# Lezione 9
# Autorizzazione

Sistemi Operativi (9 CFU), CdL Informatica, A. A. 2022/2023
Dipartimento di Scienze Fisiche, Informatiche e Matematiche
Università di Modena e Reggio Emilia
http://weblab.ing.unimo.it/people/andreolini/didattica/sistemi-operativi

# Quote of the day

(http://spaf.cerias.purdue.edu/firsts.html)

**“The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards.”**

*Eugene Howard Spafford (1956-)*

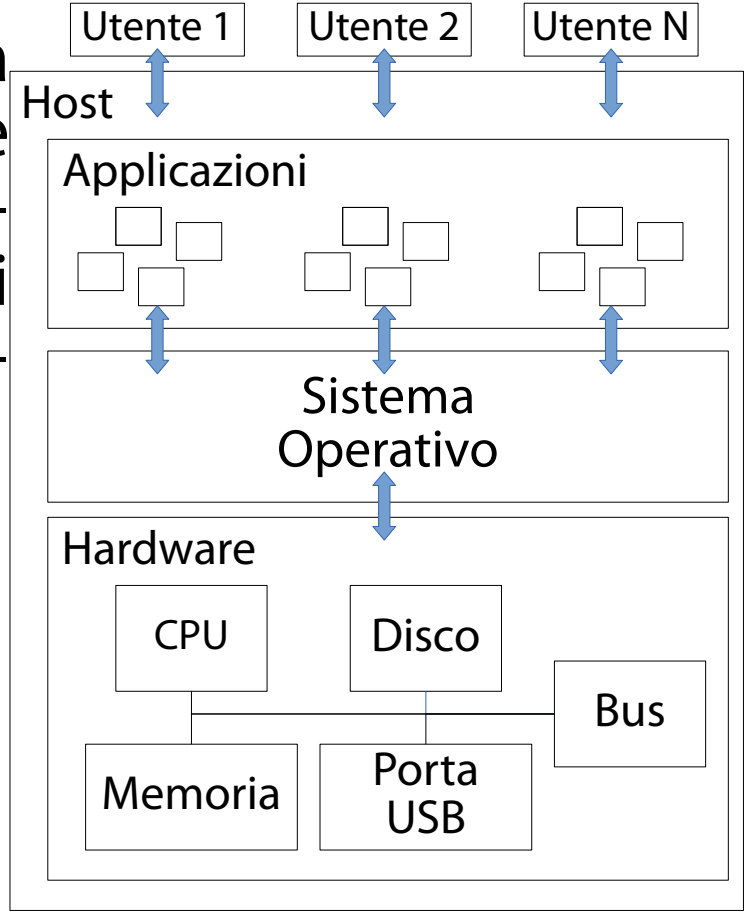
*Docente universitario*
*Pioniere della sicurezza informatica*
*(leggasi il link in alto)*

# INTRODUZIONE

# Lo scenario

(Uno studente vuole capire cosa è e come funziona l'autorizzazione in UNIX)

Uno studente in grado di usare la linea di comando vuole capire cosa è e come funziona la procedura di autorizzazione nei sistemi UNIX-like (GNU/Linux, nello specifico).



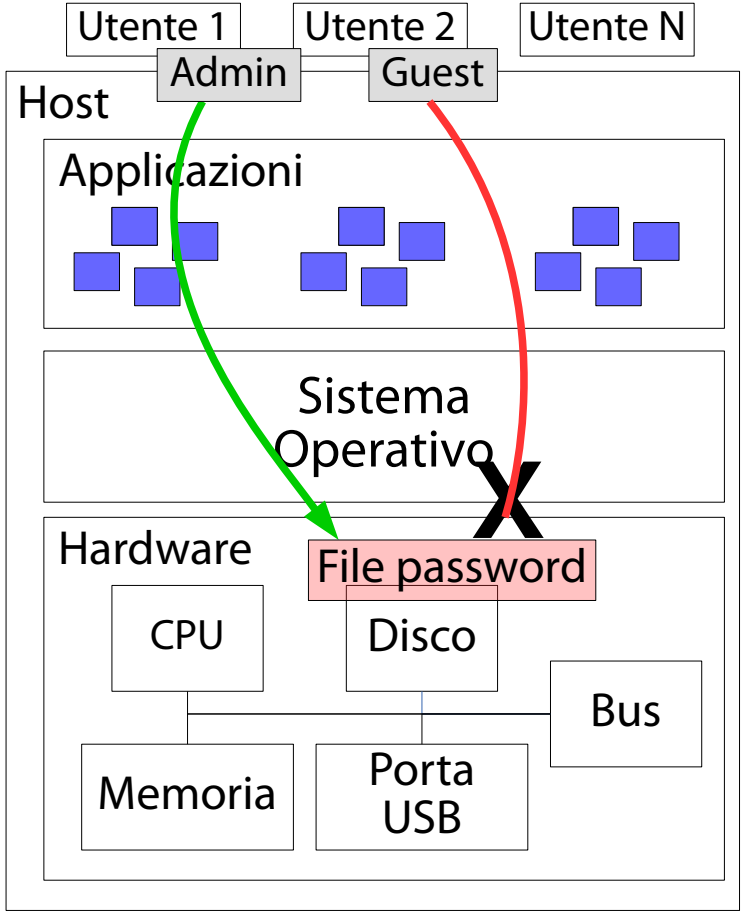
# Interrogativi 1/2

(Come limitare a specifici utenti l'accesso ad una risorsa hw/sw?)

Il SO espone risorse hw/sw agli utenti perché possano fruirne.
Tuttavia, non tutte le risorse hw/sw sono fruibili a tutti gli utenti.

Esempio: file delle password (è modificabile solo da un amministratore).

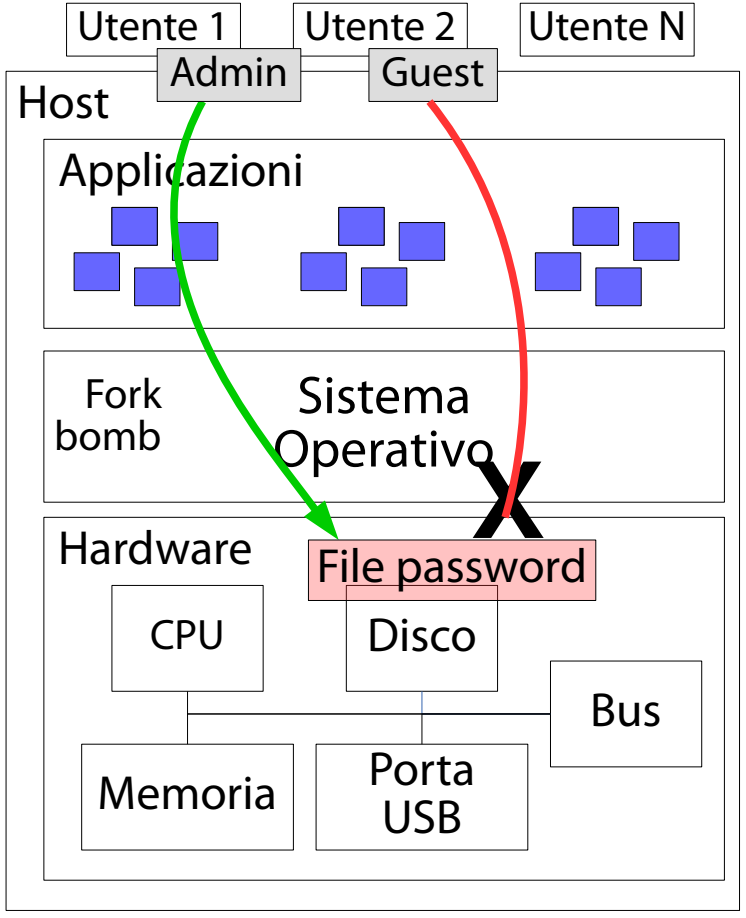
Il SO è in grado di applicare restrizioni di accesso alle risorse?



# Interrogativi 2/2

(Esistono strumenti per la restrizione dell'accesso alle risorse?)

Il SO mette a disposizione gli strumenti per fornire o negare a specifici utenti l'accesso alle risorse?
Come funzionano tali strumenti?



# Autorizzazione

(Permette di stabilire il grado di accesso di un utente ad una risorsa hw/sw)

I SO moderni forniscono astrazioni software per l'autorizzazione.

**Autorizzazione (access control):** è la procedura con cui il SO controlla se una applicazione lanciata da un utente ha i privilegi di accesso sufficienti per accedere ad una risorsa hw/sw.

# Le astrazioni principali in gioco

(Nella procedura di autorizzazione)

Le astrazioni principali in gioco nella procedura di autorizzazione sono le seguenti.

**Gruppo di lavoro.** Specifica un insieme di utenti con privilegi di accesso identici su una risorsa.

**Attributi di un file.** Specificano i privilegi minimi di accesso alla risorsa che gli utenti devono fornire per poter usufruire della stessa.

**Credenziali di applicazione.**

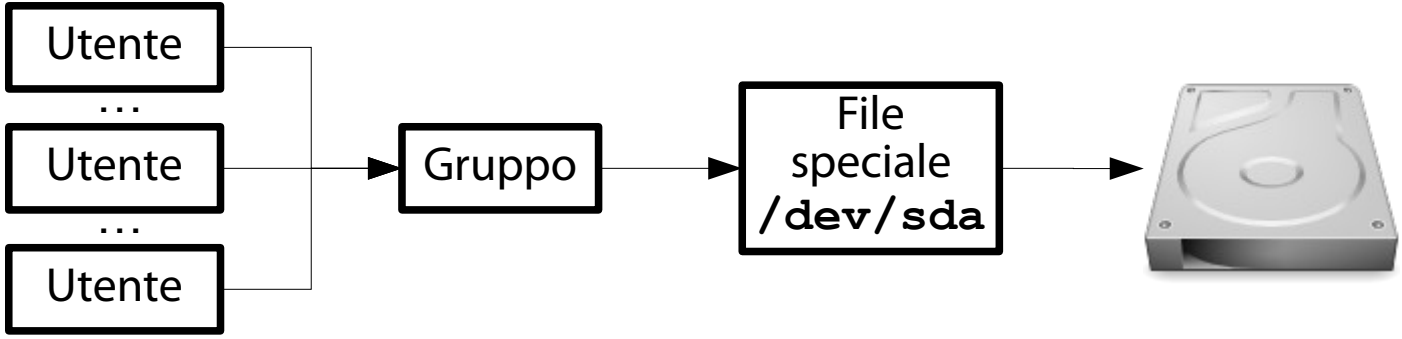
Specificano le credenziali con cui l'applicazione si presenta ad accedere alla risorsa.

