z/OS UNIX File System Administration

Ann Totten, atotten@us.ibm.com
IBM Corporation

Friday, March 4, 2011: 9:30 AM-10:30 AM
Session Number 9040

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

IBM
Language Environment
z/OS

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Session Topics

- Discussion on the supported PFS types in z/OS UNIX
- Recommended file hierarchy structure
- File system administration
- File security
- New support introduced in z/OS Release 12

BPXPRMxx updates

Defining file systems

- Customize the FILESYSTYPE, ROOT, MOUNT, NETWORK, and SUBFILESYSTYPE statements to specify your file systems. These statements define the file systems at OMVS initialization.

The FILESYSTYPE statement defines the TYPE of physical file system.

FILESYSTYPE
TYPE(type_name) ENTRYPOINT(entry_name) PARM('parm')
ASNAME(proc_name[,'start_parms'])

Typical file systems are:

- **AUTOMNT - Handles automatic mounting and unmounting of filesystems.**
  - **Module name – BPXTAMD**
- **ZFS - Handles Distributed File Service zSeries file system requests.**
  - **Module Name – IOEFSCM**
- **TFS - Handles requests to the temporary file system (TFS).**
  - **Module Name – BPXTFS**
- **HFS - Needed for regular local files requests in a HFS.**
  - **Module Name – GFUAINIT**
- **NFS - Handles requests for access to remote files.**
  - **Module Name – GFSCINIT**

BPXPRMxx member, continued – ROOT and MOUNT statements

The ROOT statement defines and mounts the root file system for a hierarchical file system.

```
ROOT FILESYSTEM('fsname')|DDNAME(ddname) TYPE(type_name) MODE(access)
    PARM('parameter') SETUID|NOSETUID AUTOMOVE[(INCLUDE|
    EXCLUDE,sysname1,sysname2,...,sysnamen)]|NOAUTOMOVE|UNMOUNT
    SYSNAME(sysname) TAG(NOTEXT|TEXT,ccsid) MKDIR(mpt1)
```

MOUNT specifies a file system that z/OS UNIX is to logically mount onto the root file system or another file system. Mount statements are processed in the sequence in which they appear.

```
MOUNT FILESYSTEM('fsname')|DDNAME(ddname) TYPE(type_name)
    MOUNTPOINT('pathname') MODE(access) PARM('parameter') TAG(NOTEXT|TEXT,ccsid)
    SETUID|NOSETUID SECURITY|NOSECURITY AUTOMOVE[(INCLUDE|
    EXCLUDE,sysname1,sysname2,...,sysnamen)]|NOAUTOMOVE|UNMOUNT
    SYSNAME(sysname) MKDIR(mpt1)
```

Display command for Physical File System information

```
D OMVS,PFS
BPXO068I 11.29.40 DISPLAY OMVS 888
 OMVS     0010 ACTIVE              OMVS=(ST,RC)
 PFS CONFIGURATION INFORMATION
  PFS TYPE   ENTRY      ASNAME    DESC      ST    START/EXIT TIME
   TFS1      BPXTFS     OOKASPT1  LOCAL     A     2009/08/23 21.47.42
   NFS       GFSCINIT   MVSNFSCL  REMOTE    A     2009/08/23 21.47.41
   CINET     BPXTCINT             SOCKETS   A     2009/08/23 21.47.41
   AUTOMNT   BPXTAMD              LOCAL     A     2009/08/23 21.47.41
   UDS       BPXTUINT             SOCKETS   A     2009/08/23 21.47.41
   ZFS       IOEFSCM    ZFS       LOCAL     A     2009/08/23 21.47.31
   HFS       GFUAINIT             LOCAL     A     2009/08/23 21.47.31

  PFS TYPE  DOMAIN        MAXSOCK  OPNSOCK  HIGHUSED
   CINET    AF_INET6        65535       52        58
            AF_INET         65535       61        67
   UDS      AF_UNIX         10000       20        20

 ..... output continued on next page
```

