Sistemi Operativi

AAF - Secondo anno - 3CFU

A.A. 2020/2021

Corso di Laurea in Matematica

Sistemi operativi - Nozioni generali

Annalisa Massini

Dipartimento di Informatica
Sapienza Università di Roma

Argomenti trattati

1. Sistemi operativi
   - Risultati principali
   - UNIX

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 principare 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

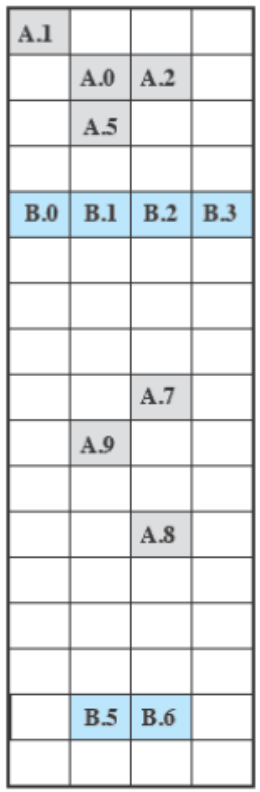
Gestione della Memoria: Memoria virtuale

- 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

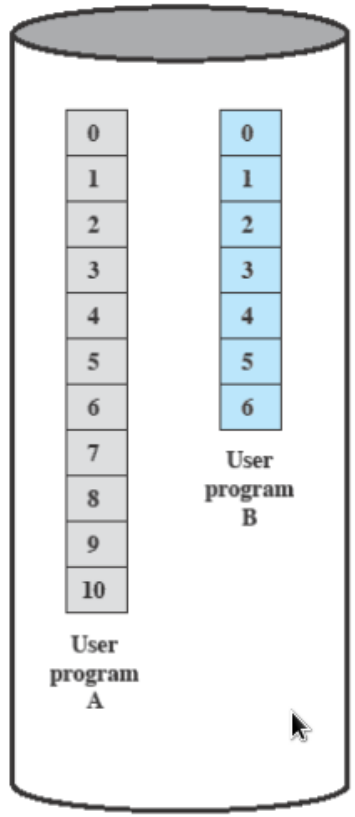
- Permette ai processi di essere contenuti in un certo numeri di blocchi di dimensione fissa, detti pagine
- L'indirizzo virtuale è un numero di pagina più uno spiazzamento (*offset*) nella pagina
- Ogni pagina può trovarsi in qualsiasi punto della memoria
- L'indirizzo vero (o fisico) è l'indirizzo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

