Università di Udine — Facoltà di Scienze MM.FF.NN.

Laurea in Informatica — A.A. 2003/04

Trasparenze del Corso di *Sistemi Operativi*

Marino Miculan

Università di Udine

Introduzione

- Cosa è un sistema operativo?
- Evoluzione dei sistemi operativi
- Tipi di sistemi operativi
- Concetti fondamentali
- Chiamate di sistema
- Struttura dei Sistemi Operativi

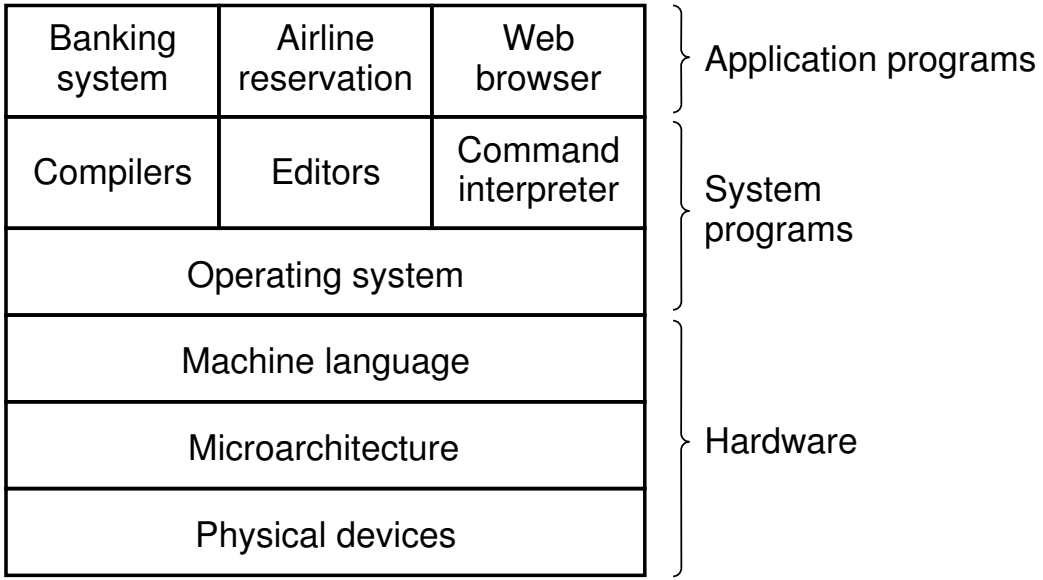
Cosa è un sistema operativo?

Possibili risposte:

- È un *programma di controllo*
- È un *gestore di risorse*
- È un *divoratore di risorse*
- È un *fornitore di servizi*
- È simile ad un *governo*: non fa niente, di per sé...
- ...

Nessuna di queste definizioni è completa

Visione astratta delle componenti di un sistema di calcolo



Componenti di un sistema di calcolo

1. Hardware – fornisce le risorse computazionali di base: (CPU, memoria, dispositivi di I/O).
2. Sistema operativo – controlla e coordina l'uso dell'hardware tra i varî programmi applicativi per i diversi utenti
3. Programmi applicativi — definiscono il modo in cui le risorse del sistema sono usate per risolvere i problemi computazionali dell'utente (compilatori, database, videgiochi, programmi di produttività personale,...)
4. Utenti (persone, macchine, altri calcolatori)

Cosa è un sistema operativo? (2)

Non c' una definizione completa ed esauriente: dipende dai contesti.

- Un programma che agisce come intermediario tra l'utente/programmatore e l'hardware del calcolatore.
- Assegnatore di risorse
  Gestisce ed alloca efficientemente le risorse finite della macchina.
- Programma di controllo
  Controlla l'esecuzione dei programmi e le operazioni sulle risorse del sistema di calcolo.
  Condivisione corretta rispetto al tempo e rispetto allo spazio

Obiettivi di un sistema operativo

Realizzare una *macchina astratta*: implementare funzionalità di alto livello, nascondendo dettagli di basso livello.

