

IBM eServer™

RACF and z/OS UNIX System Services: File Protection and Auditing

Session #D4

Laurie Ward

IBM Corporation, RACF Development

(845) 435-8028

LWard@us.ibm.com

June 2004

© 2004 IBM
Corporation

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

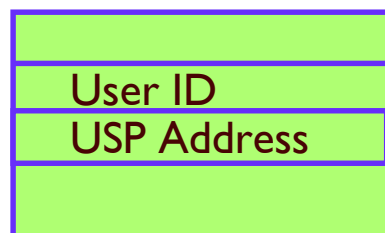
- UNIX File Systems and MVS Datasets
- File Ownership
- File Permissions
- Access Control Lists
- umask and Default File Permissions
- Executable Programs
- Auditing/Reporting



UNIX Identity

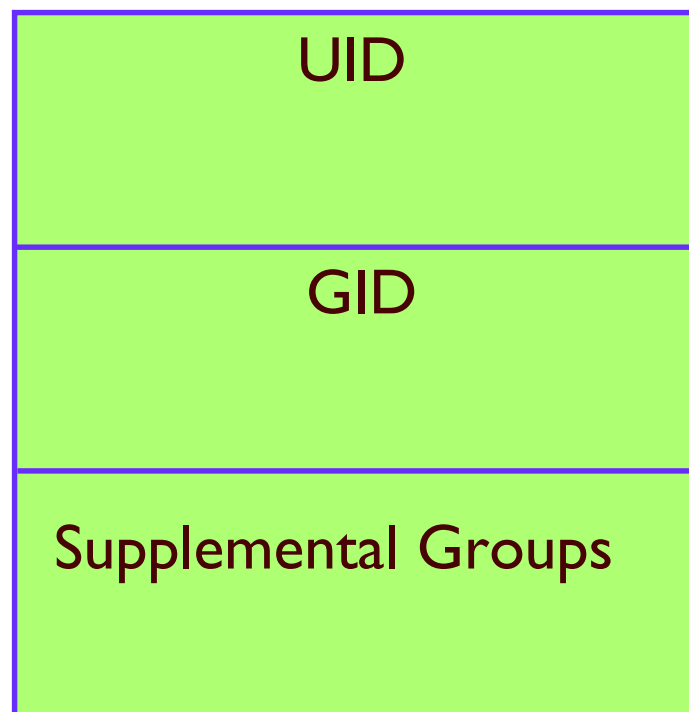
LOGON TSO

ACEE



OMVS

User Security Packet (USP)



From user's **OMVS** segment
or from **BPX.DEFAULT.USER**

From **OMVS** segment of
user's default group or from
BPX.DEFAULT.USER

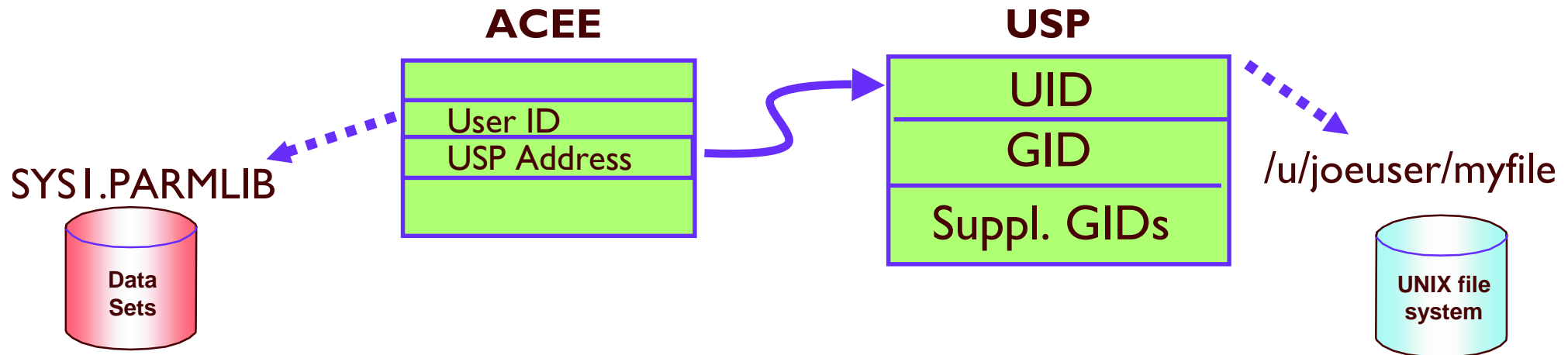
From **OMVS** segments of
user's list of groups

- **USP** created when first **UNIX** service is invoked
- use the **id** command to show user's **UNIX** identity

```
# id joeuser
```

```
uid=34(JOEUSER) gid=521(TENNIS) groups=4(GOLF),16(SOCCER)
```

