# Infrastrutture Software

- I componenti fisici di un sistema informatico sono resi accessibili agli utenti attraverso un complesso di strumenti software finalizzati all'utilizzo dell'architettura.
- Si tratta di Software (programmi) di base generalmente fornito insieme al Hardware.
- L'insieme di tali programmi di base è indicato con il termine sistema operativo (S.O.).

# Le funzioni del S.O.

- Il S.O. è un interfaccia tra l'utente e l'hardware che facilita l'utilizzo delle risorse di un sistema. In particolare deve svolgere le seguenti funzioni:
  - esecuzione di applicazioni
  - accesso ai dispositivi di I/O
  - archiviazione di dati e programmi
  - controllo d'accesso
  - contabilizzazione
  - gestione dei malfunzionamenti

# Elementi di un S.O.

- Generalmente l'insieme dei programmi costituenti il S.O. permette di gestire:
  - il processore (quali programmi eseguire e quali compiti assegnare alla CPU, traduzione dei programmi ad alto livello al linguaggio assembly mediante il supporto di compilatori e interpreti)
  - la memoria (allocazione della memoria per i diversi programmi in esecuzione, definizione di zone riservate di memoria per l'archiviazione dei dati dei programmi)
  - le periferiche (accesso ai dispositivi di I/O, si mascherano i dettagli di basso livello, gestione dei conflitti)

# Elementi di un S.O.

- il file system (archiviazione e reperimento dei dati in memoria di massa)
- gli utenti e i relativi interpreti comandi (accesso semplice alle funzionalità disponibili)

I gestori di memoria, processore, file system, periferiche e utenti interagiscono fra loro per coordinare l'accesso alle risorse da parte di utenti e SW applicativo.

# Programmi applicativi

- Sono programmi utilizzati dall'utente per svolgere attività di alto livello (elaborazione di testi e grafica, giochi, fogli di calcolo,...)
- non fanno parte dei programmi del S.O. In particolare,
  - possono far riferimento ad un set ristretto di istruzioni del processore (modalità utente vs modalità supervisore)
  - non possono decidere autonomamente quando e come avere accesso alle risorse. È il S.O. stesso a fornire loro l'accesso alle risorse.

# Tassonomia degli utenti

- Gli utenti di un S.O. possono essere
  - programmatori di sistema
  - amministratori di sistema
  - utenti applicativi

# Un po' di storia dei S.O.

- sistemi senza S.O: i programmi applicativi dovevano anche contenere delle istruzioni per la gestione a basso livello dei dispositivi fisici.
- I programmi erano caricati in memoria tramite un dispositivo di I/O in grado di leggere schede perforate contenenti la codifica in linguaggio macchina di dati e programmi.
- Pessima gestione delle risorse: si doveva prenotare la macchina al fine di stabilire un ordine di esecuzione dei programmi (Job).
- La macchina consentiva l'esecuzione di un solo programma alla volta.
- Perdita di tempo per la preparazione dell'ambiente di supporto all'esecuzione.

# Un po' di storia dei S.O.

- Sistemi batch  (o a lotti): sono dotati di una componente software (un S.O. molto rozzo) chiamata monitor in grado di automatizzare l'esecuzione dei programmi. Il monitor è dotato di un linguaggio di controllo chiamato Job Control Language.

- Si esegue un lotto di programmi (jobs) alla volta.

- Non c'è nessuna interazione con l'utente. (ottimi per attività con bassa interazione: esempio stampa estratti conto).

- Efficiente utilizzo delle risorse

- Tempi di attesa notevoli per ottenere i risultati dell'elaborazione (si deve attendere la terminazione dell'esecuzione di un intero lotto di job).

