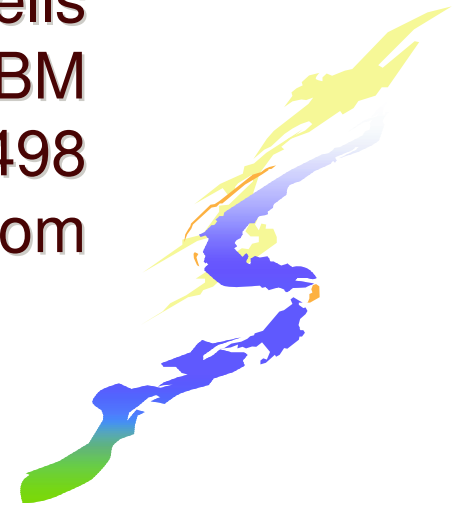
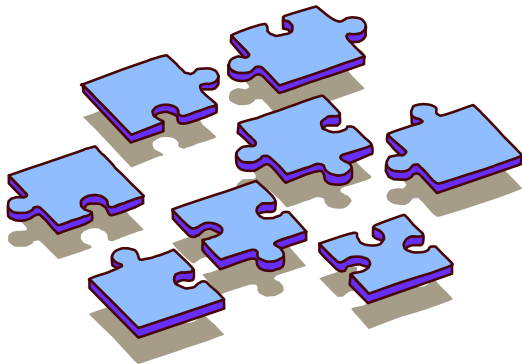




# RACF and z/OS UNIX File System Security

Vanguard Enterprise Security Expo 2005  
Session G6

Bruce R. Wells  
RACF Development, IBM  
845-435-7498  
[brwells@us.ibm.com](mailto:brwells@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:
  - RACF
  - z/OS
- 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
- Multilevel security
- Executable programs
- Auditing/Reporting



# 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 identity

## LOGON TSO

### ACEE

MVS user ID
USP Address

OMVS

## User Security Packet (USP)

• real	UID
• effective	
• saved	
• real	GID
• effective	
• saved	
Supplemental Groups	

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

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

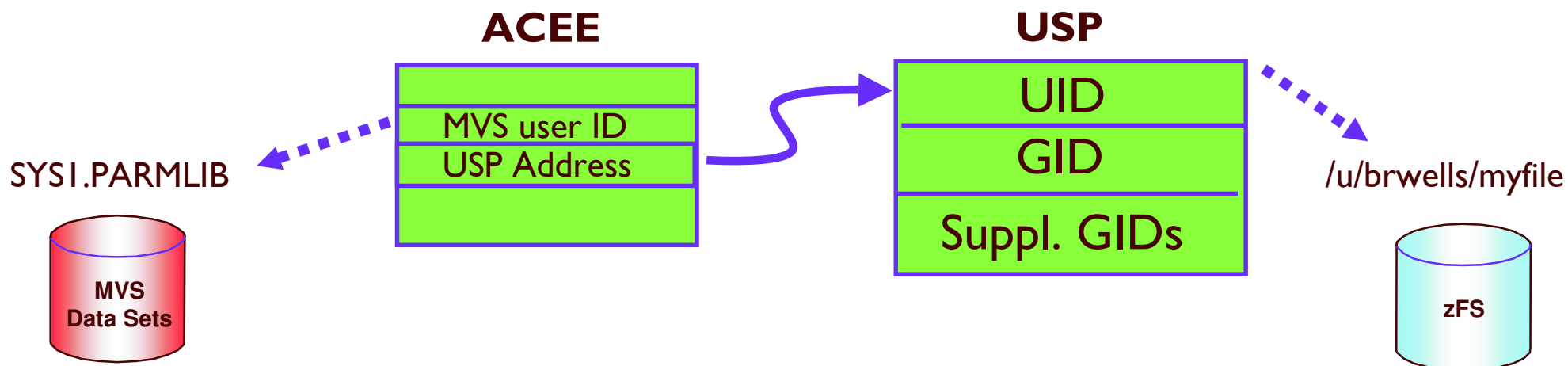
# id pierce

uid=34(PIERCE) gid=521(HOOPS) groups=4(KANSAS),16(CELTICS)



