Hacking the Linux Automounter

Linux Kongress 2005

Jeff Moyer <jmoyer@redhat.com>

Senior Software Engineer

Overview

- What is the automounter?
- Configuration
- Linux implementation
- What's to come: autofs v5
- Contributing

What is the automounter?

- Automatically mount and unmount file systems
  - NFS
  - Local file system (--bind mounts)
  - Samba*
  - etc.

- Why?
  - Manageability!
  - This goal is not achieved by the current Linux automounter.

Configuration – The Master Map

- auto.master (Linux) or auto_master (UNIX)
- Source of all further configuration
  - where automount-owned file systems are to be mounted
  - name of the map file to read
  - Format: “`mount-point map-name [mount-options]`”

- `mount-point`
  - full path to the directory used as a mount point
  - if it does not exist, it is created
  - `'+'` if the map is an included map.
  - `'/-'` if this is a direct map

The Master Map (cont'd)

- `map-name`
  - Map file to read
  - `'-hosts'` if the map is to be a “slash net” map
  - `'-null'` if we want to override a specific map entry
    - Useful when used in conjunction with included maps
- `mount-options`
  - Just what it says, options which are passed to the mount command

```
# Sample auto.master
/misc    /etc/auto.misc
/net     -hosts
/nfs     auto_nfs
/-       /etc/auto.direct
+auto_master
```

Mount Maps

- Indirect Maps
  - Describe the mount points, or directory hierarchy, under the directory specified in the master map*
- Direct Maps
  - Contain a list of full directory paths and the location from which the file system is to be mounted
- Format: "`key [mount-options] location`"

- `key`
  - directory name being looked up
  - for indirect maps, this is a relative path starting from the automount directory
  - for direct maps, this is a full path

Mount Maps (cont'd)

- `mount-options`
  - optional, comma-separated list of options applied to the mount entry
  - may be file system mount options, or map options (such as -DOS=RHEL3)
- `location`
  - specifies the file system to be mounted on key
  - conforms to one of the following
    - single file system (simple case)
    - replicated server entry
    - multi-mount entry
    - paths beginning with a '/' must be escaped with a ':'

Replicated Server Entries

- Support multiple, typically read-only sources of the same data (GFS, anyone?)
- Entries can be weighted
- Entries can come from different paths on different servers
- Server selection follows the priority:
  - lowest weight
  - closest network proximity

```
/usr/share/man  -ro  server1,server2,server3:/export/share/man
/usr/share/doc  -ro  server1(50):/export/share/doc,server2:/export/share/doc
```

- UNIX automounter implementations provide multiple servers to the mount command. NFS takes care of switching servers when one doesn't respond.

Multi-mount Maps

- Allow for the specification of an entire directory hierarchy as a single map entry
- Mount options can be specified per mount-point
- Allows one to cobble together a directory hierarchy from multiple servers
- Gets around the “no nested mounts” limitation

```
server1  -rw  \
    /              server1:/export/        \
    /bin      -ro  server1:/export/bin     \
    /usr           server1:/export/usr     \
    /usr/bin  -ro  server2:/export/usr/bin \
    /scratch       server2:/export/scratch
```

Multi-Mount Maps (cont'd)

- autofs4 limitations
  - mounted and unmounted as a single unit
  - /net is implemented as a multi-mount map
  - can cause MANY directories to be mounted at once
    - puts pressure on reserved port space

Wild Card Keys

Example mount map, auto.misc:

```
music          myserver:/export/music
*              myserver:/export/&
```

- Special Characters
  - “*” - The wildcard entry
  - “&” - substitutes whatever was entered as the key
  - “#” - comment character

Name Service Switch

- /etc/nsswitch.conf
- In theory, one interface to access multiple backing stores
- No support in libc for autofs
- Current “algorithm” has **nothing** to do with the order in nsswitch.conf!
  - if it starts with '/' and is executable, it's a program map
  - if it starts with '/etc/' and is executable, it's a program map
  - if it starts with '/' and is a file, it's a file map
  - if it starts with '/etc/' and is a file, it's a file map
  - else, it's a yp map
- Exception:
  - Red Hat packages consult nsswitch.conf when determining the source of a submount map

Special Maps

- -hosts
  - treats `key` as a server name
  - performs a showmount -e on key and sorts the output
  - generates a multimount entry and mounts it
  - browsing not recommended
- -null
  - Specified to nullify a map
  - Must be specified before the entry to be disregarded

```
/home        -null
+auto_master
/home        /etc/auto.home
```

Included Maps

- Incorporates the contents of another map file into the current map

```
auto.master:
/home    auto.home
+auto_master
/nfs     auto.nfs

auto_master:
/site    auto.site

RESULT:

/home    auto.home
/site    auto.site
/nfs     auto.nfs
```

