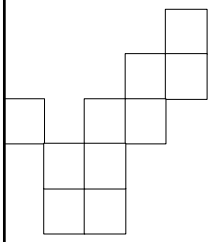


Protezione

Cantiello Lara
Amato Maurizio

Introduzione

multiprogrammazione

⇩

possibilità di condividere tra gli utenti varie risorse

- Processore
- Memoria
- Dispositivi di I/O
- Programmi
- Dati

⇩

necessità di proteggerle

Stallings, cap.15 ; Silberschatz, cap.19

Perchè è necessaria?

- **Prevenire la violazione** intenzionale e dannosa di un vincolo d'accesso
- Assicurare l'**utilizzo** delle risorse del sistema in modo **coerente** alle politiche stabilite

⇩

La protezione può **migliorare l'affidabilità** rivelando errori latenti sulle interfacce tra sottosistemi componenti

Stallings, cap.15 ; Silberschatz, cap.19

Possibili protezioni

- **Nessuna protezione**
  - Appropriata quando devono essere eseguite procedure delicate in momenti separati
- **Isolamento**
  - Ogni processo opera separatamente, senza condivisioni o comunicazioni
- **Condividere tutto o niente**
  - Il proprietario di un oggetto dichiara che è pubblico o privato

Stallings, cap.15 ; Silberschatz, cap.19

Possibili protezioni

- **Condividere mediante limitazioni di accesso**
  - Il SO controlla l'ammissibilità di ogni accesso
- **Condividere mediante capacità dinamiche**
  - Estende il concetto di controllo dell'accesso per consentire la creazione dinamica dei diritti di condivisione dell'oggetto
- **Limitare mediante l'utilizzo di un oggetto**
  - Limita l'accesso e l'uso di un oggetto
  - Es. leggere ma non stampare

Stallings, cap.15 ; Silberschatz, cap.19

Possibili protezioni

- Il SO può fornire gradi differenti di protezione per oggetti, utenti o applicazioni differenti
- Il SO deve bilanciare la necessità di consentire la condivisione con la necessità di proteggere le risorse dei singoli utenti

Stallings, cap.15 ; Silberschatz, cap.19

Alcune definizioni…

- **Protezione:** consiste nel meccanismo per il controllo dell'accesso alle risorse di un sistema da parte di programmi, processi e utenti, e la specifica di controlli e restrizioni da applicare
- **Sicurezza:** impedisce accessi non autorizzati al sistema e tentativi dolosi di manipolazioni dei dati. È la misura della fiducia sul mantenimento dell'integrità del sistema e dei suoi dati.

Alcune definizioni…

- *Def.* La protezione offre un meccanismo di imposizione di politiche che controllino l'utilizzo delle risorse
- *PRINCIPIO IMPORTANTE:* Separazione tra ***politica*** e ***meccanismo***

Alcune definizioni…***politica*** e ***meccanismo***

MECCANISMO = ***come*** deve essere eseguito?

POLITICHE = ***cosa*** deve essere fatto?

Flessibilità:

Cambiamenti delle politiche (legati a tempo e luogo) ⇨ cambiamento nel meccanismo

Preferibili meccanismi generali:

cambiamento delle politiche ⇨ solo modifica di alcuni parametri o tabelle del sistema

Meccanismi di protezione

- Protezione della memoria
- Controllo dell'accesso
  - orientato all'utente
  - orientato ai dati

Protezione della memoria

- **Obbiettivo** = funzionamento corretto dei vari processi che sono attivi
  - Se un processo P_1 scrive inavvertitamente nello spazio di memoria del processo P_2, P_2 potrebbe non essere eseguito correttamente
- **REALIZZAZIONE** = *separazione* o *condivisione* dello spazio di memoria dei vari processi

Protezione della memoria
(Separazione dello spazio di memoria)

- **Schema di memoria virtuale**
  - Segmentazione
  - Paginazione
  - entrambe
- **Isolamento completo**
  - Ogni segmento o pagina è accessibile solo dal processo al quale è assegnato
  - Nessuna entry viene duplicata nella tabella delle pagine e/o dei segmenti