# L'astrazione software "gruppo di lavoro"

(Rappresenta un insieme di utenti con privilegi identici su un file)

Il SO fornisce una astrazione software dal nome **gruppo** (**group**). Un gruppo rappresenta un insieme di utenti che condividono specifici privilegi di accesso verso file e directory.

Il gruppo è una delle tante informazioni di un account utente.



# Informazioni associate al gruppo

(Le più importanti)

**Identificatore gruppo (group id)**: un numero intero non negativo univoco per un gruppo. È usato dal nucleo del SO per distinguere gruppi diversi.

**Nome del gruppo (groupname):** stringa alfanumerica di lunghezza non nulla. È una rappresentazione ad alto livello dell'identificatore del gruppo (per l'utente umano).

**Password:** una sequenza alfanumerica per la verifica delle credenziali.

**Lista utenti:** un elenco dei nomi di login appartenenti al gruppo.

# Una password per il gruppo?

(Eh?)

Anche i gruppi hanno password?
Ebbene, sì.
È possibile aggiungersi ad un nuovo gruppo durante una sessione di lavoro.
È possibile configurare il SO in modo tale da effettuare una autenticazione con password a tal scopo.
Nella maggior parte dei casi, la password usata è quella nulla.

# Il file `/etc/group`

(Per l'identificazione dei gruppi)

Il file **`/etc/group`** contiene l'elenco dei gruppi di utenti noti al SO.

- È una base di dati primordiale.
- Elenco di record (uno per riga).
- I campi di un record sono separati dal carattere `:`.

Questo file è:

- leggibile da tutti gli utenti.
- Modificabile solo dall'utente **`root`**.

# I campi di `/etc/group`

(Informazioni associate ai gruppi; `man 5 group` per tutti i dettagli)

Nome del gruppo.
Password cifrata (nelle distribuzioni GNU/Linux attuali è conservata in un altro file, per motivi di sicurezza).
Identificatore gruppo.
Lista degli utenti appartenenti al gruppo.

# Il file `/etc/gshadow`

(Per la verifica delle credenziali del gruppo con password)

Il file **`/etc/gshadow`** contiene informazioni sulle password associate a ciascun gruppo.

- È una base di dati primordiale.
- Elenco di record (uno per riga).
- I campi di un record sono separati dal carattere `:`.

Questo file è leggibile e modificabile dal solo utente **`root`**.

- Si noti la separazione in un file separato (e ben protetto) delle informazioni riguardanti le password.

# I campi di `/etc/gshadow`

(Informazioni associate al gruppo utente; **`man 5 gshadow`** per i dettagli)

Nome del gruppo.
Password cifrata.
Utenti amministratori del gruppo.
Utenti membri del gruppo.

# “Shadow”? “Ombra”? Eh?

(Passwords get invisible, just like a shadow in the dark)

# Accesso alle risorse tramite file

(In UNIX, everything is a file)

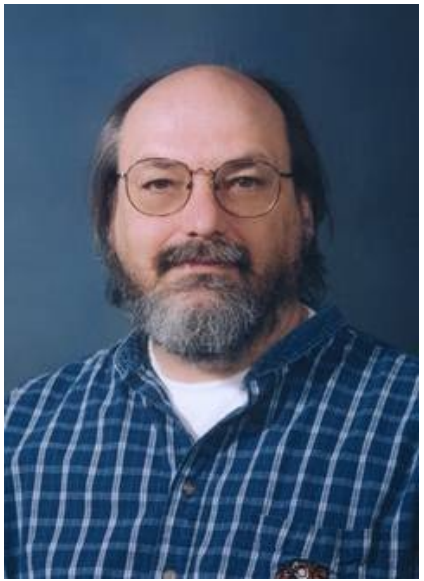
In GNU/Linux, le risorse hw/sw sono accessibili mediante file.

"In UNIX, everything is a file".

**Periferiche fisiche:** accessibili tramite file speciali nella directory **`/dev`**.

**Dispositivi virtuali:** accessibili tramite file speciali nella directory **`/dev`**.

**Archivi di dati:** accessibili tramite file regolari nel file system.

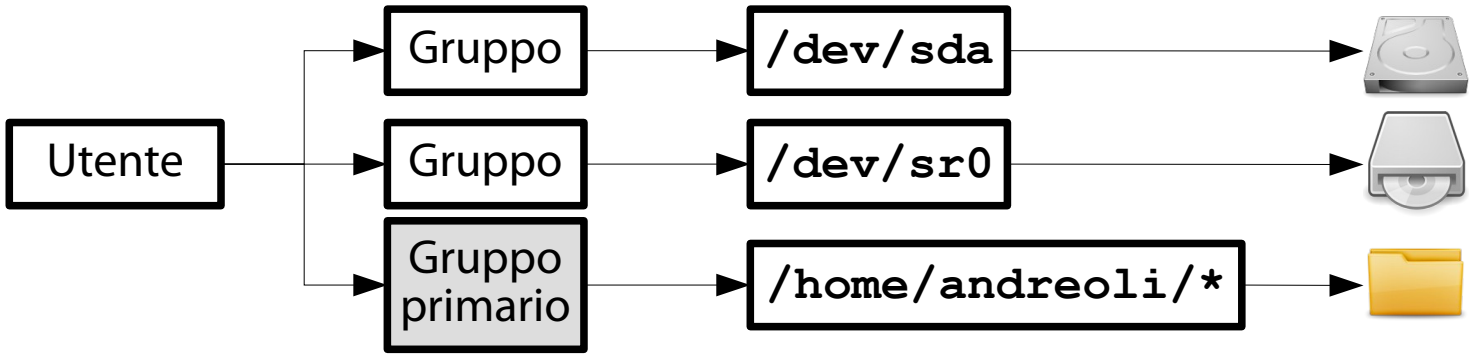


Ken Thompson
(1943-)
Autore del SO UNIX
Coautore del C
Inventore di UTF-8

# Gruppo primario, gruppi secondari

(La differenza è spiegata nella slide successiva)

Un utente può far parte di uno o più gruppi.
→ L'utente ha diversi privilegi che può far valere quando accede alle risorse.
Ad ogni istante, uno dei gruppi di lavoro può essere battezzato come **primario**. Gli altri gruppi sono detti **secondari**.



# Gli attributi di un file

(Utente creatore, gruppo di lavoro primario, permessi)

Ciascun file espone agli utenti tre attributi per il controllo degli accessi.

**Utente creatore:** nome di login dell'utente che ha creato fisicamente il file.

**Gruppo di lavoro:** il nome del gruppo primario di lavoro dell'utente nell'istante di creazione del file.

→ Il gruppo primario di un utente imposta il gruppo dell'utente creatore in un file.

**Insieme di permessi:** una rappresentazione sintetica dei privilegi di accesso al file.

# Privilegi di accesso

(Tre insiemi di utenti; tre azioni possibili su file)

I privilegi di accesso ad un file/directory sono specificati per tre insiemi di utenti.

- L'utente creatore del file.
- Gli utenti appartenenti al gruppo di lavoro.
- Tutti gli altri utenti.

I privilegi di accesso ad un file/directory specificano le operazioni possibili fra le seguenti.

- Lettura.
- Scrittura.
- Esecuzione (file) o accesso (directory).

# Rappresentazione dei permessi

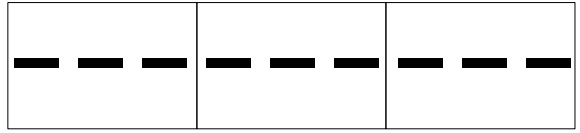
(Nove caratteri per i tre insiemi di utenti)

La rappresentazione dei permessi fornita dalle applicazioni di sistema agli utenti è una stringa di 9 caratteri.

I primi 3 caratteri indicano le azioni permesse per l'utente creatore del file.

I secondi 3 caratteri indicano le azioni permesse per gli utenti del gruppo di lavoro.

Gli ultimi 3 caratteri indicano le azione permesse per tutti gli altri utenti.



# Rappresentazione dei permessi

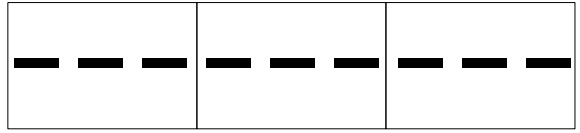
(I bit R, W, X specificano lettura, scrittura, esecuzione/accesso)

All'interno di ogni insieme:

il primo carattere specifica se è abilitato l'accesso in lettura (**`r`**) oppure no (**`-`**).

il secondo carattere specifica se è abilitato l'accesso in scrittura (**`w`**) oppure no (**`-`**).

il terzo carattere specifica se è abilitata l'esecuzione o l'accesso (**`x`**) oppure no (**`-`**).



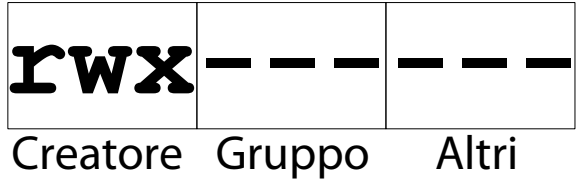
# Rappresentazione dei permessi

(Esempio di permessi per l'utente creatore)

Ad esempio, **`rwx`** per l'utente creatore indica:

- abilitazione accesso in lettura (**`r`**).
- abilitazione accesso in scrittura (**`w`**).
- abilitazione all'esecuzione/accesso (**`x`**).

→ L'utente creatore può fare tutto!



# Rappresentazione dei permessi

(Esempio di permessi per l'utente creatore)

Ad esempio, **`r-x`** per il gruppo di lavoro indica:

- abilitazione accesso in lettura (**`r`**).
- disabilitazione accesso in scrittura (**`-`**).
- abilitazione all'esecuzione/accesso (**`x`**).

→ Gli utenti del gruppo creatore possono leggere ed eseguire/accedere, ma non scrivere!

# Rappresentazione dei permessi

(Esempio di permessi per l'utente creatore)

Ad esempio, **`r-x`** per tutti gli altri utenti indica:

- abilitazione accesso in lettura (**`r`**).
- disabilitazione accesso in scrittura (**`-`**).
- abilitazione all'esecuzione/accesso (**`x`**).

→ Tutti gli altri utenti del sistema possono leggere ed eseguire/accedere, ma non scrivere!

# Credenziali di un'applicazione

(Con quali privilegi si presenta un'applicazione quando chiede l'accesso?)

Finora sono state introdotte strutture dati di natura "statica" (non mutano da sole).

- credenziali di un utente.
- gruppi di un utente.
- attributi di un file.

Anche le applicazioni in esecuzione hanno credenziali, che sono di natura "dinamica" (l'applicazione le acquisisce nel momento in cui parte):

- identificatore utente.
- identificatore gruppo primario.
- identificatori gruppi secondari.

# Identificatore utente di un'applicazione

(Quale utente ha lanciato l'applicazione?)

L'**identificatore utente** (**user identifier**) di un'applicazione è l'identificatore numerico dell'utente che ha lanciato l'applicazione.