# Un po' di storia dei S.O.

- Nei primi sistemi informatici mancava
  - interattività
  - protezione della memoria
  - temporizzazione dell'esecuzione (al fine di evitare il monopolio dell'intero sistema da parte di un solo programma)
  - gestione dell'input/output i programmatori dovevano scriversi le loro funzioni a basso livello per l'input e l'output)
  - sfruttamento parallelo delle risorse

# Un po' di storia dei S.O.

- sistemi uniprogrammati
  - in memoria può risiedere al più un programma (processo) oltre al S.O.
  - es. MS-DOS
  - ridotto uso della CPU
- sistemi multiprogrammati
  - più progammi contemporaneamente in memoria oltre al S.O.
  - es. Windows, MacOS, Unix,...
  - Uso più efficiente della CPU

# Un po' di storia dei S.O.

- I sistemi a partizione di tempo (time sharing)
  - simulano un quasi-parallelismo nell'accesso alle risorse
  - Supponiamo di avere diversi programmi in memoria, ognuno con un diverso tempo di esecuzione. Per evitare che un un programma utilizzi in modo esclusivo la CPU, il tempo viene suddiviso in unità elementari, dette quanti, da assegnare ai vari processi secondo una possibile politica (per esempio a rotazione, politica round-robin).
  - La dimensione del quanto di tempo incide fortemente sulle prestazioni del sistema.

# Processi

- Nei sistemi attuali, più copie dello stesso programma si possono eseguire concorrentemente.
- Processo = copia di un programma in esecuzione + dati necessari all'esecuzione.
- A un programma può corrispondere più di un processo (processi figli e processi padre, es. videoscrittura + stampa).
- Un S.O. è un'infrastruttura SW di supporto all'esecuzione concorrente di più processi.

# Modelli organizzativi di S.O.

- Ogni S.O. può essere caratterizzato funzionalmente da
- un nucleo (kernel)
- un insieme di processi di servizio.
- Modello monolitico vs modello a strati

# Modello monolitico

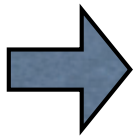
- Il S.O è costituito da un unico processo kernel che provvede alla gestione di tutti i servizi di tutte le risorse del sistema (processore, memoria, periferiche, file system, ...)
- gli utenti e i processi applicativi interagiscono con il kernel attraverso l'esecuzione di servizi (chiamate di sistema).
- si hanno due modalità di funzionamento
  - modalità utente: i processi eseguiti in questa modalità non possono accedere a tutte le risorse liberamente. Es. vietato interagire direttamente con i dispositivi di I/O.
  - modalità supervisore (kernel mode): usata dal kernel per implementare le funzionalità necessarie. Nessun limite nelle operazioni effettuabili.

# Modello a strati

- consiste in un nucleo piuttosto semplice che gestisce solamente il processore e l'esecuzione dei processi. Interagisce direttamente con l'hardware.
- Una serie di processi di servizio standard organizzati su diversi livelli (macchine virtuali).
  - ogni processo di un livello implementa delle funzionalità facendo uso dei servizi messi a disposizione dal livello inferiore
- È un'organizzazione che favorisce la portabilità del S.O. su diverse architetture HW.

# Esempio

- HW
- Nucleo
- Gestore memoria
- Gestore periferiche
- Gestore file
- Interprete comandi



S.O.

- Applicazioni utente (Utility di supporto, debugger, compilatori, editor di sistema...)

# Traduzione dei programmi

- La CPU si comporta da esecutore per programmi scritti in linguaggio macchina (assembly)
  - E' un linguaggio a basso livello difficile da usare.
  - Soluzione: usare un linguaggio L di alto livello più potente ed espressivo. Utilizzare degli opportuni software per tradurre i programmi scritti in L in programmi assembly.
    - Interpreti
    - Compilatori

# Gestione dei processi

- Nelle architetturre HW moderne si tende ad elaborare l'informazione in maniera parallela.
- 3 livelli di parallelismo
  - parallelismo a livello di dati
  - parallelismo a livello di istruzioni
  - parallelismo a livello di programmi
- i primi due livelli di parallelismo sono gestibili solo mediante architetture HW parallele (molte CPU), il modello di Von Neumann può implementare solo il parallelismo a livello di programmi.
- il gestore dei processi ha il compito di implementare in maniera efficiente  il parallelismo a livello di programmi.

