Sistemi Operativi

AAF - Secondo anno - 3CFU

A.A. 2021/2022

Corso di Laurea in Matematica

Sistemi operativi - Nozioni generali

Annalisa Massini

Dipartimento di Informatica
Sapienza Università di Roma

Argomenti trattati

1. Introduzione ai SO

2. Sistemi operativi
   - Funzioni e obiettivi
   - Storia ed evoluzione
   - Risultati principali
   - UNIX

Sistema Operativo

- Sfrutta le risorse hardware di un sistema computerizzato
  - uno o più processori
  - memoria primaria (RAM)
  - memoria secondaria (dischi)
  - dispositivi di input/output

- Fornisce un insieme di servizi agli utenti
  - in particolare: offre un ambiente di esecuzione **facilitato** alle applicazioni utente

Sistemi Operativi: Esempi

- Windows
  - ultima versione: Windows 10
- macOS
  - ultima versione: macOS Mojave
- Linux Ubuntu
  - ultima versione: Ubuntu 19.04 (Disco Dingo)
- Windows Phone, iOS, Android

Sistemi operativi

Funzioni e obiettivi

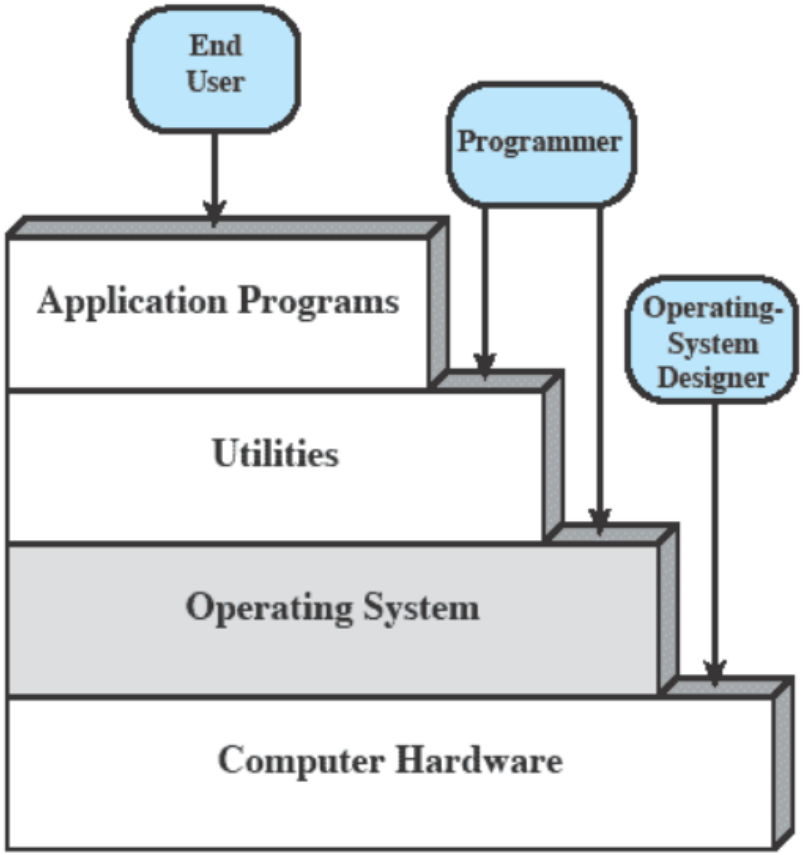
Sistema Operativo

- **Obiettivi** di un sistema operativo
  - **Convenienza** - rende il computer più facile da usare
  - **Efficienza** - permette di usare le risorse del computer in modo più efficiente
  - **Capacità di evolvere** - deve essere progettato in modo da
    - gestire aggiornamenti hardware o nuovo hardware
    - introdurre nuove funzioni e offrire nuovi servizi
    - permettere di correggere (inevitabili) errori individuati nel corso del tempo

Sistema Operativo

- Il SO rappresenta un'**interfaccia tra utente e computer**
  - Hw e sw forniscono applicazioni all'utente e possono essere visualizzati secondo uno schemacon diversi i livelli
  - L'utente (utilizzatore di applicazioni) non è interessato ai dettagli hw della macchina
  - L'utente vede il sistema come fornitore di un insieme di applicazioni, dove per applicazione intendiamo un programma sviluppato in un qualche linguaggio di programmazione

Strati e Utenti



- Il SO maschera i dettagli hw al programmatore e fornisce un'interfaccia che facilita l'uso del sistema

Servizi Offerti da un SO

- Sviluppo di programmi
  - compilatori, editor e debugger
  - system calls
  - visione semplificata della memoria RAM
  - alcuni servizi non fanno parte del *core* del SO, ma sono forniti con il SO
- Esecuzioni di programmi
  - app(licazioni)
  - servizi
  - anche più applicazioni e servizi contemporaneamente
  - il SO gestisce la temporizzazione delle varie operazioni

Servizi Offerti da un SO

- Accesso ai dispositivi di input/output
  - il SO nasconde al programmatore i dettagli dei singoli dispositivi di I/O
  - nel caso dei dispositivi di memoria di massa, tramite filesystem
- Accesso controllato ai files
  - il SO gestisce i diversi formati dei (oltre ai diversi dispositivi)
  - in caso di utenti multipli, serve un meccanismo di protezione per l'accesso ai file
- Accesso al sistema
  - il SO controlla l'accesso a tutto il sistema o alle singole risorse
  - serve protezione delle risorse e dei datie risoluzione dei conflitti

Servizi Offerti da un SO

- Rilevamento e risposta agli errori
  - errori di hardware interno ed esterno
  - errori software
  - richiesta di un applicativo non soddisfacibile
  - il SO sceglie la soluzione più appropriata (terminare programma, ripetere operazione, segnalare il tipo di errore)
- Accounting (chi fa cosa)
  - collezione di statistiche dell'uso del sistema
  - monitoraggio delle performance
  - usato per capire cosa occorre migliorare
  - usato per far pagare in base all'uso del sistema

Sistema Operativo

- Quindi un sistema operativo:
  - è un **programma** che controlla l'esecuzione dei programmi applicativi
  - fornisce un'**interfaccia** tra le applicazioni e l'hardware
  - fornisce un'**interfaccia** tra utente e computer
    - l'utente finale non vuole (e non deve) preoccuparsi dei dettagli hardware e vede il computer come un insieme di applicazioni