Gestione della Memoria: memoria virtuale



Main Memory

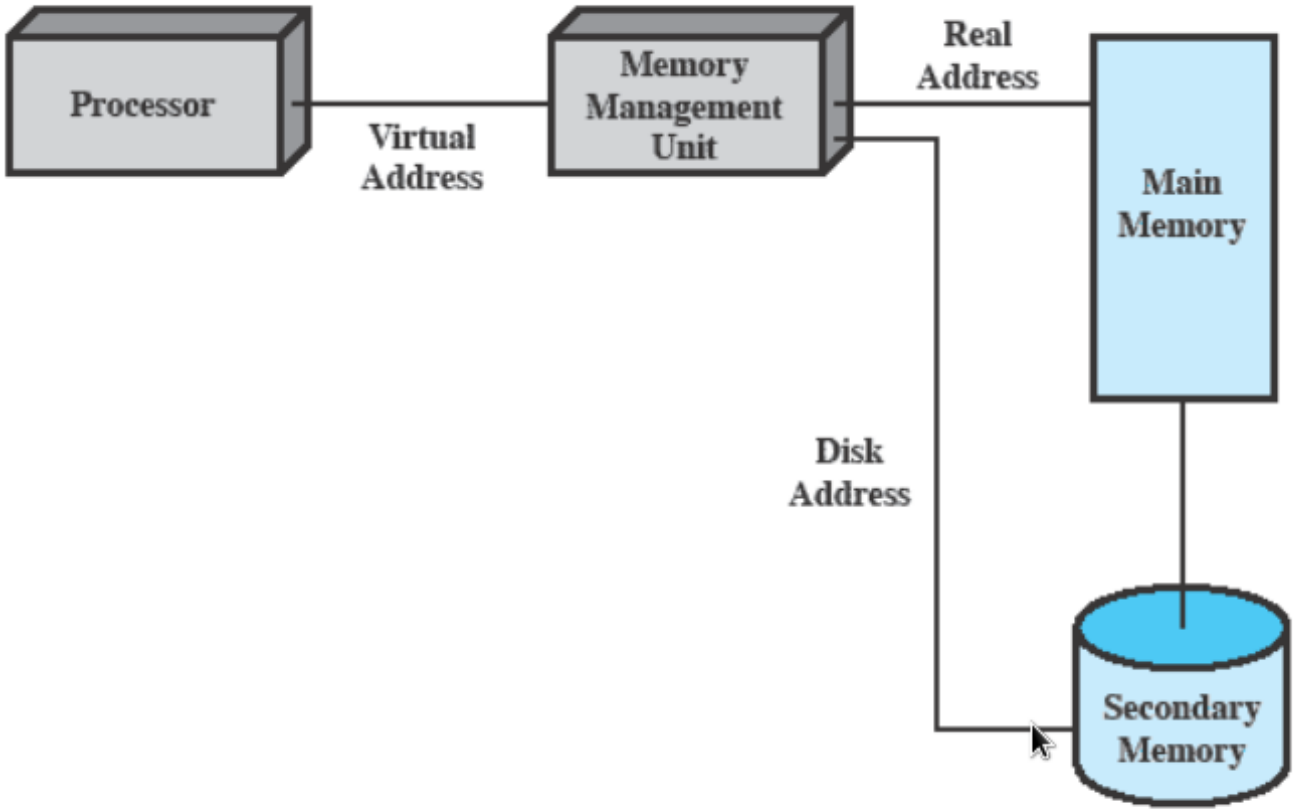
Main memory consists of a number of fixed-length frames, each equal to the size of a page. For a program