How Permissions Work

Special Notices

Information in this document is correct to the best of our knowledge at the time of this writing. Please send feedback by fax to "AIXServ Information" at (512) 823-4009.

Please use this information with care. IBM will not be responsible for damages of any kind resulting from its use. The use of this information is the sole responsibility of the customer and depends on the customer's ability to evaluate and integrate this information into the customer's operational environment.

About This Document

This is how AIX controls access to files and directories. All files and directories have permissions for:

Owner (usually who created the file)

Group (anyone that has this group in their group set or as a primary group)

Other (any user that is not the owner or doesn't have this group)

How do I see the permissions on a file? 1
Which set of permissions apply to me? 2
What do the permission letters mean? 2
For a directory: 2
For a file: 3
What Determines The Permissions On A File Or Directory When I Create It? 3
How Do I Change The Owner Or Group On A File Or Directory? 4
How do I change the permissions on a file or directory? 4
What are Access Control Lists (acl's)? 5
Filesystems and the underlying mount point permissions 7
Permissions on NFS mounted filesystems 7
Ftp Doesn't Use My Umask Setting When Creating A File? 8
Reader's Comments 9

How do I see the permissions on a file?

The ls -l filename or ls -ld directory_name will show a string of 10 characters on the left:

```
-rwxrwxrwx
```

This character string can be broken up into the three permission sets:

```
-rwx rwx rwx
^   ^   ^
|   |   | Permissions for others
|   |
|   | Permission for users that have this group as
|     their primary group or as part of their group set.
|
| Permissions for the owner of the file or directory
```

This command also shows a owner and group for the file or directory:

```
-rwxrwxrwx [some number] joe joegroup
    ^                    ^     ^
    permissions          owner group
```

The displayed name for the owner is taken from /etc/passwd. The file inode store the user id (a number). If you see a number instead of a name, either the /etc/passwd file is not readable or that user id is missing from the file. The name of the group is taken from /etc/group. Again, only a number is stored in the files inode.

Which set of permissions apply to me?

The system first checks to see if you are the owner, if so, you get the owners permissions, even if these permissions are more restrictive that the "group" or "other" set.

Then the system checks to see if you have the group listed as either your primary group or as part of your group set. If so, you get the group permissions, even if these permissions are more restrictive than the "other" set.

If you are not the owner or part of that group, you get the permissions for other.

What do the permission letters mean?

The meaning of permissions differ for a file or a directory. Permissions to read and write in a directory (a directory is just a file with a list of name and pointers) are independent of permissions to read and write the files pointed at by these entries.

The relationship of the directory to a file is:

```
directory --> inode --> data
|             |         | actual user data
|             |
|             | contains the permissions for the file and a
|             | pointer to where on disk the data resides.
|
| contains a list of the filenames and the pointers to the
| inodes that contain the permissions for the file.
```

Directory permissions are required to find a file before opening it for use.

For a directory:

```
r   Ability to read the list of files in a directory
such as with the ls command.
```

NOTE: You cannot execute a command inside the directory if the only permissions you have is to read the directory listing.

```
x    Ability to pass through the directory (search).
This allows you operate on files in the directory if you
already know the filenames.  Without read permission, you
cannot read the entries but are allowed to access them.
To use a file the user must have search permission for ALL
directories in the path. They must also have appropriate
permission for the file itself.
```

In general, both read (r) and search (x) are required for most directory operations.

```
w    Ability to create or delete files in the
directory when coupled with the search (x) permission.
If you have permissions to write at the directory level
but no permissions on the file inside, you may still delete
the file even though you may not modify the file. If you
have only write permissions, you can not change or modify
the file.
```

NOTE: If you have write and search permissions at the directory level, you may delete the directory entry unless the link permissions bit is set on the directory.

```
t    Link permission. This bit when set prevents any
user except the owner of a file, root or the owner of
the directory  from deleting the file, even though the
permissions on the directory would otherwise allow it.
This also means the search (x) bit is set for other.
T    Same as lowercase t but the search (x) bit is
not set for other.
s    Not used in the owner position. For the group
position, this becomes the group inheritance bit (SGID).
Any files created in this directory have the same group
as the directory entry. In AIX, when a new filesystem is
created, this bit is set by default. This also means the
search (x) bit is set for the group.
S    Same as the lowercase s but means the search
(x) permission is not set for the group.
```

For a file:

```
r    ability to read the file.
w    ability to change/modify  a file. Cannot
create or delete it unless you have write permission for
the directory.
x    ability to execute the file.
```

What Determines The Permissions On A File Or Directory When I Create It?

