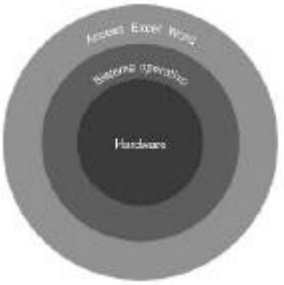
Sistema Operativo Compilatore

Maurizio Palesi
Salvatore Serrano

Il Sistema Operativo

- Sistema operativo: strato di interfaccia fra l'utente e l'hardware che permette di:
  - ➔ Superare problemi legati alle limitazioni delle risorse hardware
  - ➔ Favorire la condivisione delle risorse hardware, regolandone l'accesso da parte di utenti/programmi diversi
- Compito principale di un sistema operativo:
  - ➔ 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:
  - 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"
- 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 contemporanea di più programmi (mantenendo una sola CPU).
  - nel caso di multi-programmazione il sistema operativo deve gestire l'unità di elaborazione (CPU) suddividendola tra i vari programmi.

Sicurezza e Protezione

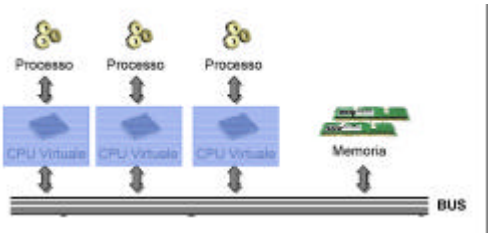
- Controllo degli accessi
  - I sistemi operativi multi-utente possiedono meccanismi per *l'identificazione degli accessi al sistema*
    - ✓ Procedura di accesso al sistema: *login*
      - A ogni utente è associato uno *username* e una *password*
  - Protezione
    - ✓ Ogni utente può accedere solo 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

- Un moderno S.O. è organizzato secondo una architettura “a strati” (a cipolla)
- Ogni strato implementa una macchina virtuale più potente della 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

Nucleo del Sistema Operativo (Kernel)

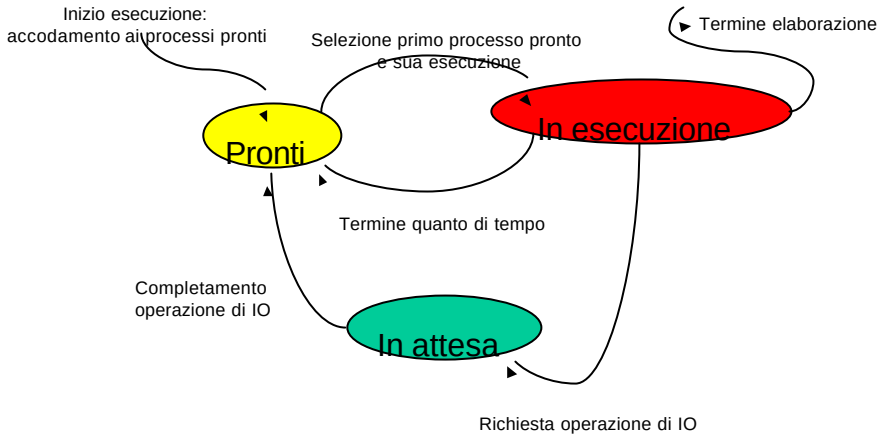
- Compiti del kernel:
  - ➔ Esegue i programmi e risponde agli eventi generati dalle periferiche
- Requisito Fondamentale:
  - ➔ consentire a utenti/programmi diversi la condivisione delle risorse
  - ➔ Offrire virtualmente ad ogni utente/programma tutta la macchina
    - ✓ Come se si avessero a disposizione tante “CPU virtuali”



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
    - ✓processo figlio

Stati di un processo



Politiche di scheduling di un processo

- Round-Robin
  - Assegnare a rotazione la disponibilità di un'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 FIFO
- Esempio: Windows 2000
  - Round-Robin con priorità (con alcune varianti)
  - Quanto: 13-30ms

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 ed è in attesa di poter accedere al disco (dove risiedono i dati da stampare), che 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.

