

Attributi basici

{-,d,b,c,p}	tipologia di file: normale, directory, block-device, character-device, pipe
UID (2/4 byte)	identificatori del proprietario e del suo gruppo
GID (2/4 byte)	
rwx rwx rwx	permessi di accesso per proprietario, gruppo, altri (codifica ottale)
SUID (1 bit)	specifica di identificazione dinamica
SGID (1 bit)	per chi utilizza il file
Sticky (1 bit)	per le directory rimuove la possibilità di cancellare files se non si è l'owner

La prima è una parte di informazioni che ci dice esattamente l'oggetto di I/O, su cui eventualmente vogliamo lavorare, che cosa è. Ossia la tipologia del file su cui lavoriamo.

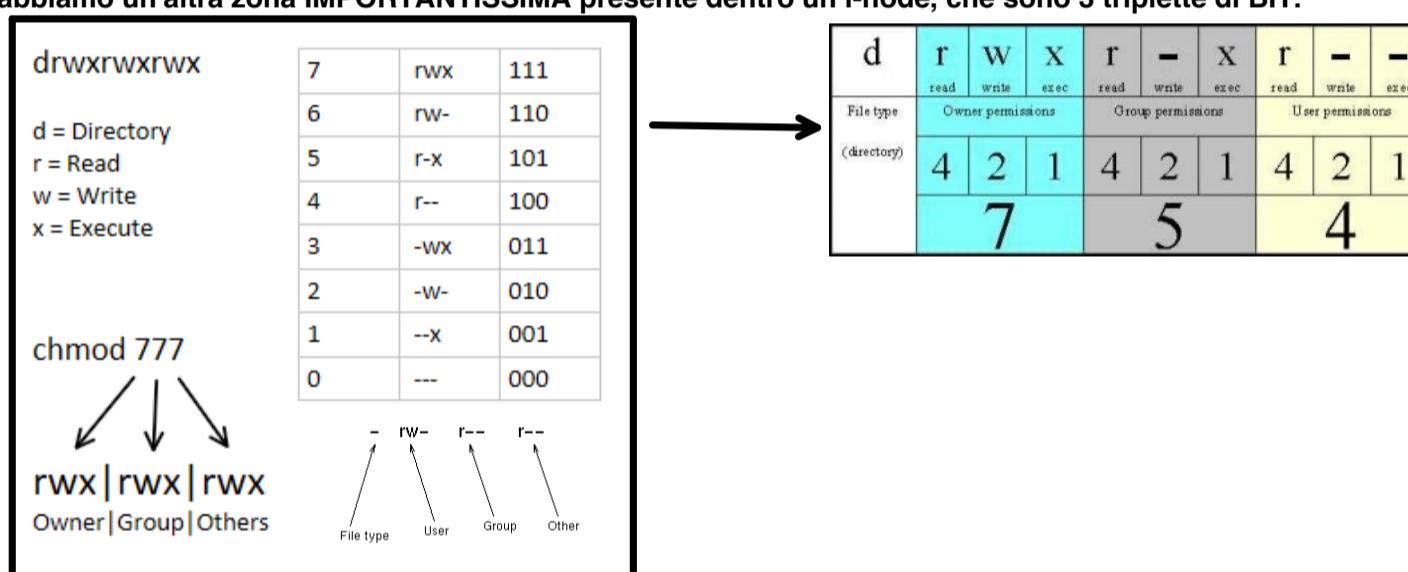
Abbiamo poi lo UID (UserID) e il GID(GroupID): essi sono due codici numerici che identificano il proprietario del file e il gruppo di appartenenza del proprietario.

Quando noi lavoriamo su un sistema UNIX gli utenti sono raggruppati in gruppi che eventualmente sono distinti fra di loro, un utente può essere spostato da un gruppo ad un altro, ma la cosa interessante che noi abbiamo all'interno dell'organizzazione dei file system UNIX è che, a ciascuno di questi utenti, viene associato un codice numerico, e non solo perché anche i gruppi hanno un codice numerico.

È tutto basato su codici numerici.

Anche se poi in realtà noi siamo abituati a lavorare con nomi di utenze: quando noi entriamo in login sul sistema NON forniamo il nostro codice numerico, forniamo la nostra username, ora cercheremo di capire come l'username è relazionato a questi codici numerici.

Poiabbiamo un'altra zona IMPORTANTESSIMA presente dentro un i-node, che sono 3 triplette di BIT.