Display command for Physical File System information

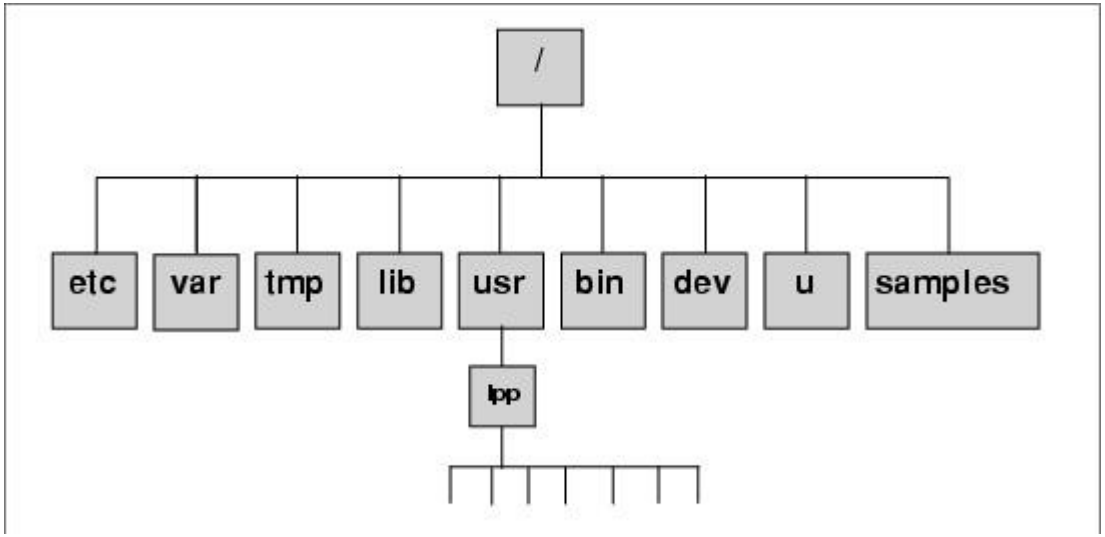

```
….from previous page

  SUBTYPES OF COMMON INET
PFS NAME    ENTRY       START/EXIT TIME        STATUS   FLAGS
 TCP341      EZBPFINI    2009/08/23 21.52.02    ACT      CD
 TCP342      EZBPFINI    2009/08/23 21.52.06    ACT
 TCP343      EZBPFINI    2009/08/23 21.51.55    ACT
 TCP344      EZBPFINI                           INACT

PFS TYPE      FILESYSTYPE PARAMETER INFORMATION
 NFS           AttrCaching(Y)
 ZFS           PRM=(ST,S1)
 HFS           SYNCDEFAULT(30) VIRTUAL(2560) FIXED(100)
               CURRENT VALUES: FIXED(100) VIRTUAL(2560)

PFS TYPE   STATUS INFORMATION
 AUTOMNT   TIME=2009/08/24 21:11:52 SYSTEM=NPF      USER=SETUP
           POLICY=/etc/auto.master
```

Hierarchical file system concepts

- The hierarchical file system consists of the following:
  - Files contain data or programs. A file containing a load module or shell script or REXX program is called an executable file. Files are kept in directories.
  - Directories contain files, other directories, or both. Directories are arranged hierarchically, in a structure that resembles an upside-down tree, with the root directory at the top and the branches at the bottom. The **root** is the first directory for the file system at the top of the tree and is designated by a slash (/).
  - Additional local or remote file systems, which are mounted on directories of the root file system or of additional file systems.
- z/OS UNIX files are organized in a hierarchical file system as in other UNIX systems.