# Gestione dei processi

- Il componente in grado di eseguire i processi è la CPU (processore)
- Se abbiamo un unico processore il parallelismo è simulato.
- Un processo può trovarsi in un uno dei seguenti stati:
  - in esecuzione: ovvero il processo ha a disposizione il processore per l'esecuzione del proprio codice
  - pronto: ovvero il processo è in grado e in attesa di essere eseguito, non appena il processore diviene disponibile (i processi in questo stato vengono generalmente posti in coda, FIFO policy)
  - in attesa: ovvero il processo non è in grado di essere eseguito perché in attesa di qualche evento esterno (es. input da tastiera)

# Evoluzione di un processo

1. il processo viene creato e messo nella coda dei processi pronti

2. il primo processo nella coda dei processi pronti viene posto in stato di esecuzione; se la coda era inizialmente vuota, il processo in esecuzione coincide con quello creato al punto precedente

3. il processo in esecuzione ha la piena disponibilità del processore fino al verificarsi di uno dei seguenti eventi:

    a. scade il quanto di tempo a disposizione del processo; il processo viene rimesso nella coda dei processi pronti
    b. il processo effettua un'operazione I/O; il processo è messo in stato d'attesa
    c. il processo termina

4. in base a quale evento si è verificato in 3, il processo evolve nel seguente modo:

    a. il processo attenderà il suo turno nella coda per essere nuovamente eseguito
    b. il processo viene rimesso nella coda dei processi pronti, una volta che l'operazione di I/O è terminata
    c. il processo è rimosso dall'elenco dei processi esistenti

# Problemi del parallelismo

- Morte per fame (Starvation): un processo in stato di pronto non viene mai eseguito.
  - non accade nel caso di politica FIFO, può accadere nel caso di utilizzo di code con priorità.
- Blocco critico (Deadlock): un insieme di processi rimane permanentemente bloccato. Es. due processi A e B. A richiede i dati di B per essere eseguito, B richiede i dati di A per essere eseguito. Oppure, mai invitare 5 filosofi a cena!!!
  - Tecniche di verifica statica per evitare deadlock
  - Tecniche di eliminazione dei deadlock a runtime

# Gestione della memoria

- Nei S.O. multiprogrammati possiamo avere molti processi in memoria centrale
- Problema: la memoria centrale è tipicamente di piccole dimensioni (Max 1Gb).
- Si rende necessaria l'implementazione di un gestore della memoria che scarichi in memoria di massa i processi poco utilizzati e che inoltre permetta di
  - ridurre la necessità di spazio mantenendo in memoria solo parte delle istruzioni e dei dati del processo.
  - ridurre la necessità di spazio condividendo il codice fra diversi processi che corrispondono allo stesso programma.
    (segmentazione della memoria: segmento codice, segmento dati)

# Gestione della memoria

- il gestore della memoria permette di trasferire il contenuto di un'area di memoria centrale in un'area di memoria di massa chiamata memoria di swap.
- si trasferiscono nell'area di swap i processi in attesa e alcuni dei processi in stato di pronto al fine di liberare memoria centrale.
- Per lo scarico nell'area di swap si segue il principio di località.

# Gestione delle periferiche

- Il gestore delle periferiche mette a disposizione delle funzionalità di lettura/scrittura da/verso dispositivi periferici di alto livello, indipendenti dalla struttura HW delle periferiche.
- Tali comandi di alto livello sono realizzati facendo uso di meccanismi HW e SW di basso livello quali
  - driver (programmi SW destinati alla gestione delle periferiche)
  - controller (interfacce di I/O HW per il trasferimento di dati gestiti generalmente via interrupt)

