

# File Protection

- The first level of protection is that only valid users are allowed to login to the system.
- The system administrator, called the **superuser** with user name `root`, sets up the valid users with the `adduser` (or equivalent) command.
- If you don't know a valid user name, you cannot login.
- To see a list of all valid user names: `cat /etc/passwd`
- A typical, traditional user entry looks like

```
user1:abFgLmKx:101:50:J Random Hacker:/usr/user1:/bin/sh
```

user name   encrypted password   user ID   group ID   comment   login directory   login shell

- Each user is given a unique user ID (called the **uid**), and is put into a group.
- Each group has a unique group ID (called the **gid**); group information is stored in the `/etc/group` file.

## Password

- Since `/etc/passwd` is usually allowed to be seen by everyone, anyone could log in as you, if they guess your password.
- To prevent this, give yourself a password:

```
passwd
```

```
old password:
```

```
new password:
```

```
retype password:
```

*Note: Passwords will not be echoed.*

- Now no one can login as you, unless they know your password.

- Your password is encrypted, and, traditionally, entered into */etc/passwd*.
- Newer versions of UNIX place passwords into another file (such as */etc/shadow*, */etc/master.passwd*, or */etc/secure*), that only the superuser has access to. In such a system, your password entry in */etc/passwd* is replaced by a \* or x character.

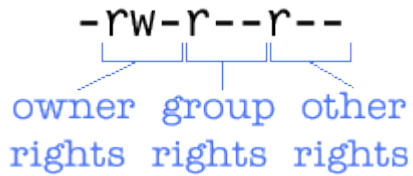
## File Security

- The following information is kept on each file that exists. It is stored on a special part of the disk called the **i-list** or **i-table**.
  - The user ID and group ID of its owner.
  - Its protection bits (twelve in total).
  - The physical disk or tape address of the file contents.
  - Its size in bytes.
  - Time of creation, last use, and last modification. Time is stored as the number of seconds since midnight, January 1, 1970.
  - The number of links to a file, that is, the number of times it appears in a directory.
  - A code indicating whether the file is a directory, an ordinary file, or a special file.
- To get the most information about your files:

```
ls -l
-rw-r----- 1 user1 group 116 nov  9 16:17 .profile
drwxr-x---  2 user2 group  64 nov 14 16:30 dir1
```

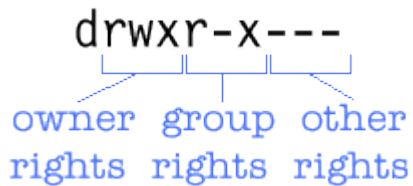
file type      protection bits      owner number of links      group name      length in bytes      date last modified      time      filename

## Protection Bits for an Ordinary File



Bit Being Set	Allow
r	Read
w	Write
x	Execute
-	Permission denied

## Protection Bits for a Directory



Bit Being Set	Allow
r	Listing directory contents
w	Adding and removing entries in the directory
x	Searching through the directory; changing in to the directory
-	Permission denied

- When you create a new file, it typically gets these protection bits by default:

`-rw-r--r--`

- When you create a new directory, it typically gets these protection bits by default:

`drwxr-xr-x`

- The default can be changed.
- *Note:* Members of your group are not considered part of "the world" (others); this enables you to restrict the permissions for members of your group.

## Changing File Protection

- Use the `chmod` command.
- This command can only be executed by the owner of the file or by root.
- You can change the protection for any of the three categories:
  - owner - (u)
  - group - (g)
  - other - (o)
- You can add or take away permissions
  - + [r][w][x] - Grant permission
  - - [r][w][x] - Revoke permission
  - = [r][w][x] - Set permissions absolutely

**chmod Command**

If a file has initial rights	<pre> -rw-r-----           u g o </pre>
To allow others than those in your group to read the file	<i>chmod o+r file.txt</i>
Now the access mask is	<pre> -rw-r--r--           u g o </pre>
To give everyone execute rights	<i>chmod +x file.txt</i>
If a file has initial rights	<code>-rwxr-xr-x</code>
Execute command	<i>chmod o-rx file</i>
Now the access mask looks like	<code>-rwxr-x---</code>
If you want to protect your file from accidentally being deleted by yourself	<i>chmod u-w file</i>
Many UNIX gurus find it convenient to use a less user-friendly form of <i>chmod</i> , using <b>octal</b> values for protection bits to set the permissions absolutely.	
Given that the protection bits look like this:	<pre> -rw-r-----           421421421           6 4 0 </pre>
To give others read permission	<i>chmod 644 file.txt</i>
Now the access mask looks like this	<pre> -rw-r--r--           6 4 4 </pre>

## Advanced Protection Bits

- You must use this form of the *chmod* command to set the advanced bits

```
chmod xyyy file.txt
```

where *yyy* are the user, group, and others digits as previously discussed.

- Bit "x" has the following meanings

Value of x	Meaning
4	Set user ID on execution ( <b>setuid</b> )
2	Set group ID on execution ( <b>setgid</b> )
1	Set "sticky" bit
0	Advanced permissions denied

- If you set file mode to *4xxx*
  - When this file is executed by those allowed by the permission bits, their user ID will temporarily be changed to that of the owner of the program.
  - This allows you to write a program that manipulates files on people's behalf, but never lets them get at those files on their own.
  - When you set advanced bit 4, the file protections appear with an s in the third owner field, where the x would normally go:

```
-rws--x--x
```

- For example, the permission bits for file */etc/passwd* are *-r--r--r--*; the */bin/passwd* command has permission bits *-rws--x--x*. Both are owned by *root*. When you change your password with */bin/passwd*, the *set user ID* mode is put into effect.
- Such programs are referred to as **setuid** or **suid** programs, and can pose a potential security threat if misused.

## File Ownership

- When you create a file, it is stamped with your user ID and group ID.
- You and the superuser are the only users allowed to change the mode (*viachmod*).
- Only the superuser can change the owner ID and group ID of a file.

```
chown username FILE.TXT
```

```
chgrp groupname FILE.TXT
```

## Further Protection

- Permission bits allow you to keep other users out of your files, but not the superuser (*root*).
- For protection of very sensitive files, you can encrypt them.
- To encrypt *MYFILE*: *crypt key < MYFILE > encryptedfile*
- To decrypt *encryptedfile*: *crypt key < encryptedfile > MYFILE*
- Software with better encryption exists today. For example, PGP ("Pretty Good Privacy") and GNU PG ("Privacy Guard").
- ***If you forget the key, you can forget the file!***