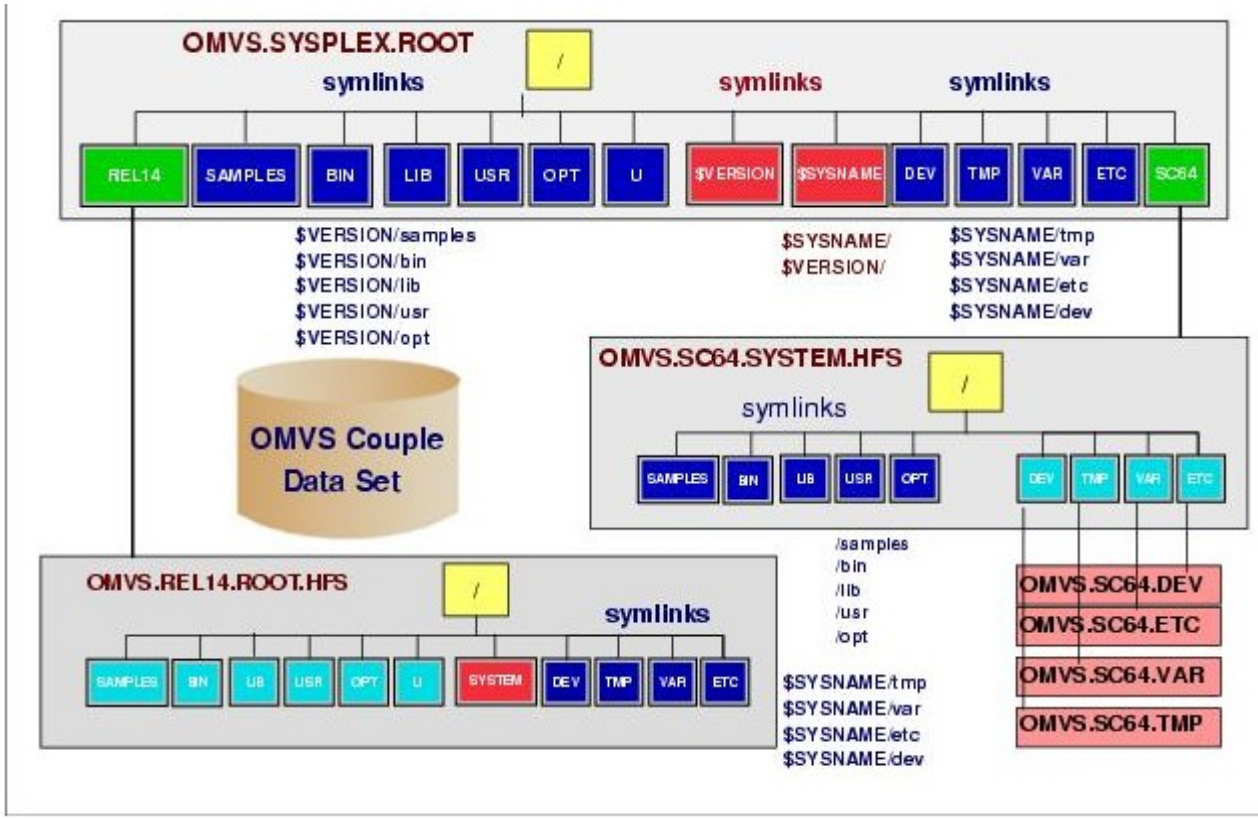
Hierarchical file system concepts

- Figure 6-1 Logical view of the z/OS UNIX file structure. Source: Redbook: UNIX System Services z/OS Version 1 Release 7 Implementation (ISBN 073849609X - IBM Form Number SG24-7035-01)



Hierarchical file system concepts

- Figure 6-27 All the z/OS UNIX file sharing structures used in a sysplex sharing environment. Source: Redbook: UNIX System Services z/OS Version 1 Release 7 Implementation (ISBN 073849609X - IBM Form Number SG24-7035-01)