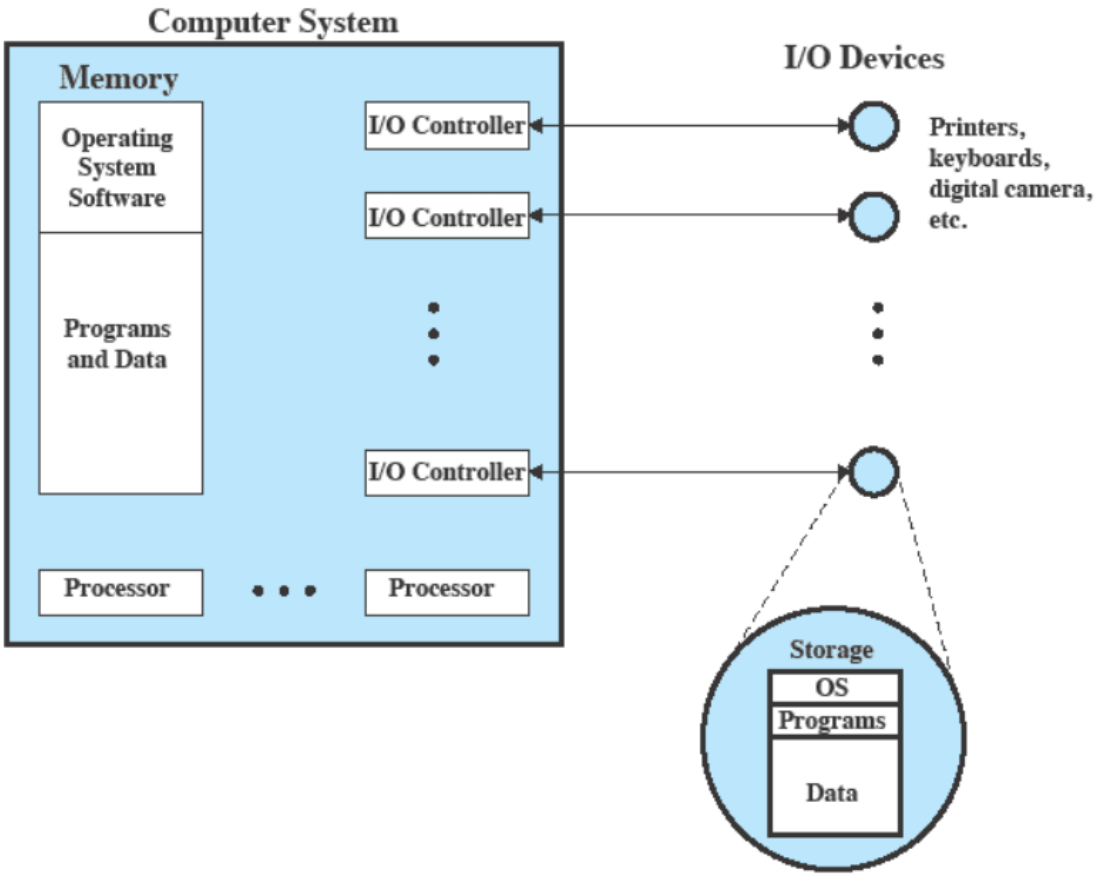
Sistema Operativo

- Un computer è un insieme di risorse per elaborazione, memorizzazione e trasferimento di informazioni

- Il SO è **responsabile della gestione delle risorse**
  - funziona allo stesso modo del software normale: è un programma in esecuzione
    - tuttavia, lo fa con privilegi più alti
  - concede il controllo del processore ad altri programmi
    - e controlla l'accesso alle altre risorse (RAM, I/O)

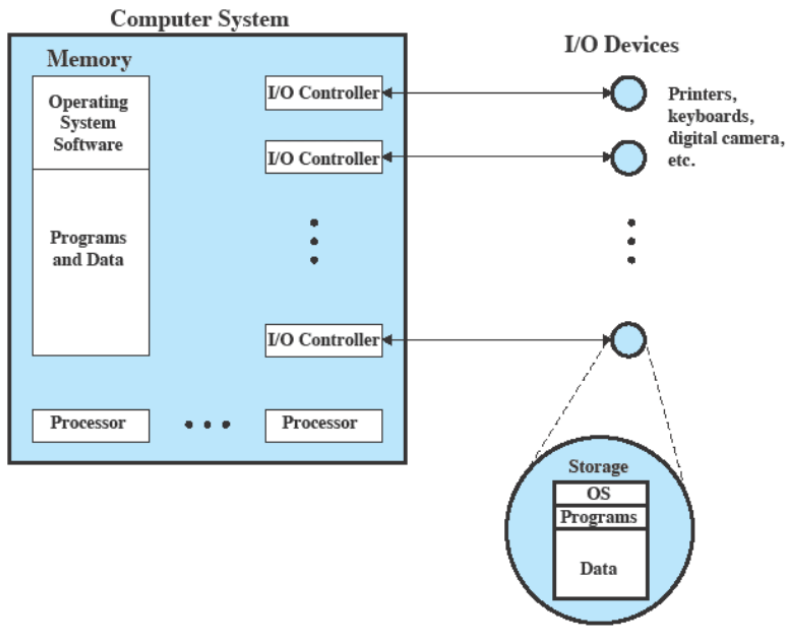
Sistema Operativo

- SO come **responsabile della gestione delle risorse**



Il Kernel

- Il **Kernel** o **nucleo** è:
  - la parte di sistema operativo che si trova sempre in memoria principale
  - contiene le funzioni più usate



Sistemi operativi

Storia ed evoluzione

Storia dei SO

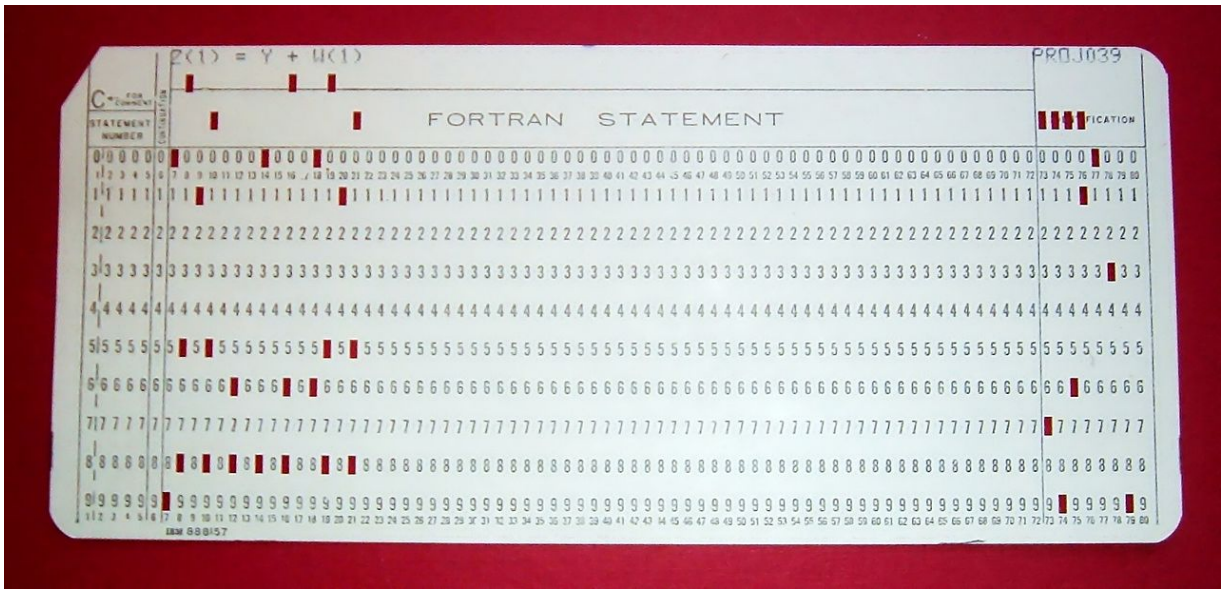
- I sistemi operativi **primitivi** (**anni Quaranta**) erano molto diversi da quelli attuali
- La loro evoluzione è dovuta principalmente a:
  - aggiornamento dell'hardware o nuovo tipo hardware
  - nuovi servizi
  - correzione di errori
- `http://www.computerhistory.org/timeline/`

Storia dei SO

- Computazione seriale (**anni Quaranta**)
  - nessun sistema operativo
  - per fornire comandi ad un computer, si usavano speciali **console**, con spie luminose ed interruttori e una stampante per l'output

Storia dei SO

- Computazione seriale (**anni Quaranta**)
  - già dall'inizio l'inserimento dell'input viene parzialmente semplificato con dispositivi per leggere schede perforate (esistenti già da 2 secoli)



Storia dei SO

- Semplice sistema non interattivo o **batch OS** (**anni Cinquanta/Sessanta**)
  - **monitor**: programma esterno di monitoraggio
  - software per controllare sequenze di eventi
  - possibilità di raggruppare lavori (**jobs**) da eseguire insieme
  - il programma, una volta concluso, ritorna il controllo al programma esterno di monitoraggio
- Linguaggio di controllo dei job - **JCL**
  - per dare istruzioni al monitor
  - per specificare che compilatore usare
  - per specificare quali dati di input usare

Monitor e Caratteristiche Hardware

- Protezione della memoria
  - non permette che la zona di memoria contenente il monitor venga modificata
- Timer
  - impedisce che un job monopolizzi l'intero sistema
- Istruzioni privilegiate
  - alcune istruzioni macchina possono essere eseguite solo dal monitor
- Interruzioni
  - i primi modelli di computer non le avevano

