratto → poi funzionerà per tutti I problem di quella classe

Analisi del problema

- Comprensione di dati INPUT e dati OUTPUT (cosa mi è fornito, cosa mi è richiesto)

Risoluzione del problema

- Definizione di una trasformazione F che dati I dati in input mi fornisca I dati in output

Verifica della soluzione

- La soluzione deve essere testata
- Va definite il modello di test

Esempio: istanza, problema e soluzione

ISTANZA DEL PROBLEMA

VOGLIO ORDINARE IN MODO CRESCENTE IL VETTORE
[2 77 1 935 11 19 773 15 3]

IDENTIFICAZIONE DEL PROBLEMA

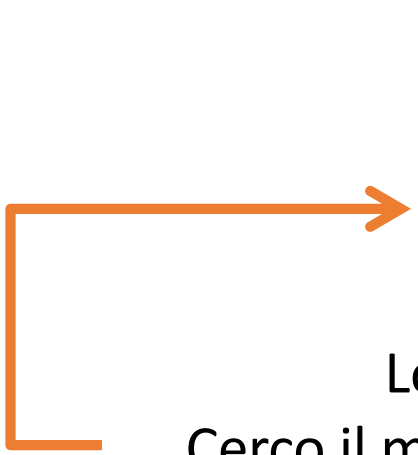
ORDINAMENTO DEI DATI IN UN VETTORE

ANALSI DEL PROBLEMA

INPUT: VETTORE
OUTPUT: VETTORE ORDINATO

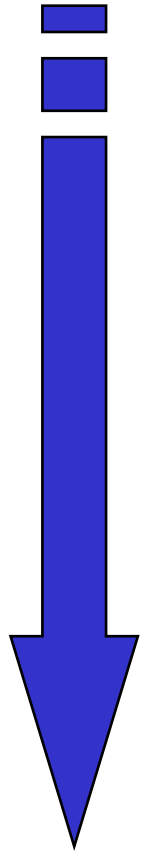
Esempio: istanza, problema e soluzione

DATO IL VETTORE IN INGRESSO
[2 77 1 935 11 19 773 15 3]



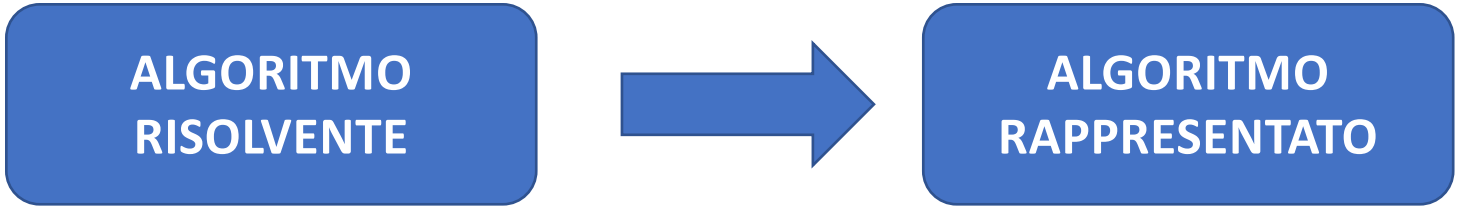
Cerco il minimo
Lo metto da parte
Lo elimino dal vettore
Cerco il minimo nel vettore rimanente

VETTORE IN USCITA
[1 2 3 11 15 19 77 773 935]



La formalizzazione della soluzione

LINGUAGGIO DI RAPPRESENTAZIONE



- Soluzione astratta del problema
- Sequenza ordinata di azioni elementari

- Sequenza di azioni comprensibili all'esecutore
- Soluzione adattata all'esecutore (CPU)

Esempio: algoritmo e linguaggio

Problema: Calcolare la data successiva ad una data fornita in input

Pseudo linguaggio

Data in input

Se il giorno non è l'ultimo del mese → aggiungo un giorno
e mantengo inalterato mese e anno

Se il giorno è l'ultimo del mese e il mese non è dicembre
→ output è il primo giorno del mese successivo

Se il mese è dicembre → il giorno successivo è il 1
gennaio dell'anno successivo

Esempio: algoritmo e linguaggio

Problema: Calcolare la data successiva ad una data fornita in input

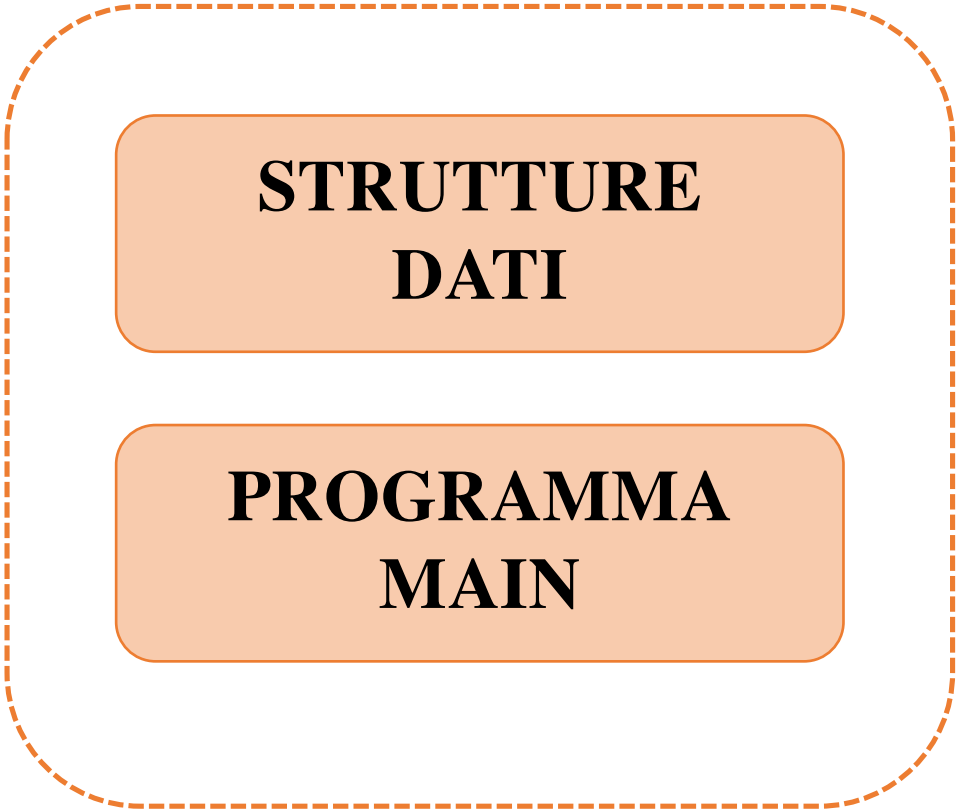
Linguaggio C

```
int main()
{
   int giorno,mese,anno;
   puts("Dammi 3 numeri, rispettivamente giorno, mese ed anno:");
   scanf("%d%d%d",&giorno,&mese,&anno);
   if (giorno<30)
   giorno=giorno+1;
   else {
       if (giorno==30 && mese==12){
                   giorno=1;
                   mese=1;
                   anno=anno+1;
                           }else {
                                 if (giorno==30 && mese<12)
                                   giorno=1;
                                   mese=mese+1;
                                   anno=anno;
                                 }
       }
   printf("%d/%d/%d",giorno,mese,anno);
   system("PAUSE");
   return 0;
}
```

