



IBM eServer™

Session R05
RACF and z/OS UNIX System Services

Networking and Security Technical Conference
October 25-28, 2004
Anaheim, California

Peggy LaBelle
IBM Corporation, RACF Development
(845) 435-5910
plabelle@us.ibm.com

Disclaimer

- The information contained in this document is distributed on an "as is" basis, without any warranty either express or implied. The customer is responsible for use of this information and/or implementation of any techniques mentioned. IBM has reviewed the information for accuracy, but there is no guarantee that a customer using the information or techniques will obtain the same or similar results in its own operational environment.
- In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used. Functionally equivalent programs that do not infringe IBM's intellectual property rights may be used instead. Any performance data contained in this document was determined in a controlled environment and therefore, the results which may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.
- It is possible that this material may contain references to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM Products, programming or services in your country.
- IBM retains the title to the copyright in this paper as well as title to the copyright in all underlying works. IBM retains the right to make derivative works and to republish and distribute this paper to whomever it chooses.

Trademarks

- **The following are trademarks or registered trademarks of the International Business Machines Corporation:**
 - OS/390
 - z/OS
 - RACF
- **UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.**

Agenda

- z/OS UNIX introduction
- z/OS UNIX users
 - UNIX vs. z/OS identity
 - Defining a UNIX user and group
 - And more...
- z/OS UNIX file systems and MVS datasets
 - File ownership, file permissions, ACLs
 - Executable programs
 - And more...
- Auditing z/OS UNIX

IBM eServer™ IBM

What is z/OS UNIX System Services?

- Base element of z/OS
 - Formerly known as the OpenEdition product and OMVS
- UNIX interface for z/OS providing
 - File system containing directories and files
 - Application interfaces for porting programs and data
 - Commands
- Services integrated with z/OS
 - Invoke UNIX programs from TSO or BATCH; invoke LINKLIB programs from shell
 - Manage file system from shell, TSO, console
 - Open data sets, UNIX files, from any environment

5 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

Makes application development easier:

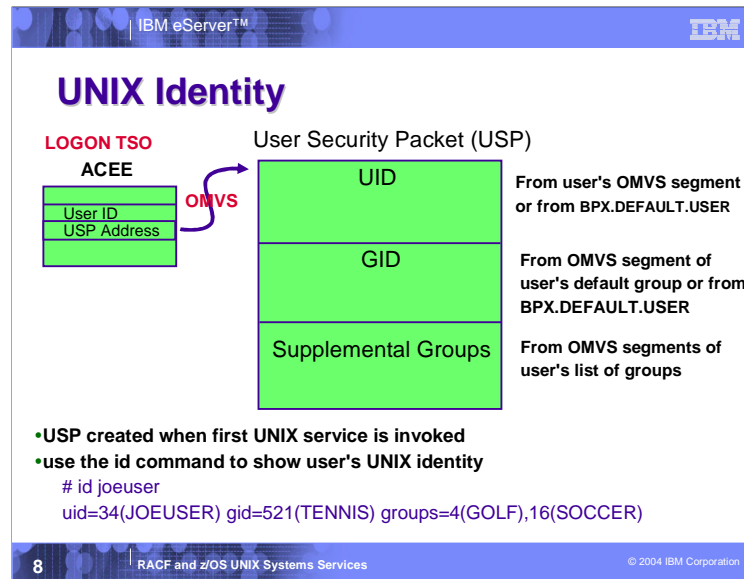
- Standard (open) programming interface
- Interoperability in networks
- Portable programs
- Portable data

Required by products such as TCP/IP, Lotus Domino, LDAP, EIM, PKI

How is it related to RACF?

- External security product is required
- User identification and authentication
- Protection of files
- Protection of services (su, chmod, chown, etc)
- Auditing of security events

z/OS Unix System Services Identities



When a UNIX service is invoked, the ACEE (z/OS security context) is supplemented with a USP containing the user's UNIX identity

UNIX identity consists of a numeric UID taken from the OMVS segment of the USER profile, a numeric GID taken from the OMVS segment of the GROUP profile associated with the user's current connect group (usually the user's default group, unless (s)he specified a different group at logon), and a list of GIDs taken from each group (with an OMVS segment) to which the user is connected, up to 300.

IBM eServer™ IBM

UNIX Identity

•When accessing data sets and other RACF-protected resources:

- 8-character User ID (and group name) is checked against RACF profile

•When accessing UNIX files and directories:

- Numeric UID and GIDs are checked against file owner and permissions

9 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

When a data set is accessed, the user ID from the ACEE is used for authority checking using RACF profiles.

When a UNIX file is accessed, the effective UID, effective GID, and supplemental GIDs are used for authority checking using the security information which is stored with the file in the file system.

BUT when UNIXPRIV is checked, the user ID is used!

IBM eServer™ IBM

UNIX User definition

- User profiles need OMVS segments
 - UID - 0 to 2147483647 user identifier (0 is superuser)
 - HOME - home and initial working directory
 - PROGRAM - initial program to execute
 - Other fields contain various resource limits
- Group profiles need OMVS segments
 - GID - 0 to 2147483647 group identifier
 - User's current connect group *and default group* need GID
- UIDs and GIDs should be unique
- Values can take defaults (from BPX.DEFAULT.USER ... more later...)

10 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

UNIX limits in OMVS segment are: max address space size, cpu time, files per process, memory map size, and threads per process..they are discussed shortly...

Also, default GID can be taken from BPX.DEFAULT.USER (In the absence of BPX.DEFAULT.USER) current connect group **MUST** have a GID. If default group does not have a GID, you get an error message when adding the UID to the user, and when entering the shell. **You *can* enter the shell, but (at least) getpwnam() won't work, and so 'ls -l' won't be able to map to names.**

IBM eServer™ IBM

User Definition Example

```
ADDGROUP UNIXGRP OMVS(GID(100))
ADDUSER ANTOINE PASSW(XXXX) DFLTGRP(UNIXGRP)
      OMVS(UID(8) HOME(/u/antoine) PROGRAM(/bin/sh))
      TSO(ACCTNUM(12345) PROC(PROC01))
LISTUSER ANTOINE OMVS NORACF
```

USER=ANTOINE
OMVS INFORMATION

UID = 000000008
HOME = /u/antoine
PROGRAM = /bin/sh
CPUTIMEMAX= NONE
ASSIZEMAX= NONE
FILEPROCMA= NONE
...

11 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

The limit fields are discussed shortly..

There is no reason to prevent general users from updating their own HOME and PROGRAM fields; use the FIELD class. But by all means, do not let them update their own UID!!! Letting them update their own limits would also not be a very good idea.

IBM eServer™ IBM

User Definition ... SUPERUSER!

- A superuser is defined as
 - UID 0, any GID
 - Trusted or privileged, any UID, any GID
- A superuser can:
 - Pass all z/OS UNIX security checks
 - Affect any UNIX process on the system
 - Change his identity to another UID
 - Use setrlimit to increase system limits
- Not used when accessing z/OS resources
 - But a superuser **may** be able to assume any z/OS user ID

12 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

A superuser is a user running with UID 0. A started task that is trusted or privileged is also considered a superuser.

Changing to another UID can be further controlled using BPX.SERVER and BPX.DAEMON. But if you're not using these profiles, a superuser can switch to any identity

Note that this **can** (indirectly) allow a superuser to access z/OS resources with another user ID

IBM eServer™ IBM

User Definition ... SUPERUSER!

- A superuser may gain access to a SPECIAL or OPERATIONS RACF user ID
- To the best of your ability, you should avoid assigning UID(0) to administrators
 - Use UNIXPRIV class or BPX.SUPERUSER to restrict functions
 - Use FACILITY class resources BPX.DAEMON and BPX.SERVER to limit identity changing
- UID(0) for started task users, and UNIX daemons, is generally OK
 - use the NOPASSWORD attribute to prevent these from being logged onto

13 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

A superuser is essentially SPECIAL and OPERATIONS if you are not protecting servers and daemons (discussed later). In this scenario, the superuser can simply write a program which does a `setuid()` to any user ID in the system, thus allowing them to access that user's UNIX files and z/OS datasets. By `setuid()`ing to an OPERATIONS user, one could conceivably access any data set. By `setuid()`ing to a SPECIAL user, one could conceivably execute any RACF command using the `R_admin` interface.

SUPERUSER Granularity: UNIXPRIV Class

- Used to assign subset of SUPERUSER authority to a user
- Goal: Reduce the number of users needing full SUPERUSER authority
- Partial list of functions you can grant:
 - ability to read or write any UNIX file
 - ability to change file ownership
 - ability to change file permissions/ACLs
 - ability to send signals to any process
 - ability to mount/unmount file systems

Contrast this with authority to BPX.SUPERUSER. With BPX.SUPERUSER, you issue the su command to get into "superuser mode", do your task, and exit. With UNIXPRIV, you always have that authority, so be careful not to make mistakes!

IBM eServer™ IBM

UNIXPRIV Resource Names

Examples:

Resource Name	Privilege	Access Required
SUPERUSER.FILESYS	Read any file; read/search any directory	READ
SUPERUSER.FILESYS	Write any file; also privileges of READ access	UPDATE
SUPERUSER.FILESYS	Write any directory; also privileges of UPDATE access	CONTROL
SUPERUSER.PROCESS.KILL	Use kill() callable service to send signals to any process	READ

See [z/OS UNIX System Services Planning](#) for complete list of UNIXPRIV resources

15 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

File execution (as opposed to directory searching) is not covered by UNIXPRIV

Other UNIXPRIV resource names

SUPERUSER.FILESYS.CHOWN - READ access allows user to use the chown command to change ownership of any file.

