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

Laurea in Informatica — A.A. 2003/04

Trasparenze del Corso di *Sistemi Operativi*

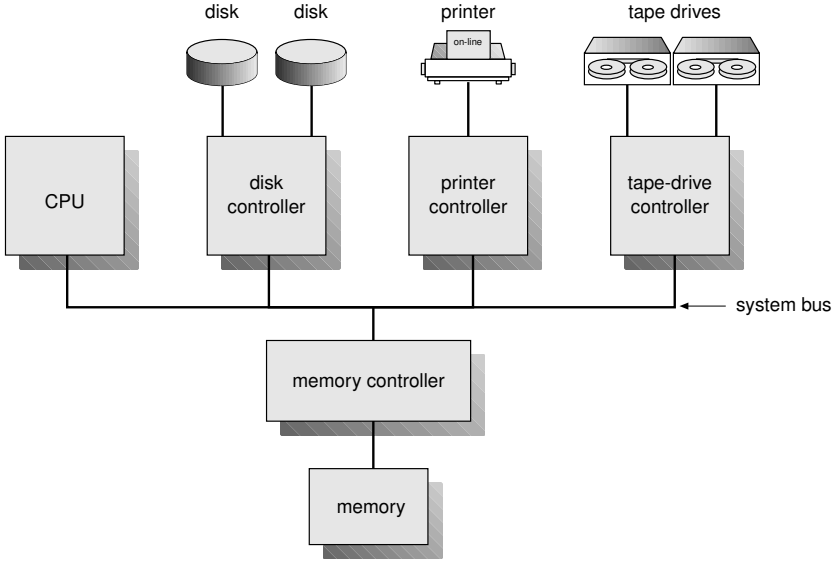
Marino Miculan

Università di Udine

Struttura dei Sistemi di Calcolo

- Operazioni dei sistemi di calcolo
- Struttura dell'I/O
- Struttura della memoria
- Gerarchia delle memorie
- Protezione hardware
- Invocazione del Sistema Operativo

Architettura dei calcolatori



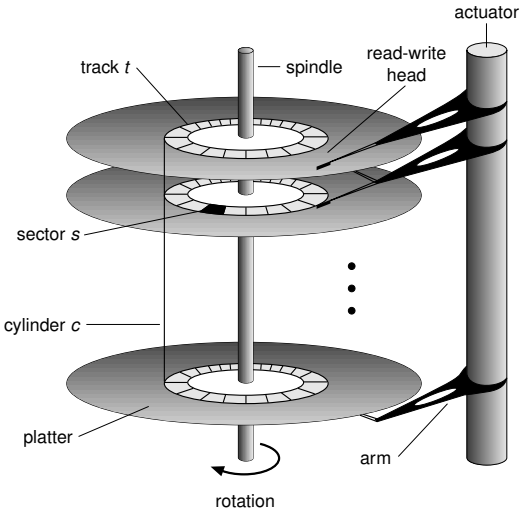
Struttura della Memoria

- Memoria principale – la memoria che la CPU può accedere direttamente.
- Memoria secondaria – estensione della memoria principale che fornisce una memoria non volatile (e solitamente più grande)

Dischi magnetici

piatti di metallo rigido ricoperti di materiale ferromagnetico, in rotazione

- la superfice del disco è logicamente divisa in *tracce*, che sono sottodivise in *settori*
- Il *controller dei dischi* determina l'interazione logica tra il dispositivo ed il calcolatore

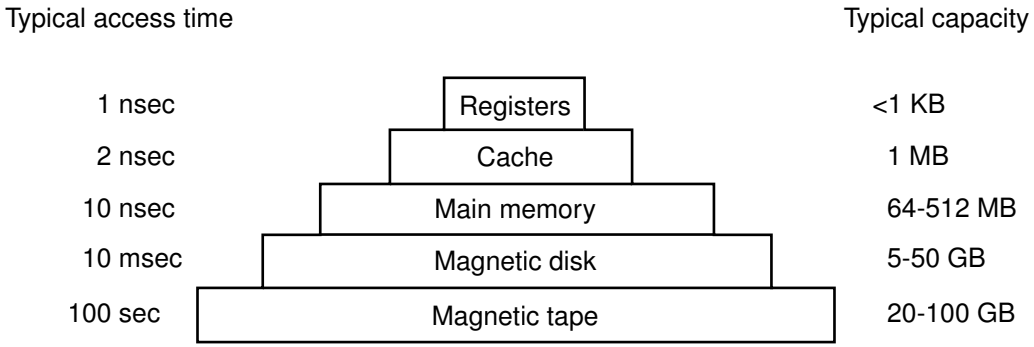


Gerarchia della Memoria

I sistemi di memorizzazione sono organizzati gerarchicamente, secondo

- velocità
- costo
- volatilità

Caching – duplicare i dati più frequentemente usati di una memoria, in una memoria più veloce. La memoria principale può essere vista come una cache per la memoria secondaria.

