

Permissions

Joseph Hallett

January 12, 2023



In the beginning there was root...

And it was good.

The root user is all powerful.

- ▶ The *super user*
- ▶ The *system administor*
- ▶ UID 0

And so root beget init which beget login...

And so other users came to pass

- ▶ Each less powerful than the original root



For inside the password file...

In the bowels of the computer's configuration directory /etc/:

```
$ grep -Ev '^_' /etc/passwd | column -ts :
```

Username	Password	UID	GID	GECOS	Home Directory	Shell
root	*	0	0	Charlie &	/root	/bin/ksh
daemon	*	1	1	The devil himself	/root	/sbin/nologin
operator	*	2	5	System &	/operator	/sbin/nologin
bin	*	3	7	Binaries Commands and Source	/	/sbin/nologin
build	*	21	21	base and xenocara build	/var/empty	/bin/ksh
sshd	*	27	27	sshd privsep	/var/empty	/sbin/nologin
www	*	67	67	HTTP Server	/var/www	/sbin/nologin
nobody	*	32767	32767	Unprivileged user	/nonexistent	/sbin/nologin
joseph	*	1000	1000	Joseph Hallett,,,	/home/joseph	/usr/local/bin/bash

See man 5 passwd or your OS's manual pages.

(Can anyone spot what OS I use?)

And inside the group file....

```
$ grep -Ev '^_' /etc/group | column -ts :
```

Groupname	Password	GID	Members
wheel	*	0	root,joseph
daemon	*	1	daemon
kmem	*	2	root
sys	*	3	root
tty	*	4	root
operator	*	5	root
bin	*	7	
wsrc	*	9	joseph
users	*	10	
auth	*	11	
games	*	13	
staff	*	20	root,joseph
wobj	*	21	joseph
sshd	*	27	
guest	*	31	root
utmp	*	45	
crontab	*	66	
www	*	67	
network	*	69	
authpf	*	72	
dialer	*	117	
nogroup	*	32766	
nobody	*	32767	
joseph	*	1000	

Something very similar

- ▶ Each group can have *multiple* members
- ▶ No passwords ever actually listed
 - ▶ (They're in `/etc/shadow`)

For all files were owned by a user and a group...

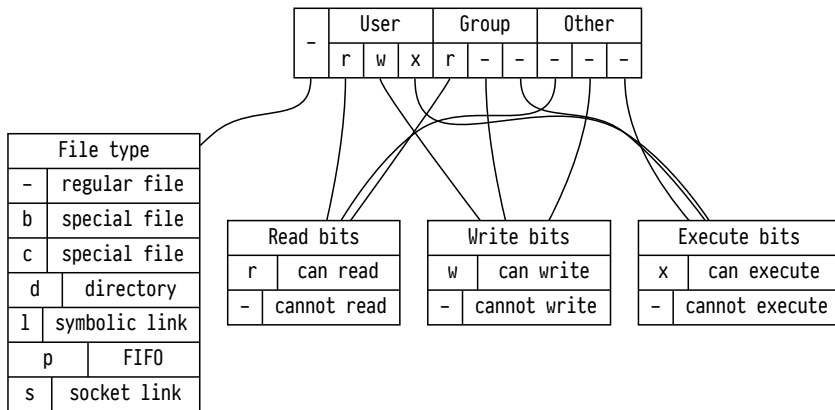
```
ls -l /etc/
```

Permissions		UID	GID	File flags	Size	Filename	
drwxr-xr-x	5	root	wheel	-	512B	May 20 2022	ConsoleKit
drwxr-xr-x	2	root	wheel	-	512B	Nov 25 13:25	ImageMagick
drwxr-xr-x	7	root	wheel	-	512B	Nov 16 20:19	X11
-rw-r--r--	1	root	wheel	-	20.5K	Nov 6 12:41	abcde.conf
drwx-----	2	root	wheel	-	512B	Nov 16 19:39	acme
-rw-r--r--	1	root	wheel	-	1.7K	Sep 22 19:03	adduser.conf
drwxr-xr-x	2	root	wheel	-	512B	Nov 16 19:39	amd
-rw-r--r--	1	root	wheel	-	271B	Oct 30 19:14	anthy-conf
drwxr-xr-x	3	root	wheel	-	512B	Nov 25 13:27	apache2
-rw-r--r--	1	root	wheel	-	1.8K	Nov 14 10:34	authentication_milter.json

...

