Automount mini−Howto

Table of Contents

Automount mini−Howto

`don@sabotage.org`

v0.4, 17 April 1999

This file describes the autofs automounter, how to configure it, and points out some problems to avoid.

1.Introduction

1.1 Automount − what and why

Automouting is the process where mounting (and unmounting) of certain filesystems is done automatically by a daemon. If the filesystem is unmounted, and a user attempts to access it, it will be automatically (re)mounted. This is especially useful in large networked environments and for crossmounting filesystems between a few machines (especially ones which are not always online). It may also be very useful for removable devices, or a few other uses, such as easy switching between a forced−on ascii conversion mount of a dos filesystem and a forced−off ascii conversion mount of the same dos fs.

1.2 Types of automounting

There are two types of automounters in linux; *AMD* and *autofs*. AMD is the automount daemon, and supposedly works like the SunOS AMD. It is implemented in user space, meaning it's not part of the kernel. It's not necessary for the kernel to understand automounting if you NFS mount to the local host, through the AMD daemon, which routes all automount filesystem traffic through the NFS system. Autofs is a newer system assisted by the kernel, meaning that the kernel's filesystem code knows where the automount mount points are on an otherwise normal underlying fs, and the automount program takes it from there. Only autofs will be described in this mini−howto.

2.Installation

Because autofs is implemented in kernel−space, your kernel must have support compiled in. In 2.0.xx it is an experimental option, but appears to be quite stable. In 2.2.xx it is a normal option.

The automount program and its configuration files are also necessary; using the rpms (from RedHat, as part of the install) is a great way to go. The automount program should be started by an rc script under the `/etc/rc.d/init.d` directory . The rpm installs this, but you will need to make sure it gets started, either by linking it from your rc?.d directory, using Redhat's control−panel, or on another distribution by getting the thing started anyway you care to. Non−rpm distros will have to do whatever's applicable to their system. And don't look too hard at what the rc script does; if you're reading this howto you probably don't want to know.

3.Configuration

Installing the RPM's will get you to this point easily enough, but here's the part you might not be sure about if you haven't done this before.

There are two files in /etc, one called `auto.master` and one called `auto.misc`. My auto.master looks like this:

```
/auto   /etc/auto.misc  --timeout 60
```

The first entry is not the mount point. It's where the set of mount points (found in the second entry) are going to be. The third option says that the mounted filesystems can try to unmount themselves 60 seconds after use. They can't unmount if being used, of course.

Auto.misc is a "map file". The map file can have any name; this one is named auto.misc because it originally controlled /misc. Multiple map files can be defined in auto.master. My auto.misc looks like this:

```
kernel          -ro,soft,intr          ftp.kernel.org:/pub/linux
cd              -fstype=iso9660,ro     :/dev/cdrom
zip             -fstype=auto           :/dev/hdd4
floppy          -fstype=vfat           :/dev/fd0
```

The first column (the "key") is the mount point. In this case it would be /auto/floppy or whatever. The middle set are the options; read the mount manpage for details on this. And the last column specifies where the fs comes from. The "kernel" entry is supposed to be an NFS mount. The : on all the other lines means its a local device.

4.That long wait to unmount

Some of you may be eyeing that 60 second timeout and thinking, *that's a long time to wait to eject a floppy.. Maybe I'll just sync the disks and pop it out mounted and nobody will notice.* Let me suggest `saner alternatives`. First of all, you can change the timeout. But that could be a little inefficient; telling the system to unmount stuff after only 15 seconds or whatever. Depending on your setup, you may be able to simply run the umount command as a normal user. But there is actually a way to ask the automount program to umount. If you send (with the program kill) the signal `SIGUSR1` to the automount process, it will unmount everything it can. But before people start making unmount buttons on their window managers, there's a little problem.

The automount process is run by root, and it will only accept signals from root. Half of the reason you're probably doing automounting is so you can mount an unmount *without* being root. It would be easy to make a suid−root C program which does the dirty deed. However, by using sudo it is possible to allow users to send the proper kill signal. The only problem is that sudo will not let you use ` to process subcommands, which you would have to do to find the current PID. You should have a program called killall, which will let you do this: (thanks for the suggestions)

```
ALL     ALL=NOPASSWD:/usr/bin/killall -USR1 automount
```

Otherwise, you would have to allow your users to send −SIGUSR1 to all processes. That has various effects on programs; it will recycle some window managers, but kills xemacs. So here's hoping there's no buffer overruns in killall...