# Gestione delle periferiche

- I driver permettono
  - di astrarre le operazioni di lettura/scrittura
  - permettono di implementare operazioni di scrittura/lettura a basso livello affidabili (se un'operazione non va a buon fine viene ripetuta, nel caso in cui no sia possibile eseguirla viene segnalato il malfunzionamento)
  - di virtualizzare la presenza di più periferiche (spooling)
    - esempio: la coda di stampa

# Interfacce di I/O

- il calcolatore comunica con l'ambiente esterno (le periferiche) mediante delle interfacce di ingresso/uscita, che hanno il compito di tradure i segnali che giungono dal calcolatore in informazioni comprensibili alle periferiche e vice versa.
- La trasmissione dei dati può essere
  - seriale (l'informazione è trasmessa un bit per volta)
  - parallela (più bit trasmessi in parallelo)
- Alcuni standard
  - RS-232C, USB, FireWire (tx seriale)
  - Centronics (tx parallela), ATA

# Interfacce di I/O

- Ogni interfaccia di I/O è dotata almeno dei seguenti registri
  - Registro dati, utilizzato per scambiare i dati tra periferica e calcolatore. Connesso con il bus dati.
  - Registro di stato (o di controllo), nel quale transitano informazioni di controllo necessarie alla sincronizzazione tra CPU e periferica. Connesso con il bus di controllo.

# Sincronizzazione

- Periferiche e CPU hanno generalemnte diverse velocità e necessitano di sincronizzazione.
- Ci sono tre diversi metodi di sincronizzazione:
  - a controllo di programma
  - a interruzione
  - con accesso diretto alla memoria

# Sync. a controllo di programma

- La sincronizzazione è completamente gestita dalla CPU.
- La CPU esegue un ciclo (detto ciclo di polling) che ispeziona/scrive periodicamente il registro dati.
- Esempio: stampa di una linea di caratteri mediante una stampante, ogni singolo carattere viene trasferito alla stampante mediante il registro dati, solo quando un carattere è stato stampato viene trasferito nel registro dati il seguente.
- Vantaggio: è una gestione della sincronizzazione semplice.
- Svantaggio: rischio di sovraccarico della CPU

# Sync. a interruzione

- Elimina il problema di sovraccarico della CPU tipico della sincronizzazione a controllo di programma.
- Ogni interfaccia è dotata della possibilità di notificare il suo status alla CPU mediante un segnale chiamato interruzione (o interrupt).
- Quando la CPU riceve un interrupt, interrompe la sua attività ed esegue un programma di risposta all'interruzione per gestire la comunicazione con l'interfaccia (e quindi con la periferica).
- La CPU è occupata solo per il trasferimento dei dati. (si evitano i tempi di attesa del ciclo di polling)

# Sync. con accesso diretto alla memoria

- Se si hanno grossi e frequenti trasferimenti di dati, la gestione della sincronizzazione mediante interrupt rischia di essere inefficiente.
- Esistono delle componenti HW chiamate DMA (Direct Memory Access) che sostituiscono la CPU nella gestione del trasferimento dati.
- La CPU controlla i DMA, comunica loro solo l'indirizzo di memoria da cui leggere o sul quale scrivere e la quantità di dati da trasferire, dopodiché il DMA gestisce l'intero processo di trasferimento.
- Nelle architetture più sofisticate i DMA sono processori dedicati all'input/output.

# Il File System

- È la componente del S.O. che si occupa della gestione della memoria di massa e dell'organizzazione logica dei dati
- Le operazioni supportate da un file system sono:
  - recupero dei dati precedentemente memorizzati
  - eliminazione di dati
  - modifica dei dati
  - copia dei dati fra diversi dispositivi di memoria di massa

# Il File System

- I dati sono memorizzati in contenitori logici detti file. Ogni file è identificato da un nome (il nome può essere affiancato da un'estensione che individua il tipo di file)
- Per una miglior organizzazione dei dati, i file possono essere memorizzati in contenitori detti directory o cartelle.
- il file system è generalmente organizzato secondo una struttura ad albero.
- Possiamo individuare un file fornendo il percorso assoluto o relativo all'interno dell'albero del file system.
- L'accesso all'informazione può avvenire attraverso opportuni comandi testuali o grafici

