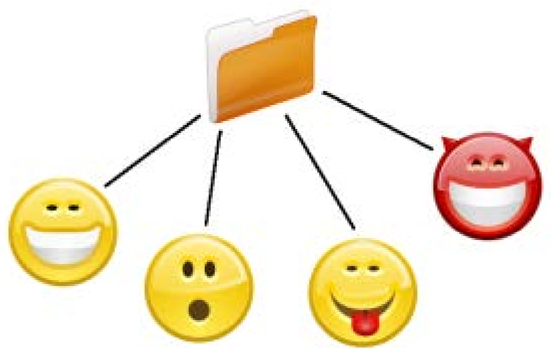
Local File Sharing in Linux



Would you like to share files among multiple users on the same Linux system?
Surprisingly, this is trickier to accomplish than it appears, so here is a method that works.
The problem: You would like to allow multiple users on the same Linux system to share files among themselves. Only users who are members of a special sharing group are allowed access to a common sharing directory, and any user may read, write, and modify the permissions or any file created by any other user. All other users are denied access. Additionally, we want to allow remote access to the shared directory for remote users who do not have a local account on the system.

Sounds simple, doesn’t it? However, unexpected surprises abound. Linux possesses an effective permission system to restrict who may do what with a file, and in order to achieve all of these goals, we must become creative in finding ways to work around the built-in permission system.

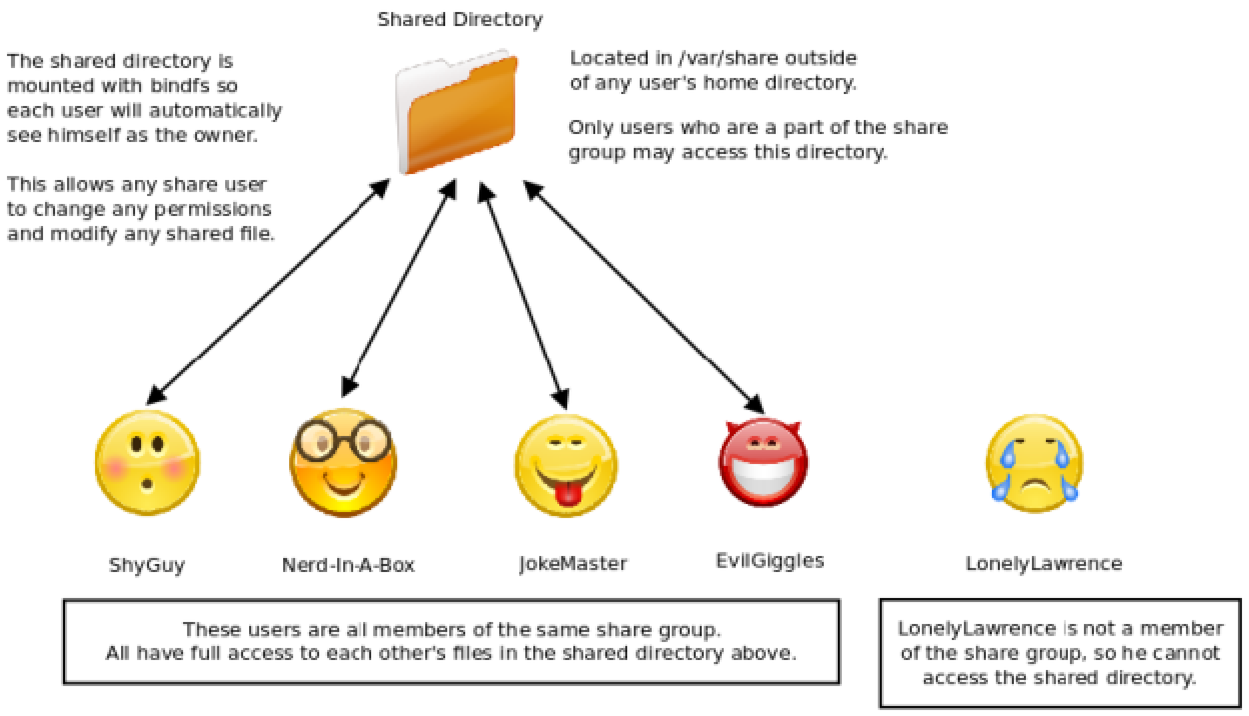
No magical program or mystical command exists that will enable local file sharing as described above. Various methods are possible, but all have drawbacks. The first thought that might come to mind is to create a share group of users allowed to share files with each other, create a shared directory, assign the Set Group ID (SGID) permission to it, and set the directory’s group to that of the share group. This works to a limited extent, but it poses problems when files are copied from an NTFS/FAT file system. Also, the default umask setting denies write permission to the share group, and only the owner may change file permissions.