Protezione della memoria
(Condivisione dello spazio di memoria)

Lo stesso segmento o pagina può apparire in più di una tabella

Segmentazione o combinazione di segmentazione e paginazione ⇨ L'applicazione vede la struttura a segmenti e dichiara condivisibili o non condivisibili i singoli segmenti

Paginazione pura ⇨ La struttura della memoria è trasparente all'applicazione, quindi è più difficile discriminare i due tipi di memoria

Protezione della memoria
(Un Esempio: IBM System/370)

- Ad ogni frame è associata una **chiave a 7 bit**
  - 2 bit ⇨ Algoritmo di rimpiazzamento delle pagine
    - La pagina associata al frame è stata usata o modificata
  - 5 bit ⇨ meccanismo di protezione
    - 1 bit = protezione del fetch (recupero della memoria)
      - indica se la chiave si applica solo alle scritture, o sia alle scritture e alle letture
    - 4 bit = chiave di controllo dell'accesso
      - **PSW** (Program Status Word) = contiene informazioni di controllo del processo in esecuzione
      - quando un processo tenta di accedere ad una pagina o di iniziare un'operazione DMA su questa pagina, la chiave PSW corrente viene confrontata con il codice di accesso

Controllo dell'accesso orientato all'utente

- Spesso indicato come *autenticazione*
- ***logon*** (collegamento utente)
  - ***ID*** (Identificatore dell'utente)
  - ***Password***
  - Sistema ID/password inaffidabile
    - Password dimenticata
    - O rivelata a terzi

Controllo dell'accesso orientato all'utente (in ambiente distribuito)

- Centralizzato
  - la rete fornisce il servizio di logon
    - Chi è autorizzato all'utilizzo della rete?
    - A chi si può collegare l'utente?
- Decentralizzato
  - La rete è un mezzo di comunicazione trasparente
  - Logon gestita dall'host di destinazione
  - Problemi di sicurezza per la trasmissione delle password sulla rete
- Due livelli di controllo
  - Servizio di logon fornito dai singoli host
  - Protezione fornita dalla rete per ristringere l'accesso

Controllo dell'accesso orientato ai dati

- Associato ad ogni utente ci può essere un profilo che specifica le operazioni e li accessi ai file consentiti
- Il sistema operativo può introdurre delle regole, basate sul profilo utente
- BD: Il sistema di gestione della base di dati può controllare l'accesso a record specifici o a porzioni di record

Dominio di Protezione

- ***SOGGETTO*** = entità in grado di accedere agli oggetti. In genere un processo.
  - Ogni utente o applicazione ottiene l'accesso ad un oggetto mediante un processo che lo rappresenta
- ***OGGETTO*** = qualsiasi cosa il cui accesso è controllato
  - HW: CPU, segmenti di memoria, stampanti, dischi, unità a nastri
  - SW: file, programmi, semafori

Dominio di Protezione
(Requisiti dei soggetti)

- Ad un processo deve essere permesso di accedere **solo** alle risorse per le quali ha ricevuto **l'autorizzazione**
- Un processo deve essere in grado di accedere **solo alle risorse di cui ha correntemente bisogno** per eseguire il proprio compito *(Privilegio minimo)*

Dominio di Protezione
(Requisiti degli oggetti)

- Ogni oggetto ha un nome unico ed è accessibile solo tramite operazioni ben definite e significative dipendenti dall'oggetto stesso
  - Unità a nastri ⇨ lette, scritte,riavvolte
  - File ⇨ creati, aperti, letti, scritti, chiusi, cancellati

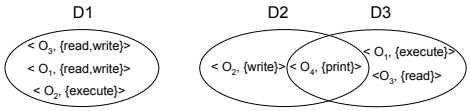
Dominio di Protezione

- ***DOMINIO DI PROTEZIONE*** = specifica le risorse a cui può accedere un processo che opera al suo interno
  - Formato da una coppia ordinata ***<nome oggetto, insieme dei diritti>***
  - Es. <file F, {read,write}>
- ***Diritto di accesso*** = possibilità di eseguire un'operazione su un oggetto
  - Ovvero, il modo in cui un soggetto accede a un oggetto