Multi-Map Entries

- Only supported in auto.master, and only supported under Linux

```
/home        file auto.home -- yp auto_home
```

- the later maps are simply appended to the first
- no limit on the number of maps to concatenate
- Collisions are OK
  - Use the first instance of the key we find

Submount maps

- Use another map to define the contents of this mount point
- Can be thought of as a master map
- specified via the -fstype mount option

```
auto.master:
/lanhosts        /etc/auto.lanhosts

/etc/auto.lanhosts:
server1  -fstype=autofs   file:auto.server1
server2 -fstype=autofs    file:auto.server2

auto.server1:
foo      server1:/export/foo
bar      server1:/export/bar
baz      server1:/export/share/baz
```

Submount Maps (cont'd)

```
/lanhosts  <-- fstype = autofs
         /server1  <-- fstype = autofs
                 /foo  <-- fstype = nfs
                 /bar
                 /baz
         /server2  <-- fstype = autofs
```

```
# mount | grep lanhosts
automount(pid10523) on /lanhosts type autofs
   (rw,fd=5,pgrp=10523,minproto=2,maxproto=4)
automount(pid10532) on /lanhosts/server1 type autofs
   (rw,fd=5,pgrp=10523,minproto=2,maxproto=4)
server1:/export/foo on /lanhosts/server1/foo type nfs (rw)
...
```

Autofs v4 Direct Maps

- Implemented as submount maps
  - for each element of the path, a submount is defined
- 2 key problems with this
  - the top-level path component will be overmounted by an autofs file system
  - as a result of the above, you cannot have a top-level direct mount
  -

Autofs4 Direct Map Example

```
auto.direct:
/nfs/os/linux/usr         linuxserver:/export/usr
/nfs/os/linux/bin         linuxserver:/export/bin
/nfs/os/linux/local       linuxserver:/export/local
/nfs2/foo                 fileserver:/export/foo

/nfs  <-- fstype = autofs
    /os  <-- fstype = autofs
       /linux  <-- fstype = autofs
             /usr  <-- fstype = nfs
             /bin  <-- fstype = nfs
             /local  <-- fstype = nfs
/nfs2  <-- fstype = autofs
     /foo  <-- fstype = nfs
```

Autofs4 Direct Map Example (cont'd)

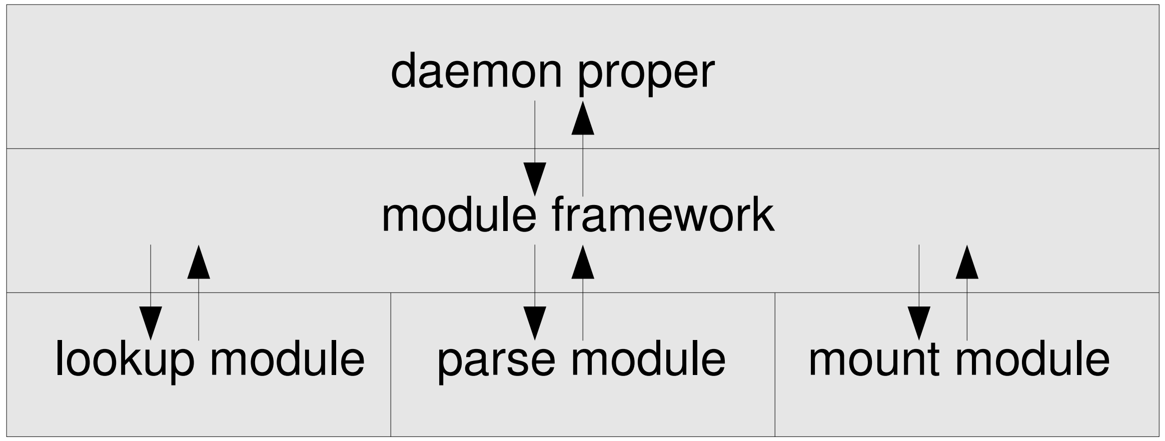
```
automount(pid13258) on /nfs type autofs
   (rw,fd=4,pgrp=13252,minproto=2,maxproto=4)

automount(pid13262) on /nfs2 type autofs
   (rw,fd=4,pgrp=13252,minproto=2,maxproto=4)

automount(pid13270) on /nfs/os type autofs
   (rw,fd=4,pgrp=13252,minproto=2,maxproto=4)

automount(pid13276) on /nfs/os/linux type autofs
   (rw,fd=4,pgrp=13252,minproto=2,maxproto=4)


root     13252  0.0  0.2  1808  720 ?        Ss   12:10   0:00
   /usr/sbin/automount --timeout=60 /- file /etc/auto.direct

root     13258  0.0  0.2  1808  736 ?        S    12:10   0:00
   /usr/sbin/automount --submount --timeout=60 /nfs file /etc/auto.direct

root     13262  0.0  0.2  1808  728 ?        S    12:10   0:00
   /usr/sbin/automount --submount --timeout=60 /nfs2 file /etc/auto.direct

root     13298  0.0  0.2  1812  736 ?        S    12:12   0:00
   /usr/sbin/automount --submount --timeout=60 /nfs/os file /etc/auto.direct

root     13307  0.0  0.2  1808  728 ?        S    12:12   0:00
   /usr/sbin/automount --submount --timeout=60 /nfs/os/linux file
   /etc/auto.direct
```