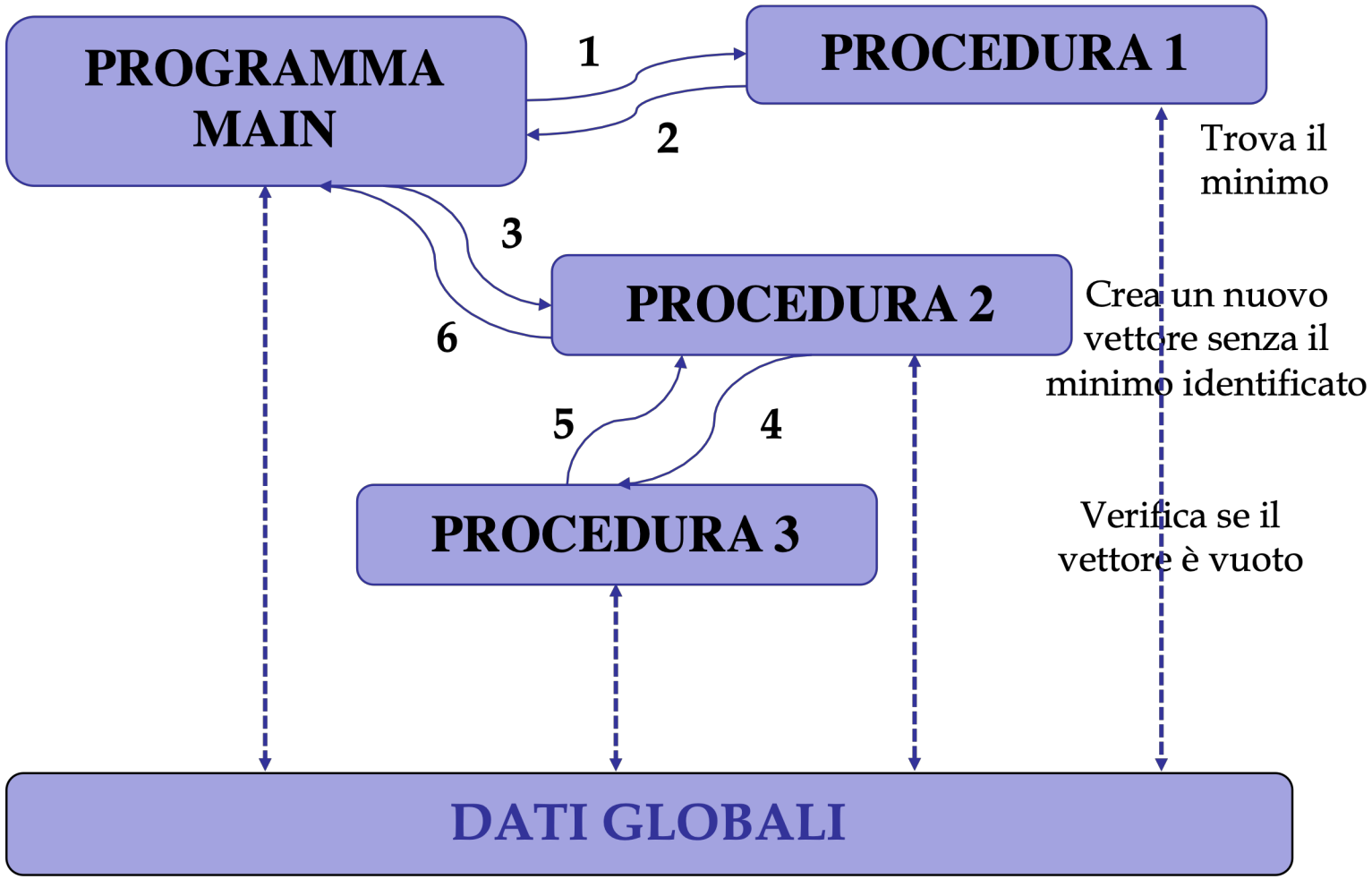
Gestione dei dati: Variabili, Costanti, Array

- Ogni elemento di un programma ha associata un'etichetta o **identificatore**
- Una variabile è un elemento il cui valore può variare nel tempo
- Una costante riceve un valore all'inizio dell'esecuzione che poi non varia più
- Un array è un vettore di elementi identici

