

www.dis.uniroma1.it/~midlab

Sistemi Operativi II

Corso di Laurea in Ingegneria Informatica

Protezione

Cap. 19 'Operating System Concepts', Silberschatz,
Galvin, Gagne

Sommario

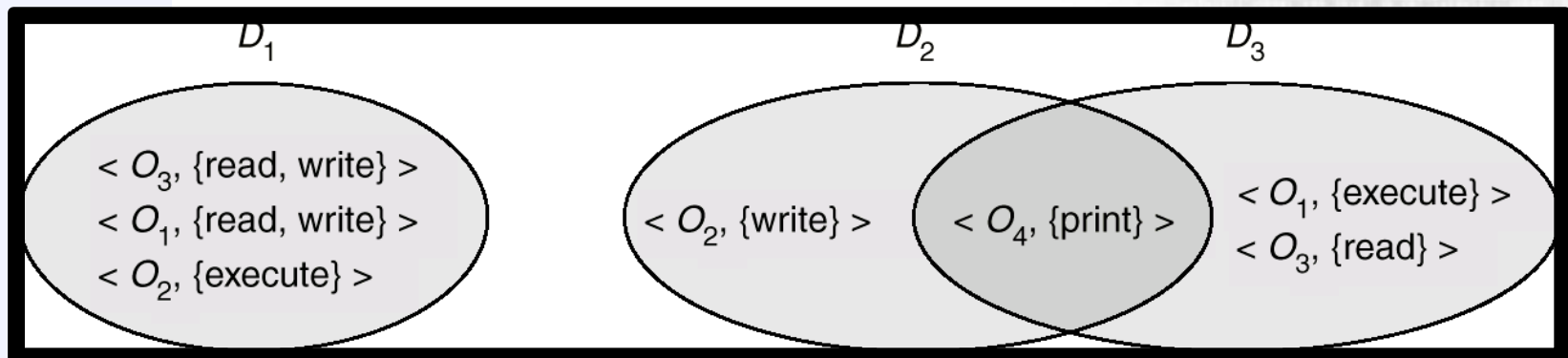
- Gli obiettivi della protezione
- Il dominio di protezione
- La matrice di accesso
- Implementazione della matrice di accesso
- Revoca dei diritti di accesso
- Esercizio
- La protezione in Unix

A cosa serve la protezione

- Un sistema di un calcolatore è un insieme di processi e oggetti (oggetti hardware e software)
- Ogni oggetto ha un nome univoco e vi si può accedere attraverso un insieme ben definito di operazioni
- Lo scopo della protezione è assicurare che ad ogni oggetto si acceda in modo corretto e solo da parte di quei processi che ne hanno facoltà

Struttura dei domini

- Diritto di accesso = $\langle \textit{nome dell'oggetto}, \textit{insieme di diritti} \rangle$ dove *insieme di diritti* è un sottoinsieme di tutte le operazioni valide che possono essere eseguite sull'oggetto
- Dominio = insieme di diritti di accesso
- L'associazione processo-dominio può essere statica o dinamica



Commutazione dinamica di dominio

- Ogni **utente** può essere un dominio L'insieme degli oggetti cui si può accedere dipende dall'identità dell'utente. La commutazione di dominio avviene quando l'utente cambia
- Ogni **processo** può essere un dominio L'insieme di oggetti cui si può accedere dipende dall'identità del processo. La commutazione di dominio avviene tramite l'invio di un messaggio verso un processo da parte di un altro processo che si pone in attesa di una risposta dal primo (paradigma client-server)
- Ogni **procedura** può essere un dominio L'insieme di oggetti cui si può accedere corrisponde alle variabili locali definite all'interno della procedura. La commutazione di dominio avviene quando viene eseguita una chiamata di procedura

