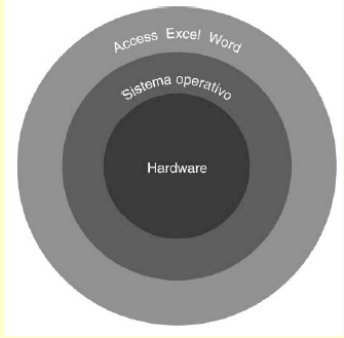
Il Sistema Operativo

Prof. Orazio Mirabella

Il sistema operativo

- Strato di interfaccia fra l'utente e l'hardware che permette di:
  - ➔Superare problemi legati alla gestione delle risorse hardware
  - ➔Favorire la condivisione delle risorse hardware, *regolandone l'accesso* da parte di utenti/programmi diversi

- Compito principale:
  - ➔Fornire un sistema *"virtuale"*, più semplice da usare rispetto l'hardware che si ha effettivamente a disposizione



Classificazione dei Sistemi Operativi

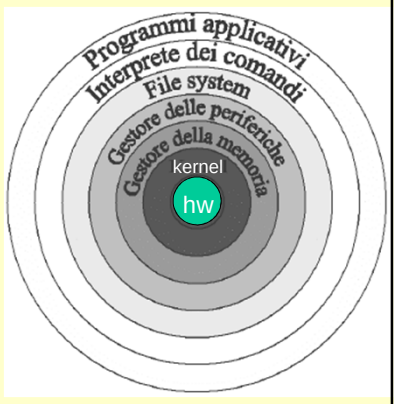
- **In base al numero di utenti:**
  - ➔**Mono-utente:** un solo utente alla volta può utilizzare il sistema
  - ➔**Multi-utente:** più utenti contemporaneamente possono interagire con la macchina.
    - ✓ nel caso di più utenti collegati, il sistema operativo deve fornire a ciascun utente l'astrazione di un sistema "dedicato". La versione più aggiornata di tale concetto è il «Cloud».
- **In base al numero di programmi in esecuzione:**
  - ➔**Mono-programmato:** il sistema può gestire l'esecuzione di al più un programma alla volta.
  - ➔**Multi-programmato:** il sistema operativo è in grado di portare avanti l'esecuzione di più programmi dando l'impressione della contemporaneità (sebbene ci sia una sola CPU).
    - ✓ nel caso di multi-programmazione il sistema operativo deve gestire l'unità di elaborazione (CPU) suddividendola tra i vari programmi.

Gestire Sicurezza e protezione

- Controllo degli accessi:
  - ➔Meccanismi per *l'identificazione degli accessi al sistema*
  - ➔Procedura di accesso al sistema: *login*
    - ✓A ogni utente è associato uno *username* e una *password*
- Protezione
  - ➔Ogni utente può accedere solo a determinati file e risorse
    - ✓Permessi di scrittura, lettura ed esecuzione
      - – *Administrator* (o *root*) utente privilegiato
        - » Accesso a qualsiasi file o risorsa
- Personalizzazione
  - ➔Ogni utente può configurare (nei limiti dei permessi ad esso associati) il proprio ambiente operativo

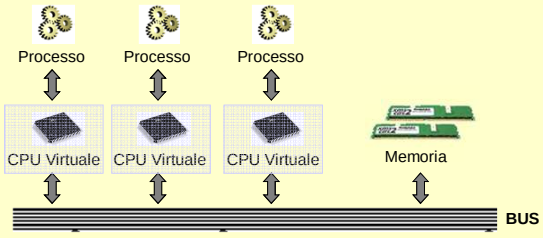
Architettura di un Sistema Operativo

- **Modello a strati gerarchici:**
  - **Struttura organizzata su diversi livelli**
  - **Gerarchie di macchine *virtuali* con il compito di gestire specifiche risorse fornendo *meccanismi logici di accesso*, che ne regolamentino l'uso e ne *mascherino* l'effettivo funzionamento**



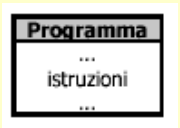
Il nucleo del sistema operativo (kernel)