- Eseguire programmi utente e rendere più facile la soluzione dei problemi dell'utente
- Rendere il sistema di calcolo più facile da utilizzare e programmare
- Utilizzare l'hardware del calcolatore in modo sicuro ed efficiente

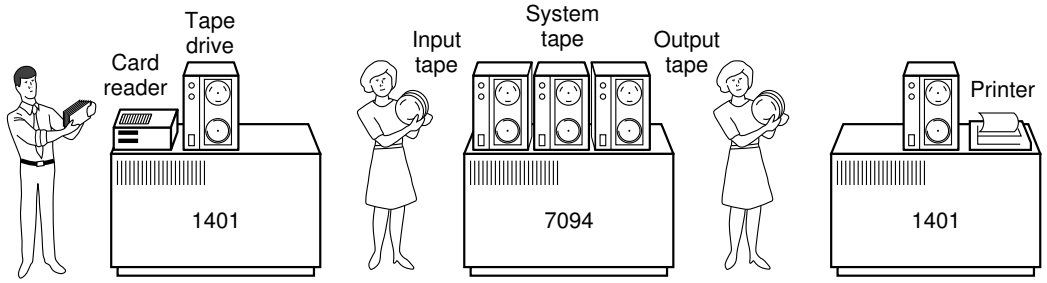
Questi obiettivi sono in contrapposizione. A quale obiettivo dare priorità dipende dal contesto.

Primi sistemi – Macchine nude e crude (primi anni '50)

- Struttura
  - Grossi calcolatori funzionanti solo da console
  - Sistemi single user; il programmatore era anche utente e operatore
  - I/O su nastro perforato o schede perforate
- Primi Software
  - Assemblatori, compilatori, linker, loader
  - Librerie di subroutine comuni
  - Driver di dispositivi
- Molto sicuri
- Uso inefficiente di risorse assai costose
  - Bassa utilizzazione della CPU
  - Molto tempo impiegato nel setup dei programmi

Semplici Sistemi Batch

- Assumere un operatore
- Utente $\neq$ operatore
- Aggiungere un lettore di schede



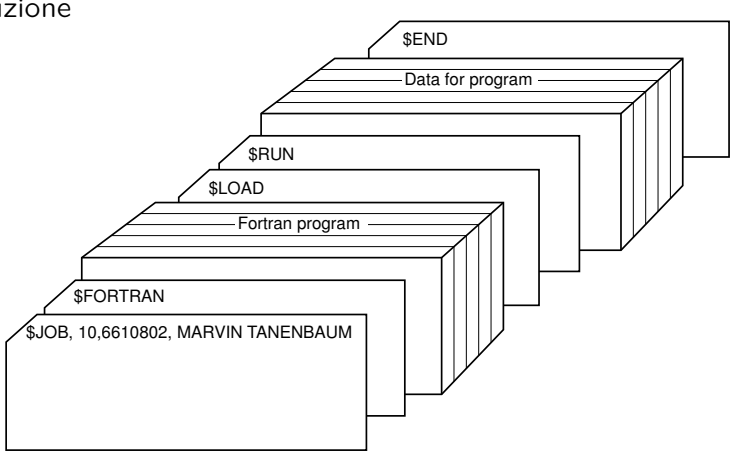
- Ridurre il tempo di setup raggruppando i job simili (*batch*)
- Sequenzializzazione automatica dei job – automaticamente, il controllo passa da un job al successivo. Primo rudimentale sistema operativo
- Monitor residente
  - inizialmente, il controllo è in monitor
  - poi viene trasferito al job
  - quando il job è completato, il controllo torna al monitor

Semplici Sistemi Batch (Cont.)

