

Sistema di protezione

- Un processo potrebbe tentare di modificare il programma o i dati di un altro processo o di parte del S.O. stesso.
- Protezione: politiche (cosa) e meccanismi (come) per controllare l'accesso di processi alle risorse del sistema di elaborazione
- Esempi:
 - l'hardware di indirizzamento della memoria assicura che un processo possa operare solo entro il proprio spazio di indirizzi.
 - l'I/O system impedisce l'accesso diretto ai dispositivi, etc.
- All'hardware é affidato il compito di *rilevazione* di errori, come op code illegali o riferimenti in memoria illegali, possibili effetti di errori di programmazione o di comportamenti deliberatamente intrusivi.
- Tali errori vengono *segnalati* e affidati alla *gestione* al S.O. tramite il meccanismo delle trap.
- In presenza di errore o di violazione della protezione il S.O. provvede a terminare il processo, segnala la terminazione anomala ed effettua un dump della memoria.

Sistema di protezione

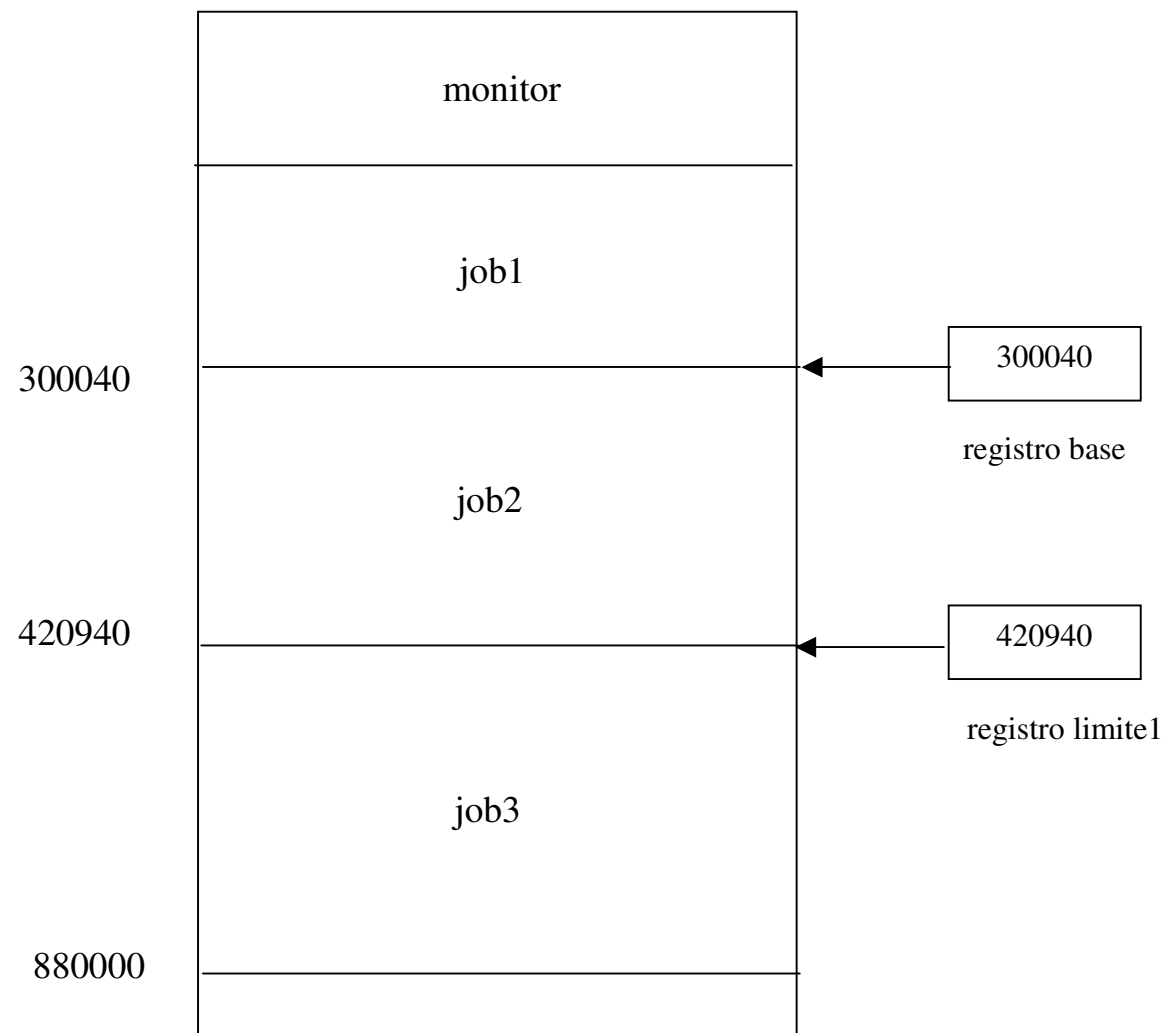
- Ciascun processo opera in un *dominio di protezione* che specifica le risorse a cui il processo può accedere e le operazioni consentite.
- Diritto di accesso (coppia ordinata <risorsa, diritti>): abilitazione alla esecuzione di una operazione sulla risorsa; un dominio di protezione è una collezione di diritti di accesso.
- Lista degli accessi di un oggetto: operazioni consentite da ciascun dominio.
- Lista delle *capabilities*: lista di oggetti e di operazioni consentite su tali oggetti riferita ad un dominio. Per eseguire una operazione su un oggetto il processo deve specificare la capability (indirizzo protetto mantenuto dal S.O.).