Modi operativi

Il meccanismo di protezione della memoria e le operazioni privilegiate portano al concetto di **modo operativo**

- I programmi utente vengono eseguiti in **modalità utente**
  - alcune istruzioni non possono essere eseguite
- Il monitor viene eseguito in **modalità sistema**
  - detta anche modalità **kernel**
  - le istruzioni privilegiate possono essere eseguite
  - le aree protette della memoria possono essere accedute

Sistemi Batch: Sottoutilizzazione

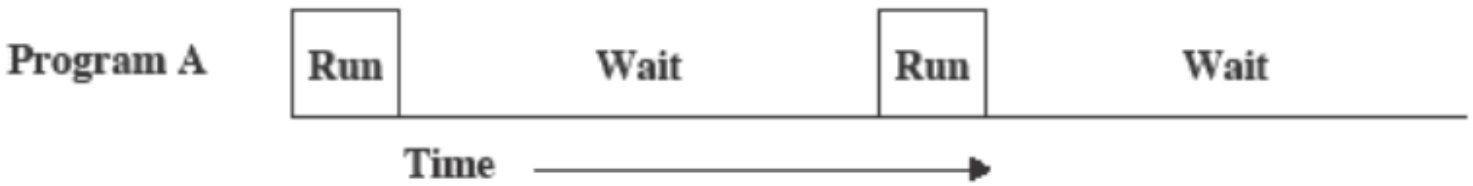
Problema dei sistemi batch: il 96% del tempo è sprecato ad aspettare i dispositivi di I/O

Read one record from file	15 μs
Execute 100 instructions	1 μs
Write one record to file	15 μs
TOTAL	31 μs

$$\text{Percent CPU Utilization} = \frac{1}{31} = 0.032 = 3.2\%$$

Programmazione Singola

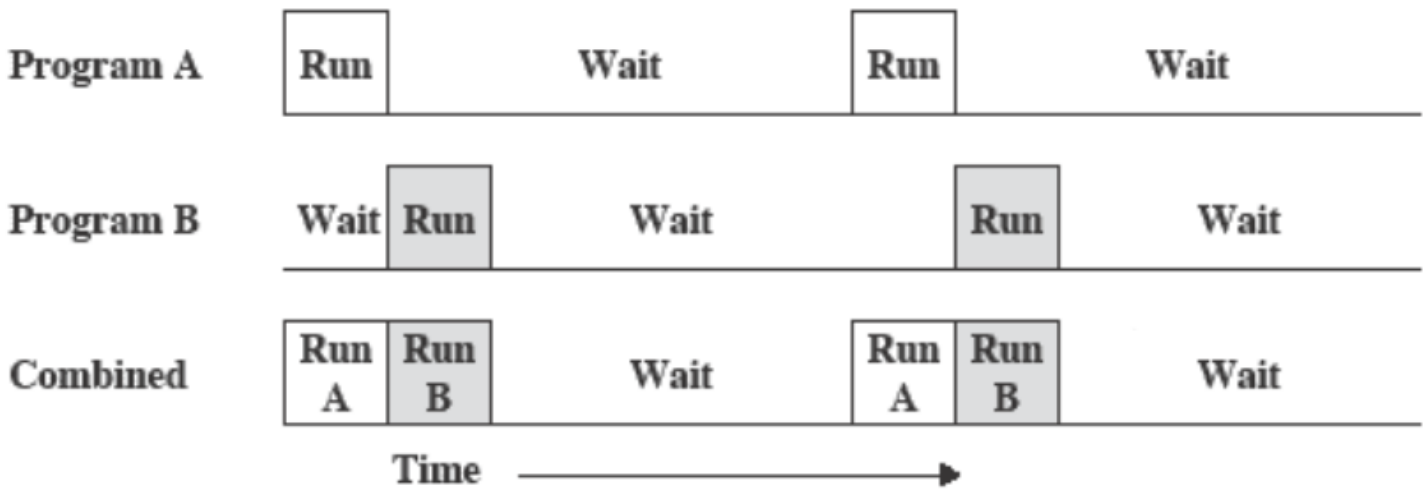
Il processore deve aspettare che le istruzioni di I/O siano completate prima di procedere



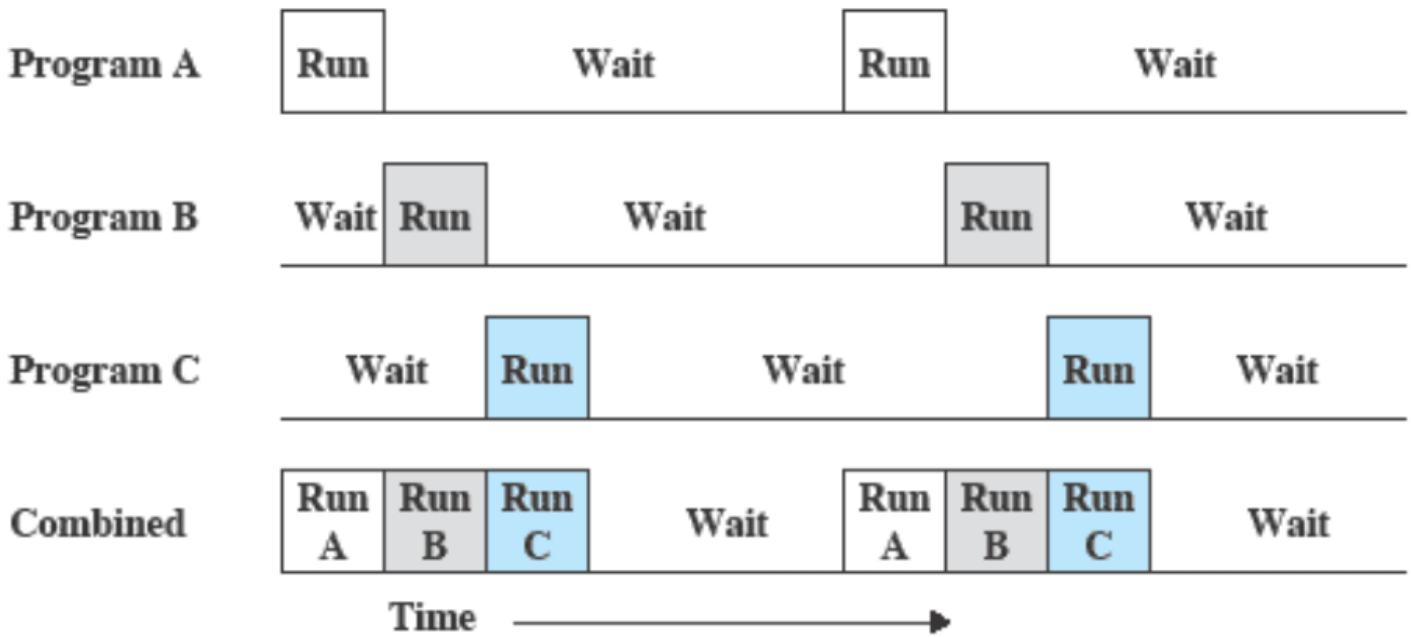
Multiprogrammazione

Con l'approccio **multiprogrammazione** o **multitasking**

- se un job deve aspettare che si completi un'operazione di I/O, il processore può passare ad un altro job
- il processore gestisce più job batch allo stesso tempo