SUPERUSER.FILESYS.MOUNT - READ allows user to mount file system with nosetuid option. UPDATE allows user to mount file system with setuid option.

SUPERUSER.FILESYS.QUIESCE - READ/UPDATE allows user to issue quiesce/unquiesce commands for a file system mounted with nosetuid/setuid.

SUPERUSER.FILESYS.PFSCTL - READ access allows user to use the pfctl() callable service.

SUPERUSER.FILESYS.VREGISTER - READ allows a server to use the vreg() callable service to register as a VFS file server.

SUPERUSER.SETPRIORITY - READ access allows a user to increase own priority.

IBM eServer™ IBM

User definition ... System Resource Limits

- UNIX System Services provides system resource limits to customize the performance of the kernel for your installation
 - Maximum settings for a user or process
 - Specified in parmlib member in BPXPRMxx
 - Can be reset by SETOMVS or SET OMVS command
 - SUPERUSER can choose to exceed these limits
- RACF provides similar settings on user level
 - Allows specific limits for individual users
 - Some users (usually servers) need more resources than normal users

16 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

Choices before OS/390 V2R8:

increase system limit for all users

bad for system reliability, performance

give server UID(0) or BPX.SUPERUSER authority and modify server code to request a higher limit

requires modification to server code

With OS/390 V2R8 and later:

assign higher limits specifically to server user IDs (or others) that need them

IBM eServer™ IBM

User definition ...

System Resource Limits

Global Resource Maximum Value	Keyword in BPXPRMxx	OMVS Segment Keyword on RACF ADDUSER / ALTUSER command
CPU Time	MAXCPU TIME	CPUTIMEMAX
Address Space Region Size	MAXASSIZE	ASSIZEMAX
Open Files Per Process	MAXFILEPROC	FILEPROCMAX
Processes Per UID	MAXPROCUSER	PROCUSERMAX
Threads Per Process	MAXTHREADS	THREADSMAX
Amount of Storage mapped by mmap()	MAXMMAPAREA	MMAPAREAMAX

Example:
ALTUSER SERVER1 OMVS(THREADSMAX(400))

17 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

As usual, TSO command abbreviations are allowed, so you don't necessarily need to type all of these fully.

MAXPROCUSER - may impact the number of users attempting to use kernel services through use of the default UID established in the BPX.DEFAULT.USER profile.

OMVS segment keywords on ADDUSER and ALTUSER allow specific limits for individual users:

- CPUTIMEMAX(cpu-time)
- ASSIZEMAX(address-space-size)
- FILEPROCMAX(files-per-process)
- PROCUSERMAX(processes-per-UID)
- THREADSMAX(threads-per-process)
- MMAPAREAMAX(memory-map-size)

Listed via LISTUSER

IBM eServer™ IBM

Default UNIX User and Group Identity

- BPX.DEFAULT.USER in the FACILITY class can be used to assign default OMVS segment data
 - RDEFINE FACILITY BPX.DEFAULT.USER APPLDATA('DFTUSER/DFTGROUP')
 - ADDUSER DFTUSER OMVS(... ..) NOPASSWORD
 - ADDGROUP DFTGROUP OMVS(GID(nnn))
- Assigned when user/group doesn't have an OMVS segment
- Can be overridden on a per-user basis
 - ALTUSER BOB OMVS(NUID)
- Use of default identity is noted in any audit record produced
- Should have only limited use
 - TCP/IP from z/OS to z/OS, or, just getting your feet wet with UNIX System Services

Don't use UID(0) !!!!

18 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

Consider making default user NOPASSWORD so nobody can try to log on to it


All OMVS fields are eligible for use

UID - choose a number that jumps out visually (not 0!!!)

HOME - all default users will have access to each other's files. Consider putting it in /tmp. Also consider making it root, where they will not have write access

PROGRAM - usually /bin/sh. If you don't want default users to be able to use the shell, set it to /bin/echo

resource limits like CPUTIMEMAX, PROCUSERMAX, and THREADSMAX are shared among all users using the default segment, so you might want to make the default limits higher than you otherwise would

IBM eServer™ 

Prevention of Shared IDs

- New SHARED.IDS profile in the UNIXPRIV class
- Acts as a system-wide switch to prevent assignment of an ID which is already in use
- No generic characters allowed in name: discrete profile name must be used
- APAR OW52135
 - OS/390 2.10: UW89970 - also applies to z/OS 1.1
 - z/OS 1.2: UW89971
 - z/OS 1.3: UW89972
 - In base of z/OS 1.4

<http://www.ibm.com/servers/eserver/zseries/zos/racf/whatsnew.html>

19 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

The APAR does not include ISPF panel support. It is included in z/OS V1.4.

IBM eServer™ IBM

Prevention of shared IDs

- Requires Application Identity Mapping (AIM) stage 2 or 3
- Does not affect pre-existing shared IDs
 - Customer must clean those up separately, if desired
 - Not a pre-req for using the new support
 - Can use IRRICE reports to find shared UIDs and GIDs

20 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

IRRIRA00 utility – RACF Internal Reorganization of Aliases Utility Program (OS/390 V2R10)

In final stage, stage 3, provides an alternative to the use of mapping profiles to associate RACF user and group names with

z/OS UNIX ids

LOTUS Notes ids

Novell Directory Services ids.

The IRRIRA00 utility converts from the use of UNIXMAP profiles to the use of alternate database indices. It is a fairly clean and straightforward process, but some customers have complained about:

1) the time it takes to move to stage 1 with a large RACF database. IRRIRA00 doesn't give you any 'progress reports'

2) the fact that AIM chokes when there are somewhere in the vicinity of 130 UID(0) users. If you have 130 super users, you should really think long and hard about that. That's 130 users with essentially SPECIAL and OPERATIONS!!! Use BPX.SUPERUSER, UNIXPRIV, and set up server and daemon level protection (more later).

Prevention of Shared IDs ... Example

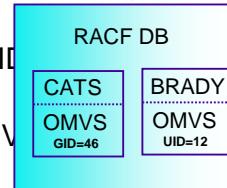
**NEW!**

- RDEFINE UNIXPRIV SHARED.ID
UACC(NONE)
- SETROPTS RACLIST(UNIXPRIV
REFRESH
- ADDUSER MARCY OMVS(UID(12))

**IRR52174I Incorrect UID 12. This value is already in use
by BRADY.**

- ADDGROUP DOGS OMVS(GID(46))

**IRR52174I Incorrect GID 46. This value is already in use
by CATS.**



IBM eServer™ IBM

Prevention of shared IDs ... New SHARED keyword

- There are valid reasons to assign a non-unique UID/GID
 - E.G. Assigning UID(0) to started task user IDS
- Do so using the new SHARED keyword in the OMVS segment of the ADDUSER, ALTUSER, ADDGROUP, and ALTGROUP commands
- SHARED requires SPECIAL, or at least READ authority to SHARED.IDS
 - Profile level audit settings can be used to log successes and failures to SHARED.IDS


22 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

Can generally only log successes in the UNIXPRIV class. We make an exception for SHARED.IDS because it is the only way to log the failure...you won't get a command related (e.g. ADDUSER, ALTGROUP) audit record.

No FIELD class checking for SHARED operand because there's no corresponding field in the RACF database! This is a change from existing fields within non-base segments which always correspond to a template field in the RACF database. Processing for the SHARED keyword is just pertinent for the life of the command, and is not an 'attribute' which needs to be stored in the USER profile.

IBM eServer™ IBM


Prevention of Shared IDs ... Example



- PERMIT SHARED.IDS CLASS(UNIXPRIV) ID(UNIXGUY) ACCESS(READ)
- SETROPTS RACLIST(UNIXPRIV) REFRESH

RACF DB


BPXOINIT
OMVS
UID=0



AU OMVSKERN OMVS(UID(0) SHARED)

AG (G1 G2 G3) OMVS(GID(9) SHARED)

✓ OK!



AU MYBUDDY OMVS(UID(0) SHARED)


IRR52175I You are not authorized to specify the SHARED keyword.

23 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

Even if MVSGAL would otherwise have been authorized to define the user (e.g. via CLAUTH(USER), FIELD access, group authority, etc), she may not assign a shared UID or GID unless she is SPECIAL, or has access to SHARED.IDS.

By the way, I feel compelled to say that we recommend you permit groups to RACF profiles instead of users, but it's so much simpler to show examples using users!!!

The SHARED keyword will be ignored when: SHARED.IDS is not defined, UID/GID is not also specified, the ID value specified is unique, and for ALTUSER/ALTGROUP when the specified ID is the value already assigned (regardless of whether it is unique or not).

IBM eServer™ 

SEARCH Enhancement to Map UIDs and GIDs

- SEARCH CLASS(USER) UID(0)
OMVSKERN
BPXOINIT
SUPERGUY
- SEARCH CLASS(GROUP) GID(99)
RACFDEV
- SEARCH CLASS(USER)
UID(1234567)
ICH31005I NO ENTRIES MEET SEARCH CRITERIA
- Available with OW52135


24 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

No authority to the user/group profile is required in order to list it using the new keyword. This is consistent with UNIX mapping interfaces.

The new keywords require AIM. If you don't have AIM, then continue to RLIST UNIXMAP profiles.