Display command for Mounted File System information

Use DISPLAY OMVS,FILE to display status of all mounted file systems

```
D OMVS,FILE
BPXO045I 11.40.31 DISPLAY OMVS 548
OMVS      0010 ACTIVE              OMVS=(ST,RD)
TYPENAME   DEVICE ----------STATUS----------- MODE  MOUNTED    LATCHES
TFS1           74 ACTIVE                      RDWR  06/30/2010  L=95
  NAME=OMVSSPA.SVT.S8.TMP.TFS                       08.54.11    Q=0
  PATH=/NPB/tmp
  MOUNT PARM=       -s 4000
  OWNER=NPB       AUTOMOVE=U CLIENT=Y
ZFS             1 ACTIVE                      READ  06/30/2010  L=14
  NAME=OMVSSPA.SVT.SYSPLEX.ZFS                      08.43.26    Q=0
  PATH=/
  OWNER=NP4       AUTOMOVE=Y CLIENT=N
HFS            81 ACTIVE                      RDWR  06/30/2010  L=102
  NAME=OMVSSPA.TOTTEN.HFS4                          08.59.12    Q=0
  PATH=/u/totten/hfs04
  OWNER=NP7       AUTOMOVE=Y CLIENT=Y
```


SHARE in Anaheim 2011

Display OMVS,FILE,filter

Use filters to see only the file systems that you want

`D OMVS,FILE,O`

- Displays mounted file systems that are z/OS UNIX owned on the system where the command was issued

`D OMVS,FILE,O=sysname`

- Displays mounted file systems that are z/OS UNIX owned on the system named *sysname*

`D OMVS,FILE,N=OMVSSPA.*`

- Displays mounted file systems that have a name that matches the pattern

`D OMVS,FILE,T=ZFS`

- Displays mounted file systems that are of type ZFS

`D OMVS,FILE,E`

- Displays mounted file systems that are in an exception state (QUIESCED, UNOWNED, etc).

SHARE in Anaheim 2011

Display OMVS,MF

Use this display command to view the 10 most recent mount failures

```
D OMVS,MF
BPXO058I 14.21.04 DISPLAY OMVS 329
OMVS     0010 ACTIVE              OMVS=(ST,RD)
SHORT LIST OF FAILURES:
TIME=08.54.11  DATE=2010/06/30        MOUNT RC=0081  RSN=1288005C
  NAME=OMVSSPA.SVT.JAVA.HFS
  TYPE=HFS
  PATH=/javawas
  PLIB=BPXPRMRD
TIME=08.54.04  DATE=2010/06/30        MOVE  RC=0079  RSN=119E04B7
         PATH=/SY2
         SYSNAME=CAT
etc....
```

```
D OMVS,MF=all or D OMVS,MF=a
```

- Prints the 50 most recent mount or move failures

```
D OMVS,MF=purge or D OMVS,MF=p
```

- Purges the saved failure information

Defining a user file system

Before a user is ready to log on to the z/OS UNIX shell using the TSO commands OMVS or ISHELL, you need to accomplish a few very important steps:

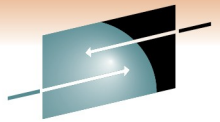
- Allocate space for a user file system in the HFS or zFS file system by creating a data set with a standard naming convention chosen by your installation.
- The data sets that define the file systems should be RACF-protected by creating a profile in the DATASET class and then permitting authorized users access to it.
  - Note: These steps can be done dynamically by automount.

Note: For the following administration steps, the administrator must have superuser authority to issue the commands. These commands are needed only for HFS file systems.

- Issue the CHOWN command to make the user owner of his directory.
- Issue the CHGRP command to make his default group the owning group of his directory.
- Issue the CHMOD command to change the permission bits for the user's directory to 700

Note: We should emphasize that the intended results from all three commands above are entirely a matter of the security policy adopted by your organization. You are in no way bound to use these commands in the suggested manner.

Using the Automount facility



The automount facility automatically mounts file systems at the time they are accessed.

- Using the automount facility provides many advantages:
  - Management of file systems is easier.
  - Resources are not consumed until they are requested.
  - You can reclaim system resources if that file system has not been used for a period of time.

Setting up the Automount facility

- Add the following statement to your BPXPRMxx parmlib member.

FILESYSTYPE TYPE(AUTOMNT) ENTRYPOINT(BPXTAMD)

- Either restart OMVS or
  - Issue SETOMVS RESET to activate the automount PFS.
  - Issue SET OMVS=(xx) will process FILESYSTYPE statements.
- Customize the configuration files before you can start using the automount facility.