Implementazione di dominio in UNIX

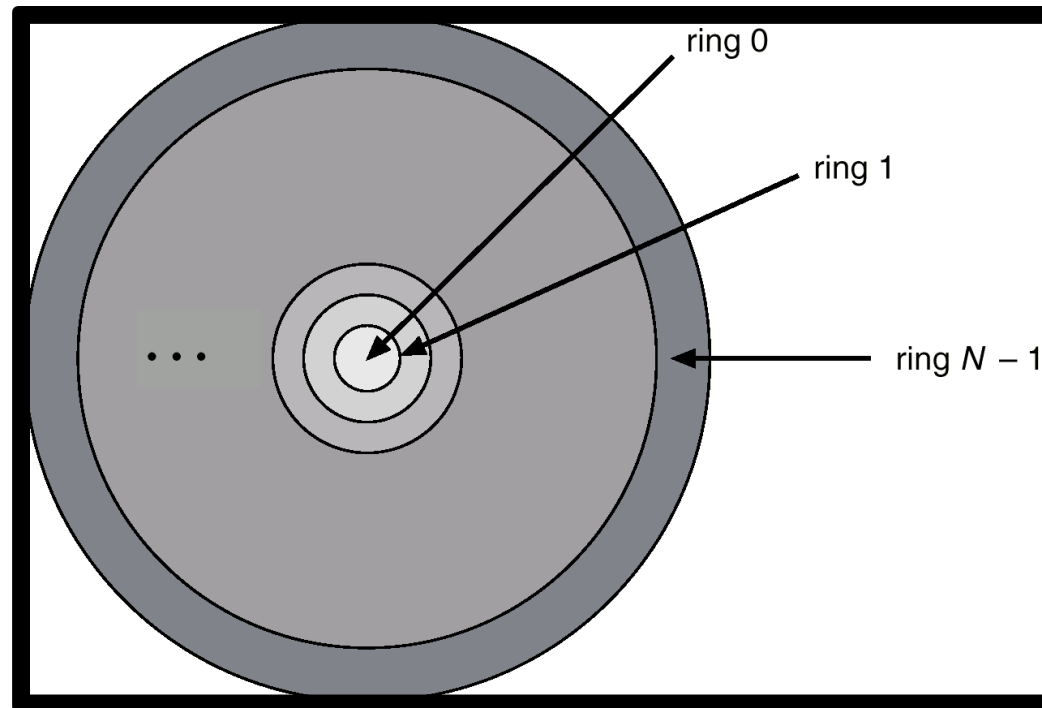
- Il dominio è associato all'utente
- In modalità duale, il sistema consta di 2 domini: utente e supervisore
- In un ambiente multiprogrammato:
 - Dominio di un processo = id-utente
 - Commutazione di dominio via file system
 - ✓ Ad ogni file è associato un id del proprietario e un bit di dominio (bit *setuid*)
 - ✓ Quando un file è eseguito e il suo *setuid* = *on*, l'id-utente del processo diviene quello del proprietario del file. Quando l'esecuzione termina, l'id-utente torna ad essere quello iniziale

Implementazione di dominio in UNIX

- Serie problematiche di sicurezza da considerare, usando *setuid*
- Alternative maggiormente restrittive:
 - Mettere programmi ad uso privilegiato in cartelle speciali di proprietà del superutente
 - I processi mandano richieste ad un processo demone, che viene avviato al boot (sistemi operativi TOPS-20)

Implementazione di dominio in Multics

- Siano D_i e D_j due anelli di dominio qualunque
- Se $j < i$, D_i è un sottoinsieme di D_j



Gli anelli di Multics

Implementazione di dominio in Multics

- Spazio di indirizzamento segmentato; ogni segmento è un file
- Descrizione del segmento
 - ◊ Identificativo del numero di anello
 - ◊ Tre bit di accesso (R/W/E)
- Ogni processo ha un contatore *numero-anello-corrente*
- Il campo anello del descrittore di segmento include i valori: estremi di accesso ($b1, b2$), limite ($b3$), lista dei *gate* (punti di accesso attraverso i quali i segmenti possono essere richiamati)