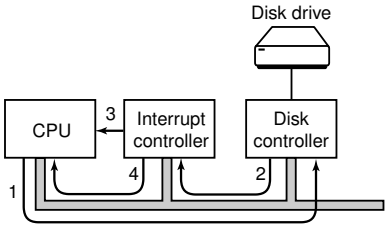


Operazioni dei sistemi di calcolo

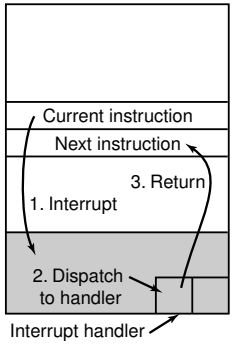
- I dispositivi di I/O e la CPU possono funzionare concorrentemente
- Ogni controller di dispositivo gestisce un particolare tipo di dispositivo.
- Ogni controller ha un buffer locale
- La CPU muovi dati da/per la memoria principale per/da i buffer locali dei controller
- l'I/O avviente tra il dispositivo e il buffer locale del controller
- Il controller informa la CPU quando ha terminato la sua operazione, generando un *interrupt*.

Schema comune degli Interrupt

- Gli interrupt trasferiscono il controllo alla routine di servizio dell'interrupt. Due modi:
  - *polling*
  - *vettore di interruzioni:* contiene gli indirizzi delle routine di servizio.



(a)



(b)

Gestione dell'interrupt

- L'hardware salva l'indirizzo dell'istruzione interrotta (p.e., sullo stack).
- Il S.O. preserva lo stato della CPU salvando registri e program counter in apposite strutture dati.
- Per ogni tipo di interrupt, uno specifico segmento di codice determina cosa deve essere fatto.
- Terminata la gestione dell'interrupt, lo stato della CPU viene riesumato e l'esecuzione del codice interrotto viene ripresa.
- Interrupt in arrivo sono *disabilitati* mentre un altro interrupt viene gestito, per evitare che vadano perduti.
- Un *trap* è un interrupt generato da software, causato o da un errore o da una esplicita richiesta dell'utente (istruzioni `TRAP`, `SVC`).
- Un sistema operativo è *guidato da interrupt*

I/O sincrono

I/O sincrono: dopo che l'I/O è partito, il controllo ritorna al programma utente solo dopo che l'I/O è stato completato

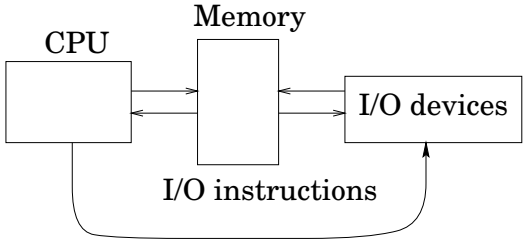
- l'istruzione **wait** blocca la CPU fino alla prossima interruzione
- oppure, un ciclo di attesa (*busy wait*); accede alla memoria
- al più una richiesta di I/O è eseguita alla volta; non ci sono I/O paralleli

I/O asincrono

I/O asincrono: dopo che l'I/O è partito, il controllo ritorna al programma utente senza aspettare che l'I/O venga completato

- *chiamata di sistema* (*System call*) – richiede al sistema operativo di sospendere il processo in attesa del completamento dell'I/O.
- una *tabella dei dispositivi* mantiene tipo, indirizzo e stato di ogni dispositivo di I/O.
- Il sistema operativo accede alla tabella dei dispositivi per determinare lo stato, e per mantenere le informazioni relative agli interrupt.

Direct Memory Access (DMA)



- Usata per dispositivi in grado di trasferire dati a velocità prossime a quelle della memoria
- I controller trasferiscono blocchi di dati dal buffer locale direttamente alla memoria, senza intervento della CPU.
- Viene generato un solo interrupt per blocco, invece di uno per ogni byte trasferito.

Protezione hardware

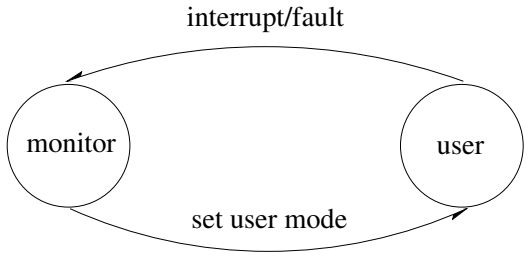
- Funzionamento in dual-mode
- Protezione dell'I/O
- Protezione della Memoria
- Protezione della CPU

Funzionamento Dual-Mode

- La condivisione di risorse di sistema richiede che il sistema operativo assicuri che un programma scorretto non possa portare altri programmi (corretti) a funzionare non correttamente.
- L'hardware deve fornire un supporto per differenziare almeno tra due modi di funzionamento
  1. *User mode* – la CPU sta eseguendo codice di un utente
  2. *Monitor mode* (anche *supervisor mode*, *system mode*, *kernel mode*) – la CPU sta eseguendo codice del sistema operativo

Funzionamento Dual-Mode (Cont.)