5.Questions

5.1 I don't see /auto/floppy, or whatever mountpoint I'm looking for.

If automount is setup properly, whatever mount point you're looking for will be there if you try and use it, even though you don't see it when not in use. If you're browsing the directory with a graphical tool, you may need to type in the name manually; most programs will try what you give it, and the drive will be mounted before it notices. Unfortunately not being able to choose from the available invisible mount points is probably the major drawback of autofs. If it really bugs you, edit the configuration files. (Hint, the ones that end in .c for "configuration")

One workaround several people have tried is to create symbolic links to where automount will create something once it's mounted. This will likely prevent the program from complaining a directory doesn't exist (if the mount works, that is) but careless directory listings will cause filesystems to be mounted.

5.2 How do I see what's mounted?

The `df` command. `mount` with no options will do the same, plus show the options its mounted with.

5.3 I put in a win95 disk ("vfat") and it was autodetected as only a regular FAT disk.

This is not a problem with automount. As of this writing, the "auto" fs type does not attempt a vfat mount before it successfully mounts an msdos filesystem. VFAT is the Win95 and WinNT long filenames crammed into a FAT/MSDOS filesystem.

According to one of the authors of mount, since mount is only a wrapper around a system call which must specify the filesystem type, it's still the responsibility of the user to come up with the fs type. Having mount take a list of filesystems to try in order, rather than the current "heuristic" is under consideration. Some users have simply not compiled msdos into the kernel; this prevents it from being tested prior to vfat. This will work for most people; a few actually need msdos fs and it caused me quite some frustration to not have a module handy when I actually needed it.

I'm sure that if anyone wants to go to the effort of finding the owner(s) of the mount program, your comments would be welcome. So unless you don't compile msdos in, for now this means that you can't mount vfat unless you give up the ability to autodetect all other fs's. Hopefully it will be configurable someday. In the mean time, feel free to create multiple mount points with different fs types specified.

5.4 My filesystem `/auto/grumblesmurf` is mounted and `kill −SIGUSR1` won't unmount it.

It's being used by something. Root probably can't manually unmount it either. If you're the one who caused it to be mounted (i.e. it can't be someone else using it) look around for a shell that might be in that directory. If there are none, look for something else (particularly something that might have gone though that directory like a directory browser) that might have left an invisible foot in the door so to speak. If you've given up looking, try using the fuser program.

5.5 What happens if I make / the directory for the automounter

Oooh. Well, out of a statistical sample of only one person, none of the results were positive. You have been warned. If you want /grumblesmurf, then I suggest a symbolic link. Much safer.

5.6 Can I have two map files on the same directory?

Not as far as I know. Try using one map file, with specific options for individual entries.

5.7 I'm using SuSE 6.0 and needed `−−−timeout` instead of `−−timeout`

Uh. Ok, I've made a note about it. Another solution to "timeout not working" problems would be to add a `−t time` option to the autofs script.

5.8 How do I set the permissions and ownership for the filesystem (IE FAT)

Check the man page for mount for some of the options, such as setting the uid=value or umask=value options. One option that appears to be missing for FAT filesystems is mode=value. Sorry. Check in with the people who do mounting.

5.9 Who do I thank for autofs?

Not me. I didn't have anything to do with it. I just wanted to bring everyone's attention to what a great job had been done with autofs, and how easy it is to use. Compared to the original perpetrators of AMD (Hint, they sell an overpriced unice with prehistoric versions of free tools) the autofs is very well documented and the implementors have my sincere thanks. Everything is stamped copyright Transmeta so sorry I can't provide a credits list, but I would bet Peter Anvin is responsible for quite a bit of it. Peter also held a session on autofs at linuxworldexpo on March 3, 1999.

5.10 Where can I learn more about automounting?

There's a autofs tutorial at http://www.linuxhq.com/lg/issue24/nielsen.html. See also am−utils at http://www.cs.columbia.edu/~ezk/am−utils

(Thanks for these URLs)