L'identificatore utente dell'applicazione viene confrontato con l'identificatore utente del creatore del file. Se corrispondono, i permessi accordati all'applicazione sono quelli della prima terna.

# Identificatore gruppo primario

(Qual è il gruppo di lavoro primario dell'utente che ha lanciato l'applicazione?)

L'**identificatore del gruppo primario** (**primary group identifier**) è l'identificatore numerico del gruppo primario dell'utente nel momento che ha lanciato l'applicazione.

L'identificatore di gruppo dell'applicazione viene confrontato con l'identificatore di gruppo di lavoro del file. Se corrispondono, i permessi accordati all'applicazione sono quelli della seconda terna.

# Identificatore gruppi secondari

(Quali sono i gruppi di lavoro secondari dell'utente che ha lanciato l'app.?)

Gli **identificatori dei gruppi secondari** (**secondary group identifier**) sono gli identificatori numerici dei gruppo secondari di lavoro dell'utente nel momento che ha lanciato un'applicazione.

Gli identificatori di gruppo dell'applicazione sono confrontati uno per uno con l'identificatore di gruppo del creatore del file. Se uno di essi corrisponde, i permessi accordati all'applicazione sono quelli della seconda terna.

# Domanda

(Che succede se gli identificatori non corrispondono?)

Che cosa succede se:

l'identificatore utente di un'applicazione non corrisponde all'identificatore utente del creatore del file

E

l'identificatore di gruppo di un'applicazione non corrisponde all'identificatore gruppo primario del creatore del file

?

→ I permessi accordati al processo sono quelli della terza terna.

# “Guardi, guardi, lo vede il dito?”

(“Lo vede che stuzzica, e prematura anche?”)

# Credenziali di accesso e percorso file

(Un piccolo dettaglio da ricordare)

La procedura di controllo degli accessi va applicata ad ogni componente del percorso assoluto di un file.
Se, ad esempio, un file ha percorso assoluto:

**`/dev/sda`**

la procedura di controllo degli accessi va applicata ai seguenti percorsi:

**`/`**

**`/dev`**

**`/dev/sda`**

# Esempio di accesso ad una risorsa

(L'utente **`andreoli`** vuole aprire il file **`/dev/sda`** e leggere i blocchi del disco)

Si supponga che l'utente **`andreoli`** voglia leggere il file speciale di nome **`/dev/sda`**, per esempio lanciando il comando seguente:

```
less -Mr -f /dev/sda
```

Il file speciale **`/dev/sda`** rappresenta il primo disco rigido. Leggendo **`/dev/sda`** si leggono i blocchi del disco a basso livello.
Come funziona l'algoritmo di accesso alla risorsa?

# Esempio di accesso ad una risorsa

(I preliminari, parte I)

Il file **`/`** ha i seguenti attributi:

| **`rwxr-xr-x`** | **`root`** | **`root`** |
|---|---|---|
| Permessi | Utente creatore | Gruppo utente |

Il file **`/dev`** ha i seguenti attributi:

| **`rwxr-xr-x`** | **`root`** | **`root`** |
|---|---|---|
| Permessi | Utente creatore | Gruppo utente |

Il file **`/dev/sda`** ha i seguenti attributi:

| **`rw-rw----`** | **`root`** | **`disk`** |
|---|---|---|
| Permessi | Utente creatore | Gruppo utente |

# Esempio di accesso ad una risorsa

(I preliminari, parte II)

I gruppi dell'utente **`andreoli`** sono i seguenti:

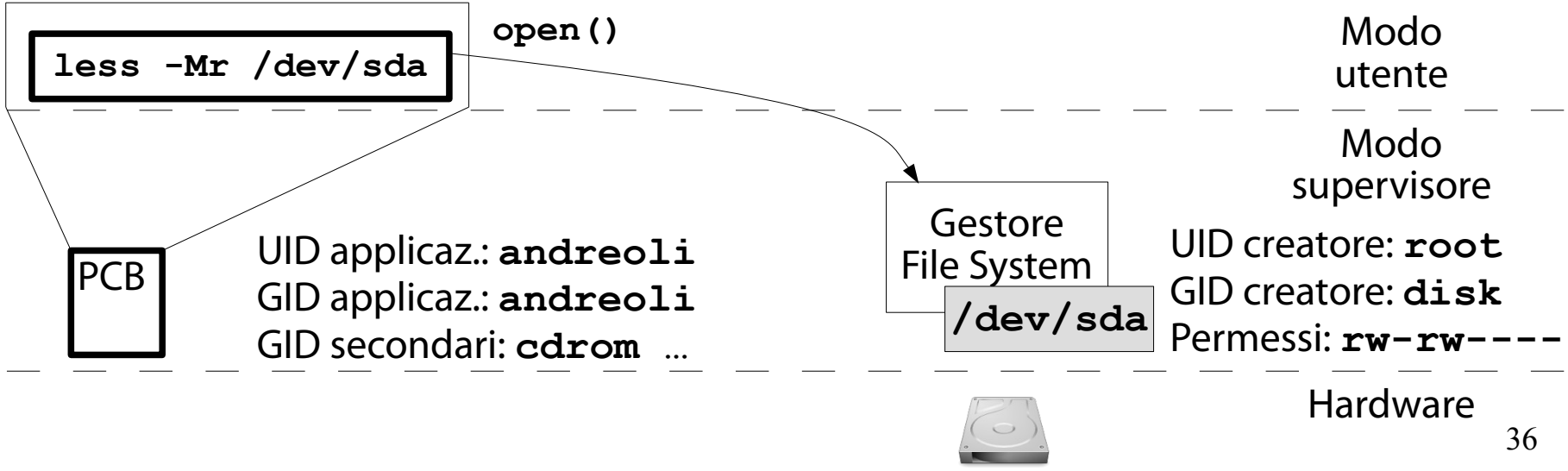
**`andreoli cdrom floppy audio dip video plugdev netdev scanner bluetooth vboxsf`**

Gli identificatori utente/gruppo dell'applicazione **`less`** sono **`andreoli/andreoli`**.

# Esempio di accesso ad una risorsa

(Apertura di **`/dev/sda`**)

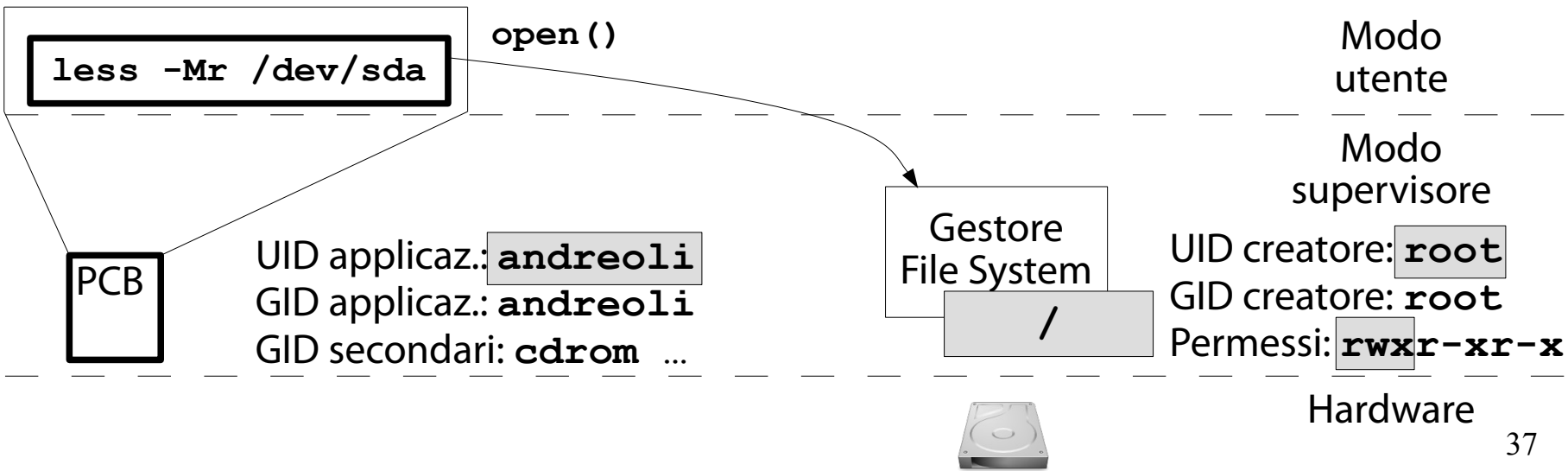
L'applicazione **`less`** apre il file **`/dev/sda`** in lettura con la funzione **`open()`**.
Parte la procedura di autorizzazione.



# Controllo accessi /

(Identificatore utente di **`less`** =?= Identificatore utente creatore del file)

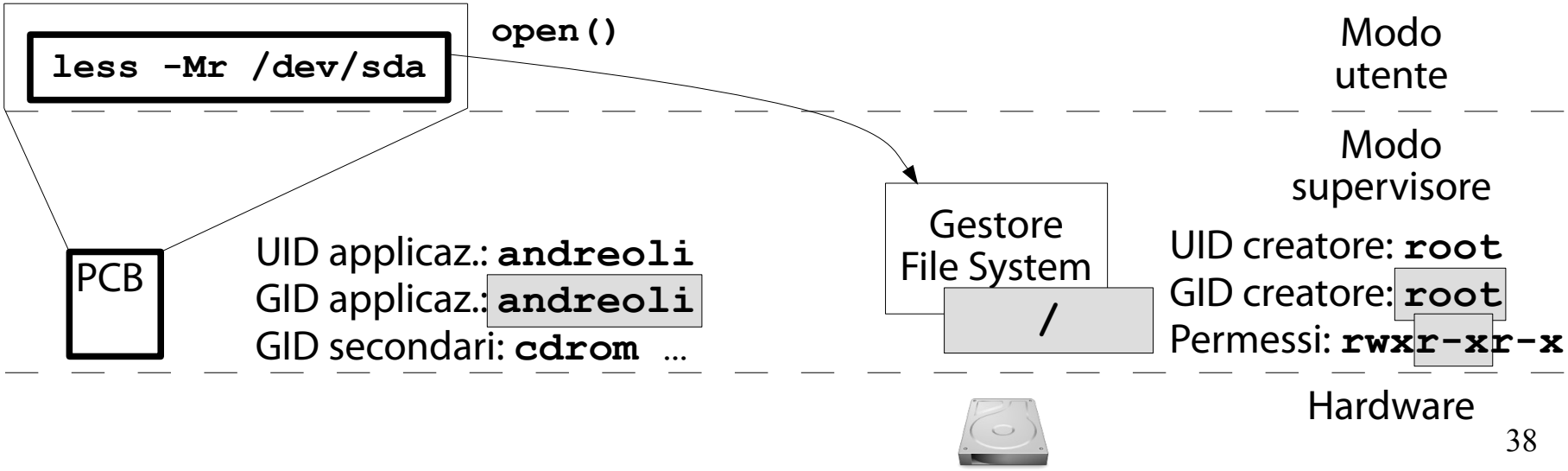
L'identificatore utente dell'applicazione **`less`** (**`andreoli`**) è uguale all'identificatore utente del creatore del file (**`root`**)? NO → Bisogna controllare il gruppo primario.