- La CPU ha un *Mode bit* che indica in quale modo si trova: supervisor (0) o user (1).
- Quando avviene un interrupt, l'hardware passa automaticamente in modo supervisore



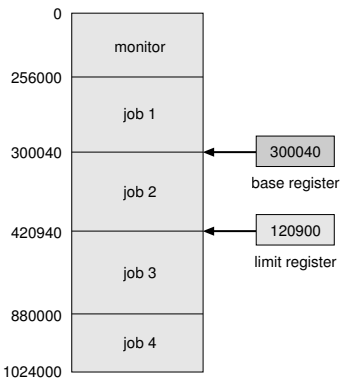
- Le *istruzioni privilegiate* possono essere eseguite solamente in modo supervisore

Protezione dell'I/O

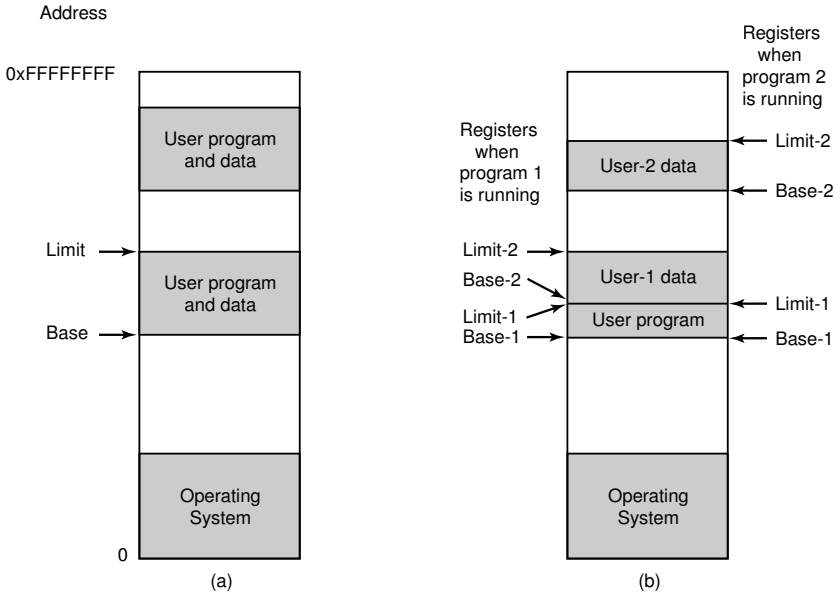
- Tutte le istruzione di I/O sono privilegiate
- Si deve assicurare che un programma utente non possa mai passare in modo supervisore (per esempio, andando a scrivere nel vettore delle interruzioni)

Protezione della Memoria

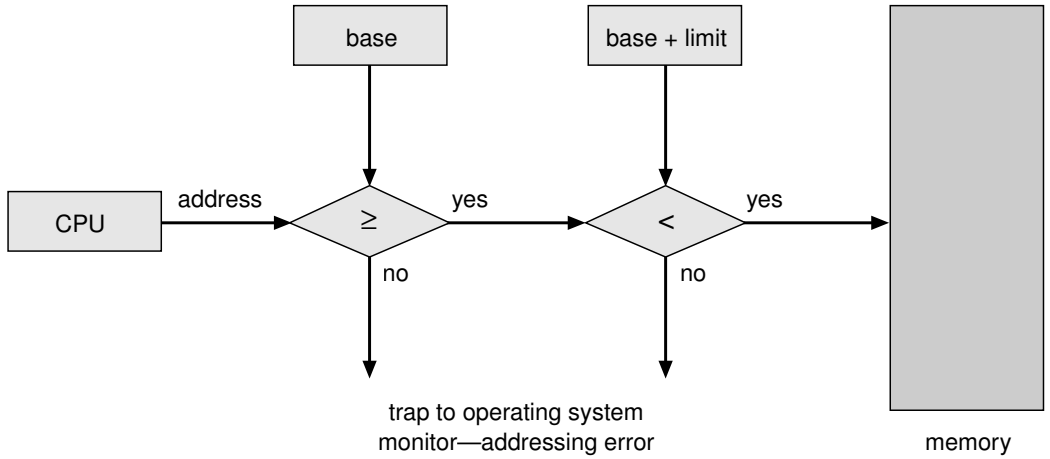
Si deve proteggere almeno il vettore delle interruzioni e le routine di gestione degli interrupt



- Per avere la protezione della memoria, si aggiungono due registri che determinano il range di indirizzi a cui un programma può accedere:
  registro base contiene il primo indirizzo fisico legale
  registro limite contiene la dimensione del range di memoria accessibile
- la memoria al di fuori di questo range è protetta



Protezione della Memoria (Cont.)



- Essendo eseguito in modo monitor, il sistema operativo ha libero accesso a tutta la memoria, sia di sistema sia utente
- Le istruzioni di caricamento dei registri base e limite sono privilegiate

Protezione della CPU

- il *Timer* interrompe la computazione dopo periodi prefissati, per assicurare che periodicamente il sistema operativo riprenda il controllo
  - Il timer viene decrementato ad ogni *tick* del clock (1/50 di secondo, tipicamente)
  - Quanto il timer va a 0, avviene l'interrupt
- Il timer viene usato comunemente per implementare il time sharing
- Serve anche per mantenere la data e l'ora
- Il caricamento del timer è una istruzione privilegiata