to execute, some or all of its pages must be in main memory.



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



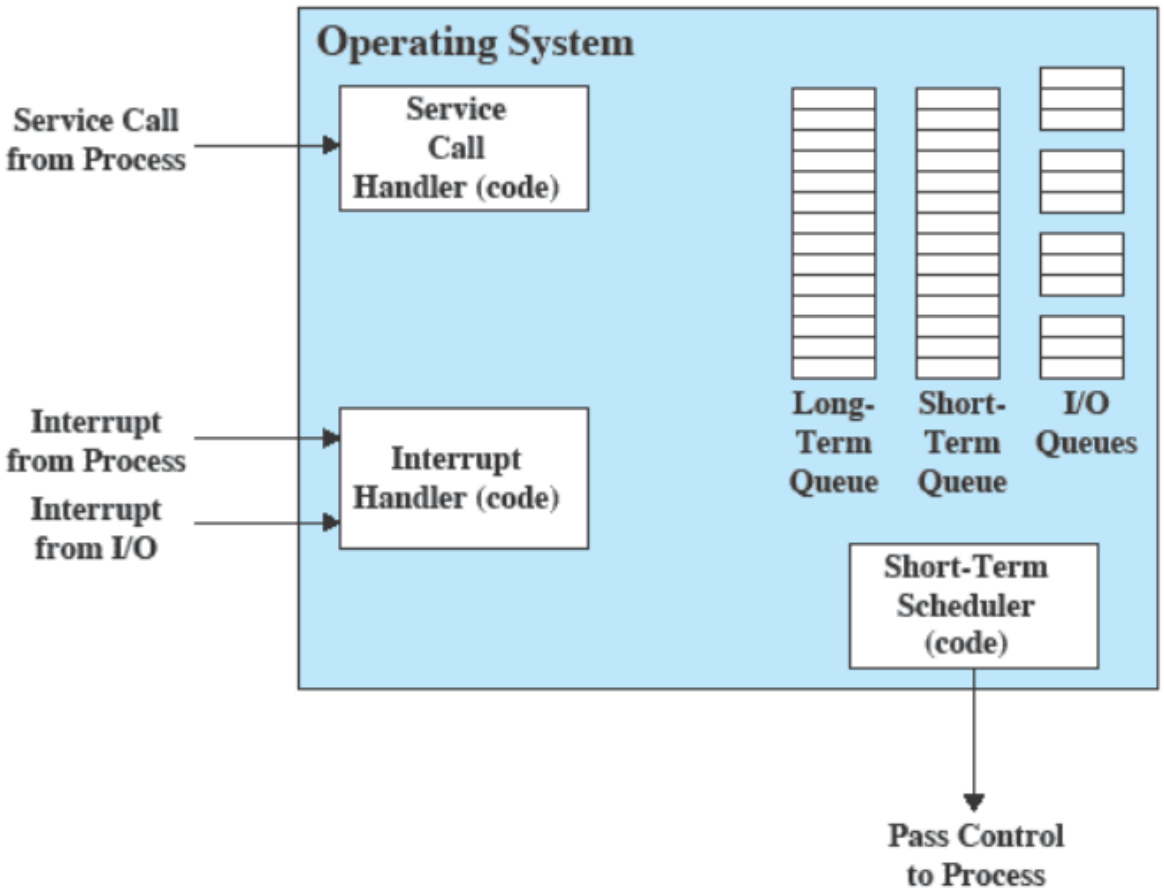
Protezione dell'Informazione e Sicurezza