/etc/auto.master

MapName

- Activate the automount facility.
  - From the shell as superuser ID, issue: /usr/sbin/automount

OR

  - Add the following lines to the /etc/rc file:

```
# Start the automount facility
/usr/sbin/automount
```

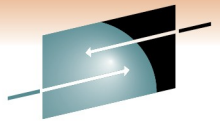
Automount files

- /etc/auto.master
- Specifies a list of directories to be managed, along with their MapName files.

- MapName
- The MapName file contains the mapping between a subdirectory of a directory managed by automount and the mount parameters.
  - It contains information that automount uses to
    - Determine file system to be mounted and mount point
    - Allocate the file system, if appropriate
    - How long to keep the file system mounted if it is not in use

SHARE in Anaheim 2011

Automount files, continued



Note: The automount facility allows the master and map files to reside in MVS data sets. Although the default remains /etc/auto.master, another file name can be specified on the command line. The data set can be a sequential data set or a member of a PDS.

The data set name must be specified as a fully qualified name and can be uppercase or lowercase.

Example:

/usr/sbin/automount "//sys1.parmlib(amtmst01)"

Notice the double quotes around the name to avoid unwanted shell processing.

/u //sys1.parmlib(amtmapu)

Notice there are no double quotes around the name in the master file since this is not processed by the shell.

Automount files, continued

automount [-e] [-a|q] [-s] [Master filename]

When run with no arguments, automount reads the /etc/auto.master file to determine all directories that are to be configured for automounting and the filenames that contain their configuration specifications.

-e Displays recent error information from automount attempting to create a new zFS or HFS file system. Typically, one allocation error value and reason code is displayed for the last allocation error.

-a Indicates that the policy being loaded is to be appended to the existing policy rather than replace the existing policy. For example: /usr/sbin/automount -a

Note: -a is mutually exclusive with -q.

-q Displays the current automount policy.

-s Checks the syntax of the configuration file. No automount is performed.

Automount generic entry

The following is an example of a generic entry:

```
------ /etc/auto.master ------
/u /etc/u.map
------ /etc/u.map ---------
  name          *
  type          ZFS
  filesystem     OMVS.ZFS.USER.<uc_name>
  mode          rdwr
  duration      30
  delay         10
  parm          FSFULL(50,5)
  allocuser  space(5,2) storclas(SMS1)
```

Automount specific entry

The following is an example of a specific entry:

Given the **/etc/auto.master** and **/etc/u.map** files as shown below whenever the directory **/u/totten** is referred to by a command such as cd or cp, the automount facility mounts the OMVS.TOTTEN.ZFS data set.

```
------ /etc/auto.master ------
/u /etc/u.map
------ /etc/u.map ---------
name totten
filesystem OMVS.TOTTEN.ZFS
duration nolimit
```

For more information, see the automount command in *z/OS UNIX System Services Command Reference.*

Automount example

The automount facility scans the /etc/auto.master file first to see what MapName file or files should be read. Assume the /u directory is being managed.

```
$ cd /u/totten
$ df -Pkv .
Filesystem          1024-blocks        Used  Available  Capacity Mounted on
OMVS.ZFS.USER.TOTTEN          351360       15812     335452         5% /u/totten
ZFS, Read/Write, Device:96203, ACLS=Y
File System Owner : AQTS         Automove=Y       Client=N
Filetag : T=off   codeset=0
$ df -Pkv /u
Filesystem          1024-blocks        Used  Available  Capacity Mounted on
*AMD/u                        4           4          0       100% /u
AUTOMNT, Read/Write, Device:66, ACLS=N
File System Owner : AQTS         Automove=Y       Client=N
Filetag : T=off   codeset=0
```

File security

- UNIX objects are protected with POSIX permission bits

User	Group	Other
read **w**rite e**x**ecute	**r**ead **w**rite e**x**ecute	**r**ead **w**rite e**x**ecute

- Can only specify permissions for file owner (user), group owner, and everybody else
- **Access Control Lists** permit/restrict access to specific users and groups
- ACLs are used in conjunction with permission bits.