# Organizzazione fisica dei dati

- I file sono memorizzati sul supporto fisico (dispositivi di memoria di massa) in blocchi di piccole dimensioni.
- I blocchi potrebbero essere non contigui!
- Due tecniche di memorizzazione
  - File Allocation Table (FAT)
  - I-nodes

# FAT

- L'associazione tra nome del file e blocchi fisici che contengono il file viene gestita mediante una lista concatenata: a partire dal primo blocco, in coda ad ogni blocco di dati viene riportato il successivo.
- Tale struttura viene memorizzata in un'area del disco chiamata FAT (File allocation Table)
- Per ogni file, nella FAT è mantenuto un descrittore di file contenente i seguenti dati
  - indirizzo del primo blocco del file
  - data e ora di modifica, dimensioni,...
- L'organizzazione mediante FAT funziona bene per file di piccole dimensioni
- Usata in Windows 95/98

# I-NODES

- L'organizzazione in blocchi è basata su delle strutture dati chiamate i-node, le quali possono mantenere sia informazioni inerenti al descrittore del file sia inerenti ai blocchi di dati.
- Un i-node può contenere informazioni inerenti ai blocchi di dati quando il file è piccolo, altrimenti può indicare insiemi di altri inode.
- Ai file di grandi dimensioni è associato un albero di i-nodes, le cui foglie contengono i riferimenti ai blocchi di dati.
- La struttura ad albero rende più efficiente l'accesso ai dati.
- È una metodologia di memorizzazione tipicamente usata nei sistemi UNIX.

# Controllo degli accessi

- I S.O. multiutente possiedono meccanismi di identificazione degli accessi al sistema.
- La modalità più usata prevede di associare ad ogni utente un account (i.e. nome) e una password.
  - in questo modo è possibile
    - personalizzare l'ambiente operativo di ogni utente
    - facilitare la distribuzione dei costi di gestione
    - fornire una diversa visibilità del sistema a differenti tipi di utente

# Controllo degli accessi

- Oltre a fornire un controllo sugli utenti è anche possibile stabilire un sistema di protezione dei file.

- Esempio: nei sistemi Unix, si possono distinguere 3 tipi di utenti: il proprietario di un file, il gruppo di utenti a cui appartiene il propietario del file e il resto del mondo.

    - Per ogni tipologia di utente possiamo abilitare le operazioni di lettura (R) del file, scrittura del file (W), esecuzione del file (X).

- Nei S.O. multiutente è anche possibile proteggere l'accesso alle varie risorse di sistema mediante le Access Control List (ACL), ovvero liste che specificano quali utenti possono utilizzare determinate risorse (Es. ad alcuni utenti può essere proibito l'uso di una stampante di rete)

# Le applicazioni

- Le applicazioni (programmi applicativi) nascondono i dettagli di basso livello all'utente.

- Un programma applicativo è composto da tre componenti fondamentali:

  - sottosistema di interfaccia con l'utente (interfaccia utente): si incarica di cotrollare gli input dell'utente e presentare gli output dei risultati computati. Le interfacce possono essere grafiche o testuali.

  - sottosistema di logica applicativa: implementa gli algoritmi per la manipolazione di dati che caratterizzano l'applicazione

  - sottosistema di gestione dei dati: si occupa dell'organizzazione dei dati, in particolare reperimento e memorizzazione.

