Solaris CIFS Service

"Seamless, ubiquitous, cross-protocol file sharing"

Jarod Nash
Revenue Product Engineering,
Sun Microsystems, Inc.

Solaris CIFS Server: TOI Overview

- Background
- Workgroup Demo
  - Explanation of Workgroup Mode setup
  - Including sharemgr
- Domain Demo
  - Domain Model setup
  - Including identity mapping (idmap)
- Other CIFS changes in Solaris

Solaris CIFS Server: Background

- "Seamless, ubiquitous, cross-protocol file sharing"
  Alan Wright, Project Lead for CIFS Server
- CIFS server is now a first class citizen in Solaris
  - Putback into Development/Nevada October 2007
    - Available in Solaris Express and OpenSolaris 2008.03
    - 25+ ARC cases, 800 files, approximately 370,000 lines of code (including 180,000 lines of new code)
- Tight integration with NFS, ZFS, and Active Directory
  - Windows/CIFS concepts such as Security Identifiers and Access Tokens are now native to Solaris kernel

Workgroup Mode Demo

Workgroup Setup: Things to Know

- Only works with local /etc/passwd users
  - NIS/NIS+/LDAP not supported
- "Works out of the box" - not quite true:
  - PAM modification required to generate SMB passwords
  - SMB passwords need to be generated for local accounts
    - Only needed to be done at setup, after this the PAM change takes care of password updating automatically
- Windows "GUEST" account not supported
  - Samba: guest ok = yes
  - All access must be via *authenticated* local accounts

Workgroup Setup

- Enable CIFS Server (along with dependencies)
  - `svcadm enable -r smb/server`
- Set mode for SMB server
  - Not needed if using default workgroup "WORKGROUP"
  - `smbadm join -w WORKGROUP`
- Add following line to pam.conf – this will generate SMB style passwords for local users:
  - `other password required pam_smb_passwd.so.1 nowarn`
- Generate SMB passwords for existing local users
- Done!

sharemgr(1m)

- New share management admin interface
- Manages NFS and CIFS shares
  - To a large extent, NFS and CIFS behave similarly
- sharemgr works with *groups* of shares associated with sharing protocols
  - Protocols may be nfs, smb or both
    - Default is both
- Two automatic groups:
  - default - legacy share(1m) interface
  - zfs - shares created via zfs(1) interface

share(1m)/sharemgr(1m) Examples

```
# cat /etc/dfs/dfstab
...
share -F nfs -o ro /export/install
# sharemgr show -vp default
default nfs=()
          /export/install   nfs=() nfs:sys=(ro="*")
```

Traditional dfstab entry

Auto-generated “default” sharemgr group entry

```
# sharemgr create -P nfs home
# sharemgr add-share -s /export/home home
# sharemgr show -vp home
home nfs=()
          /export/home
# cat /etc/dfs/dfstab
...
share -F nfs /export/home @home
```

Create group, then add share

Auto-generated dfstab entry

zfs(1m)/sharemgr(1) Examples

```
# zfs list | grep pool/home
pool/home           57K  9.84G    21K  /pool/home
pool/home/jarod     18K  9.84G    18K  /pool/home/jarod
pool/home/rayh      18K  9.84G    18K  /pool/home/rayh
# zfs sharenfs=on pool/home
# sharemgr show -vp zfs
zfs
    zfs/pool/home nfs=()
          /pool/home
          /pool/home/jarod
          /pool/home/rayh
```

ZFS simple administration

Auto-generated “zfs” group with 1 share and 3 resources

zfs(1m)/sharemgr(1) Examples

```
# zfs list | grep install
pool/install              57K  9.84G  21K /pool/install
pool/install/snv_95       18K  9.84G  18K /pool/install/snv_95
pool/install/snv_96       18K  9.84G  18K /pool/install/snv_96
# zfs sharenfs=ro pool/install
# sharemgr show -vp zfs
zfs
    zfs/pool/home nfs=()
          /pool/home
          /pool/home/jarod
          /pool/home/rayh
    zfs/pool/install nfs=() nfs:sys=(ro="*")
          /pool/install
          /pool/install/snv_95
          /pool/install/snv_96
```