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

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

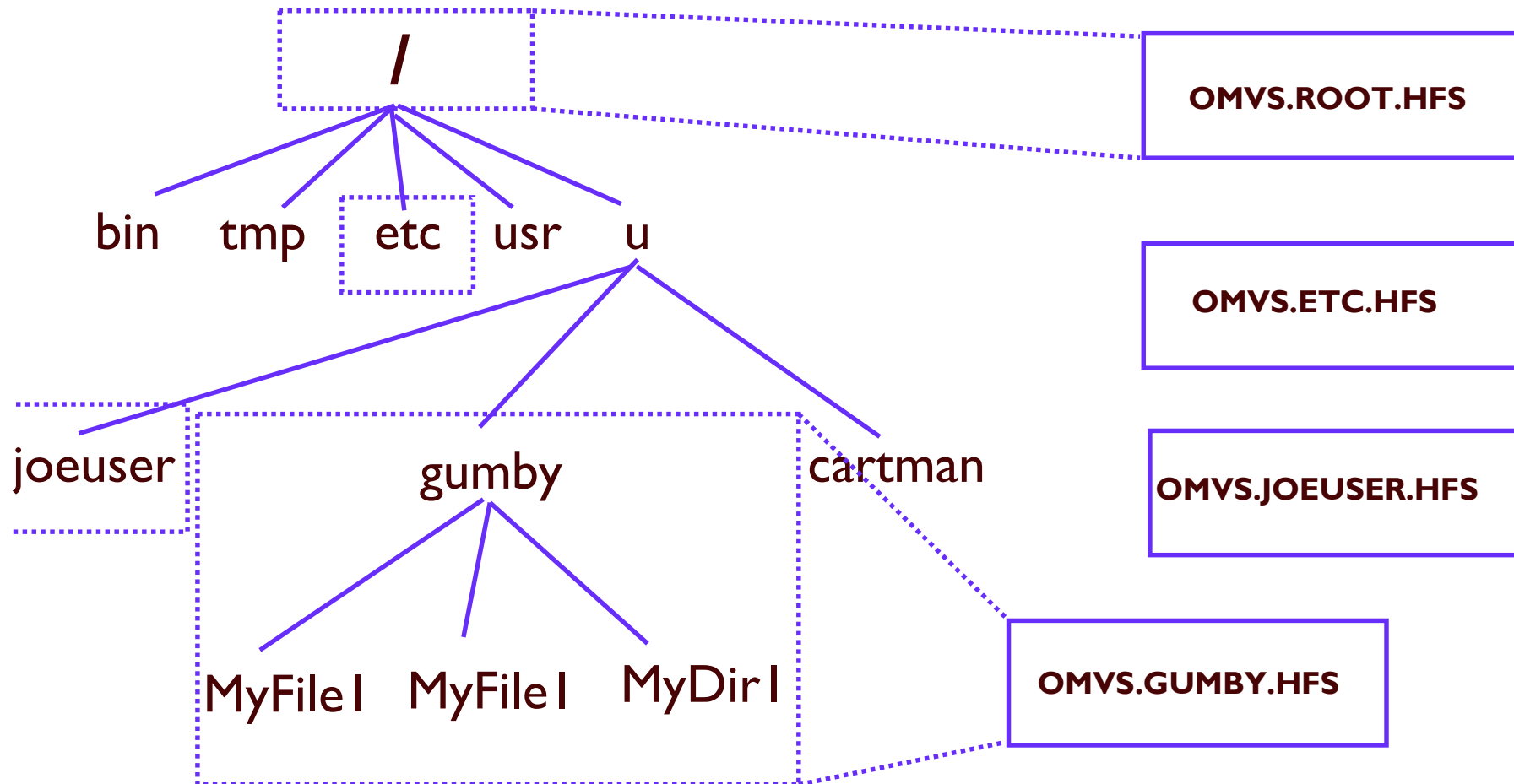
OMVS.HFS.BRUCE

OMVS.HFS.CHARLEY

OMVS.HFS.DAN

OMVS.HFS.GARY

Data Sets are MOUNTed into a hierarchical structure



TSO MOUNT FILESYSTEM(OMVS.JOEUSER.HFS)

MOUNTPOINT('/u/joeuser') MODE(RDWR) TYPE(HFS)

Using UNIX Files

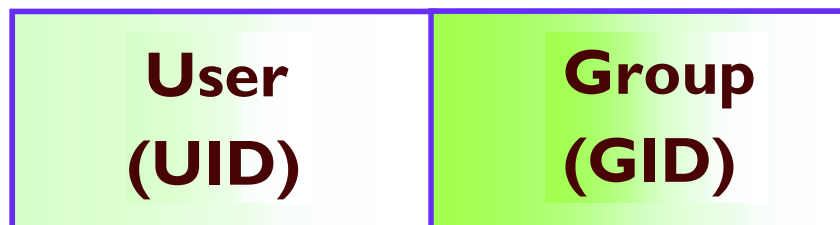
- UNIX file services integrated with z/OS
 - Invoke UNIX programs from TSO or BATCH; invoke LINKLIB programs from shell
 - Manage file system from shell, TSO, console
 - Open data sets, UNIX files, from any environment
- Enter shell from TSO using OMVS command
- Examples from TSO
 - oedit /u/joeuser/myfile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