- Problemi
  1. Come fa il monitor a sapere la natura del job (e.g., Fortran o assembler?) o quale programma eseguire sui dati forniti?
  2. Come fa il monitor a distinguere
     (a) un job da un altro
     (b) dati dal programma
- Soluzione: schede di controllo

Schede di controllo

- Schede speciali che indicano al monitor residente quali programmi mandare in esecuzione

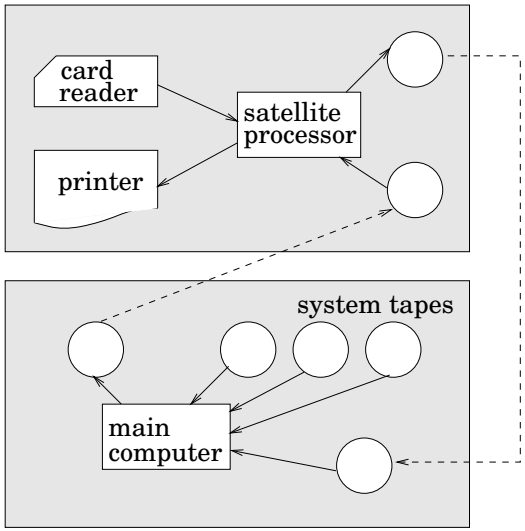


- Caratteri speciali distinguono le schede di controllo dalle schede di programma o di dati.

Schede di controllo (Cont.)

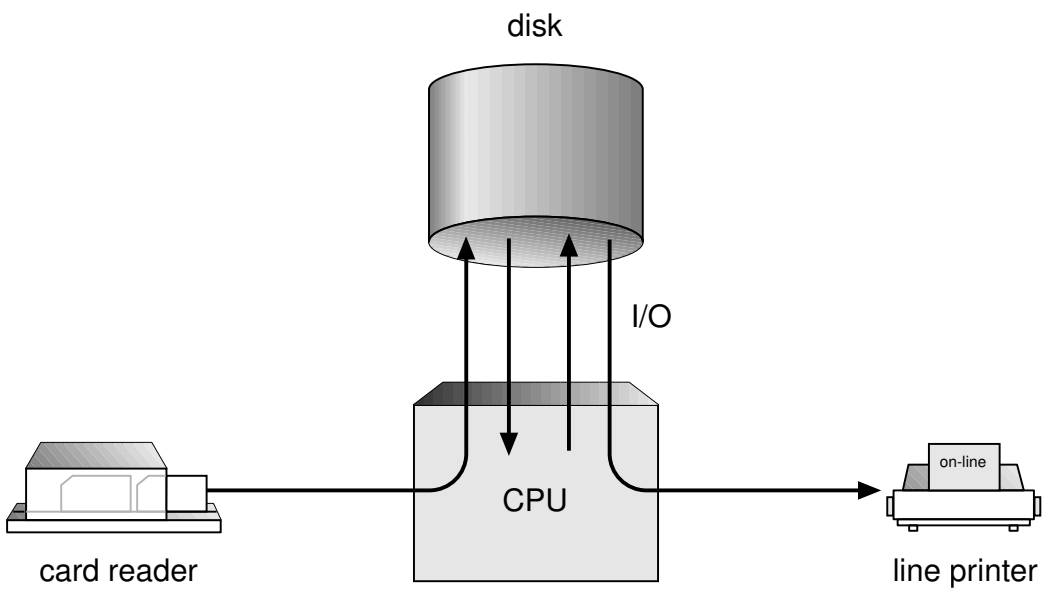
- Una parte del monitor residente è
  - Inteprete delle schede di controllo – responsabile della lettura e esecuzione delle istruzioni sulle schede di controllo
  - Loader – carica i programmi di sistema e applicativi in memoria
  - Driver dei dispositivi – conoscono le caratteristiche e le proprietà di ogni dispositivo di I/O.
- Problema: bassa performance – I/O e CPU non possono sovrapporsi; i lettore di schede sono molto lenti.
- Soluzione: operazioni off-line – velocizzare la computazione caricando i job in memoria da nastri, mentre la lettura e la stampa vengono eseguiti off-line

Funzionamento Off-Line



- Il computer principale non è limitato dalla velocità dei lettori di schede o stampanti, ma solo dalla velocità delle unità nastro.
- Non si devono fare modifiche nei programmi applicativi per passare dal funzionamento diretto a quello off-line
- Guadagno in efficienza: si può usare più lettori e più stampanti per una CPU.

