

# Protezione

# Protezione

---

- I sistemi operativi svolgono anche compiti di protezione delle risorse hw/sw dei computer.
  - Obiettivi della protezione
  - Dominio di protezione
  - Matrice d'accesso
  - Implementazione della matrice d'accesso
  - Revoca dei diritti d'accesso
  - Sistemi basati su abilitazioni
  - Protezione basata sul linguaggio

# Protezione

---

- Un sistema operativo deve definire meccanismi e politiche per controllare gli accessi a tutte le risorse che esso gestisce.
- Utenti, programmi e processi devono essere controllati nell'accesso a dispositivi, file, programmi, informazioni.
- Le politiche di protezione definiscono cosa controllare.
- I meccanismi di protezione definiscono i modi per farlo.
- Protezione e sicurezza sono due aspetti di uno stesso problema.

# Protezione

---

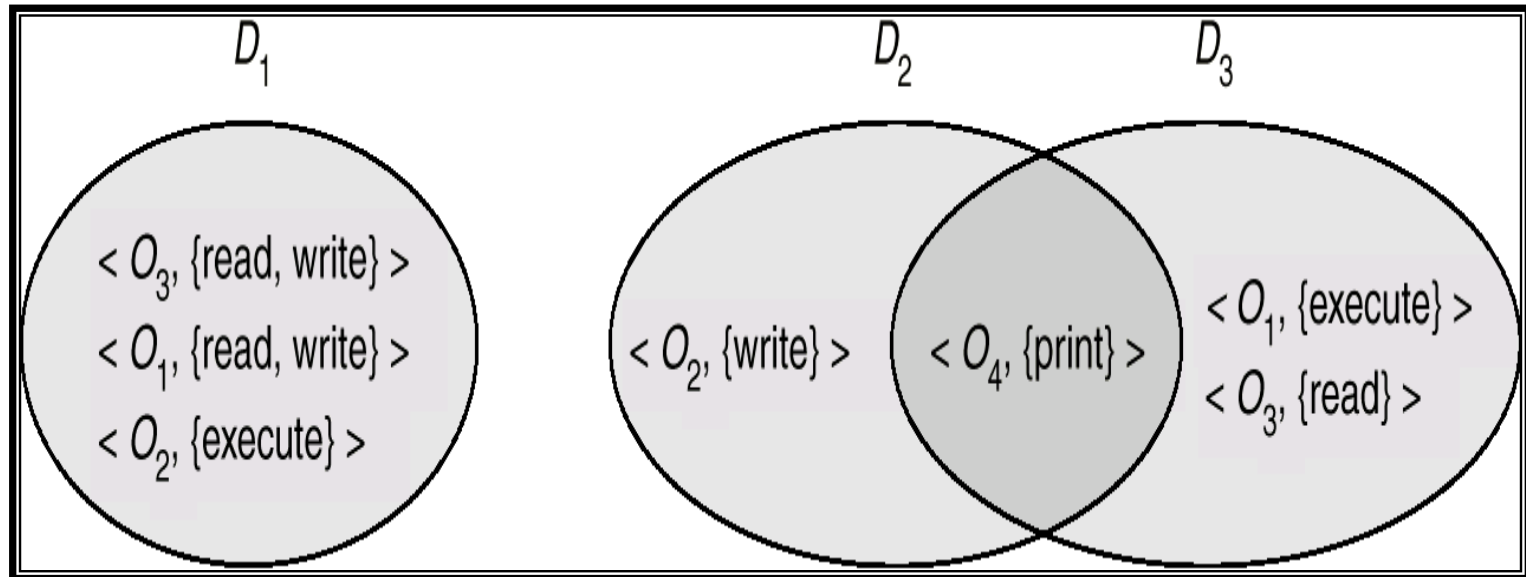
- Un sistema di elaborazione consiste di un insieme di oggetti hardware (CPU, memoria, mouse, ...) o software (file, compilatori, programmi utente, ...).
- Ogni oggetto ha un identificatore unico e può essere acceduto/usato tramite un insieme di operazioni ben definite.
- *Problema della **protezione*** - assicurare che ogni oggetto sia acceduto/usato correttamente e soltanto dai processi che hanno il permesso di farlo.

# Dominio di protezione

---

- **Privilegio minimo:** un processo può accedere solo alle risorse che sono necessarie per svolgere il suo compito.
- **Dominio di protezione  $D$ :** include l'insieme delle risorse (oggetti) a cui un processo può accedere.
- In un dominio per ogni oggetto usato sono indicati:  
***<nome\_oggetto, diritti\_di\_accesso>***
- **Diritti di accesso :** sottoinsieme delle operazioni possibili.
- **Dominio di protezione  $D$  = insieme di oggetti con i diritti di accesso.**

# Dominio di protezione



- Associazione statica o dinamica degli oggetti ai domini.
- Diversi livelli di dominio: utente, applicazione, processo, metodo.

