Permessi base e speciali in Unix-Linux

Le autorizzazioni di base vengono assegnati utilizzando tre tipi di accesso:

Lettura, scrittura ed esecuzione.

Questi tipi di accesso vengono utilizzati per determinare l'accesso ai file per il proprietario, il gruppo del file, e altri (tutti gli altri). La lettura, scrittura ed esecuzione autorizzazioni possono essere rappresentati come le lettere **`r`** , **`w`** , e **`x`** . Possono anche essere rappresentati come numeri binari come ciascun permesso o è acceso o spento **(`0`)**. Quando rappresentato come un numero, l'ordine è sempre letto come **`rwx`** , dove **`r`** ha un valore on di **`4`** , **`w`** ha un valore su **`2`** e **`x`** ha un valore su **`1`** .

La Tabella riassume le possibili possibilità numerici e alfabetici. Durante la lettura della colonna "Directory Listing", una `-` è usato per rappresentare un permesso che è impostato su off.

Permessi Tabella UNIX ®

Valore	Autorizzazione	Directory Listing
0	No lettura, no scrittura, no execute	`---`
1	No lettura, no scrittura, si esecuzione	`--x`
2	No di lettura, si scrittura, non eseguire	`-w-`
3	No di lettura, si scrittura, si esecuzione	`-wx`
4	Leggere, non scrivere, non esecuzione	`r--`
5	Leggere, nessuna scrittura, esecuzione	`rx`
6	Leggere, scrivere, non esecuzione	`rw-`
7	Leggere, scrivere, eseguire	`rwx`

Trucco con LS: Utilizzare il `-l` argomento per ls (1) per visualizzare un lungo elenco di directory che include una colonna d'informazioni sui permessi del file per il proprietario, il gruppo, e tutti gli altri. Ad esempio, un `ls -l` in una directory arbitraria può mostrare:

```
ls -l
 totale 530
 -rw-r - r-- 1 ruota radice 512 5 Settembre 12:31 myfile
 -rw-r - r-- 1 ruota radice 512 5 Settembre 12:31 altrofile
 -rw-r - r-- ruota 1 root 7680 5 Settembre 00:31 email.txt
```

Il primo carattere (più a sinistra) nella prima colonna indica se il file è un file regolare, una directory, un dispositivo di carattere speciale, una presa, o qualsiasi altro dispositivo speciale pseudo-file. In questo esempio, il `-` indica un file regolare. I tre caratteri successivi, `rw-` in questo esempio, indicano i permessi per il proprietario del file. I tre caratteri successivi, `r--` , indicano i permessi per il gruppo che il file appartiene. Gli ultimi tre caratteri, `r--` , indicano i permessi per il resto del mondo. Un trattino significa che il permesso sia spento. In questo esempio, i permessi sono impostati in modo che il proprietario può leggere e scrivere il file, il gruppo può leggere il file, e il resto del mondo, in grado di leggere solo il file. Secondo la tabella di cui sopra, i permessi per questo file sarebbero `644` , dove ogni cifra rappresenta i tre parti del permesso del file.

Come i permessi di controllo del sistema su dispositivi? Unix-like tratta la maggior parte dei dispositivi hardware come un file che i programmi possono aprire, leggere e scrivere dati. Questi file speciali di dispositivo sono memorizzati in `/dev/` .

Anche gli Elenchi sono anche trattati come file. Hanno il lettura, scrittura ed esecuzione. Il bit eseguibile per una directory ha un significato leggermente diversa da quella del file. Quando una directory è contrassegnato eseguibile, significa che è possibile modificare in quella directory usando il comando **cd (1)** . Ciò significa anche che è possibile accedere ai file all'interno della directory, fatte salve le autorizzazioni per i file stessi.

Per eseguire un elenco di directory, l'autorizzazione di lettura deve essere impostato sulla directory. Per eliminare un file che si conosce il nome di, è necessario disporre di scrittura *ed* esecuzione per la directory contenente il file.

Ci sono più bit di permessi, ma sono utilizzati principalmente in circostanze particolari, come binari setuid e directory appiccicose. Per ulteriori informazioni sui permessi dei file e su come impostare, fare riferimento a **chmod (1)** .