SHARE in Anaheim 2011

Access Control Lists (ACLs) Overview

- Traditional UNIX approach
- Contained within the file system
  - File security is portable
  - Deleted automatically if the file is removed
- Not protected by RACF profiles
- Managed using UNIX shell commands, or ISHELL
- Supports inheritance for new files and subdirectories

SHARE in Anaheim 2011

Participating File Systems

- HFS - Hierarchical File System
- zFS – z/Series File System
- TFS - Temporary File System
- NFS - with NFSv4 support
  - Note: There may be remote ACL management restrictions due to differences in ACL implementations on various platforms.

Terminology

- **base ACL entries** = permission bits
  - user::*rwx*
  - group::*rwx*
  - other::*rwx*

- **extended ACL entries**
  - user:***uid***:*rwx*
  - group:***gid***:*rwx*

  - default:user:***uid***:*rwx*
  - default:group:***gid***:*rwx*

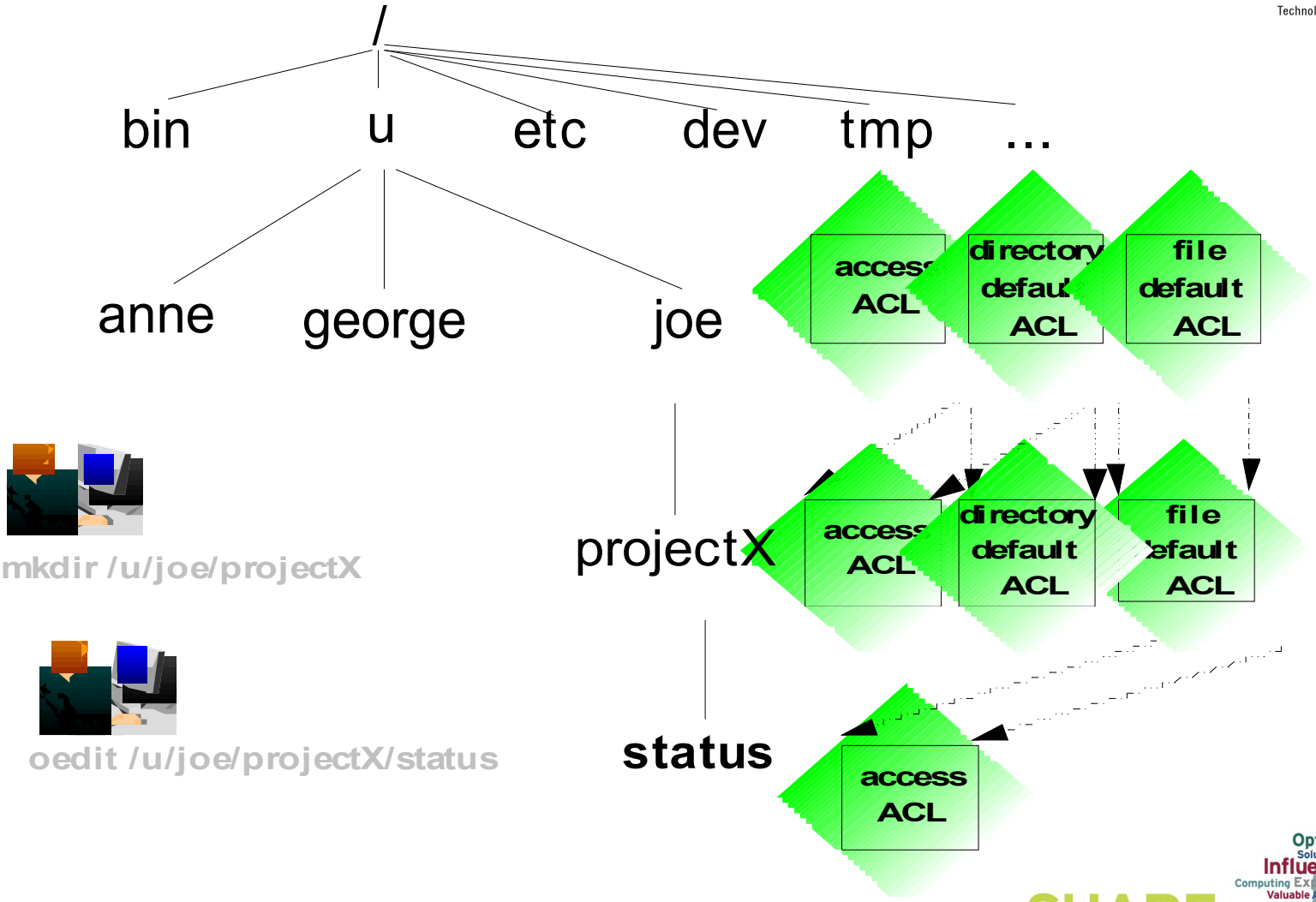
  - fdefault:user:***uid***:*rwx*
  - fdefault:group:***gid***:*rwx*