Share readonly

This shows the 2 shares in "zfs" group with differing protocol security settings

sharemgr(1m) and smb protocol

- Differences in sharing behaviour/functionality between NFS and CIFS:
  - SMB shares use resource names while NFS shares are path based; for NFS, resource is merely an alias
    - SMB: jarod=/export/home/jarod
    - NFS: /export/home/jarod=/export/home/jarod
  - NFS only allows a given path to be shared once, while SMB allows it to be shared multiple times
    - This is handled with SMB via differing resource names
  - If an NFS path is shared, it is not also possible to share parent or any subdirectory (in same filesystem). SMB allows this
    - This is flagged in sharemgr output with “not-shared-with=”

smb/sharemgr(1) Examples

```
# sharemgr create home
# sharemgr add-share -s /export/home/jarod home
Resource name is required by at least one enabled protocol
in group
# sharemgr add-share -r jarod -s /export/home/jarod home
# sharemgr add-share -r home -s /export/home home
# sharemgr add-share -r newhome -s /export/home home
# sharemgr show -vp home
home smb=() nfs=()
          jarod=/export/home/jarod
          /export/home  not-shared-with=[nfs]
                  newhome=/export/home
                  home=/export/home
```

SMB protocol requires "resource name"

Parent directory /export/home is shared twice

Parent directory and double sharing reflected here

zfs/smb/sharemgr(1) Examples

```
# zfs list pool/home/jarod
NAME                USED  AVAIL  REFER  MOUNTPOINT
pool/home/jarod      18K   134G    18K  /pool/home/jarod
# zfs set sharesmb=on pool/home/jarod
# sharemgr show -vp zfs
zfs
    zfs/pool/home/jarod smb=()
          pool_home_jarod=/pool/home/jarod
# zfs set sharesmb=name=jarod pool/home/jarod
# sharemgr show -vp zfs
zfs
    zfs/pool/home/jarod smb=()
          /pool/home/jarod
                jarod=/pool/home/jarod
                pool_home_jarod=/pool/home/jarod
```

Create default SMB share with path based resource

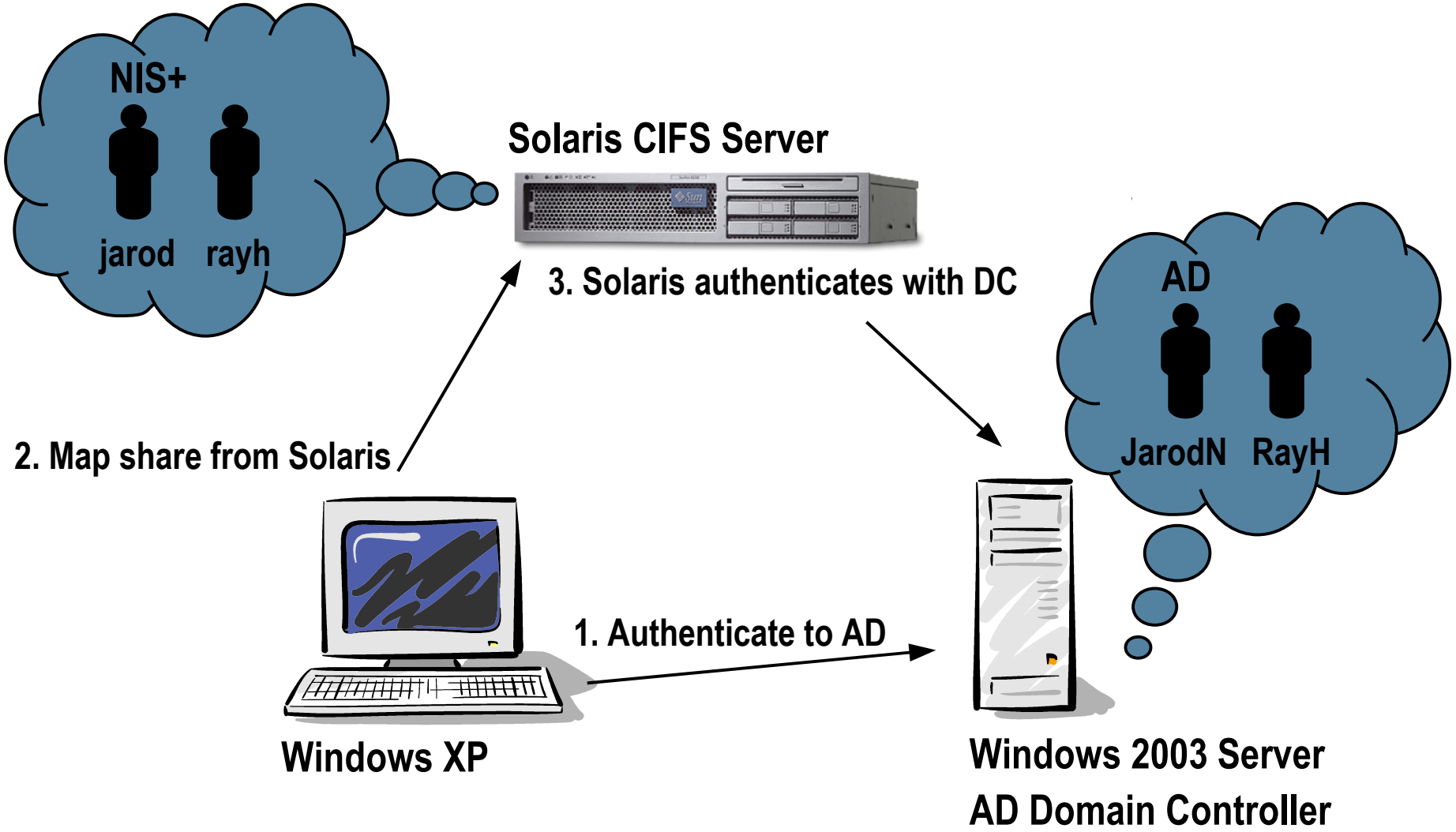
Use pseudo property “name” to set SMB resource value`