Invocazione del sistema operativo

- Dato che le istruzioni di I/O sono privilegiate, come può il programma utente eseguire dell'I/O?
- Attraverso le *system call* – il metodo con cui un processo richiede un'azione da parte del sistema operativo
  - Solitamente sono un interrupt software (**trap**)
  - Il controllo passa attraverso il vettore di interrupt alla routine di servizio della trap nel sistema operativo, e il mode bit viene impostato a "monitor".
  - Il sistema operativo verifica che i parametri siano legali e corretti, esegue la richiesta, e ritorna il controllo all'istruzione che segue la system call.
  - Con l'istruzione di ritorno, il mode bit viene impostato a "user"

Struttura dei Sistemi Operativi

- Componenti del sistema
- Servizi del Sistema Operativo
- Chiamate di sistema (*system calls*)
- Programmi di Sistema
- Struttura del Sistema
- Macchine Virtuali
- *Progettazione ed Implementazione del Sistema*
- *Generazione del Sistema*

Componenti comuni dei sistemi

1. Gestione dei processi
2. Gestione della Memoria Principale
3. Gestione della Memoria Secondaria
4. Gestione dell'I/O
5. Gestione dei file
6. Sistemi di protezione
7. Connessioni di rete (*networking*)
8. Sistema di interpretazione dei comandi

Gestione dei processi

- Un *processo* è un programma in esecuzione. Un processo necessita di certe risorse, tra cui tempo di CPU, memoria, file, dispositivi di I/O, per assolvere il suo compito.
- Il sistema operativo è responsabile delle seguenti attività, relative alla gestione dei processi:
  - creazione e cancellazione dei processi
  - sospensione e riesumazione dei processi
  - fornire meccanismi per
    - sincronizzazione dei processi
    - comunicazione tra processi
    - evitare, prevenire e risolvere i *deadlock*

Gestione della Memoria Principale

- La *memoria principale* è un (grande) array di parole (byte, words. . .), ognuna identificata da un preciso indirizzo. È un deposito di dati rapidamente accessibili dalla CPU e dai dispositivi di I/O.
- La memoria principale è *volatile*. Perde il suo contenuto in caso di *system failure.*
- Il sistema operativo è responsabile delle seguenti attività relative alla gestione della memoria:
  - Tener traccia di quali parti della memoria sono correntemente utilizzate, e da chi.
  - Decidere quale processo caricare in memoria, quando dello spazio si rende disponibile.
  - Allocare e deallocare spazio in memoria, su richiesta.

Gestione della memoria secondaria

- Dal momento che la memoria principale è volatile e troppo piccola per contenere tutti i dati e programmi permanentemente, il calcolatore deve prevedere anche una *memoria secondaria* di supporto a quella principale.
- La maggior parte dei calcolatori moderni utilizza *dischi* come principale supporto per la memoria secondaria, sia per i programmi che per i dati.
- Il sistema operativo è responsabile delle seguenti attività relative alla gestione dei dischi:
  - Gestione dello spazio libero
  - Allocazione dello spazio
  - Schedulazione dei dischi

Gestione del sistema di I/O

- Il sistema di I/O consiste in
  - un sistema di cache a buffer
  - una interfaccia generale ai gestori dei dispositivi (*device driver*)
  - i driver per ogni specifico dispositivo hardware (controller)

Gestione dei File

- Un *file* è una collezione di informazioni correlate, definite dal suo creatore. Comunemente, i file rappresentano programmi (sia sorgenti che eseguibili (*oggetti*)) e dati.
- Il sistema operativo è responsabile delle seguenti attività connesse alla gestione dei file:
  - Creazione e cancellazione dei file
  - Creazione e cancellazione delle directory
  - Supporto di primitive per la manipolazione di file e directory
  - Allocazione dei file nella memoria secondaria
  - Salvataggio dei dati su supporti non volatili

Sistemi di protezione

- Per *Protezione* si intende un meccanismo per controllare l'accesso da programmi, processi e utenti sia al sistema, sia alle risorse degli utenti.
- Il meccanismo di protezione deve:
  - distinguere tra uso autorizzato e non autorizzato.
  - fornire un modo per specificare i controlli da imporre
  - forzare gli utenti e i processi a sottostare ai controlli richiesti

Networking (Sistemi Distribuiti)

- Un *sistema distribuito* è una collezione di processori che non condividono memoria o clock. Ogni processore ha una memoria propria.
- I processori del sistema sono connessi attraverso una *rete di comunicazione.*
- Un sistema distribuito fornisce agli utenti l'accesso a diverse risorse di sistema.
- L'accesso ad una risorsa condivisa permette:
  - Aumento delle prestazioni computazionali
  - Incremento della quantità di dati disponibili
  - Aumento dell'affidabilità

Interprete dei comandi

- Molti comandi sono dati al sistema operativo attraverso *control statement* che servono per
  - creare e gestire i processi
  - gestione dell'I/O
  - gestione della memoria secondaria
  - gestione della memoria principale
  - accesso al file system
  - protezione
  - networking

Interprete dei comandi (Cont.)

- Il programma che legge e interpreta i comandi di controllo ha diversi nomi:
  - interprete delle schede di controllo (sistemi batch)
  - interprete della linea di comando (DOS, Windows)
  - shell (in UNIX)
  - interfaccia grafica: Finder in MacOS, Explorer in Windows, gnome-session in Unix. . .

  La sua funzione è di ricevere un comando, eseguirlo, e ripetere.