Dominio = insieme di diritti di accesso

Dominio di Protezione

- I domini possono non essere disgiunti



Associazione processo-dominio

- **Associazione statica**
  - L'insieme delle risorse disponibili per un processo viene fissato per tutta la durata del processo
  - Violazione del privilegio minimo
    - fornisce più diritti di quanti ne siano necessari
  - È **necessario** che il contento del dominio sia modificabile
    - Il dominio dovrebbe riflettere sempre i minimi diritti di accesso necessari

Associazione processo-dominio

- **Associazione dinamica**
  - un processo deve poter passare da un dominio all'altro
  - Permesso di modifica del contenuto di un dominio
    - In alternativa si crea un nuovo dominio con il contenuto modificato

Associazione processo-dominio

- Creazione di un dominio

	Insieme oggetti accessibili	**Cambio di dominio**
Dominio utente	Dipende dall'identità dell'utente	Logout di un utente e login di un altro
Dominio processo	Dipende dall'identità del processo	Invio di un messaggio tra processi e attesa di risposta
Dominio procedura	Variabili locali definite all'interno della procedura	Chiamata di procedura

Matrice d'acceso

- ***Righe*** = domini
- ***Colonne*** = oggetti
- ***Elemento*** = insieme di diritti d'accesso
  - ***access(i, j)***, operazioni richiamabili da un processo in esecuzione nel dominio D_i sull'oggetto O_j

Matrice d'acceso

- Offre un meccanismo per specificare diverse politiche
  - un processo in D_i può accedere **solo** agli oggetti sulla riga i e **solo** nel modo permesso dagli elementi della matrice di accesso

Esempio di Matrice d'acceso

oggetto / dominio	F_1	F_2	F_3	**Stampante laser**
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Matrice d'acceso

- **Associazione statica e dinamica**
  - Passaggio di dominio
    - ***switch*** su uno oggetto (dominio)
  - Modifica contenuto della matrice
    - Poiché ogni elemento della matrice di accesso può essere modificato singolarmente, deve essere considerato come un oggetto da proteggere

Matrice d'acceso (Operazioni possibili)

Matrice di accesso con domini come oggetti:

oggetto / dominio	F_1	F_2	F_3	**Stampante laser**	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			

Il passaggio da D_i a D_j è **permesso** ⟺ Il diritto di accesso **switch ϵ access(i,j)**

Matrice d'acceso (Operazioni possibili)

- **asterisco (*)**
  - possibilità di copiare un diritto d'accesso da un dominio (riga) ad un altro
- ***copy***
  - Permette di copiare il diritto di accesso *solo* all'interno della colonna (oggetto) per la quale il diritto stesso è definito
- ***transfer***
  - Un diritto viene copiato da access(i,j) ad access(k,j) e viene successivamente rimosso da access(i,j)
- ***limited copy***
  - Quando il diritto *R** viene copiato da access(i,j) ad access(k,j), viene creato solo il diritto *R* e non *R**
  - Un processo in D_k non può copiare ulteriormente il diritto *R*

Matrice d'acceso (Operazioni possibili)

Matrici di accesso con diritti *copy:*

oggetto / dominio	F_1
D_1	execute
D_2	execute
D_3	execute

oggetto / dominio	F_2	F_3
D_1	write*	execute
D_2	read*	execute
D_3		execute

F_1	F_2	F_3
execute		write*
execute	read*	execute
execute	read	

Matrice d'acceso (Operazioni possibili)

- ***owner***
  - Permette di aggiungere diritti e rimuoverne altri

owner ϵ access(i,j) ⟺ Un processo in $\mathbf{D_i}$ può aggiungere e rimuovere diritti di elementi sulla **colonna j**

Matrice d'acceso (Operazioni possibili)

Matrici di accesso con diritti *owner:*

oggetto / dominio	F_1
D_1	owner execute
D_2	
D_3	execute

oggetto / dominio	F_2	F_3
D_1	write*	owner execute
D_2	read* owner	read* owner write*
D_3		