Automount Architecture

- User-space Daemon
  - parse maps
  - create automount directories
  - perform mounts and unmounts
  - triggerring expiry of mounts
- Autofs file system
  - trap file system access to automount owned directories
  - provide daemon with information on mount point usage

Automount Loadable Modules

- Loadable modules
  - lookup
    - files, nis, nisplus
  - parse
    - sun, hesiod
  - mount
    - autofs, generic, nfs, etc.



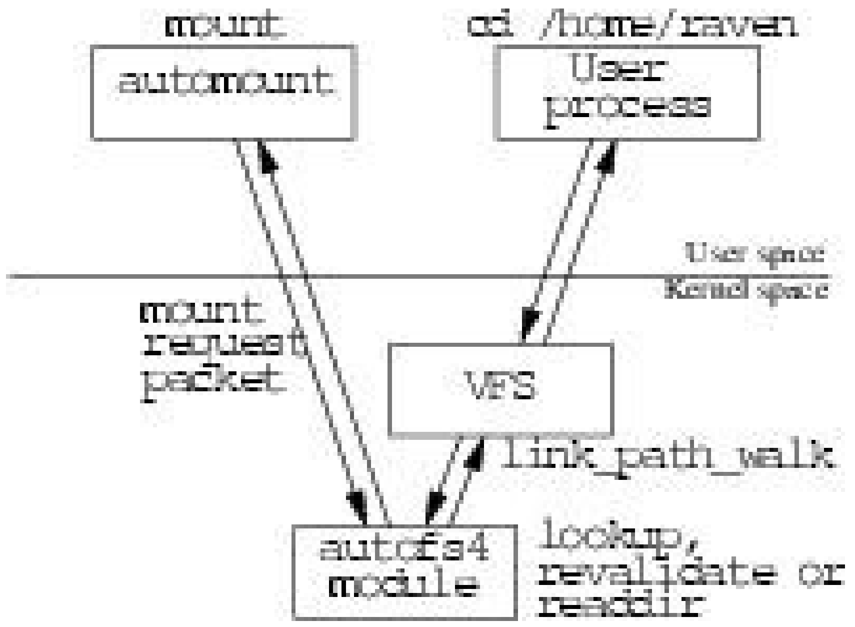
Autofs Loadable Modules

- Benefits
  - Easy to maintain out-of-tree modules
- Drawbacks
  - Introduces artificial separation
  - specifically, causes problems for included maps

