

Native NFSv4 ACLs on Linux

Andreas Grünbacher
agruen@suse.de
SuSE Labs Conference,
Harrachov 2006

September 15, 2006



Novell[®]

Outline

Motivation and Existing Approaches

NFSv4 ACLs and POSIX

Wrap-up

Motivation and Existing Approaches

File Sharing Across OS Boundaries

Many organizations today have heterogeneous Windows/UNIX computing environments

We can gain a lot by supporting those environments optimally
UNIX systems use the POSIX permission model for file access control

Most UNIX systems also support POSIX-draft ACLs these days
Windows uses a considerably different ACL model

Mapping between the two ACL models is difficult:

- Various user-observable mapping artifacts; confused and frustrated users
- In the end, it's not good enough

Can we avoid to map by supporting NFSv4 ACLs natively?

Existing Approaches (1)

NetApp / EMC NAS

Both companies offer multi-protocol NAS devices

- POSIX file permission bits + CIFS ACLs
- The mode / ACL set last “wins”, and determines access

Samba 4 Development

Implements CIFS ACLs in user-space

- Nice for Samba itself
- Other processes will be subject to different permissions => UNIX locked out

Novell NSS ACLs

They don't integrate with Windows; UNIX?; own mgmt. interface

Existing Approaches (2)

IBM JFS2 (AIX), IBM GPFS (Closed Source)

Both support traditional POSIX permissions + NFSv4 ACLs

The mode / ACL set last “wins”, and determines access

Solaris ZFS (GPLv2-incompatible License)

ZFS: weird mapping algorithms, some surprising behavior

Linux NFSv4

Server maps between POSIX-draft and NFSv4 ACLs

- Server exposes POSIX-draft ACLs over the network as NFSv4 ACLs
- Clients see NFSv4 ACLs as extended attributes

Problems With Existing Approaches

- NAS: disabling of file access models is not acceptable
- User-space: implementing NFSv4 ACLs in user-space would not strengthen Linux
- RFC3530 does not specify the interactions between the ACL and the mask well enough
 - Both POSIX-compliant and POSIX-incompatible implementations are allowed under RFC 3530

Why Are NFSv4 ACLs Useful?

They can be implemented in a fully POSIX compliant way

More powerful (and complex) than POSIX-draft ACLs

Some CIFS weirdnesses gone => more appropriate for UNIX

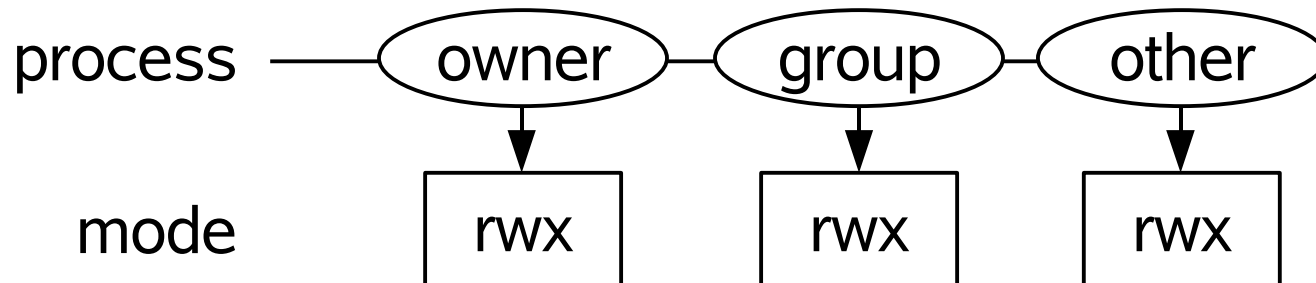
NFSv4 ACLs are close to CIFS ACLs => Mapping is much easier