Servizi dei Sistemi Operativi

- Esecuzione dei programmi: caricamento dei programmi in memoria ed esecuzione.
- Operazioni di I/O: il sistema operativo deve fornire un modo per condurre le operazioni di I/O, dato che gli utenti non possono eseguirle direttamente,
- Manipolazione del file system: capacità di creare, cancellare, leggere, scrivere file e directory.
- Comunicazioni: scambio di informazioni tra processi in esecuzione sullo stesso computer o su sistemi diversi collegati da una rete. Implementati attraverso *memoria condivisa* o *passaggio di messaggi*.
- Individuazione di errori: garantire una computazione corretta individuando errori nell'hardware della CPU o della memoria, nei dispositivi di I/O, o nei programmi degli utenti.

Funzionalità addizionali dei sistemi operativi

Le funzionalità addizionali esistono per assicurare l'efficienza del sistema, piuttosto che per aiutare l'utente

- Allocazione delle risorse: allocare risorse a più utenti o processi, allo stesso momento
- Accounting: tener traccia di chi usa cosa, a scopi statistici o di rendicontazione
- Protezione: assicurare che tutti gli accessi alle risorse di sistema siano controllate

Chiamate di Sistema (System Calls)

- Le chiamate di sistema formano l'interfaccia tra un programma in esecuzione e il sistema operativo.
  - Generalmente, sono disponibili come speciali istruzioni assembler
  - Linguaggi pensati per programmazione di sistema permettono di eseguire system call direttamente (e.g., C, Bliss, PL/360).
- Tre metodi generali per passare parametri tra il programma e il sistema operativo:
  - Passare i parametri nei *registri*.
  - Memorizzare i parametri in una tabella in memoria, il cui indirizzo è passato come parametro in un registro.
  - Il programma *push*a i parametri sullo *stack*, e il sistema operativo li *pop*pa.

Tipi di chiamate di sistema

Controllo dei processi: creazione/terminazione processi, esecuzione programmi, (de)allocazione memoria, attesa di eventi, impostazione degli attributi,...

Gestione dei file: creazione/cancellazione, apertura/chiusura, lettura/scrittura, impostazione degli attributi,...

Gestione dei dispositivi: allocazione/rilascio dispositivi, lettura/scrittura, collegamento logico dei dispositivi (e.g. mounting)...

Informazioni di sistema: leggere/scrivere data e ora del sistema, informazioni sull'hardware/software installato,...

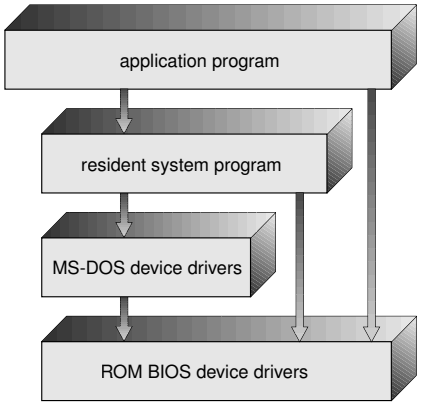
Comunicazioni: creare/cancellare connessioni, spedire/ricevere messaggi,...

Programmi di sistema

- I programmi di sistema forniscono un ambiente per lo sviluppo e l'esecuzione dei programmi. Si dividono in
  - Gestione dei file
  - Modifiche dei file
  - Informazioni sullo stato del sistema e dell'utente
  - Supporto dei linguaggi di programmazione
  - Caricamento ed esecuzione dei programmi
  - Comunicazioni
  - Programmi applicativi
- La maggior parte di ciò che un utente vede di un sistema operativo è definito dai programmi di sistema, non dalle reali chiamate di sistema.

Struttura dei Sistemi Operativi – L'approccio semplice

- MS-DOS – pensato per fornire le massime funzionalità nel minore spazio possibile.
  - non è diviso in moduli (è cresciuto oltre il previsto)
  - nonostante ci sia un po' di struttura, le sue interfacce e livelli funzionali non sono ben separati.



Struttura dei Sistemi Operativi – L'approccio semplice (Cont

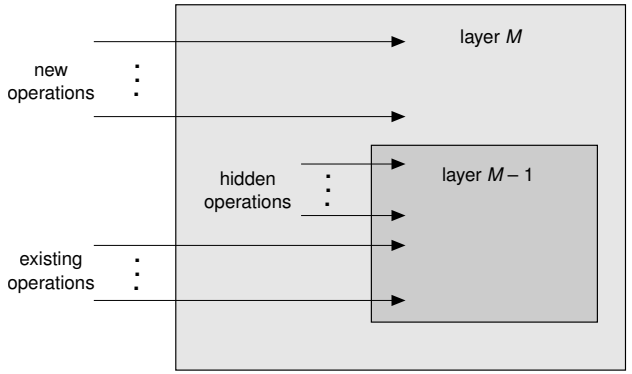
- UNIX – limitato dalle funzionalità hardware, lo UNIX originale aveva una debole strutturazione. Consiste almeno in due parti ben separate:
  - Programmi di sistema
  - Il kernel
    - consiste in tutto ciò che sta tra le system call e l'hardware
    - implementa il file system, lo scheduling della CPU, gestione della memoria e altre funzioni del sistema operativo: molte funzionalità in un solo livello.