- Sono 3 sistemi indipendenti (gestione più semplice e rapida dello sviluppo del SW)

# Sviluppo delle applicazioni

- Le applicazioni possono essere sistemi complessi. Dunque si devono progettare con cura
  - la programmazione di un'applicazione è solo una fase nella catena di sviluppo del SW.
- Lo sviluppo di un SW può essere suddiviso nelle seguenti attività:
  - analisi e specifica dei requisiti
  - progettazione
  - sviluppo e test
  - rilascio
  - manutenzione

# Ciclo di vita del SW

- L'insieme delle attività necessarie allo sviluppo di un'applicazione formano il ciclo di vita del software.
- Il ciclo di vita del software può essere strutturato secondo diversi modelli:
  - modello a cascata: attività svolte in sequenza
  - modello a prototipi: si comincia lo sviluppo direttamente implementando dei prototipi
  - modello evolutivo: si opera in modo incrementale, iniziando a sviluppare le componenti più critiche del sistema e arrichendo via via il sistema di nuove funzionalità.

# Tipi di applicazioni

- Word processors: videoscrittura, elaborazione del testo
- spreadsheets (o fogli elettronici): per costruire tabelle, grafici e fare dei calcoli.
- Browser (per navigare in internet)
- Basi di dati (per memorizzare e reperire efficientemente l'informazione)
- per presentazioni
- per l'elaborazione grafica e video (fotoritocco, disegno, montaggio digitale...)
- per la posta elettronica e altri tipi di servizi di rete

# Word Processors

- si possono distinguere in due categorie
  - WYSIWYG (What You See Is What You Get): es. Word
  - non WYSIWYG: prevedono un linguaggio per la formattazione del testo, es. LaTeX.

# Esempio LaTeX

```
$$E_1=\{e\mid e\in\cI^- \mbox{ and }
       \ll e\rr\not\in\bigcup_{\cI'\in \cA_1}\cI'\}$$
```

$$E_1 = \{e \mid e \in \mathcal{I}^- \text{ and } [e] \notin \bigcup_{\mathcal{I}' \in \mathcal{A}_1} \mathcal{I}'\}$$

# Prestazioni di un sistema

- Si può misurare il tempo complessivo necessario al completamento di un compito (elapsed time), che comprende l'accesso al disco, alla memoria, attività di I/O, tempo di CPU,...
    - è influenzato dagli altri processi attivi, in particolare dal sovraccarico apportato dal sistema operativo
- Tempo effettivo di elaborazione (tempo di CPU), non comprende il tempo di attesa dovuto alle operazioni di I/O e all'esecuzione di altri processi.

# Tempo di CPU

$$t_{CPU} = n_{CK} * T_{CK}$$

$t_{CPU}$: tempo di CPU

$n_{CK}$: numero di cicli di CLOCK necessari al completamento del processo

$T_{CK}$: periodo di CLOCK

oppure

$$t_{CPU} = n_{CK} / f_{CK}$$

$t_{CPU}$: tempo di CPU

$n_{CK}$: numero di cicli di CLOCK necessari al completamento del processo

$f_{CK}$: frequenza di CLOCK

# Tempo di CPU

Consideriamo $n_I$ numero di istruzioni che vengono eseguite

$$CPI = n_{CK} / n_I$$

CPI: numero medio di cicli di CLOCK per istruzione

Quindi,

$$t_{CPU} = CPI * n_I * T_{CK}$$

Oppure,

$$t_{CPU} = (CPI * n_I) / f_{CK}$$

# Tempo di CPU

$$t_{CPU} = (CPI * n_I) / f_{CK}$$

Il tempo di CPU è influenzato da

1. frequenza di CLOCK
2. numero medio di cicli di CLOCK per istruzione
3. numero di istruzioni

# CISC vs RISC

- Due filosofie di progettazione dei processori
  - CISC (complex instruction set computer)
  - RISC (reduced instruction set computer)

| **Tipo di CPU** | **CISC** | **RISC** |
|---|---|---|
| CPI | alto | basso |
| $n_I$ | basso | alto |
| $f_{CK}$ | media | alta |

C-RISC soluzione intermedia!

# Altri indicatori

- Il tempo di CPU è generalmente il parametro più importante nella valutazione delle prestazioni di un sistema informatico
- Ci sono comunque altri indicatori
  - MIPS (Mega Instructions per Second)
  - MFLOPS (Mega Floating Point Operations per Second)