All other SEARCH keywords are ignored when you use UID or GID (except of course the CLASS keyword, which must specify USER or GROUP as appropriate. Any other class will cause the UID/GID keywords to be ignored)

No more excuses! You will start to see more new function which requires AIM.

IBM eServer™ 

Automatic UID/GID Assignment

- New AUTOUID keyword in the OMVS segment of the ADDUSER and ALTUSER commands
- New AUTOGID keyword in the OMVS segment of the ADDGROUP and ALTGROUP commands
- Derived values are guaranteed to be unique
- Must use in conjunction with SHARED.IDS

 **ADDUSER MELVILLE OMVS(AUTOUID)**
IRR52177I User MELVILLE was assigned an OMVS UID value of 4646.

ADDGROUP WHALES OMVS(AUTOGID)
IRR52177I Group WHALES was assigned an OMVS GID value of 105.

25 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

Anyone authorized to assign a UID/GID, can use AUTOUID/AUTOGID

As with the SHARED keyword, there is no RACF database template field for AUTOUID or AUTOGID, and thus no FIELD class protection; except of course for the UID and GID fields themselves.

IBM eServer™ IBM

Automatic UID/GID Assignment ... BPX.NEXT.USER

- Uses APPLDATA of new BPX.NEXT.USER profile in the FACILITY class to derive candidate UID/GID values
- APPLDATA has a qualifier for UID and a qualifier for GID separated by a forward slash (/)
 - Left qualifier is starting UID value or range of UID values
 - Right qualifier is starting GID value or range of GID values
 - qualifiers can be null, or specified as 'NOAUTO', to prevent automatic assignment of UIDs or GIDs

26 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

APPLDATA verified at time of use, not when defined
FACILITY class need not be active or RACLISTed

When AUTOUID or AUTOGID is issued, RACF

- extracts the APPLDATA from BPX.NEXT.USER
- parses out the starting value
- checks to see if it is already in use
- if so, the value is incremented and checked again until an unused value is found
- assigns the value to the user or group
- replaces the APPLDATA with the new starting value

The administrator can change the APPLDATA at any time using

IBM eServer™ IBM

Automatic UID/GID Assignment ... APPLDATA syntax

- Examples
 - RDEFINE FACILITY BPX.NEXT.USER APPLDATA('data')
 - good *data*
 - 1/0
 - 1-50000/1-50000
 - NOAUTO/100000
 - 10000-20000/NOAUTO
- Want an easy way to assign a unique GID to all your groups?
 - SEARCH CLASS(GROUP) NOLIST CLIST('ALTGROUP' OMVS(AUTOGID)')
 - EX EXEC.RACF.CLIST

27 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

A qualifier can be specified as 'NOAUTO', or simply omitted, to prevent automatic ID assignment.

Might want this if you have a convention for users, such as employee serial number
Values must be valid UIDs or GIDs. If specifying a range, end of range must be greater than start of range. No white space allowed.

Validity is checked at time of use, not at time of definition (So, RDEFINE/RALTER won't fail, but subsequent AU, ALU, AG, ALG will). Give it a quick test to be sure.

No more complaining about having to assign GIDs to all your groups!

Could also be used for users, but, there are other things in the OMVS segment of USER profiles which will need addressing (e.g. HOME, PROGRAM). ISHELL can do this for you.

AUTOUID/AUTOGID will not reassign a new value if one already exists

Ways of Assuming Another UNIX Identity

- Executing a set-id file
 - changes effective UID/GID to that of file owner
- Issuing the su command
 - used to switch to 'superuser mode' or to another user entirely
- Various C language functions such as `setuid()`, `setgid()`, `pthread_security_np()`
 - Used by UNIX servers and daemons

IBM eServer™ IBM

set-UID and set-GID files

- Executable files which change the effective UID/GID to that of the file owner
 - UNIX file access now based on owner (user and/or group) of set-id file
 - does *not* change the MVS identity
 - locate your set-uid files with `find / -perm -4000` or by using `irrhfsu`
- `chmod u+s,g+s myprogram`
 - must be file owner or have superuser privilege
- `ls -l myprogram`
`-rwsr-s--x 2 JILL DEPTD60 8192 Feb 8 10:51 myprogram`

29 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

When a set-UID or set-GID file executes, the effective UID/GID is changed to that of the file owner.

You can locate set-uid files with '`find / -perm -4000`' or by using `irrhfsu`.

This is similar to Program Access to Data Sets (PADs).

IBM eServer™ IBM

set-UID and set-GID files (continued)

- Changing file ownership (chown command), or writing to the file, resets set-uid and set-gid bits
- Consider mounting remote/untrusted file systems with the NOSETUID option

30 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

The set-uid and set-gid bits are reset when the owner of the file is changed or the file itself has changed.

When a file system is mounted with the NOSETUID option, any set-uid and set-gid bits are ignored. Use this on untrusted or remote file systems, since they may have set-uid 0 (superuser) executables that could damage your system.

set-uid files can be dangerous in that some user who is superuser on his desktop risc box can create set-uid root programs, and if these files are mounted on z/OS, they will retain their UID 0 owner, and set-uid status. MOUNT has NOSETUID options to avoid this sort of thing.

IBM eServer™ IBM

su Command

- plain 'su' command switches to superuser mode (effective UID = 0)
 - requires READ access to BPX.SUPERUSER resource in the FACILITY class
 - changes only the UID, not the z/OS user ID
- su *userid* changes identity to another user
 - must know the user's password, or
 - have READ access to BPX.SRV.*userid* resource in the SURROGAT class
 - effective UID **and** MVS user ID is changed

31 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

Note that su w/ *userid* completely assumes the target user's identity: both user ID and UID (switching to superuser keeps your *userid*). Use "-s" to avoid a password prompt if you have SURROGAT authority, or just hit enter at the password prompt

If you have SURROGAT authority, you can issue **su -s *userid*** to skip the password prompt. If you forget to specify **-s**, then just hit enter at the prompt and your SURROGAT authority will be honored.

Essentially, su creates a new shell environment with the authority of the new user.

Controlling Daemons

- Daemon - a process that changes identities, usually has UID(0)
- Daemon programs call identity changing services to alter the UID and user ID of an address space
 - seteuid()
 - setuid()
 - spawn() with a user ID
- Define FACILITY class profile BPX.DAEMON
 - The daemon address space must be kept clean

Daemon programs can issue...

- setuid() - set uid
 - seteuid() - set effective uid - invokes z/OS SAF services to change the z/OS identity of the address space
 - spawn() - create and start a child process that runs a program in a new Address Space (fork and exec) - security info from the parent AS will be propagated to the child unless `_BPX_USERID` environment variable specifies otherwise (R9 C/C++ RTL pg 1409)
- ... to change the OS/390 identity in an address space or process in order to run work on behalf of a user. C/C++RTL

Daemon authority - user ID that is authorized to change to any z/OS identity that has an OMVS segment without knowing the target user's password.

Or 'BPX.SRV.target_userid' authority can change identities w/o a pw. Class SURROGAT (*web notes*)

Only Superusers with access to BPX.DAEMON authority can invoke these functions without authenticating the user, or without having surrogate authority. If not defined, an uncontrolled executable can be loaded .

May want to protect privileged z/OS users by not allowing them to have an OMVS segment or a UID.

ALU privuser OMVS(NOUID) (*R8 Unix Planning p 250*)

Clarification: any user (uid) can use setuid() if they either authenticate or have surrogate authority. This does not change when BPX.DAEMON is defined. BPX.DAEMON scopes which superusers can use setuid without authentication/surrogate and enforces program control.

If BPX.DAEMON is defined, the `__passwd()` service will enforce program control

All programs loaded must be controlled

PROGRAM profiles covering all programs from z/OS libraries (UACC READ is OK)

Controlled attribute for programs from the HFS

Set with `extattr +p`

Issuer needs authority to BPX.FILEATTR.PROGCTL

Turned off automatically if file is changed

Ignored if HFS mounted with `noseuid` or `nosecurity`

IBM-supplied daemons shipped in `/usr/bin` with sticky bit on so SYS1.LINKLIB copy will be used, or shipped with `+p` extended attribute

IBM eServer™ IBM

Controlling Servers

- Server - a process that does work for clients
- A well-behaved server does the following:
 - Verifies the client's identity
 - RACF or application password, digital certificate
 - Creates a thread for the client's work (like MVS task)
 - Associates the client's user ID with the thread
 - pthread_security_np (BPX1TLS)
 - Checks the client's authority when accessing z/OS resources
 - auth_check_resource_np (BPX1ACK)
 - _check_resource_auth_np()
- Define FACILITY class profile BPX.SERVER
 - Clean address space is required

33 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

Ensure control over Server to ensure they do what you want. Servers designed on other platforms have their own security...
Not all Servers authenticate their clients or pass out resources properly

pthread_security_np = Create/Delete the security environment for the callers thread (*RACF InitACEE Callable Svc*)

auth_check_resource_np = Callable service to determine a User's Access to Protected z/OS Resource (*R9 Unix ASM Callable Services p 59*)

Caller must have READ access to BPX.SERVER Facility class profile (Or UID=0 if BPX.SERVER not defined)

_check_resource_auth_np() = C function call to determine Access to z/OS Resources

Caller must have read permission to FACILITY class BPX.SERVER
 (Or UID=0 if BPX.SERVER not defined) (R 9 C/C++RTL)

BPX.SERVER profile is useful for programs that will establish a task-level(z/OS) or thread-level(UNIX) security environment.