Comunicazione tra i processi

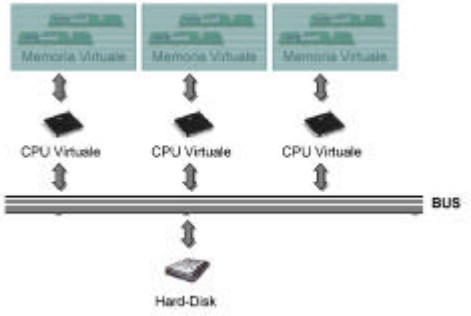
- Scambio di dati tra i processi
  - ➔Mediante uno spazio di memoria comune (*shared memory*)
  - ➔Mediante scambio di messaggi
- Sincronizzazione
  - ➔Metodi per il controllo degli eventi (*semafori*)
    - ✓Un processo P1 deve attendere che P2 abbia prodotto determinati risultati prima di iniziare una determinata elaborazione

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

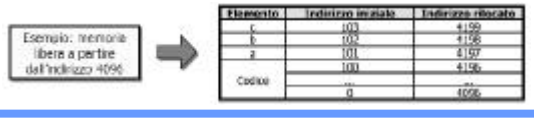


Rilocazione del codice

- Output del *linker*
  - ➔Codice in linguaggio macchina in cui tutti i nomi simbolici e i riferimenti sono stati risolti (espressi come *indirizzi* di memoria)
  - ➔Questo spazio di memoria (logico) non coincide necessariamente con la memoria in cui risiederà il programma durante l'esecuzione (spazio fisico)
    - ✓E' necessario rilocare il codice del programma

Codice non rilocato prodotto dal *linker*

Codice rilocato in memoria, pronto per l'esecuzione

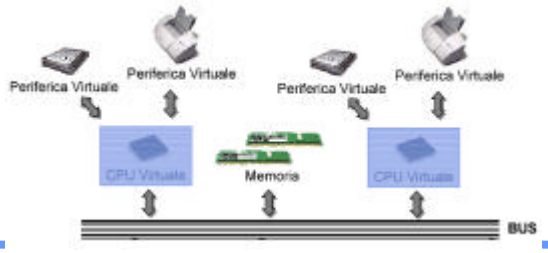


Rilocazione del codice

- Statica
  - ➔Eseguita direttamente dal linker
  - ➔È necessario conoscere in anticipo in quale parte della memoria sarà caricato il programma
- Dinamica
  - ➔È una necessità nei casi della multi-programmazione
  - ➔Eseguita continuamente durante l'esecuzione di ogni istruzione del programma

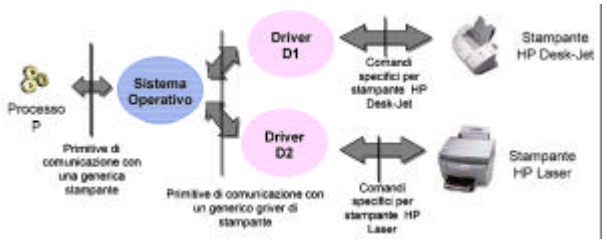
Gestore delle periferiche

- Periferiche astratte
  - ➔Le caratteristiche fisiche delle periferiche e le operazioni di IO 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



Gestore dei file (filesystem)

- Funzioni principali di un FS:
  - Fornire un meccanismo di identificazione univoca dei file
  - Implementare i meccanismi per accedere ai file
  - Realizzare metodi per il controllo di accesso ai file
  - Allocare spazio su disco per la memorizzazione dei file
  - Deallocare spazio su disco con l'operazione di cancellazione
  - Fornire un'interfaccia utente per:
    - ✓ Creazione, cancellazione, spostamento, ispezione di file e directory
- I più comuni sono
  - FAT 16: Windows3.x, Windows95
  - FAT 32: Windows95(S), Windows98, Windows ME
  - NTFS (New Technology File System): Windows NT, Windows 2000 Pro e Server
  - Ext2: Unix, Linux