SMB resource values

Domain Mode Setup: Things to Know

- Integrates with Windows Active Directory (AD)
  - CIFS Server joins AD domain and talks to Domain Controller (DC) to authenticate access automatically
- Identity mapping between Windows Users and Unix Users performed by idmap service
- ZFS strongly recommended
  - UFS unchanged to support CIFS and lacks functionality
    - eg Windows Identity semantics, ACLs, case-insensitivity

Domain Mode Setup: Demo Network



Domain Mode Setup #1

- Enable CIFS Server (along with dependencies)
  - `svcadm enable -r smb/server`
- Set SMB mode: DNS and Kerberos
  - Configure /etc/resolv.conf:
    - Domain set to AD domain, nameserver set to DC IP
  - Configure Kerberos: (AD uses Kerberos authentication)
    - In the /etc/krb5/krb5.conf file, specify the fully qualified AD domain name, in uppercase characters, as the default realm. Also, specify the fully qualified host name of the domain controller as the value for the kdc, admin_server, and kpasswd_server parameters. kpasswd_protocol is set to SET_CHANGE.

Domain Mode Setup #2

- Sync clocks
  - Solaris CIFS Server system needs to be within five minutes of the system clock of the DC
    - Kerberos requirement
  - `ntpdate` *`DC-host`*
- Join AD using a user with Domain Administrator privileges, domain-name is a fully qualified domain name:
  - `smbadm join -u` *`username domain-name`*
- Restart CIFS Server (known bug)
  - `svcadm restart smb/server`

Domain Mode Setup #3: ID Mapping

- Windows AD needs to be mapped to Solaris namespace for true interoperability, but not a requirement
- idmap provides a number of mapping mechanisms:
  - Fixed - Windows “Well Known” Identities
  - Directory - AD or LDAP explicit mappings
  - Local - Solaris CIFS Server idmap rules
  - Ephemeral and Local-SID
    - More about this in a few slides
- Daemon and Admin:
  - idmapd(1m): cache and dynamic ID allocation
  - idmap(1m): Mapping lookups, rule manipulation

Domain Mode Setup #4

- Create idmap rule for all Windows users in all domains to Unix users with same names:
  - `idmap add -d winname:"*@*" unixuser:"*"`
- Create idmap rule to fix up username mismatch:
  - `idmap add winuser:JarodN unixuser:jarod`
- Done!

