

Sistema di protezione (1)

- 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

Sistema di protezione (2)

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.

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

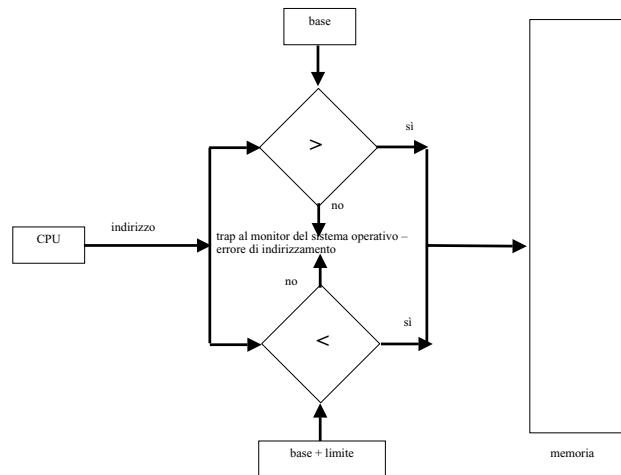
Sistema di protezione (3)

- Questi errori vengono *segnalati* e affidati alla *gestione* del S.O. tramite il meccanismo delle *trap*.
- In presenza di errore o di violazione della protezione il S.O. provvede a:
 - o terminare il processo,
 - o segnalare la terminazione anomala ed
 - o effettuare un *dump* della memoria.

Sistema di protezione (4)

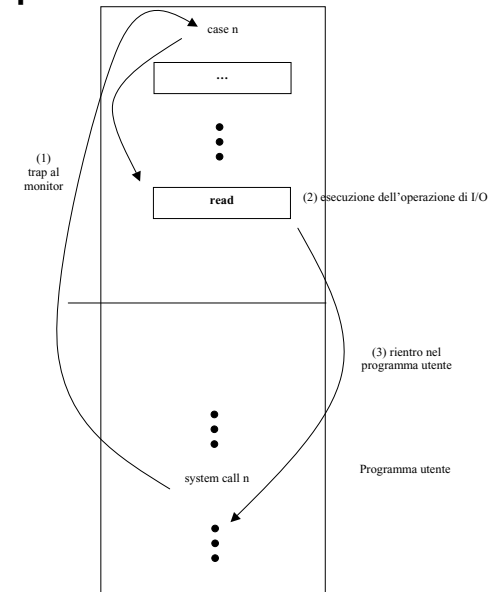
- Ciascun processo opera in un *dominio di protezione* che specifica le risorse a cui il processo può accedere e le operazioni consentite.
- Diritto di accesso (coppia ordinata <risorsa, diritti>): abilitazione all'esecuzione di un'operazione sulla risorsa; un dominio di protezione è una collezione di diritti di accesso.

Protezione della memoria con registri limite

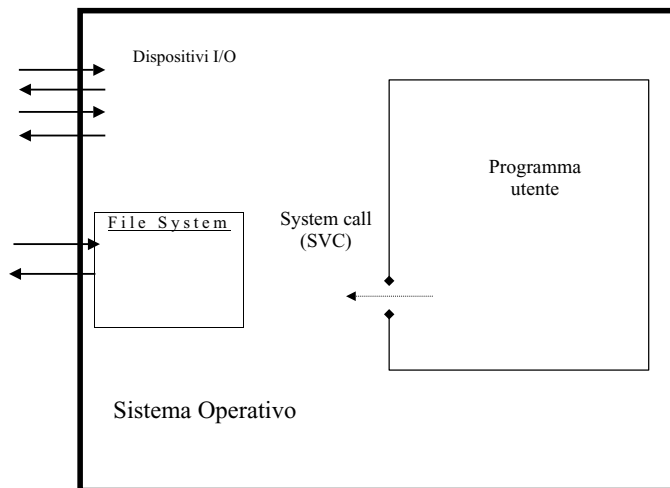


Esecuzione delle operazioni di I/O

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.



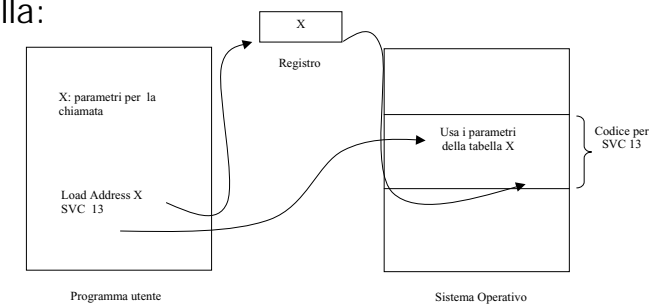
Confinamento del programma utente



System call

Tipicamente l'operando dell'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 (1)

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.