Protezione

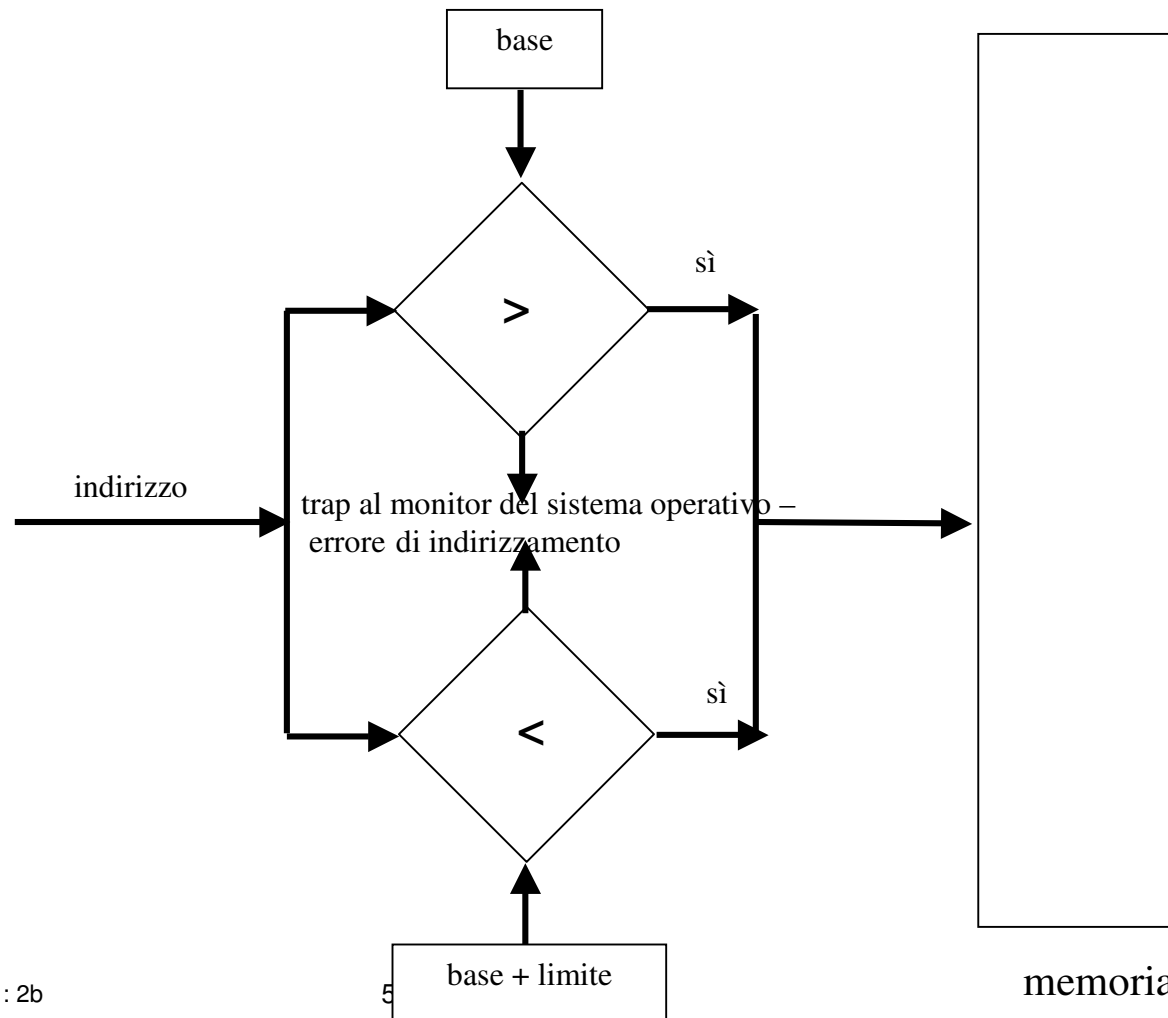
- Il sistema di protezione richiede l'esistenza di *piu` modi* di funzionamento della CPU:
 - *supervisor mode* (system mode, monitor mode)
 - *user mode*
- Il passaggio dal modo user al modo supervisor avviene tramite *interruzione* :
 - esterna (asincrona)
 - interna (sincrona, trap), generata da una SVC (SuperVisor Call o System call).
- Il passaggio dal modo supervisor al modo user avviene tramite una istruzione speciale di *cambiamento di modo* eseguita dal S.O. prima della cessione del controllo ad un processo di utente.
- Istruzioni *privilegiate* (eseguite *solo* in system mode):
 - I/O
 - modifica dei registri che delimitano le partizioni logiche di memoria
 - manipolazione del sistema di interruzione
 - cambiamento di modo
- halt
- Protezione della memoria (registri barriera, registri limite)
- Protezione della CPU (time limit)