Programmazione non strutturata



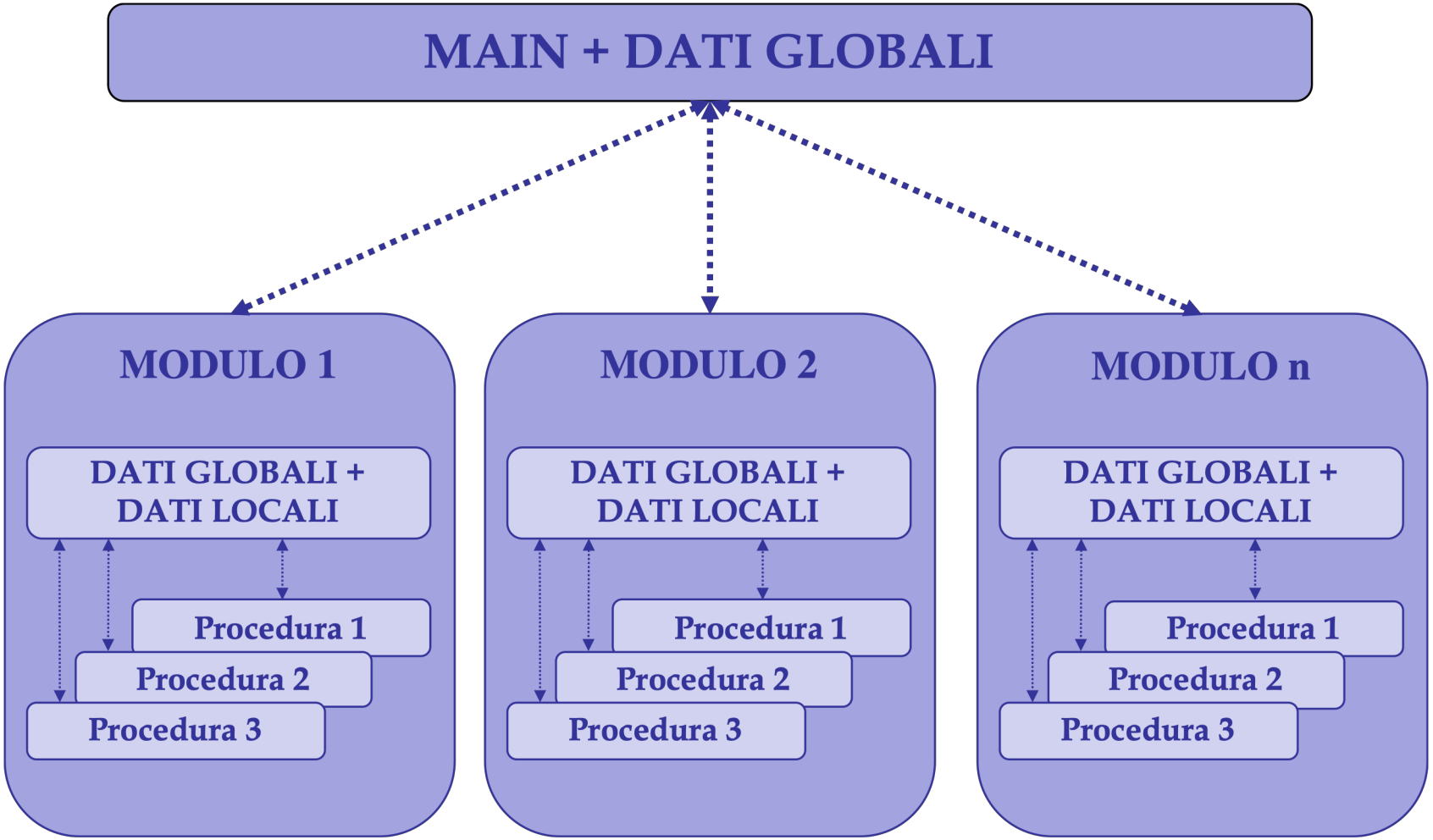
Programmazione procedurale



Programmazione procedurale

- Una procedura viene invocata col suo nome
- Una procedura accetta delle variabili che prendono il nome di **argomenti**
- Esempi:
  - LeggiDatiDaFile(`c:\pippo.dat')
  - StampaSuVideo(`salve mondo')
  - Addiziona(totale, nuovodato)

Programmazione modulare



Il compilatore

- E’ un programma eseguibile
- Traduce le istruzioni scritte in un linguaggio ad alto livello in istruzioni comprensibili per il computer
- Può produrre codice macchina o codici intermedi
- Il processo di compilazione si divide in varie fasi

Interpretazione ed esecuzione

File sorgente (testo, linguaggio di alto livello)

↓

Compilazione

↓

File codice binario (binario, pseudo istruzioni macchina)

↓

Interpretazione

↓

Codice macchina nativo in memoria e sua esecuzione

Interpretazione ed esecuzione

File sorgente (testo, linguaggio di alto livello)

↓

Compilazione

↓

File oggetto (binario, linguaggio macchina)

↓

Linking

↓

File eseguibile (binario, linguaggio macchina, pronto per l'esecuzione)

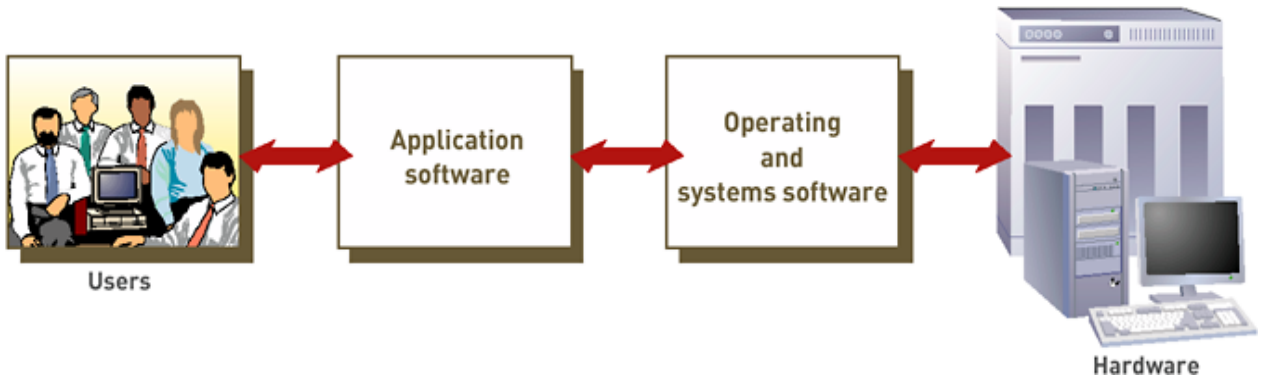
Programmazione modulare

Sistemi operativi

Con il termine *sistema operativo* si intende l'insieme di programmi e librerie che opera direttamente sulla macchina fisica...

… mascherandone le caratteristiche specifiche...