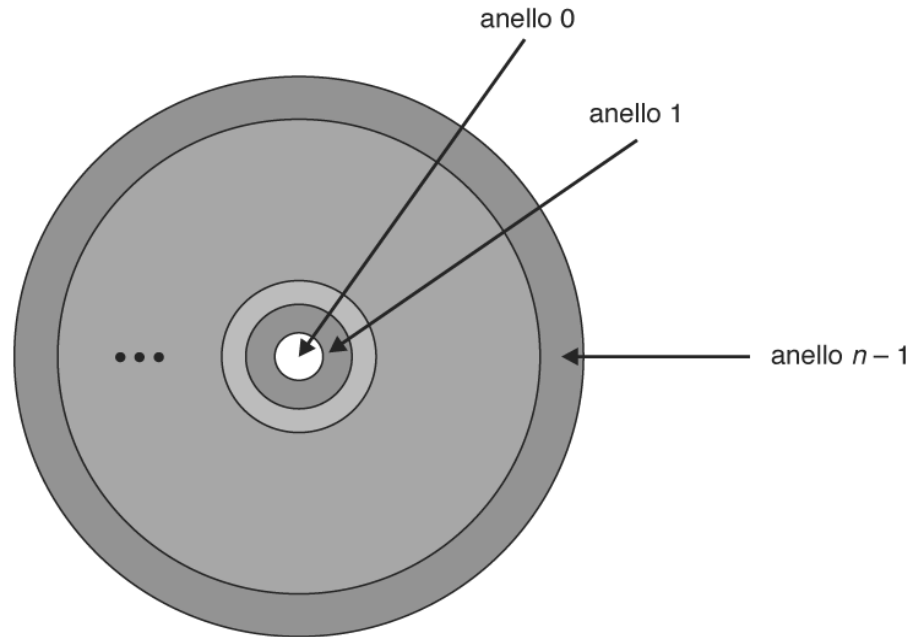
# Esempio di domini in UNIX

---

- Il sistema consiste di due domini:
  - *User*
  - *Supervisor*
  
- UNIX
  - Dominio = *user-id*
  - Nell'accesso ai file il cambio di dominio è svolto dal file system.
    - ▶ Ad ogni file è associato un bit di dominio (*setuid bit*).
    - ▶ Quando un file viene acceduto
      - se il *setuid* = *on*, allora lo *user-id* è modificato assegnandolo al valore del proprietario del file. Al termine delle operazioni lo *user-id* viene riportato al valore precedente.
      - Se *setuid* = *off*, lo *user-id* non può essere modificato.
  - Questo meccanismo serve a permettere ai programmi utente operazioni di sistema (se sono autorizzati a farlo)

# Esempio di domini in Multics

- I domini sono gerarchicamente organizzati ad anelli (da 0 a 7).
- Siano  $D_i$  e  $D_j$  due anelli di dominio.
- Se  $j < i \Rightarrow D_i \subseteq D_j$ : gli anelli più interni hanno maggiori privilegi.
  - Es.:  $D_3 \subseteq D_2$  (*insieme di oggetti di  $D_3$  è un sottoinsieme di oggetti di  $D_2$* ).



Anelli in Multics



# Esempio di domini in Multics

---

- Il cambio di dominio in Multics avviene passando da un anello ad un altro: ad esempio nell'invocazione di un processo o di una procedura di un livello diverso.
- La protezione ad anelli di Multics non implementa la politica del privilegio minimo.
- Il meccanismo di protezione di Multics è complesso e non molto efficiente.
- Se gli anelli sono solo due è simile al modello di UNIX.

# Matrice d'accesso

---

- In questo modello la protezione viene realizzata tramite una matrice detta **matrice d'accesso**.
- Le **righe** della matrice rappresentano i **domini**.
- Le **colonne** della matrice rappresentano gli **oggetti**.
- Un elemento **access( $i, j$ )** è l'insieme delle operazioni che il processo che esegue nel dominio  $D_i$  può eseguire sull'oggetto  $O_j$ .

# Matrice d'accesso

oggetto dominio	$F_1$	$F_2$	$F_3$	stampante
$D_1$	read		read	
$D_2$				print
$D_3$		read	execute	
$D_4$	read write		read write	

Se un processo in  **$D_i$**  tenta di operare su un oggetto  **$O_j$** , l'operazione, per poter essere eseguita, deve essere presente nel dominio.