Protezione della memoria con registri limite

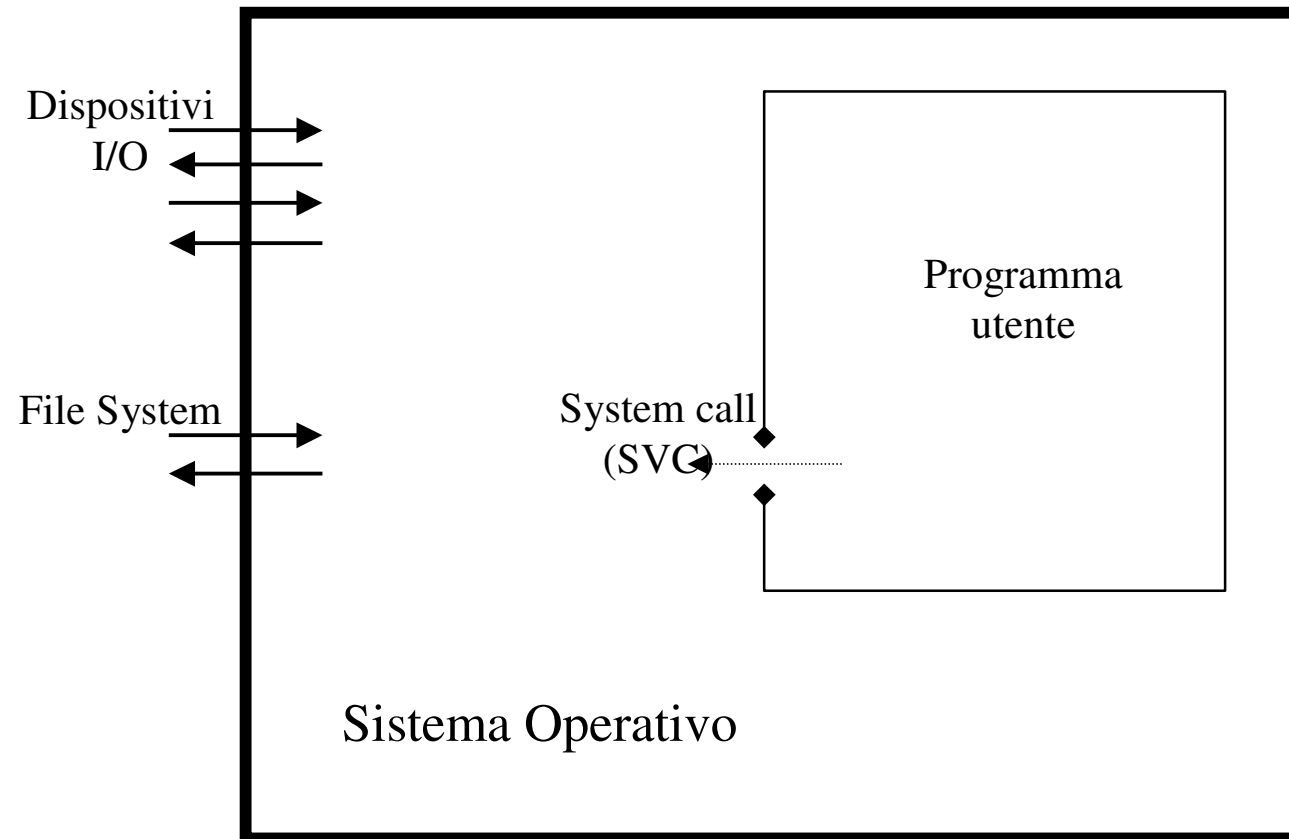
- Prima di mettere un processo in esecuzione, il S.O. ne confina lo spazio logico di memoria mediante registri limite:



Protezione della memoria con registri limite

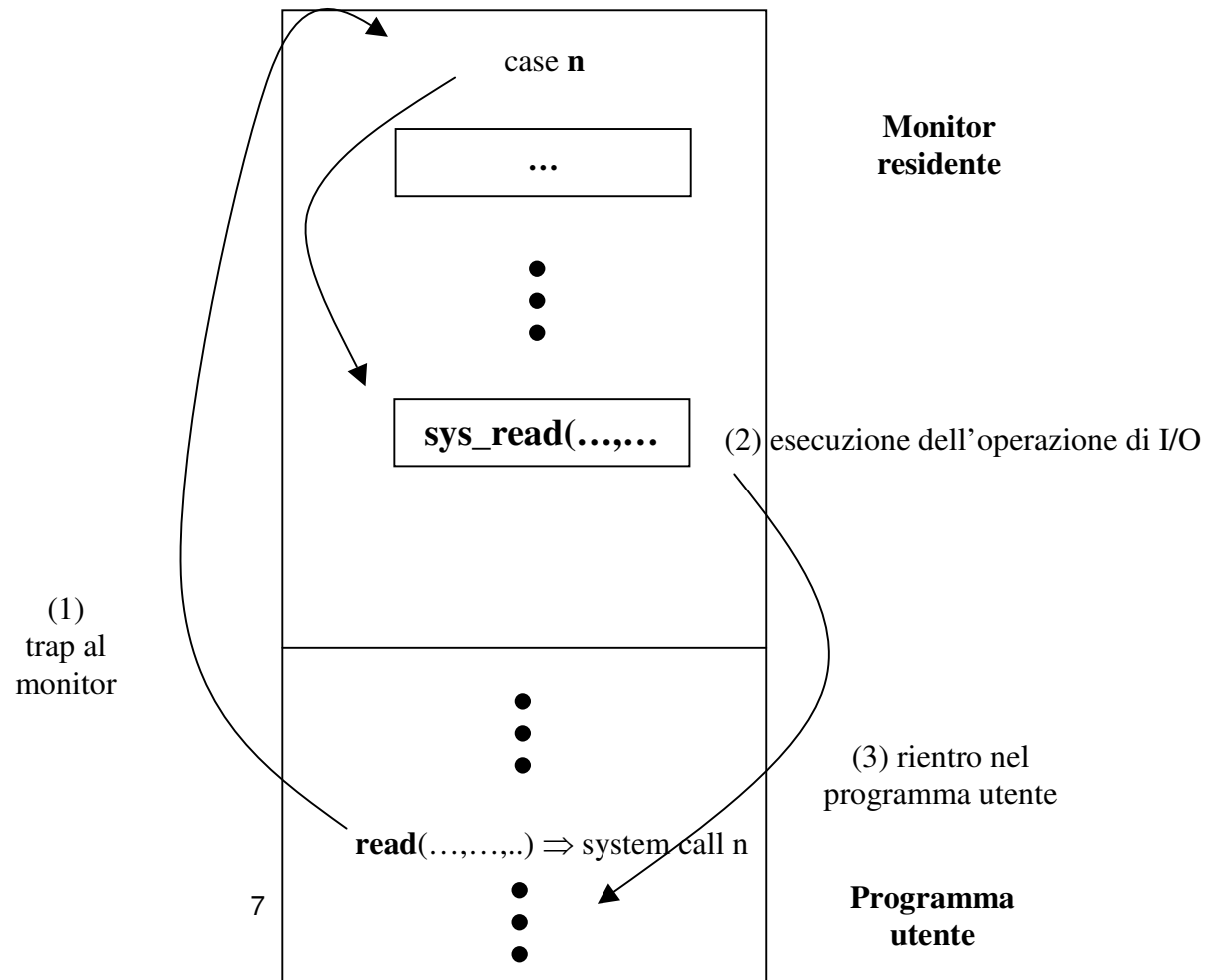


Confinamento del programma utente



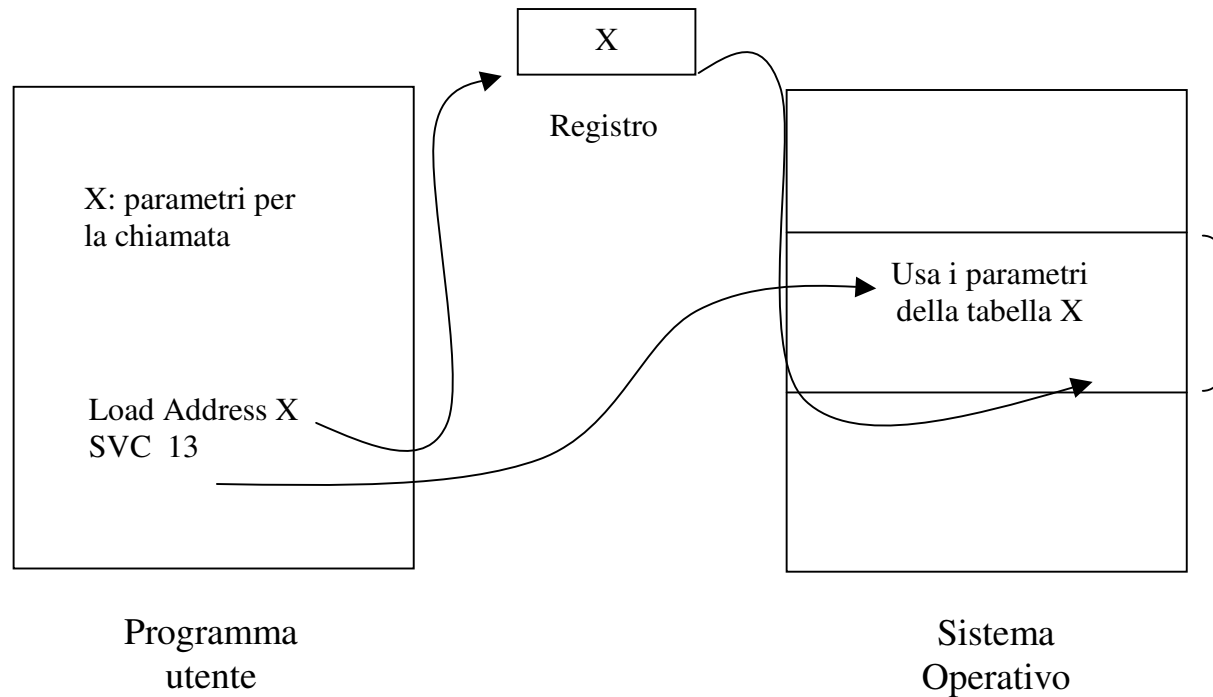
Esecuzione delle operazioni di I/O

- Le istruzioni di I/O sono privilegiate. Il programma utente richiede *al S.O.*, tramite una *system call*, di eseguire l'operazione di I/O.



System call

- Tipicamente l'operando della istruzione di system call ne specifica il tipo (INT n), mentre il passaggio degli eventuali parametri avviene tramite registri o per indirizzo.
- Passaggio dei parametri mediante tabella:



System call

- Costituiscono *l'interfaccia* tra un programma in esecuzione ed il S.O.
- Istruzioni assembly, procedure chiamabili da linguaggi high-level. Nei linguaggi di alto livello sono tipicamente mascherate dal supporto a tempo di esecuzione.
- Categorie principali di system call:
 - a) controllo dei processi e dei job
 - b) manipolazione dei file e dei dispositivi
 - c) gestione delle informazioni
 - d) comunicazione

System call

- a) controllo dei processi e dei job
 - end, abort
 - load, execute
 - create process, terminate process
 - get, set process attributes
 - wait for time, wait for event, signal event
 - allocate, free memory
 - dump, trace
- b) manipolazione dei file e dei dispositivi
 - create, delete file
 - open, close
 - read, write, reposition file or device
 - get, set file or device attributes
 - request, release device
- c) gestione delle informazioni
 - get, set time or date