Matrice di accesso

- La protezione di un sistema può essere descritta come una matrice
- Le righe rappresentano i domini
- Le colonne rappresentano gli oggetti
- $Accesso(i, j)$ è l'insieme delle operazioni che un processo in esecuzione nel Dominio _{i} può invocare sull'Oggetto _{j}

Matrice di accesso

object domain	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 1

Uso della matrice di accesso

- Se un processo nel dominio D_i prova ad eseguire l'operazione op sull'oggetto O_j , allora op deve essere nella matrice d'accesso
- Per estendere alla protezione dinamica, introduciamo le seguenti operazioni volte ad aggiungere o togliere diritti di accesso:
 - ✓ *Proprietario di O_j*
 - ✓ *Copia op da O_i a O_j*
 - ✓ *Controllo: D_i può modificare i diritti di accesso di D_j*
 - ✓ *Trasferimento: commutazione da D_i a D_j*

Uso della matrice di accesso

- Il concetto di matrice d'accesso consente la separazione del meccanismo di protezione dalla politica di protezione:
 - ♦ Meccanismo di protezione
 - ✓ Il sistema operativo fornisce la matrice di accesso e le sue regole
 - ✓ Assicura che la matrice venga manipolata esclusivamente da agenti autorizzati e che le regole siano rigidamente eseguite
 - ♦ Politica di protezione
 - ✓ L'utente stabilisce la politica
 - ✓ Chi può accedere quale oggetto e in che modo

Significato della matrice di accesso

- Ogni colonna rappresenta la lista di controllo di accesso per un oggetto. Definisce chi può eseguire quale operazione. Esempio:
 - Dominio 1 = Read, Write
 - Dominio 2 = Read
 - Dominio 3 = Read
- Ogni riga rappresenta la lista delle facoltà (capability). Per ogni dominio, quali operazioni sono consentite, su quali oggetti. Esempio:
 - Oggetto 1: Read
 - Oggetto 4: Read, Write, Execute
 - Oggetto 5: Read, Write, Delete, Copy

Matrice di accesso di figura 1 avente per oggetti anche domini

object \ domain	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 2

Matrici di accesso con diritti di copia

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

(a)

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

(b)

Matrici di accesso con diritti di proprietà

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

(a)

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

(b)

Matrice di accesso modificata tramite il diritto di controllo

object \ domain	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			

object \ domain	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 della matrice di accesso

1. Global table

- A set of ordered triples $\langle \text{domain}, \text{object}, \text{right-set} \rangle$ whenever an operation M is attempted on O_j within domain D_i . If the triple $\langle D_i, O_j, R_k \rangle$ with M in R_k is found, M is allowed to continue. Otherwise access is denied and an exception occurs
- Additional I/O is needed as the table cannot be usually kept in main memory
- Difficult to take advantages of special grouping of objects or domains (e.g., an object that can be read by everyone)

Implementazione della matrice di accesso

2. Access Lists for Objects

- Each column of the access matrix can be implemented as an access list where each element is a pair $\langle \text{domain}, \text{rights-set} \rangle$. Whenever an operation M is attempted on O_j within domain D_i , we search the access list of O_j . If the pair $\langle D_i, R_k \rangle$ with M in R_k is found, M is allowed to continue. Otherwise access is denied and an exception occurs
- To get better performance, define a default set of access rights for each object and check it first

Implementazione della matrice di accesso

3. Capability Lists for domains

- Each row of the access matrix can be implemented as a capability list of the domain
- A capability list of a domain is a list of objects together with the operations allowed on those objects
- The capability is not directly accessible by a process

Revoca dei diritti di accesso

- *Access List*: delete access rights from access list
 - ✓ Immediate vs. delayed
 - ✓ Selective vs. general
 - ✓ Partial vs. total
 - ✓ Temporary vs. permanent
- Revoking access rights is easy in access-list scheme
- Revoking access rights is difficult in capability-list scheme as capabilities are distributed through the system

Esercizio

Un SO gestisce 5 utenti: u1, u2, u3, u4 e root. Gli utenti u1 ed u2 appartengono al gruppo g1, u3 ed u4 al gruppo g2. Ogni utente è proprietario di un oggetto. Riempire una matrice di accesso usando l'operazione SWITCH, in modo tale che ogni utente di un gruppo possa effettuare operazioni di READ/WRITE/EXECUTE sugli oggetti del gruppo e root possa effettuare operazioni su tutti gli oggetti. Non deve essere possibile effettuare operazioni su oggetti di gruppi diversi e sull'oggetto di root.

