Setting up and using the automounter

Table of contents

1. Revision history

Date	Description
2006-02-22	Initial version.
2006-06-05	Added ghosting and the ghosting patch.

2. Introduction

In the 1990s, workstation/server networks were growing from large to huge. Typically, a large site had a number of servers, exporting the users' home directories (as well as possibly project directories), and number of workstations, importing ("mounting") these directories. This lead to a large number or mounts, having bad impact on both performance and reliability. Typically, a client importing a file system from a server that, due to own or networks failure, was temporarily unaccessible, was either extremely slowed down, or it simply hang. Also, for file systems mounted, but not really used, network traffic was generated just to keep the mounts active. As a solution to these problems, the automounter was suggested. With the automounter, file systems (like user home directories) was mounted only when accessed, and unmounted after a certain period of inactivity. Typically, there was a user data base, consisting on information on what file system, from what server to mount for a particular user. This was typically taken from a LAN-wide user database, such as NIS (formerly called YP), later NIS+, LDAP, or other.

How does this relate to the situation with a dBox in a home network? The large workstation net problems are hardly an issue. The ability to take mount maps from information systems like NIS is most definitely not there. However, the automounter gives some more comfort: It is not necessary for a file system to be available when booting the dBox, in order to access it later. The need to manually mount file system before recoding (or through files like `recording.start`) is eliminated.

There seems to be a general consensus that the automounter is for large networks. I have personally been using it for my home network, consisting of two or three hosts (Solaris and/or Linux), since 1999. It is not too hard to setup, has very low overhead, is highly reliable, and makes life a little bit more comfortable.

3. Building an image (or YADD) with automounter support

Support for the automounter is contained in the Tuxbox CVS. In the *newmake* CVS branch, this support *almost* allows for automatic build. It would probably not be major effort to port the *newmake* the MAIN files, however at the time of this writing, this has not been done.

To support the automounter, the kernel configuration option `CONFIG_AUTOFS4_FS` has to be turned on. For this, the line

```
CONFIG_AUTOFS4_FS=y
```

has to be added to the kernel configuration file, `cdk/Patches/linux-$version-dbox.config` (for YADD) and `cdk/Patches/linux-$version-dbox.config-flash` (for images). Here, `$version` denotes the actual kernel version used, at the time of this writing it amounts to "2.4.32".

In *newmake*, automake is an unpack-patch-build-install-delete-target. Installing in YADD is done with the make target `automount`, for images with the make target `flash-automount` (installing in `$flashprefix/root`). Therefore, the elegant way is to include the line `make automount` in (e.g.) `yadd-neutrino-local.sh` and/or the line `make flash-automount` in `root-local.sh`, possibly together with an appropriate `/etc/auto.net` configuration file instead of the default one (which is effectively empty).

That's all!

4. Setting up the automounter

The typical Unix/Linux setup has been simplified. I believe that not very many dBox users are interested in reading NIS-maps for the automounter, however, they may have some servers, NAS-devices, and some Notebooks around. The automounter is therefore, per default, configured only with one "map", and one configuration file. The startup file, `/etc/init.d/start_automount` also contains some configuration possibilities. Possibly the most interesting is the variable `AUTOFSMOUNTDIR`, indicating the directory under which the mounts will take place. This directory must be absolute, must reside in a writable area, and should not exist at booting time. The default file defines it to `/var/autofs`.

The default file loads the necessary modules for mounting NFS file system. When mounting CIFS file systems, it may be necessary to load the CIFS module; uncomment the appropriate line in `start_automount`.