Examples: IBM http Server (formerly Web Server)

DCE Application Server

Daemon and server programs must be protected

Sticky bit on in file system copy - get copy from LPA which is program controlled.

SYS1.LINKLIB is protected with PROGRAM and DATASET profiles

program control **extended attribute "extattr +p"** for HFS-resident programs

nosetuid = no program controlled entities

PADS - Program Access to Data Sets - make the use of a protected program a condition for access to a dataset or load module.

SETR WHEN(PROGRAM(pgm_name))

extattr command will fail if BPX.FILEATTR.* is not defined, or if the FACILITY class is inactive.

IBM eServer™ IBM

Auditing Users and Processes

- Controlled by audit classes PROCESS and PROCACT
 - SETROPTS AUDIT
 - PROCESS - UNIX process creation and deletion
 - SETROPTS LOGOPTIONS
 - PROCESS - changes to process identity
 - PROCACT - attempts to alter another identity's process (e.g. kill, ptrace, etc)
- RACF UAUDIT attribute honored
 - Use carefully – creates many audit records
- Some events are always audited
 - Attempt to create a process for a user with a missing or incomplete OMVS segment
 - Creation of a process which uses the default OMVS segment

34 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

UAUDIT is mentioned for completeness...to demonstrate that it applies to UNIX as well. However, be careful as it will generate **lots** of audit records.

The default UID auditing has two aspects:

- 1) creation of the process is always audited
- 2) subsequent events, *if audited*, specify that a default UID was used



UNIX Auditing ... The results

- Type 80 SMF records
- ICH408I messages for resources and services

```
ICH408I USER(SYS) GROUP(TST) NAME(OOPS)  
CLASS(PROCESS)  
OMVS SEGMENT NOT DEFINED
```

- RACFRW information is incomplete
- Use SMF Data Unload utility (IRRADU00)

See: [z/OS Security Server RACF Auditor's Guide](#)
and [z/OS Security Server Macros and Interfaces](#)



The ISPF Shell ... A Panel Interface

- ISPF interface to UNIX administration
 - Create and set up the file system
 - Display/change file attributes
 - copy files to/from data sets
 - Set up z/OS UNIX users and groups
 - Change attributes for z/OS UNIX users
 - Display and manage UNIX processes
 - And much more! ...
- Invoke with TSO ISHELL command
- Normal RACF authority checking applies

36 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

Setup -> User ... can be used to set up a new OMVS user...HFS dataset and all

Setup -> User list ... will show all users and their UIDs. This can be sorted by UID.

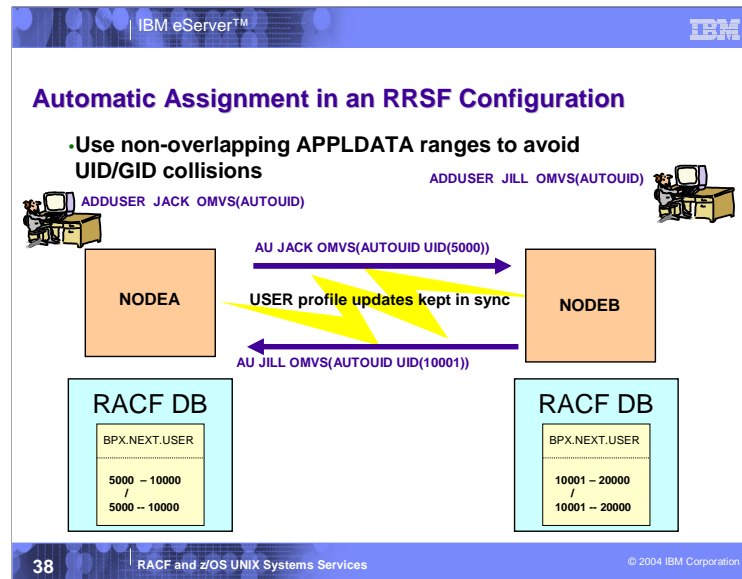
Setup -> All users ... can assign unique UID, initial program, and home directory based on a specified prefix. The UID will start at one higher than the current high value. You will also be asked if you want to assign GIDs to all groups during this setup. If you don't, and a given user's group has no GID, processing will stop.

Setup -> All groups ... can assign a unique GID to all groups. There is no "Group list" function, and so you can't see a sorted list of current GID values.

Setup -> Permit field access ... will do the steps shown in the appendix.

The ISPF Shell ... A Panel Interface

```
File Directory Special_file Tools File_systems Options Setup Help
-----
UNIX System Services
Enter a pathname and do one of these:
- Press Enter.
- Select an action bar choice.
- Specify an action code or command on
1. User...
2. User list...
3. All users...
4. All groups...
5. Permit field access...
6. Character Special...
7. Enable superuser mode(SU)
-----
Return to this panel to work with a different pathname.
More: +
/u/brwells
_____
_____
_____
EUID=0
Command ==>
F1=Help F3=Exit F5=Retrieve F6=Keyshelp F7=Backward F8=Forward
F10=Actions F11=Command F12=Cancel
_____
d 03/047
```



In an RRSF environment where user updates are kept in sync across the network, you want to avoid UID/GID collisions when AUTOUID/AUTOUID is used on multiple nodes. This is accomplished by specifying unique ranges of ids in the BPX.NEXT.USER APPLDATA for each node, thus making sure the various nodes do not derive the same value as another node. Assuming that the nodes are kept in sync, then even explicit UID/GID assignment should not result in collisions, unless the same value is explicitly assigned on different nodes at almost the same time, and they "cross in the mail." Even then, the assignment should fail on the remote node, and the administrator will be notified, and get a chance to fix their error.

IBM eServer™ IBM

Automatic Assignment in an RRSF Configuration

- Use the ONLYAT keyword to manage BPX.NEXT.USER

```
RDEF FACILITY BPX.NEXT.USER APPLDATA('5000-10000/5000-10000')
ONLYAT(NODEA.MYID)
RDEF FACILITY BPX.NEXT.USER APPLDATA('10001-20000/10001-20000')
ONLYAT(NODEB.MYID)
```

39 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

This is only a consideration if you are using RRSF automatic command direction for the FACILITY class. Since you went to so much trouble to plan for non-overlapping BPX.NEXT.USER ranges on each of the nodes, you wouldn't want NODEA's update being propagated to the other nodes and wiping out their local changes! So, you must use the ONLYAT keyword, even when changing the profile on which you are logged on. ONLYAT tells RACF not to propagate the command outbound. Note that the AT keyword is not sufficient, since it will still be subject to propagation.

RACF updates the APPLDATA itself when deriving unique UIDs and GIDs. But RACF is careful to prevent the propagation of these internal updates.

Recap

- UNIX systems require an integer value as a user or group identity
- This identity is contained in the RACF database, which acts as the 'user and group registry'
- Various mechanisms can effect a change in user/group identity of a process
- Traditional UNIX security mechanisms are extended on z/OS
- UNIX security events are audited in a familiar RACF fashion

z/OS Unix System Services File Systems

IBM eServer™ IBM

File Systems are contained in MVS Data Sets

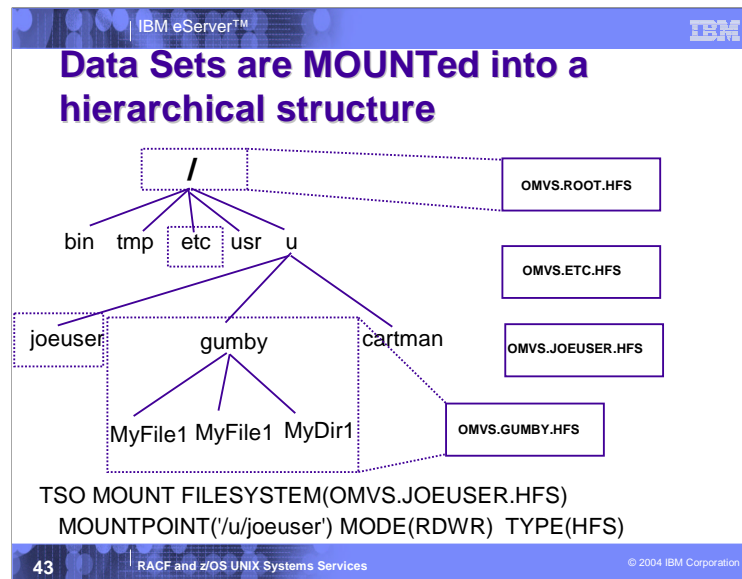
- Created using ALLOCATE (ISPF 3.2)
- Protected from MVS access methods by
 - Naming convention
 - Do not name file systems with User ID high level qualifier
 - RACF profiles (e.g. OMVS.HFS.*)
- Only UNIX processes can access the files and directories contained within the data sets
 - OMVS.HFS.BRUCE
 - OMVS.HFS.CHARLEY
 - OMVS.HFS.DAN
 - OMVS.HFS.GARY

42 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

Since the operating system underlying UNIX System Service is MVS, the UNIX file systems are stored within MVS datasets. You want to prevent end users from accessing these datasets through traditional MVS access methods. You should protect them with RACF profiles.

Can't access the files and directories from the 'outside'. That is, using traditional MVS access methods.

Must access the files using UNIX semantics from a UNIX process.