SHARE in Anaheim 2011

ACL Inheritance

- Can establish default (or 'model') ACLs on a directory
- They will get automatically applied to new files/directories created within the directory
- Separate default ACL used for files and (sub)directories
- Can reduce administrative overhead

ACL Inheritance example



SHARE in Anaheim 2011

shell commands

setfacl — *set, remove, modify ACL entries*

- Allowed by file owner

  or

- superuser
  - UID 0

    or
  - READ access to

    SUPERUSER.FILESYS.CHANGEPERMS

getfacl — *display owner, group, ACL entries*

- Allowed by anyone with directory search access

SHARE in Anaheim 2011

setfacl

set ACL contents

- setfacl -s ***entries*** [***path*** …]
  - set (replace) entire ACL
  - must include base ACL entries (permission bits)
- setfacl -S ***file*** [***path*** …]
  - set (replace) entire ACL from file
  - must include base ACL entries (permission bits)
- setfacl -D ***type*** … [***path*** …]
  - delete extended ACL entries of matching type
- setfacl -m|M|x|X ***EntryOrFile*** [***path*** …]
  - modify or delete extended ACL entries

setfacl

- An ACL can be set from contents of a file
  - `setfacl -S ~/acls/ateam  reldir`

    where ~/acls/ateam contains an entire ACL (e.g.):

    u::rwx
    g::r-x
    o::---
    g:shut:rwx
    g:testers:r-x

- Allows use of "named ACLs"

- An ACL can be set from stdin, and thus piped in from a getfacl command
  - `getfacl YourFile | setfacl -S - MyFile`

getfacl

display ACL contents

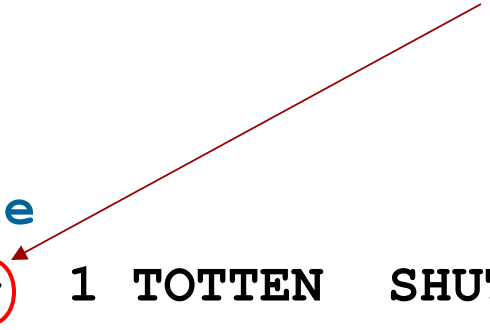
- getfacl MyFile
  - Displays file name, user owner, and group owner
  - Displays base POSIX permissions in "ACL format"
  - Displays access ACL entries

```
#file:  MyFile
#owner: TOM
#group: RACFDEV
user::rwx
group::r--
other::r--
user:ANN:rwx
group:RACFDEV:r-x
```

ls command

list file / directory attributes

- ls command indicates existence of extended ACL entries



```
ls -l MyFile
-rwxrwxr-x+  1 TOTTEN   SHUT    44 Apr  3 14:49 MyFile
```

find

find files with matching criteria

- **`find` *`path`* `-acl a|d|f`**
  - find all files with an ACL of a given type, or types
- **`find` *`path`* `-acl_user userid`**
  `-acl_group groupid`
  `-acl_entry acl_text`
  - find files with ACL entries for a specific user/group
- **`find` *`path`* `-acl_count number`**
  - find files with (more than) number ACL entries