… e fornendo agli utenti un insieme di funzionalità di alto livello



Esempi

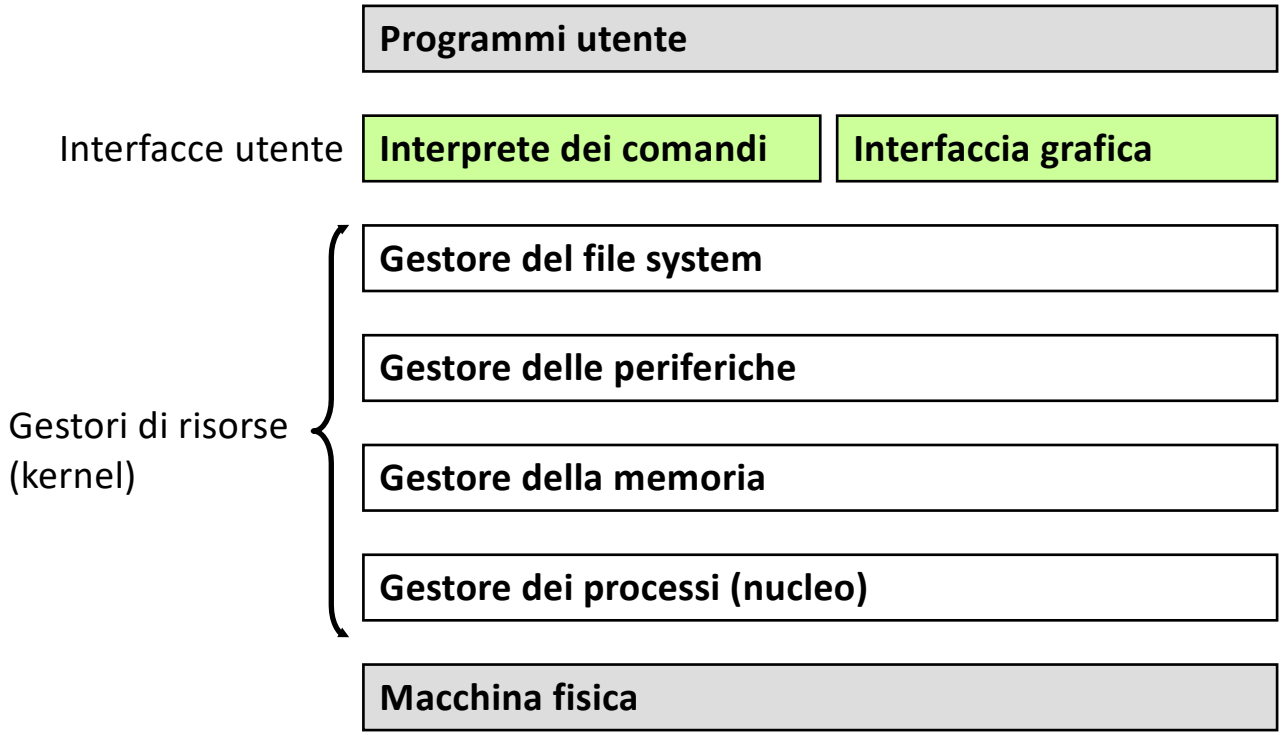
- MS-DOS
- MS-Windows (95, 98, 2000, XP, Vista, 7, 8, 10)
- Unix
  - Commerciali: Sun Solaris, IBM AIX, HP-UX,…
  - Linux
  - BSD
- Mac OS
  - Mac OS X
- Altri (IBM AS/400, Symbian,…)

Programmazione modulare

Architettura di un SO

- Un moderno S.O. è organizzato secondo una architettura *“a strati”* (a cipolla)
- Ogni strato implementa una macchina virtuale più potente del precedente
  - Appoggiandosi alle funzionalità offerte dallo strato precedente
- Tale approccio permette una chiara separazione tra interfaccia e implementazione delle diverse funzionalità
- Ogni strato è costituito da un insieme di programmi e librerie
  - I meccanismi di chiamate tra livelli possono essere diversi
    - chiamate a sottoprogrammi
    - interruzioni sincrone o asincrone
    - invio di messaggi a processi

Architettura di un SO

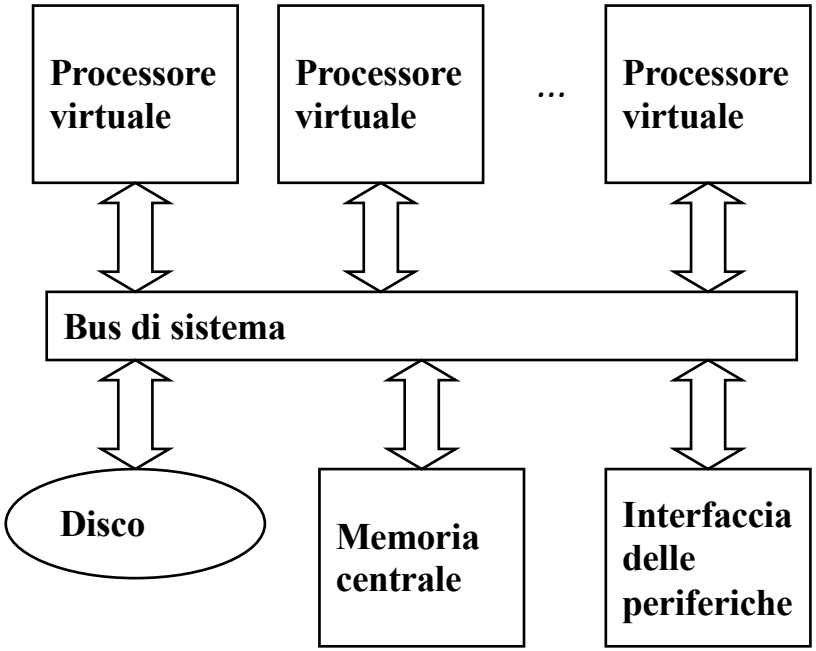


Il nucleo Kernel

- Si appoggia direttamente sulla macchina fisica
- Effettua la gestione dei processi
- In un sistema multitasking realizza una macchina virtuale in cui ad ogni processo è assegnata un processore virtuale
- Comprende i principali programmi di risposta ad interruzione
- Realizza le primitive di sincronizzazione e scambio messaggi tra processi

Il nucleo Kernel

- La macchina virtuale realizzata dal nucleo

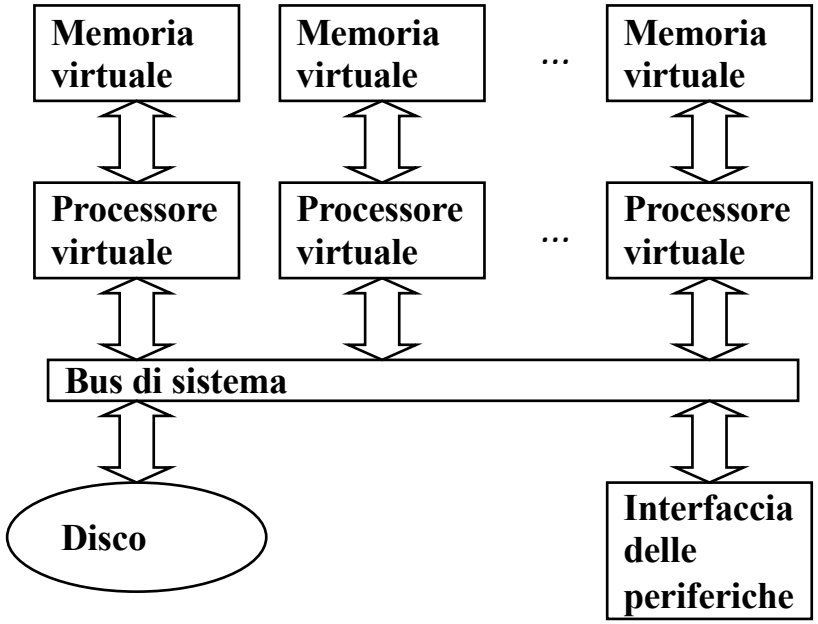


Il gestore della memoria

- Realizza le funzionalità di allocazione della memoria
  - Supera i limiti della memoria fisica e mostra ai processi uno spazio di memoria virtuale
- Partiziona la memoria tra i vari processi che la richiedono
  - Garantendo la protezione delle diverse zone di memoria

