v9fs: Plan 9 Resource Sharing for Linux

About

v9fs is a Unix implementation of the Plan 9 9p remote filesystem protocol.

This software was originally developed by Ron Minnich <rminnich@sandia.gov> and Maya Gokhale. Additional development by Greg Watson <gwatson@lanl.gov> and most recently Eric Van Hensbergen <ericvh@gmail.com>, Latchesar Ionkov <lucho@ionkov.net> and Russ Cox <rsc@swtch.com>.

The best detailed explanation of the Linux implementation and applications of the 9p client is available in the form of a USENIX paper:

> https://www.usenix.org/events/usenix05/tech/freenix/hensbergen.html

Other applications are described in the following papers:

- XCPU & Clustering http://xcpu.org/papers/xcpu-talk.pdf
- KVMFS: control file system for KVM http://xcpu.org/papers/kvmfs.pdf
- CellFS: A New Programming Model for the Cell BE http://xcpu.org/papers/cellfs-talk.pdf
- PROSE I/O: Using 9p to enable Application Partitions http://plan9.escet.urjc.es/iwp9/cready/PROSE_iwp9_2006.pdf
- VirtFS: A Virtualization Aware File System pass-through http://goo.gl/3WPDg

Usage

For remote file server:

```
mount -t 9p 10.10.1.2 /mnt/9
```

For Plan 9 From User Space applications (http://swtch.com/plan9):

```
mount -t 9p `namespace`/acme /mnt/9 -o trans=unix,uname=$USER
```

For server running on QEMU host with virtio transport:

```
mount -t 9p -o trans=virtio <mount_tag> /mnt/9
```

where mount_tag is the tag associated by the server to each of the exported mount points. Each 9P export is seen by the client as a virtio device with an associated "mount_tag" property. Available mount tags can be seen by reading /sys/bus/virtio/drivers/9pnet_virtio/virtio<n>/mount_tag files.

Options

<table>
<tr><td>trans=name</td><td>select an alternative transport. Valid options are currently:

<table>
<tr><td>unix</td><td>specifying a named pipe mount point</td></tr>
<tr><td>tcp</td><td>specifying a normal TCP/IP connection</td></tr>
<tr><td>fd</td><td>used passed file descriptors for connection (see rfdno and wfdno)</td></tr>
<tr><td>virtio</td><td>connect to the next virtio channel available (from QEMU with trans_virtio module)</td></tr>
<tr><td>rdma</td><td>connect to a specified RDMA channel</td></tr>
</table>
</td></tr>
<tr><td>uname=name</td><td>user name to attempt mount as on the remote server. The server may override or ignore this value. Certain user names may require authentication.</td></tr>
<tr><td>aname=name</td><td>aname specifies the file tree to access when the server is offering several exported file systems.</td></tr>
</table>

<table>
<tr><td>cache=mode</td><td>specifies a caching policy. By default, no caches are used.

none
default no cache policy, metadata and data alike are synchronous.
loose
no attempts are made at consistency, intended for exclusive, read-only mounts
fscache
use FS-Cache for a persistent, read-only cache backend.
mmap
minimal cache that is only used for read-write mmap. Northing else is cached, like cache=none</td></tr>
<tr><td>debug=n</td><td>specifies debug level. The debug level is a bitmask.

<table><tr><td>0x01</td><td>display verbose error messages</td></tr><tr><td>0x02</td><td>developer debug (DEBUG_CURRENT)</td></tr><tr><td>0x04</td><td>display 9p trace</td></tr><tr><td>0x08</td><td>display VFS trace</td></tr><tr><td>0x10</td><td>display Marshalling debug</td></tr><tr><td>0x20</td><td>display RPC debug</td></tr><tr><td>0x40</td><td>display transport debug</td></tr><tr><td>0x80</td><td>display allocation debug</td></tr><tr><td>0x100</td><td>display protocol message debug</td></tr><tr><td>0x200</td><td>display Fid debug</td></tr><tr><td>0x400</td><td>display packet debug</td></tr><tr><td>0x800</td><td>display fscache tracing debug</td></tr></table></td></tr>
<tr><td>rfdno=n</td><td>the file descriptor for reading with trans=fd</td></tr>
<tr><td>wfdno=n</td><td>the file descriptor for writing with trans=fd</td></tr>
<tr><td>msize=n</td><td>the number of bytes to use for 9p packet payload</td></tr>
<tr><td>port=n</td><td>port to connect to on the remote server</td></tr>
<tr><td>noextend</td><td>force legacy mode (no 9p2000.u or 9p2000.L semantics)</td></tr>
<tr><td>version=name</td><td>Select 9P protocol version. Valid options are:

<table><tr><td>9p2000</td><td>Legacy mode (same as noextend)</td></tr><tr><td>9p2000.u</td><td>Use 9P2000.u protocol</td></tr><tr><td>9p2000.L</td><td>Use 9P2000.L protocol</td></tr></table></td></tr>
<tr><td>dfltuid</td><td>attempt to mount as a particular uid</td></tr>
<tr><td>dfltgid</td><td>attempt to mount with a particular gid</td></tr>
<tr><td>afid</td><td>security channel - used by Plan 9 authentication protocols</td></tr>
<tr><td>nodevmap</td><td>do not map special files - represent them as normal files. This can be used to share devices/named pipes/sockets between hosts. This functionality will be expanded in later versions.</td></tr>
<tr><td>access</td><td>there are four access modes.

user
if a user tries to access a file on v9fs filesystem for the first time, v9fs sends an attach command (Tattach) for that user. This is the default mode.
<uid>
allows only user with uid=<uid> to access the files on the mounted filesystem
any
v9fs does single attach and performs all operations as one user
clien
ACL based access check on the 9p client side for access validation</td></tr>
<tr><td>cachetag</td><td>cache tag to use the specified persistent cache. cache tags for existing cache sessions can be listed at /sys/fs/9p/caches. (applies only to cache=fscache)</td></tr>
</table>

Behavior

This section aims at describing 9p 'quirks' that can be different from a local filesystem behaviors.

- Setting O_NONBLOCK on a file will make client reads return as early as the server returns some data instead of trying to fill the read buffer with the requested amount of bytes or end of file is reached.

Resources

Protocol specifications are maintained on github: http://ericvh.github.com/9p-rfc/

9p client and server implementations are listed on http://9p.cat-v.org/implementations

A 9p2000.L server is being developed by LLNL and can be found at http://code.google.com/p/diod/

There are user and developer mailing lists available through the v9fs project on sourceforge (http://sourceforge.net/projects/v9fs).

News and other information is maintained on a Wiki. (http://sf.net/apps/mediawiki/v9fs/index.php).

Bug reports are best issued via the mailing list.

For more information on the Plan 9 Operating System check out http://plan9.bell-labs.com/plan9

For information on Plan 9 from User Space (Plan 9 applications and libraries ported to Linux/BSD/OSX/etc) check out https://9fans.github.io/plan9port/