Mount steps involve:

Allocate the HFS data set

mkdir a home directory for the user in /u

MOUNT the HFS data set (file system)

chown the /u/joeuser directory to joeuser

Mount table specified in BPXPRMxx

file systems can be mounted dynamically with the TSO MOUNT command or the UNIX mount command

view file system with DISPLAY OMVS,F or df command

IBM eServer™ IBM

Using UNIX Files

- UNIX file services integrated with z/OS
 - Invoke UNIX programs from TSO or BATCH; invoke LINKLIB programs from shell
 - Manage file system from shell, TSO, console
 - Open data sets, UNIX files, from any environment
- Enter shell from TSO using OMVS command
- Examples from TSO
 - oedit /u/joeuser/myfile
 - oshell ls -E
- Example from BATCH
 - //FRED DD PATH='/u/fred/list/wilma'
 - ISPF Interface: ISHELL

44 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

There are many ways to access UNIX files. You can enter a UNIX shell with the OMVS command and issue UNIX commands.

Or you can manipulate UNIX files from the TSO command line or from JCL.

UNIX File Security

- UNIX invokes RACF through SAF services
- No profiles in RACF database
- Access control by permission bits and access control lists (ACLs)
 - read, write, execute permissions (non-hierarchical)
 - POSIX-compliant 'owner', 'group', 'other' classes
 - ACL entries for individual users and groups
- File security info stored with file in file system
 - owning UID and GID
 - permission bits and ACLs
 - audit settings
 - extended attributes (APF, program-controlled, etc)

RACF, of course, gets involved in providing security for UNIX files. No RACF profiles, however, are used to provide security. All the security information for a file is kept with the file.

IBM eServer™ IBM


File Access Control with Permission Bits

File Owner

User (UID)	Group (GID)
------------	-------------

Permission Bits

OWNER rwx	GROUP r-x	OTHER ---
--------------	--------------	--------------

 **oedit /etc/profile**

User

effective UID
effective GID
Supplemental Groups

As per the UNIXPRIV profile
RESTRICTED.FILESYS.ACCESS

IF no access, check
SUPERUSER.FILE
SYS in UNIXPRIV
class

NEW!
z/OS R3

[See RACF Security Administrator's Guide for detailed list of steps](#)

46 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

This chart illustrates the checking which is done to determine file access.

It does not illustrate the complete list of steps, but rather, demonstrates the basic POSIX access algorithm, supplemented by RACF's UNIXPRIV checking. This does not include ACL checking. That will be covered later.

The z/OS V1R3 SAG was updated with a comprehensive list of authorization steps, like what has always been in there for RACF profile checking. It's worth a 5-minute read.

See speaker note for the upcoming flowchart slide for an overview of the steps which were omitted from this slide.

OW53183 (R3): UNIXPRIV used for shell test operators

IBM eServer™ IBM

Making the RESTRICTED attribute applicable to UNIX files

NEW!
z/OS R3

- UNIX 'OTHER' bits analogous to RACF profile UACC
 - but RESTRICTED attribute does not apply by default
- Define RESTRICTED.FILESYS.ACCESS in the UNIXPRIV class with UACC(NONE)
 - RESTRICTED applies to 'OTHER' bits system-wide
- For exceptions, permit RESTRICTED user with READ access
 - This does **not** grant access to the file (that's what an ACL is for), it just allows the 'OTHER' bits to be checked

47 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

The RESTRICTED attribute for a user prevents the user from getting access based on the UACC in a RACF profile. In z/OS R3, the profile RESTRICTED.FILESYS.ACCESS profile in the UNIXPRIV class allows that to apply to UNIX files as well.

Note that RESTRICTED.FILESYS.ACCESS will be checked for RESTRICTED users regardless of whether an ACL exists, so this function can be exploited regardless of whether you plan to use ACLs or not.

Using UACC(READ) on RESTRICTED.FILESYS.ACCESS doesn't work, since by definition, a RESTRICTED user cannot be granted access via UACC!!! This would be a meaningless thing to do anyway, given that you simply would not define RESTRICTED.FILESYS.ACCESS if you want 'other' bits to be checked for RESTRICTED users.

Even today, a RESTRICTED user can be permitted to SUPERUSER.FILESYS and thus be granted access to files, though I cannot think of why anyone would do that! In any case, it will continue to work that way.

IBM eServer™ IBM

Output of ls (list files) Command

```
# ls -E
total 192
-rw-r--r--+ --s- 1 BPXROOT 2001      700 Mar 20 16:45 Odyssey
--wx--S--- --s- 1 ACE     SYS1      30 Aug 23 2001 Program2
-r-srwsrws --s- 1 BPXROOT KNIGHTS  8240 Aug 23 2001 SetuidPgm
drwxr-xr-x  2 BPXROOT SYS1      8192 Mar 20 16:38 TestDirectory
-rwsr---t  --s- 1 ACE     JESTER    8240 Aug 11 2001 prog1
-rwsr-x-x  ---- 2 BPXROOT SYS1      8240 Aug 11 2001 rac
lrwxrwxrwx  1 BPXROOT SYS1        3 Aug 20 16:43 racSymlink->rac
-rwsr-x-x  ---- 2 BPXROOT SYS1      8240 Mar 11 2001 raclink
-rwsr-x---  aps- 1 BPXROOT SYS1      8240 Aug 20 16:39 racp
-rw-r--r-- --s- 1 1969    SYS1       99 Mar 20 16:46 woodstock
```

Annotations:

- File type and permissions: -rw-r--r--+
- User and group number: 1 BPXROOT 2001
- File name: Odyssey
- Extended attributes: aps-
- Number of links: 2

48 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

owning UIDs and GIDs are translated to RACF user IDs and group names, if possible. UNIXMAP or AIM will speed this up, especially when there are unassigned UIDs/GIDs

The file Odyssey is owned by a group which could not be mapped. woodstock is owned by a user which could not be mapped. This can happen from chown being issued with a numeric, NFS mounting a remote file system, or a user having been deleted

SetuidPgm has the setuid bit on, Program2 has the setgid bit on, but the execute bit is not on.

prog1 has the sticky bit on. It will be fetched from LINKLIB

racp has the program control and apf bits on (need AC=1)

raclink is a hard link to rac, racSymlink is a symbolic link

+ in the permissions means that an ACL entry exists

IBM eServer™ IBM

Using the UNIX 'find' command

- find can search for files using all sorts of criteria
 - file type
 - user and group ownership
 - presence of ACLs
 - presence of specific ACL entries
 - file permissions (including set-uid/set-gid bits)
 - audit settings
- Use find and shell command substitution
 - `setfacl -m g:racftest:rxw $(find /u/bruce -acl_group racfdev)`
- See UNIX Command Reference

49 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

When you need to update many UNIX files with the same updated, the UNIX find command is very useful. Use the find command to 'select' the files, and then use shell command substitution to make the output of the find command into input for another command (such as setfacl).

Since ACLs are not 'shared', there will normally be lots of them dispersed throughout your HFS. the find command, in conjunction with shell substitution, will help you to manage them over time E.G.

to add an entry to every directory within a subtree

to permit the group RACFTEST to every ACL which has an entry for RACFDEV (this is the example shown)

to permit a user to all files owned by JOEUSER

to delete all ACLs within a given subtree

See the UNIX Planning Guide for more examples

IBM eServer™ IBM

chown Command - Change File Owner

- Change owning user and group of a file
 - `chown tiger:golf /u/arnie/store`
- Change owner of all files in a directory
 - `chown lou /prog/ibm/*`
- Change owner of all files in a directory, and its subdirectories
 - `chown -R uxadmin /u/deluser`
- Change owner of all of lou's files to sam
 - `chown sam $(find /u -user lou)`
- Change owner of all orphaned files to BYE
 - `chown bye $(find /u -nouser)`
- Change owning group of a file
 - `chgrp $stestgrp myfile`

50 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

This shows various formats of the chown/chgrp command, and demonstrates some features of this particular command, and of the UNIX shell in general

the * is a shell wild card operator. In this example, the shell will apply the chown command to any file/directory within the /prog/ibm directory (but will not recurse to lower levels)

chown, and some other commands (e.g. chmod, but not sefacI) have a 'recursive' option, -R.

shell command substitution can be used to apply a command to a list of file names returned by 'find'

The dollar sign is a special shell character, so if it is used in a group name, it must be 'escaped' by preceding it with a backslash.

IBM eServer™ IBM

chmod Command - Change File Mode (permissions)

- change permissions of a file
 - chmod u=rwx,g=rwx,o=rx a-file
- change permissions of a file with octal notation
 - chmod 775 a-file
- Set **a**ll read bits on for all files in a directory and its subdirectories using relative perms
 - chmod -R **a**+r MyDirectory

51 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

This slide demonstrates some of the basic chmod options. You can use symbolic form or octal notation.

Bit definition is:

4000 Set-user-ID bit

2000 Set-group-ID bit

1000 Sticky bit

0400 User read

0200 User write

0100 User execute (or list directory)

0040 Group read


0020 Group write


0010 Group execute

0004 Other read

0002 Other write

0001 Other execute

IBM eServer™ 

Access Control Lists (ACLs) 

- Contained within the file system
 - File security is portable
- Enabled with SETROPTS CLASSACT(FSSEC)
 - Can be defined prior to activating FSSEC
- Not implemented by RACF profiles
 - Access algorithm behaves as much as possible like that of RACF profile access
 - UNIXPRIV profiles do affect file access checking
- Can contain a maximum of 1024 entries
 - Entry consists of a type (user or group) and identifier (UID or GID) and permissions (read, write, and execute)

52 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

ACL is Deleted automatically with file (even on downlevel systems)

Managed by RACF using SAF callable services

R_setfacl, makeFSP, ck_access, query_file_security_options

Access Control Lists (ACLs) ...