# UNIX identity

---



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

- 8-character MVS user ID (and group names) is checked against RACF profile

- When accessing UNIX files and directories:

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

# Types of file systems

---

- HFS - Hierarchical file system
- zFS - z-series file system
- TFS - temporary (or toy) file system
- DFS - distributed file system
- NFS - network file system

unique to z/OS





# File Systems are contained in MVS Data Sets

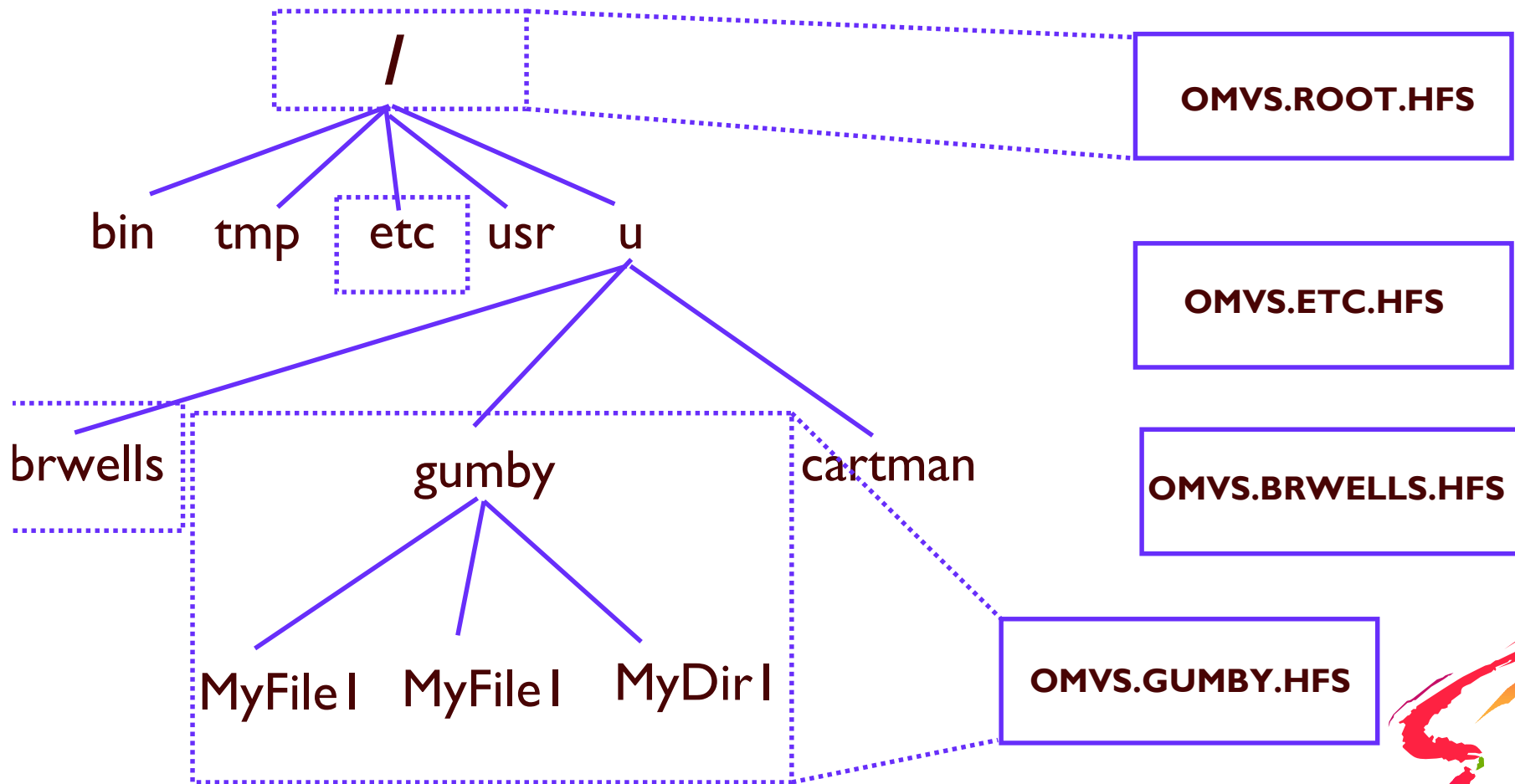
---

- Created using ALLOCATE (ISPF 3.2)
- Protected from MVS access methods by
  - Naming convention
  - 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