Gestore dei files

- Componenti principali del filesystem windows:
  - Ogni disco è suddiviso in unità di allocazione chiamate *cluster*
  - Partition Table
    - ✓ Si trova nel Master Boot Record di ogni HD (il primo settore di ogni disco) e contiene le informazioni sulle partizioni
  - Directory Table
    - ✓ Contiene le informazioni sui file e sulle sottodirectory contenute in una directory
  - FAT (File Allocation Table)
    - ✓ Permette di individuare i cluster occupati da un file
      - L'indice della tabella rappresenta il numero di cluster
      - Il contenuto è il numero del successivo cluster occupato da un dato file
    - ✓ È il "cuore" del filesystem, per sicurezza viene duplicata per proteggerla da cancellazioni o danneggiamenti accidentali

Interazione con l'utente

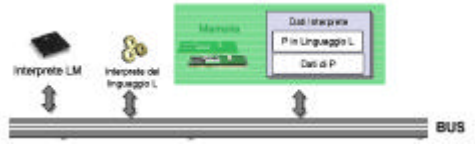
- Interprete dei comandi
  - Riceve i comandi tramite i dispositivi di input
  - Esegue i programi assocciati a tali comandi
    - ✓ Lettura della memoria di massa del programma da eseguire
      - Filesystem
    - ✓ Allocazione della memoria centrale necessaria e caricamento del programma
      - Gestore della memoria
    - ✓ Creazione, attivazione e gestione del processo
      - Kernel
- Programmi di utilità
  - Come l'interprete dei comandi sono direttamente visibili all'utente
    - ✓ Compilatori, editor, programmi di backup, utility di deframmentazione etc. etc.
- Interfacce utente
  - A caratteri
  - Grafiche: GUI (Graphical User Interface)

Ambiente di programmazione

- CPU: esegue programmi scritti in linguaggio macchina
  - E' possibile codificare qualsiasi algoritmo in linguaggio macchina
- Ambiente di Sviluppo
  - 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.
  - Traduttore
    - ✓ opera la traduzione di un programma sorgente scritto in un linguaggio ad alto livello in un programma oggetto scritto in linguaggio macchina
    - ✓ Compilatori: accettano in ingresso l'intero programma e producono in uscita la rappresentazione dell'intero programma in linguaggio macchina.
    - ✓ Interpreti: traducono ed eseguono direttamente ciascuna istruzione del programma sorgente, istruzione per istruzione.
  - Linker
    - ✓ 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.
- Linguaggi ad alto livello
  - Minore tempo di sviluppo dei programmi
  - Maggiore riusabilità del codice
  - Modularità e standardizzazione nella progettazione del software

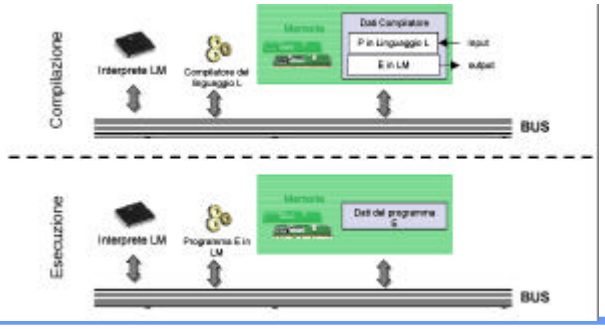
Interprete

- 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



Compilatore

- Funzionamento di un compilatore
  - ➔Traduce il programma P (*sorgente*) scritto in liguaggio L in un programma E (*eseguibile*) equivalente in linguaggio macchina
  - ➔La traduzione viene eseguita una sola volta
  - ➔Il programma E può essere eseguito più volte



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

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

Compilatore

- 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