System call (3)

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

System call (2)

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 (4)

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

System call (5)

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

Struttura del S.O.

~~#~~Necessità di modularizzazione, date le dimensioni

~~#~~Sistema a *livelli*: il livello più interno è l'hw, quello più esterno l'interfaccia utente

~~#~~Affinché il livello L_i possa richiedere i servizi al livello L_{i-1} deve conoscerlo precisamente, tuttavia l'*implementazione* di tali servizi deve risultare totalmente nascosta.

Programmi di sistema

Di varia natura:

~~#~~manipolazione dei file (editor, cp, mv, rm, mkdir, ...)

~~#~~Informazioni di stato (date, time, who, df, ...)

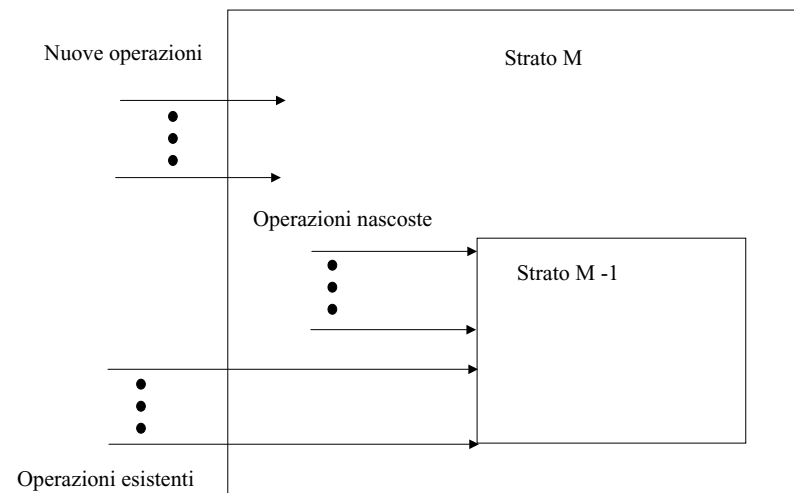
~~#~~sviluppo software ed esecuzione (traduttori, linker e loaders, debuggers)

~~#~~comunicazione (rlogin, ftp, mail)

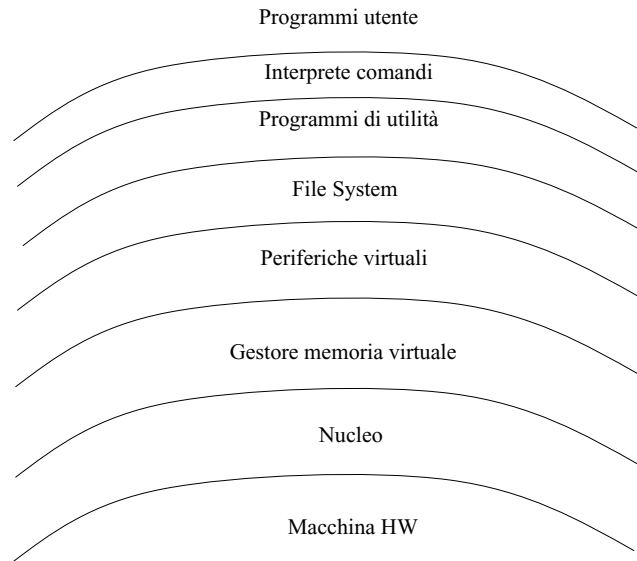
~~#~~applicativi (spreadsheet, latex, ...)

~~#~~*interprete dei comandi*: esegue i comandi realizzati come programmi di sistema speciali (unix)

Struttura del S.O.



La struttura "a cipolla"



Pagina 21

Struttura del S.O. (2)

- 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.

Pagina 23

Struttura del S.O. (1)

La stratificazione più opportuna può risultare non evidente; è dipendente dall'evoluzione tecnologica dell'hw.

Sistema a *macchine virtuali*: usando lo *scheduling* della CPU e la tecnica della memoria virtuale, si possono creare macchine virtuali, una per ogni processo.

- o Si consegue il massimo livello di protezione, a scapito dell'efficienza.

Pagina 22

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

Pagina 24

Struttura gerarchica del S.O.

- 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