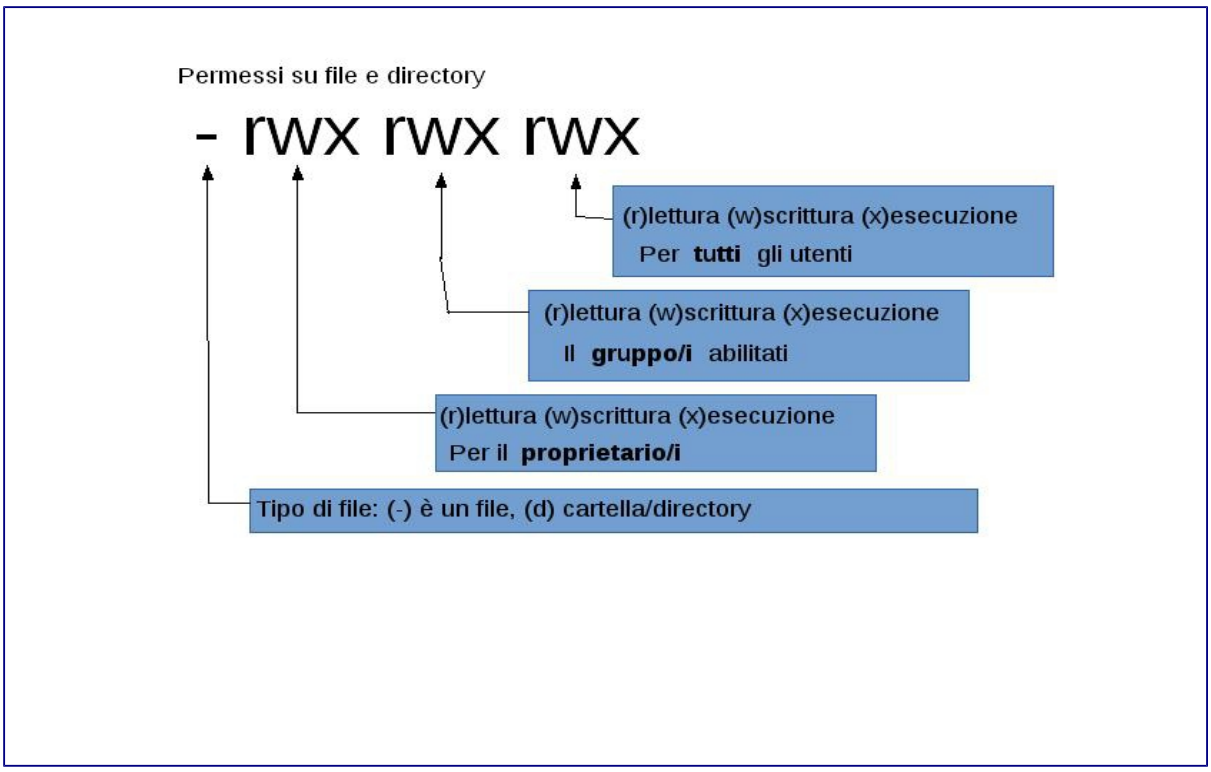
ATTENZIONE: Se un file è in esecuzione è consigliabile che possa anche leggere almeno in livello di root.

Quelli più usati sono :

chmod **777** (tutti possono fare tutto) file eseguibile

chmod **644** (tutti possono solo leggerlo e il proprietario fare tutto salvo eseguirlo) file documento

chmod **664** (tutti possono solo leggerlo e il proprietario e il suo gruppo fare tutto salvo eseguirlo) file documento



Permessi simbolici

Permessi simbolici usano caratteri invece di valori ottali per assegnare i permessi a file o directory. Permessi simbolici usano la sintassi di (chi) (azione) (permessi), in cui sono disponibili i seguenti valori:

Opzione	Lettera	Rappresenta
(Chi)	u	Utente
(Chi)	g	~~Proprietario~~ gruppo
(Chi)	o	Altro
(Chi)	un	All (“tutti”)
(Azione)	+	Aggiunta di autorizzazioni
(Azione)	–	Rimozione delle autorizzazioni
(Azione)	=	Le autorizzazioni esplicitamente del set
(permessi)	r	Leggere

Opzione	Lettera	Rappresenta
(permessi)	w	Scrivere
(permessi)	x	Eseguire
(permessi)	t	Bit Sticky
(permessi)	s	Impostare UID o GID

Questi valori sono usati con **chmod (1)** , ma con le lettere al posto dei numeri. Ad esempio, il seguente comando potrebbe bloccare altri utenti di accedere `FILE` :

```
% chmod go= FILE
```

Un elenco separato può essere fornito quando più di una serie di modifiche apportate a un file deve essere fatta. Ad esempio, il seguente comando rimuove il gruppo e permesso di scrittura al `FILE` , e aggiunge l'esecuzione per tutti:

```
% chmod go-w,a+x FILE
```

Per rendere eseguiile un file sh

```
% chmod a+x FILE
```

Le Bandiere dei file

Oltre ai permessi dei file, gli UNIX-like supportano l'uso di "bandiere dei file". Questi flag aggiungono un ulteriore livello di sicurezza e di controllo sui file, ma non le directory.

ATTENZIONE: Con bandiere di file, anche `root` può essere impedito di rimuovere o modificare i file.

Bandiere dei file vengono modificati utilizzando **chflags (1**) . Ad esempio, per attivare il flag indelebile sistema sul file `file1` , emettere il seguente comando:

```
# chflags sunlink file1
```

Per disattivare il undeletable flag di sistema, mettere un "no" davanti alla `sunlink` :

```
# chflags nosunlink file1
```

Per visualizzare le bandiere di un file, utilizzare **`-lo`** con **ls (1)** :

```
# ls -lo file1
-rw-r - r-- 1 trhodes trhodes sunlnk 0 1 Marzo 05:54 file1
```

Diverse bandiere di file possono essere aggiunti o rimossi dalla sola `root` dell'utente. In altri casi, il proprietario del file può impostare i suoi flag di file. Fare riferimento a **chflags (1)** e **(2) chflags** per ulteriori informazioni.