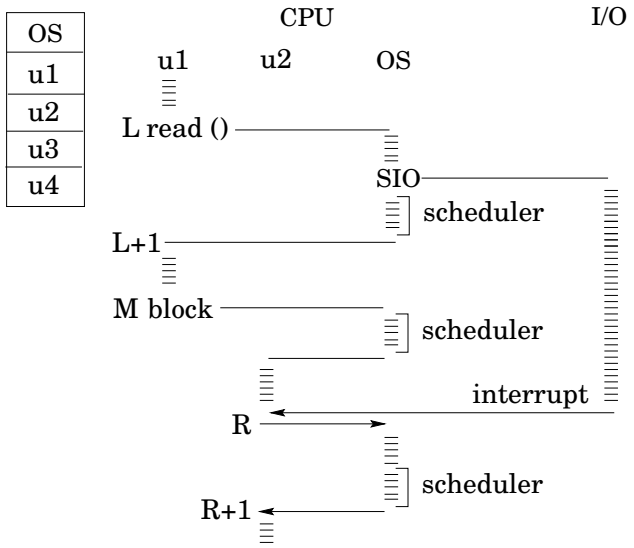
Spooling



- Spool = Simultaneous peripheral operation on-line
- Sovrapposizione dell'I/O di un job con la computazione di un altro job. Mentre un job è in esecuzione, il sistema operativo
  - legge il prossimo job dal lettore di schede in un'area su disco (coda dei job)
  - trasferisce l'output del job precedente dal disco alla stampante
- *Job pool* – struttura dati che permette al S.O. di scegliere quale job mandare in esecuzione al fine di aumentare l'utilizzazione della CPU.

Anni 60: Sistemi batch Multiprogrammati

Più job sono tenuti in memoria nello stesso momento, e la CPU fa a turno su tutti i job



Caratteristiche dell'OS richieste per la multiprogrammazione

- routine di I/O devono essere fornite dal sistema
- Gestione della Memoria – il sistema deve allocare memoria per più job
- Scheduling della CPU – il sistema deve scegliere tra più job pronti per l'esecuzione
- Allocazione dei dispositivi

Anni 70: Sistemi Time-Sharing – Computazione Interattiva

- La CPU è condivisa tra più job che sono tenuti in memoria e su disco (la CPU è allocata ad un job solo se questo si trova in memoria)
- Un job viene caricato dal disco alla memoria, e viceversa (*swapping*)
- Viene fornita una comunicazione on-line tra l'utente e il sistema; quando il sistema operativo termina l'esecuzione di un comando, attende il prossimo "statement di controllo" non dal lettore di schede bensí dalla tastiera dell'utente.
- Deve essere disponibile un file system on-line per poter accedere ai dati e al codice

Anni 80: Personal Computer

- *Personal computers* – sistemi di calcolo dedicati ad un singolo utente
- I/O devices – tastiere, mouse, schermi, piccole stampanti
- Comodità per l'utente e reattività
- Interfaccia utente evoluta (GUI)
- Possono adottare tecnologie sviluppate per sistemi operativi più grandi; spesso gli individui hanno un uso esclusivo del calcolatore, e non necessitano di avanzate tecniche di sfruttamento della CPU o sistemi di protezione.

Anni 90: Sistemi operativi di rete

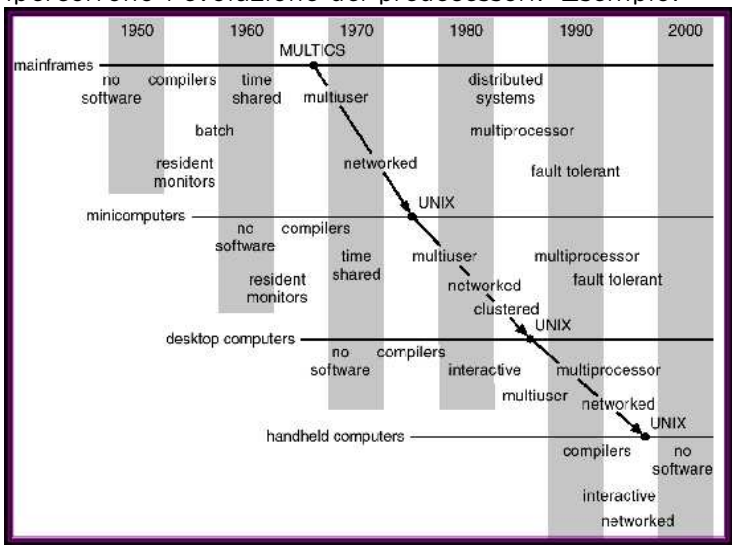
- Distribuzione della computazione tra più processori
- Sistemi *debolmente accoppiati* – ogni processore ha la sua propria memoria; i processori comunicano tra loro attraverso linee di comunicazione (e.g., bus ad alta velocità, linee telefoniche, fibre ottiche,. . .)
- In un sistema operativi di rete, l'utente ha coscienza della differenza tra i singoli nodi.
  - Trasferimenti di dati e computazioni avvengono in modo esplicito
  - Poco tollerante ai guasti
  - Complesso per gli utenti