'SO MOUNT FILESYSTEM(OMVS.BRWELLS.HFS)

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



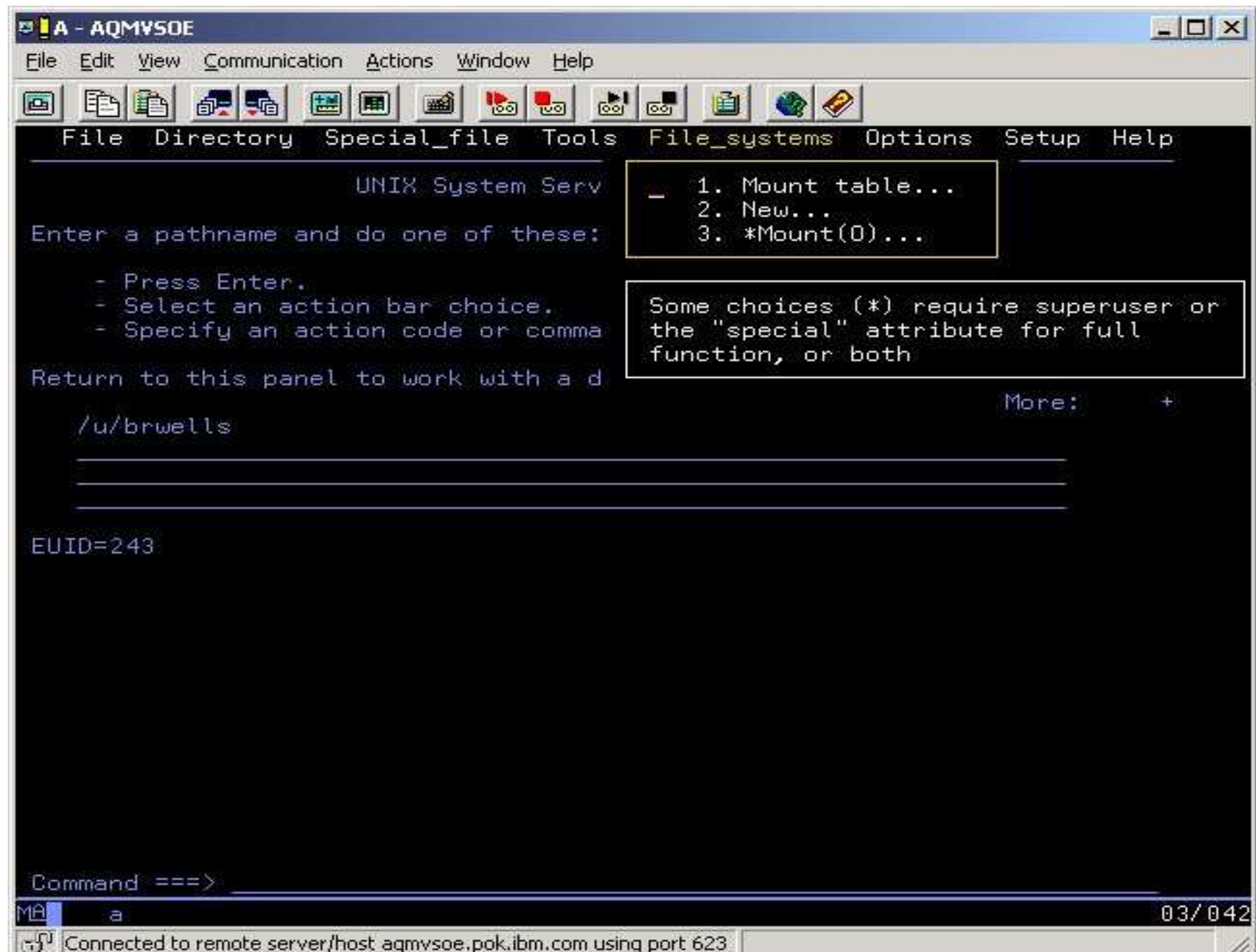
# MOUNT Options

---

- **MODE(RDRW | READ)**
  - Can mount a file system read-only to prevent it from being modified
- **NOSETUID**
  - Specifies that the set-uid, set-gid, APF, and program-control bits are not honored
- **NOSECURITY**
  - Specifies that no security checks are to be performed
  - Can still audit successful accesses
  - Superset of the NOSETUID option



# The ISPF Shell ... A Panel Interface



# 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 (APE program controlled, etc)



# UNIX File Security Packet (FSP)

## FSP contents

initialized to ...