- NOTE:
  - These rules are not generic, just for demo

Domain Mode Demo

Sorting ACLs

- Windows GUI needs ACLs to be sorted
  - Deny ACEs should appear before allow
- ZFS trivial ACL which represents traditional Unix permission bits is not sorted
- If a file's ACL is viewed and saved by a Windows client, the ACL will be sorted which will change the file's effective permissions

Sorting ACLs Example

File's ACLs viewed and saved by Windows Client. Order changed, Unix permissions now different

```
$ touch file
$ ls -v file
-rw-r--r--   1 jarod    techies        0 Sep 24 10:24 file
     0:owner@:execute:deny
     1:owner@:read_data/write_data/append_data/write_xattr/write_attributes/write_acl/write_owner:allow
     2:group@:write_data/append_data/execute:deny
     3:group@:read_data:allow
     4:everyone@:write_data/append_data/write_xattr/execute/write_attributeswrite_acl/write_owner:deny
     5:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow

$ ls -v file
-r--r--r--+  1 jarod    techies        0 Sep 24 10:24 file
     0:everyone@:write_data/append_data/write_xattr/execute/write_attributes/write_acl/write_owner:deny
     1:user:jarod:execute:deny
     2:group:techies:write_data/append_data/execute:deny
     3:everyone@:read_data/read_xattr/read_attributes/read_acl/synchronize:allow
     4:user:jarod:read_data/write_data/append_data/write_xattr /write_attributes/write_acl/write_owner/
synchronize:allow
     5:group:techies:read_data/synchronize:allow
```

SID: Security Identifier

S-1-5-21-4020443685-79182764-1390981100-1134

- CIFS equivalent of POSIX UIDs/GIDs
- Universally unique identifier for a user or group
  - Effectively an infinite number of unique users
    - As long as they are in sets of 4 billion per system or AD domain
    - In practice no domains exist with even billions of users/groups
  - SID encodes system or AD, plus user RID suffix
    - RID = Relative Identifier
- When data is moved between systems, ownership attributes/ACEs retain semantic meaning
  - System or AD portion of SID ensures this
  - Unix suffers from UID/GID reuse