Your user id (uid) is used to set the owner of the file. Your primary group is used to set the group (unless the directory the file is created in has the SGID permission bit set. Then the group is the same as the group on the directory entry.) The id command can be used to see the uid and primary group for that user. Your umask setting is used to set the initial permissions. Type umask at the command line to see the current setting. To change the umask do: umask number, like: umask 022.

NOTE: You do not use the equal (=) sign when setting umask.

Umask does not set the execute bit on a text file or shell script, only on a executable program (like from a compiler) or rectory. To determine what umask to use, take the umask setting and subtract it from 777 for a directory or compiled program or subtract from 666 for a text file or shell script.

```
Examples:  Directory/Program              File
umask 022
                 777                      666
               - 022                    - 022
                 ---                      ---
       result    755                      644
```

ls -l shows

```
rwxrw-rw-                rw-r--r--
```

How Do I Change The Owner Or Group On A File Or Directory?

The chown and chgrp commands are used to change owner and group. Only the root user can change the owner of a file. You can change the group of a file only if you are a root user or if you own the file. If you own the file but are not a root user, you can change the group only to a group of which you are a member. The chown command can be used to change the owner and group at the same time.

To use the chgrp command:

```
chgrp new_group_name file_name
```

To use the chown command:

```
chown new_owner filename,   or
chown new_owner.new_group filename
```

How do I change the permissions on a file or directory?

The chmod command is used to change the permissions on a file or directory. Permissions can be expressed either using letters or numbers.

CAUTION: If you are using acl's on a file or directory, using the numeric mode of the chmod command will DISABLE the acl.

The following general sytax for numeric mode should be run on one line as:

```
chmod
[special bits][owner][group]
[other]
```

The numeric mode uses numbers for each position, owner, group and other. For each position:

```
read (r) = 4
write (w) = 2
execute [or search] (x) = 1
```

These numbers are added together to get the numbers to use for the chmod command. For example:

```
owner has read + write + execute = 4+2+1 = 7
group has read + write           = 4+2   = 6
other has read                   = 4     = 4
```

The chmod command would be: chmod 764 filename

To set the special bits, suid, sgid and link bits requires a 4th number for the chmod command.

```
suid = 4
sgid = 2
link = 1
```

To add the suid (set user id) permissions to the above example: chmod 4764 filename

To use the symbolic mode, you specify which position owner (u), group (g), other (o) or all (a) to change and what symbol to add (+) or subtract (-). To add read + write to the owner use both:

```
chmod u+r u+w filename or  chmod u+rw filename
```

If the sgid (set group id) bit is set on a directory, the symbolic mode must be used to unset it:

```
chmod g-s filename
```

The numeric mode will not produce a error but the sgid bit will still be set.

For further information on the chmod command, see InfoExplorer or the man page for chmod.

What are Access Control Lists (acl's)?

ACL's are a extension to standard permission bits. They allow "fine grain" control for each file or directory by modifying the standard permissions that are assigned to individuals or groups. There are three assignments you can make for each group or user:

permit Grants the specified access to the file or directory.

deny Restricts the specified access to the file or directory.

specify Precisely defines the file or directory access.

NOTE: If a user is denied a particular access by either a deny or a specify keyword, no other entry can override that access denial.