F_1	F_2	F_3
		write*
	read* owner write*	read* owner write*
	write	write

Matrice d'acceso (Operazioni possibili)

- ***control***
  - Può essere applicato solo a oggetti di dominio (riga)

control ϵ access(i,j) ⟺ Un processo in $\mathbf{D_i}$ può rimuovere diritti di elementi sulla **riga j**

Matrice d'acceso (Operazioni possibili)

Matrice di accesso con domini come oggetti:

oggetto / dominio	F_1	F_2	F_3	Stampante laser	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			

Matrice d'acceso (Operazioni possibili)

Matrice di accesso con domini come oggetti:

oggetto / dominio	F_1	F_2	F_3	Stampante laser	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	~~read~~ write		~~read~~ write		switch			

Matrice d'acceso (Operazioni possibili)

- ***Problema della reclusione***
  - Garantire che nessuna informazione possa migrare all'esterno del proprio ambiente esecutivo
  - *Copy* e *owner* permettono di limitare la propagazione dei diritti di accesso ma non delle informazioni

Realizzazione della matrice di accesso

- Generalmente la **matrice è sparsa**
- Altre implementazioni:
  - Tabella globale
  - Liste d'accesso per oggetti
  - Liste delle abilitazioni per domini
  - Meccanismo chiave-serratura

Tabella globale

- Insieme di triple ordinate
  - <dominio,oggetto,insieme dei diritti>
- operazione *M* su un oggetto O_j del dominio D_i
  - $<D_i,O_j,R_k>$, dove M ϵ R_k
  - se non viene trovata si lancia un'eccezione
- **Svantaggio**
  - Tabella troppo grande
  - Le ridondanze non possono essere gestite tramite speciali raggruppamenti

Liste di accesso per oggetti

- Associano le colonne della matrice agli oggetti
- Coppie ordinate <dominio,insieme dei diritti> per ogni oggetto
- Definisce tutti i domini per il cui insieme dei diritti d'accesso quell'oggetto è non vuoto
- Un'alternativa:
  - Lista + insieme di default dei diritti di accesso
    - Gli utenti che non hanno diritti speciali possiedono un insieme di diritti per difetto

Liste delle abilitazioni per domini

- *Abilitazione (capability)* = nome fisico o indirizzo che rappresenta un oggetto
  - originariamente proposte come ***puntatori sicuri***
- Associano le righe della matrice ai domini
  - Ma non accessibili direttamente a un processo in esecuzione nei domini
- Lista di oggetti e operazioni ammesse su di essi

Liste delle abilitazioni per domini

- Operazione M sull'O_j
  - M specifica come parametro l'abilitazione (puntatore) per O_j
  - Possesso dell'abilitazione = accesso permesso
- Le abilitazioni si distinguono dagli altri dati per:
  - Un bit che funge da e*tichetta (tag)*
  - Spazio di indirizzi di un programma diviso in due:
    - Dati e istruzioni

Stallings, cap.15 ; Silberschatz, cap.19

Meccanismo chiave-serratura

- ***Serrature***
  - lista di sequenze di bit uniche per ogni oggetto
- ***Chiavi***
  - lista di sequenze di bit uniche per ogni dominio
  - gestita dal SO
- Nessuno degli utenti può modificarle o esaminarle direttamente

Un processo in esecuzione in un dominio può accedere ad un oggetto ⟺ Una chiave del dominio e una serratura dell'oggetto combaciano

Stallings, cap.15 ; Silberschatz, cap.19

Revoca

- In sistemi a protezione dinamica può rendersi necessario revocare permessi
- La revoca può essere:
  - *Immediata o ritardata*
  - *Selettiva o generale*
  - *Parziale o totale*
  - *Temporanea o permanente*

Stingson cap. 15 – Silbershats cap. 19

Revoca *immediata o ritardata*

- **Immediata** quando la revoca necessita un'applicazione istantanea, ovvero, nel momento in cui se ne presenta la necessità
- **Ritardata** quando l'applicazione della revoca non risulta di urgente necessità o dettata da restrizioni o esigente future