- 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

Elementi per lo scheduling e l'allocazione



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**

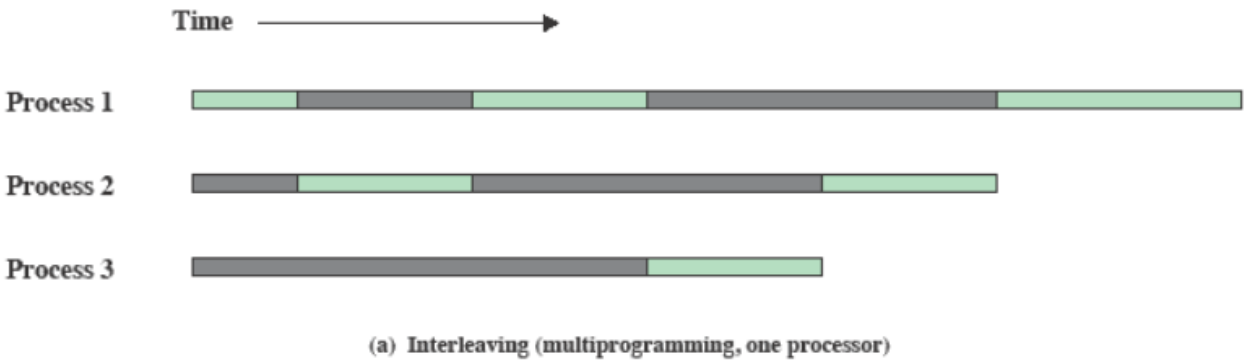
I moderni Sistemi Operativi

- 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

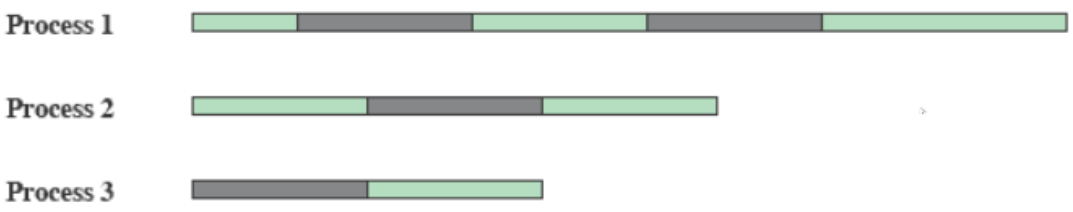
Multithreading

- 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

Multiprogrammazione e Multiprocessing



(a) Interleaving (multiprogramming, one processor)



(b) Interleaving and overlapping (multiprocessing; two processors)