Struttura dei Sistemi Operativi – Unix originale

(the users)		
shells and commands compilers and interpreters system libraries		
system-call interface to the kernel		
signals terminal handling character I/O system terminal drivers	file system swapping block I/O system disk and tape drivers	CPU scheduling page replacement demand paging virtual memory
kernel interface to the hardware		
terminal controllers terminals	device controllers disks and tapes	memory controllers physical memory

Struttura dei sistemi operativi – Approccio stratificato

- Il sistema operativo è diviso in un certo numero di strati (livelli); ogni strato è costruito su quelli inferiori. Lo strato di base (livello 0) è l'hardware; il più alto è l'interfaccia utente.
- Secondo la modularità, gli strati sono pensati in modo tale che ognuno utilizza funzionalità (operazioni) e servizi solamente di strati inferiori.

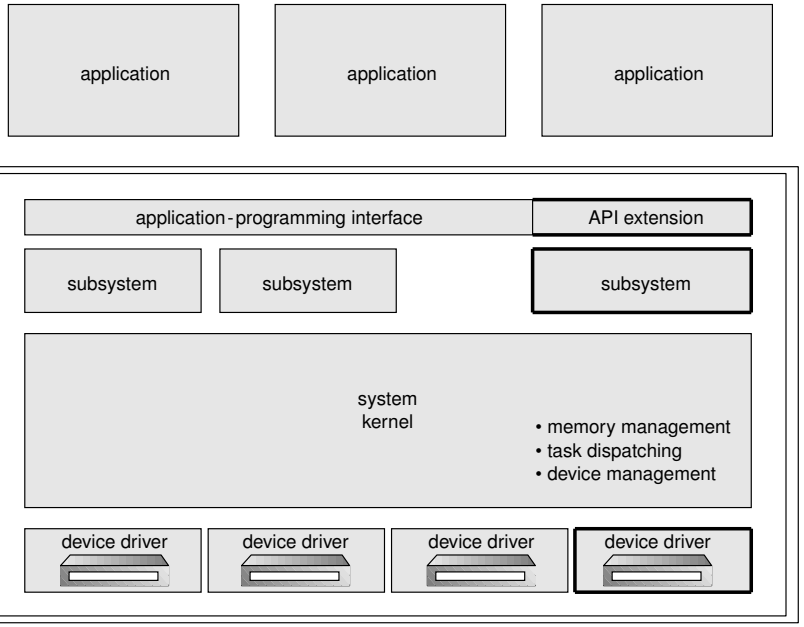


Struttura dei sistemi operativi – Stratificazione di THE OS

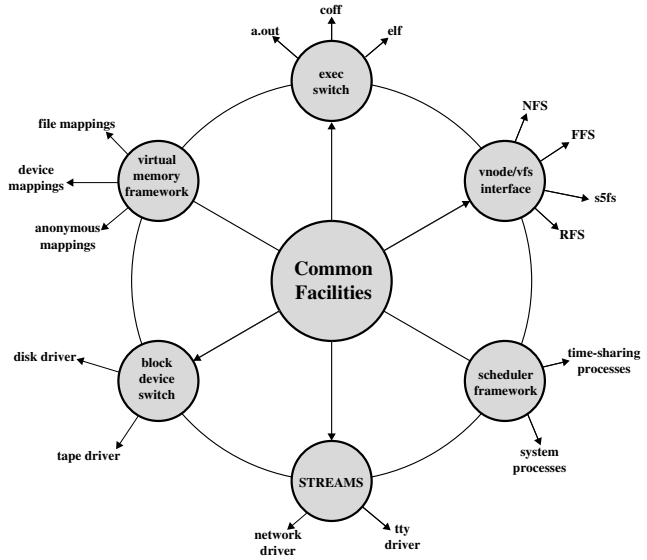
- La prima stratificazione fu usata nel SO THE. Sei strati come segue:

layer 5:	user programs
layer 4:	buffering for input and output devices
layer 3:	operator-console device driver
layer 2:	memory management
layer 1:	CPU scheduling
layer 0:	hardware

Struttura dei Sistemi Operativi – Stratificazione di OS/2



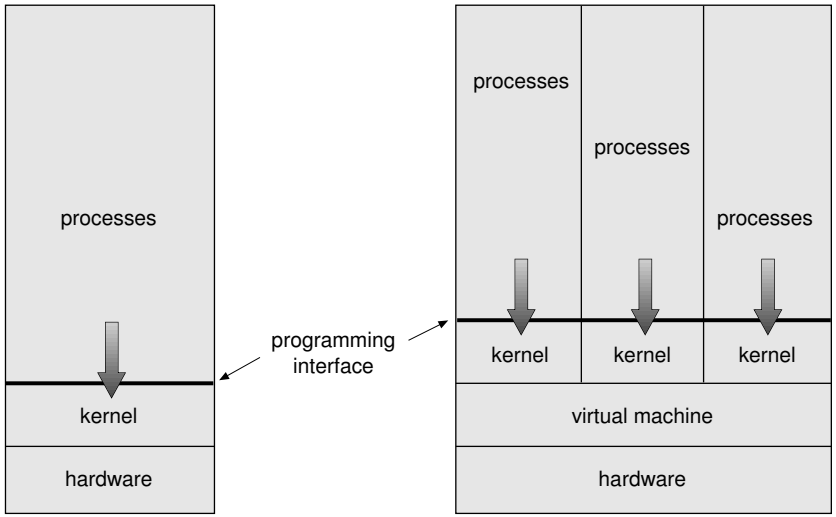
Struttura dei Sistemi Operativi – Solaris