Il gestore della memoria

- La macchina virtuale realizzata dal gestore della memoria

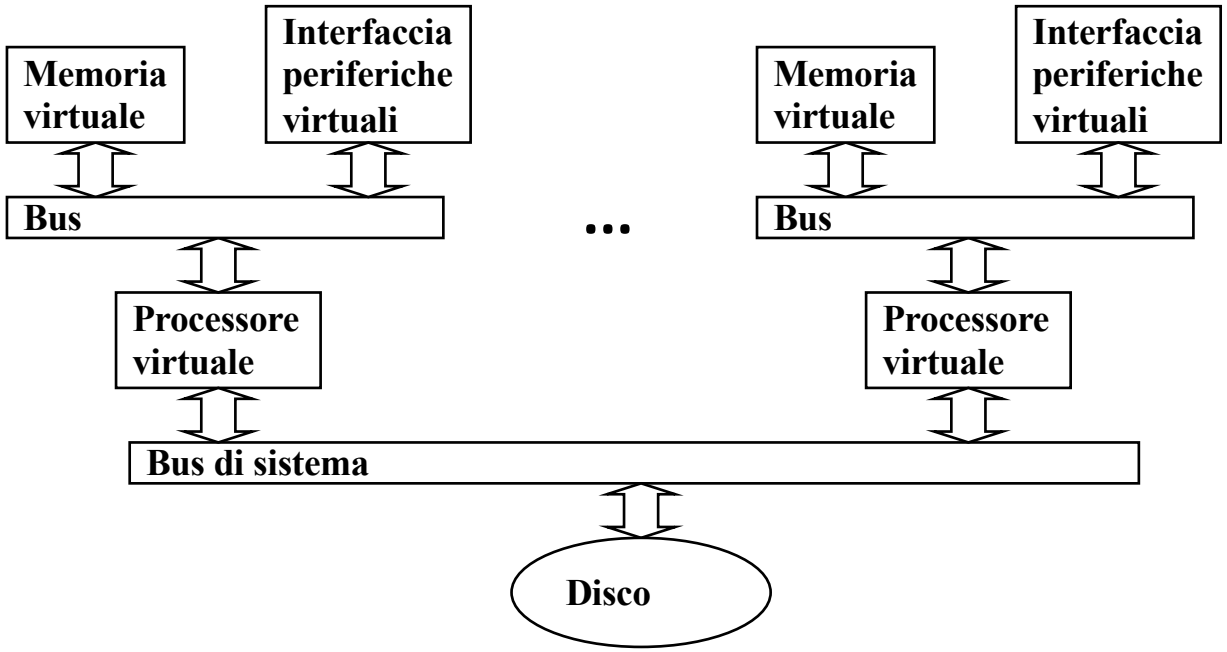


Il gestore delle periferiche

- Maschera le caratteristiche fisiche delle periferiche
- Fornisce agli strati superiori un insieme di procedure di alto livello per l'accesso alle diverse periferiche
- Offre ad ogni processo la visibilità di un insieme di periferiche virtuali dedicate
- Gestisce, almeno in parte, i malfunzionamenti delle periferiche

Il gestore delle periferiche

- La macchina virtuale realizzata dal gestore delle periferiche



Il gestore del file system

- E’ responsabile della gestione delle periferiche di massa
  - Hard disk
  - CD / DVD
  - Memory stick USB
  - ...
- Fornisce agli strati superiori un insieme di procedure per l’accesso al file system
- Garantisce la protezione nell’accesso ai file

Il gestore del file system

- Le funzioni di base che sono supportate da un file system sono
  - Il recupero di dati precedentemente memorizzati
  - L'eliminazione di dati obsoleti
  - La modifica/aggiornamento di dati preesistenti
  - La copia di dati
    - Tra supporti di memorizzazione diversi (es. da HD a CD)
    - In cartelle diverse nello stesso supporto

Il gestore del file system

- I dati contenuti nella memoria di massa vengono strutturati e gestiti mediante una organizzazione in file
- Un file è un contenitore logico identificato da un nome (filename)
- I filename generalmente sono composti da due parti
  - ad es. curriculum_vitae.doc
  - Il filename vero e proprio (curriculum_vitae)
  - L'estensione (doc)

Il gestore del file system

- L'estensione è spesso associata al programma che ha generato il file e individua pertanto la tipologia del contenuto del file
  - .exe → file eseguibili
  - .txt → file di testo
  - .doc → documenti di testo (MS Word)
  - .wav → file audio
  - .bmp → immagine in formato bitmap
  - ...
- Ad ogni file sono poi associati dal sistema operativo altri dati
  - Data di creazione / modifica
  - Lunghezza del file (in byte)
  - Utenti/Gruppi che possono accedere ai file

