Geek Thoughts

Random thoughts and ideas from a French geek in London

Thursday, 29 April 2010

Shared Folders in Ubuntu with setgid and ACL

Introduction

There is an often requested feature on Linux (or UNIX) to have the ability to create shared directories similar to what is possible in Windows, that is a directory in which every person who has been given access can read, write or modify files. However, because Linux file systems such as ext4 enforce file permissions that are stricter than any of the windows file systems such as FAT or NTFS, creating such a directory is not obvious. Of course, if you put your shared directory on a FAT or NTFS partition, it will automatically behave just like in Windows but that requires a separate partition and doesn't allow you to enforce permissions on a per-group basis. So here's a quick guide on how to do this with Ubuntu. The same principles apply to other Linux distributions so should be portable.

Use Cases

Let's go through a couple of classic use cases first, to identify

Followers

Friends

- Suddenly single at 32.....
- Coofer Cat

exactly what we want to do.

Project Folder

In a company or university setting where users are assigned to project teams or departments, it can be useful to create shared folders where all members of the team can drop files that are useful for the whole team. They need to be able to create, update, delete files, all in the same folder. They also need to be able to read, update or delete files created by other members of the team. However, users external to the team should only have read access.

Web Development

For anybody doing web development on Linux, a classic problem is when you have to deal with development or test web servers. The default web server process runs with the `www-data` user and the document directory is owned by the same user. It would be great if all web developers on the team were able to update the document directory on the server while not requiring root access to do so.

Linux Default Behaviour

Linux has the concept of user groups. You can check what groups your user belongs to by typing the following on the command line:

```
$ groups
bruno adm dialout cdrom plugdev lpadmin admin sambashare
```

On a default Linux installation, groups are used to give access to specific features to different users, such as the ability to administer the system or use the CD-ROM drive. But one of the core feature of user groups is to support file permissions. Each file has separate

- I got 99 problems
- Un Souvenir
- MaYa D on life
- Only Somewhat Boring

More About Me

- My photos on flickr
- My photos on DHD Multimedia Gallery
- Music I listen to
- Travels

Blog Archive

- ▶ 2011 (2)
- ▼ 2010 (25)
  - ▶ December (1)
  - ▶ November (1)
  - ▶ October (1)
  - ▶ September (5)
  - ▶ August (4)
  - ▶ July (2)
  - ▶ May (2)
  - ▼ April (2)
    - Shared Folders in Ubuntu with setgid and ACL
    - Ubuntu Lucid Netbook Remix from Alternate CD
  - ▶ March (3)
  - ▶ February (3)
  - ▶ January (1)
- ▶ 2009 (26)

sets of read, write and execute permissions for the user who is the owner of the file, the group that owns the file and *others*, that is everybody else. Whenever a user attempts to read, write or execute a file, the system will decide whether he can do it based on the following rules:

- if the user is the owner of the file, user permissions apply,
- otherwise, if the user is part of the group that owns the file, group permissions apply,
- otherwise, *others* permissions apply.

So to configure a shared directory as defined above, we need to:

- create a user group for the team,
- assign all team member users to that user group,
- create a directory and configure it so that all users in the group can:
  - add new files to the directory,
  - modify any existing file in the directory,
- and of course, all this should work without users having to do anything special.

How To

Enable ACL

The first thing we need to do is to enable ACL support on the partition where we will create the shared directory. ACL extend the basic Linux permission implementation to enable more fine grained control. As this requires the file system to be able to store more permission meta-data against files, it needs to be configured accordingly. We can do this by adding the `acl` option to the relevant line in `/etc/fstab`, such as:

► 2008 (31)
► 2007 (53)
► 2006 (139)
► 2005 (168)
► 2004 (98)

Tags

- linux (33)
- technology (31)
- quirky (30)
- rants (23)
- ubuntu (22)
- development (15)
- howto (15)
- tips (14)
- photography (13)
- news (12)
- web (12)
- software (11)
- travel (11)
- graphics (9)
- fractals (8)
- mandelbrot (8)
- os-x (8)
- images (7)
- julia (7)
- maths (7)
- octave (7)
- food (6)
- thoughts (6)
- mobile (5)
- apple (4)
- canon (4)
- charity (4)
- firmware (4)

```
UUID=b8c490d0-0547-4e1f-b052-7130bacfd936 /home ext4 defaults,acl 0
```

The partition then needs to be re-mounted. If the partition to re-mount is `/`, `/usr` or `/home`, you will probably need to restart the machine. Otherwise, the following commands should re-mount the partition:

```
$ sudo umount partition
$ sudo mount partition
```

where *partition* is the mount point of the partition as defined in `/etc/fstab`, such as `/var/www`.