Macchine Virtuali

- Una *macchina virtuale* porta l'approccio stratificato all'estremo: tratta hardware e il sistema operativo come se fosse tutto hardware.
- Una macchina virtuale fornisce una interfaccia *identica* all'hardware nudo e crudo sottostante.
- Il sistema operativo impiega le risorse del calcolatore fisico per creare le macchine virtuali:
  - Lo scheduling della CPU crea l'illusione che ogni processo abbia il suo processore dedicato.
  - La gestione della memoria crea l'illusione di una memoria virtuale per ogni processo
  - Lo spooling può implementare delle stampanti virtuali
  - Spazio disco può essere impiegato per creare "dischi virtuali"

Macchine Virtuali (Cont.)



(a) Macchina non virtuale; (b) Macchine virtuali

Vantaggi/Svantaggi delle Macchine Virtuali

- Il concetto di macchina virtuale fornisce una protezione completa delle risorse di sistema, dal momento che ogni macchina virtuale è isolata dalle altre. Questo isolamento non permette però una condivisione diretta delle risorse.
- Un sistema a macchine virtuali è un mezzo perfetto per l'emulazione di altri sistemi operativi, o lo sviluppo di nuovi sistemi operativi: tutto si svolge sulla macchina virtuale, invece che su quella fisica, quindi non c'è pericolo di far danni.
- Implementare una macchina virtuale è complesso, in quanto si deve fornire un *perfetto* duplicato della macchina sottostante. Può essere necessario dover emulare ogni singola istruzione macchina.
- Approccio seguito in molti sistemi: Windows, Linux, MacOS, JVM,...

Exokernel

- Estensione dell'idea di macchina virtuale
- Ogni macchina virtuale di livello utente vede solo un *sottoinsieme* delle risorse dell'intera macchina
- Ogni macchina virtuale può eseguire il proprio sistema operativo
- Le risorse vengono richieste all'exokernel, che tiene traccia di quali risorse sono usate da chi
- Semplifica l'uso delle risorse allocate: l'exokernel deve solo tenere separati i domini di allocazione delle risorse

Meccanismi e Politiche

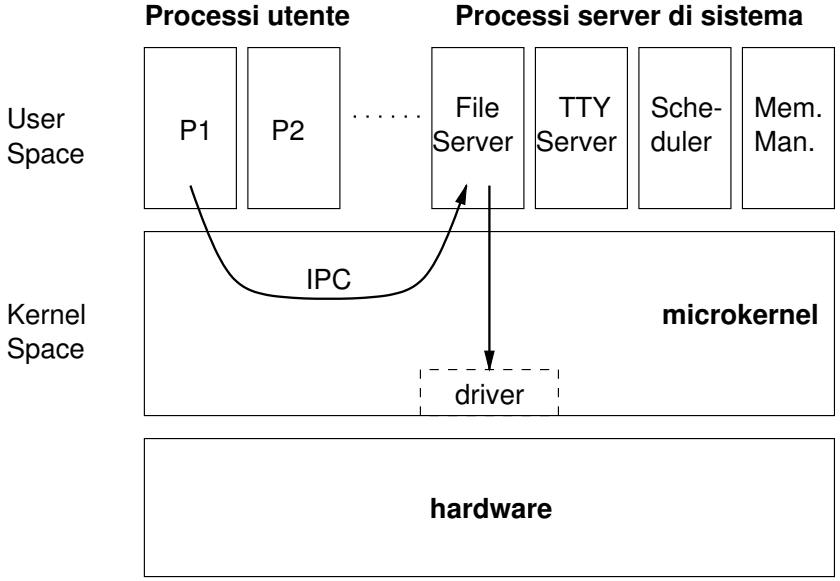
- I kernel tradizionali (monolitici) sono poco flessibili
- Distinguere tra *meccanismi* e *politiche*:
  - i meccanismi determinano *come* fare qualcosa;
  - le politiche determinano *cosa* deve essere fatto.

  Ad esempio: assegnare l'esecuzione ad un processo è un meccanismo; scegliere *quale* processo attivare è una politica.
- Questa separazione è un principio molto importante: permette la massima flessibilità, nel caso in cui le politiche debbano essere cambiate.
- Estremizzazione: il kernel fornisce solo i meccanismi, mentre le politiche vengono implementate in user space.