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 rwx	Group rwx	Other rwx	
flags specified by open()	Flags			Chmod command
	set-uid	set-gid	sticky	
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
SECLABEL of covering dataset	Security label			chlabel command

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

---

<u>Security Field</u>	<u>Required authority</u>
Owning UID	<ul style="list-style-type: none"> <li>• UID 0</li> <li>• File owner if <code>CHOWN.UNRESTRICTED</code> is defined in the <b>UNIXPRIV</b> class</li> <li>• <b>READ</b> access to <b>UNIXPRIV</b> profile <code>SUPERUSER.FILESYS.CHOWN</code></li> </ul>
Owning GID	<ul style="list-style-type: none"> <li>• UID 0</li> <li>• Owner, if a member of new group</li> <li>• File owner if <code>CHOWN.UNRESTRICTED</code> is defined in the <b>UNIXPRIV</b> class</li> <li>• <b>READ</b> access to <b>UNIXPRIV</b> profile <code>SUPERUSER.FILESYS.CHOWN</code></li> </ul>
File mode (permissions and flags) and ACL	<ul style="list-style-type: none"> <li>• UID 0</li> <li>• File owner</li> <li>• <b>READ</b> access to <b>UNIXPRIV</b> profile <code>SUPERUSER.FILESYS.CHANGEPERMS</code></li> </ul>
Security Label	<ul style="list-style-type: none"> <li>• <b>RACF SPECIAL</b></li> </ul>
Owner audit options	<ul style="list-style-type: none"> <li>• UID 0</li> <li>• File owner</li> </ul>
Auditor audit options	<ul style="list-style-type: none"> <li>• <b>RACF AUDITOR</b></li> </ul>
Extended attributes	<p><b>READ</b> access to <b>FACILITY</b> class profile named:</p> <ul style="list-style-type: none"> <li>• <b>APF - BPX.FILEATTR.APF</b></li> <li>• <b>Program control - BPX.FILEATTR.PROGCTL</b></li> <li>• <b>shared library - BPX.FILEATTR.SHARELIB</b></li> </ul>

# 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	SYSI	30	Aug	23	2000	Program2
-r-srwxrwx	--s-	1	BPXROOT	KNIGHTS	8240	Aug	23	2000	SetuidPgm
drwxr-xr-x		2	BPXROOT	SYSI	8192	Mar	20	16:38	TestDirectory
-rwxr----t	--s-	1	ACE	JESTERS	8240	Aug	11	2000	progl
-rwxr-x--x	----	2	BPXROOT	SYSI	8240	Aug	11	2000	rac
lrwxrwxrwx		1	BPXROOT	SYSI	3	Aug	20	16:43	racSymlink -> rac
-rwxr-x--x	----	2	BPXROOT	SYSI	8240	Mar	11	2000	raclink
-rwxr-x---	aps-	1	BPXROOT	SYSI	8240	Aug	20	16:39	racp
-rw-r--r--	--s-	1	1969	SYSI	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