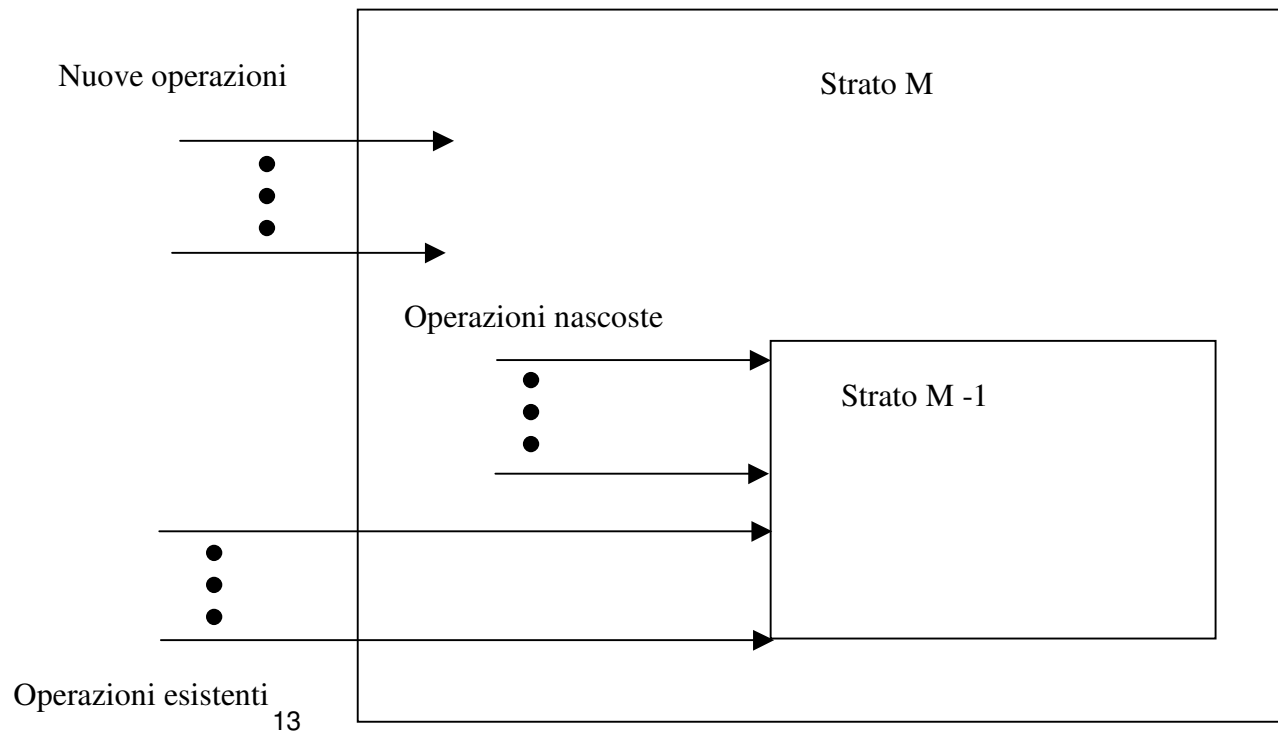
- get, set system data
 - get, set attributes
- d) comunicazione
- create, delete communication connection
 - open, close communication
 - send, receive message

Programmi di sistema

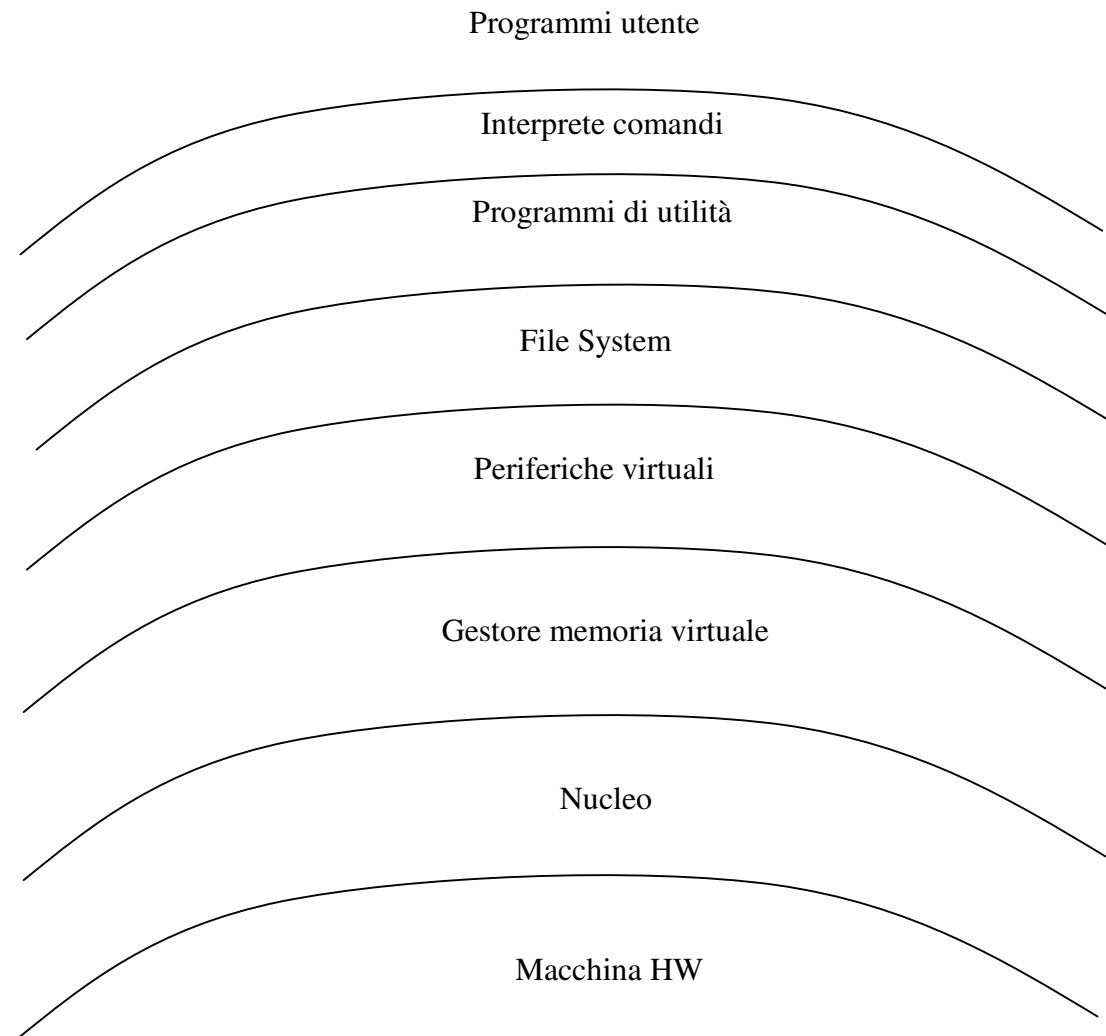
- Di varia natura:
 - manipolazione dei file (editor, cp, mv, rm, mkdir, ...)
 - Informazioni di stato (date, time, who, df, ...)
 - sviluppo software ed esecuzione (traduttori, linker e loaders, debuggers)
 - comunicazione (rlogin, ftp, mail)
 - applicativi (spreadsheet, latex, ...)
 - *interprete dei comandi*: esegue i comandi realizzati come programmi di sistema speciali (unix)
- I programmi di sistema sono determinanti per la *visione d'utente* del S.O., mentre le system call ne riflettono la struttura interna.
- La progettazione della interfaccia con l'utente e` indipendente dalla struttura interna del S.O.