Sono i permessi di accesso all'oggetto di I/O: La prima tripla specifica quali sono i permessi di accesso che il proprietario del file (UTENTE ASSOCIAZIONE ALLO UID) ha sul file stesso.

La seconda tripla indica i permessi per gli utenti appartenenti allo stesso gruppo del gruppo del proprietario, quindi GUID.

L'ultima tripla indica i permessi per gli utenti in altri gruppi sconosciuti, il resto del mondo.

chmod (abbreviazione dalla lingua inglese di *change mode*, *cambio modalità*) è un comando dei sistemi operativi Unix e Unix-like, e più in generale dei sistemi POSIX, che è utilizzato per modificare i permessi di accesso di file e directory. È anche usato per cambiare alcuni flag che identificano modalità speciali come *setuid*, *setgid* e lo *sticky bit*. L'invocazione di chmod per soddisfare la richiesta effettuata di cambiare i permessi risente del filtro applicato da umask.

Rappresentazione ottale [modifica | modifica wikitesto]
Nella rappresentazione ottale ogni cifra indica i permessi per i differenti utenti.
• chmod UGO nomefile
La cifra U rappresenta il livello di permessi per l'utente, G il livello di permessi del gruppo e O il livello di permessi generale
La tabella seguente indica il significato dei singoli valori

Valore binario (rwx)	Valore decimale	Permesso
111	7	lettura, scrittura ed esecuzione
110	6	lettura e scrittura
101	5	lettura ed esecuzione
100	4	solo lettura
011	3	scrittura ed esecuzione
010	2	solo scrittura
001	1	solo esecuzione
000	0	nessuno

Esempi:
• chmod 734 nomefile
assegna tutti i permessi all'utente, scrittura ed esecuzione per il gruppo e solo lettura per tutti gli altri.
• chmod 777 nomefile
assegna tutti i permessi all'utente corrente, al suo gruppo ed anche a tutti gli altri.
• chmod -R 777 nomedirectory

1. **chmod** (Change mode - Cambia modalità)
2. **chown** (Change ownership - Cambia proprietà)
3. **chgrp** (Change group - Cambia gruppo)

Permessi a Livello Utente
Le opzioni che puoi usare per controllare i permessi a livello utente sono:

- u – Concede permessi a un utente
- g – Concede permessi a un gruppo (un gruppo di utenti)
- o – Concede permessi agli altri (chi non appartiene a nessuna delle categorie precedenti).

Permessi al Livello di File
Queste opzioni controllano i permessi al livello di file.

- r – Concede permessi di lettura
- w – Concede permessi di scrittura
- x – Concede permessi di esecuzione

come il precedente ma riguarda una directory e tutti i file esistenti all'interno della stessa.

Come gestire i permessi in modalità simbolica

```
chmod u+rwx,go+r install.sh
```

Comando per applicare la casistica sopra citata usando la modalità simbolica

Smembriamo ogni parte e cerchiamo di capirne il significato:

- `u+rwx` rappresenta l'aggiunta dei privilegi di lettura, scrittura, esecuzione per gli utenti
- `go+r` rappresenta l'aggiunta dei privilegi di lettura per gruppi e altri

Queste opzioni devono essere precedute dall'operatore '+' o '-'.

'+' indica che si aggiunge un nuovo permesso, '-' indica che si revoca un permesso esistente.

Ecco un esempio:

```
chmod +r sample.txt
```

Comando per aggiungere permessi di lettura a un file

Il comando qui sopra aggiunge permessi di lettura al file `sample.txt`.

Qui abbiamo un file chiamato `install.sh` che ha tutti i permessi (Lettura, Scrittura, Esecuzione). Rimuoviamo il permesso di esecuzione per questo file di script.

```
chmod -x install.sh
```

Comando per rimuovere i permessi di esecuzione per un file

```
sudo chmod -R <permessi> <nomedirectory>
```

Sintassi per rimuovere i permessi di lettura da una directory ricorsivamente

Poi abbiamo **SetUserID (SUID)** e **SetGroupID (SGID)**, ENTRAMBI AD UN BIT. Cosa significare avere un file marcato con un SUID pari a 1?

Questa cosa è curiosa quando parliamo di oggetti che in realtà sono dei programmi.