4.1. The configuration file /etc/auto.net

Every (non-comment) line in the configuration file `/etc/auto.net` describes a mount file system. It has three fields: the mount name, the mount parameters (see the example for syntax), and the server file system. For example, the first (non-comment) line states that the file system `/pictures` on the server `myserver` will be mounted as `$AUTOFSMOUNTDIR/pictures` (with the default values, this is `/var/autofs/pictures`). Names instead of IP-numbers are accepted, as long as they can be resolved using the normal host name resolving mechanisms (normally `/etc/hosts`, sometimes supported with a name server, declared in `/etc/resolv.conf`).

```
# This is an example of an automounter map
#
```

```
# Mount name    Parameters                server file system
pictures       -fstype=nfs,ro,nolock   myserver:/pictures
music          -fstype=nfs,ro,nolock   yourserver:/audio
movies         -fstype=nfs,ro,nolock   192.168.42.42:/filme
recording      -fstype=nfs,rw,nolock   herserver:/garbage
#
# This example is from Papst
musik
-fstype=cifs,ro,soft,user=root,password=dbox2,unc=//192.168.0.2/Musik
//192.168.0.2/Musik
```

If sent the `USR1`-signal, the automounter will unmount the unused file system (that it has mounted). The `TERM` will force the automounter (the `automount` process) to unmount unused file systems, and to exit cleanly. A convenient way for this is

```
kill -USR1 `cat /var/run/automount.pid`
```

and

```
kill -TERM `cat /var/run/automount.pid`
```

respectively.

4.2. Ghosting

Users often find the standard behavior of the automounter confusing: When listing a directory, it is nothing there, although the automounter will create entries under certain circumstances! Even worse, trying to select, for example a recording directory from Neutrino, is simply not possible if the directory is not presently mounted.

For this, recently so-called *ghosting* was implemented in the automounter and the Linux kernel. Using this feature, the top level mounted directories will still be visible, also when the directories are not mounted. Thus it is possible to use the Neutrino file selector to select an automounted directory, also if it is not mounted when the file selector starts.

To use ghosting, a patch needs to be applied to the kernel. Using this patch (which patches the kernel configuration files for `CONFIG_AUTOFS4_FS=y` too) will integrate the patching into the build process. Furthermore, the automount deaemon needs to be started with the `-g`-option. (This is the default in the newmake setup).

4.3. Troubleshooting

First of all, what cannot be mounted from the command line, the automounter cannot mount either. Try the mounts from the command line, to check for misspellings, that the options are sensible, and that the server's export permissions are appropriate.

Check that the automount process is running using the `ps` command. Check that the `$AUTOFSMOUNTDIR` (default is `/var/autofs`) is present. Using the example above, the command `ls /var/autofs/pictures` (note: cannot use the "TAB"-command line completion in the shell as long as the file system is not mounted) should mount the appropriate file system on `/var/autofs/pictures`. The `df` command can be used to list the file systems presently mounted.

5. Questions and answers

5.1. Neutrino "automounts" file systems. How does this relate?

Not at all. Since everything a program makes can be called "automatic", someone (probably who never had heard of automounting in our sense) decided to call non-interactive mounting on boot "automounting". Sad. And confusing.

5.2. Will the changes to the kernel configuration file have any bad effects when cvs update-ing?

In general, no. CVS will detect that the file is locally modified, and, as long as possible, merge future changes in CVS with local changes. However, watch out when the kernel version changes!

5.3. What needs to be done the MAIN branch to support automount?

`cdk/Makefile.am` needs to be updated (probably just inserting the content of the *newmake* file `cdk/make/automount.mk`), `rcS` and/or `rcS.insmod`, as well as the image building rules (for installing `auto.net` and `start_autofs`, probably in a writable place). Then modify the image building rule to install the required kernel modules and `/sbin/automount`.

5.4. Can I extend my favorite image with the automounter?

Sure. Just replace the kernel with one with CONFIG_AUTOFS4_FS enabled, add the required kernel modules, make sure that they match the kernel version, install `/sbin/automount`, `auto.net`, modify `rcS`, and install `start_automount`.

Shorter version of the above: If you are able to all this, you will be building your own images anyhow. Therefore, it does not seem very logical.

Even shorter version: Forget it!

5.5. Is this stuff restricted to Neutrino? Will it work with Enigma?

No. Yes.

5.6. Can I combine Neutrino mounting with the automounter?

Yes, as long as the mount points do not conflict.

5.7. How do I get at the debug messages?

Unfortunatelly, automount is designed to do all its logging though the syslog facility, an

this is per default not present on the Tuxbox, at least not in images. To enable the syslog facility, enable the option `CONFIG_SYSLOGD` in the busybox configuration file, and recompile busybox.