SHARE in Anaheim 2011

find *command substitution*

- Useful in command substitution
  - Permit group ALPHA to search every directory under /u/totten/tools

```
setfacl -m g:ALPHA:r-x $(find /u/totten/tools -type d)
```

  - Remove user TED from all ACL entries

```
setfacl -qx u:TED,d:u:TED,f:u:TED $(find / -acl_user TED)
```

  - Add the group ALPHA to every access list in /u/shr/ which contains an entry for UNIXGRP:

```
setfacl -m  g:ALPHA:rwx $(find /u/shr -acl_entry UNIXGRP)
```

Other Interfaces to manipulate ACLs

- **Application Programming interfaces:**
  - Language Environment (LE) provides C services
  - REXX provides similar functions
  - Low level Logical File System (LFS) interface also available
- **ISHELL support**

SHARE in Anaheim 2011

RACF Access Checking with ACLs

- Takes into account base POSIX permissions and access ACLs
- ACLs only used if the **FSSEC** class is active
  - **SETROPTS CLASSACT(FSSEC)**

  will activate use of ACLs in Unix file authority checks
  - Make sure that FSSEC is **not active** until you are ready to use ACLs
    - The class need not be active to create ACLs
- **setfacl** can be used to create ACLs at any time

SHARE in Anaheim 2011

Multilevel security

- Multilevel security is a security policy that allows the classification of data and users based on a system of hierarchical security levels combined with a system of non-hierarchical security categories.

- Traditionally, access to z/OS® UNIX® resources is based on POSIX permissions and access control lists (ACLs). In a multilevel-secure z/OS UNIX environment, authorization checks are performed for security labels in addition to POSIX permissions, to provide additional security.

  - See z/OS V1R8.0-V1R9.0 Planning for Multilevel Security and the Common Criteria
    GA22-7509-06

HFS to zFS Migration Health Check

➢ New in z/OS Release 12

- **Health Check Description**
  - health check to notify users that they should migrate all HFS file system to zFS.

- **Problem**
  - As of R1.7 HFS was no longer considered the strategic file system in favor of zFS. This check will be used to highlight any HFS file systems still being used so that they can be migrated to zFS.

- **Solution Abstract**
  - A new check was created called USS_HFS_DETECTED that will create a report of every HFS file system mounted with the intention of getting the user to migrate to zFS. The exception message will point to the USS Planning guide which contains information on migrating to zFS. The test is valid is non-sysplex and share file system environment.

SHARE in Anaheim 2011

Discussion List

Customers and IBM participants also discuss z/OS UNIX on the *mvs-oe discussion list*.

This list is not operated or sponsored by IBM.

To subscribe to the mvs-oe discussion, send a note to:

listserv@vm.marist.edu

Include the following line in the body of the note, substituting your first name and last name as indicated:

subscribe mvs-oe *first_name last_name*

After you are subscribed, you will receive further instructions on how to use the mailing list.

Helpful sites

- For help with customizing z/OS UNIX, check out our Web-based wizard at www.ibm.com/servers/eserver/zseries/zos/wizards/
- The z/OS UNIX home page on the World Wide Web contains technical news, customer stories, and information about tools. You can visit it at www.ibm.com/servers/eserver/zseries/zos/unix/
- You can access IBM message explanations directly from the LookAt Web site at http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/

Publications

- UNIX System Services Planning
  - GA22-7800
- UNIX System Services Command Reference
  - SA22-7802
- UNIX System Services Assembler Callable Services
  - SA22-7803
- UNIX System Services User's Guide
  - SA22-7801-05
- UNIX System Services Messages and Codes
  - SA22-7807-05
- IBM Health Checker for z/OS: User's Guide
  - SA22-7994-00
- z/OS V1R11.0 Distributed File Service zSeries File System Administration z/OS V1R11.0 SC24-5989-11
- z/OS V1R8.0-V1R9.0 Planning for Multilevel Security and the Common Criteria GA22-7509-06