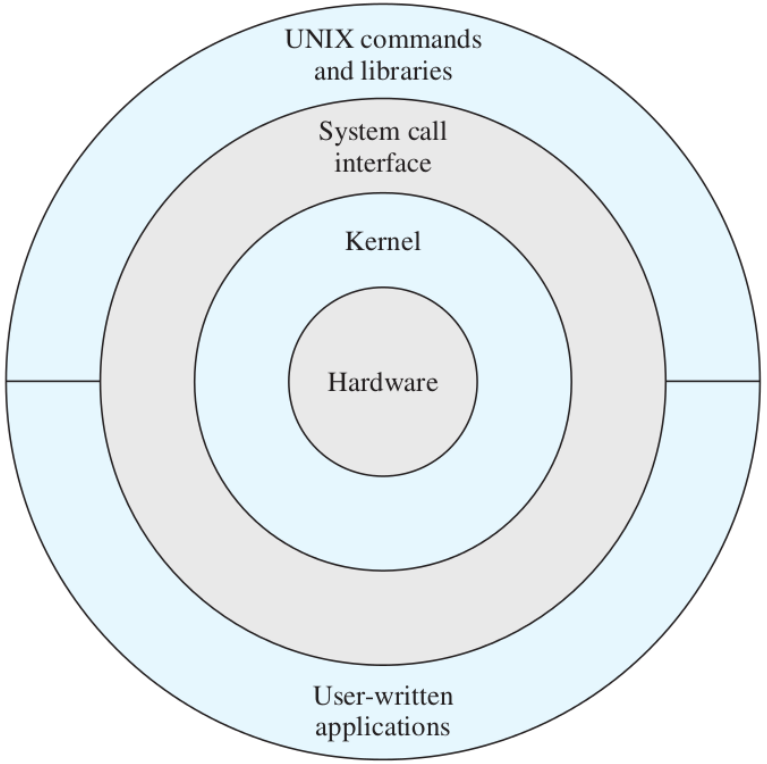
Blocked Running

Sistemi operativi

Il sistema operativo UNIX

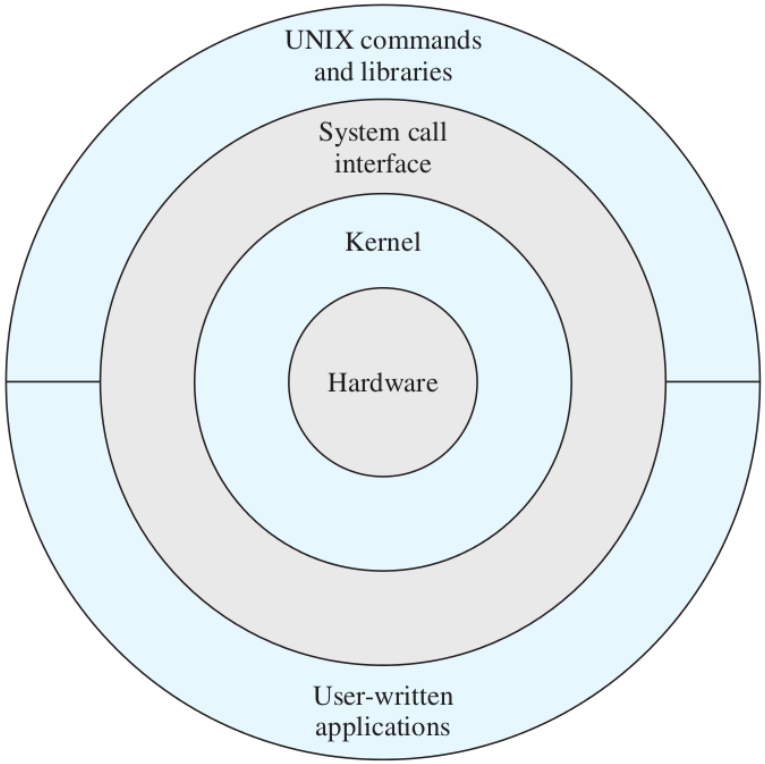
Architettura di UNIX

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



Architettura di UNIX

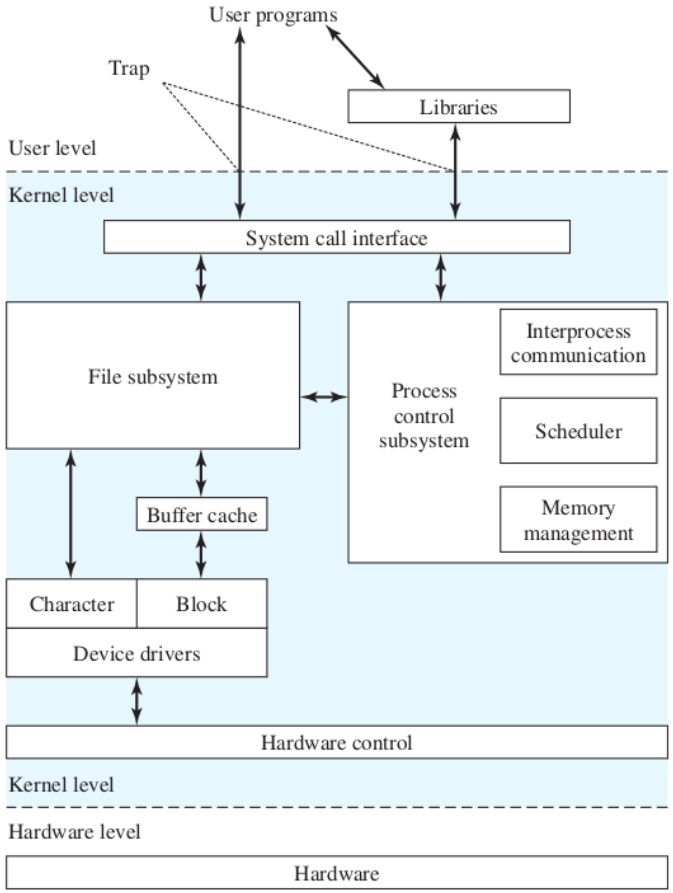
- Nello schema dell'architettura di UNIX l'hardware è completamente circondato dal SO
- Il SO viene chiamato **kernel** perchè isolato da utente e applicazioni
- UNIX comprende un insieme di servizi e interfacce considerate parte del SO:
  - shell
  - interfaccia software
  - componenti del compilatore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