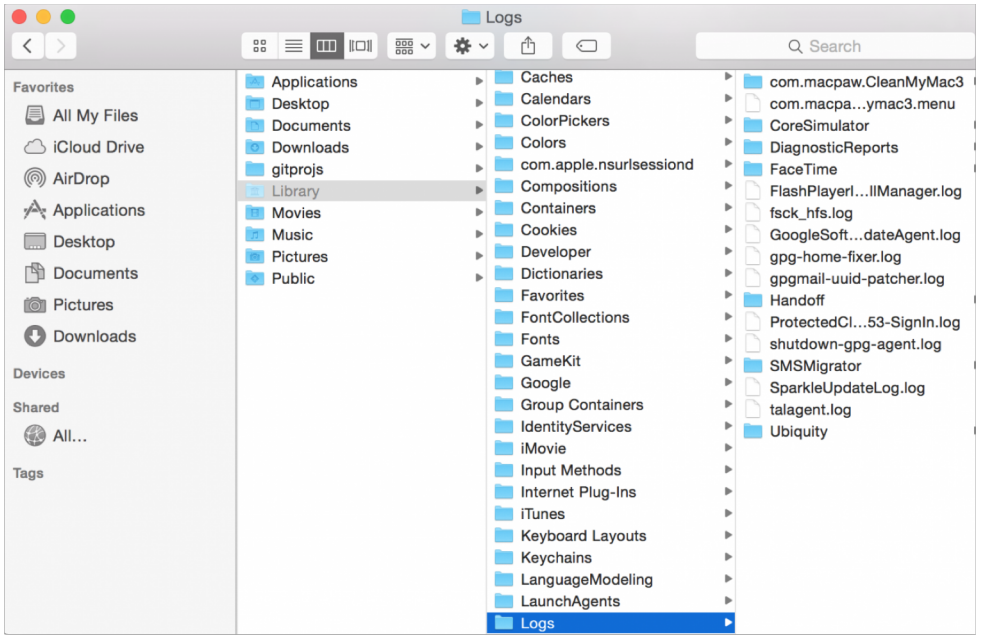
Il gestore del file system

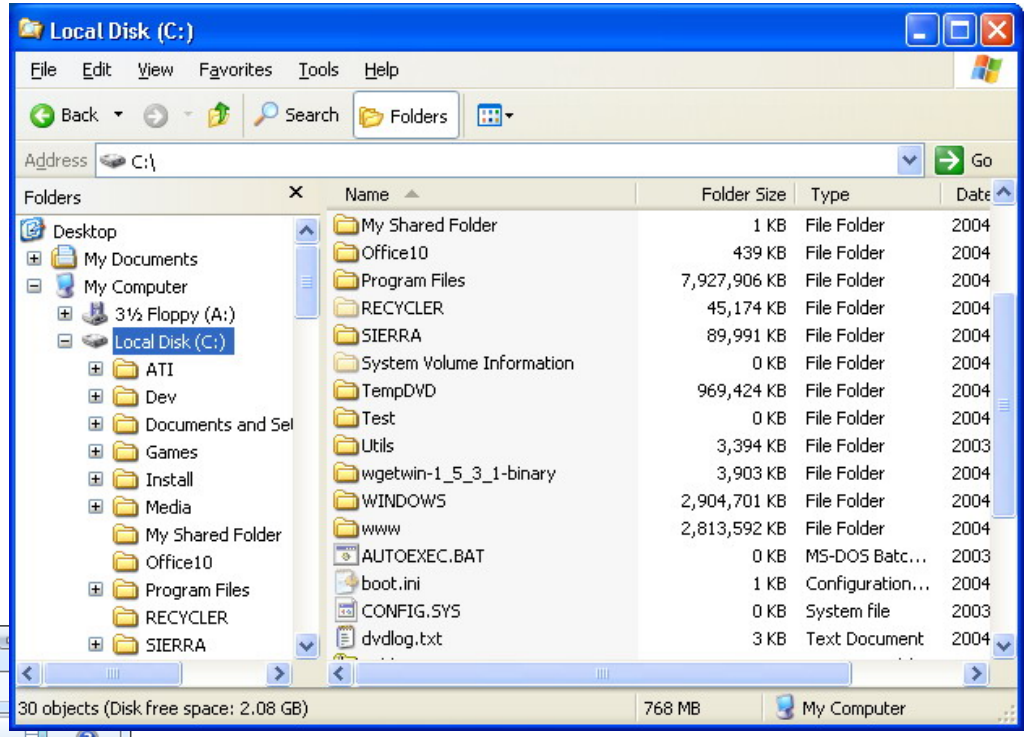
- I file vengono suddivisi in più contenitori logici, chiamati directory, cataloghi o cartelle (folders)
  - Le cartelle sono organizzate secondo una struttura ad albero

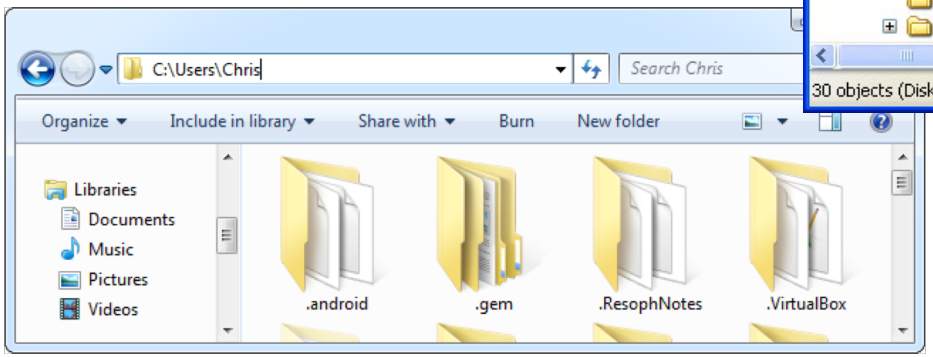
- Il file system contiene una directory detta radice (ROOT) dell'albero che può contenere file o altre cartelle
- Ciascun file è individuato univocamente dal suo nome completo o percorso assoluto
  - Ad es. D:\downloads\temp\002.part
- Due file con lo stesso nome in due cartelle distinte, ad esempio
  - D:\Immagini\Compleanno\foto1.jpg
  - D:\Documenti\foto1.jpg