Multiprogrammazione



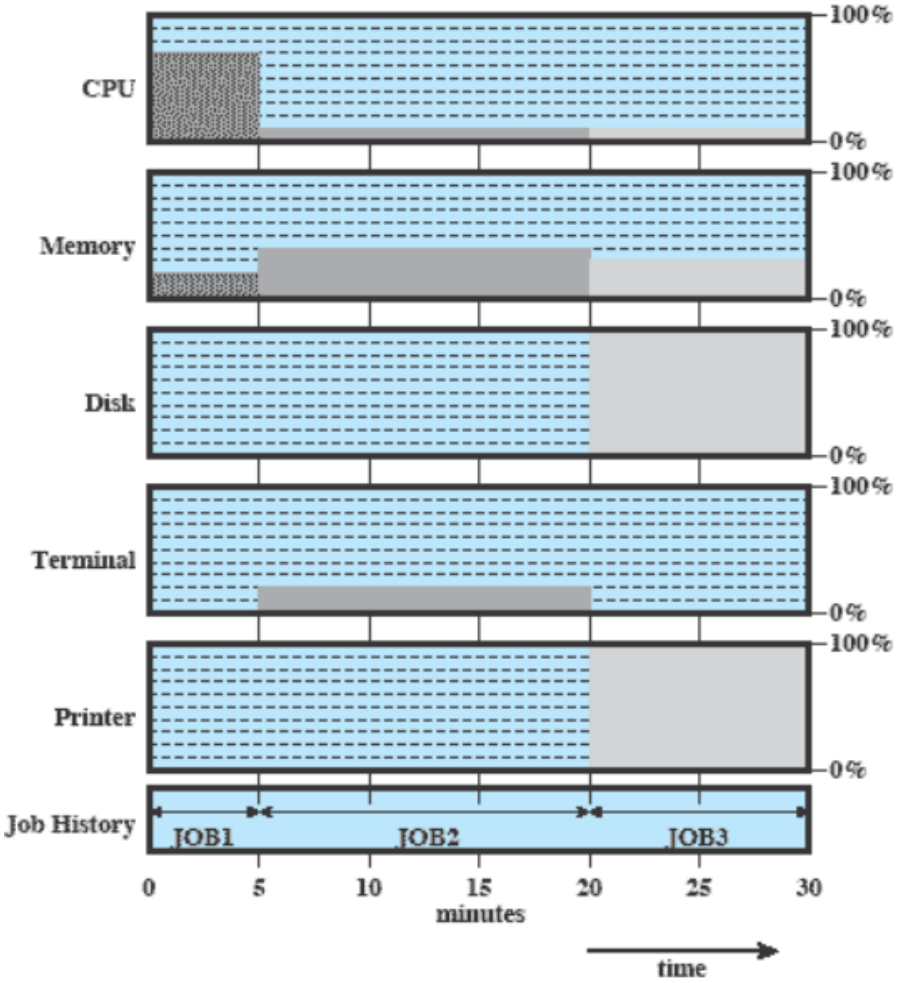
(c) Multiprogramming with three programs

Esempio

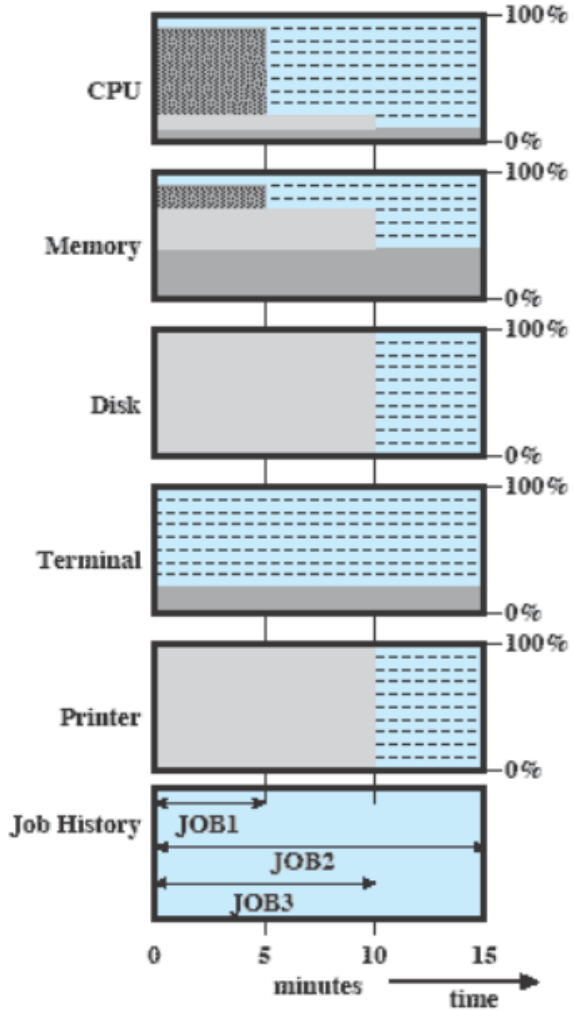
- Consideriamo un computer con: 250 Mbyte di memoria disponibile, disco, terminale, stampante
- Consideriamo tre programmi con le caratteristiche in tabella, che in ambiente batch vengono eseguiti in sequenza

	JOB1	JOB2	JOB3
Type of job	Heavy compute	Heavy I/O	Heavy I/O
Duration	5 min	15 min	10 min
Memory required	50 M	100 M	75 M
Need disk?	No	No	Yes
Need terminal?	No	Yes	No
Need printer?	No	No	Yes

Istogrammi di Utilizzo



(a) Uniprogramming



(b) Multiprogramming

Uso del Processore

- Prime 4 righe: dagli istogrammi la media per uniprogramming e per multiprogramming
- Elapsed time: tempo per vedere completati tutti e 3 i job
- Throughput: $\frac{\text{numero job completati}}{\text{ore}}$
- Mean response time: media dei tempi di completamento (nel caso di uniprogramming: 5, 20, 30 $\rightarrow$ 18.3)

	Uniprogramming	Multiprogramming
Processor use	20%	40%
Memory use	33%	67%
Disk use	33%	67%
Printer use	33%	67%
Elapsed time	30 min	15 min
Throughput	6 jobs/hr	12 jobs/hr
Mean response time	18 min	10 min

Sistemi Time-Sharing

- Letteralmente, sistemi a condivisione di tempo (**dagli anni Settanta**)
- La multiprogrammazione è usata per gestire contemporaneamente più jobs **interattivi**
- Il tempo del processore è condiviso tra più utenti
- Più utenti contemporaneamente accedono al sistema tramite terminali

Batch vs. Time Sharing

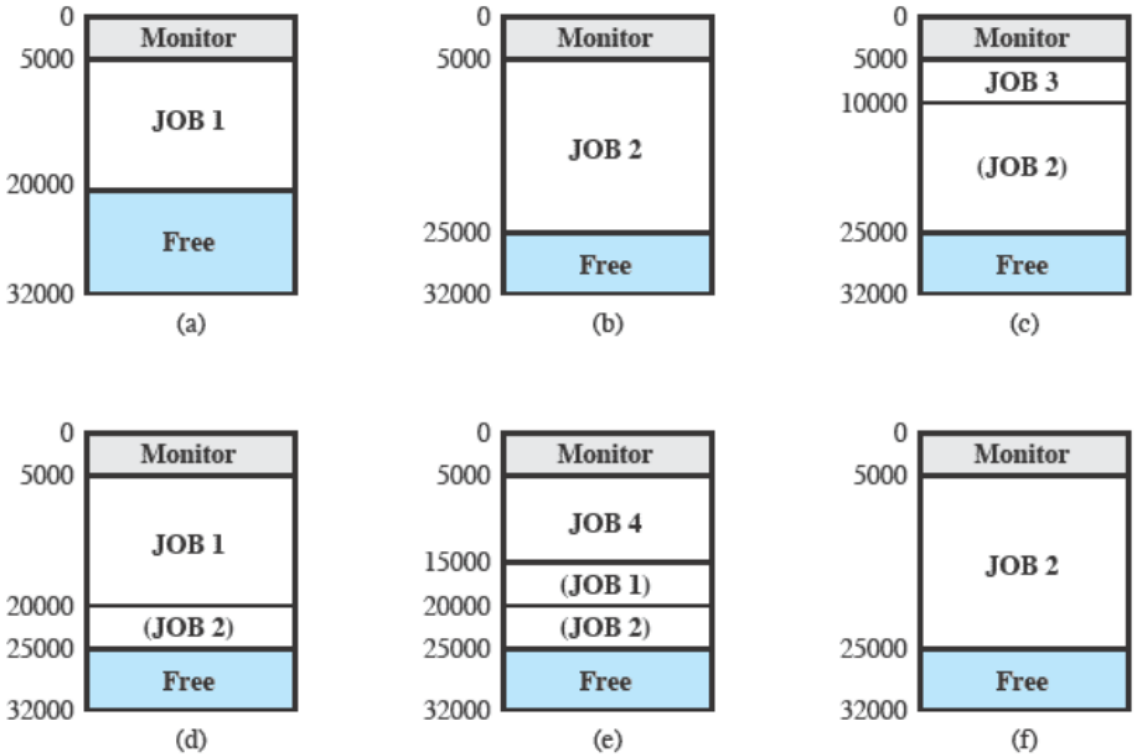
	Batch	**Time Sharing**
Scopo principale	Massimizzare l'uso del processore	Minimizzare il tempo di risposta
Provenienza delle direttive al SO	Comandi del job control language, sottomessi con il job stesso	Comandi dati da terminale