# Uso della matrice d'accesso

---

- L'uso della matrice d'accesso permette di separare i meccanismi di protezione dalle politiche di protezione.
  - Meccanismi
    - ▶ Il sistema operativo fornisce matrice di accesso + regole.
    - ▶ Deve assicurare che i diritti di accesso specificati nella matrice di accesso vengano osservati.
  - Politiche
    - ▶ Gli utenti e il supervisore definiscono le politiche.
    - ▶ Quando un oggetto viene creato o diviene attivo possono essere definite gli opportuni diritti di accesso.

# Uso della matrice d'accesso

---

- Può essere estesa per una protezione dinamica.
- Definizione dei passaggi da un dominio all'altro (*switch*).
- Operazioni sulla matrice di accesso per aggiungere e cancellare diritti di accesso.
- Diritti per operazioni speciali sulla matrice di accesso:
  - ▶ *owner* di  $O_i$
  - ▶ *copy* operazione da  $D_i$  a  $D_j$
  - ▶ *control* –  $D_i$  può modificare i diritti di accesso di  $D_j$
  - ▶ *switch* – cambio di domino da  $D_i$  a  $D_j$

# Matrice d'accesso con domini come oggetti

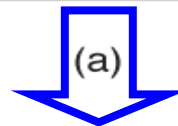
## Matrice d'Accesso Estesa

oggetto \ dominio	$F_1$	$F_2$	$F_3$	stampante	$D_1$	$D_2$	$D_3$	$D_4$
$D_1$	read		read			switch		
$D_2$				print			switch	switch
$D_3$		read	execute					
$D_4$	read write		read write		switch			

- Dal dominio  **$D_1$**  si può passare al dominio  **$D_2$** .
- Dal dominio  **$D_2$**  si può passare ai domini  **$D_3$**  e  **$D_4$** .
- Dal dominio  **$D_4$**  si può passare al dominio  **$D_1$** .

# Matrice d'accesso con diritti Copy

oggetto \ dominio	$F_1$	$F_2$	$F_3$
$D_1$	execute		write*
$D_2$	execute	read*	execute
$D_3$	execute		



oggetto \ dominio	$F_1$	$F_2$	$F_3$
$D_1$	execute		write*
$D_2$	execute	read*	execute
$D_3$	execute	read	

(b)

# Matrice d'accesso con diritti Owner

oggetto \ dominio	$F_1$	$F_2$	$F_3$
$D_1$	owner execute		write
$D_2$		read* owner	read* owner write
$D_3$	execute		



oggetto \ dominio	$F_1$	$F_2$	$F_3$
$D_1$	owner execute		
$D_2$		owner read* write*	read* owner write
$D_3$		write	write

(b)



# Matrice d'accesso con diritti *Control*

oggetto \ dominio	$F_1$	$F_2$	$F_3$	stampante	$D_1$	$D_2$	$D_3$	$D_4$
$D_1$	read		read			switch		
$D_2$				print			switch	switch control
$D_3$		read	execute					
$D_4$	write		write		switch			

Es.: Dal dominio  **$D_2$**  si possono modificare i diritti del dominio  **$D_4$** .

# Implementazione della matrice d'accesso

---

- **La matrice di accesso è sparsa** (contiene molti elementi vuoti) quindi usare una tabella globale è inefficiente.
- Metodi più efficienti della tabella globale:
- Ogni colonna è una **lista di accesso per oggetti**
  - Definisce i domini che possono usare gli oggetti  
Esempio: per  $F1$   
 $D1 = Read$   
 $D4 = Write$
- Ogni riga è una **lista di abilitazioni per domini**
  - Per ogni dominio sono elencati gli oggetti con i diritti  
Esempio: per  $D3$   
 $F2 - Read$   
 $F3 - Execute$

# Implementazione della matrice d'accesso

- Un terzo modo di implementare la matrice di accesso è basato sul **meccanismo lock-key**
- Il **meccanismo lock-key** (serratura-chiave) è un compromesso tra le due liste viste finora:
  - ogni oggetto ha una lista unica di bit detta **lock**,
  - mentre ogni dominio ha un'altra lista detta **key**.
- Un **processo** in esecuzione in un dominio **può accedere a un oggetto** soltanto **se la sua key è in grado di aprire il lock** interessato.
- Questo meccanismo viene gestito dal sistema operativo, gli utenti non possono modificare le key.