# Controllo accessi /

(Identificatore gruppo di **`less`** =?= Identificatore gruppo primario creatore)

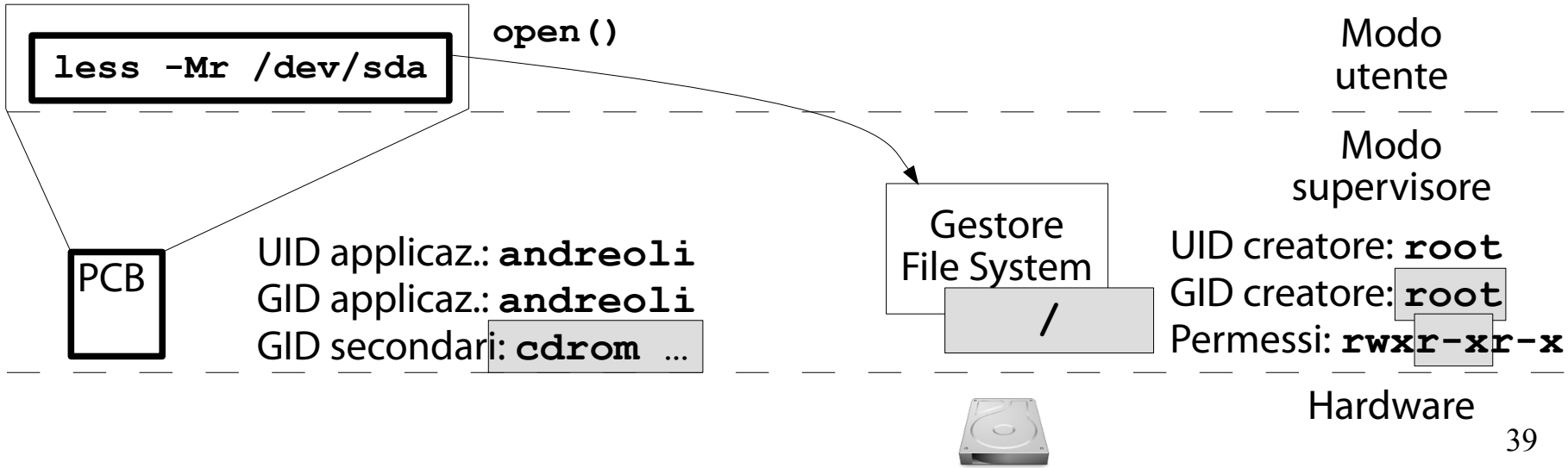
L'identificatore di gruppo dell'applicazione **`less`** (**`andreoli`**) è uguale all'identificatore del gruppo primario del creatore del file (**`root`**)? NO → Bisogna controllare i gruppi secondari.



# Controllo accessi /

(Identificatore gruppo secondario =?= Identificatore gruppo primario creatore)

Tra gli identificatori dei gruppi secondari dell'applicazione **`less`** (**`cdrom floppy audio dip video plugdev netdev scanner bluetooth vboxsf`**) ce ne è uno uguale all'identificatore del gruppo primario del creatore del file (**`root`**)? NO → Si applica l'ultima terna.



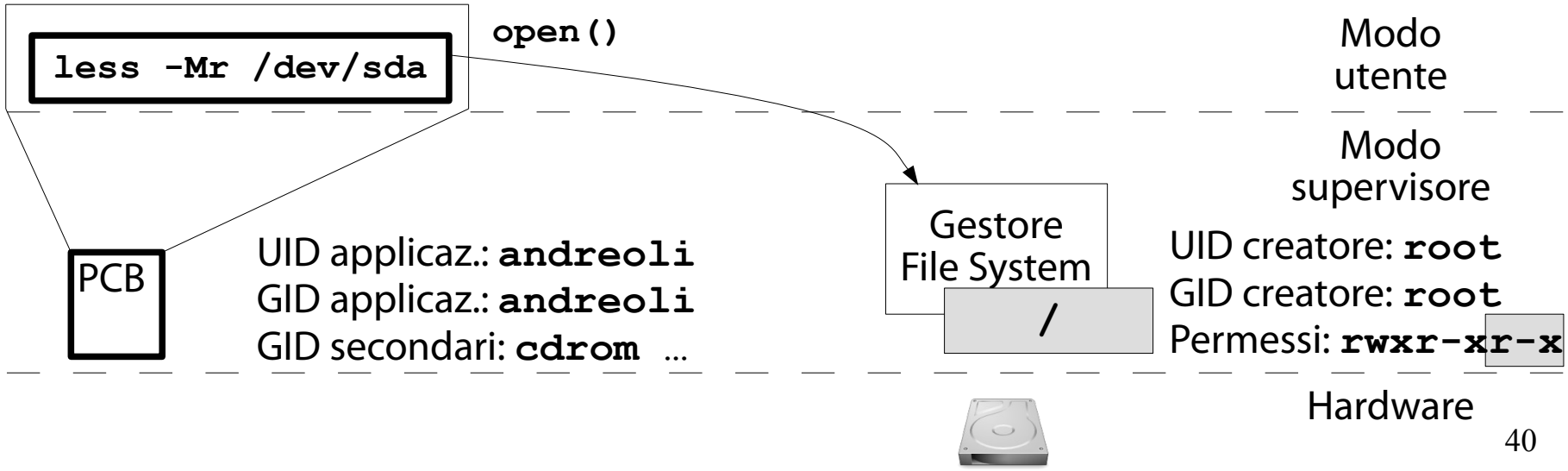
# Controllo accessi /

(I permessi accordati a **`less`** sono quelli della terza terna: **`r-x`**)

I privilegi ammissibili per **`less`** sulla directory **`/`** sono lettura (**`r`**) ed accesso (**`x`**).

**`less`** vuole aprire **`/`** (**`r`**) ed entrarci (**`x`**), per aprire **`/dev`**.

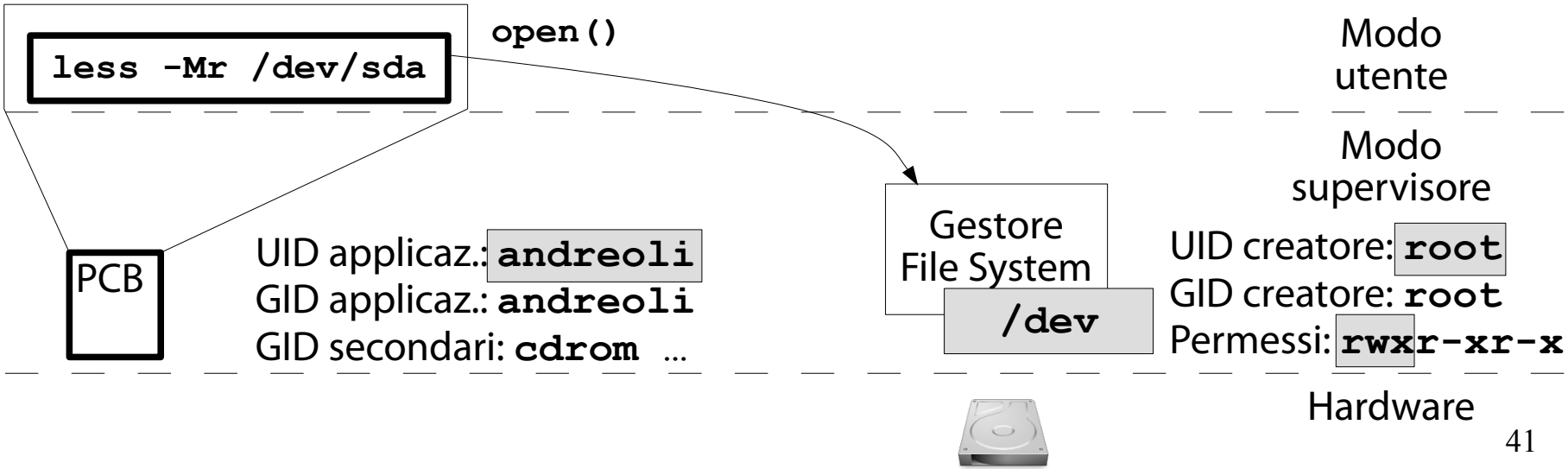
Gli viene accordato il permesso.



# Controllo accessi /dev

(Identificatore utente di **`less`** =?= Identificatore utente creatore del file)

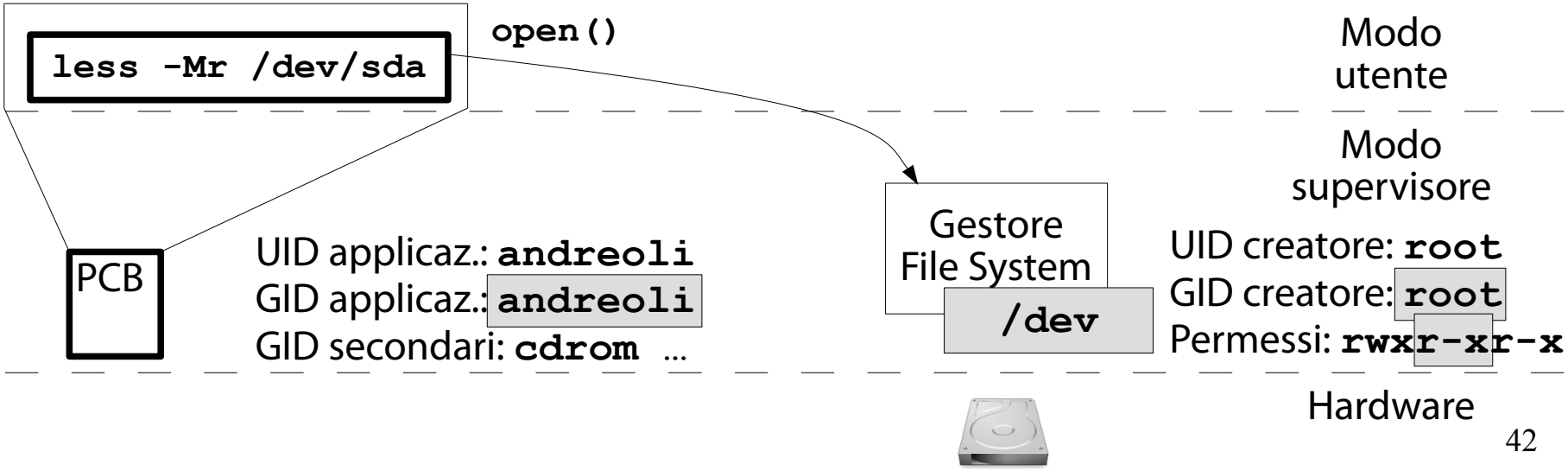
L'identificatore utente dell'applicazione **`less`** (**`andreoli`**) è uguale all'identificatore utente del creatore del file (**`root`**)? NO → Bisogna controllare il gruppo primario.



