Getting Started with MacFUSE: DLS How-To

Posted Jan 16th 2007 1:00PM by Jay Savage
Filed under: Utilities, Features, Macintosh, Open Source, How-Tos

The big splash in the Mac community--and the rest of the world--last week was obviously the iPhone. For Mac users, though, the iPhone announcement may have distracted from the really big news: Amit Singh's release of a MacFUSE beta, his port of the Linux FUSE API to OS X. If you're wondering what, exactly, that means, FUSE stands for Filesystem in USErland, and it provides a generic interface that lets the operating system see virtually anything as a filesystem. Historically, adding new filesystem recognition to an operating system has meant modifying the kernel for each new FS. FUSE, though, provides a single interface that filesystem modules use to interface with the OS. Best of all, anything that provides the correct interface can be interpreted as a filesystem. One enterprising Python programmer even developed a script to let users mount their GMail accounts and use the extra space in their accounts to save files.

What does this mean for Mac users? A lot. First and foremost, the FUSE NTFS driver seems to work with MacFUSE, so we can finally use NTFS volumes as well as FAT volumes. Web developers and anyone else who manages files via SFTP should rejoice, too. SSHFS (included with the MacFUSE binary) allows users to mount a remote SSH/SFTP directory as if it were a local disk. That means no more synchronizing files with an SFTP client. And while the GMailFS python bindings need a little work, the fixes look trivial, and soon we should all be able to put our extra GMail space to better use.

Amit doesn't think that MacFUSE is ready for production use yet, hence the "b" after the version number, so if you're using it for anything important, make sure you've got backups. If you do run into trouble with it, updates are being released almost daily at the moment. That said, though, my trials of it seem pretty stable. The only issues I've seen have been network related. If the system doesn't get a response to a remote query, it will hang. That can mean the dreaded "Spinning Beachball" in the Finder. In most cases, the problem eventually clears itself. If you're navigating via the shell, a simple Ctrl-C cancels the hanging action.

So how do you get MacFUSE up and running for yourself? Glad you asked. There are currently two ways to get MacFUSE: Amit's pre-compiled binaries and the compilable source from the Google code repository. The best thing is to compile the source yourself. Amit has said that the binaries are a "one-off" for early adopters who want to try out the port, and he doesn't plan on releasing them regularly. Installing from source also guarantees that you're getting the latest stable code. Furthermore, MacFUSE installs itself to directories under '/usr/local'. If you have already installed any of the libraries it uses via MacPorts, the MacFUSE installer may clobber your current installation. So use the source.

If you do decide to install the compiled binaries, you can skip to Step 11.

Amit has nice directions up at the Google wiki page, if you've used XCode and Subversion before, you may want to head over there and check them out. The rest of us, though, have some work to do.

If you're not familiar with the OS X command line and the way people write about it, anything below that follows a '$' is something you type at the command line in a Terminal.app window. Anything in italics is something you replace with information you supply. For instance, I would write the lines below as:

$ less INSTALL
$ CFLAGS="-D__FreeBSD__=10" python setup.py *target*

Step 1) Make sure you are logged in as a user that is authorized to administer your computer. You're going to be installing things to system directories.

Step 2) Make sure that you have Xcode installed. It is probably installed on your machine already. Open up a Finder window and click on your primary hard drive. If you see a folder named 'Developer,' you're probably in business. Just to be sure, though, check for '/Developer/Applications/Xcode'. If Xcode isn't there, you can install it from your OS X install disks, or download it from Apple.

Step 3) Get Subversion. Subversion is a versioning system developer use to keep track of projects as they're coding them. It's also any easy way to distribute code to users. If you've ever compiled other open-source projects, you may have already installed Subversion. If not, you need to get it now. If you use MacPorts, a quick 'sudo port install subversion' should give you everything you need. If not, Matthew Porter has made compiled Universal Binaries available. Just double-click the installer as usual.

Step 4) Check out MacFUSE. "Checking out" is what programers call getting the latest version of the code of the subversion repository. It's pretty simple. Just open up a Terminal.app window and do the following (without the quotes):

$ svn checkout http://macfuse.googlecode.com/svn/trunk/ macfuse

That will download the MacFUSE code to a directory called 'macfuse' in your home directory. Now

$ cd macfuse

to switch into the directory you just downloaded.

Step 5) Compile and install the kernel extension:

$ cd fusefs
$ xcodebuild -target fusefs -configuration Release

That will take you to the kernel source directory and compile the source into the working kernel extension. Then,

```
$ sudo cp -pR build/Release/fusefs.kext /System/Library/Extensions/
$ sudo chown -R root:wheel /System/Library/Extensions/fusefs.kext
```

These lines move the kernel extension into the proper folder and change it's owner to 'root' which is unix-speak for 'the user that administers the system'. 'sudo' is the unix command to perform an action as root. The system should ask you for your password after you type the first line.