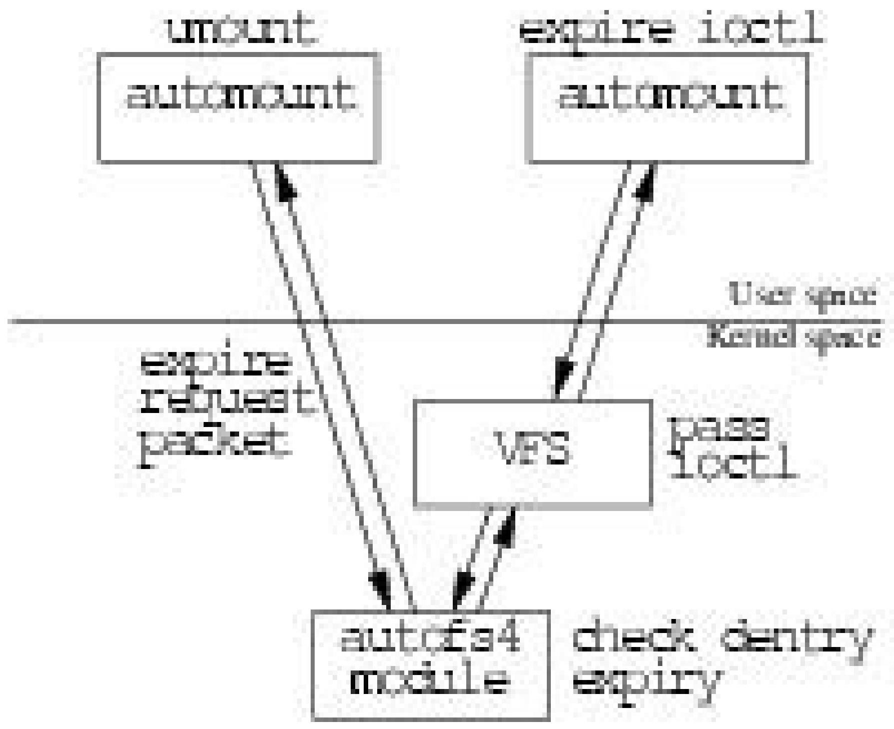
Autofs4 File System

- Virtual file system
- Register triggers for
  - `readdir`
  - `lookup`
  - `d_revalidate`

Mount Diagram



Expiry Diagram



Autofs Inherent Race Conditions

- Expiry
  - ioctl(IOC_EXPIRE_MULTI)
  - kernel checks the use count of a directory hierarchy
  - it checks out okay, so we tell the daemon to go ahead and expire the tree
  - an application traverses into the directory hierarchy we are trying to expire
  - the unmount fails
- Is this a big deal?
  - Not really.

Autofs5

- Goal: 100% compatibility
- Big ticket items
  - Direct map support
  - lazy mount and unmount of multimount maps
  - Utilize the name service switch
  - Included maps

Autofs5 – Direct Maps

- Need to install hooks in the file system which trigger an automount, without mounting an autofs file system
  - file system stacking was considered to be too complex
  - Hackery ensues...
  - `->follow_link` is never filled in for a directory inode, we can use that!
- Setting up a direct mount trigger now looks like this:
  - Create the required directory in the hierarchy
    - the directory can exist on the host file system, on an nfs mounted file system, cifs, etc.
  - Install our own follow_link routine
    - This routine now is called (from `link_path_walk`) when a program accesses the directory

Autofs5 – Direct Maps (cont'd)

- Using this method, we now only need 1 daemon for all direct mounts
- Updates and removals from the map are processed automatically, but not additions

Autofs5 – Lazy mount/unmount

- Turns out we can leverage the dirct map work for this, too
- Still create our autofs directory hierarchy, but now don't mount everything at once.
- Triggers are installed in the appropriate top level directories, and are mounted upon access

Autofs5 Features (cont'd)

- Utilize the name service switch
  - master map
  - submount maps
  - Need to either write a parser for the nsswitch.conf file format, or implement a nss module for automount

```
automount: nis [NOTFOUND=return] ldap
```

- Support included maps
  - detect recursion
  - “integration” of modules

Summary

- Added
  - extended to support multi-map entries (not to be confused with multi-*mount* entries)
- Missing
  - /etc/nsswitch.conf is not currently consulted
  - -null map not supported
  - Included maps are not supported
- Different
  - Direct maps
  - -browse is not the default, and is called -- ghost in Linux
  - -hosts maps are implemented as multi-mount maps
    - lazy mounting and unmounting is not implemented

Contributing

- Master Map Utility
  - get rid of most of the init script
  - maybe even replace the entire parser
- CIFS!
  - mount with proper uid/gid
  - authentication
- LDAP
  - currently only supports anonymous access
- Interaction with new bind mount semantics (or old)
- Real replicated server support
  - changes to mount, nfs client code, minimal changes to autofs
- Bug whacking
- Testing

More Information

- autofs mailing list
  - autofs@linux.kernel.org
- my people page
  - http://people.redhat.com/jmoyer/
- official autofs distribution
  - ftp://ftp.kernel.org/pub/linux/kernel/people/raven/

- Related Projects
  - autofsng
  - amd
  - autodir

References

[1] Linux Kernel source, Version 2.4 and 2.6, http://www.kernel.org/.

[2] Hal Stern, Mike Eisler and Richardo Labiaga, Managing NFS and NIS, 2nd Edition, O'Reilly, June 2001

[3] W. Richard Stevens, Bill Fenner, and Andrew M. Rudoff, UNIX Network Programming, The Sockets Networking API, Volume 1, Third Edition, Addison-Wesley Professional Computing Press, 2004.

[4] Travis Bar, Nicolai Langfeldt, Seth Vidal and Tom McNeal, Linux NFS-HOWTO, http://nfs.sourceforge.net/nfs-howto/, 2002-08-25.

[5] FiST: Stackable File System Language and Templates, Eraz Zadok et al., http://www.filesystems.org/.

[6] Sun™ Microsystems NFS Administration Guide, Chapter 5, http://docs.sun.com/, 1995.

[7] Robert Love, Linux Kernel Development, Second Edition, Novell Press, 2005.