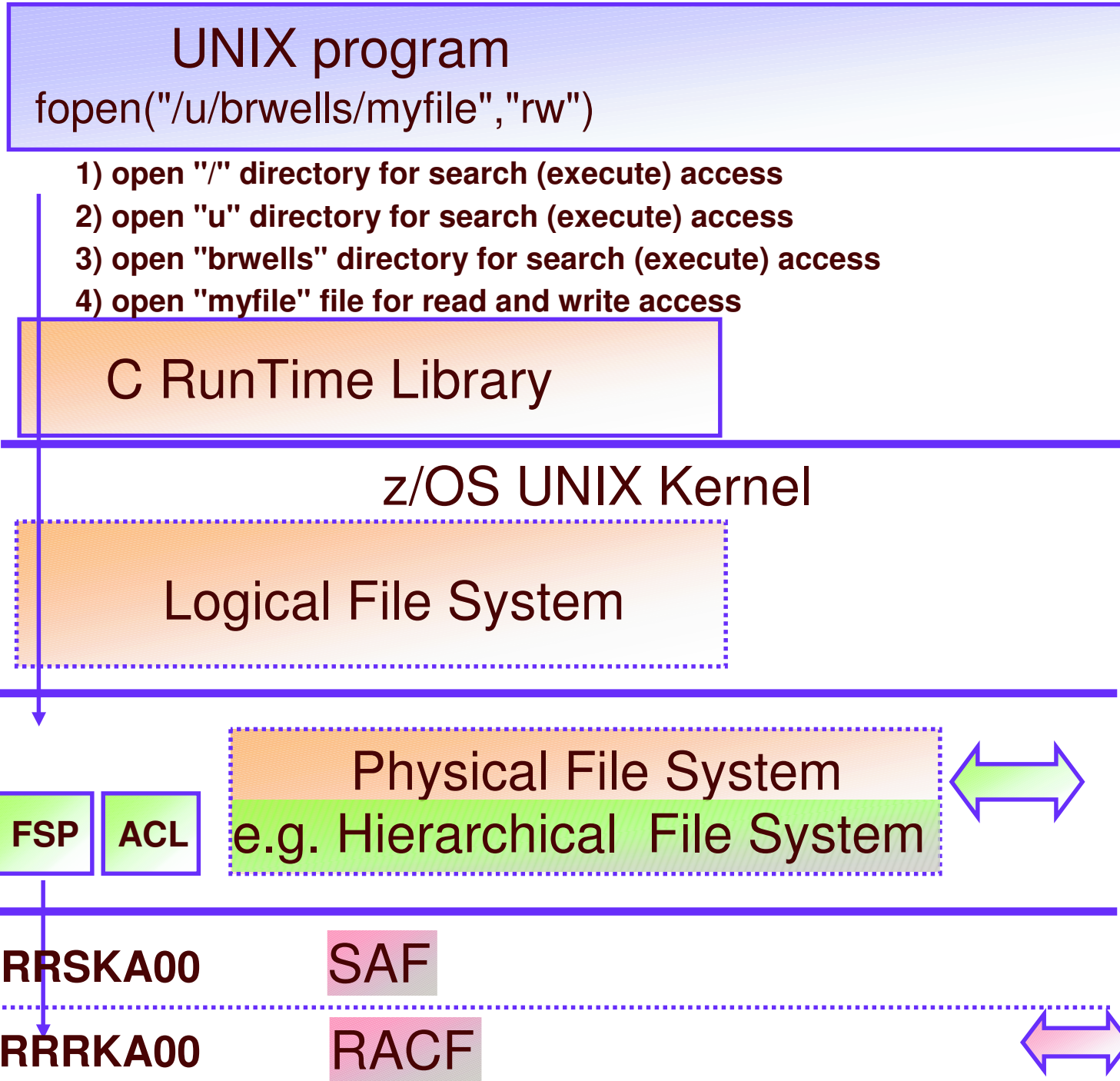
# chown Command - Change File Owner

---

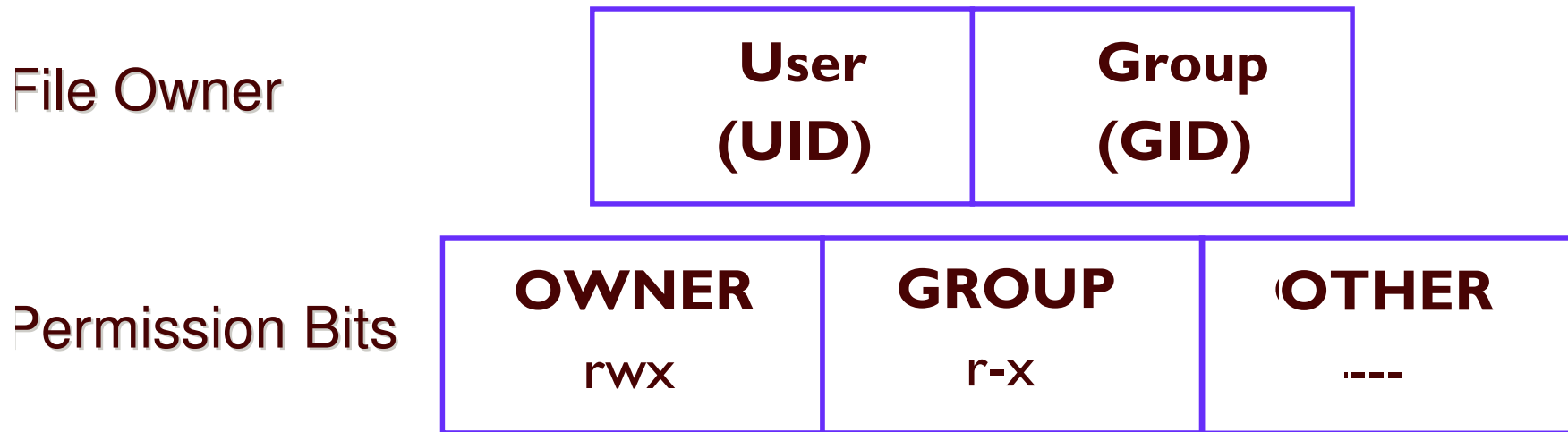
- Change owning user and group of a file
  - `chown flem:snopes /u/varner/store`
- Change owner of all files in a directory
  - `chown lou /prog/ibm/*`
- Change owner of all files in a directory, and its subdirectories
  - `chown -R uxadmin /u/deluser`
- Change owner of all of lou's files to sam  
`chown 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`