  fanno riferimento a due file che in generale possono essere DIVERSI

Il gestore del file system

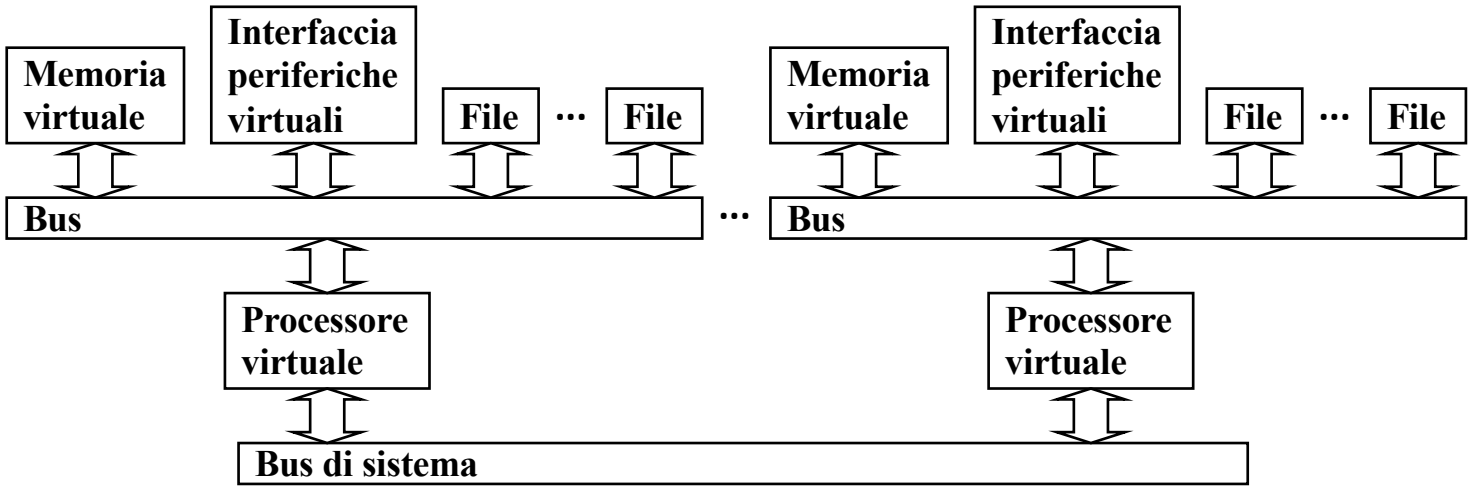






Il gestore del file system

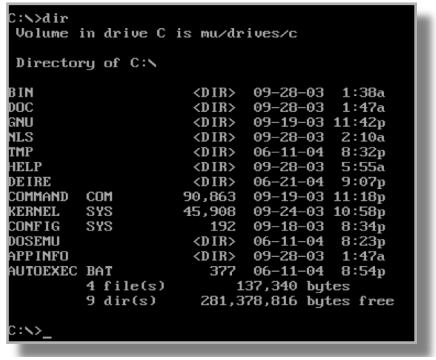
- La macchina virtuale realizzata dal gestore del file system



L’interprete dei comandi e l’interfaccia grafica

- Costituiscono l’interfaccia verso l’utente
- Consentono l’interazione dell’utente con il s.o. e con i programmi applicativi in esecuzione
  - Permettono di accedere ai programmi conservati su memoria di massa e mandarli in esecuzione
    - allocando la memoria necessaria
    - creando il processo relativo
- Nel caso di un sistema multiutente forniscono ai diversi utenti la visione di una macchina virtuale dedicata

Prompt dei comandi



Tipologie di sistemi

Sistemi monotask:

- Permettono l'esecuzione di un solo programma utente per volta (es. DOS)
- Il computer a disposizione del programma dall'inizio alla fine della sua esecuzione
- Coda dei job, gestita FIFO (first in, first out) e/o con priorità

Sistemi multitask

- Permettono l'esecuzione di più programmi utente contemporaneamente
- Classificazione ulteriore:
  - Multitasking cooperativo (Windows 3.1, MacOS originario)
  - Multitasking preemptive (Windows 95/98/NT, Unix)
  - Time sharing o meno

Sistemi monotask: svantaggi

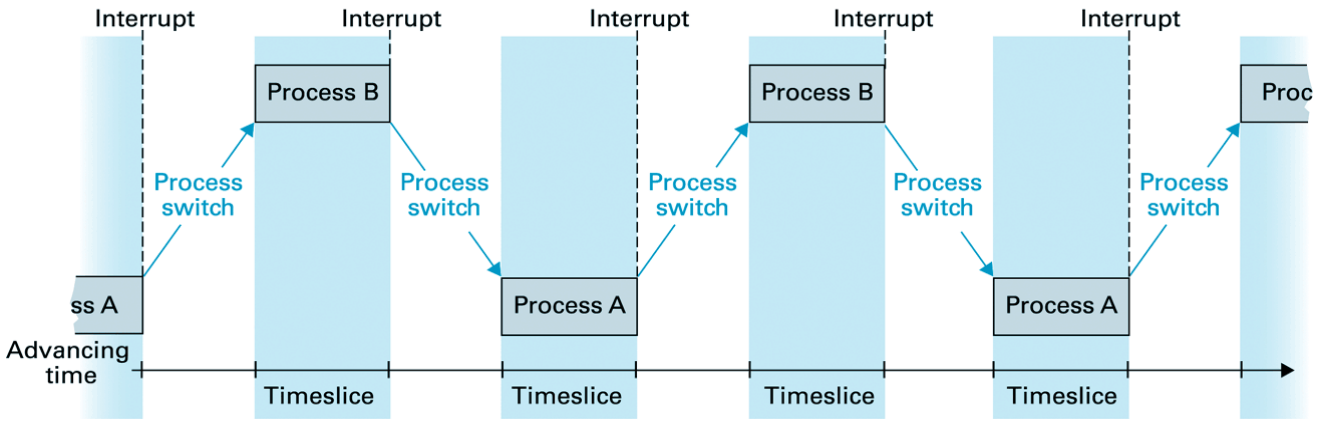
- Nessuna interazione utente-programma durante l'esecuzione di qualche compito
- Lentezza: la CPU non può essere usata da nessun processo mentre il programma in esecuzione svolge operazioni di I/O (molto piu' lente di letture/scritture in Memoria)
- Esempio: DOS è un SO monotasking; non si può fare niente altro mentre si formatta un floppy o si memorizzano dati su disco

I processi

- Processo ≠ programma !
- Processo = esecuzione di un programma, composto da:
  - codice eseguibile (il programma stesso)
  - Dati
- Lo stesso programma può essere associato a più processi:
  - Un programma può essere scomposto in varie parti e ognuna di esse può essere associata a un diverso processo
  - Lo stesso programma può essere associato a diversi processi quando esso viene eseguito più volte, anche simultaneamente

Sistemi multitask time sharing

- Ripartizione del tempo di CPU tra tutti i processi che la vogliono
- Ogni job rimane in esecuzione solo per un quanto di tempo, poi l'esecuzione passa al prossimo job e il primo va in attesa
  - Esecuzione globale più veloce
- Durata del quanto di tempo tra 100 e 200 millisecondi
  - Granularità molto fine
- A ogni utente sembra di avere la CPU tutta per lui, solo più lenta