Step 6) Compile and install the other extensions. your browser may wrap the lines in the post, but make sure that each '$' command is typed on one line.

```
$ xcodebuild -target load_fusefs -configuration Release
$ sudo cp
build/Release/load_fusefs /System/Library/Extensions/fusefs.kext/Contents/

$ sudo chown
root:wheel /System/Library/Extensions/fusefs.kext/Contents/Resources/loac
$ sudo chmod
u+s /System/Library/Extensions/fusefs.kext/Contents/Resources/load_fusefs
$ sudo cp -pR fusefs.fs /System/Library/Filesystems/
$ sudo chown -R root:wheel /System/Library/Filesystems/fusefs.fs
$ xcodebuild -target mount_fusefs -configuration Release
$ sudo cp
build/Release/mount_fusefs /System/Library/Filesystems/fusefs.fs/
$ sudo chown
root:wheel /System/Library/Filesystems/fusefs.fs/mount_fusefs
$ sudo ln -
s /System/Library/Filesystems/fusefs.fs/mount_fusefs /usr/local/bin/mount_
```

Step 7) Download and install pkg-config. This is a program that helps other programs install themselves. Your individual filesystem drivers will use it later. Download the latest version to your desktop. Once it's downloaded, go back to the Terminal ad do the following. If your browser didn't download the file to the desktop, replace "~/Desktop" below with whatever directory the file downloaded to. The backslashes mean that whatever follows should be typed on the same line.

```
$cd ~/Desktop
$ tar -xzvf pkg-config*.gz
$ cd pkg-config*
$ CFLAGS="-O -g -arch i386 -arch ppc -
isysroot /Developer/SDKs/MacOSX10.4u.sdk" \
LDFLAGS="-arch i386 -arch ppc" \
./configure --prefix=/usr/local --disable-dependency-tracking
$ make
$ sudo make install
```

Step 8) Download and install FUSE itself. Get the latest version from SourceForge.

```
$ cd ~/Desktop
$ tar -xzvf fuse-2.6.1.tar.gz
$ cd fuse-2.6.1
$ patch -p1 < /path/to/fuse-2.6.1-macosx.patch
$ CFLAGS="-D__FreeBSD__=10 -O -g -arch i386 -arch ppc -
isysroot /Developer/SDKs/MacOSX10.4u.sdk" \
LDFLAGS="-arch i386 -arch ppc" \
./configure --prefix=/usr/local --disable-dependency-tracking
$ make
$ sudo make install
```

Step 9) Download and install getext and glib. The are two libraries that help programs interact with each other and the OS. You can get gettext from here and glib from here. You can also install them via MacPorts. Again, if you downloaded them to someplace other than the Desktop, change the code below to reflect that.

```
$ cd ~/Desktop
$ tar -xzvf gettext-0.16.1.tar.gz
$ cd gettext-0.16.1
$ CFLAGS="-O -g -arch i386 -arch ppc -
isysroot /Developer/SDKs/MacOSX10.4u.sdk" \
LDFLAGS="-arch i386 -arch ppc -fno-common" \
./configure --prefix=/usr/local --disable-dependency-tracking \
--with-libiconv-prefix=/Developer/SDKs/MacOSX10.4u.sdk/usr
$ make
$ sudo make install

$ cd ~/Desktop
$ tar -xzvf glib-2.12.7.tar.gz
$ cd glib-2.12.7
$ CFLAGS="-O -g -arch i386 -arch ppc -
isysroot /Developer/SDKs/MacOSX10.4u.sdk -I/usr/local/include" \
LDFLAGS="-arch i386 -arch ppc -L/usr/local/lib" \
./configure --prefix=/usr/local --disable-dependency-tracking
```

If you are on a **PowerPC,** you must open up the config.h file in the current directory using TextEdit. Find the line that says

#define G_ATOMIC_POWERPC 1

Add '//' at the beginning to make it read

// #define G_ATOMIC_POWERPC 1

Then, on **both PPC and Intel**, continue with:

```
$ make
$ sudo make install
```

Step 10) Download and install SSHFS. this will be your first FUSE filesystem. It will left you mount remote SSH/SFTP directories as if they are local disks. You can get SSHFS from SourceForge. I'm sure you can guess what comes next.

```
$ cd ~/Desktop
$ tar -xzvf sshfs-fuse-1.7.tar.gz
$ cd sshfs-fuse-1.7
$ patch -p1 < /path/to/sshfs-fuse-1.7-macosx.patch
$ CFLAGS="-D__FreeBSD__=10 -O -g -arch i386 -arch ppc -
isysroot /Developer/SDKs/MacOSX10.4u.sdk" \
LDFLAGS="-arch i386 -arch ppc" \
./configure --prefix=/usr/local --disable-dependency-tracking
$ sudo make install
```

Step 11) Mount your SSH filesystem! Congratulations, you should now have a working MacFUSE installation, and your first FUSE filesystem to go with it.