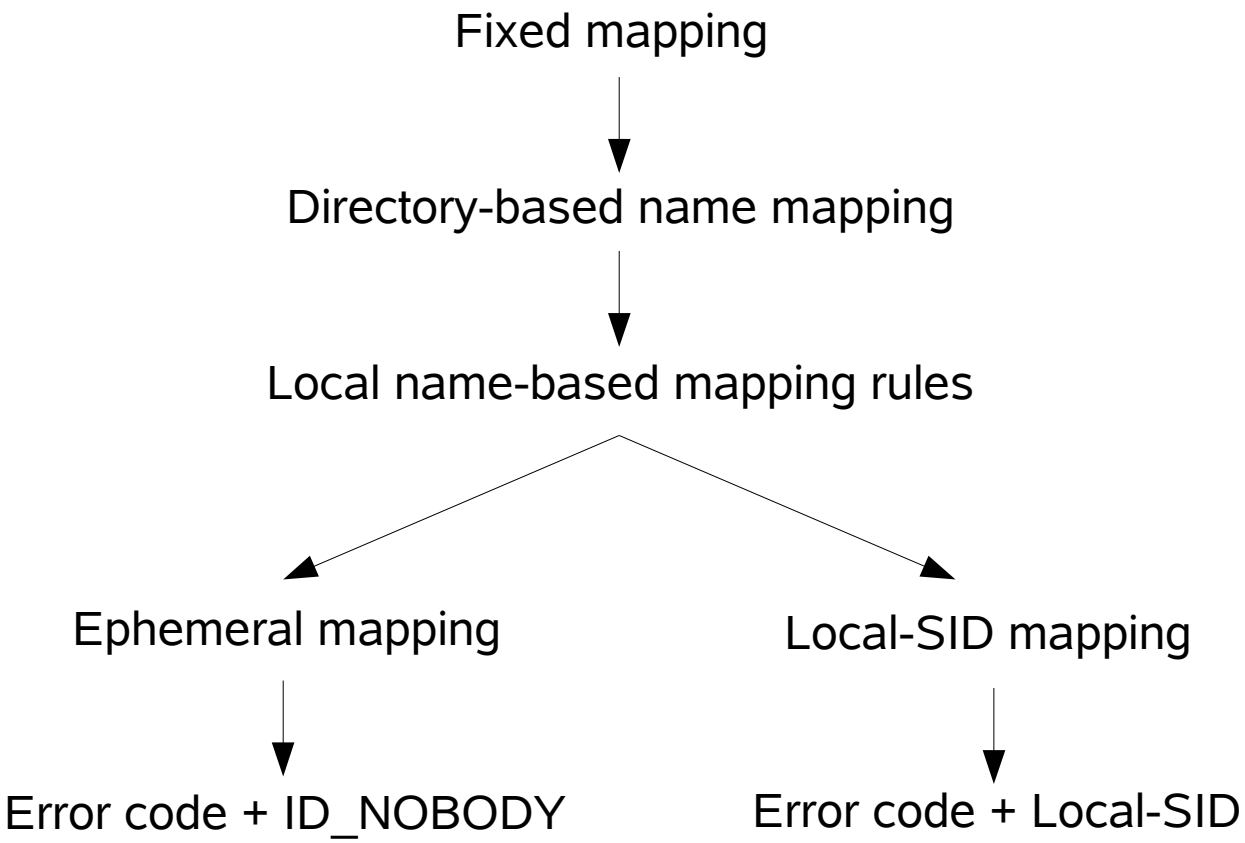
SIDs and UIDs/GIDs

- Sharing protocol interoperability has seen several solutions to the SID/UID/GID storing/mapping problem:
  - 1. Store SID, map to/from UID/GID
    - Microsoft Services for Unix (SFU)
  - 2. Store UID/GID, map to/from SID
    - Samba offers name mapping, algorithmic, namesevice entries
  - 3. Store both
    - Netapp's WAFL uses NTFS/Unix “qtrees” which store one and map the other ID

Identity Mapping: The Sun Way

- Maintain CIFS ownership semantics
  - Don't rely on mapping to recover SID, store the SID!
    - Encoded as FUIDs (Filesystem Unique IDs) in ZFS ACLs as ZFS ACLs are used to represent CIFS Security Descriptors (SDs)
- Make Administration and Setup easy
  - No requirement for large scale SID/UID mapping setup
  - No UID partitioning
    - eg take RID, add 70,000 => UID (Samba's algorithmic approach)
  - Preserve POSIX identity model
    - Can't insist on SIDs, especially if no AD being used!
- Flexible

Mapping Mechanisms: Overview



Identity Mapping: Ephemeral

- Unknown Windows identities mapped to dynamically allocated UIDs/GIDs
  - Uses next available UID/GID from 2^{31} to $2^{32} - 2$ (ephemeral UIDs/GIDs)
  - Solaris UIDs/GIDs are now unsigned, but max out at MAX_INT, leaving the unsigned range above for ephemerals
- Not retained across reboots
- Stored in SQLite 2.0 DB /var/run/idmap/idmap.db
  - Preserved across service restarts
  - If lost then previously allocated ephemeral uids/gids never reused (unless rebooted); idmapd uses in-kernel state to determine bottom of valid range of IDs

Ephemeral Mapping Example

```
$ idmap show -c winuser:john
winuser:john@example.com -> uid:2147508225

$ idmap show -c uid:2147508225 winuser
uid:2147508225 -> winuser:john@example.com
```

Suppose a rule is added to map john to Unix user jd123456

```
$ idmap add winuser:john@example.com unixuser:jd123456

$ idmap show -c winuser:john uid
winuser:john@example.com -> uid:123456
```

Ephemeral uid 2147508225 still maps to john

```
$ idmap show -c uid:2147508225 winuser
uid:2147508225 -> winuser:john@example.com
```

Identity Mapping: Local SIDs

- If a non-ephemeral UID/GID cannot be mapped by name, then it is mapped to an algorithmically generated SID called a local-SID
- The local-SID is generated as follows:
  - local-SID for UID = <machine SID> - <1000 + UID>
  - local-SID for GID = <machine SID> - <2^{31} + GID>
- <machine SID> is a unique SID generated by the idmap service for the host on which it runs

Unified Access Control Model

- Security Descriptors (SDs) used by CIFS for protection
  - Encoded as ACLs in ZFS or UFS with differing results:
    - ZFS ACLs similar to Windows ACLs
    - UFS ACLs have no explicit deny or audit ACEs, only 3 permission bits and different access check algorithm – USE ZFS!
- SDs have separate lists for access and audit entries
  - ZFS stores all entries in one list
  - Split/merged appropriately by Solaris CIFS during translation
- ZFS ACL inheritance different for POSIX and CIFS
  - Default ZFS behavior accommodates POSIX
  - CIFS service applies Windows inheritance rules for CIFS operations

ZFS Enhancements for CIFS

- Case Insensitivity
  - Not required by SMB protocol, but some Windows applications rely on case-insensitive behaviour
  - At filesystem creation time, can specify *sensitive, insensitive* or *mixed* (both behaviours)
  - foo.txt and Foo.txt not distinguishable to CIFS client
    - Name mangling used to resolve conflicts, for example:
      - foo.txt -> FOO~8.txt
      - Foo.txt -> FOO~9.txt
  - CIFS clients can open original name: returned file will be the first case-insensitive match