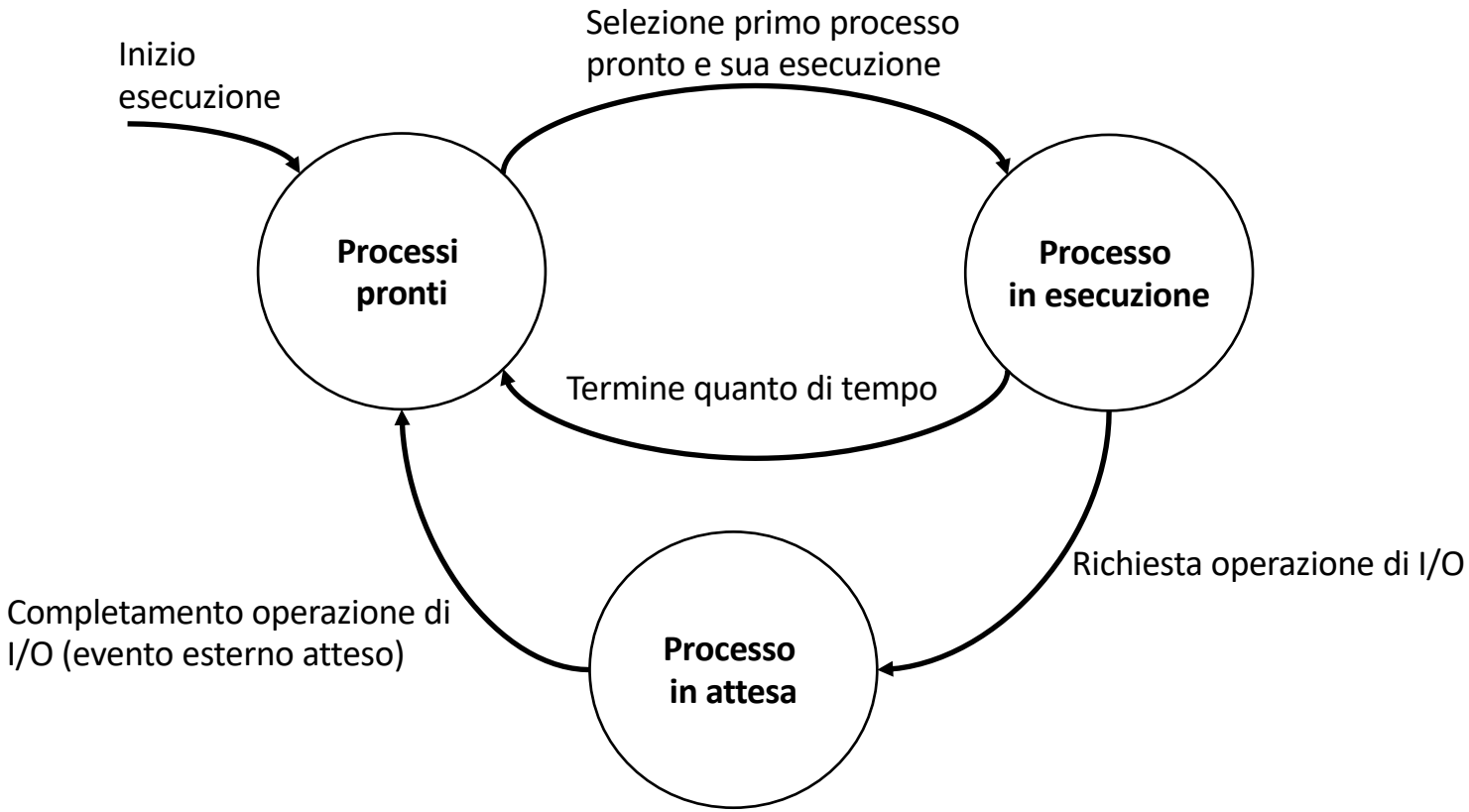


Attesa

- Se il processo richiede operazioni ad altri dispositivi (es. Operazioni di I/O), la CPU rimarrebbe inutilizzata
  - lo scheduler mette il processo in stato di attesa,
  - il dispatcher sceglie un nuovo processo tra i pronti dalla tabella,
  - quando l'operazione sarà finita, lo scheduler dichiarerà di nuovo pronto il processo

- Permette un utilizzo molto più efficiente delle risorse di elaborazione
  - Esempio:
    - durante la digitazione di un documento di testo, l'utente compie molte pause per riflettere sul contenuto che sta scrivendo
    - questo tempo è usato dal sistema per compiere altre operazioni in contemporanea (ad es. gestire la ricezione di e-mail)

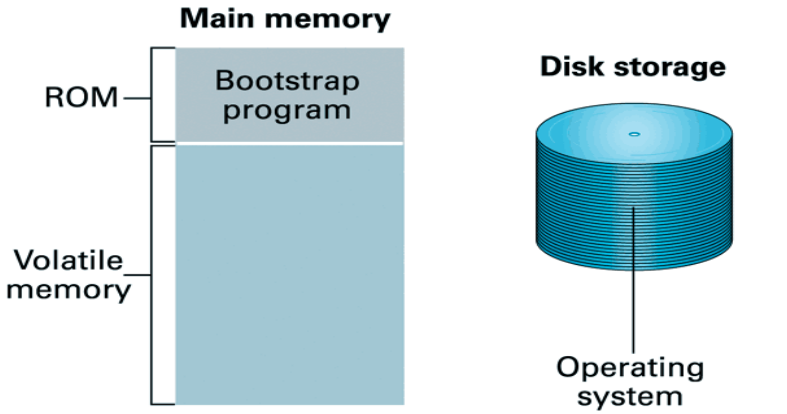
Stati di un processo nel multitasking



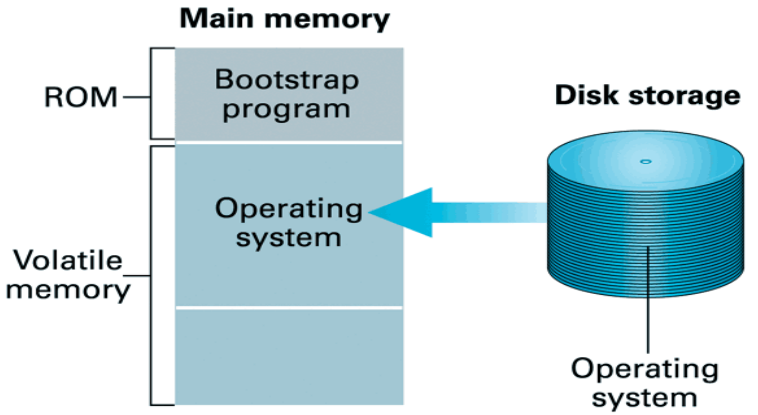
L’avvio del sistema operativo: Bootstrap

- All’inizio, la CPU ha un indirizzo fissato nel registro contatore di programma, che punta all’indirizzo nella ROM (Read Only Memory) ove inizia il programma di bootstrap che è sempre memorizzato lì
- L’esecuzione del programma di bootsrap trasferisce il kernel del Sistema Operativo da una parte prestabilita della memoria di massa (hard-disk, floppy-disk, CD-Rom,…) in memoria principale
- Quindi l’esecuzione prosegue con un salto all’area di memoria principale contenente il Sistema Operativo (che quindi viene mandato in esecuzione)
- Tra le prime operazione del kernel del Sistema Operativo vi sono tipicamente quelle di caricamento di altri componenti software:
  - driver delle periferiche installate
  - programmi di sistema lanciati automaticamente all’avvio

Bootstrap



Step 1: Machine starts by executing the bootstrap program already in memory. Operating system is stored in mass storage.



Step 2: Bootstrap program directs the transfer of the operating system into main memory and then transfers control to it.