Standardized by the IETF => somewhat of a neutral ground

NFSv4 ACLs and POSIX

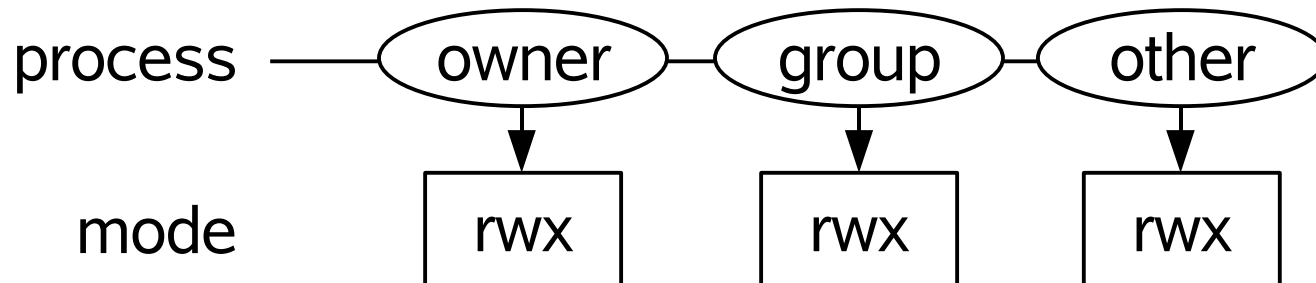
POSIX Permission Model (1)

- Each process either is in the file owner, group, or other class of processes.
- Each of these classes is associated with three permission bits in the file mode.



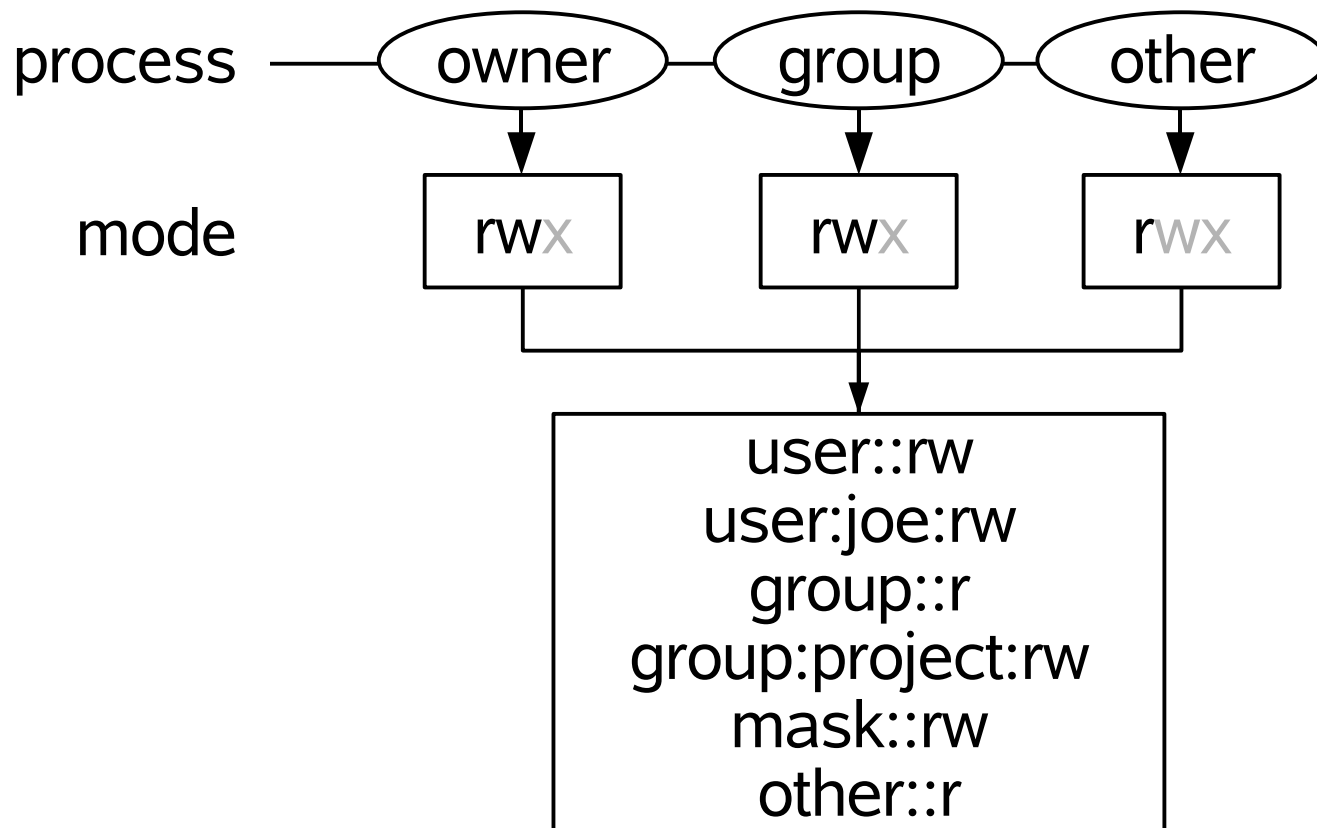
POSIX Permission Model (2)

