Protezione

Protezione

La protezione riguarda i meccanismi per il **controllo dell'accesso** alle risorse in un sistema di calcolo da parte degli utenti e dei processi.

Meccanismi di imposizione

- fissati in fase di progettazione
- determinati dal gestore del sistema
- definiti dai singoli utenti per proteggere i propri file e programmi

Protezione

- Obiettivi della protezione.
- Domini di protezione.
- Matrice di accesso.
- Implementazione della matrice di accesso.
- Revoca dei diritti di accesso.
- Sistemi basati sulle capacità.
- Protezione basata sul linguaggio.

Obiettivi della protezione

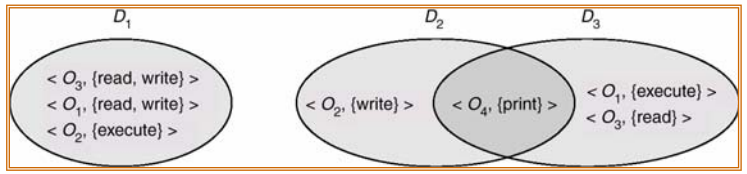
- Un sistema operativo è costituito da un insieme di oggetti, hardware e software.
- Ogni oggetto ha un unico nome, ed è accessibile solo attraverso un insieme ben definito di operazioni.
- Protezione: assicurare che l'accesso ad ogni oggetto sia effettuato correttamente e solo da quei processi che sono autorizzati a farlo.

Principi di Protezione

- Principio guida – principio del minimo privilegio
  - Programmi, utenti e sistemi dovrebbero avere solo i privilegi strettamente necessari per eseguire i loro compiti - Need-to-known-principle
  - Audit trails
  - Ruoli – Modello RBAC

Struttura del dominio di protezione

- **Diritto di accesso** = <*nome-oggetto, insieme diritti*> dove *insieme-diritti* è un insieme di operazioni che possono essere applicate sull'oggetto.
- **Dominio** = insieme di diritti di accesso.



Realizzazione di domini

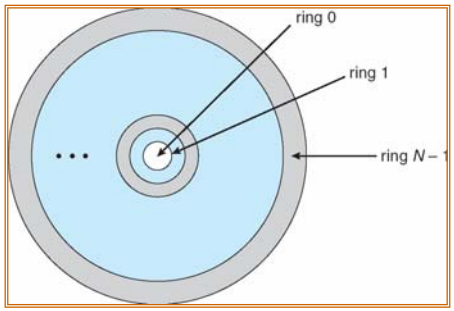
- Ogni **utente** può essere un dominio. L'insieme di oggetti a cui si può accedere dipende dall'identità dell'utente
- Ogni **processo** può essere un dominio. L'insieme di oggetti a cui si può accedere dipende dall'identità del processo
- Ogni **procedura** può essere un dominio. L'insieme di oggetti a cui si può accedere corrisponde alle variabili locali all'interno della procedura.

Implementazione del dominio UNIX

- UNIX
  - Dominio = user-id
  - Il cambio di dominio è effettuato attraverso il file system.
    - Un bit di dominio (i.e., setuid bit) è associato ad ogni file.
    - Quando il file viene eseguito e il bit di dominio è *on*, l'user-id del processo diventa quello del proprietario del file; quando l'esecuzione termina, l'user-id viene resettato.

Implementazione del dominio Multics

- Siano D_i e D_j due anelli del dominio;
- Se $j<i \Rightarrow D_j \subseteq D_i$



Struttura ad anelli di MULTICS

Matrice di accesso

- Il modello di protezione può essere visto in modo astratto come una matrice, chiamata **matrice di accesso**.
- Le righe rappresentano i domini di protezione.
- Le colonne rappresentano gli oggetti.
- L'elemento access(i, j) definisce un gruppo di operazioni che un processo, eseguito nel dominio D_i, può invocare sull'oggetto O_j.

Matrice di accesso

domain \ object	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Figura A

Uso di una matrice di accesso

- La matrice di accesso separa il meccanismo dalle politiche.
  - **Meccanismo**
    - Il sistema operativo fornisce la matrice di accesso e le regole possibili.
    - Assicura che la matrice sia manipolata solo da agenti autorizzati e che le regole d'accesso siano rispettate.
  - **Politica**
    - L' utente decide la politica.
    - Chi può accedere a quali oggetti ed in quale modo.