- Are displayed with the `getfacl` UNIX command and created, modified, and deleted with the `setfacl` UNIX command
 - Must be UID(0), file owner, or have READ access to UNIXPRIV resource
SUPERUSER.FILESYS.CHANGEPERMS
- Support inheritance
 - 3 types of ACLs
 - Access ACL
 - Directory Default ACL
 - File Default ACL

IBM eServer™ IBM

File Access Control with Permission Bits and ACLs

Permission Bits

	OWNER rwx	GROUP rwx	OTHER rwx
ACL c o i n s e t t e r s o l	User1 rwx	Group1 rwx	As per UNIXPRIV profile RESTRICTED.FILESYS.ACCESS
	User2 rwx	Group2 rwx	IF no access, check SUPERUSER.FILESYS or SUPERUSER.FILESYS.ACLOVERRIDE
	Usern rwx	Groupn rwx	

IF FSSEC class active [See RACF Security Administrator's Guide for detailed list of steps](#)

54 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

This chart extends the previously shown access algorithm to include ACLs.

Order of checking

- 1) Owner bits
- 2) User ACL entries
- 3) Group bits
- 4) Other bits

getfacl and setfacl commands

- Can also be used to display/modify the POSIX permission bits
 - allows use of a single interface
 - chmod only necessary to set sticky, set-uid, and set-gid bits
- ACL can be set from contents of a file
 - thus, output of getfacl can be piped into setfacl via stdin
 - Reduces typing
 - Allows use of "named ACLs"

IBM eServer™ IBM

getfac1 ...

- getfac1 Myfile
 - Displays file name, user owner, and group owner
 - Displays base POSIX permissions in "acl format"
 - These can be suppressed

```
#file: MyFile
#owner: BPXROOT
#group: SYS1
user::rw-
group::r--
other::r--
```

56 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

This shows the output of the getfac1 command on a file with no ACL entries.

–m Specifies that the comment header (the first three lines of each file’s output) is not to be displayed.

–o Displays only the extended ACL entries. Does not display the base ACL entries.

IBM eServer™ IBM

setfac1 ...

- Create an access ACL with an entry for user bruce and group racf
- **setfac1 -m user:bruce:rwx,group:racf:r-x MyFile**
- **getfac1 MyFile**

```

#file: MyFile
#owner: BPXROOT
#group: SYS1
user::rw-
group::r-x
other::r--
user:BRUCE:rwx
group:RACF:r-x

```

says modify acl entry, or add it if it does not exist

57 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

setfac1 can also be used to change the posix permission bits. This lets you use getfac1/setfac1 without having to also do ls and chmod commands (though ls and chmod must be used to see/update the set-uid-, set-gid, and sticky bits)

to update the posix bits, use the 'u' qualifier for the owner bits, the 'g' qualifier for the group bits, and the 'o' qualifier for the other bits, and omit the middle qualifier for the user or group name. For example, the moral equivalent of **chmod 755 myfile** would be **setfac1 -m u::rwx,g::r-x,o::r-x myfile**

IBM eServer™ IBM

setfac ...

- Delete entry for bruce
 - `setfac -x user:bruce MyFile`
- Replace ACL with specified contents
 - `setfac -s user:jim:r-x,u::rwx,g::r-x,o::--- MyFile`
- Replace ACL with entries contained within a file
 - `setfac -S acfile MyFile`
- Create ACL from existing ACL
 - `getfac thatfile | setfac -S - thisfile` ("-" denotes stdin)
- Delete the access ACL
 - `setfac -D a MyFile`
- Apply ACL entries in file 'TeamX' to all directories in a subtree
 - `setfac -S TeamX $(find /u/joeuser/projectX -type d)`

58 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

the -x option removes the specified ACL entries. The permission qualifier need not be specified

the -s option means replace the current contents of the ACL with the entries specified on the command line. -s requires that you also specify the posix permissions

the -s, -m, and -x options have capital letter options which take their input from a file. Note that I can maintain an ACL in a file (a 'named ACL', if you will) and apply that ACL to files at will.

'-' as a file name is a special notation for stdin, which allows you to pipe output from a getfac command into setfac

And of course, our old friend 'find' is our only mechanism to apply setfac to a set of files.

IBM eServer™ IBM

Overriding UNIXPRIV authority with ACL entries

- By default, UNIXPRIV authority will override a restrictive ACL entry
- To have ACL entries override on a system-wide basis, define UNIXPRIV class profile named SUPERUSER.FILESYS.ACLOVERRIDE with UACC(NONE)
- To make an exception, permit a user/group with whatever access they require to SUPERUSER.FILESYS
- Override profile only checked if an ACL entry (user or group) denied file access

59 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

This will **not** override UID(0)/trusted/privileged

Can use this to provide a mechanism of scoping UNIXPRIV access authority to certain file system subtrees

Define the override profile with UACC(NONE)

Define ACL on top directory of a given subtree and permit the user (or group) with limited access permissions

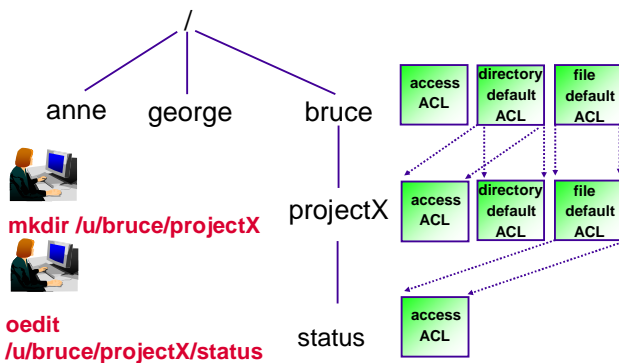
if permitting by group, make sure the 'group' bits, or another group entry does not provide access

ACL Inheritance

- Can establish default (or 'model') ACLs on a directory
- Get automatically applied to new files/directories created within the directory
- Separate default used for files and subdirectories
- Reduces administrative overhead

ACL inheritance allows you to automatically apply ACL protection to newly created objects

ACL Inheritance ...



getfac and setfac ...

- Create a directory default ACL
 - setfac -m default:user:bruce:rwx MyDir
 - or: setfac -m **o**:u:bruce:rwx MyDir
 - getfac -d MyDir

#file: MyDir
#owner: BPXROOT
#group: SYS1
default:user:BRUCE:rwx

additional
qualifier for
directory
default

getfac and setfac ...

- Create a file default ACL
 - `setfac -m fdefault:user:bruce:rwX MyDir`
 - or: `setfac -m fu:bruce:rwX MyDir`
 - `getfac -f MyDir`

#file: MyDir
#owner: BPXROOT
#group: SYS1
fdefault:user:BRUCE:rwX

additional
qualifier for
file default

IBM eServer™ IBM

getfacI ...

- Display all ACLs for a directory
 - `getfacI -adf MyDir`
 - #file: MyDir
 - #owner: BPXROOT
 - #group: SYS1
 - user::rwx
 - group::r-x
 - other::r-x
 - user:JOE:--x
 - user:BUCK:rwx
 - fdefault:user:ACE:r-x
 - fdefault:user:BUCK:rwx
 - default:user:DARTH:rwx
 - default:group:RACF:--x

specifies all three
acl types

64 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

In -adf:

a is access ACL

d is directory default ACL

f is file default ACL

IBM eServer™ IBM

Default file permissions and the umask command

- Files are created with different permission settings, depending on the command or application
- file mode creation mask (umask) defends user against permissive defaults
- Display umask
 - octal format: `umask 0077`
 - symbolic format: `umask -S u=rwx,g=,o=`
- Set umask so group and other write bits cannot be set during file creation
 - `umask g-w,o-w`
 - usually done from `/etc/profile`, and `.profile`

Command	Per-mis-sions
OPUT	600
touch	666
Redirection ('>')	666
oedit	700
mkdir	777

65 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

The initial file permissions are as specified in the mode parameter of the `open()` call made by the creating application

`umask` specifies which permission bits are to be masked off during file creation. Typically, this is used to mask off the group and other 'write' bits.

This is typically done on a system-wide basis by `/etc/profile`, and on an individual user basis by the `.profile` script in the user's home directory. For you VMers out there, these files are analogous to `SYSPROF EXEC` and `PROFILE EXEC`, respectively.

`umask` **cannot** be used to initialize bits **on**

Programs in the File System

- Can designate program as APF
 - `extattr +a myprogram`
 - requires READ to FACILITY profile
BPX.FILEATTR.APF
 - `find / -attr a`
- Can designate a program as RACF program-controlled
 - `extattr +p myprogram`
 - requires READ to FACILITY profile
BPX.FILEATTR.PROGCTL

Whenever an APF or program-controlled file is written to, the extended attributes are reset, for integrity reasons. Only an authorized user (e.g. SMPE) can turn the bits back on.

Note that UID(0) is not sufficient to turn these attributes on! This aids in enforcing z/OS UNIX level security for servers and daemons.

Programs in the File System ...