# Controllo accessi /dev

(Identificatore gruppo di **`less`** =?= Identificatore gruppo primario creatore)

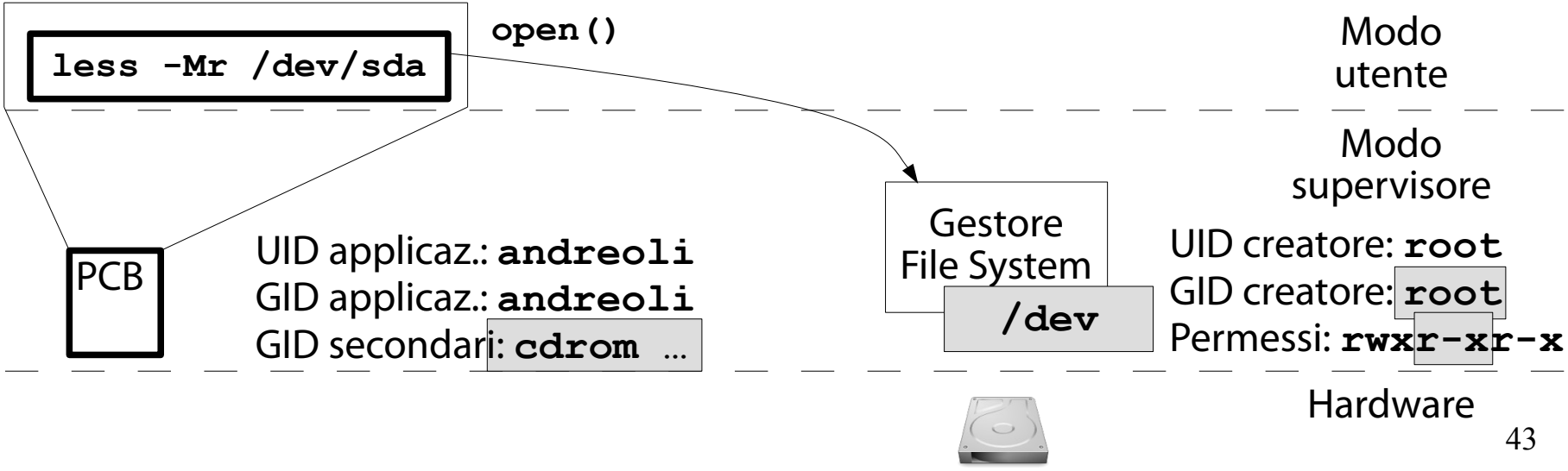
L'identificatore di gruppo dell'applicazione **`less`** (**`andreoli`**) è uguale all'identificatore del gruppo primario del creatore del file (**`root`**)? NO → Bisogna controllare i gruppi secondari.



# Controllo accessi /dev

(Identificatore gruppo secondario =?= Identificatore gruppo primario creatore)

Tra gli identificatori dei gruppi secondari dell'applicazione **`less`** (**`cdrom floppy audio dip video plugdev netdev scanner bluetooth vboxsf`**) ce ne è uno uguale all'identificatore del gruppo primario del creatore del file (**`root`**)? NO → Si applica l'ultima terna.



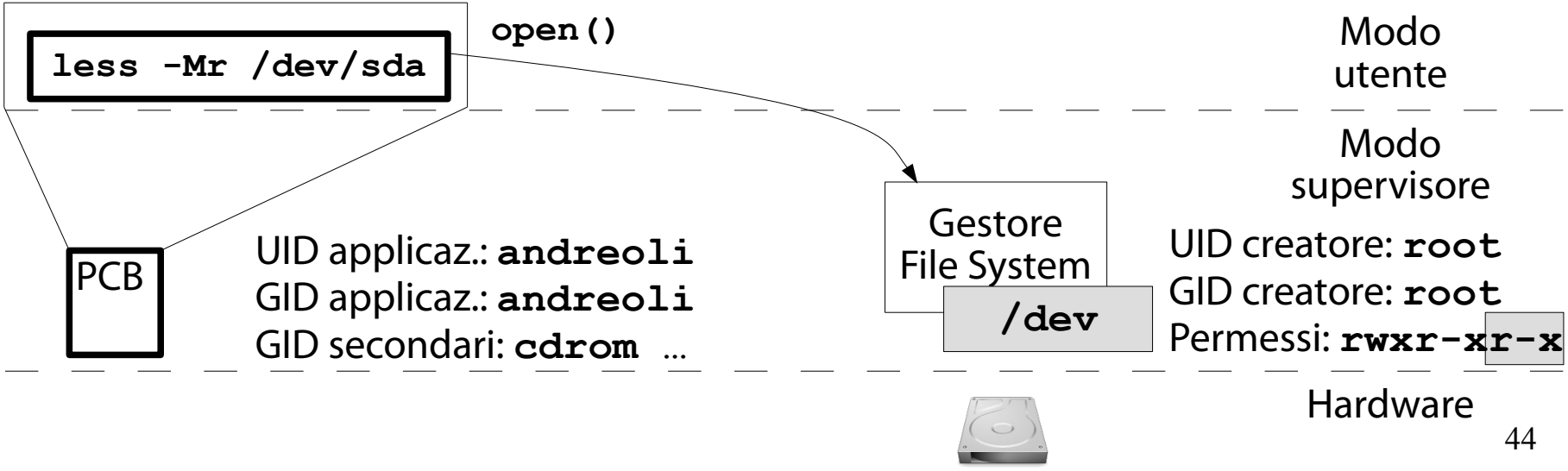
# Controllo accessi /dev

(I permessi accordati a **`less`** sono quelli della terza terna: **`r-x`**)

I privilegi ammissibili per **`less`** sulla directory **`/dev`** sono lettura (**`r`**) ed accesso (**`x`**).

**`less`** vuole aprire **`/`** (**`r`**) ed entrarci (**`x`**), per aprire **`/dev/sda`**.

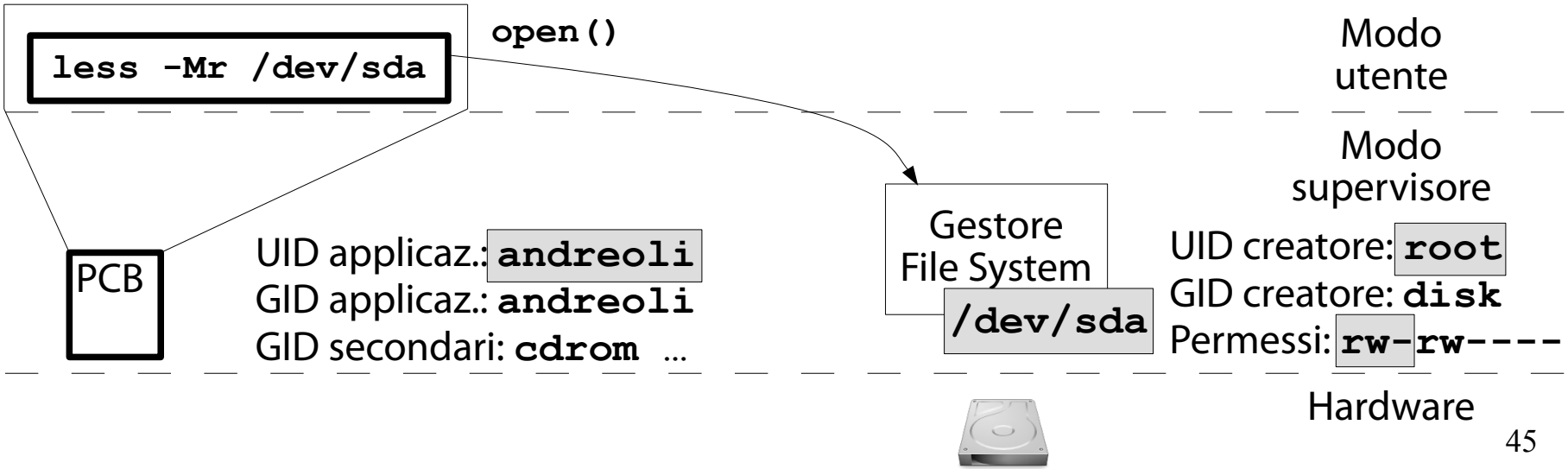
Gli viene accordato il permesso.



# Controllo accessi /dev/sda

(Identificatore utente di **`less`** =?= Identificatore utente creatore del file)

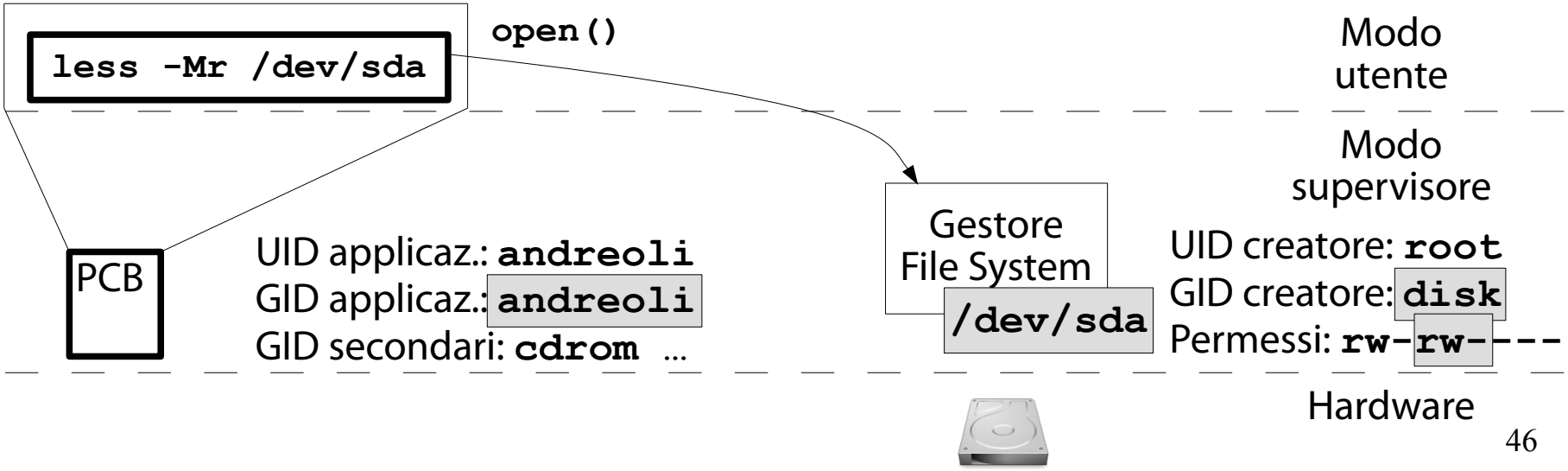
L'identificatore utente dell'applicazione **`less`** (**`andreoli`**) è uguale all'identificatore utente del creatore del file (**`root`**)? NO → Bisogna controllare il gruppo primario.