Struttura del S.O.

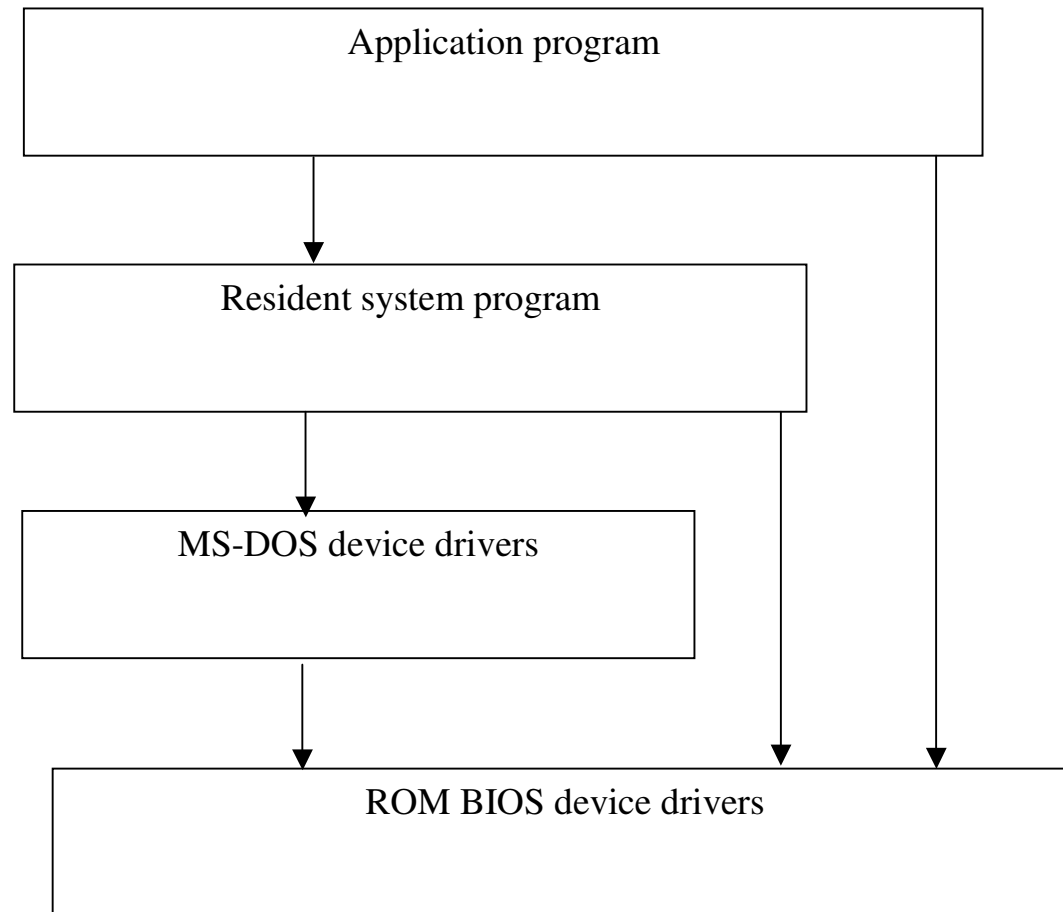
- Necessità di modularizzazione date le dimensioni
- Sistema a *livelli*: il livello piu` interno e` l'hw, quello piu` esterno l'interfaccia di utente
- Affinchè il livello Li possa richiedere i servizi al livello Li-1 deve conoscerne una *specifica* precisa, tuttavia l'*implementazione* di tali servizi deve risultare totalmente nascosta.