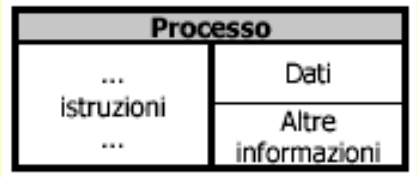
- **Compiti del kernel:**
  - Dialoga direttamente con l'hardware
  - Esegue i programmi utente
  - Risponde agli eventi (Interrupt) generati dalle periferiche
- **Requisiti Fondamentali:**
  - Consentire a utenti/processi diversi la condivisione delle risorse
  - Offrire **virtualmente** ad ogni utente/processo una CPU



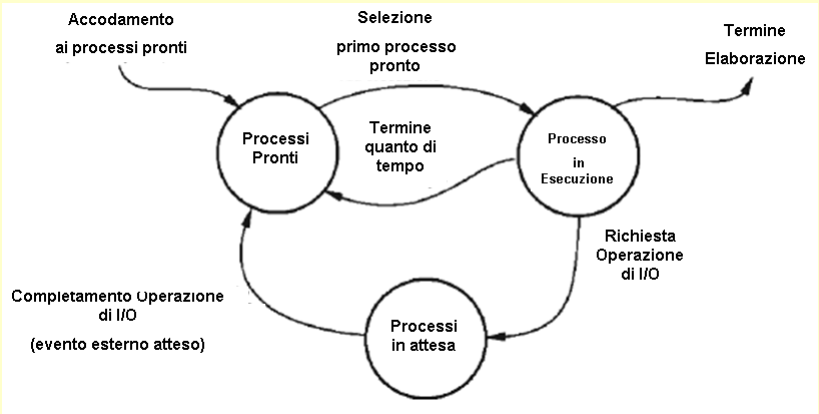
Programmi e processi

- Programma: entità statica
  - Memorizzato in genere su di un dispositivo di massa
- Processo: entità dinamica
  - Programma in esecuzione
  - Dati utilizzati dal programma
  - Informazioni relative al programma (contesto)
- Ad un programma possono corrispondere diversi processi
  - Copie contemporaneamente in esecuzione
- Un processo può a sua volta richiedere l'esecuzione di altri processi
  - processo padre
  - processi figli





Stati di un processo



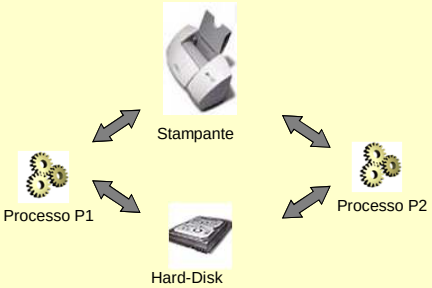
Politiche di scheduling dei processi

- Round-Robin
  - ➔Assegnare a rotazione la disponibilità di una unità di tempo (time slice) della CPU ai vari processi
    - ✓ coda *FIFO* (First In First Out)
  - ➔Un processo può anche rinunciare al tempo di CPU
    - ✓ attesa di I/O

- Round-Robin con priorità
  - ➔Ad ogni processo viene assegnata una *priorità*
    - ✓ viene scelto il processo con priorità massima
    - ✓ I processi ad uguale livello di priorità vengono trattati con politica Round-Robin FIFO

Le Problematiche di Concorrenza

- Problemi legati alla virtualizzazione delle risorse
  - ➔ Starvation: un processo non riesce mai ad accedere ad una risorsa
    - ✓ Nel caso di scheduling con priorità, un processo a bassa priorità potrebbe non riuscire mai a guadagnare la CPU
  - ➔ Deadlock: più processi bloccati a vicenda
    - ✓ Il processo P1 ha ottenuto l'accesso esclusivo alla stampante
    - ✓ P1 è in attesa di poter accedere al disco (dove risiedono i dati da stampare)
    - ✓ Il disco è però a sua volta controllato in maniera esclusiva dal processo P2.
    - ✓ P2 rilascerà il disco dopo essere riuscito a ottenere l'accesso alla stampante.