CTSS: Compatible Time-Sharing System

- **CTSS** è uno dei primi SO che supporta il time-sharing (sviluppato negli anni '60)
- Un clock di sistema genera un interrupt ogni 0,2 s circa
- Ad ogni clock interrupt il SO assegna il processore ad un altro utente
- Il monitor occupa 5000 locazioni, mentre nelle rimanenti 27000 viene caricato il programma dell'utente a cui viene assegnato il processore
- Quando si cambia utente si salva su disco il programma sospeso e si trasferisce da disco quello nuovo

CTSS: Compatible Time-Sharing System

Esempio con 4 job interattivi: Job1 15k, Job2 20k, Job3 5k, Job4 10k



Storia dei SO

Riassumendo

- **anni Quaranta** - Computazione seriale
- **anni Cinquanta/Sessanta** - Semplice sistema non interattivo o *batch*
  - multiprogrammazione
- **dagli anni Settanta** - Time-Sharing: sistemi a condivisione di tempo
  - i job sono tipicamente interattivi

Sistemi operativi

Principali risultati raggiunti nel corso del tempo

Risultati più Importanti

- Processi
- Gestione della memoria
- Sicurezza e protezione delle informazioni (privacy)
- Gestione dello scheduling e delle risorse
- Strutturazione del sistema

I processi

- Il concetto centrale in un sistema operativo è il concetto di **processo**
- Sono state date molte definizioni di processo:
  - un programma in esecuzione
  - un'istanza di un programma che gira su un computer
  - l'entità assegnata a ed eseguita da un processore
  - un'unità di attività caratterizzata da un singolo thread sequenziale di esecuzione, lo stato corrente e un insieme di risorse associato.

I processi

- Allo sviluppo del concetto di processo hanno contribuito i problemi di sincronizzazione generati principalmente da:
  - multiprogrammazione
  - condivisione di tempo - time sharing
  - transazioni real-time

Multiprogrammazione

- Introdotta per mantenere processore e dispositivi di I/O (inclusa la memoria esterna) contemporaneamente occupati per massimizzare l'efficienza
- Il processore cambia programma in esecuzione seguendo i segnali di completamento di operazioni I/O

I processi

Time sharing

- L'obiettivo è rispondere tempestivamente al singolo utente e supportare simultaneamente più utenti
- Realizzabile grazie al lento tempo di reazione dell'utente

Sistema transazionale

- Utenti che immettono richieste o eseguono aggiornamenti in un database
- Esempio: sistema di prenotazione aerei

I processi

- Il principale strumento per gestire multiprogrammazione e interazione multiutente è stato l'interruzione
- Il numero di jobs in esecuzione e la soluzione *ad hoc* dei vari casi rende la gestione soggetta ad errori di programmazione
- Le quattro principali cause di errore sono:
  - **Errori di sincronizzazione**
    - gli interrupt si perdono o vengono ricevuti 2 volte
  - **Violazione della mutua esclusione**
    - se 2 processi vogliono accedere alla stessa risorsa, ci possono essere problemi
  - **Programmi con esecuzione non deterministica**
    - un processo accede ad una porzione di memoria modificata da un altro processo
  - **Deadlock (stallo)**
    - un processo A attende un processo B che attende A

I processi

- Per risolvere questi problemi serve un metodo sistematico per monitorare e controllare i vari programmi in esecuzione nel processore
- Il concetto di **processo** fornisce la soluzione
- Possiamo pensare al processo come composto dalle seguenti componenti:
  - un programma eseguibile
  - i dati di cui il programma ha bisogno (di input, output e temporanei)
  - il contesto di esecuzione del programma
  - più tutte le informazioni di cui il sistema operativo ha bisogno per gestire il processo (tabella dei processi)

I processi

- Utilizziamo anche il concetto di **thread**:
  - un singolo processo, cui sono assegnate determinate risorse, può essere suddiviso in un insieme di più thread concorrenti che eseguono cooperativamente il lavoro del processo

- Il concetto di **thread** introduce un nuovo livello di parallelismo che deve essere gestito sia dall'hardware che dal software

Gestione della Memoria

- Gli utenti necessitano di un uso flessibile dei dati
- Per garantire una gestione efficiente e ordinata dei dati, il sistema operativo ha cinque principali responsabilità di gestione dello *storage*:
  - Isolamento dei processi
  - Protezione e controllo degli accessi
  - Gestione (compresa allocazione/deallocazione) automatica
  - Supporto per la programmazione modulare (stack)
  - Memorizzazione a lungo termine

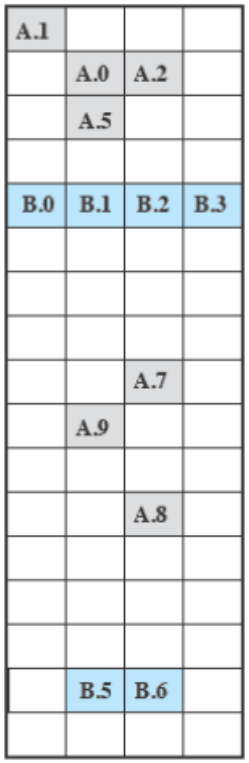
Gestione della Memoria: Memoria virtuale

- Metodi attuali: **paginazione** + **memoria virtuale**
- Nella memoria secondaria (o a lungo termine), l'informazione è memorizzata in oggetti chiamati *files*
- La memoria secondaria è usata anche per implementare la memoria virtuale
- Permette ai programmatori di usare la memoria in modo logico (trascurando i limiti fisici)

Gestione della Memoria: Paginazione della memoria

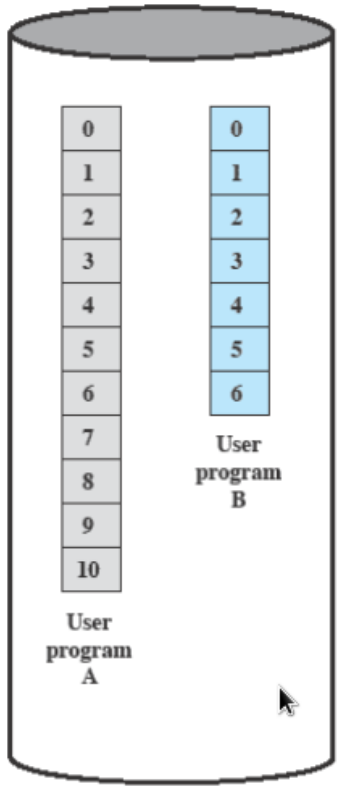
- Permette ai processi di essere contenuti in un certo numeri di blocchi di dimensione fissa, detti pagine
- L'indirizzo virtuale è un numero di pagina più uno spiazzamento (*offset*) nella pagina
- Ogni pagina può trovarsi in qualsiasi punto della memoria
- L'indirizzo vero (o fisico) è l'indirizzo in memoria principale