Stingson cap. 15 – Silbershats cap. 19

Revoca *selettiva o generale*

- **Selettiva** quando la si vuole applicare ad un singolo utente o ad un gruppo di utenti
- **Generale** quando si presenta la necessità bloccare l'utilizzo di una determinato oggetto

Stingson cap. 15 – Silbershats cap. 19

Revoca *parziale o totale*

- **Parziale** quando si revoca solo un sottoinsieme di diritti di un oggetto
- **Totale** quando si revocano tutti i diritti su di un oggetto

Stingson cap. 15 – Silbershats cap. 19

Revoca t*emporanea o permanente*

- **Temporanea** se occorre negare i diritti su di un oggetto per un determinato intervallo di tempo
- **Permanente** se occorre revocare i diritti su un oggetto a tempo indeterminato

Revoca: applicazione

- Si può avere:
  - Revoca dei **diritti di accesso**
  - Revoca delle **abilitazioni**
- La revoca dei diritti di accesso risulta semplice e veloce
- La revoca delle abilitazioni è complessa ed onerosa

Revoca dei diritti di accesso

- Supponendo uno schema a lista di accesso risulta:
  - Semplice: è necessario rimuovere l'oggetto dalla lista
  - Immediata: alla cancellazione della lista corrisponde l'istantanea eliminazione del diritto
  - Adattabile: si può implementare qualunque tipo di revoca

Revoca delle abilitazioni

- Più difficoltosa, in quanto le abilitazioni sono diffuse in tutto il sistema, e quindi è necessario prima effettuare una ricerca
- Varie possibili soluzioni:
  - Riacquisizione
  - Puntatori indietro
  - Indirezione
  - Chiavi

Revoca: *Reacquisizione*

- Le abilitazioni vengono rimosse periodicamente
- Quando un processo intende utilizzare l'abilitazione deve richiederla
- L'abilitazione può essere revocata
- Alla negazione dell'accesso viene impedita la riacquisizione dell'abilitazione

Revoca: *Puntatori indietro*

- Ogni oggetto conserva un puntatore a tutte le abilitazioni ad esso associate
- Tramite puntatore si risale alle abilitazioni permettendone la modifica
- Adottato dai sistemi MULTICS
- Schema generale ma costoso

Revoca: *Indirezione*

- Le abilitazioni puntano ad una tabella che contiene i puntatori agli oggetti
- Alla revoca viene eliminato il puntatore dalla tabella, in modo da rendere invalido il puntatore dell'abilitazione
- Gli elementi della tabella possono essere riutilizzati
- Sia l'elemento della tabella, sia il puntatore dell'abilitazione contengono il nome unico dell'oggetto
- Adottato dai sistemi CAL
- Non permette la revoca selettiva

Revoca: *Chiavi*

- Una **chiave** è una sequenza unica di bit associata ad un'abilitazione alla sua creazione
- Non può essere né esaminata né modificata dall'oggetto proprietario
- Ad ogni oggetto viene associata una **chiave master**
- La revoca si applica diversificando le due chiavi

Revoca: *Chiavi…meccanismo*

- Alla creazione dell'abilitazione alla *chiave* viene associato il valore della *chiave master*
- Alla richiesta di accesso vengono confrontate le chiavi:
  - $k_A \neq k_M \rightarrow$ viene negato l'accesso
  - $k_A = k_M \rightarrow$ viene concesso l'accesso
- La chiave master può essere cambiata con l'operazione **set-key**

Revoca: *Chiavi…osservazioni*

- La presenza di una sola chiave per oggetto non permette la revoca selettiva
- Permettendo una lista di chiavi per ogni oggetto raggruppate in tabella si può applicare una selezione
- Più chiavi possono essere associate ad un oggetto
- Più oggetti possono condividere la stessa chiave
- La revoca si basa sulla rimozione della chiave dalla lista
- L'accesso si basa sulla presenza della chiave nella lista
- Massima flessibilità