UNIX File Security

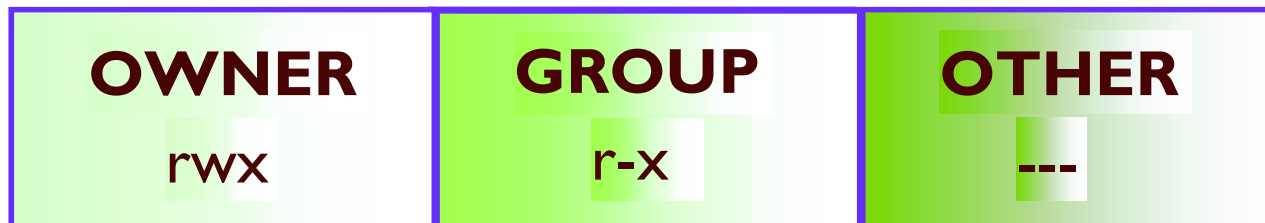
- UNIX invokes RACF through SAF services
- No profiles in RACF database
- Access control by permission bits and access control lists (ACLs)
 - read, write, execute permissions (non-hierarchical)
 - POSIX-compliant 'owner', 'group', 'other' classes
 - ACL entries for individual users and groups
- File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

File Access Control with Permission Bits

File Owner



Permission Bits



oedit /etc/profile

User

effective UID
effective GID
Supplemental Groups

As per the UNIXPRIV profile
RESTRICTED.FILESYS.ACCESS



IF no access, check
SUPERUSER.FILESYS
in UNIXPRIV class

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

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

Output of ls (list files) Command

ls -E
total 192

file type and permissions

user and group owner

file name

-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

extended attributes

number of links

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:rwx $(find /u/bruce -acl_group racfdev)`
- See UNIX Command Reference

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 \testgrp myfile`

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`



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)

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

File Access Control with Permission Bits and ACLs

Permission Bits

	OWNER rwx	GROUP rwx	OTHER rwx
ACL	User1 rwx	Group1 rwx	
coi	User2 rwx	Group2 rwx	
ns	Usern rwx	Groupn rwx	
ett			
sr			
so			
l			

IF FSSEC class active

As per UNIXPRIV profile
RESTRICTED.FILESYS.ACCESS

IF no access, check
SUPERUSER.FILESYS or
SUPERUSER.FILESYS.ACLOVERRID

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

getfac and setfac 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 getfac can be piped into setfac via stdin
 - Reduces typing
 - Allows use of "named ACLs"

getfac ...

- getfac 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--

setfac ...

- Create an access ACL with an entry for user bruce and group racf
 - **setfac -m user:bruce:rw,group:racf:r-x MyFile**
 - **getfac MyFile**

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

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

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 aclfile 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)`