La struttura "a cipolla"



La struttura a livelli di MS-DOS

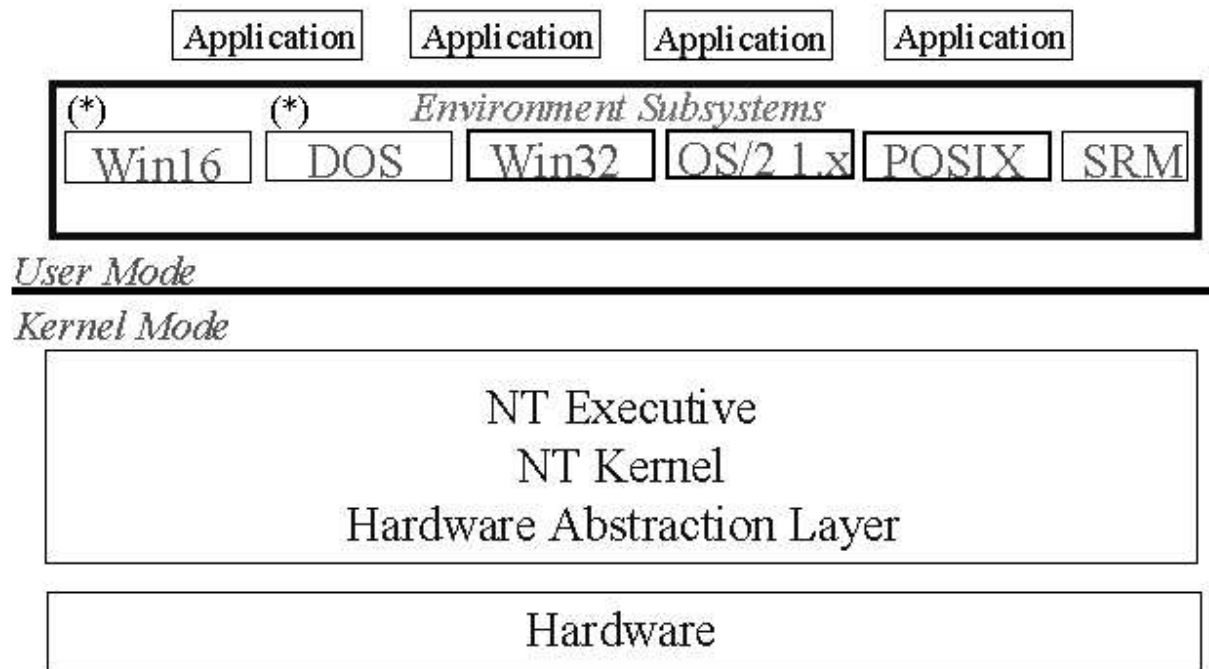


La struttura a livelli di UNIX

(the users)		
Shells and commands Compilers and interpreters System libraries		
<i>System-call interface to the kernel</i> 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		
<i>Kernel interface to the hardware</i>		
Terminal controllers terminals	Device controllers Disk and tapes	memory controllers physical memory

La struttura a livelli di Windows NT

Windows NT Architecture



(*) not an environment subsystem

Struttura del S.O.

- La stratificazione più opportuna può risultare non evidente; è dipendente dall'evoluzione tecnologica dell'hw.
- Sistema a *macchine virtuali* (VM IBM): usando lo scheduling della CPU e la tecnica della memoria virtuale, si possono creare macchine virtuali, una per ogni processo. Si consegue il massimo livello di protezione, a scapito dell'efficienza.
- Realizzazione in linguaggi ad alto livello (UNIX BSD4.3: 3% assembly, il resto in C)
- *Nucleo* o *kernel*: mette a disposizione le system call ai programmi di sistema ed applicativi.

Nucleo di un S.O.

- Fornisce un meccanismo per la creazione e la distruzione dei processi
- Provvede allo scheduling della CPU, alla gestione della memoria e dei dispositivi di I/O
- Fornisce strumenti per la sincronizzazione dei processi
- Fornisce strumenti per la comunicazione tra processi

Struttura gerarchica del Sistema Operativo

- L0: bare machine
- L1: processor management (lower module) / scheduler
- L2: memory management
- L3: processor management (upper module) [messaggi, creazione/distruzione processi]
- L4: device management
- L5: information management