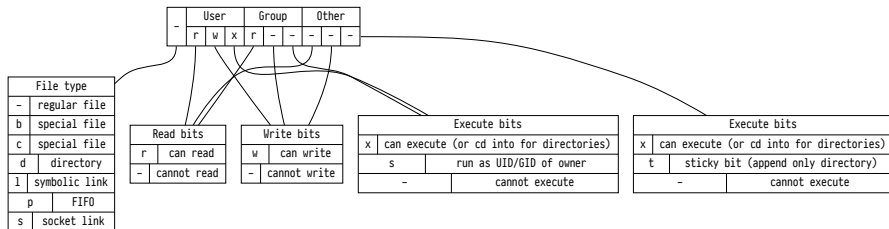
UNIX Discretionary Access Controls

And the owner of each file could set the *permissions* for each file



Actually its a *bit* more complex

And the owner of each file could set the *permissions* for each file



And, honestly, on some systems/filesystems it gets *even more* complex

- ▶ But this is 99.99% of everything you'll ever see or use

So what are those weird extra bits for

The sticky bit `t` is mostly for log directories and temporary directories

- ▶ You should be able to append to log files, but not delete them

The `setuid/setgid` bits are used for privilege separation.

For example how do you update your password?

Passwords are normally stored securely in the shadow file `/etc/shadow`, or equivalent

- ▶ But I use OpenBSD...

```
ls -l /etc/spwd.db
```

```
-rw-r-- 1 root _shadow 40960 Dec 22 15:03 /etc/spwd.db
```


Changing passwords

The passwd program changes your password:

```
ls -l $(command -v passwd)
```

```
-r-sr-xr-x 1 root bin 21208 Jan 12 03:08 /usr/bin/passwd
```

Other useful setuid programs

su switch to user (by default root) with *their* password

sudo switch to user if the sysadmin says you're allowed to with your password

doas modern rewrite of sudo with less bugs and Spiderman references

See `man su` or `man sudo` or `man doas`...

- ▶ Or Michael W. Lucas's excellent *Sudo Mastery*
- ▶ (You can do a lot with sudo...)

Generally setuid programs are dangerous and you want to use them extremely carefully!

Sysadmining

How do you change who owns a file?

```
ls -l exam
```

```
-rw-r--r-- 1 joseph joseph 0 Jan 12 11:49 exam
```

```
chown joseph:staff exam
```

```
# Alternatively...
```

```
chown :staff exam
```

```
ls -l exam
```

```
-rw-r--r-- 1 joseph staff 0 Jan 12 11:49 exam
```

```
(See man 1 chown)
```

How do you change a file's permissions

```
chmod go-wx exam
```

```
ls -l exam
```

```
-rw-r--r-- 1 joseph staff 0 Jan 12 11:49 exam
```

Footnote

Some people like to use octal (base 8) to express permissions, where $r=4$, $w=2$, $x=1$...

Instead of saying `go-wx` to remove `w` and `x` bits from the group and other permissions they'll say:

```
chmod 744 exam
```

I suggest you give these people a wide berth.

- (but you should know how to do it)

Recap

Systems have users!

- ▶ The UNIX DAC lets you set file permissions!
- ▶ `setuid` and `setgid` programs exist!
- ▶ Root's firstname is *Charlie*!

`chmod` to change permissions

`chown` to change file owners

One more thing...

Traditionally the root user can do anything...

In most modern operating systems this has been split up a bit more

- ▶ For example Linux uses *capabilities* to set what things any user can do
- ▶ ...and *namespaces* to allow multiple root users with different capabilities

man 7 capabilities if you want to know more

- ▶ ...but *most of the time* you won't need to know about them...
- ▶ Unless you use Docker...

This is a lie, you really *should* know about them... but unless you're routinely in the habit of writing sysadmin tools or privileged programs you *won't normally* need to touch them. Hey, I'm a security researcher I think this stuff is fascinating but other people don't. Don't get it meself: it isn't *that* complex but hey ho. I tried.