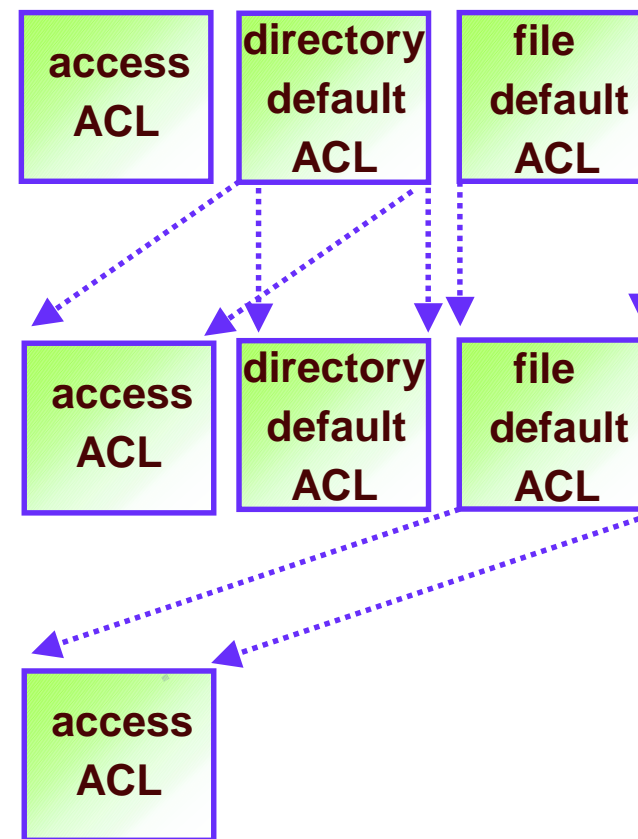
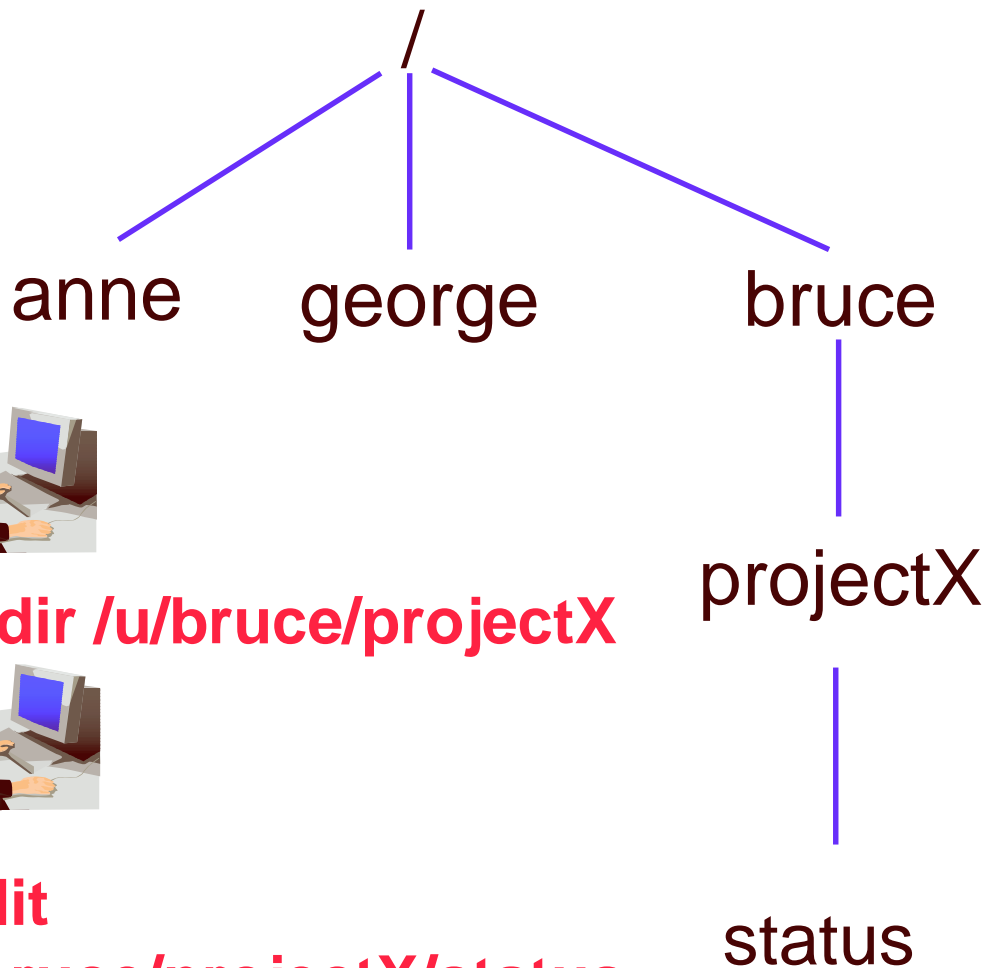
Overriding UNIXPRIV authority with ACL entries

- 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

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 ...



mkdir /u/bruce/projectX



**oedit
/u/bruce/projectX/status**

getfac and setfac ...

- Create a directory default ACL
 - setfac -m default:user:bruce:rwX MyDir
 - or: setfac -m **d**: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 ...

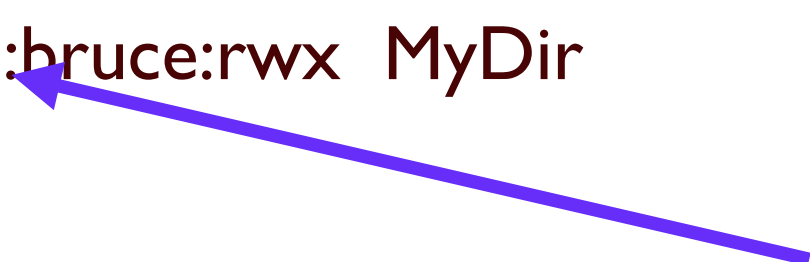
- Create a file default ACL
 - setfac -m fdefault:user:bruce:rwX MyDir
 - or: setfac -m **f**:u:bruce:rwX MyDir
 - getfac -f MyDir

#file: MyDir

#owner: BPXROOT

#group: SYS1

fdefault:user:BRUCE:rwX



additional
qualifier for
file default

getfac ...