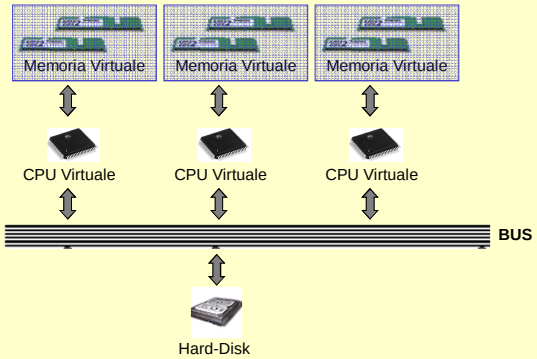


Il Gestore della memoria

- Spazio di indirizzamento virtuale
  - ➔I processi possono *ignorare l'effettiva collocazione fisica* del codice e dei dati in memoria
- *Protezione* della memoria
  - ➔I dati e le istruzioni dei programmi vengono protette, in modo che nessun altro processo possa leggerle o modificarle
- *Condivisione* della memoria
  - ➔Permettere, in modo controllato, la parziale sovrapposizione degli spazi di memoria dei vari processi

Gestore della memoria

- *Memoria vituale*
  - ➔Gli strati di livello superiore posso lavorare come se avessero a disposizione l'intera memoria centrale



Rilocazione del codice

- Output del *linker:* codice in linguaggio macchina in cui tutti i nomi simbolici e i riferimenti sono stati espressi come *indirizzi di memoria*

```
#include <stdio.h>

int a,b,c;

void main() {
    a = 3;
    b = 4;
    c = a + b;
    printf("%d+%d=%d\n",a,b,c);
}
```

Elemento	Indirizzo di memoria
c	103
b	102
a	101
Codice	100
	...
	0

➔ Questo spazio di memoria (logico) non coincide necessariamente con la memoria in cui risiederà il programma durante l'esecuzione (spazio fisico)

✓ È necessario rilocare il codice del programma

Esempio: memoria libera a partire dall'indirizzo 4096

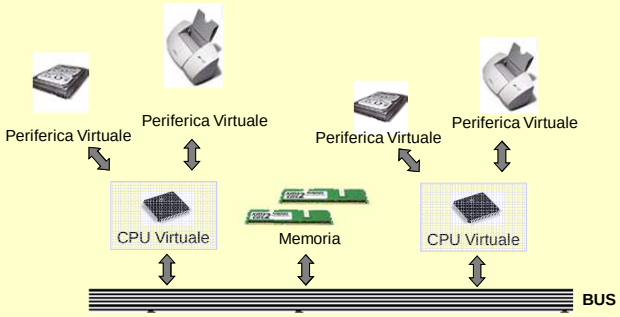
Elemento	Indirizzo iniziale	Indirizzo rilocato
c	103	4199
b	102	4198
a	101	4197
Codice	100	4196
	...	...
	0	4096

Rilocazione del codice

- Statica
  - ➔Eseguita direttamente dal linker
    - ✓ È necessario conoscere in anticipo in quale parte della memoria sarà caricato il programma
- Dinamica
  - ➔È una necessità nel caso della multi-programmazione
  - ➔Eseguita dal SO prima dell'esecuzione del programma

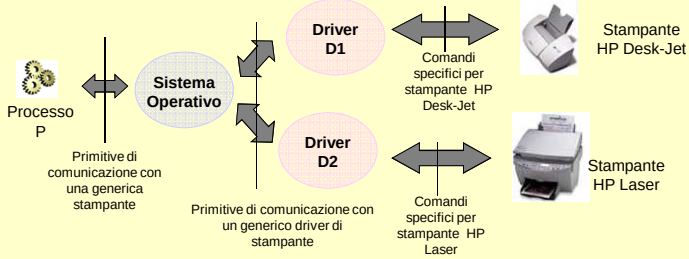
Gestore delle periferiche

- Periferiche astratte: Le *caratteristiche fisiche* delle periferiche e le operazioni di I/O che le coinvolgono vengono *mascherate*
- Vengono esposte una serie di *primitive a livello più alto* per leggere e scrivere i dati
- Ogni processo si trova ad operare con periferiche virtualmente dedicate solo ad esso
- *Gestione delle problematiche relative ai conflitti di accesso*