Supponiamo di avere un programma P associato ad uno specifico utente U, ma all'interno dell'i-nodes associato al programma P, c'è il SetUserID bit impostato al valore 1. E supponiamo che ad un certo punto un altro utente U' sta cercando di lanciare il programma P. Succede che quando l'applicazione P parte, l'utente U' acquisisce l'utenza di U.

Quando io scrivo un comando con all'inizio la parola "sudo", io sto lanciando l'applicazione SUDO che è del ROOT del sistema, ovviamente. Quando eseguo una determinata applicazione, acquisisco TEMPORANEAMENTE L'IDENTITÀ del root, si tratta di eseguire programmi per conto di qualcun altro.

Infine abbiamo lo sticky_bit, che ci dice che se un file è una directory, se questa è marcata con sticky flag a 1, abbiamo l'impossibilità di cancellare files, se non siamo l'owner.

Se noi vogliamo visualizzare i codici numerici legati agli utenti che sono presenti all'interno del sistema, ci rechiamo su `/etc/passwd`:

```
mysql:x:60:499:MySQL database admin:/var/lib/mysql:/bin/false
nm-openconnect:x:466:468:NetworkManager user for OpenConnect:/var/lib/nm-openconnect:/sbin/nologin
nm-openvpn:x:462:464:NetworkManager user for OpenVPN:/var/lib/openvpn:/sbin/nologin
nobody:x:65534:65534:nobody:/var/lib/nobody:/bin/bash
nscd:x:477:478:User for nscd:/run/nscd:/sbin/nologin
polkitd:x:473:476:User for polkitd:/var/lib/polkit:/sbin/nologin
postfix:x:51:51:Postfix Daemon:/var/spool/postfix:/bin/false
pulse:x:467:471:PulseAudio daemon:/var/lib/pulseaudio:/sbin/nologin
root:x:0:0:root:/root:/bin/bash
rpc:x:478:65534:User for rpcbind:/var/lib/empty:/sbin/nologin
rtkit:x:470:474:RealtimeKit:/proc:/bin/false
scard:x:469:473:Smart Card Reader:/var/run/pcscd:/usr/sbin/nologin
ssdm:x:464:466:SDDM daemon:/var/lib/ssdm:/bin/false
sshd:x:472:475:SSH daemon:/var/lib/sshd:/bin/false
statd:x:474:65533:NFS statd daemon:/var/lib/nfs:/sbin/nologin
systemd-coredump:x:482:482:systemd Core Dumper:/::/sbin/nologin
systemd-network:x:480:480:systemd Network Management:/::/sbin/nologin
systemd-timesync:x:481:481:systemd Time Synchronization:/::/sbin/nologin
tftp:x:476:477:TFTP account:/srv/tftpboot:/bin/false
usbmux:x:471:65533:usbmuxd daemon:/var/lib/usbmuxd:/sbin/nologin
vnc:x:465:467:User for VNC:/var/lib/empty:/sbin/nologin
francesco:x:1000:100:Francesco Quaglia:/home/francesco:/bin/bash
pesan:x:461:462:PE-COFF signing daemon:/var/lib/pesigan:/bin/false
svn:x:460:461:User for Apache Subversion svnserv:/srv/svn:/sbin/nologin
wwwrun:x:459:458:WWW daemon apache:/var/lib/wwwrun:/sbin/nologin
user1:x:1001:1000::/home/user1:/bin/bash
user2:x:1002:1000::/home/user2:/bin/bash
/etc/passwd lines 11-37/37 (END)
```

L'utenza di Francesco, è rappresentata esattamente dal codice numerico numero 1000. Il fatto che a quel codice numerico sia associata l'utenza francesco, è registrato in questo file.