- Display all ACLs for a directory

- `getfac -adf MyDir`

`#file: MyDir`

`#owner: BPXROOT`

`#group: SYS1`

`user::rwx`

`group::r-x`

`other::r-x`

`user:JOE:--x`

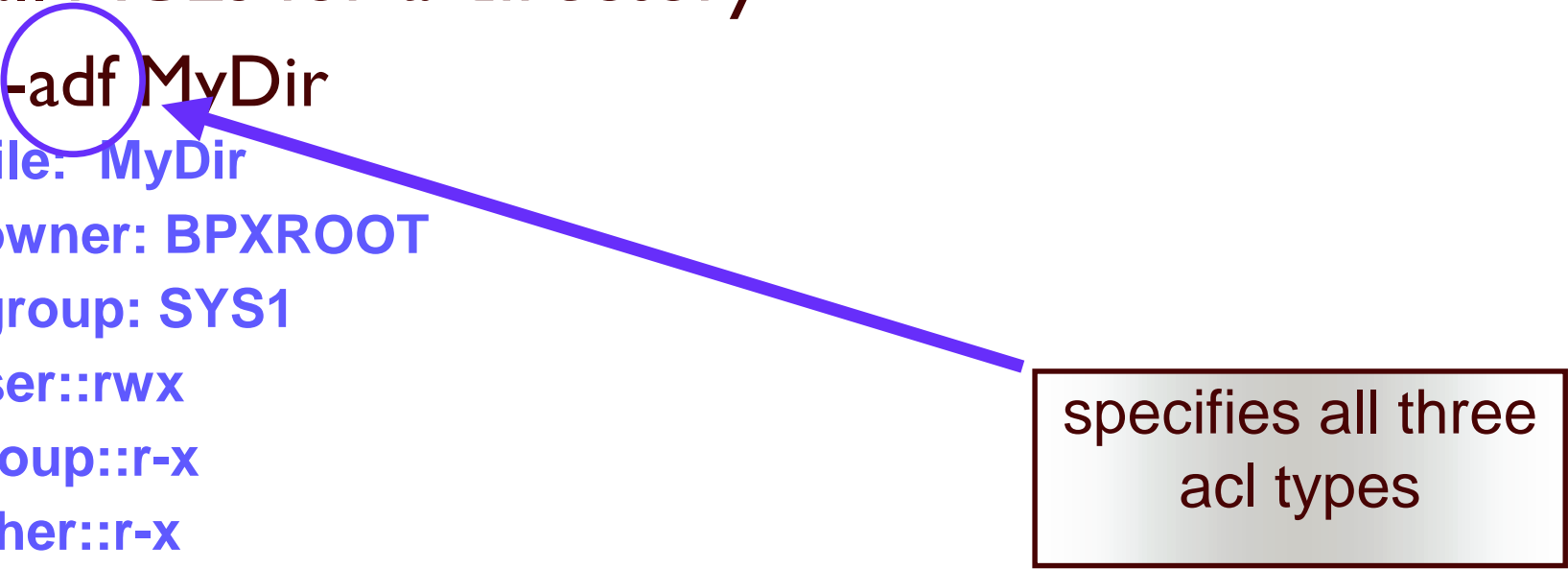
`user:BUCK:rwx`

`fdefault:user:ACE:r-x`

`fdefault:user:BUCK:rwx`

`default:user:DARTH:rwx`

`default:group:RACF:--x`



specifies all three
acl types

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
Redirec- tion ('>')	666
oedit	700
mkdir	777

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
 - `find / -attr p`

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)

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:

SETRPTS LOGOPTIONS(ALWAYS(FSSEC)) !!!