What we need is a way to automatically assign default permissions to all files created within the shared directory and make each user see himself as the owner of every file so he may modify file permissions.

bindfs to the rescue!

Linux does not possess a built-in way to assign default permissions to a specified directory, so we will rely upon a program called bindfs to assign default permissions automatically and allow each user to become the owner of each file.

Sharing files among local users on the same computer


Shared Directory
The shared directory is mounted with bindfs so each user will automatically see himself as the owner.
This allows any share user to change any permissions and modify any shared file.
Located in /var/share outside of any user's home directory.
Only users who are a part of the share group may access this directory.
ShyGuy
Nerd-In-A-Box
JokeMaster
EvilGiggles
LonelyLawrence
These users are all members of the same share group. All have full access to each other's files in the shared directory above.
LonelyLawrence is not a member of the share group, so he cannot access the shared directory.

Setup

1. Install bindfs

```
sudo apt-get install bindfs
```

2. Create a shared directory

Create a common directory where users will share files.

```
sudo mkdir /var/share
```

For this example, the shared directory named share is created in /var. Place it anywhere outside of a user's home directory. This prevents the shared directory from being accidentally deleted when a user is deleted from the system and eliminates any dependencies on a particular user.

For security, root should be the owner and group of share.

3. Change permissions on /var/share so only root has access

```
sudo chmod 0700 /var/share
```

There is no need to assign SGID or a share group since bindfs will handle that for us.

4. Create a share group

This is best managed from the Group settings GUI, but the command line will also work.

```
sudo groupadd share
```

Add all users to this group who are allowed access to the shared directory.

Add a single user:

```
sudo usermod -aG share inky
```

Add multiple users:

```
sudo sed -i '/^share:/s/\(.*\)/\1,inky,blinky,pinky/' /etc/group
```

Separate each username with a comma. No spaces.

5. Create a readme file

As root, create a readme.txt file inside /var/share that explains the purpose for the shared directory along with any other instructions, jokes, or pieces of advice you wish to include.

```
sudo gedit /var/share/readme.txt
```

This is intended to be a convenience feature for new users since the first thought they might have is *“What is this directory used for?”* A readme file will help explain.
However, there is a slight problem with readme.txt: Any user can delete or change it! *Oh, no!* To prevent this, set the immutable attribute on readme.txt. (Assuming you are in /var/share.)

```
sudo chattr +i readme.txt
```

Now, the readme.txt cannot be deleted or modified in any way, making it constantly present in /var/share. Not even root can change it. To make modifications, clear the immutable attribute first:

```
sudo chattr -i readme.txt
```

When the changes are complete, set the immutable attribute again. Only users with sudo privileges are allowed to change the immutable attribute, so make sure only trusted users on the system are allowed to use sudo. Regular users should be denied sudo privileges anyway and those users who are allowed to use sudo usually know what they are doing, so neither of these issues should be a problem.

6. Create symbolic links to the shared directory

For each user, create a soft link (symbolic link) named share to /var/share.

```
sudo ln -s /var/share /home/inky/share
```

This makes it more convenient for users to access the shared directory and shows that a shared directory is present on the system. The user simply clicks the share link and the shared directory is instantly available. This is much easier than navigating the file system to find /var/share.

To automatically add the share link to all new users at creation time, create a link in the skeleton directory /etc/skel.

```
sudo ln -s /var/share /etc/skel
```

All new users will be given a link to the shared directory within their home directories, but they will be denied access to share until added to the share group.