Create Group

We then need to create the group to which we will give shared access, let's call that group *teamgroup*:

```
$ sudo groupadd teamgroup
```

Try to give the group a meaningful name while keeping it short. If it's meant to be a team group, give it the name of the team, such as *marketing*. Note the following restrictions on Debian and Ubuntu for group names (taken from the man page):

> It is usually recommended to only use groupnames that begin with a lower case letter or an underscore, followed by lower case letters, digits, underscores, or dashes. They can end with a dollar sign. In regular expression terms: [a-z_][a-z0-9_-]*[$]?
>
> On Debian, the only constraints are that groupnames must neither start with a dash (-) nor contain a colon (:) or a whitespace (space: , end of line: \n, tabulation: \t, etc.).

Groupnames may only be up to 32 characters long.

We then need to assign users to that group:

```
$ sudo usermod -a -G teamgroup teamuser
```

Where *teamuser* is the login name of the user to assign to the group. This assignment will take effect next time the user logs in. Make sure that you do not forget the `-a` option otherwise you will wipe out all existing group assignment for that user, rather than just adding a new one.

Create the Folder

The next step is to create the shared folder. This is easy:

```
$ cd /path/to/parent
$ mkdir teamfolder
```

Where */path/to/parent* is the path to the parent folder and *teamfolder* is the name of the folder you want to create. We then assign group ownership of the folder to the group previously created:

```
$ chgrp teamgroup teamfolder
```

And give write access to the group on that folder:

```
$ chmod g+w teamfolder
```

Let's check what this folder looks like:

```
$ ls -l
drwxrwxr-x 2 teamuser teamgroup 4096 2010-03-03 14:32 teamfolder
```


- imagemagick (2)
- latex (2)
- lucid (2)
- maps (2)
- maverick (2)
- meter (2)
- music (2)
- peru (2)
- php (2)
- privacy (2)
- python (2)
- security (2)
- skype (2)
- sound (2)
- subversion (2)
- support (2)
- svg (2)
- vala (2)
- vegetarian (2)
- wi-fi (2)
- wireless (2)
- workplace (2)
- National Grid (1)
- Siemens (1)
- accessibility (1)
- acl (1)
- activism (1)
- alternate (1)
- asus (1)
- banking (1)
- beer (1)
- blog (1)
- books (1)
- bsi (1)
- bugs (1)
- calibre (1)
- cern (1)
- change (1)
- climate (1)
- colour (1)
- courier (1)

Now, let's try to create a new file in that directory:

```
$ touch teamfolder/test1
$ ls -l teamfolder
-rw-r--r--  1 teamuser teamuser 5129 2010-03-03 14:34 test1
```

That looks good and any other user who is part of *teamgroup* should be able to create files in this directory. However, group members will not be able to update files created by other members of the group for the following reasons:

- the group that owns the file is the user's primary group, rather than *teamgroup*,
- the file's permissions only allow the owner of the file to update it, not the group.

Set the `setgid` Bit

We'll solve the first problem by setting the `setgid` bit on the folder. Setting this permission means that all files created in the folder will inherit the group of the folder rather than the primary group of the user who creates the file.

```
$ chmod g+s teamfolder
$ ls -l
drwxrwsr-x 2 teamuser teamgroup 4096 2010-03-03 14:32 folder
```

Note the `s` in the group permissions instead of the `x` that was there previously. So now let's try to create another test file.

```
$ touch teamfolder/test2
$ ls -l teamfolder
-rw-r--r--  1 teamuser teamuser  5129 2010-03-03 14:34 test1
```

- courier (1)
- css (1)
- dbus (1)
- document (1)
- ebook (1)
- eee (1)
- energy efficiency (1)
- error (1)
- eurostar (1)
- fairtrade (1)
- fancyhdr (1)
- firefox (1)
- footer (1)
- frameworks (1)
- friends (1)
- furniture (1)
- gears (1)
- geekthoughts (1)
- github (1)
- gmail (1)
- gnome (1)
- gnuplot (1)
- government (1)
- guadec (1)
- gutsy (1)
- hardy (1)
- hlc (1)
- housing (1)
- ifthen (1)
- installation (1)
- interface (1)
- jaunty (1)
- java (1)
- junit (1)
- kml (1)
- laptop (1)
- lastpage (1)
- law (1)
- leopard (1)
- letter (1)
- linkedin (1)

```
-rw-r--r--  1 teamuser teamgroup 5129 2010-03-03 14:35 test2
```

So now whenever a file is created in the team directory, it inherits the team's group.