Revoca: *Chiavi…sicurezza*

- La tipologia descritta richiede un moderatore
- Non tutti possono operare (creare, modificare, cancellare) sulle liste
- Solo il proprietario dell'oggetto può operare sulla lista (scelta logica)
- Si riduce ad una scelta di *politica*

Hydra

- Sistema di protezione basato su abilitazioni
- Notevole flessibilità
- Di interesse sono i concetti di:
  - **Insieme dei diritti:** *predefiniti* e *definiti dall'utente* (del sistema di protezione)
  - **Diritti ausiliari**
  - **Amplificazione dei diritti**
  - **Affidabilità** di una procedura

Hydra: Insieme dei diritti

- Il sistema fornisce un insieme di diritti di accesso di base (lettura, scrittura,….)
- Permette all'utente di definire altri diritti:
  - Limitati al programma utente
  - Protetti nell'accesso ad essi
- Per entrambi il sistema fornisce una protezione

Hydra: diritti ausiliari

- Le operazioni sugli oggetti sono definite in modo procedurale
- Le procedure che implementano tali operazioni sono anch'essi oggetti
- L'utente può generare procedure proprie
  - Con il vincolo di notifica al sistema
- Per un oggetto definito dall'utente, le operazioni consentite sull'oggetto diventano *diritti ausiliari*
- I diritti ausiliari possono essere descritti in un'abilitazione per un'istanza del tipo
- Per operare sull'oggetto l'operazione chiamata deve essere presente nell'abilitazione per quell'oggetto

Hydra: amplificazione dei diritti

- È uno schema che permette di dichiarare una procedura affidabile
- L'affidabilità è legata ad un parametro formale di tipo specificato
- I diritti posseduti da una procedura affidabile sono indipendenti e possono superare quelli della procedura chiamante
- L'affidabilità non è universale

Hydra: amplificazione dei diritti…contro

- La procedura che ottiene come parametro un oggetto da un processo che non ha i permessi di modifica su questo
- Il processo chiamato potrebbe ottenere i permessi di modifica tramite amplificazione dei diritti
- Il requisito di protezione può essere aggiurato
- Non è garantito che la procedura non produca errori o eccezioni

Hydra: meccanismo di chiamata

- Hydra con la soluzione delle **call** fornisce una soluzione diretta al *problema dei sottosistemi mutuamente sospetti*
- Il problema consiste nella possibilità che le procedure durante lo svolgimento possano produrre eccezioni danneggiando i dati forniti o non rilasciando le abilitazioni acquisite
- Analogamente, il processo chiamante potrebbe ottenere informazioni riservate delle procedura

Hydra: le *call*

- Per ovviare al problema precedentemente descritto viene creato un sottosistema che interagisce direttamente con il kernel tramite primitive
- Le primitive sono fornite dal kernel per operare sulle risorse definite dal sottosistema
- In questo modo le risorse di sistema vengono utilizzate solo tramite sottosistema e call
- Le politiche di accesso alle risorse possono essere gestite dal creatore del sottosistema, ma non applicate tramite il sistema standard delle applicazioni

Cambrige CAP

- Più semplice ed economico di Hydra, ma non più inefficiente
- Caratteristiche:
  - Microcodice CAP
  - Abilitazione di dati
  - Abilitazione di software
  - Procedure protette

CAP: abilitazione di dati

- Sono i diritti standard forniti dal sistema (lettura, scrittura ed esecuzione) sugli oggetti
- Limitati all'oggetto
- Interpretati dal **microcodice CAP**
- Quest'ultimo può essere visto come il selettore del sottosistema di protezione

CAP: abilitazione software

- È un'abilitazione protetta
- Può essere eseguita solo da procedure *protette*
- Tali procedure:
  - Non vengono processate dal microcodice CAP
  - Sono definite dal programmatore del sottosistema
  - Possono modificare la propria abilitazione software (metodi **seal** e **unseal)**
  - Possono decidere sulle abilitazioni del sottosistema di cui sono responsabili

