Università di Ferrara
Laurea Triennale in Informatica
A.A. 2021-2022

Sistemi Operativi e Laboratorio

12. Protezione

Prof. Carlo Giannelli

http://www.unife.it/scienze/informatica/insegnamenti/sistemi-operativi-laboratorio

http://docente.unife.it/carlo.giannelli

https://ds.unife.it/people/carlo.giannelli

Protezione

- **Protezione:** Garantire che le **risorse** di un sistema di elaborazione siano accedute solo dai **soggetti** autorizzati.

- **Risorse** fisiche e logiche (fisiche: CPU, memoria, stampanti; logiche: file, semafori, …)

- **Soggetti**: utenti, processi, procedure.

- Servono metodologie, modelli, strumenti per la **specifica dei controlli** e la loro **realizzazione** (enforcement).

Protezione

- **Obiettivo della protezione:**
  assicurare che ciascun componente di programma/processo/utente attivo in un sistema usi le risorse del sistema solo **in modi consistenti** con le **politiche** stabilite per il loro uso.

- **Separazione tra politiche e meccanismi:**
  un sistema di protezione deve essere in grado di realizzare una varietà di politiche.

Protezione o Sicurezza?

Protezione e Sicurezza sono due temi vicini ma diversi.

- La **protezione** serve per prevenire errori o usi scorretti da parte di processi/utenti che operano nel sistema.

- La **sicurezza** serve per difendere un sistema dagli attacchi esterni.

Sicurezza

La sicurezza ha moltissimi aspetti:

Autenticazione/Authentication
(Autorizzazione/Authorisation tema di protezione)

- Riservatezza → Privacy
- Disponibilità → Availability
- Integrità → Integrity
- Paternità → Non-repudiability

Sicurezza

Autenticazione
Verifica dell'identità dell'utente attraverso:

- Possesso di un oggetto (es., smart card)
- Conoscenza di un segreto (password)
- Caratteristica personale fisiologica (impronta digitale, venature retina)

Problema della mutua autenticazione

Si noti che **l'autorizzazione** (protezione) serve per specificare le azioni concesse a ogni utente.

Autenticazione ≠ Autorizzazione

Sicurezza

- **Riservatezza:** previene la lettura non autorizzata delle informazioni (es. messaggi cifrati. Se intercettati, non rivelano comunque il contenuto).

- **Integrità:** previene la modifica non autorizzata delle informazioni (es. un messaggio spedito dal mittente è ricevuto tale e quale dal destinatario).

- **Disponibilità:** garantire in qualunque momento la possibilità di usare le risorse.

- **Paternità:** chi esegue un’azione non può negarne la paternità (per esempio un assegno firmato)

Protezione (e least privilege)

- In qualunque momento un processo (o un utente) può accedere solo agli oggetti per cui è autorizzato.

- Nell'informatica moderna è importante rispettare il principio del "**least privilege**", cioè in ogni istante un processo deve poter accedere **solo a quelle risorse strettamente necessarie** per compiere la sua funzione (in questo modo si limita il danno che un processo con errori può creare nel sistema).

Esempi di **least privilege**:

1. Processo **P** chiama una procedura **A**. **A** deve poter accedere a sue variabili e parametri formali passati, e non a tutte le variabili processo **P**.
2. Un amministratore di sistema che deve fare il backup di tutto un file system, deve avere i diritti di lettura su tutto, non quelli di scrittura.

Protezione e controllo degli accessi

- Nei sistemi operativi ci sono dei componenti incaricati di verificare che i processi possano accedere alle sole risorse per cui sono autorizzati.

- Si parla di **Reference Monitor**, come il componente del sistema operativo che media tra le richieste di accesso dei processi e le risorse.

- TUTTE le richieste di accesso passano dal Reference Monitor.

- È importante che tutte le decisioni di accesso alle risorse siano concentrate in un unico componente.

Dominio di protezione

- Quali soluzioni per gestire il controllo degli accessi e garantire quindi una corretta protezione delle risorse?

- Consideriamo il **dominio di protezione**, che definisce un insieme di risorse (oggetti) e i relativi tipi di operazione (diritti di accesso) sugli oggetti stessi che sono permesse a processi (soggetti) appartenenti a tale dominio.

- Per esempio, un processo opera all'interno di un **dominio di protezione** che specifica le risorse che il processo può usare (e con che diritti di accesso).

- Esempio:

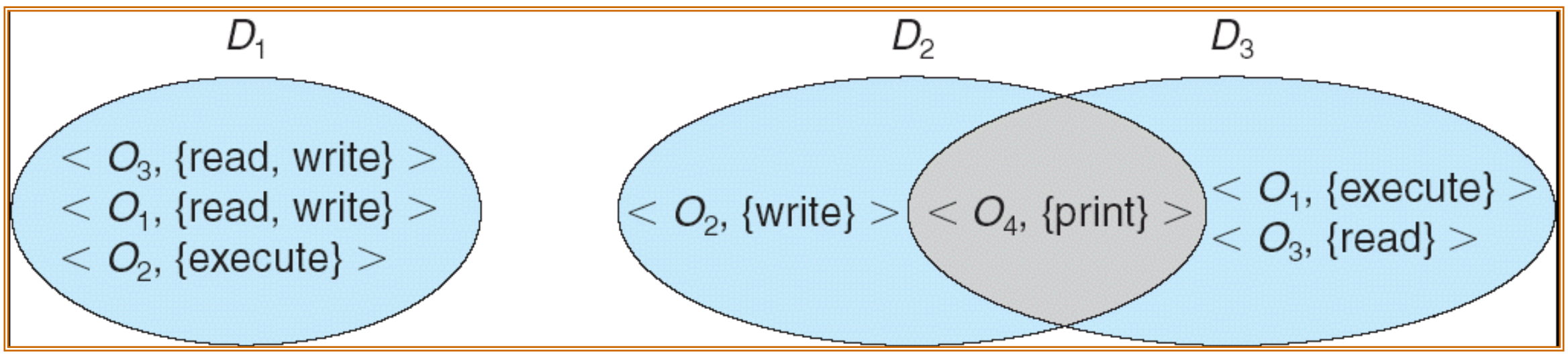
D1	D2	D3
$<O_3,\{read,write\}>$		$<O_4,\{print\}>$
$<O_1,\{read,write\}>$	$<O_2,\{write\}>$	$<O_1,\{execute\}>$
$<O_2,\{execute\}>$		$<O_3,\{read\}>$

Per esempio, un processo che appartiene al dominio D2 può solo scrivere sulla risorsa O_2

Domain Structure

- Access-right = <*object-name*, *rights-set*>
  dove *rights-set* è un sottoinsieme di tutte le operazioni valide che possono essere eseguite sull'oggetto.

- Domain = insieme di access-rights



Dominio di protezione

Il dominio è un concetto astratto che può essere realizzato in una varietà di modi:

- un dominio per ogni **utente**. L'insieme degli oggetti che l'utente può accedere dipende dall'**identità dell'utente**. Il cambio di dominio è **legato all'identità dell'utente** (avviene quando cambia l'utente, es. Unix).
- un dominio per ogni **processo**. Ogni riga descrive gli oggetti e i diritti di accesso per un processo. Il cambio di dominio corrisponde **all'invio di un messaggio** a un altro processo.
- un dominio per ogni **procedura**. Il cambio del dominio corrisponde **alla chiamata di procedura.**

Come si può realizzare un modello basato sui domini di protezione?

Matrice degli accessi (modello di protezione)

oggetto / dominio	F_1	F_2	F_3	disco	stamp.
D_1	read		read		
D_2				read	print
D_3		read	execute		
D_4	read write		read write		

diritto di accesso

- **access(i,j)** definisce l'**insieme dei diritti di accesso** che un processo che opera nel dominio **i** può esercitare sull'oggetto **j**

- Si può realizzare come un **insieme ordinato** di *triple* <*dominio, oggetto, insieme dei diritti*> (tavola globale)

- Quando un'operazione **M** deve essere eseguita nel dominio $\mathbf{D_i}$ su $\mathbf{O_j}$, si cerca la tripla $<\mathbf{D_i,O_j,R_k}>$ con $\mathbf{M \in R_k}$ Se esiste, l'operazione può essere eseguita; diversamente, si ha situazione di **errore**.

Matrice degli accessi (modello di protezione)

oggetto / dominio	F_1	F_2	F_3	disco	**D3**
D_1	read		read		
D_2				read	**switch**
D_3		read	execute		
D_4	read write		read write		

- Domini **statici** o **dinamici**
  - caso statico: l'associazione processo/dominio non può cambiare durante la vita del processo.
  - caso dinamico: c'è un diritto "switch" per permettere a un processo di cambiare dominio di protezione.

- **Problemi** della matrice degli accessi: dimensioni matrice troppo grandi (e sparse).
- Soluzioni meno generali ma più efficienti e diffuse: **access control list**, **capability**

Access Control List (ACL)

- **Per ogni oggetto** viene indicata la coppia ordinata <*dominio, insieme dei diritti*> limitatamente ai domini con un insieme di diritti non vuoto

- Quando **deve essere eseguita** un'operazione **M** su un oggetto $\mathbf{O_j}$ nel dominio $\mathbf{D_i}$, si cerca nella lista degli accessi $< \mathbf{D_i}, \mathbf{R_k} >$ con $\mathbf{M} \in \mathbf{R_k}$

- Se non esiste, si cerca in **una lista di "*default*"**; se non esiste, si ha condizione di **errore**.

- Per motivi di **efficienza**, si può cercare prima nella lista di *default* e successivamente nella lista degli accessi

- Esempio: **file system**, lista degli accessi **associata al file** contiene: *nome utente (il dominio) e diritti di accesso*

Capability List

- Per ogni dominio viene indicato l'**insieme degli oggetti e dei relativi diritti di accesso** (*capability list*)
  D_1: <O_1, diritti>,<O_2, diritti>, etc.
  D_2: <O_2, diritti>,<O_5, diritti>, etc.
  etc.

- Spesso un oggetto è identificato **dal suo nome fisico** o dal **suo indirizzo** (capability). Il possesso della *capability* corrisponde all'autorizzazione a eseguire una certa operazione.

- Quando un processo opera in un dominio, **chiede di esercitare un diritto di accesso** su un oggetto. Se ciò è consentito, il processo **entra in possesso di una capability** per l'oggetto e può eseguire l'operazione.

- La lista delle *capability* non è direttamente **accessibile** a un processo in esecuzione in quel dominio. È **protetta** e **gestita** dal S.O. Non può migrare in qualsiasi spazio direttamente accessibile a un processo utente (non può essere **manipolata** dai processi).