To actually use a FUSE filesystem, though, you have to do a little configuration. First, decide where you want to mount your remote directory. On unix, remote filesystems (all filesystems, really) are attached to directories, so before you can mount one, you have to have someplace to put it. For filesystems it uses, the system makes directories under /Volumes. FUSE won't use /Vloumes, though, so you need to make a mountpoint someplace in your home directory. You can put is anywhere you can create files, but the Desktop isn't a good choice because Finder will make a link for the filesystem appear there, anyway. You only have to make the directory the first time you mount the filesystem. I usually make a directory called 'ssh':

```
$ cd
$ mkdir ssh
```

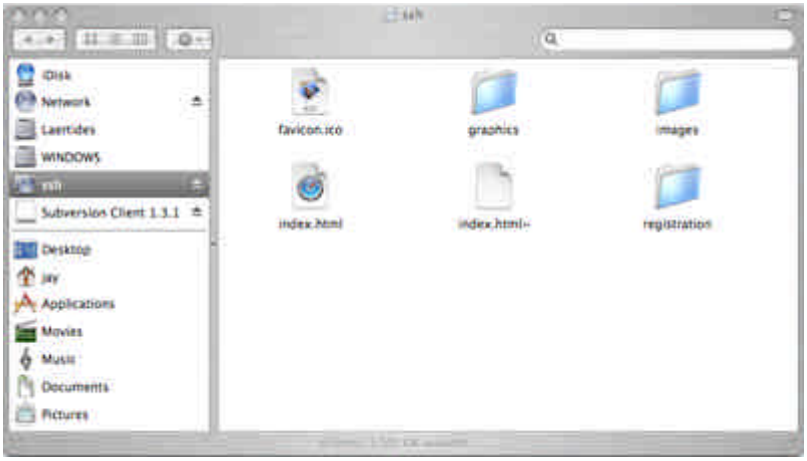
Now all you have to do is mount it. To do that you need you username for the SSH server, the name of the server, your password, and the directory on the server you want to mount locally. You also need to know what name you want Finder to use for the volume. Then to mount, it's just:

```
$ sshfs username@server:path /local/path -oping_diskarb,volname=name
```

For instance, to mount the public_html directory on remote server so that I can edit a web page, I would just type:

```
$ sshfs jay@web.example.com:public_html /Users/jay/ssh -
oping_diskarb,volname=ssh
```

If all went well, everything in my 'public_html' folder on the server should now appear on my local machine at '/Users/jay/ssh'. Better yet, there should be a remote volume icon named 'ssh' on my desktop that I can click on to open the volume in Finder, just as if it were a local disk or an iDisk:



When I'm done working on the server, I just type:

$ umount /Users/jay/ssh

And the disk is unmounted (ejected). MacFUSE isn't integrated with the finder's eject button yet.

And that's it. Downloading and compiling everything may take a little while, but as you can see, once everything is installed, using your FUSE volumes is dead simple. Here are a few pointers to avoid some common pitfalls, though:

- As I said before, don't try to make mountpoints in /Volumes or on the Desktop
- Use the command line to navigate until you get a feel for how robust your network connection is. That way if anything hangs you can just kill the operation with Ctrl-C and not have to deal with the Finder's "Beachball of Death". If you find that simple operation like 'cd' and 'ls' are hanging, try again later from a more stable connection.
- If you use TextMate, don't try the mate command in a remote directory. I don't know why, but mate, particularly 'mate .', seems to hang in sshfs mounts. opening files and directories from the open dialog in TextMate's File menu works fine.
- Make sure you unmount remote directories before putting your machine to sleep, and before you disconnect from the network. Losing a connection to a mounted FS can lead to the beachball.
- When you install other FUSE filesystems, make sure you add 'CFLAGS="-D__FreeBSD__=10"' before './configure' when you follow the developers' installation instructions.
- Remeber the MacFUSE isn't integrated with the Finder's unmount function yet. Dragging FUSE volumes to the trash woun't unmount them.
- Make sure /usr/local/bin is in your PATH.

Once you get comfortable with SSHFS, feel free to poke around some of the other FUSE filesystems out there ntfs-3g, CryptoFS, and EncFS are reported to work, as are some others listed on the wiki. Just follow the install directions that come with the distributions. Just FYI, though: I've had some trouble getting the ntfs driver to install. If i get it working, I'll post the fix. If you get it working first, let us know!

Edit:

Fixed pkg-config link
Added note about /usr/local/bin