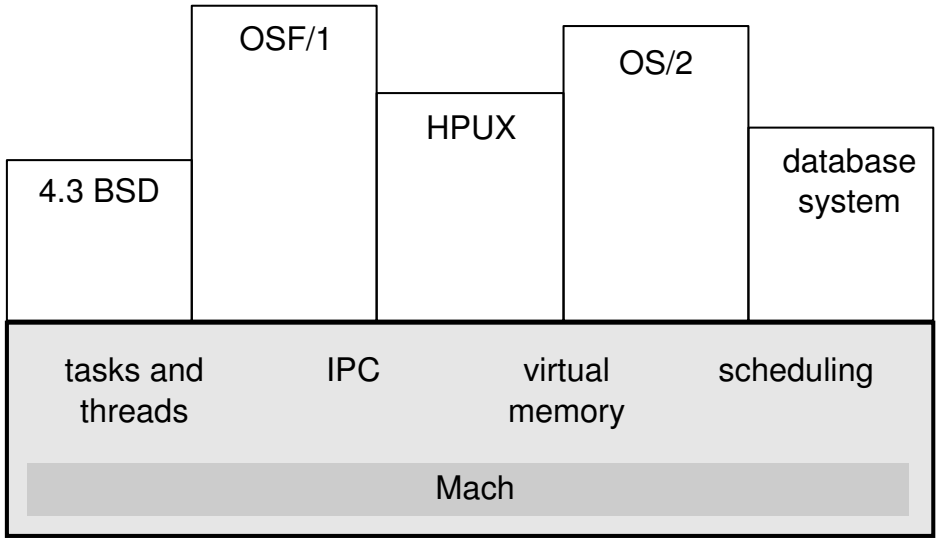
Sistemi con Microkernel

- *Microkernel*: il kernel è ridotto all'osso, fornisce soltanto i meccanismi:
  - Un meccanismo di comunicazione tra processi
  - Una minima gestione della memoria e dei processi
  - Gestione dell'hardware di basso livello (driver)
- Tutto il resto viene gestito da processi in spazio utente: ad esempio, tutte le politiche di gestione del file system, dello scheduling, della memoria sono implementate come processi.
- Meno efficiente del kernel monolitico
- Grande flessibilità; immediata scalabilità in ambiente di rete
- Sistemi operativi recenti sono basati, in diverse misure, su microkernel (AIX4, BeOS, GNU HURD, MacOS X, QNX, Tru64, Windows NT ...)

Microkernel: funzionamento di base



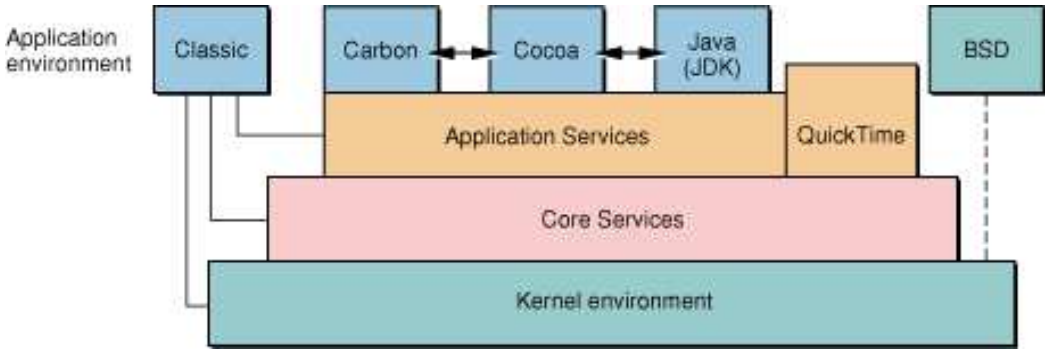
Microkernel: Mach



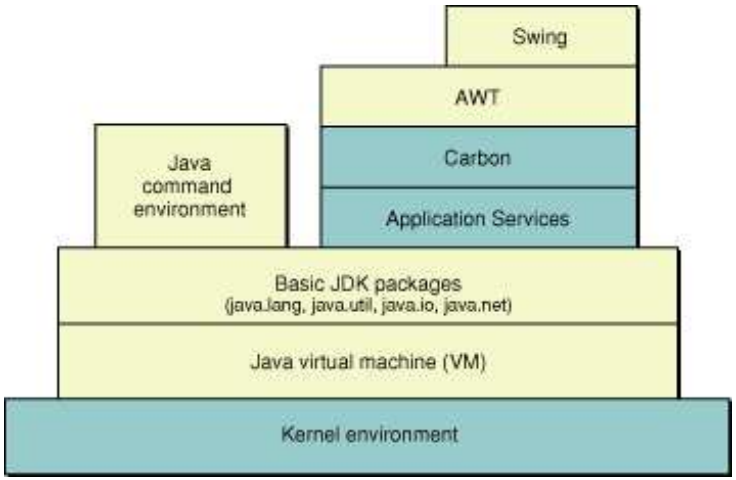
Soluzione stratificata/microkernel: Mac OS X

- Application (or execution) environments: Carbon, Cocoa, Java, Classic, and BSD Commands.
- Application Services. Servizi di sistema usati da tutti gli ambienti applicativi (collegati con la GUI). Quartz, QuickDraw, OpenGL. . . .
- Core Services. Servizi comuni, non collegati alla GUI. Networking, tasks, . . .
- Kernel environment. Mach 3.0 per task, thread, device drivers, gestione della memoria, + BSD che implementa rete, file system, thread POSIX.

Soluzione stratificata/microkernel: Mac OS X



Soluzione stratificata/microkernel: Mac OS X



Soluzione stratificata non più microkernel: Windows 2000