Alternate Data Streams

- NTFS treats files as a collection of streams
  - Regular file content is the *default* or *unnamed* stream
  - AKA a *Resource Fork*
- Named streams used to store arbitrary data
  - Metadata (statistics, notes, history) for documents
  - Viruses!
- Implemented using Solaris extended attributes
  - SUNWsmb prefix assigned to CIFS stream names
    - SUNWsmb prefix not seen by CIFS clients (use runat(1) on Solaris)
  - CIFS service explicitly assigns mode 0400
  - Unnamed stream UID/GID assigned to stream xattr file

Extensible Attributes

- Not to be confused with extended attributes
  - Extensible attributes maintained in extended attribute files:
    SUNWattr_ro - readonly system attributes
    SUNWattr_rw - read/write system attributes
- Supports DOS attributes
  - Archive, Readonly, Hidden, System
- Same API used by CIFS also supports other systems:
  - MacOS/BSD, eg
    - IMMUTABLE, APPENDONLY
  - Solaris Virus Scanning (VSCAN ICAP implementation)
    - AV_QUARANTINED, AV_MODIFIED

ls(1) Changes
NOTE: OpenSolaris has /usr/gnu/bin before /bin

```
# ls -/ c test.txt
-rw-r—r--   1 root   root   0 Jan 14 16:51 test.txt
              {A-RS---m--}
# ls -/ v test.txt
-rw-r--r-- 1 root   root   0 Jan 14 16:51 test.txt
              {archive,nohidden,readonly,system,noappend
only,nonodump,noimmutable,av_modified,noav_quarantined,non
ounlink}
# ls -l -% all file
-rw-r—r--   1 root   root   0 Jan 14 16:51 test.txt
              timestamp: atime          Jan 14 16:51:16 2008
              timestamp: ctime          Jan 14 16:53:07 2008
              timestamp: mtime          Jan 14 16:51:16 2008
              timestamp: crtime         Jan 14 16:51:16 2008
```

NOTE: Two part switch to ls
Some may find this ugly!

chmod(1) Changes
NOTE: OpenSolaris has /usr/gnu/bin before /bin

```
# chmod S+cRS test.txt
# chmod S+v'{readonly,system}' test.txt
# ls -/c
-rw-r-r--  1 root  root  0 Jan 14 16:51 test.txt
           {A-RS---m--}
# chmod S-cRS test.txt
# chmod S-v'{readonly,system}' test.txt
# ls -/c
-rw-r-r--  1 root  root  0 Jan 14 16:51 test.txt
           {A------m--}
```

Use both compact (+c) and verbose (+v) forms to set the System Attributes Readonly and System

- tar(1), cpio(1) and pax(1) are aware of the new system attributes and have appropriate switches for archiving/unpacking

References

- Sun Documentation:
  - docs.sun.com/app/docs/doc/820-2429
- OpenSolaris site:
  - opensolaris.org/os/project/cifs-server/
  - opensolaris.org/os/project/cifs-server/docs/
- Genunix Guide to Solaris CIFS:
  - www.genunix.org/wiki/index.php/Getting_Started_With_the_Solaris_CIFS_Service
  - www.genunix.org/wiki/index.php/Solaris_CIFS_Service_Troubleshooting
- Developer Recipes: Setting up NAS
  - developers.sun.com/openstorage/articles/opensolaris_nas.html

Acknowledgements

- Sun Internal CIFS/Windows Interoperability TOIs
- Numerous blogs
- Sun Documents
- Stacey Marshall, Owen Roberts and Ray Hassan for their important help in assembling the material and providing feedback

microsystems

Solaris CIFS Service

"Seamless, ubiquitous, cross-protocol file sharing"

Jarod Nash
Revenue Product Engineering,
Sun Microsystems, Inc.

Diagnostic Information

- Use chkcfg.sh and gendiag.sh
  - `opensolaris.org/os/project/cifs-server/files/`
- Diagnostic Information to Gather
  - `sharemgr show -vp`
  - `sharectl get smb`
  - `smbadm list`
  - `zfs get all`
  - `/etc/krb5/krb5.conf`
  - `/etc/pam.conf`
  - `/etc/resolv.conf`
  - Network captures (wireshark, netmon)
  - Dtrace output
  - Environment (uname -a, Client OS, version and service packs