Gestione della Memoria: memoria virtuale



Main Memory

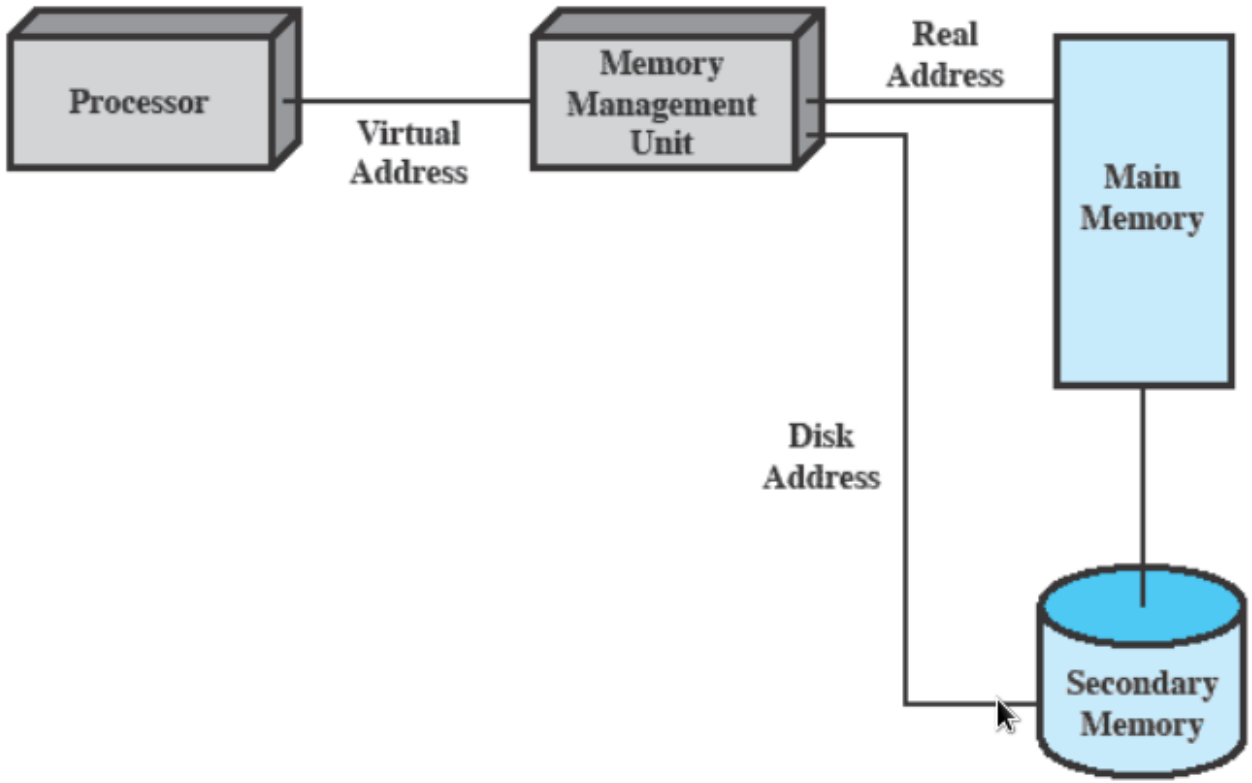
Main memory consists of a number of fixed-length frames, each equal to the size of a page. For a program to execute, some or all of its pages must be in main memory.



Disk

Secondary memory (disk) can hold many fixed-length pages. A user program consists of some number of pages. Pages for all programs plus the operating system are on disk, as are files.

Gestione della Memoria: Indirizzamento della Memoria Virtuale



Protezione dell'Informazione e Sicurezza

- Sistemi time-sharing e uso di connessione in rete hanno generato **problemi di protezione dell'informazione**
- Svariati tipi di attacco
- Hardware e SO supportano meccanismi di protezione e sicurezza
- Le azioni dei SO possono essere raggruppate in 4 categorie:
  - **Disponibilità** (availability)
    - proteggere il sistema contro interruzioni di servizio
  - **Confidenzialità**
    - garanzia che gli utenti non leggano informazioni per le quali non hanno l'autorizzazione
  - **Integrità dei dati**
    - protezione dei dati da modifiche non autorizzate
  - **Autenticità**
    - verifica identità degli utentie la validità di messaggi/dati

Pianificazione e Gestione delle Risorse

- **Compiti chiave del SO** sono la gestione delle risorse disponibili e la pianificazione del loro utilizzo da parte dei processi attivi
- Tre fattori importanti
  - **Equità** (fairness)
    - dare accesso alle risorse in modo egualitario ed equo
  - **Velocità di risposta differenziata**
    - a seconda del tipo di processo
  - **Efficienza**
    - massimizzare l'uso delle risorse per unità di tempo (throughput), minimizzare il tempo di risposta, e servire il maggior numero di utenti possibile

Struttura del Sistema Operativo

- Il sistema (HW + SO) viene visto come una **serie di livelli**
  - Ogni livello effettua un sottoinsieme delle funzioni del sistema
  - Ogni livello si basa sul livello immediatamente più in basso, che effettua alcune operazioni di livello più basso
  - Decomposizione del problema in vari sottoproblemi più semplici

Livelli

- Livello 1
  - circuiti elettrici
  - gli oggetti sono registri, celle di memoria, porte logiche
  - le operazioni sono, ad esempio, resettare un registro o leggere una locazione di memoria
- Livello 2
  - insieme delle istruzioni macchina
  - operazioni come add, subtract, load, and store
- Livello 3
  - aggiunge il concetto di procedura (o subroutine), con operazioni di chiamata e ritorno
- Livello 4
  - interruzioni

Livelli: Multiprogrammazione

- Livello 5
  - processo come programma in esecuzione
  - sospensione e ripresa dell'esecuzione di un processo
- Livello 6
  - dispositivi di memorizzazione secondaria
  - trasferimento di blocchi di dati
- Livello 7
  - crea uno spazio logico degli indirizzi per i processi
  - organizza lo spazio degli indirizzi virtuali in blocchi

Livelli: Dispositivi Esterni

- Livello 8
  - comunicazioni tra processi
- Livello 9
  - salvataggio di lungo termine di file con nome
- Livello 10
  - accesso a dispositivi esterni usando interfacce standardizzate
- Livello 11
  - associazione tra identificatori interni ed esterni
- Livello 12
  - supporto di alto livello per i processi
- Livello 13
  - interfaccia utente

I moderni Sistemi Operativi

- Nuovi elementi di progettazione:
  - nuovi sistemi operativi
  - nuove versioni di SO esistenti
- I moderni sistemi operativi sanno gestire:
  - **nuovo hardware**: multiprocessori, reti ad alta velocità, varietà dei dispositivi di memoria, ecc.
  - **nuove applicazioni**: multimedia, accesso ad internet, ecc.
  - **nuovi attacchi alla sicurezza**: virus, tecniche di hacking, ecc.