Set Default ACL

The second issue is related to `umask`, the default mask applied when creating files and directories. By default `umask` is set to the octal value 0022, as demonstrated if you run the following:

```
$ umask
0022
```

This is a negative mask that is applied to the octal permission value of every file or directory created by the user. By default, a file is created with permissions `rw-rw-rw-`, equivalent to 0666 in octal and a directory is created with permissions `rwxrwxrwx`, equivalent to 0777 in octal. `umask` is then subtracted from that default to give the effective permission with which files and directories are created. So for a file, 0666-0022 gives 0644, equivalent to `rw-r--r--` and for a directory 0777-0022 gives 0755, equivalent to `rwxr-xr-x`. This default is sensible for most situations but needs to be overriden for a team directory. The way to do this is to assign specific ACL entries to the team directory. The first thing to do is to install the `acl` package to obtain the necessary command line tools. Well, in fact, the first thing to do would be to enable `acl` on the relevant partition but we already did that at the very beginning.

```
$ sudo apt-get install acl
```

Now that the package is installed, we have access to the `setfacl` and `getfacl` commands. The first one sets ACLs, the second one

- linkedin (1)
- links (1)
- local (1)
- maemo (1)
- management (1)
- medibuntu (1)
- movies (1)
- n900 (1)
- netbook (1)
- open source (1)
- openerp (1)
- openoffice (1)
- patents (1)
- phishing (1)
- photoshop (1)
- plastic (1)
- praise (1)
- pulseaudio (1)
- pygraph (1)
- science (1)
- scripting (1)
- search (1)
- sony (1)
- space (1)
- spam (1)
- sports (1)
- ssh (1)
- standard (1)
- tennis (1)
- testing (1)
- train (1)
- typography (1)
- unix (1)
- upgrade (1)
- usa (1)
- usability (1)
- viruses (1)
- webcam (1)
- webdav (1)
- wikipedia (1)
- xmas (1)

and `getfacl` commands. The first one sets ACLs, the second one reads them. In this particular case, we need to set *default* ACLs on the team folder so that those ACLs are applied to files created inside the directory rather than the directory itself. The syntax is a bit complicated: the `-d` option specifies that we want to impact the default ACLs, while the `-m` option specifies that we want to modify the ACLs and expects an ACL specification to follow.

```
$ setfacl -d -m u::rwx,g::rwx,o::r-x teamfolder
$ touch teamfolder/test3
-rw-r--r--  1 teamuser teamuser  5129 2010-03-03 14:34 test1
-rw-r--r--  1 teamuser teamgroup 5129 2010-03-03 14:35 test2
-rw-rw-r--  1 teamuser teamgroup 5129 2010-03-03 14:36 test3
```

There we go, it all works as expected: new files created in the team folder are created with the team's group and are group writeable. To finish off, let's have a look at how the folder's ACLs are stored:

```
$ getfacl teamfolder
# file: teamfolder
# owner: teamuser
# group: teamgroup
user::rwx
group::rwx
other::r-x
default:user:rwx
default:group:rwx
default:other:r-x
```

Granting and Revoking Access

Granting a user write access to the team folder is now extremely easy: you can just add that user from the team's group when he

xmas (1)
zend (1)

joins the team:

```
$ sudo usermod -a -G teamgroup joiner
```

Where *joiner* is the user ID of the user joining the team. Revoking access is nearly as easy, you just need to remove the user from the team's group. Unfortunately, there is no way to do this in a simple command so you will have to edit the file `/etc/group`, find the group and remove the user ID from that group.

Variations

Restrict Delete and Rename to Owner

By default, any user who has write access to a file can delete or rename it. This means that any member of the team can delete or rename any file created by another member. This is generally OK but if it is not, it can also be restricted by setting the *sticky bit* on the directory:

```
$ chmod +t teamfolder
$ ls -l
drwxrwsr-t 2 teamuser teamgroup 4096 2010-03-03 14:32
```

This feature is used on the `/tmp` directory to ensure that all files created in that directory can only be deleted by their owners.

Restrict Access for Others

Another variation that may be more useful is to completely deny access for users that are not part of the team. it may be that a particular team is working on some sensitive stuff and you don't want anybody outside the team to see it. To do this, we just revoke

all permissions and ACLs for *others* on the team folder:

```
$ chmod o-rx teamfolder
$ setfacl -d -m o::--- teamfolder
```

References

- Howto: Linux Add User To Group
- Using ACLs with Fedora Core 2
- `setuid` (and `setgid`) on Wikipedia
- Sticky bit on Wikipedia

Posted by Bruno at 14:17

Labels: acl, howto, linux, ubuntu

8 comments:

Rick said...

Very helpful, just what I've been looking for, thanks.
06 June, 2010 23:27

Anonymous said...

I've used this method in the past, but I was curious if you have a solution for when a user creates a new file under the shared directory and their local umask removes the group write permissions.

Is there anyway to force the group permission to propagate downward?
24 June, 2010 15:16