# Revoca dei diritti d'accesso

---

Se si usa :

- ***Lista d'accesso*** – Si cancellano i diritti di accesso dalla lista.
  - Semplice
  - Immediato
  
- ***Lista di abilitazioni*** – Occorre trovare le abilitazioni che sono distribuite sui domini prima di poterli revocare.
  - Riacquisizione
  - Puntatori all'indietro
  - Indirizione
  - Chiavi

# Esempio di domini in Windows NT

---

- Per gestire i diritti di accesso globali di un gruppo di utenti in una rete di computer, Windows NT fa uso del concetto di dominio.
- Un dominio di rete definisce un raggruppamento logico di server di rete e altri computer che condividono informazioni comuni sulla protezione e sugli account per ogni utente.
- All'interno dei domini, gli amministratori creano un account per ogni utente. Gli utenti si collegano poi al dominio e non ai server singoli del dominio.
- Esiste un database directory (SAM), che memorizza tutte le informazioni di sicurezza e di account utente per un dominio.
- Quando un utente si collega a un dominio, Windows NT Server controlla il nome e la password dell'utente confrontandola con quella memorizzata nel database di directory.

# Sistemi basati su abilitazioni

---

- **Hydra** (microkernel per SO non più usato)
  - Insieme fissato di diritti di accesso conosciuti e interpretati dal sistema.
  - Definizione di diritti user-defined (tramite capability) per essere usati da programmi utente; il sistema fornisce la protezione di accesso nell'uso di questi diritti.
  - Amplificazione dei diritti (di procedure di sistema).
  
- **Cambridge CAP System** (SO degli anni '70)
  - *Abilitazioni sui dati* - fornisce diritti standard *read*, *write*, *execute* di segmenti di memoria associati agli oggetti.
  - *Abilitazioni software* - interpretazione effettuata tramite procedure protette.

# Protezione basata sul linguaggio

---

- La specifica della protezione in un linguaggio di programmazione permette la descrizione di alto livello di accesso ed uso di risorse.
- L'implementazione di un linguaggio permette di realizzare protezioni software quando altri meccanismi non sono disponibili.
- Si possono generare chiamate a meccanismi di protezione di basso livello (s.o. oppure hardware) quando queste sono disponibili.

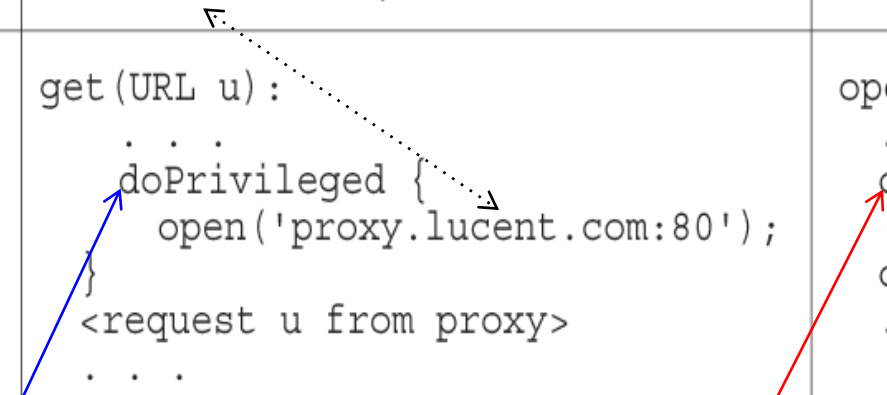
# Protezione in Java

---

- In Java la **protezione è gestita** dalla Java Virtual Machine (JVM).
- **Ad ogni classe è assegnato un dominio di protezione** caricato dalla JVM sulla base dell'affidabilità dell'URL dal quale la classe è caricata.
- Il dominio di protezione indica quali operazioni una classe può (e non può) eseguire.
- Se un metodo viene invocato per eseguire una operazione privilegiata, lo stack viene ispezionato per verificare che l'operazione possa essere eseguita dal metodo.



# Ispezione dello stack

dominio di protezione:	<i>applet</i> non fidata	caricatore di URL	interconnessione
permesso della socket:	nessuno	*.lucent.com:80, connect	qualsiasi
classe:	<pre>gui: . . . get(url); open(addr); . . .</pre>	<pre>get(URL u): . . . doPrivileged {     open('proxy.lucent.com:80'); } &lt;request u from proxy&gt; . . .</pre> 	<pre>open(Addr a): . . . checkPermission (a, connect); connect(a); . . .</pre>

Asserzione del privilegio di accedere ad una risorsa

Controllo del permesso di accesso ad una risorsa