I moderni Sistemi Operativi

- La richiesta di cambiamenti a carico dei SO richiede nuove forme di organizzazione
- Nelle proposte di nuovi approcci e progetti la maggior parte del lavoro cade nelle seguenti categorie:
  - **architettura a microkernel**
  - **sistemi operativi distribuiti**
  - **progettazione orientata agli oggetti**
  - **multithreading**
  - **symmetric multiprocessing**

I moderni Sistemi Operativi

- Architettura a microkernel
  - al kernel sono assegnate solo poche, essenziali funzioni
  - spazi degli indirizzi
  - comunicazione tra processi (InterProcess Communication, IPC)
  - scheduling di base
- Sistemi operativi distribuiti
  - in grado di dare l'illusione di una memoria principale e secondaria singola
- Progettazione orientata agli oggetti
  - usata per estendere un piccolo kernel con dei moduli
  - permette al programmatore di personalizzare un sistema operativo senza mettere a repentaglio l'integrità del sistema

I moderni Sistemi Operativi

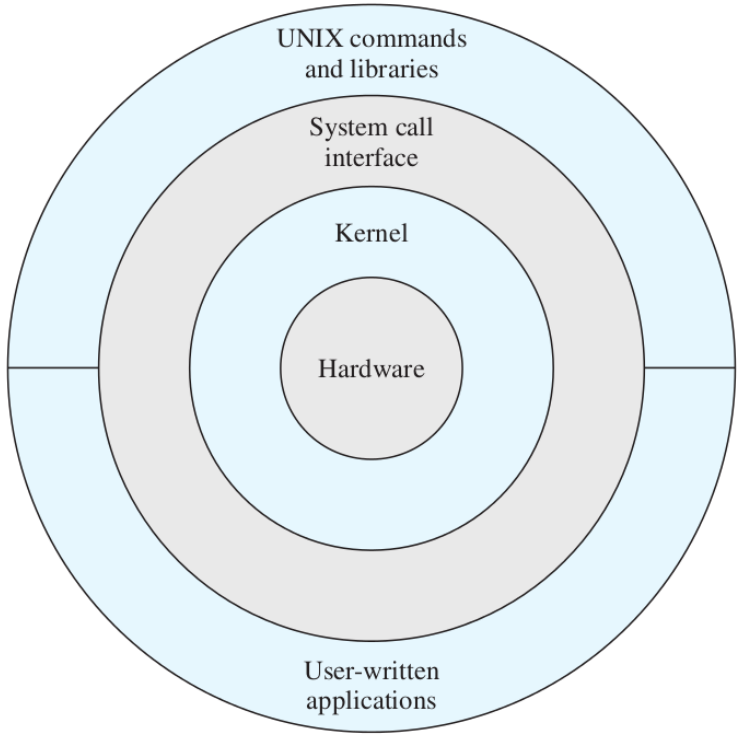
- Multithreading
  - Il processo viene diviso in threads, che possono essere eseguiti in modo concorrente
    - thread: unità di lavoro eseguibile
    - viene eseguito sequenzialmente e può essere interrotto
    - un processo è un insieme di uno o più thread
- Symmetric multiprocessing (SMP)
  - le moderne architetture offrono più di un processore
  - questi processori condividono la stessa memoria principale e gli stessi dispositivi di input/output
  - tutti i processori possono eseguire le stesse operazioni

Sistemi operativi

Il sistema operativo UNIX

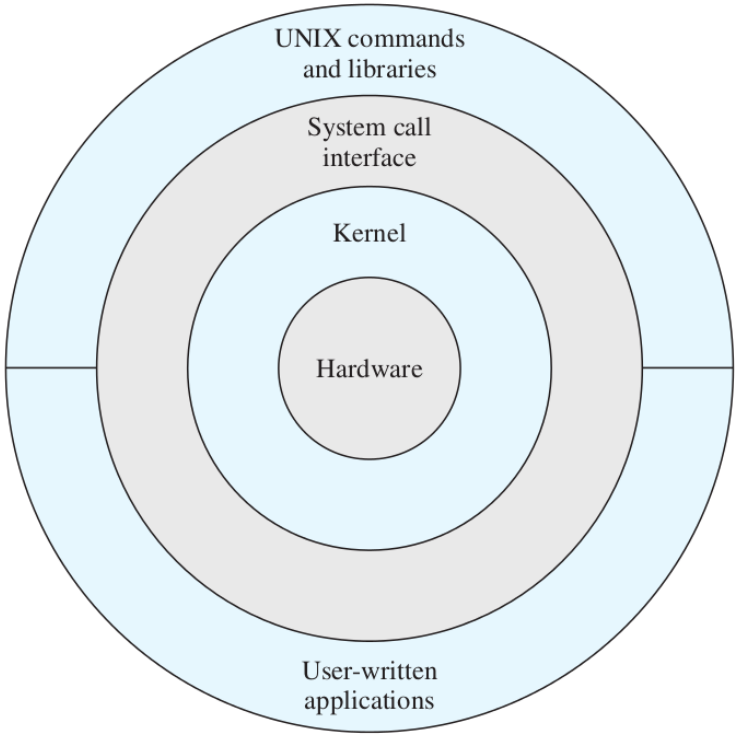
Architettura di UNIX

- UNIX viene progettato negli anni '70 (Bell Labs)
- Per la prima volta il SO viene scritto in un linguaggio ad alto livello, il C, invece che in assembly
- Anche oggi (quasi) tutte le implementazioni di UNIX sono scritte in C



Architettura di UNIX

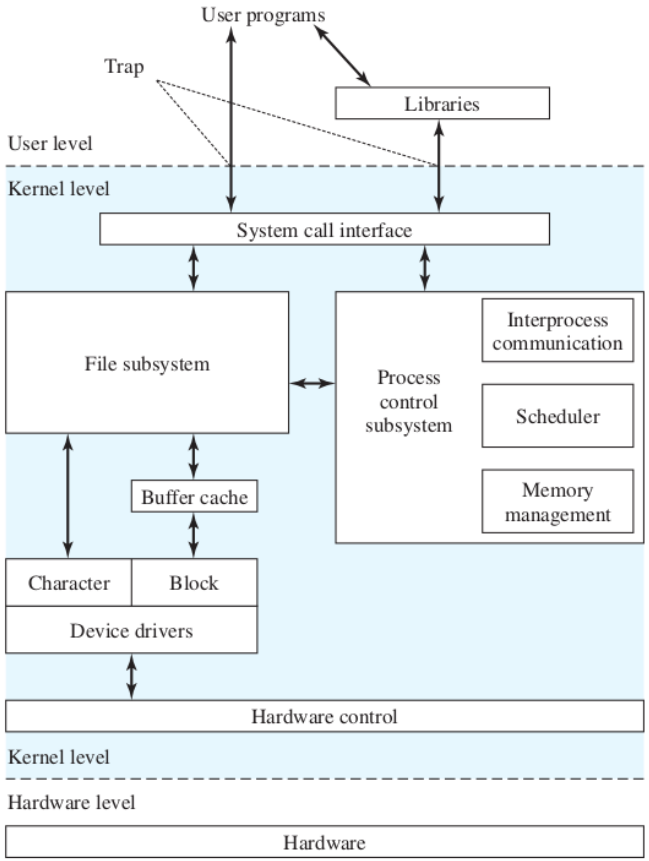
- Nello schema dell'architettura di UNIX l'hardware è completamente circondato dal SO
- Il SO viene chiamato **kernel** perchè isolato da utente e applicazioni
- UNIX comprende un insieme di servizi e interfacce considerate parte del SO:
  - shell
  - interfaccia software
  - componenti del compilatore C (loader, assembler, compiler)
  - strato più esterno con applicazioni utente e interfaccia utente/compilatore C



UNIX tradizionale

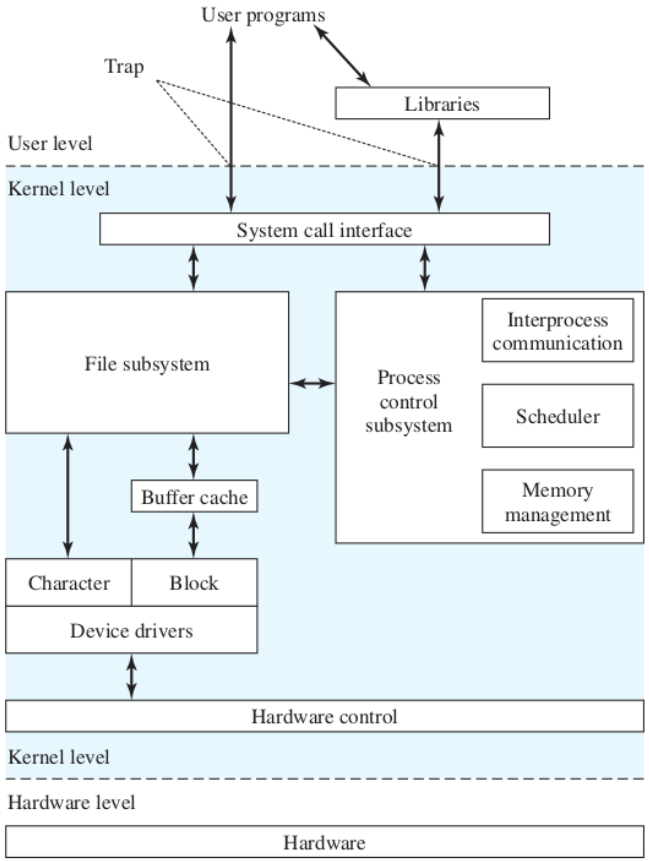
Schema del kernel

- I programmi utente invocano i servizi del SO direttamente o tramite programmi di libreria
- L'interfaccia di chiamate al sistema rappresenta il confine con l'utente e permette di accedere a funzioni specifiche del kernel tramite sw di livello più alto
- Il SO contiene routine che interagiscono direttamente con l'HW