# Controllo accessi /dev/sda

(Identificatore gruppo di **`less`** =?= Identificatore gruppo primario creatore)

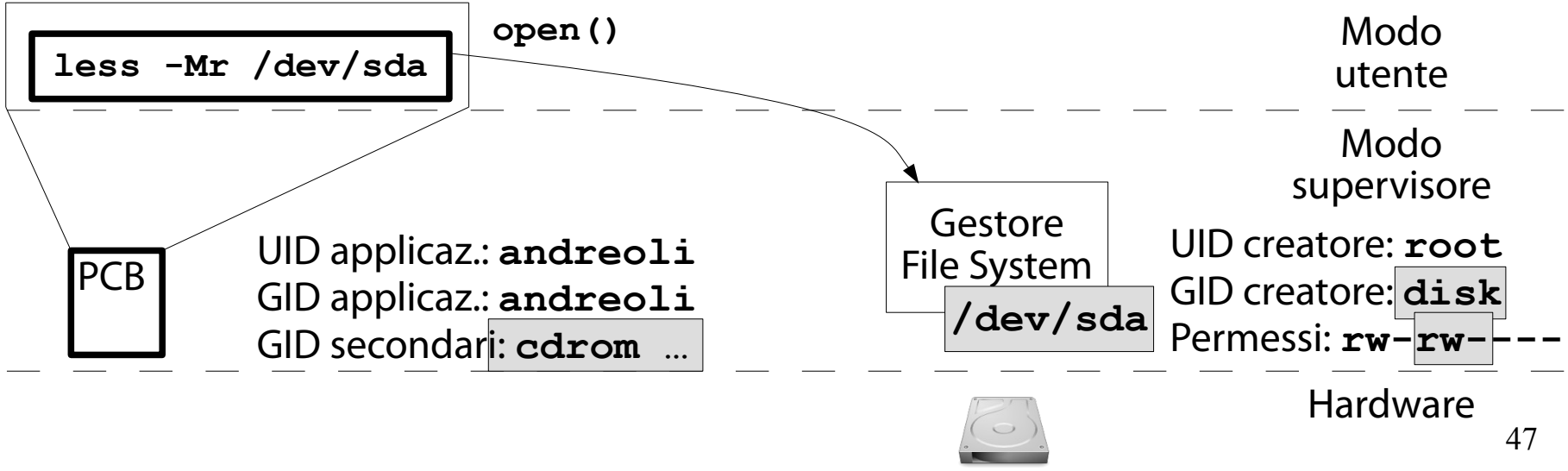
L'identificatore di gruppo dell'applicazione **`less`** (**`andreoli`**) è uguale all'identificatore del gruppo primario del creatore del file (**`disk`**)? NO → Bisogna controllare i gruppi secondari.



# Controllo accessi /dev/sda

(Identificatore gruppo secondario =?= Identificatore gruppo primario creatore)

Tra gli identificatori dei gruppi secondari dell'applicazione **`less`** (**`cdrom floppy audio dip video plugdev netdev scanner bluetooth vboxsf`**) ce ne è uno uguale all'identificatore del gruppo primario del creatore del file (**`disk`**)? NO → Si applica l'ultima terna.



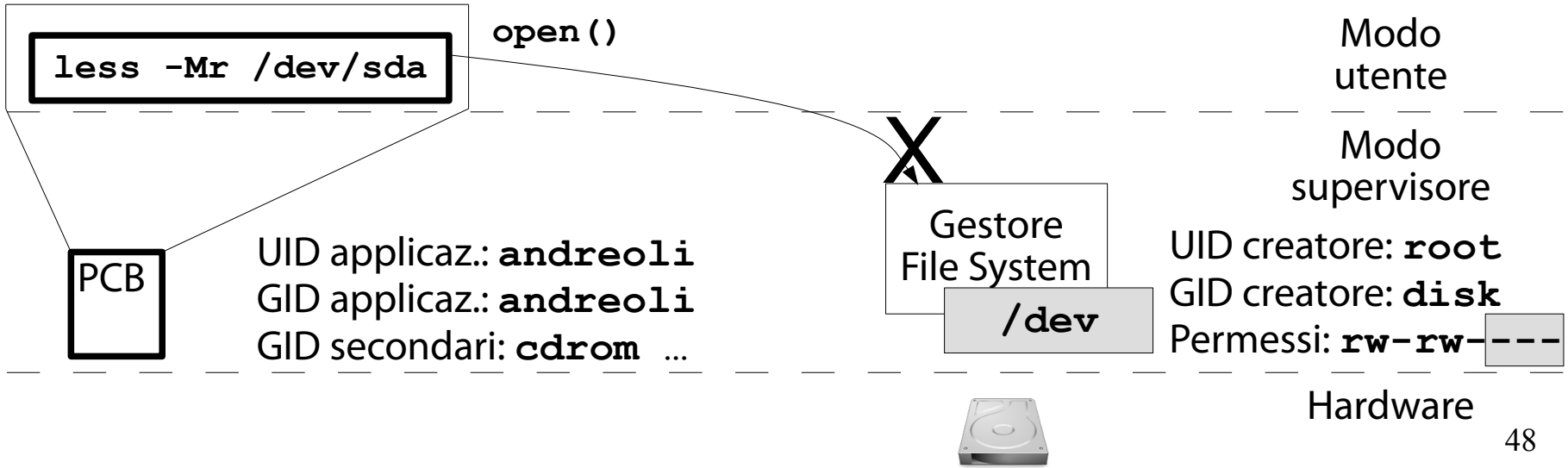
# Controllo accessi /dev/sda

(I permessi accordati a **`less`** sono quelli della terza terna: **`---`**)

I privilegi ammissibili per **`less`** sulla directory **`/dev/sda`** sono nulli (**`---`**).

**`less`** vuole aprire in lettura **`/dev/sda`** (**`r`**).

Gli viene negato il permesso.



# BIG FAT WARNING

(You have been warned)

Lo schema ora proposto è una semplificazione della realtà.
Sono stati omessi dettagli molto importanti.

L'obiettivo dello schema attuale è quello di introdurre dolcemente (si spera) i concetti principali dello schema di gestione dei permessi.

# Domanda da 1M USD

(Rivolta a voi, cari studenti)

Quali credenziali avrebbe dovuto presentare l'utente **`andreoli`** per poter accedere al file speciale **`/dev/sda`**?

# GESTIONE DEI GRUPPI

# Scenario e interrogativi

(Come sono gestiti i gruppi? È possibile visionare le proprietà di un gruppo?)

**Scenario:** in un SO moderno (time sharing, multi-utente), un amministratore vuole creare, modificare, rimuovere gruppi.

**Domande:**

Esistono strumenti per creare, modificare, rimuovere gruppi?

Esistono strumenti per prendere visione delle caratteristiche di un gruppo?

# Creazione di un gruppo in Debian

(Si usa il comando **`addgroup`**)

Debian GNU/Linux e derivate semplificano molto la procedura di creazione di un nuovo gruppo.
È sufficiente eseguire il comando seguente da amministratore:

**`addgroup`** ***`nome_gruppo`***

Ad esempio:

**`addgroup`** ***`prove`***

Il comando **`addgroup`** non è interattivo.

# Il risultato del comando **`addgroup`**

(Un gruppo nuovo di zecca)

Il risultato del comando **`addgroup`** è un gruppo nuovo di zecca (nel caso in questione, il gruppo di nome **`prove`**).

# Esercizio 1 (1 min.)

Create un nuovo gruppo di nome **`studenti`**.

# Una domanda spontanea

(E sacrosanta!)

Nelle slide precedenti si è appurato che **`adduser`** e **`useradd`** creano un gruppo primario avente lo stesso nome dell'utente.

Ad esempio:

**`adduser prova`** → crea gruppo primario **`prova`** (e lo associa all'utente di nome **`prova`**).

Domanda: a che serve allora **`addgroup`**?

# La risposta

(Altrettanto ovvia ed altrettanto sacrosanta!)

**`addgroup`** serve a creare gruppi che, in futuro:

potranno essere associati ad un utente come gruppi secondari.

potranno divenire nuovi gruppi primari di un utente.

# Analisi del gruppo

(Visione del file **`/etc/group`**)

Si apra il file **`/etc/group`** e si cerchi il record relativo al gruppo di nome **`prove`**:

```
grep prove /etc/group
```

Si dovrebbe ottenere il record seguente:

```
prove:x:1003
```

# Analisi del gruppo

(Visione del file **`/etc/group`**)

Si apra il file **`/etc/group`** e si cerchi il record relativo al gruppo di nome **`prove`**:

```
grep prove /etc/group
```

Si dovrebbe ottenere il record seguente:

```
prove:x:1003
```

Nome del gruppo

# Analisi del gruppo

(Visione del file **`/etc/group`**)

Si apra il file **`/etc/group`** e si cerchi il record relativo al gruppo di nome **`prove`**:

```
grep prove /etc/group
```

Si dovrebbe ottenere il record seguente:

```
prove:x:1003
```

Password
(celata all'utente)

# Analisi del gruppo

(Visione del file **`/etc/group`**)

Si apra il file **`/etc/group`** e si cerchi il record relativo al gruppo di nome **`prove`**:

```
grep prove /etc/group
```

Si dovrebbe ottenere il record seguente:

```
prove:x:1003
```

Identificatore gruppo (parte da 1000, non è necessariamente uguale a 1003)

# Esercizio 2 (1 min.)

Trovate l'identificatore del gruppo **`studenti`**.

# Rimozione di un gruppo in Debian

(Si usa il comando **`delgroup`**)

Debian GNU/Linux e derivate semplificano molto la procedura di rimozione di un gruppo esistente.
È sufficiente eseguire il comando seguente da amministratore:

```
delgroup nome_gruppo
```

Ad esempio:

```
delgroup prove
```

Il comando **`delgroup`** non è interattivo (non ha bisogno di esserlo).

# Il risultato del comando **`delgroup`**

(Group has been ripped off, just like Clint would)

Il risultato del comando **`delgroup`** è la rimozione completa del gruppo.

- Cancellazione del gruppo dalle lista dei gruppi.
- Cancellazione del gruppo dall'elenco dei gruppi di ogni utente.

# Esercizio 3 (1 min.)

Cancellate il gruppo **`studenti`**.