- For new files and after a `chmod(2)`, each process is granted no more than its associated file permissions allow.
- Permissions that go beyond that may be enabled by explicit user action on a per-file bases. (Alternate File Access Control Mechanism)



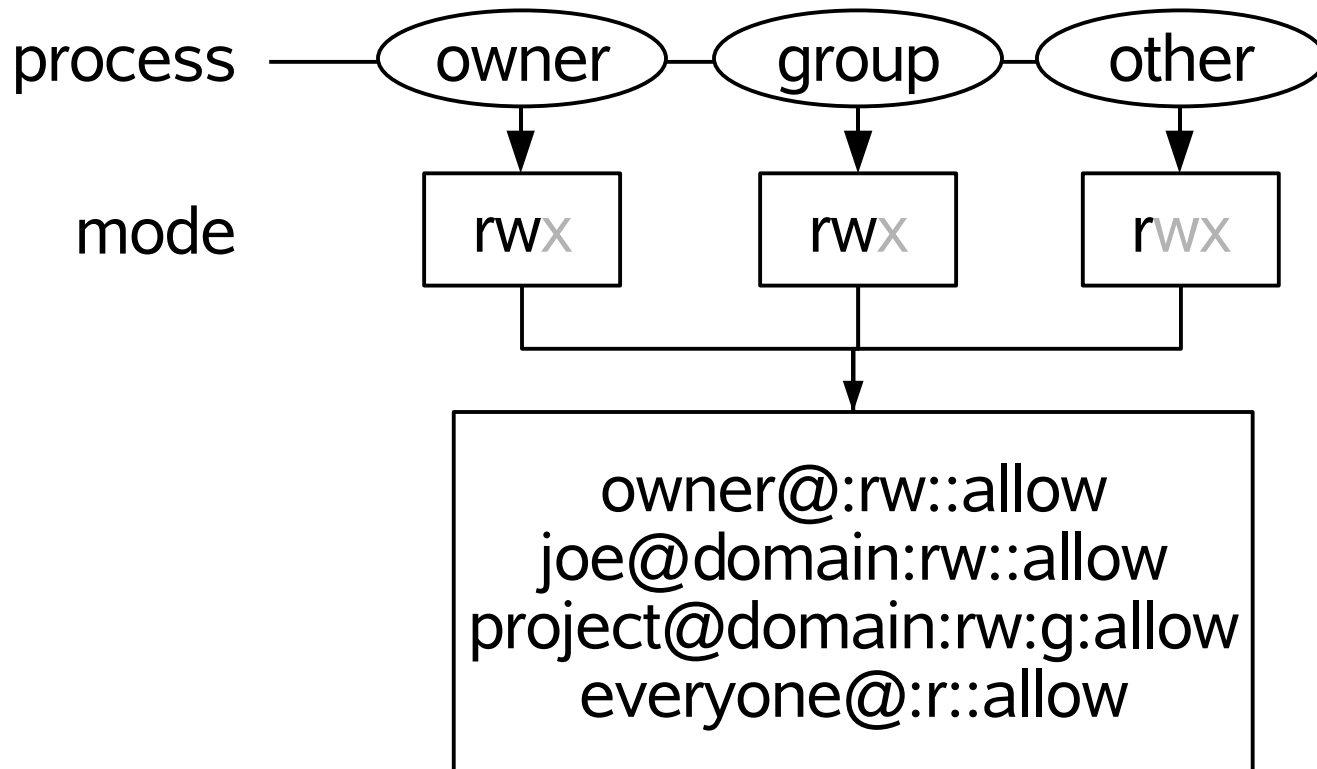
POSIX Permission Model (3)