# Access Checking Architecture



# File Access Control with Permission Bits



`oedit /etc/profile`

User

effective UID
effective GID
Supplemental Groups

IF no access, check  
SUPERUSER.FILESYS  
in UNIXPRIV class

As per the UNIXPRIV profile  
RESTRICTED.FILESYS.ACCESS



[See z/OS Security Administrator's Guide Appendix F for detailed list of steps](#)

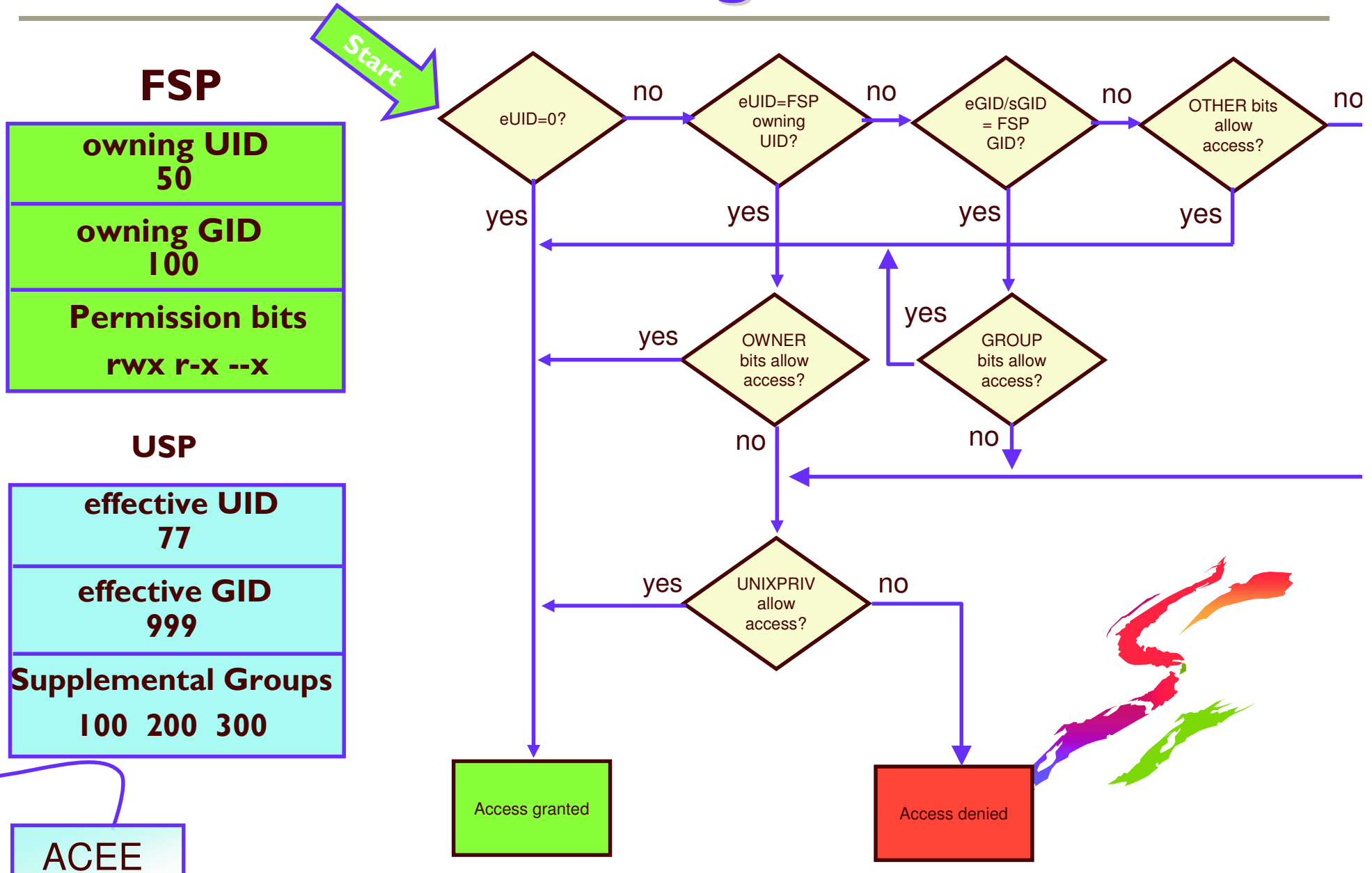
# Making the RESTRICTED attribute applicable to UNIX files