# Una osservazione ed una domanda

(Potreste non trovare **`addgroup`** e **`delgroup`** in altri distribuzioni GNU/Linux)

I due comandi **`addgroup`** e **`delgroup`** sono tipici delle distribuzioni Debian GNU/Linux e derivate.
Con tutta probabilità non li troverete nelle altre distribuzioni GNU/Linux non basate su Debian.

La diaspora dei sistemi UNIX colpisce ancora…

La domanda nasce spontanea: quali comandi usano, allora, le altre distribuzioni GNU/Linux?

# I comandi **groupadd** e **groupdel**

(Questi li trovate in tutte le distribuzioni GNU/Linux)

Le altre distribuzioni GNU/Linux usano i comandi **groupadd** e **groupdel**, analoghi dal punto di vista concettuale.

> **man groupadd** e **man groupdel** per tutti i dettagli.

Questi comandi presentano grossomodo le stesse difficoltà di **addgroup** e **delgroup**.

# Esercizio 4 (1 min.)

Provate a creare nuovamente il gruppo **`studenti`**, questa volta usando il comando **`groupadd`**.
Verificate la creazione del nuovo gruppo.

# Esercizio 5 (1 min.)

Provate a rimuovere nuovamente il gruppo **`studenti`**, questa volta usando il comando **`groupdel`**.
Verificate la rimozione del nuovo gruppo.

# Modifica di un gruppo esistente

(Si usa il comando **`groupmod`**)

Il comando **`groupmod`** permette di modificare le proprietà e le risorse di un gruppo esistente.
La sintassi è semplice:

**`groupmod [opzioni]`** ***`nome_gruppo`***

Che cosa è possibile modificare, esattamente?

Nome gruppo, identificatore gruppo, password.

# Un esempio concreto

(Proviamo a modificare alcune caratteristiche del gruppo **`prove`**)

Si faccia piazza pulita di un eventuale gruppo **`prove`** preesistente.

```
delgroup prove
```

Si crei nuovamente un gruppo **`prove`**:

```
addgroup prove
```

Si provi a cambiare il gruppo **`prove`** nel modo che segue.

Nome gruppo **`prove`** → **`prove2`**

Identificatore gruppo **`1001`** → **`1100`**

# Le opzioni di **`groupmod`** richieste

(**`-n`** e **`-g`**)

Spulciando la pagina di manuale di **`groupmod`**

```
man groupmod
```

si dovrebbero individuare le opzioni seguenti:

- **`-n`**: cambia il nome del gruppo
- **`-g`**: cambia l'identificatore gruppo

# Il comando richiesto

(Tutto sommato, abbastanza semplice da costruire)

Il comando richiesto è, in definitiva:

```
groupmod -n prove2 -g  1100 prove
```

Lo si provi!

# Esercizio 6 (3 min.)

Create un gruppo **`studenti`** (se non è già esistente).
Modificate le seguenti proprietà del gruppo:

- Nome gruppo → **`studenti2`**;
- Identificatore del gruppo → **`1300`**.

Verificate la corretta applicazione delle modifiche.

# Analisi dei gruppi di un utente

(Si usa il comando **`groups`**)

Il comando **`groups`** mostra i gruppi di appartenenza di un utente. La sintassi è semplice:

**`groups [`*`nome_di_login`*`]`**

Se l'amministratore vuole vedere i gruppi di un utente, bisogna fornire il corrispettivo nome di login come parametro.
Se un utente (incluso l'amministratore) vuole stampare i propri gruppi, può eseguire **`groups`** senza parametri.

# Alcuni esempi concreti

(Si stampano i gruppi dell'utente **`root`**)

Per stampare i gruppi dell'utente **`root`**, si esegue il comando **`groups`**:

```
groups root
```

In generale, l'utente root è associato al solo gruppo **`root`**. Perché?

Non essendo **`root`** soggetto a restrizioni sull'accesso alle risorse, non ha bisogno di gruppi secondari che le definiscono.

# Alcuni esempi concreti

(Si stampano i gruppi dell'utente attivo sul terminale)

Per stampare i gruppi dell'utente attivo (ad esempio, **`studente`**), si può procedere in due modi diversi.

Si diventa superutente e si esegue il comando **`groups studente`**:

```
su -
groups studente
```

In alternativa, si esegue **`groups`** da utente **`studente`**:

```
groups
```

# Modifica dei gruppi di un utente

(Da parte di amministratore, tramite diversi comandi)

Il SO mette a disposizione diversi comandi per modificare i gruppi di lavoro primario e secondari di un utente.

Perché tanti comandi per svolgere la stessa operazione?

Diaspora UNIX…

# Il comando **`usermod`**

(Presente su tutti i sistemi UNIX; opzioni **`-g`** e **`-G`**)

Nei sistemi UNIX il comando **`usermod`** è usato per la modifica dei gruppi di lavoro primario e secondari.

Opzione **`-g`** ***`nome_gruppo`***: specifica il nuovo gruppo di lavoro primario;

Opzione **`-G ng_1,ng_2,...`**: specifica il nuovo l'insieme completo dei gruppi secondari.

# Un esempio concreto

(Si assegnano nuovi gruppi primario e secondari all'utente **`prova`**)

Si provi ad assegnare all'utente **`prova`** i seguenti gruppi di lavoro primario e secondari.

Gruppo primario: **`root`**.

Gruppi secondari: **`disk cdrom floppy audio`**.

I comandi richiesti sono i seguenti (servono le credenziali di **`root`**):

```
usermod -g root prova
usermod -G disk,cdrom,floppy,audio prova
```

# Esercizio 7 (3 min.)

Create un utente **`studente2`** (se non esiste già).
Impostate i seguenti nuovi gruppi per l'utente **`studente2`**.
Gruppo primario: **`root`**.
Gruppi secondari: **`disk cdrom floppy audio dip video plugdev games netdev scanner bluetooth`**.
Verificate la corretta applicazione dei nuovi gruppi.

# BIG FAT WARNING

(You have been warned)

L'opzione **`-G`** di **`usermod`** vuole l'elenco di tutti i gruppi di lavoro secondari.
Se si vuole inserire un solo gruppo di lavoro secondario, occorre fornire sia lui sia tutti i gruppi secondari esistenti.
Se si scrive **`-G`** ***`nuovo_gruppo`***, tutti gli altri gruppi saranno cancellati!

# Esercizio 8 (2 min.)

Provate ad aggiungere un gruppo secondario **`root`** all'utente **`studente2`** usando il seguente comando sbagliato:

```
usermod -G root studente2
```

Notate qualcosa di strano?

# Aggiunta di utenti

(Si usano le opzioni **-g** e **-a** del comando **usermod**)

Per aggiungere gruppi secondari è necessario specificare anche l'opzione **-a** di **usermod**.

Ad esempio, per aggiungere il gruppo secondario **root** all'utente **prova**:

```
usermod -a -G root prova
```

Si verifichi l'aggiunta del gruppo secondario con il comando:

```
groups prova
```

# Esercizio 9 (2 min.)

Aggiungete i seguenti gruppi secondari all'utente **`studente2`**:

**`disk cdrom floppy audio dip video plugdev games netdev scanner bluetooth`**

# Un altro comando per gestire gruppi

(Il comando **gpasswd**)

Un comando alternativo per la gestione di gruppi utente è **gpasswd**.

**man gpasswd** per tutti i dettagli.

Per aggiungere un utente ad un gruppo:

**gpasswd -a *nome_di_login gruppo***

Per rimuovere un utente da un gruppo:

**gpasswd -d *nome_di_login gruppo***

# Alcuni esempi concreti

(Comando **`gpasswd`**, opzioni **`-a`** e **`-d`**)

Per aggiungere il gruppo **`lp`** all'utente **`prova`**:

```
gpasswd -a prova lp
```

Per rimuovere il gruppo **`lp`** dall'utente **`prova`**:

```
gpasswd -d prova lp
```

# BIG FAT WARNING

(You have been warned)

Si provi a modificate uno o più gruppi di lavoro secondari dell'utente con cui ci si è autenticati.

Le modifiche non sono visibili fino al prossimo login!

# Un esempio concreto

(Aggiunta di un gruppo secondario, visione dell'elenco dei gruppi)

Se, ad esempio, si è autenticati come utente **`studente`**, si aggiunga il gruppo **`root`** all'elenco dei gruppi secondari:

```
gpasswd -a studente root
```

In un altro emulatore di terminale si stampi l'elenco completo dei gruppi secondari:

```
groups
```

Il gruppo **`root`** non è ancora presente.

# GESTIONE DEI PERMESSI

# Scenario e interrogativi

(Come modificare i permessi di un file? Come vedere i permessi di un file?)

**Scenario:** in un SO moderno (time sharing, multi-utente), un utente vuole impostare impostare i permessi di accesso alle risorse.

**Domande:**

Esistono strumenti per modificare i permessi di file e/o directory?

Esistono strumenti per prendere visione dei permessi di un file?

# Visione degli attributi di un file

(Si usa l'opzione `-l` del comando `ls`)

L'opzione **`-l`** del comando **`ls`** mostra gli attributi principali di un file, tra cui quelli di interesse nella presente lezione.

- Utente creatore del file.
- Gruppo del file.
- Permessi del file.

Ad esempio, create un file vuoto nella vostra home directory:

```
touch /home/studente/file.txt
```

Visionatene l metadati:

```
ls -l /home/studente/file.txt
```

# Gli attributi di interesse

(**Creatore**, gruppo, permessi)

```
studente@debian:~$ ls -l file.txt
-rw-r--r-- 1 studente studente 0 nov  8 23:22 file.txt
studente@debian:~$
```

Utente creatore del file

# Gli attributi di interesse

(Creatore, **gruppo**, permessi)

```
studente@debian:~$ ls -l file.txt
-rw-r--r-- 1 studente studente 0 nov  8 23:22 file.txt
studente@debian:~$
```

Gruppo di lavoro del file

# Gli attributi di interesse

(Creatore, gruppo, **permessi**)

```
studente@debian:~$ ls -l file.txt
-rw-r--r-- 1 studente studente 0 nov  8 23:22 file.txt
studente@debian:~$
```

Permessi del file

# Modifica dell'utente creatore

(Si usa il comando **chown**)

Il comando **chown** modifica l'utente creatore di un file e/o di una directory. L'uso è semplice:

```
chown [OPZIONI] nome_di_login file
```

Ad esempio, per modificare il creatore del file **file.txt** da **studente** a **root** eseguite il seguente comando da amministratore:

```
chown root /home/studente/file.txt
```

# Modifica del gruppo del file

(Si usa il comando **chgrp**)

Il comando **chgrp** modifica il gruppo di un file e/o di una directory. L'uso è semplice:

```
chgrp [OPZIONI] nome_gruppo file
```

Ad esempio, per modificare il gruppo del file **file.txt** da **studente** a **root** eseguite il seguente comando da amministratore:

```
chgrp root /home/studente/file.txt
```

# Modifica di utente creatore e gruppo

(Comando **`chown`**, primo argomento del tipo ***`utente:gruppo`***)

Il comando **`chown`** fornisce una scorciatoia per impostare simultaneamente utente creatore e gruppo di un file.