POSIX ACL



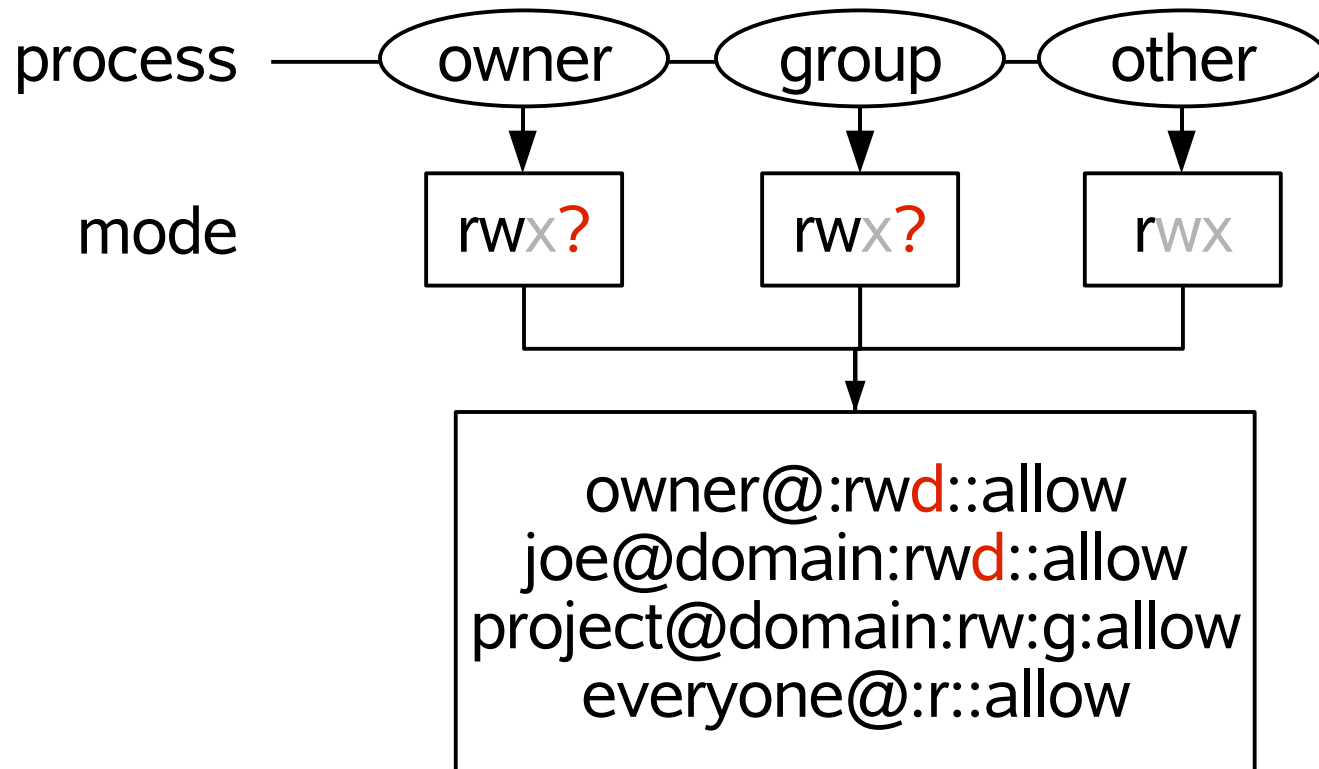
POSIX Permission Model (4)