CAP: i sottosistemi

- Le abilitazioni vengono divise in sottosistemi di cui ogni procedura protetta è responsabile
- Le procedure protette definite dall'utente devono essere verificate dal sistema prima di essere messo in funzione
- Questa gestione permette l'applicazione di una vasta e diversificata serie di politiche

CAP: pro e contro

- PRO
  - Notevole economia nella realizzazione delle politiche
  - Grande possibilità di adattamento alle politiche
- CONTRO
  - Assenza di un manuale utente o interfacciamento (tipo call)
  - La necessità da parte del progettista dei sottosistemi di conoscere a fondo i meccanismi del sistema

Protezione basata sul linguaggio (1)

- In un sistema con abilitazioni dinamiche spesso:
  - l'implementazione di politiche di protezione sono limitate dai meccanismi di supporto presenti
  - gli ambienti di protezione sono più grandi di quanto non sia necessario per aumentare l'efficienza
  - la completa validazione d'accesso richiede un overhead notevole

Protezione basata sul linguaggio (2)

- Per ovviare a questi problemi si è presentata la possibilità di inserire le politiche di protezione dei dati direttamente nel codice di programmazione
- Questo approccio presenta notevoli vantaggi:
  - Le necessità di protezione vengono semplicemente dichiarate
  - I requisiti di protezione possono essere definiti indipendentemente dalle funzioni fornite dal sistema operativo
  - Il progettista di un sottosistema non deve fornire imposizioni
  - I privilegi di accesso sono legati alla dichiarazione del tipo da parte del linguaggio

Protezione basata sul linguaggio (3)

- Il sistema non offre un kernel di protezione potente come Hydra o CAP
- La "sicurezza" offerta è soggetta a molte supposizioni sull'efficienza del sistema
- Necessita, in fase di compilazione, un cernita tra i processi che potenzialmente possono incorrere in violazioni e i processo non nocivi
- Si basa sul presupposto che il codice generato non sia modificato né prima né durante l'esecuzione

Kernel VS Compilatore

- **Sicurezza:** La presenza di un kernel assicura una sicurezza maggiore, in quanto non vincolato dalla bontà del traduttore, della gestione della memoria e dalla sicurezza dei file dai cui viene caricato il programma

Kernel VS Compilatore

- **Flessibilità:** Mentre un sistema con kernel limita l'implementazione delle politiche di sicurezza definite dall'utente, un linguaggio permette sia la dichiarazione che l'implementazione direttamente tramite codice. Se questo non fornisce i mezzi adatti, può essere esteso e/o adattato, cosa non facile se parliamo di un kernel

Kernel VS Compilatore

- **Efficienza:** Una maggior efficienza si ottiene quando l'imposizione della sicurezza è supportata dall'hardware, o dal microcodice. La possibilità del compilatore di verificare off-line l'imposizione d'accesso, evita l'overhead fisso per le chiamate del kernel

Riepilogo

- La protezione basata sul linguaggio:
  - Ci permette la descrizione ad alto livello delle politiche per l'allocazione e l'utilizzo
  - Offre la possibilità di fornire software per la protezione quando non è disponibile hardware
  - Può interpretare le specifiche di protezione ed interagire con qualsiasi sistema di protezione

Soluzioni

- Per rendere efficiente la protezione viene implementato il concetto di *abilitazioni software*
  - Esiste un processo che possiede tale abilitazione, e ha il "potere" di utilizzare le primitve **seal** e **unseal** per bloccare le risorse
  - Un processo che non ha tali abilitazioni, potrà solo copiare o leggere la risorsa, ma non modificarla

Problematiche di distribuzione delle abilitazioni

- La distribuzione ed assegnazione delle abilitazioni deve essere sicure
  - nessun processo può utilizzare una risorsa se non abilitato
- La specifica delle operazioni che un processo può eseguire deve essere finita
  - un processo deve avere solo i permessi che gli sono necessari e l'amplificazione dei propri permessi deve essere concessa solo dal meccanismo di controllo
- Anche la specifica della sequenza di elaborazione deve essere programmata
  - il meccanismo di protezione deve determinare anche la sequenza di operazioni che un processo o più processi posso eseguire sulla stessa risorsa