Gestore delle periferiche

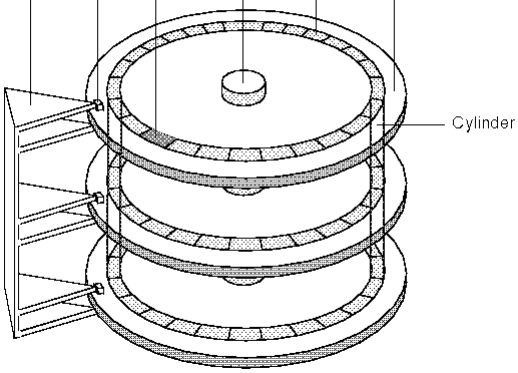
- Drivers
  - ➔Programmi per la gestione delle periferiche
  - ➔Inclusi nel sistema operativo
    - ✓ Spesso sono realizzati e forniti dai produttori delle periferiche stesse
  - ➔Nascondono al programma applicativo e al resto del SO l'effettiva modalità con cui avviene lo scambio dei dati con le periferiche
    - ✓ Generalmente tale modalità è diversa per ogni tipo di dispositivo



Cosa è l'Hard disk

- E' una memoria di massa non volatile che immagazzina grandi quantità di dati sotto forma di bolle magnetiche microscopiche.

La gestione dei dischi è compito del File System

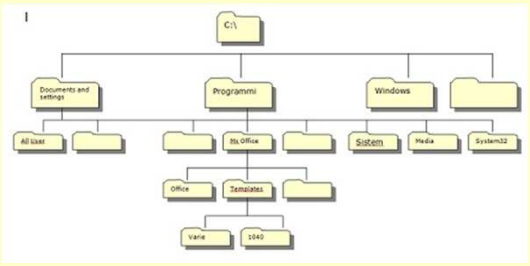


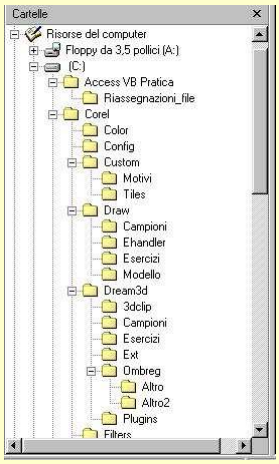
Gestore dei files (filesystem)

- Funzioni principali di un File System:
  - ➔ Fornire un meccanismo di identificazione univoco dei files
  - ➔ Implementare i meccanismi per accedere ai files
  - ➔ Realizzare metodi per il controllo d'accesso ai files
  - ➔ Allocare spazio su disco per la memorizzazione dei files
  - ➔ Deallocare e Riutilizzare lo spazio su disco con l'operazione di cancellazione
  - ➔ Fornire un'interfaccia utente per: creazione, cancellazione, spostamento, ispezione di file e directory
  - ➔ mascherare le caratteristiche fisiche dei dispositivi di memorizzazione e delle interfacce

La Directory

La Directory è strutturata come un albero con rami e foglie. Le sottocartelle permettono di gestire in ordine le informazioni.





Gestore dei files (filesystem)

- Il filesystem windows:
  - ➔Nei Floppy Disk: utilizza il settore come unità minima di allocazione
  - ➔Negli Hard Disk: non utilizza il settore come unità di allocazione, ma il *cluster*.
    - ✓ gruppo di n settori (n = 2, 4, 8, 16, 32)
  - ➔Componenti principali:
    - ✓ Partition Table
      - – nel boot sector della partizione stessa e contiene le informazioni sulla partizione.
    - ✓ Directory Table
      - – contiene informazioni sui file e le sottodirectory contenute in una directory
    - ✓ FAT (File Allocation Table)
      - – permette di individuare i cluster occupati da un file
      - – è il “cuore” del filesystem, per sicurezza essa viene duplicata per proteggerla da cancellazioni o danneggiamenti accidentali

