Classificazione di system calls

A. Gestione dei processi

B. Manipolazione dei file

C. Manipolazione di dispositivi (memorie e I/O)

D. Manutenzione delle informazioni

E. Comunicazioni

Che novita'!

A. Gestione dei processi

Process management

Call	Description
pid = fork( )	Create a child process identical to the parent
pid = waitpid(pid, &statloc, options)	Wait for a child to terminate
s = execve(name, argv, environp)	Replace a process' core image
exit(status)	Terminate process execution and return status

- Fork ⇢ ***ritorno del controllo dopo la creazione ?!***
- Execute
- End
- Abort
- …

Debugging e monitoring

- Esecuzione step by step
- Profilo temporale del program counter

B. Manipolazione dei file

File management

Call	Description
fd = open(file, how, ...)	Open a file for reading, writing or both
s = close(fd)	Close an open file
n = read(fd, buffer, nbytes)	Read data from a file into a buffer
n = write(fd, buffer, nbytes)	Write data from a buffer into a file
position = lseek(fd, offset, whence)	Move the file pointer
s = stat(name, &buf)	Get a file's status information

- Create
- Delete
- Open
- Read
- …

C. Manipolazione di dispositivi

- Request
- Release
- Alloc
- …

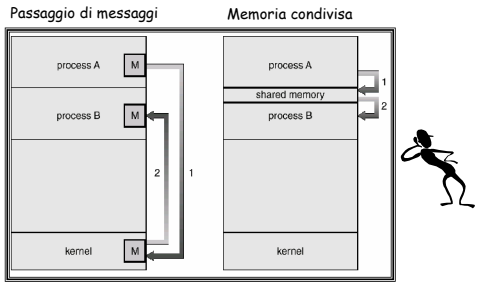
D. Manutenzione delle informazioni

- Time
- Date
- …

E. Comunicazioni…

- Get Hostid
- Open connection
- Close connection
- Send
- …

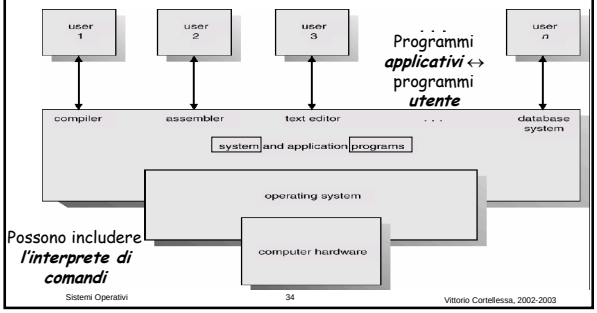
… e due modelli di comunicazione



Daemon : ogni processo speciale di OS che gestisce la comunicazione tra processi

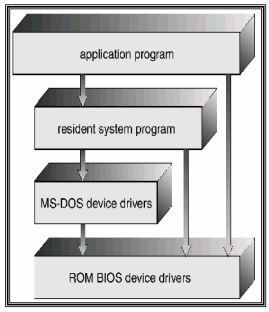
Programmi di Sistema

Di differente complessita', vanno da ***semplici interfacce a system calls*** a programmi complessi quali i ***compilatori***



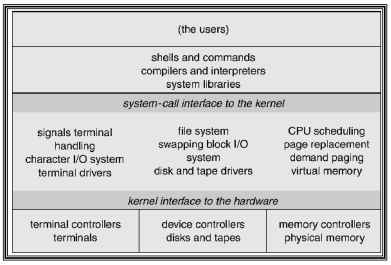
Possono includere ***l'interprete di comandi***

Struttura interna di un OS - un esempio di design : MS-DOS



Interfacce e livelli di funzionalita' non sono ben separati

… ed un altro esempio meglio strutturato: UNIX

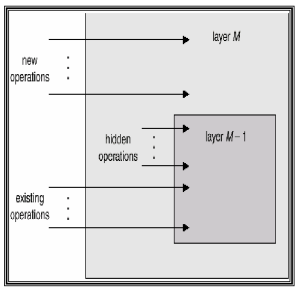


Il ***kernel*** consiste di tutto cio' che sta al di sotto delle interfacce alle system call e al di sopra dell'hardware

Un criterio generale di design: approccio stratificato

- OS e' suddiviso in un numero di livelli, ognuno ***costruito sui sottostanti***
- Il livello piu' basso e' ***l'hardware*** (livello 0) e il piu' alto e' ***l'interfaccia utente*** (livello N)

Relazioni tra livelli… e caratteristiche



- Information hiding
- Data encapsulation
- Modularita'
- Facile verifica e debugging
- Bassa efficienza

Da stratificazione a MACCHINA VIRTUALE

Usualmente si definisce macchina virtuale l'insieme costituito dall'***hardware*** e dal ***kernel di OS***

Ogni utente ha una ***copia identica dell'immagine della macchina*** che sta utilizzando

E' il punto di arrivo della stratificazione: OS crea ***l'illusione*** ad ogni utente che sta utilizzando una ***macchina dedicata***

Come si virtualizza…

Le risorse della macchina fisica vengono condivise e gestite in modo da creare macchine virtuali

ESEMPI

- Il time sharing della CPU crea l'illusione che ogni utente abbia la sua propria CPU
- Una tecnica di spooling ed un file system permettono di virtualizzare una stampante
- Le comuni partizioni non sono altro che una virtualizzazione dei dischi

Requisiti di *progettazione* di un OS

Utente	Progettista
– comodo da usare	– facile da progettare
– facile da imparare	– facile da implementare
	– facile da manutenere
	– flessibile
– ***affidabile***	– ***affidabile***
– ***sicuro***	– ***senza errori***
– ***veloce***	– ***efficiente***

Scelte di progetto

Meccanismo
Strumento atto a svolgere un certo compito
Politica
Modo di utilizzo dello strumento

La ***separazione tra meccanismo e politica*** permette flessibilita':
politiche potranno essere modificate senza modificare i meccanismi

ESEMPIO

Meccanismo
Strutture dati per la gestione di piu' processi pronti ad essere eseguiti
Politica
Strategia che decide in ogni momento quale processo deve essere eseguito

Scelte di implementazione

- Assembler o ***linguaggio*** ad alto livello?!
- Linguaggio ad alto livello:
  - piu' veloce da scrivere
  - piu' compatto
  - piu' facile da comprendere e correggere
  - molto maggiore la portabilita'
  - ***perdita in efficienza***

Installazione e partenza

- OS deve essere ***configurato*** per ogni specifica piattaforma di installazione
- Input della configurazione da adottare
- ***Due alternative*** : ricompilazione del kernel o tabelle di selezione di moduli
- ***Booting*** – far partire un computer caricando in memoria il kernel di OS
- ***Bootstrap program*** – codice residente in ROM capace di localizzare il kernel, caricarlo in memoria e avviarne l'esecuzione