Come si leggono i permessi in Unix

```
-rw-r--r-- 1 bea bea    1352  2008-02-01 14:35 appunti_reti_sistemi.txt
-rw-r--r-- 1 root root  70824  2008-01-24 14:45 assistenza_form.pdf
drwxr-xr-x 10 luther blisset 4096  2008-01-24 14:45 dati
lrwxrwxrwx 1 dario sistemi    7  2007-07-19 11:23 aMule -> .aMule/
```

I comandi per gestire i permessi in Unix

chmod [opzioni] permessi file

Esempio:

chmod -R u=rwx,g=rx,o=r /home/bea/dati

che equivale a:

chmod -R 754 /home/bea/dati

chown [opzioni] utente[:gruppo] file

Esempio:

chown -R bea:bea /home/bea/dati

Uso avanzato di chmod ([opzioni])

```
[ugoa...][[+ -=][rwxXstugo...]....][,...]
```

L'operatore «+» aggiunge i permessi specificati a quelli già esistenti di ogni file, «-» li rimuove, «=» li imposta come gli unici permessi del file. Le lettere «rwxXstugo» selezionano i nuovi permessi per gli utenti interessati: leggibile (r), scrivibile (w), eseguibile (o accessibile per le directory) (x), eseguibile solo se il file è una directory o è già eseguibile per alcuni utenti (X); imposta l'user o group ID sull'esecuzione (s), il bit «sticky» (appiccicoso) (t), gli stessi permessi del proprietario del file (u), gli stessi permessi degli altri utenti nel gruppo del file (g), i permessi degli altri utenti che non sono nel gruppo del file (o).

Esempi:

chmod g-s file

rimuove il bit «set-group-ID» (sgid)

chmod ug+s file

imposta sia il bit suid che il bit sgid

Permessi speciali

Sticky (save text image) - Il nome deriva dal significato originale: mantieni il testo del programma sul device di swap. Oggi, se impostato per una directory, significa che solo il proprietario del file o della directory può rimuovere il file dalla detta directory (solitamente usato in directory come /tmp, scrivibili da chiunque).

SGID - attribuisce a chi avvia un file eseguibile i privilegi del gruppo al quale appartiene l'utente proprietario del file. Relativamente alle cartelle, attribuisce la proprietà del file all'utente che lo ha creato in quella cartella e al suo gruppo.

SUID - attribuisce a chi avvia un file eseguibile i privilegi dell'utente proprietario del file. Relativamente alle cartelle, attribuisce la proprietà del file all'utente che lo ha creato in quella cartella.

Espressioni numeriche

Un permesso numerico è dato da una fino a quattro cifre ottali (0-7), ricavate sommando i bit con i valori 4, 2, 1. Le cifre omesse sono considerate zeri iniziali. La prima cifra seleziona gli attributi set user ID (4), set group ID (2) e sticky (1). La seconda cifra seleziona i permessi per il proprietario del file: lettura (4), scrittura (2) ed esecuzione (1); il terzo i permessi per gli altri utenti nel gruppo del file, con gli stessi valori; il quarto per gli utenti non nel gruppo, con gli stessi valori.

Il comando umask

Si usa per negare di default i permessi di accesso ai file appena creati. Può essere definito a livello delle partizioni tramite il file `/etc/fstab` o impostato dal comando `umask` seguito da 3 o 4 cifre ottali.

Esempi:

umask 022 – Il gruppo e gli altri non possono modificare il file

umask 177 – Paranoia!

su e sudo

su – Switch user

sudo – Eseguire il comando come un altro utente.

La configurazione di sudo avviene intervenendo
su `/etc/sudoers`