NFSv4 ACL



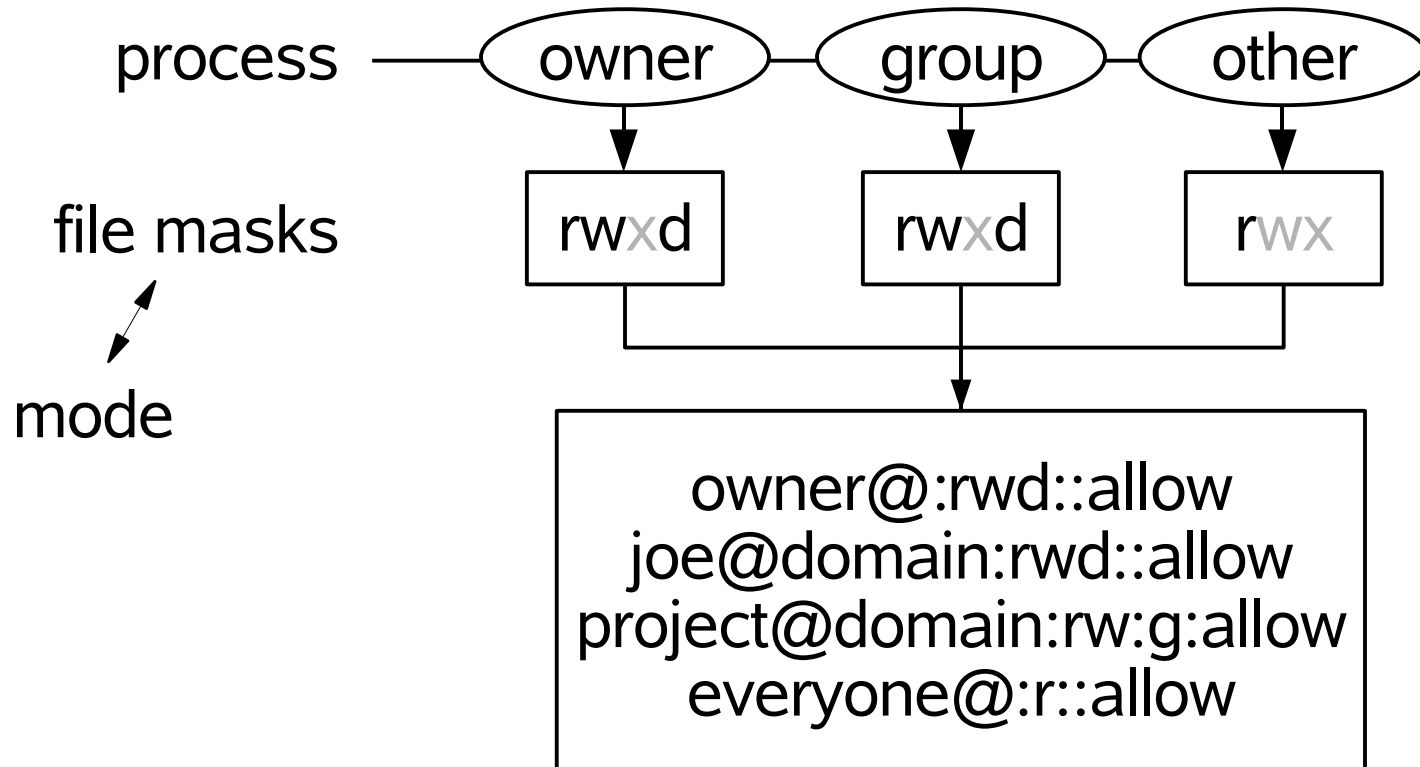
POSIX Permission Model (5)

Alternate Access Control Mechanism: Delete permission?



POSIX Permission Model (6)

File Masks allow to grant more permissions than the file mode can express



Interoperability Goals

Setting the ACL

- The ACL shall be effective => set masks
- The mode shall reflect the ACL as closely as possible

Setting the mode

- Nobody shall have more permissions than the mode allows => set masks
- Setting the mode restrictive and then permissive again shall not lose permissions

Getting the ACL

- The returned ACL must not grant any masked permissions, and should be as close to the effective permissions as possible => somewhat complicated, but a solved problem

Project Website

The Native NFSv4 ACL Project is hosted at
<http://www.suse.de/~agruen/nfs4acl/>

Stuff to be found there

- Design document
- Kernel Patches
- User-land utility + minimal test suite

Wrap-up

Benefits And Drawbacks

- + Better interoperability than anybody else offers today
- + No permission information lost (almost)
- + Makes Linux much better suited as a multi-protocol server and in mixed environments
- + No more excuses to not use Linux as the file server
- + Linux applications can fully be used on the same files

- Second ACL model to understand
- Not all the code is completely trivial
- May still want to map between POSIX ACLs and NFSv4 ACLs – POSIX ACLs are unlikely to go away anytime soon

Status

Working code posted to the LKML last week: NFSv4 ACLs on Ext3

Several things still missing, including:

- Support for other filesystems (ext3 only so far)
- NFSv4 server and client support
- VFS currently only supports MAY_READ, MAY_WRITE, MAY_EXEC, MAY_APPEND

Been working with CITI

- Bruce Fields is supportive
- Trond Myklebust doesn't care a lot I guess

Talking with the Samba team about Samba support

- They are supportive

Conclusions

Good interoperability is very important for us

Native NFSv4 ACLs will significantly improve interoperability

The proposed approach allows to cleanly implement NFSv4 ACLs natively, without breaking POSIX

I hope that the idea will catch on, so that we can successfully grab more of the Windows market!

Questions?

Novell.[®]