È sufficiente specificare il primo argomento con una stringa **`nome_di_login:gruppo`**.

```
chown nome_di_login:gruppo file
```

# Un esempio concreto

(Modifica simultanea di utente creatore e gruppo di **`file.txt`**)

Ad esempio, per modificare simultaneamente le seguenti modifiche sul file di testo di nome **`/home/studente/file.txt`**:

utente creatore → **`studente`**
gruppo del file → **`studente`**

è possibile usare il solo comando **`chown`**:

```
chown studente:studente /home/studente/file.txt
```

# Cambio ricorsivo di utente e/o gruppo

(Si usa l'opzione **-R** dei comandi **chown** e **chgrp**)

L'opzione **-R** di **chown** e **chgrp** opera ricorsivamente su una directory.

Ad esempio, per impostare utente e gruppo alla coppia **studente:studente** per tutti i file e le sottodirectory contenute in **/home/studente**, si esegue il comando seguente:

```
chown -R studente:studente /home/studente
```

# Esercizio 10 (2 min.)

Create un file vuoto dal nome **`lista.txt`** nella vostra home directory. Cambiate utente creatore e gruppo di **`lista.txt`** nel modo seguente:

utente creatore → **`root`**
gruppo del file → **`root`**

Verificate la corretta applicazione degli attributi utente creatore e gruppo del file.

# Modifica dei permessi del file

(Si usa il comando **chmod**)

Il comando **chmod** modifica i permessi del file. L'uso è meno semplice dei comandi precedenti:

```
chmod [OPZIONI] permessi file
```

# Rappresentazioni dei permessi

(Comprese dal comando **`chmod`**)

Il comando **`chmod`** comprende due distinte rappresentazioni dei permessi.

**Rappresentazione testuale**. I permessi da applicare ad un file/directory sono rappresentati tramite una stringa.
**Rappresentazione ottale**. I permessi da applicare ad un file/directory sono rappresentati tramite un numero in base 8 (ottale).

# Rappresentazione testuale

(Molto simile alla rappresentazione dei permessi fornita da **`ls`**)

Nella rappresentazione testuale, **`chmod`** accetta una o più **stringhe di permessi** separate da una virgola:

**`str1,str2,...,strN`**

Ciascuna stringa di permessi ha il formato seguente:

**`insieme_di_utenti±insieme_di_permessi`**

Il simbolo ± indica la presenza di un + (aggiunta) o di un – (rimozione).

# Insieme degli utenti

(Specifica su quale insieme di utenti vale il permesso)

L'insieme degli utenti è una stringa composta dai caratteri **`u`**, **`g`**, **`o`**, **`a`**.

- **`u`** → I permessi si applicano all'utente creatore.
- **`g`** → I permessi si applicano al gruppo del file.
- **`o`** → I permessi si applicano ai restanti utenti.
- **`a`** → I permessi si applicano a tutti gli utenti.

# Esempi di insiemi degli utenti

(Tanto per (non) chiarire le idee)

| | | |
|---|---|---|
| **ugo** | → | tutti gli utenti del sistema. |
| **a** | → | tutti gli utenti del sistema. |
| **go** | → | gli utenti appartenenti al gruppo del file<br>E<br>il resto del mondo |

# Insieme dei permessi

(Specifica i permessi associati ad un insieme di utenti)

L'insieme dei permessi è una stringa composta dai caratteri **`r`**, **`w`**, **`x`**, **`s`**.

- **`r`** → Si applica il permesso di lettura.
- **`w`** → Si applica il permesso di scrittura.
- **`x`** → Si applica il permesso di esecuzione (file)
  O
  di ingresso (directory)
- **`s`** → Si applica il bit SETUID/SETGID

**ATTENZIONE!** L'insieme presentato è semplificato al minimo indispensabile.

# Esempi di permessi

(Tanto per (non) chiarire le idee)

**`rwx`** → lettura, scrittura, esecuzione

**`rw`** → lettura, scrittura

**`rx`** → lettura, esecuzione

**`rws`** → lettura, scrittura, esecuzione SETUID/SETGID

# Esempio di esecuzione di **chmod**

(Con permessi in rappresentazione testuale)

Si supponga di voler assegnare al file di testo **/home/studente/file.txt** il seguente insieme di permessi: **rw-rw-r--**.

Si costruisce la stringa dei permessi.

Permessi di lettura e scrittura per l'utente creatore e per gli utenti appartenenti al gruppo del file: **ug+rw**.

Permessi di lettura per gli altri utenti: **o+r**.

Il comando richiesto è, pertanto:

```
chmod ug+rw,o+r /home/studente/file.txt
```

# Esercizio 11 (3 min.)

Create un file vuoto dal nome **`lista2.txt`** nella vostra home directory. Usando **`chmod`** con rappresentazione testuale dei permessi, impostate i permessi seguenti sul file:

**`rwxr-xr-x`**

Verificate la corretta applicazione dei permessi.

# Rappresentazione ottale

(Molto diversa dalla rappresentazione dei permessi fornita da **`ls`**)

Nella rappresentazione ottale, **`chmod`** accetta un **numero intero** rappresentato in **base ottale**.

Il numero può essere lungo da una a quattro cifre. Se le cifre sono meno di quattro, si prependono degli zeri.
Ad esempio: 4 → 0004, 755 → 0755.

# Insieme degli utenti

(Specifica su quale insieme di utenti vale il permesso)

Si consideri un insieme di permessi **0755**.

| | | |
|---|---|---|
| Seconda cifra (**7**) | → | insieme di permessi per l'utente creatore. |
| Terza cifra (**5**) | → | insieme di permessi per gli utenti appartenenti al gruppo del file. |
| Quarta cifra (**5**) | → | insieme di permessi per tutti gli altri utenti. |

# Insieme dei permessi

(Specifica i permessi associati ad un insieme di utenti)

Si consideri un insieme di permessi **`0755`**.

L'insieme dei permessi è un numero intero lungo 4 bit (da 0 a 7), ottenibile sommando i numeri 0, 1, 2, 4.

- **`0`** → nessun permesso.
- **`4`** → Si applica il permesso di lettura.
- **`2`** → Si applica il permesso di scrittura.
- **`1`** → Si applica il permesso di esecuzione (file)
  O
  di ingresso (directory)

# Esempi di permessi

(Tanto per (non) chiarire le idee)

**7** → lettura, scrittura, esecuzione

**6** → lettura, scrittura

**5** → lettura, esecuzione

**4** → lettura

**2** → scrittura

**1** → esecuzione

# Permessi SETUID/SETGID

(Prima cifra del numero in ottale)

Si consideri un insieme di permessi **0755**. Il valore della prima cifra stabilisce ulteriori permessi (SETUID/SETGID).

| | | |
|---|---|---|
| Prima cifra = **4** | → | È impostato il bit SETUID. |
| Prima cifra = **2** | → | È impostato il bit SETGID. |
| Prima cifra = **1** | → | È impostato lo sticky bit (non trattato in questo corso). |
| Prima cifra = **0** | → | Non è impostato nient'altro. |

# Esempio di esecuzione di **`chmod`**

(Con permessi in rappresentazione ottale)

Si supponga di voler assegnare al file di testo **`/home/studente/file.txt`** il seguente insieme di permessi: **`rw-rw-r--`**.

Si costruisce il numero in base ottale.

# Esempio di esecuzione di **chmod**

(Con permessi in rappresentazione ottale)

Permessi di lettura e scrittura per l'utente creatore: seconda cifra impostata a **4+2 = 6**.

Permessi di lettura e scrittura per tutti gli utenti appartenenti al gruppo del file: terza cifra impostata a **4+2 = 6**.

Permessi di lettura per gli altri utenti: quarta cifra impostata a **4**.

Assenza di bit SETUID/SETGID: prima cifra impostata a **0**.

Il comando richiesto è, pertanto:

```
chmod 0664 /home/studente/file.txt
```

# Esercizio 12 (3 min.)

Copiate il file **`/usr/bin/passwd`** nella vostra home directory. Usando **`chmod`** con rappresentazione ottale dei permessi, impostate i permessi seguenti sul file:

**`rwsr-xr-x`**

Verificate la corretta applicazione dei permessi.

# Per sorridere un po’

(Dr. Who never lies)

# Interazione con i dispositivi

(Tramite l'aggiunta in gruppi opportuni)

I dispositivi (tipicamente, periferiche) sono accedute tramite file speciali presenti nella directory **`/dev`**.
Esempio (già visto):

file speciale **`/dev/sda`** → primo disco rigido SATA.

L'accesso a **`/dev/sda`** è mediato da opportuni permessi.

Quali sono? Un utente normale li possiede già? Di quale gruppo è richiesta l'appartenenza?

# Visione degli attributi di **/dev/sda**

(Per capire quali permessi è necessario presentare)

Per vedere gli attributi del file speciale /dev/sda potete digitare il comando:

```
ls -l /dev/sda
```

Si ottiene l'output seguente:

```
studente@debian:~$ ls -l /dev/sda
brw-rw---- 1 root disk 8, 0 nov  8 17:33 /dev/sda
studente@debian:~$
```

# Quali permessi servono?

(Per leggere **`/dev/sda`**)

Per leggere **`/dev/sda`** da utente normale è necessario presentare le credenziali seguenti.

```
studente@debian:~$ ls -l /dev/sda
brw-rw---- 1 root disk 8, 0 nov  8 17:33 /dev/sda
studente@debian:~$
```

Essere l'utente **`root`**

Far parte del gruppo **`disk`**

→ Per un utente normale, l'unico modo per leggere **`/dev/sda`** è quello di inserirsi nel gruppo **`disk`**.

# Aggiunta del gruppo all'utente

(Facile, una volta che si conosce il gruppo)

L'accesso diretto al file fallisce. Se digitate:

```
less -Mr -f /dev/sda
```

il SO risponderà con un messaggio del tipo “Permesso negato”.

Ci si aggiunga al gruppo **`disk`**, scrivendo il comando seguente da utente **`root`**:

```
gpasswd -a studente disk
```

# Lettura del disco a basso livello

(Richiede un nuovo login utente)

L'aggiunta del gruppo diventa operativa ad un nuovo login. Si termini la sessione corrente e ci si autentichi nuovamente.

In seguito, si verifichi l'appartenenza al gruppo **`disk`**:

```
groups
```

L'output dovrebbe contenere la stringa **`disk`**.

Si tenti una lettura del disco a basso livello:

```
less -Mr -f /dev/sda
```

Si dovrebbe vedere il contenuto del disco.