I `setuid` , `setgid` e `sticky` Permessi

Altro che le autorizzazioni già discussi, ci sono altre tre impostazioni specifiche che tutti gli amministratori dovrebbero conoscere:

I permessi `setuid` , `setgid` e `sticky`.

Queste impostazioni sono importanti per alcune operazioni UNIX-like in quanto forniscono funzionalità normalmente non concessa agli utenti normali.

Per capire la differenza tra l'ID utente reale ed ID utente effettivo deve essere sempre osservato.

L'ID utente reale è l'UID che possiede o avvia il processo. L'UID efficace è l'ID utente del processo dove viene eseguito.

A titolo di esempio, **passwd (1)** viene eseguito con l'ID utente reale quando un utente cambia la propria password. Tuttavia, per aggiornare il database delle password, il comando viene eseguito come l'effettiva ID della `root` dell'utente. Questo permette agli utenti di modificare le password senza vedere l'errore "Autorizzazione negata".

Il permesso setuid può essere impostato premettendo un set di autorizzazioni con il numero quattro (4) cifra, come mostrato nel seguente esempio:

```
# chmod 4755 suidexample.sh
```

Le autorizzazioni per `suidexample.sh` ora hanno il seguente aspetto:

```
1 trhodes trhodes x -rwsr-xr-63 29 Agosto 06:36 suidexample.sh
```

Si noti che un `s` è ora parte del set di autorizzazioni designato per il proprietario del file, sostituendo il bit eseguibile. Questo permette di utility che necessitano di autorizzazioni elevate, come **passwd (1)** .

Nota:

Il **`nosuid` mount (8)** è opzione che può causare tali binari di fallire in silenzio, senza avvertire l'utente. Tale opzione non è completamente affidabile come `nosuid` involucro che può essere in grado di aggirarlo.

Per visualizzare questo in tempo reale, aperti due terminali. Su uno, tipo `passwd` come utente normale. Mentre attende una nuova password, controllare la tabella dei processi e guardare le informazioni utente per **passwd (1)** :

In Terminale A:

```
  Cambiare password locale per trhodes
Vecchia Password:
```

In Terminale B:

```
# ps aux | grep passwd
trhodes 5232 0.0 0.2 3420 1608 0 R + 2:10 0: 00.00 grep passwd
```

```
radice 5211 0.0 0.2 3620 1724 2 I + 2:09 0: 00.01 passwd
```

Sebbene **passwd (1)** è gestito come un normale utente che usa l'effettivo UID di `root` .

Il `setgid` permesso esegue la stessa funzione del `setuid` permesso; tranne che altera le impostazioni del gruppo. Quando un'applicazione o utility esegue con questa impostazione, sarà concesso le autorizzazioni in base al gruppo che possiede il file, non l'utente che ha avviato il processo.

Per impostare il `setgid` autorizzazione su un file, fornire **chmod (1)** con il numero di due (2) cifra:

```
# chmod 2755 sgidexample.sh
```

Nel seguente elenco, si noti che la `s` è ora nel campo designato per le impostazioni di autorizzazione del gruppo:

```
-rwxr-sr-x 1 trhodes trhodes 44 31 Agosto 01:49 sgidexample.sh
```

Nota:

In questi esempi, anche se lo script di shell in questione è un file eseguibile, non sarà eseguito con un EUID differente o ID utente effettivo. Questo perché gli script di shell non possono accedere alle setuid (2) chiamate di sistema.

I `setuid` e `setgid` bit dei permessi può abbassare la sicurezza del sistema, consentendo autorizzazioni elevate. Il terzo permesso speciale, la `sticky bit` , in grado di rafforzare la sicurezza di un sistema.

Quando il `sticky bit` è impostato su una directory, permette la cancellazione dei file solo dal proprietario del file. Ciò è utile per evitare la cancellazione del file in elenchi pubblici, come ad esempio `/tmp` , da parte di utenti che non possiedono il file. Per utilizzare questa autorizzazione, prefisso il set di autorizzazioni con il numero uno (1):

```
# chmod 1777 /tmp
```

Il `sticky bit` autorizzazione apparirà come una `t` alla fine del set di autorizzazioni:

```
# ls -al / | grep tmp
```

Guardiani

Oltre questi esistono i Guardiani (in inglese Guardian detti anche PIG, porci) che sono processi che controllano gli utenti (di solito salvo il root) di cosa fa su un file messo sotto osservazione e possono intervenire per evitare la cancellazione, modifica, lettura, esecuzione dello stesso.
In questo caso l'unica possibilità in caso di problemi è intervenire sulla configurazione del Guardian.
APP-Armor, Selinux, SecureLinux sono Guardian.