The "acledit" command is used to create an acl. First you must set the EDITOR environment variable with the full path to your favorite text editor like:

```
export EDITOR=/usr/bin/vi.
```

then use:

```
acledit filename
```

this will bring up a screen like:

```
attributes:
base permissions
owner(rcunning):  rwx
group(staff):  r--
others:  ---
extended permissions
disabled
```

To set the extended permissions, change the "disabled" word to "enabled" like:

```
extended permissions
enabled
```

Use the permit, deny or specify keywords to define the extended permissions. The above example shows that only the owner can write to this file. The group staff can read the file and other has no permissions. To allow user joe to read and write the file, use:

```
extended permissions
enabled
permit   rw-   u:joe
```

To allow group joegroup to read the file, use:

```
permit   r--   g:joegroup
```

You can fine tune the permissions by combining the multiple entries on the same line. If you want to allow pete read and write access ONLY if he is currently part of the system group, use:

```
permit   rw-    u:pete, g:system
```

To add permissions for several users or group, seperate lines are used like:

```
permit   rw-   u:joe
permit   rw-   u:pete
```

Further information on the "acledit command" can be found in InfoExplorer or the man pages.

Two other acl commands can be used, alcget and aclput:

- Aclget filename: will output the permissions to standard out.
- Aclget filename<file_with_permissions: will set acl's on a file.

To copy the acl's from one file to another, you can use:

```
aclget filename | aclput new_filename
```

The "ls -el command" can be used to see if a acl is set on a file. "ls -el .profile" shows:

```
-rwxr-----+  the + in the last position means the file has
a acl enabled.
```

CAUTION: Using the chmod command with numeric arguments will disable the acl for a file or directory.

Filesystems and the underlying mount point permissions

Filesystems are mounted over a directory mount point. Both the mount point and the filesystem have permissions. The permissions on the mount point are determined by the current umask setting at the time the filesystem is created. Even thought the permissions on the mounted filesystem takes precedence over the permissions on the mount point, the mount point must have at least search permissions for everyone, ie: 111, to avoid unpredictable results. Current versions of AIX will warn the user if the permissions on the mount point and the filesystem is not the same. Remember, after the filesystem is mounted, you cannot see the permissions on the mount point. You must umount filesystem before the mount point permissions can be examined and changed.

When a filesystem is created, the default permissions come from a proto file (see mkfs in InfoExplorer for more details) and the SGID (group inheritance) bit

is set. The users umask is not involved for filesystems, just for the underlying mount point.

Permissions on NFS mounted filesystems

Network File Systems (NFS) are mounted on a local directory. The directory mount point is normally created when you first create the network mounted filesystem in smit. The mount point uses the current umask setting to determine the permissions when it is created. In newer versions of AIX, the mount point for nfs filesystems will always be at least 111 to insure proper operation.

NFS mounted filesystems use a special user id called nobody. This UID is normally a very large number so as not to conflict with any real user id. Unless the nfs server has a entry in /etc/passwd for your user id (not text name) the permissions you have when you remote mount a filesystem is for the pseudo user id "nobody". If your user id happens to match some valid id on the remote system, you are now the owner of any files that owner created. This can lead to unpredictable results. If you need to be able to create and own files on the remote system, your local system and the server system must have matching user entries (both user name and the same id number) in the /etc/passwd files. The filesystem must also be exported as read/write.

The root user is a special case. Since root is user id 0 on all systems, without special protection any system that mounted the filesystem would become root on the server system. Because of this, the nfs filesystem must be exported with root access for any hostnames that you require special root access from. If the filesystem is not exported in this manner then root becomes user nobody and has the permissions of the "other" group.

Ftp Doesn't Use My Umask Setting When Creating A File?

The ftp command does not use the users umask setting. Use the ftp "site umask x" subcommand to change the ftp umask.

Reader's Comments

Please fax this form to (512) 823-4009, attention "AIXServ Information". You may also e-mail comments to: elizabet@austin.ibm.com. These comments should include the same customer information requested below.

Use this form to tell us what you think about this document. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

If you need technical assistance, contact your local branch office, point of sale, or 1-800-CALL-AIX (for information about support offerings). These services may be billable. Faxes on a variety of subjects may be ordered free of charge from 1-800-IBM-4FAX. Outside the U.S. call 415-855-4329 using a fax machine phone.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Note: If you have a problem report or item number, supplying that number may help us determine why a procedure did or did not work in your specific situation.

Problem Report or Item #: Branch Office or Customer #:

Be sure to print your name and fax number below if you would like a reply:

Name: Fax Number:

END OF DOCUMENT (How.perms.work.cmd, 4FAX# 5888)