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)`

Output of ls (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---  --- ---  1 BPXROOT  SYS1     8240 Aug  20  16:39 racp
-rw-r--r--  -s- ---  1 1969     SYS1      99 Mar  20  16:46 woodstock

```



f = failures

s = successes

a = all (successes and failures)

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 90*n* 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`

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)

Supplemental Material

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)

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)

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

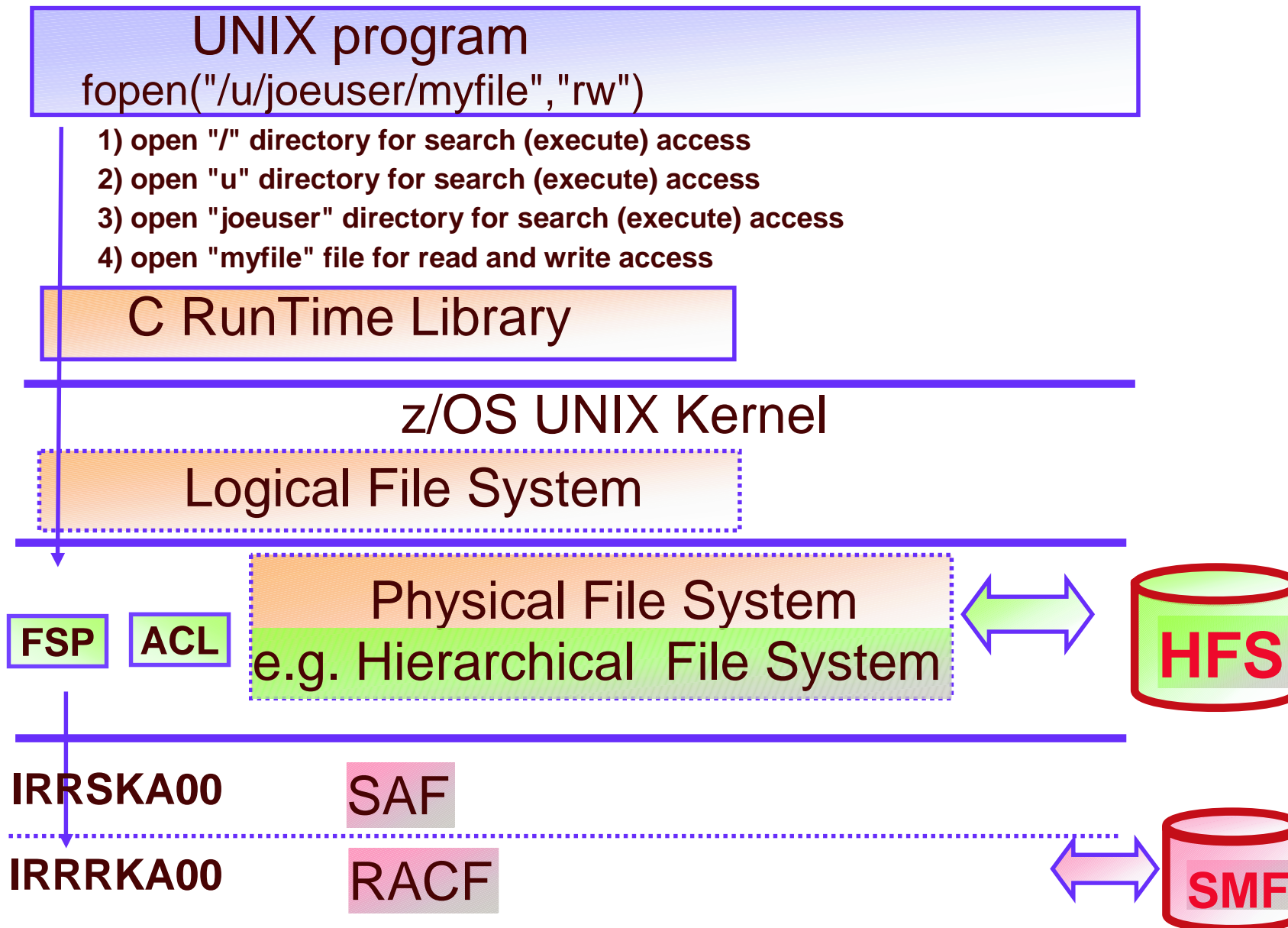
UNIX File Security Packet (FSP)

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
	owner r w x	group r w x	other 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