Conclusioni

- La protezione basata sul linguaggio offre un interessante piano di sviluppo
  - Ottima personalizzazione
  - Distribuzione dei compiti
  - Riduzione dei costi di protezione
  - Predisposizione verso i sistemi distribuiti

Esempi

```
Linux >> la
rwxr--r-x   lib.h
rwxr--r-x   prog.cpp
r-xr-xr-x   a.out
Linux >>
```

- Unix
  - Dominio associato all'utente
  - Cambio di dominio determinato dal cambio utente
  - Ogni file (risorsa) ha un bit di identificazione utente [**User ID**]
  - Ogni file ha una sequenza di bit che identifica i permessi concessi su tale oggetto
  - I permessi sono divisi per dominio e per operazione
    - [ROOT] [USER] [OTHER]
    - [READ] [WRITE] [EXECUTE]

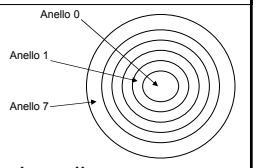
Esempi: Unix

- Quindi n processo potrà utilizzare il file in modo vincolato
  - Il processo potrà usare il file se l'operazione richiesta è presente nel suo dominio
- Solo l'utente *root* (superUser) può eseguire tutte le operazioni

Esempi: problematiche setUID

- Qualora si richiede che una risorsa sia utilizzata da tutti, si presenta la necessità di cambiare l'user-id in root durante l'esecuzione di un processo, permettendo qualsiasi operazione sulla risorsa
- Alternativamente si possono creare processi specifici per la richiesta di risorse comuni inseriti in una cartella con permessi privilegiati, acquisendone questi e avendo accesso alla modifica del user-id
- Soluzione più sicure ed efficiente è quella di creare un processo *daemon* eseguito al boot che deve essere interpellato per l'utilizzo di qualunque risorsa condivisa

Esempi



- MULTICS
  - Domini gerarchici, disposti ad anello
  - 8 possibili anelli concentrici (livelli)
  - Più l'anello è interno, maggiore sono i privilegi
  - Ogni anello ha un numero di identificazione
  - Un file associato ad ogni anello contiene contiene lo spazio degli indirizzi

Esempi: MULTICS

- Ad ogni processo è assegnato l'id dell'anello in cui viene eseguito
  - Tale processo non potrà accedere alle risorse i cui indirizzi appartengono ad un livello inferiore
  - Può accedere ad i livelli superiore, ma vincolato dalla politica associata ad esso

Esempi: MULTICS

- Il cambio dei domini è vincolato da uno stretta politica restrittiva
  - Ad ogni processo dovrà essere associato:
    - *Un intervallo di accesso:* una coppia di interi tali che b1 < b2
    - *Un limite:* un intero b3 > b2
    - *Una lista di ingressi (gates):* identifica i punti di accesso per un livello superiore

Esempi: MULTICS

- Se un processo con id = **i** richiama un processo che ha un id = **j**
  - Se b1 < j < b2 → l'anello di esecuzione rimane **i** ed il processo non viene bloccato
  - Se j < b1 → il processo viene comunque eseguito ma acquisisce indice **j** e i dati utilizzati dal processo dovranno essere copiati nel libello superiore
  - Se j > b2 → l'accesso sarà vincolato e limitato ai gates ed **j** dovrà essere ≤ b3

MULTICS: Svantaggi

- Non consente l'applicazione del principio del *privilegio minimo*
  - Se un oggetto deve essere accessibile dal dominio **j** ma non nel dominio **i**, allora **j < i**
  - In questo modo ogni segmento in **i** è accessibile anche **j**

⇩

NON HO DOMINI DISGIUNTI

Conclusione

- In un sistema di portezione si deve
  - Separare la politica dai meccanismi
  - Sceglie un schema che non penalizzi l'efficienza del sistema
  - Valutare l'impatto del sistema con l'utenza per determinarne l'efficienza