Le password sono rappresentate all'interno del file /etc/shadow

```
nobody:!:17899::::::
nsqd:!:17899::::::
polkitd:!:17899::::::
postfix:!:17899::::::
pulse:!:17899::::::
root:$6$NTmJwsbam$uqAdBUJWVF1wgsRcgjNHj5BVgLtU0luWzIFyfDREGvzWjYroIHOAdS5L05HjR4N2N9.P098ULrfz4IZ7/xUBH0:17899::::::
rpc:!:17899::::::
rtkit:!:17899::::::
scard:!:17899::::::
ssdm:!:17899::::::
sshd:!:17899::::::
statd:!:17899::::::
systemd-coredump:!::17899::::::
systemd-network:!::17899::::::
systemd-timesync:!::17899::::::
tftp:!:17899::::::
usbmux:!:17899::::::
vnc:!:17899::::::
francesco:$6$xnclUBnDrxrB$aFu62A5Jql/0BbD1mAjv1X0XLTcJSMSMQkHh5lUGNYERDVKwJG87iQyJ28Xlp5A6j1JQMKXR.Ugs32toz6Tuc0:17899:0:99999:7:::
pesan:!:17899::::::
svn:!:17899::::::
wwwrun:!:18275::::::
user1:$1$.JE5Y2q6$A7UN1PV2IRcc0BIX0GYu3/:18400:0:99999:7:::
user2:$1$pKL5Qw$0o6pu4eMzoTFZeg/RlT7k1:18401:0:99999:7:::
linux-mxb5:/home/francesco/git-web-site/FrancescoQuaglia.github.io/TEACHING/SISTEMI-OPERATIVI/CURRENT/SOFTWARE-EXAMPLES/VIRTUAL-FILE-SYSTEM/UNIX #
```

Per francesco abbiamo una rappresentazione criptata della sua password.

Ma abbiamo delle System Call per andare a fare il retrieve del codice numerico associato alla nostra utenza o per cambiare il codice numerico associato alla nostra utenza e quindi associare al thread corrente un altro codice numerico, che specifichiamo come parametro, che va ad indicare che da ora in poi questo thread sta eseguendo per conto di qualche d'un altro?

Sì!

uid_t getuid()	← Accessibile a tutte le utenze
uid_t geteuid()	← Accessibile a tutte le utenze
int setuid(uid_t)	← Accessibile a “euid” root

Questo è esattamente ciò che avviene quando noi utilizziamo un approccio di login - ossia dare delle credenziali ad una applicazione che gira col codice numerico associato al ROOT - questa applicazione controlla se queste credenziali sono correttamente registrate all'interno di quei file, se tutto è apposto quel thread che ha fatto il controllo cambia la sua USERID, e quindi comincia ad eseguire per conto di un altro utente.

ESEMPIO

```
1 #include <unistd.h>
2 #include <stdio.h>
3
4 int main(int a ,char** b){
5
6     uid_t id;
7     uid_t euid;
8
9     id = getuid();
10    euid = geteuid();
11    printf("I'm running on behalf of user %d - euid user %d\n", (int)id,(int)euid);
12    printf(.. who would you like to become? );
13    scanf("%d",&id);
14    setuid(id);
15    id = getuid();
16    euid = geteuid();
17    printf("I'm now running on behalf of user %d - euid user %d\n", (int)id,(int)euid);
18    pause();
19
20 }
```

Questo è un'applicazione molto semplice in cui qui chiamiamo il geteuid() per sapere qual è il codice numerico associato all'utente per conto di cui sta eseguendo il thread che sta girando questo programma.

Poi abbiamo l'informazione in output che ci dice che stiamo eseguendo per conto di un codice numerico. E poi però ci viene chiesto che cosa vorremmo diventare. Io vorrei diventare un altro utente con un altro codice numerico, che fornisco in input alla setuid(): cerco di settare il codice di utenza associato al thread corrente che sta eseguendo questa applicazione, esattamente al codice che mi viene passato. E poi rifaccio una query al kernel per sapere qual è il mio codice effettivo ora e lo stampo.

31:37

Ora io sto girando con l'utente 1000, come avevamo visto prima. Setuid() non è aperta per tutte le utenze, se la chiamo da francesco, essa fallisce. Rimango francesco. Per fare tutto ciò devo elevarmi a SUDO.

>>sudo su

E lancio il programma esattamente come root. Il codice di utenza associato al root è ZERO.

```
EXAMPLES/VIRTUAL-FILE-SYSTEM/UNIX/user-id # ./a.out
I'm running on behalf of user 0
.. who would you like to become? █
```

```
.. who would you like to become? 90
I'm now running on behalf of user 90
█
```

90 magari all'interno di /etc/passwd non è nemmeno presente! Non interessa al kernel, serve alle applicazioni per identificare l'utente.

Quando noi abbiamo un codice numerico associato ad un thread attivo abbiamo anche un codice che ci da l'effective user ID del thread stesso.

36:47

geteuid()

**Anche se il thread sta girando per conto dell'utente X, magari quell'effective user id associato al thread, è pari a zero.
Quindi sta girando per conto di un utente X ma temporaneamente ora è in grado di fare le cose che può fare il root.**