	read	write	execute	
no auditing	Auditor audit options			chaudit -a command
	read	write	execute	
SHAREAS bit on for executable files	Extended attributes			extattr command
contents of parent's default ACL	Access Control List			setfacl command

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	<ul style="list-style-type: none"> ■ 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

Access Checking Architecture



UNIX File Access Algorithm

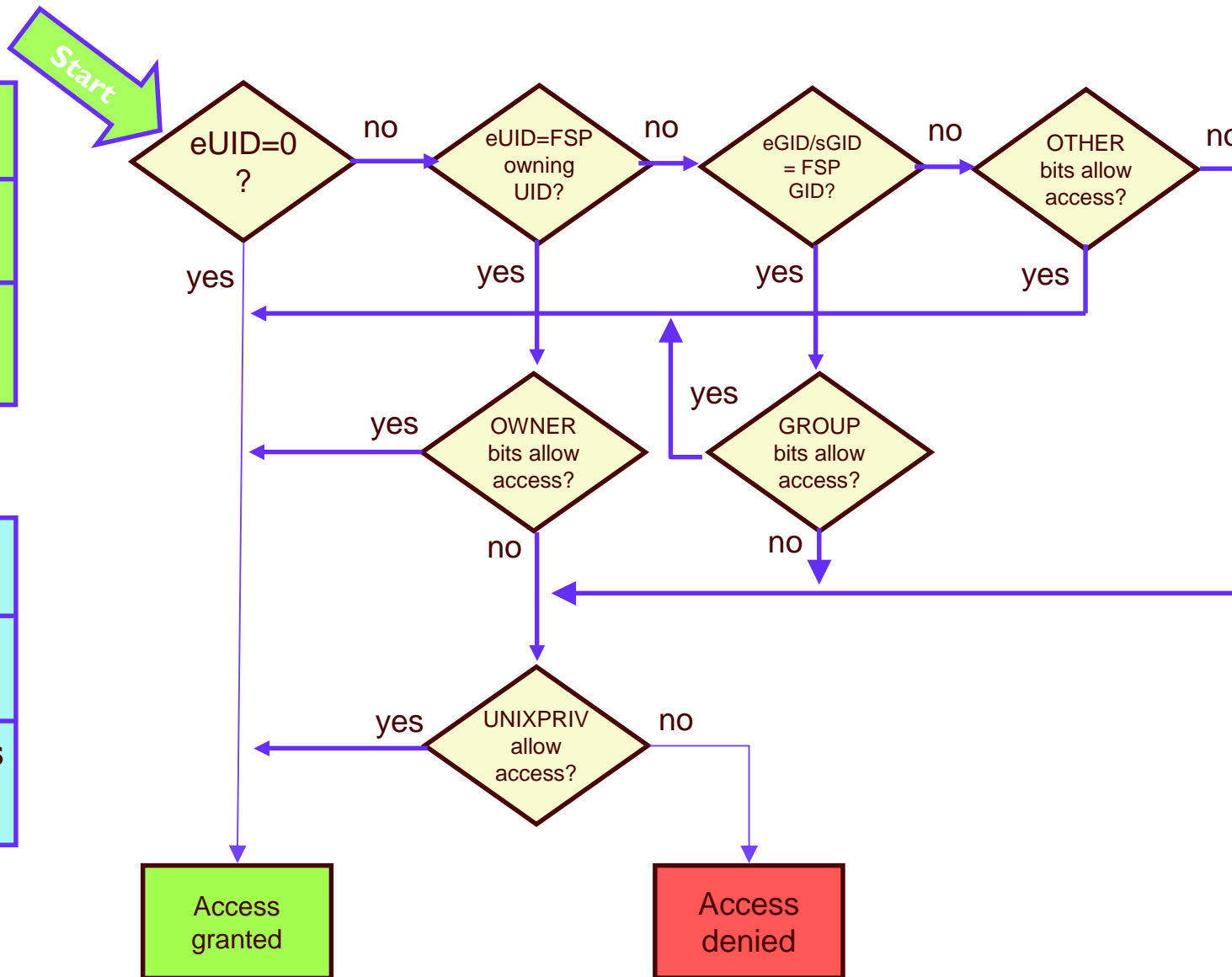
FSP

owning UID 50
owning GID 100
Permission bits rwx r-x --x

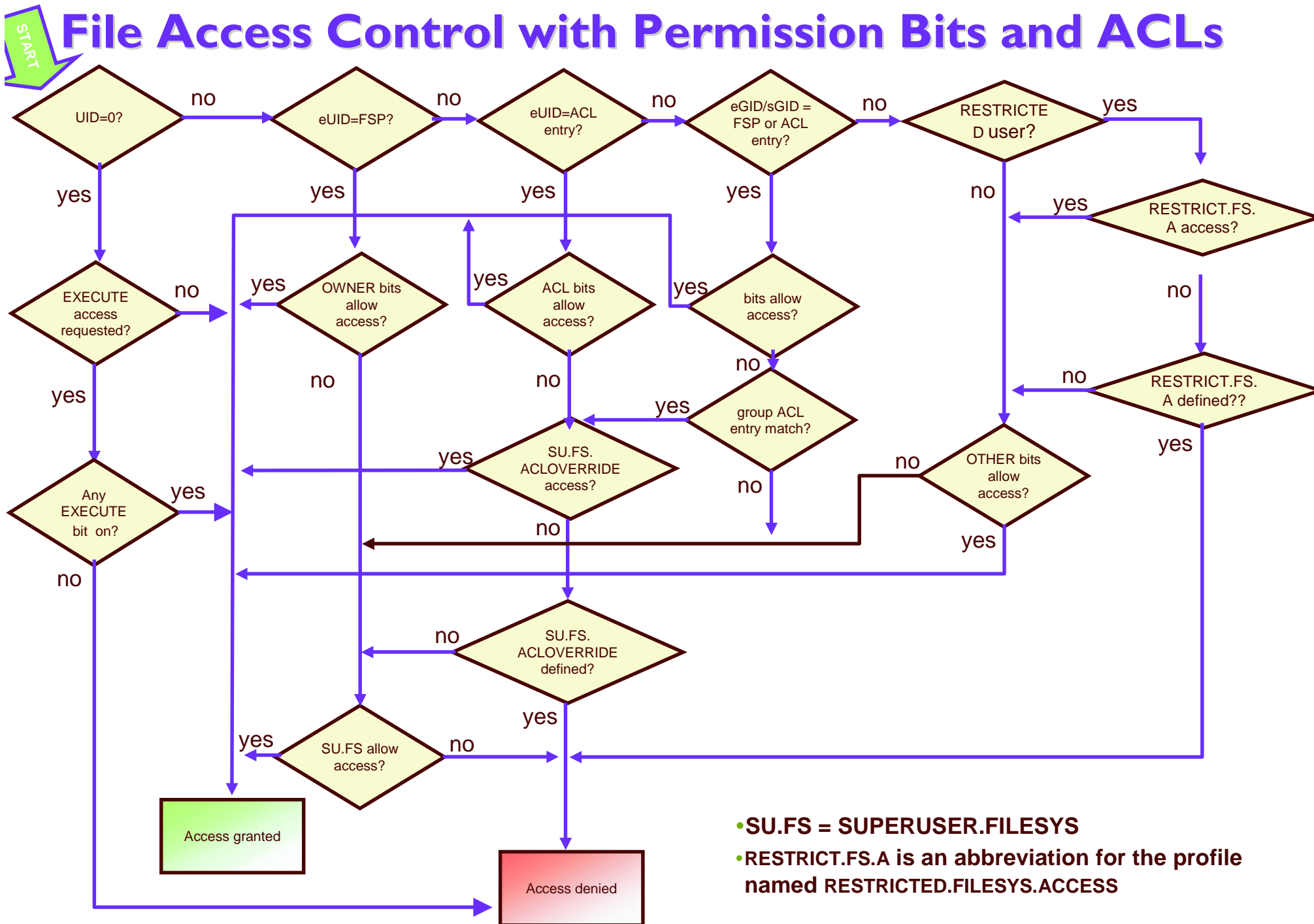
USP

effective UID 77
effective GID 999
Supplemental Groups 100 200 300

ACEE



File Access Control with Permission Bits and ACLs



- **SU.FS = SUPERUSER.FILESYS**
- **RESTRICT.FS.A is an abbreviation for the profile named RESTRICTED.FILESYS.ACCESS**