Il futuro: Sistemi operativi distribuiti

- In un sistema operativo distribuito, l'utente ha una visione *unitaria* del sistema di calcolo.
  - Condivisione delle risorse (dati e computazionali)
  - Aumento della velocità – bilanciamento del carico
  - Tolleranza ai guasti
- Un sistema operativo distribuito è molto più complesso di un SO di rete.
- Esempi di servizi (non sistemi) di rete: NFS, P2P (KaZaA, Gnutella, . . .), Grid computing. . .

L'ontogenesi riassume la filogenesi

L'hardware e il software (tra cui i sistemi operativi) in ogni nuova classe di calcolatori ripercorrono l'evoluzione dei predecessori. Esempio:



Lo zoo

Diversi obiettivi e requisiti a seconda delle situazioni

- Supercalcolatori
- Mainframe
- Server
- Multiprocessore
- Personal Computer
- Real Time
- Embedded

Sistemi operativi per mainframe

- Enormi quantità di dati ($> 1TB$)
- Grande I/O
- Elaborazione "batch" non interattiva
- Assoluta stabilità (uptime $> 99,999\%$)
- Applicazioni: banche, amministrazioni, ricerca...
- Esempi: IBM OS/360, OS/390

Sistemi operativi per supercalcolatori

- Grandi quantità di dati ($> 1TB$)
- Enormi potenze di calcolo (es. NEC Earth-Simulator, 40 TFLOP)
- Architetture NUMA o NORMA (migliaia di CPU)
- Job di calcolo intensivo
- Elaborazione "batch" non interattiva
- Esempi: Unix, o ad hoc

Sistemi per server

- Sistemi multiprocessore con spesso più di una CPU in comunicazione stretta.
- Degrado graduale delle prestazioni in caso di guasto (*fail-soft*)
- Riconfigurazione hardware a caldo
- Rilevamento automatico dei guasti
- Elaborazione su richiesta (semi-interattiva)
- Applicazioni: server web, di posta, dati, etc.
- Esempi: Unix, Linux, Windows NT e derivati

Sistemi Real-Time

- Vincoli temporali fissati e ben definiti
- Sistemi *hard real-time*: i vincoli devono essere soddisfatti
  - la memoria secondaria è limitata o assente; i dati sono memorizzati o in memoria volatile, o in ROM.
  - In confitto con i sistemi time-sharing; non sono supportati dai sistemi operativi d'uso generale
  - Usati in robotica, controlli industriali, software di bordo...
- Sistemi *soft real-time*: i vincoli possono anche non essere soddisfatti, ma il sistema operativo deve fare del suo meglio
  - Uso limitato nei controlli industriali o nella robotica
  - Utili in applicazioni (multimedia, virtual reality) che richiedono caratteristiche avanzate dei sistemi operativi

Sistemi operativi embedded

- Per calcolatori palmari (PDA), cellulari, ma anche televisori, forni a microonde, lavatrici, etc.
- Hanno spesso caratteristiche di real-time
- Limitate risorse hardware
- esempio: PalmOS, Epoc, PocketPC, QNX.

Sistemi operativi per smart card

- Girano sulla CPU delle smartcard
- Stretti vincoli sull'uso di memoria e alimentazione
- implementano funzioni minime
- Esempio: JavaCard