- Can indicate that a file system executable is to be obtained from traditional MVS search order (LPA and LINKLIB) by turning on the sticky bit
 - `chmod +t myprog`
 - must be owner or have superuser privilege
 - program name must adhere to MVS conventions (8 characters)
- set-UID and set-GID programs
 - change UNIX identity of user
 - see related presentation “Defining and Protecting UNIX Identities” (session #D3)

Whenever a set-uid or set-gid file is written to, the set-uid and set-gid bits are reset, for integrity reasons.

Similarly, if the file owner is changed (user and/or group), the appropriate set-id bit is reset

IBM eServer™ IBM

UNIX File Auditing

- Controlled by audit classes
 - SETR LOGOPTIONS, SETR AUDIT
 - DIRSRCH,DIRACC,FSOBJ,FSSEC
- And by file-level audit options
 - Similar to RALTER AUDIT() and GLOBALAUDIT()
 - Set with chaudit, not ALTDSD or RALTER
 - RACF AUDITOR can read and search any directory
- RACF UAUDIT attribute honored
- Failing mounts/unmounts always audited
- Always:
`SETR LOGOPTIONS(ALWAYS(FSSEC)) !!!`

68 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

There are three separate access-related classes: DIRSRCH, DIRACC, and FSOBJ. These log accesses to files and directories, and split into separate classes for auditing granularity. Especially DIRSRCH. You do not want to log successful directory searches. On the other hand, you may want to do so with directory accesses (DIRACC) or file accesses (FSOBJ). SETR CLASSACT has no effect on these classes.

FSSEC logs attempts to alter security information (chown, chmod, setfacl, etc). Kind of like auditing changes to RACF profiles. Unlike the 3 classes above, you will not get this logging by default. Also unlike the above, SETR CLASSACT(FSSEC) has meaning...it enables ACLs.

IBM eServer™ IBM

chaudit Command: Setting File-level Auditing Options

- Audit successful write access to a file
 - `chaudit w+s myfile`
- Audit all access to a file
 - `chaudit +sf myfile`
- Set auditor audit bits to audit all attempts to execute a program
 - `chaudit -a x+sf myprog`
- Audit all write and execute accesses to set-uid files
 - `chaudit x+sf,w+sf $(find / -perm -4000)`

69 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

a user with the RACF AUDITOR attribute can change the AUDITOR file-level settings for any file

a superuser can change the owner file-level settings for any file

a general user can set the owner file-level settings only for files which (s)he owns

There is no UNIXPRIV profile which controls audit settings

IBM eServer™ IBM

Output of Is (list files) Command

```
# ls -W
total 192
-rw-r--r--+ --- 1 BPXROOT 2001      700 Mar 20 16:45 Odyssey
--wx--S--- --- 1 ACE      SYS1       30 Aug 23 2001 Program2
-r-srwsrws -aa --- 1 BPXROOT KNIGHTS 8240 Aug 23 2001 SetuidPgm
drwxr-xr-x fff --- 2 BPXROOT SYS1     8192 Mar 20 16:38 TestDirectory
-rwsr---t  --- --a 1 ACE      JESTER    8240 Aug 11 2001 prog1
-rwsr-x--x --- --- 2 BPXROOT SYS1     8240 Aug 11 2001 rac
lrwxrwxrwx fff --- 1 BPXROOT SYS1       3 Aug 20 16:43 racSymlink->rac
-rwsr-x--x --- --- 2 BPXROOT SYS1     8240 Mar 11 2001 raclink
-rwsr-x--x --- --- 1 BPXROOT SYS1     8240 Aug 20 16:39 racp
-rw-r--r-- -s- --- 1 1969     SYS1       99 Mar 20 16:46 woodstock
```

owner audit settings

auditor audit settings

f = failures
s = successes
a = all (successes and failures)

70 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

ls -W shows the audit settings

the first column of audit settings shows the owner options, the 2nd column shows the auditor options.

'f' indicates that failed accesses for the relative permission (e.g. read, write, execute access) are being logged

's' indicates that successes are being logged

'a' indicates that all accesses (successes and failures) are being logged

the find command can locate files with certain audit settings

IBM eServer™ IBM

File System Security Reporting - HFS Unload!

- irrhfsu command available on
 - <http://www.ibm.com/servers/eserver/zseries/zos/racf/goodies.html>
- Reports on HFS security data like IRRDBU00 reports on RACF profile data
- Creates Type 900 record for each file
 - currently-mounted file systems only
- Creates Type 90n record for each ACL entry
- Runs as UNIX command, or from batch
 - irrhfsu /etc > HfsuOutFile
 - irrhfsu -f //JOEUSER.HFSU.OUTPUT /u/joeuser/dir1 dir2/subdir

71 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

For ACLs:

Type 901 describes an access ACL entry

Type 902 describes a file default ACL entry

Type 903 describes a directory default ACL entry

Recap

- UNIX file systems are contained in MVS datasets
- File systems are mounted at 'mount points' (directories) to create a hierarchical file system
- File security information is contained within the file system (not in the RACF database), and is managed using UNIX commands and interfaces
- Access Control Lists allow you to specify a list of users who can access a file
- Programs in the file system can be APF authorized and program-controlled
- Actions are auditable through RACF and SMF and can be reported on using the SMF Data Unload utility (IRRADU00)

IBM eServer™ IBM

Good Sources of Information

- UNIX System Services web site, at
 - <http://www.ibm.com/servers/eserver/zseries/zos/unix/>
Check out the Tools and Toys page
- UNIX System Services Planning manual and UNIX System Services Command Reference
 - Available online at
 - For OS/390: <http://www.ibm.com/servers/s390/os390/bkserv/>
 - For z/OS: <http://www.ibm.com/servers/eserver/zseries/zos/bkserv/>
- mvs-oe mailing list (see the Forums link at the UNIX web site above for information)

73 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

BPX1SEC1 SAMPLIB - TSO Clist of RACF commands for security setup

Good Sources of Information ...

- RACF web site, at
 - <http://www.ibm.com/servers/eserver/zseries/zos/racf/>
See Downloads page for HFS Unload
- RACF Security Administrator's Guide (UNIX chapter) and RACF Auditor's Guide
 - Available online at
 - For OS/390: <http://www.ibm.com/servers/s390/os390/bkserv/>
 - For z/OS: <http://www.ibm.com/servers/eserver/zseries/zos/bkserv/>
- racf-l mailing list (see the front-matter in any RACF book for information)

Questions?

Supplemental Material

Terminology

- POSIX - Portable Operating System Interface
- FSP - File Security Packet
- process - UNIX 'address space'
- USP - User Security Packet
- ACL - Access Control List
- SAF - System Authorization Facility
- APF - Authorized Program Facility

Initialized to...	FSP contents	Changed by...				
Effective UID	User (UID) owner	chown command				
Parent dir's group	Group (GID) owner	chown or chgrp				
Varies by function (qualified by umask)	Permission bits	chmod command				
	<table border="1"> <tr> <td>owner</td> <td>group</td> <td>other</td> </tr> <tr> <td>r w x</td> <td>r w x</td> <td>r w x</td> </tr> </table>		owner	group	other	r w x
owner	group	other				
r w x	r w x	r w x				
set-id bits off, sticky bit specified by fn	Flags Directory, set-uid, set-gid, sticky bit	chmod command				
read, write and execute failures	Owner audit options	chaudit command				
	<table border="1"> <tr> <td>read</td> <td>write</td> <td>execute</td> </tr> </table>		read	write	execute	
read	write	execute				
no auditing	Auditor audit options	chaudit -a command				
	<table border="1"> <tr> <td>read</td> <td>write</td> <td>execute</td> </tr> </table>		read	write	execute	
read	write	execute				
SHAREAS bit on for executable files	Extended attributes	extattr command				
contents of parent's default ACL	Access Control List	setfacl command				

the extended attributes are also security-related, though technically speaking, they do not reside in the FSP.

The apf attribute marks an HFS program as APF authorized. It is set with the extattr +a progname command, and requires authority to BPX.FILEATTR.APF in the FACILITY class

the progctl bit marks an HFS program as being program controlled. It is set with the extattr +p progname command, and requires authority to BPX.FILEATTR.PROGCTL in the FACILITY class

setuid, setgid and APF bits get reset when the file is changed

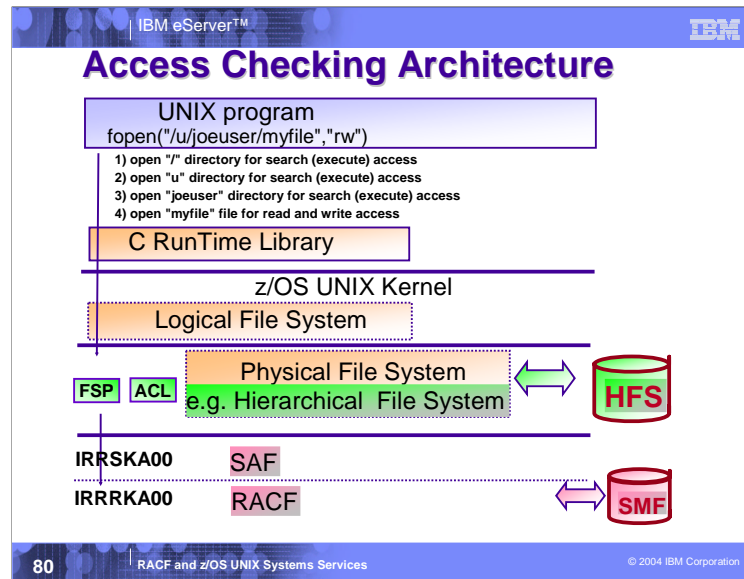
IBM eServer™ IBM

UNIX File Security Packet (FSP) ... who can change what?