UNIX tradizionale

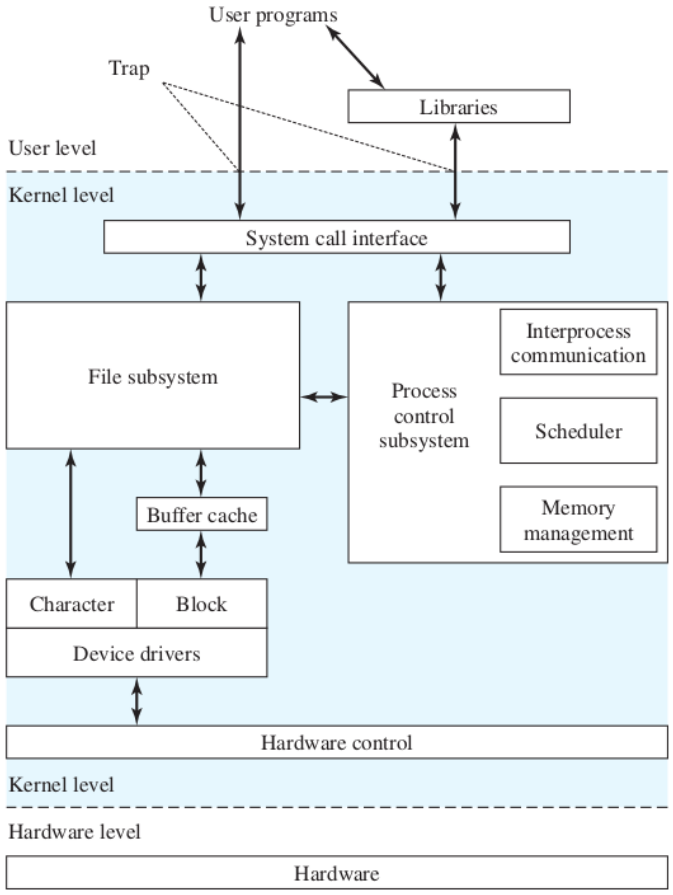
Schema del kernel

- 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



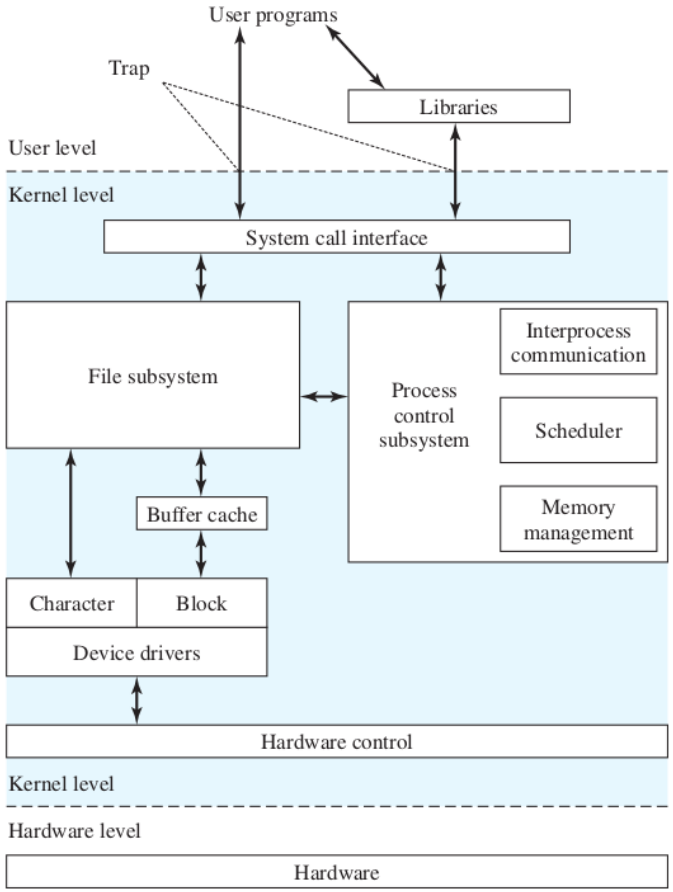
UNIX tradizionale

- 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



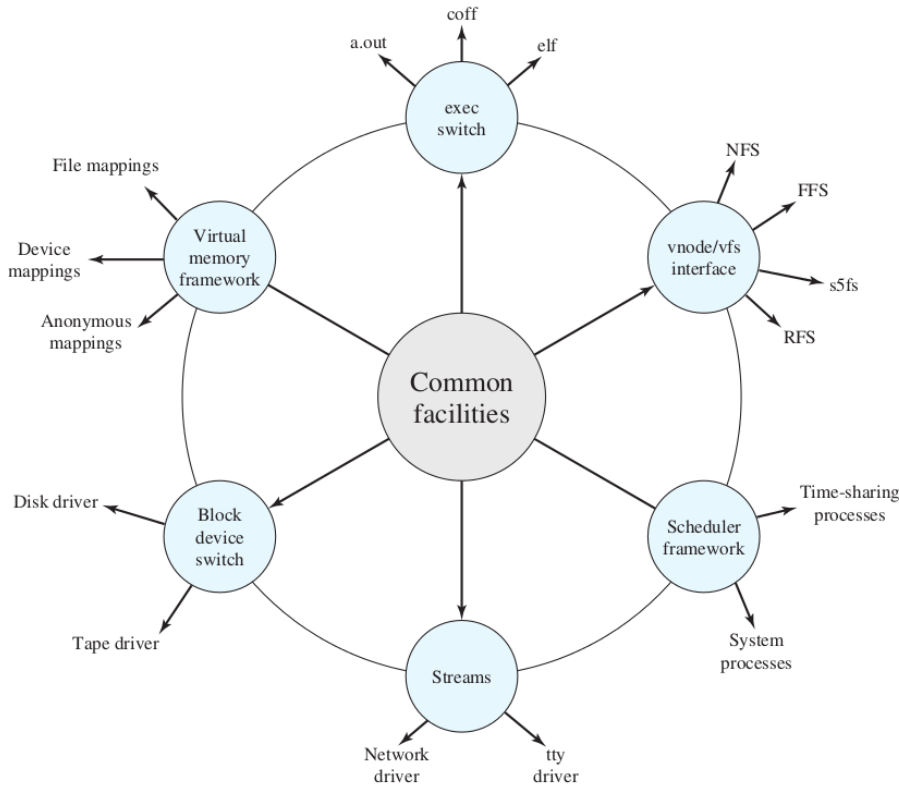
UNIX Tradizionale

- 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 che 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 access oa 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

Kernel moderno di Linux

- 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, un certo
  - 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