Automount Support

Support is available for filesystems that wish to do automounting support (such as kAFS which can be found in fs/afs/ and NFS in fs/nfs/). This facility includes allowing in-kernel mounts to be performed and mountpoint degradation to be requested. The latter can also be requested by userspace.

In-Kernel Automounting

See section "Mount Traps" of Documentation/filesystems/autofs.rst

Then from userspace, you can just do something like:

```
[root@andromeda root]# mount -t afs \#root.afs. /afs
[root@andromeda root]# ls /afs
asd  cambridge  cambridge.redhat.com  grand.central.org
[root@andromeda root]# ls /afs/cambridge
afsdoc
[root@andromeda root]# ls /afs/cambridge/afsdoc/
ChangeLog  html  LICENSE  pdf  RELNOTES-1.2.2
```

And then if you look in the mountpoint catalogue, you'll see something like:

```
[root@andromeda root]# cat /proc/mounts
...
#root.afs. /afs afs rw 0 0
#root.cell. /afs/cambridge.redhat.com afs rw 0 0
#afsdoc. /afs/cambridge.redhat.com/afsdoc afs rw 0 0
```

Automatic Mountpoint Expiry

Automatic expiration of mountpoints is easy, provided you've mounted the mountpoint to be expired in the automounting procedure outlined separately.

To do expiration, you need to follow these steps:

1. Create at least one list off which the vfsmounts to be expired can be hung.
2. When a new mountpoint is created in the ->d_automount method, add the mnt to the list using mnt_set_expiry():

   ```
   mnt_set_expiry(newmnt, &afs_vfsmounts);
   ```

3. When you want mountpoints to be expired, call mark_mounts_for_expiry() with a pointer to this list. This will process the list, marking every vfsmount thereon for potential expiry on the next call.

   If a vfsmount was already flagged for expiry, and if its usage count is 1 (it's only referenced by its parent vfsmount), then it will be deleted from the namespace and thrown away (effectively unmounted).

   It may prove simplest to simply call this at regular intervals, using some sort of timed event to drive it.

The expiration flag is cleared by calls to mntput. This means that expiration will only happen on the second expiration request after the last time the mountpoint was accessed.

If a mountpoint is moved, it gets removed from the expiration list. If a bind mount is made on an expirable mount, the new vfsmount will not be on the expiration list and will not expire.

If a namespace is copied, all mountpoints contained therein will be copied, and the copies of those that are on an expiration list will be added to the same expiration list.

Userspace Driven Expiry

As an alternative, it is possible for userspace to request expiry of any mountpoint (though some will be rejected - the current process's idea of the rootfs for example). It does this by passing the MNT_EXPIRE flag to umount(). This flag is considered incompatible with MNT_FORCE and MNT_DETACH.

If the mountpoint in question is in referenced by something other than umount() or its parent mountpoint, an EBUSY error will be returned and the mountpoint will not be marked for expiration or unmounted.

If the mountpoint was not already marked for expiry at that time, an EAGAIN error will be given and it won't be unmounted.

Otherwise if it was already marked and it wasn't referenced, unmounting will take place as usual.

Again, the expiration flag is cleared every time anything other than umount() looks at a mountpoint.