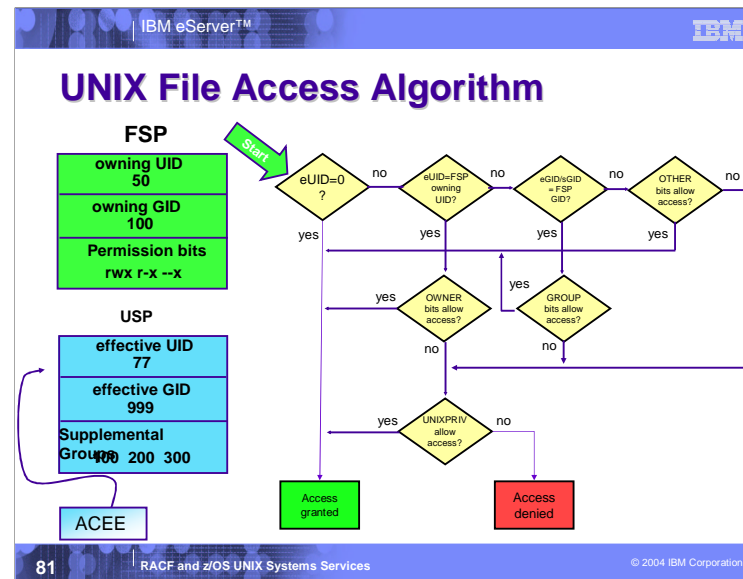
Security Field	Required Authority
Owning UID	<ul style="list-style-type: none"> ▪UID 0 ▪File owner if CHOWN.UNRESTRICTED is defined in the UNIXPRIV class ▪READ access to UNIXPRIV resource SUPERUSER.FILESYS.CHOWN
Owning GID	<ul style="list-style-type: none"> ▪UID 0 ▪Owner, if a member of new group ▪File owner if CHOWN.UNRESTRICTED is defined in the UNIXPRIV class ▪READ access to UNIXPRIV resource SUPERUSER.FILESYS.CHOWN
File mode (permissions and flags) and ACL	<ul style="list-style-type: none"> ▪UID 0 ▪File owner ▪READ access to UNIXPRIV resource SUPERUSER.FILESYS.CHANGEPERMS
Owner audit options	<ul style="list-style-type: none"> ▪UID 0 ▪File owner
Auditor audit options	▪RACF Auditor
Extended attributes	READ access to FACILITY class resource named: <ul style="list-style-type: none"> ▪APF – BPX.FILEATTR.APF ▪Program control – BPX.FILEATTR.PROGCTL ▪Shared library – BPX.FILEATTR.SHARELIB

79 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

With z/OS V1R2, SUPERUSER.FILESYS.CHANGEPERMS in the UNIXPRIV class will authorize the use of chmod against any file/directory.



This shows the underlying architecture for an access check. One file access is actually translated into a search access for each directory along the way before the file access check is performed.



This algorithm shows the basic flow, but omits TRUSTED/PRIVILEGED checking for superuser superuser exception for execute access (at least 1 execute bit must be on) exception for RACF AUDITOR (can read and search any directory) unauthenticated client processing (both client and server must be authorized to the file; the same algorithm is applied to both) Note that the RESTRICTED attribute *can* apply to UNIX files (OTHER bits treated like RACF UACC) What access is allowed in this example? r-x because one of the user's supplemental groups owns the file

UNIX File Group Ownership

- UNIX files have an owner (UID) and an owning group (GID)
- Previously owning group inherited from directory
- Can now choose
 - Assign from owner of directory, as before
 - Assign from effective GID of user creating the file
- RDEFINE UNIXPRIV
FILE.GROUPOWNER.SETGID
 - Directory set-gid bit on - assign from directory
 - Directory set-gid bit off - assign from user



IBM eServer™ IBM

Compatibility concerns with ACLs

- ACLs can be defined within a file system on an uplevel sysplex node
- File systems can be accessed from downlevel nodes
- Access attempts from downlevel nodes cannot be sure of security intent
- APAR OW53646 will force access failures on downlevel systems for files with ACLs

Node 1: z/OS V1R3, create, HFS

Node 2: OS/390 V2R10, access

Duhhh... What's an ACL?

84 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

In a SYSPLEX environment, the node on which the file system is mounted is called the 'server'. The node which is accessing a file in this filesystem is called the 'client'. The actual access check is performed by the client, based on the FSP (which contains the base permission bits), and now also the ACL, which is retrieved from disk by the server and shipped to the client. In this scenario, the server (Node 1) has all the information (i.e. FSP and ACL) necessary to make an access decision, but the client (Node 2) does not know to ask for the ACL in the first place, and certainly does not know how to apply ACL checking rules. Only the FSP is returned from Node 1.

IBM eServer™ IBM

Auditing - Anatomy of a Violation

```

CH408I USER(HUMBERT) GROUP(LOSERS ) NAME(HUMBERT HUMBERT)
/u/bruce/work/projectX/secret/documents/Forecast
CL(DIRSRCH ) FID(01C7D5D9D3F1F2001E04000004530000)
INSUFFICIENT AUTHORITY TO OPEN
ACCESS INTENT(--X) ACCESS ALLOWED(OTHER --)
EFFECTIVE UID(0000000295) EFFECTIVE GID(0000000521)

```

- Humbert tried to **OPEN** this file, but was denied. Why?
- If the class were FSOBJ, we would know that Humbert did not have permission to the file named 'Forecast' (same would be true if class were DIRACC)
- But, the class is **DIRSRCH**, which indicates that Humbert did not have search (execute) access to some directory component of the path name
- We are stuck listing each directory until we see some **OTHER** bits which are restricting access (this could be an iterative process). This part of the message might also have identified the OWNER or GROUP bits, or a USER or GROUP ACL entry
 - getfacl -e humbert /u/... will narrow down the output to applicable entries (i.e. any user ACL entry for Humbert, and any group ACL entry for any of Humbert's groups)

85 RACF and ZUS UNIX Systems Services © 2004 IBM Corporation

Remember, you only get violations if the access attempt was audited, which it will be by default

Note that failed attempts to change file security info (e.g. chown) will **not** be logged (FSSEC class) by default. FSSEC will be in the SETROPTS LOGOPTIONS DEFAULT list, but, there are no file-level audit options which control changes to file security (they **only** apply to accesses).

The effective UID and GID are shown, in case the user has changed their identity (using a set-uid file, for example). Much confusion results otherwise.

the auditid tool from the UNIX tools and toys page can be used to match pathnames to FIDs.

IBM eServer™ IBM

Auditing UNIX Files: compared with data sets

DATASET Auditing	UNIX File Auditing
SETROPTS LOGOPTIONS for DATASET class controls access logging	SETROPTS LOGOPTIONS for FSOBJ, DIRACC, and DIRSRCH classes controls access logging
SETROPTS AUDIT(DATASET) audits profile creation/deletion	SETROPTS AUDIT(FSOBJ) audits file creation/deletion
SETROPTS AUDIT(DATASET) audits changes to RACF profiles	SETROPTS LOGOPTIONS for FSSEC audits changes to file owner, permission bits and audit settings
Profile-level auditing can be specified by profile OWNER (AUDIT keyword of ALTDSD)	File-level auditing can be specified by file owner (chaudit command)
Profile-level auditing can be specified by auditor (GLOBALAUDIT keyword of ALTDSD)	File-level auditing can be specified by auditor (chaudit command with -a option)

86 RACF and z/OS UNIX Systems Services © 2004 IBM Corporation

You'll find that UNIX file auditing has been designed to closely mirror the auditing that RACF does with system-wide settings (SETROPTS) and file-level audit options:

- files have owner audit settings and AUDITOR audit settings
- by default, failed accesses are logged
- SETROPTS LOGOPTIONS ALWAYS and NEVER for the FSOBJ, DIRACC, and DIRSRCH classes override file-level options
- SETROPTS LOGOPTIONS SUCCESSES and FAILURES for the FSOBJ, DIRACC, and DIRSRCH classes 'merges' with file-level options

Auditing UNIX Files: compared with data sets ...

DATASET Auditing	UNIX File Auditing
LOGOPTIONS with ALWAYS and NEVER overrides profile settings	Same for file settings
LOGOPTIONS with SUCCESSES or FAILURES merged with profile-level settings	Same for file settings
LOGOPTIONS with DEFAULT uses the profile-level settings	Same for file settings
Default profile setting is READ failures for owner options (implies UPDATE, CONTROL, and ALTER failures too), and no settings for auditor options	Default is read, write, and execute failures for owner settings (note that UNIX permissions are not hierarchical – they are separate settings for each access type)
Display profile options with LISTDSD	Display file options with ls -W

HFS Unload

- Integrate it with current IRRDBU00 procedure

```
//JOEUSERL JOB '577018,B0011038','Joe User',  
// CLASS=2,NOTIFY=JOEUSER,MSGLEVEL=(1,1),  
// MSGCLASS=H  
//*****  
//HFSUNLD EXEC PGM=BPXBATCH,  
// PARM='PGM irrhfsu -f //SYS1.IRRDBU00.OUTPUT /'  
//STDERR DD PATH='/u/joeuser/hfsuerr',  
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),  
// PATHMODE=SIRWXU
```

This job appends the HFS unload output to your Database unload output.