Uso di una matrice di accesso

- Se un processo nel dominio D_i tenta "op" sull'oggetto O_j, allora "op" deve essere nella matrice d'accesso.
- Permette l'implementazione della protezione dinamica:
  - Cambio di dominio
  - Operazioni per aggiungere e cancellare diritti d'accesso.
    - *copia "op" da O_i a O_j*
    - *proprietario di O_i*
    - *controllo – D_i può modificare i diritti d'accesso di D_j*

Matrice di accesso con domini come oggetti

domain \ object	F_1	F_2	F_3	laser printer	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			

Figura B

Matrice di accesso con diritti di copia

domain \ object	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute		

(a)

domain \ object	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute	read	

(b)

Matrice di accesso con diritti di proprietà

domain \ object	F_1	F_2	F_3
D_1	owner execute		write
D_2		read* owner	read* owner write
D_3	execute		

(a)

domain \ object	F_1	F_2	F_3
D_1	owner execute		write
D_2		owner read* write*	read* owner write
D_3		write	write

(b)

Matrice di accesso modificata dalla Figura B

domain \ object	F_1	F_2	F_3	laser printer	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			

Implementazione di una matrice di accesso

- Colonna = lista di controllo d' accesso per un oggetto.
  Definisce chi può eseguire quale operazione:

  Dominio 1 = Leggere, Scrivere
  Dominio 2 = Leggere
  Dominio 3 = Leggere

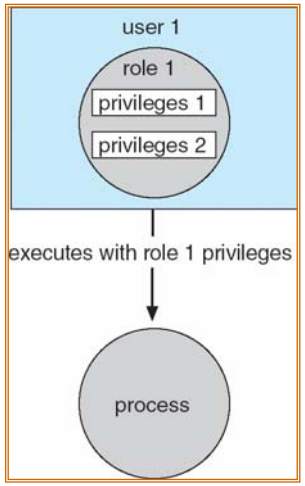
- Riga = una lista di capacità di dominio.
  Per ogni dominio, quali operazioni sono permesse su quali oggetti.

  Oggetto 1 – Leggere
  Oggetto 4 – Leggere, Scrivere, Eseguire
  Oggetto 5 – Leggere, Scrivere, Cancellare, Copiare

Controllo d'accesso

- Protezione può essere applicata a risorse diverse da file
- Solaris 10 fornisce **role-based access control (RBAC)** per implementare il principio del minimo privilegio
  - Privilegio è il diritto di eseguire una system call o di usare un' opzione in una system call
  - Può essere assegnato ai processi
  - Agli utenti vengono assegnati ruoli che garantiscono accesso a privilegi e programmi

RBAC



Revoca dei diritti di accesso

- *Lista di accessi* – Cancella i diritti di accesso dalla lista.
  - Semplice.
  - Immediata.
- *Lista di capacità* – Richiede uno schema per trovare le capacità nel sistema, prima che possano essere rimosse
  - Riacquisizione.
  - Puntatori.
  - Indirezione.
  - Chiavi.

Sistemi basati sulle capacità

- Hydra
  - Un insieme fisso di diritti di accesso, conosciuti e interpretati dal sistema.
  - L'interpretazione dei diritti definiti dall'utente è eseguita dai programmi dell'utente stesso, ma il sistema provvede a una protezione per l'uso di questi diritti.
- Cambridge CAP System
  - Capacità sui dati –fornisce diritti standard di scrittura, lettura ed esecuzione di segmenti associati ad un oggetto.
  - Capacità software – Viene interpretata da una procedura *protetta* (cioè privilegiata), che può essere scritta da un programmatore di applicazioni come parte di un sottosistema.

Protezione basata sul linguaggio

- La specificazione della protezione in un linguaggio di programmazione permette la descrizione ad alto livello delle politiche per l'allocazione e l'uso di risorse.
- Un linguaggio di programmazione può fornire una protezione software quando manca un controllo hardware automatico.
- Può interpretare le specifiche di protezione per generare chiamate di sistema o verso qualsiasi forma di sicurezza che sia fornita dall'hardware e dal sistema operativo.

Protezione in Java 2

- La protezione è gestita dalla Java Virtual Machine (JVM).
- Ad una classe viene assegnato un dominio quando viene caricata dalla Java Virtual Machine (JVM)
- Un file di regole configurabile determina i permessi garantiti al dominio.
- Ispezione dello stack – Più in generale, per accedere a una risorsa protetta, qualche metodo nella sequenza chiamante che ha dato luogo alla richiesta deve esplicitamente dichiarare il suo privilegio ad accedere alla risorsa. Nel fare ciò, questo metodo si *prende la responsabilità* della richiesta; ovviamente, non è permesso a tutti i metodi di dichiarare questo privilegio.

Ispezione dello stack

protection domain:	untrusted applet	URL loader	networking
socket permission:	none	*.lucent.com:80, connect	any

class:	gui:
	...
	get(url);
	open(addr);
	...
		get(URL u):
		...
		doPrivileged {
		open('proxy.lucent.com:80');
		}
		<request u from proxy>
		...
			open(Addr a):
			...
			checkPermission
			(a, connect);
			connect (a);
			...