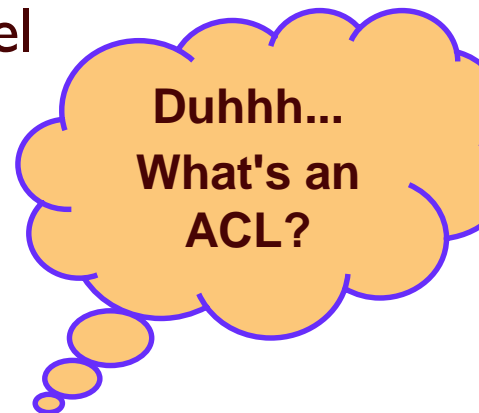
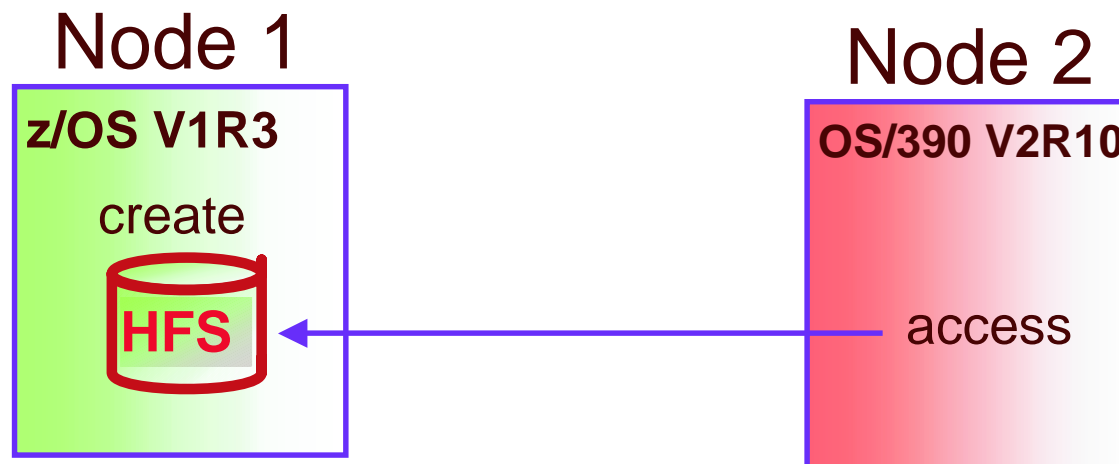
UNIX File Group Ownership

- 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



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



Auditing - Anatomy of a Violation

```
ICH408I 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)

Auditing UNIX Files: compared with data sets

DATASET Auditing	UNIX File Auditing
SETR_OPTS LOGOPTIONS for DATASET class controls access logging	SETR_OPTS LOGOPTIONS for FSOBJ, DIRACC, and DIRSRCH classes controls access logging
SETR_OPTS AUDIT(DATASET) audits profile creation/deletion	SETR_OPTS AUDIT(FSOBJ) audits file creation/deletion
SETR_OPTS AUDIT(DATASET) audits changes to RACF profiles	SETR_OPTS 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)

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
```