Interazione con l'utente

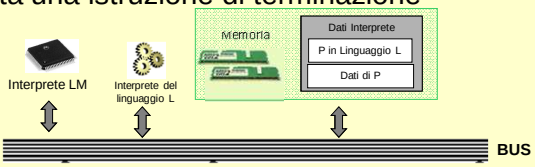
- Interprete dei comandi
  - *Riceve i comandi* tramite i dispositivi di input
  - *Esegue i programmi* associati a tali comandi
    - Lettura della memoria di massa del programma da eseguire:
      - Usa il Filesystem
    - Allocazione della memoria centrale necessaria e caricamento del programma
      - Usa il Gestore della memoria
    - Creazione, attivazione e gestione del processo
      - Usa il Kernel
- Interfacce utente
  - A caratteri
  - Grafiche: GUI (Graphical User Interface)

Ambiente di programmazione

- l'insieme dei programmi che consentono la scrittura e la verifica di nuovi programmi
  - **Editor**
    - serve per la costruzione di file che contengono testi. In particolare tramite un editor si scrive il *programma sorgente*.
  - **Compilatori**
    - accettano in ingresso l'intero programma e producono in uscita la rappresentazione dell'intero programma in linguaggio macchina (versione oggetto).
  - **o Interpreti**
    - traducono ed eseguono direttamente ciascuna istruzione del *programma sorgente*, istruzione per istruzione.
  - **Linker (solo per compilatori)**
    - nel caso in cui il programma sia suddiviso in moduli (*oggetto*) separati provvede a collegarli per formare un unico *programma eseguibile.*
  - **Debugger**
    - serve per scoprire ed eliminare errori presenti durante l'esecuzione di un programma, ma non rilevati in fase di compilazione.

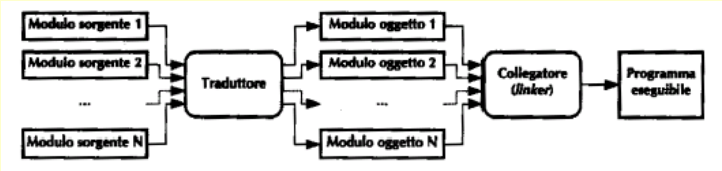
Interprete

- Funzionamento di un interprete:
  - Preleva un'istruzione I del programma P scritto nel linguaggio L
  - Decodifica I
  - Traduce I in una serie di istruzioni in linguaggio macchina M_1, M_2, … , M_n
  - Esegue M_1, M_2, … , M_n
  - Passa all'istruzione successiva di P fino a quando non si sia raggiunta una istruzione di terminazione



Compilazione e Linking

- È possibile suddividere il programma sorgente in diverse parti
  - Moduli sorgente
- **Fase di compilazione:**
  - Ogni modulo sorgente viene compilato producendo l'equivalente in linguaggio macchina
    - ✓ Modulo oggetto
  - Eventuali riferimenti a dati o routine di altri moduli vengono raggruppati
    - ✓ Tabelle dei simboli
- **Fase di linking:**
  - I moduli oggetto vengono collegati risolvendo i riferimenti contenuti nella tabella dei simboli, producendo un unico programma eseguibile



Compilazione vs Interpretazione

- I programmi commerciali sono solitamente compilati
  - ➔ Maggior velocità di esecuzione
  - ➔ Protezione del codice sorgente
- Con l'avvento di Internet è stato riavvivato l'interesse per gli interpreti
  - ➔ *JavaScript, VBScript*
- Soluzioni miste
  - ➔ Visual Basic, Java, .NET

Compilazione	Interpretazione
La traduzione viene effettuata una volta sola, prima dell'esecuzione: l'esecuzione è molto più veloce	La traduzione viene effettuata durante l'esecuzione: la stessa istruzione può venire tradotta molte volte
È possibile eseguire ottimizzazioni del codice	È praticamente impossibile ottimizzare il codice
Maggiore occupazione di memoria del programma	Minore occupazione di memoria (i linguaggi di alto livello sono più compatti)
Per ogni modifica al programma sorgente, la compilazione deve essere eseguita nuovamente	Le modifiche al programma sorgente non richiedono tempi aggiuntivi