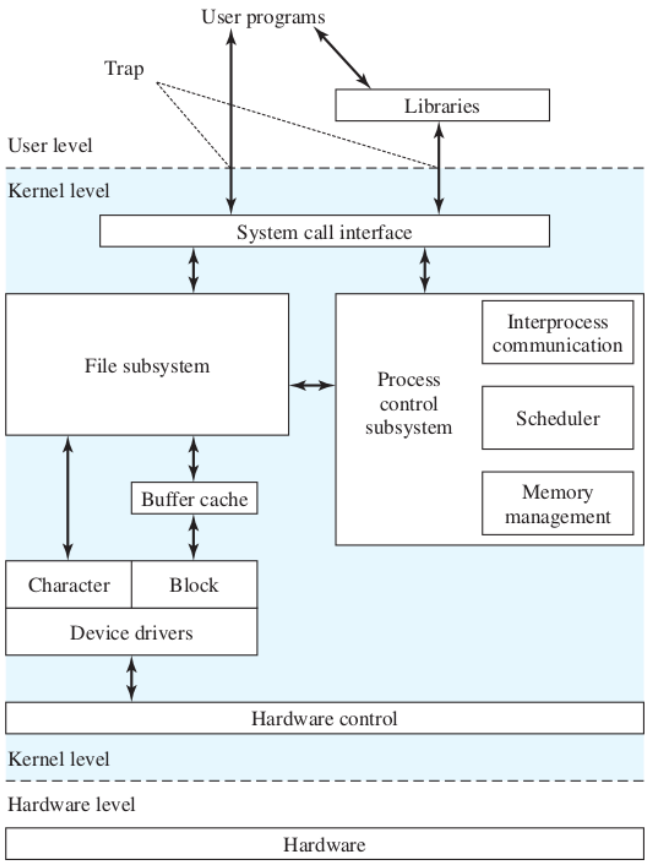
UNIX tradizionale

- Lo strato tra le due interfacce è diviso in due parti:
  - parte per il controllo di sistema: responsabile per la gestione della memoria, l'assegnazione e la sincronizzazione dei processi, la comunicazione tra processi
  - parte per la gestione dei file e dell'I/O: il file-system scambia dati tra memoria e dispositivi esterni



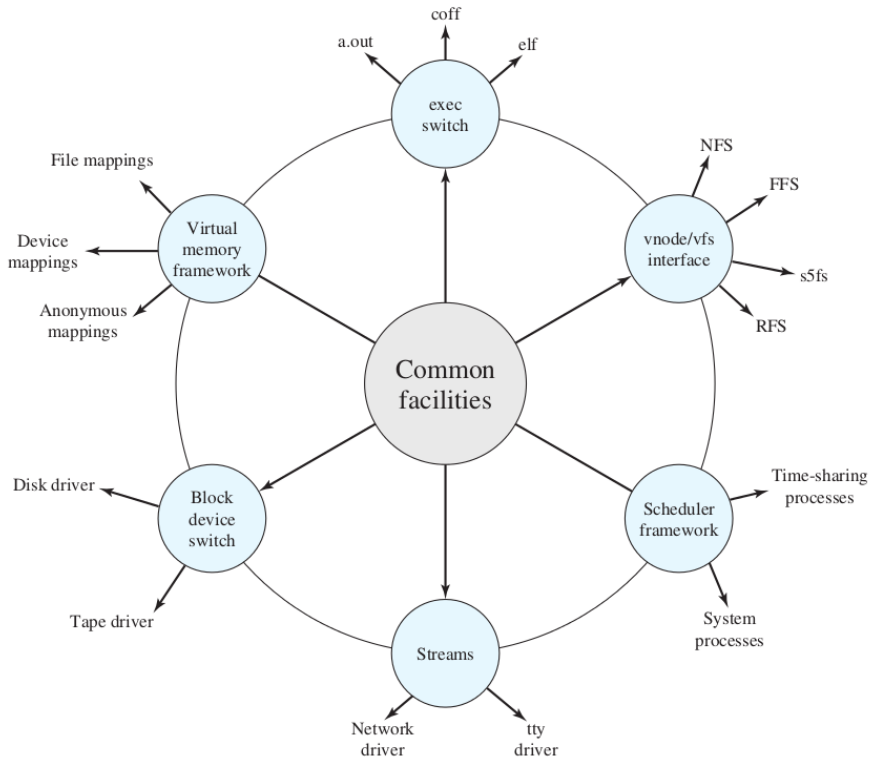
UNIX Tradizionale

- Il sistema UNIX tradizionale:
  - è stato progettato per un singolo processore e non ha la capacità di proteggere le strutture dati da accessi concorrenti di più processori
  - non ha un kernel versatile (supporta un solo tipo di file system, di politica di scheduling dei processi e formati di file eseguibili)
- L'aggiunta di nuove capacità alla versione tradizionale di UNIX ha prodotto un kernel sproporzionato, ridondante e non modulare



UNIX moderno

- Il sistema UNIX si è evoluto:
  - necessità di unificare le molte innovazioni introdotte
  - necessità di aggiungere carattestiche più attuali
  - necessità di produrre un'architettura più modulare
- C'è un piccolo **core** di funzioni e servizi scritti in maniera modulare necessari ai processi del SO
- Le funzioni nei cerchi esterni possono essere implementate in una varietà di modi



Considerazioni sul Kernel

- Molti kernel UNIX sono **monolitici**:
  - includono tutte le funzionalità del SO in un unico blocco di codice che gira come un singolo processo con un singolo spazio di memoria
  - kernel tutto in memoria dal boot allo spegnimento
  - tutte le componenti funzionali del kernel hanno accesso a tutte le strutture dati interne e a tutte le routine
  - se viene effettuato un cambiamento ad una porzione, tutti i moduli e le routine devono essere rilinkate e reinstallate e il sistema deve essere riavviato

Considerazioni sul Kernel

- Nei SO con microkernel:
  - solo una minima parte del kernel è in memoria, il resto caricato quando serve
  - sempre in memoria: scheduler, sincronizzazione
  - solo a richiesta: gestore memoria, filesystem, driver
- Un kernel monolitico è più efficiente come velocità, ma occupa più memoria e rende difficile la modularità
- Quasi tutti i sistemi operativi moderni sono a kernel monolitico
  - eccezione notabile: Mac OS X

Kernel moderno di Linux

- Linux è principalmente monolitico, ma ha i **moduli**, che gli dà vantaggi simili ai microkernel
  - è strutturato come una collezione di moduli
  - alcuni moduli particolari possono essere aggiunti e tolti a richiesta dall'immagine in memoria del kernel - **loadable modules**
  - essenzialmente, i diversi file system, i driver per determinati dispositivi di I/O, l'implementazione delle funzionalità di rete
  - un modulo non viene eseguito tramite un suo processo o thread, ma è eseguito in *kernel mode* da parte del processo corrente