7. Mount the share

This is where the magic happens. Until /var/share is mounted, nobody can access it. We will mount it with bindfs as another file system so default permissions will be assigned to the files within it automatically.

```
sudo bindfs -o perms=0770,mirror-only=@share,group=share /var/share /var/share
```

perms=0770 The default permissions assigned to all new files
mirror=@share All users in the share group become owners
group=share The default group ownership assigned to files
/var/share The shares directory to mount
/var/share Where to mount /var/share

Notice that we are mounting /var/share in itself. This is not a typo. Doing this eliminates the need to create a separate mount point.

The argument **mirror-only=@share** tells which users will see themselves as owners of the files. This is important since only file owners may modify file permissions. This allows any user with access to share to see himself as a file owner, and thus, be allowed to modify file permissions.
The **@share** section tells bindfs to grab a list of users from those listed in the share group. While individual users may also be listed, this is more convenient. Only one list (the share group) must be maintained.
The **group=share** argument assigns the default group share to files within the shared directory. Only members of group share may access the files. All others are denied access.

Test

If you are a member of the share group, then you should be able to access /var/share. Try creating files and directories. Copy a file from a FAT file system into the share directory. Now, view the permissions of the files created.

Each file shows that you are the owner, and the group is *share* with read and write permissions. Even files copied from a FAT/NTFS file system will contain these permissions.

Switch to another user (Inky, for example) who is a member of the share group and open the shared directory. Before doing anything, view the permissions of any file previously created by the other user. Notice that Inky is now the owner of the file even though Inky is not the one who created it. This is bindfs at work, and this is what we want. Since Inky is now seen as the owner, Inky may modify the permissions of the file. This includes changing the execute permission for scripts.

As Inky, create a new file, and then switch to another user (Blinky in this example). As Blinky, view the permissions of the file created by Inky and notice that Blinky is the file owner. Each user will see himself as the file’s owner, and each file will be assigned the share group ownership with read and write permissions.

By assigning default read and write permissions to the share group, there is no need to change the default umask for users.

Auto-Mounting the shared directory

For convenience, let us configure /etc/fstab so /var/share will mount automatically each time the system boots.

```
sudo gedit /etc/fstab
```

On a new line, enter this:

```
bindfs#/var/share /var/share fuse perms=0770,mirror-only=@share,group=share 0
0
```

The # after bindfs is not a typo. It must be included.

Remote Access

Users with accounts on the local machine may access the shared directory remotely by logging in via SSH. No surprise there. But what if we would like to allow users without local system accounts access to the shared directory?

One method is to create a local user specifically for this purpose. Simply share the username and password of this dedicated share account, and anyone may connect via SSH.

1. Creating the share account

The idea is to create a share user (named share) whose home directory is /var/share.

```
sudo useradd share -g share -M -d /var/share
```

-g share Make the share user a member of the share group
-M Do not create a home directory for this user
-d Set the default home directory to **/var/share**. When the user logs in, /var/share is his home directory

2. Assign a password to the share account

Remember this is the password you will give out to others in order to allow them to have access to /var/share. All remote users must log in as share using this password.

3. Change share’s UserID

To avoid local logins of the share account and to prevent it from appearing on the login screen, change share’s UID to a value of 999 or less. UIDs less than 1000 do not appear on the Ubuntu login screen, thus hiding them.

```
sudo usermod -u 999 share
```

Now, users may login under the share account using the share password.

```
ssh share@address
```

sshfs

sshfs will allow a remote user to mount the local /var/share into his file hierarchy and conveniently access the shared directory as if it were another drive on his system.

1. Install sshfs on the local system

```
sudo apt-get install sshfs
```

2. A remote user mounts the share like this

```
sshfs share@computer:/var/share local_mount_point
```

Users with local accounts may mount remotely using their own accounts, but this allows users without local accounts to mount the shared directory. In both cases, all network traffic is encrypted.

To unmount the sshfs share, the remote user must use this:

```
fusermount -u local_mount_point
```

Source : https://delightlylinux.wordpress.com/2012/04/17/local-file-sharing-in-linux/