---

- 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



# UNIX File Access Algorithm



# Anatomy of a Violation

```
ICH408I USER(REDTAIL ) GROUP(RAPTORS ) NAME(PALE MALE)
/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)
```

- REDTAIL tried to OPEN this file, but was denied. Why?
- If the class were FSOBJ, we would know that REDTAIL 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 REDTAIL did not have search (execute) access to some directory component of the path name
- We must list 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 redtail /u/...` will narrow down the output to applicable entries (i.e. any user ACL entry for REDTAIL, and any group ACL entry for any of REDTAIL's groups)

# 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 all read bits on for all files in a directory and its subdirectories using relative perms
  - `chmod -R a+r MyDirectory`





# Access Control Lists (ACLs)

---

- Each entry specifies a user (UID) or group (GID) and its allowable permissions
- Can contain a maximum of 1024 entries
- Support inheritance
- Deleted automatically with file

Top Secret  
Superbowl Pool

User	Bob	r--
User	Boss	---
Group	Admins	rw-
Group	Execs	rwX
Group	Progs	rwX



# Access Control Lists (ACLs) ...

---

- Displayed with UNIX `getfacl` command
- Created, modified, and deleted with UNIX `setfacl` command
  - Must be file owner, UID(0), or have READ access to UNIXPRIV resource `SUPERUSER.FILESYS.CHANGEPERMS`
- Enabled with `SETROPTS CLASSACT(FSSEC)`
  - Can be created prior to activation
- Contained within the file system
  - Not in RACF profiles



# 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--



# setfacl ...

---

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



# 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/brwells/projectX -type d)`



# File Access Control with Permission Bits and ACLs

---

## Permission Bits

<b>OWNER</b>	<b>GROUP</b>	<b>OTHER</b>
rwX	rwX	rwX
<b>User1</b> rwX	<b>Group1</b> rwX	As per UNIXPRIV profile RESTRICTED.FILESYS.ACCESS  IF no access, check SUPERUSER.FILESYS or SUPERUSER.FILESYS.ACLOVERRID
<b>User2</b> rwX	<b>Group2</b> rwX	
<b>Usern</b> rwX	<b>Groupn</b> rwX	

C L  
o i  
n s  
t t  
r  
o  
l

IF FSSEC class active

[See z/OS RACF Security Administrator's Guide Appendix F for detailed list of steps](#)



# Overriding UNIXPRIV authority with ACL entries

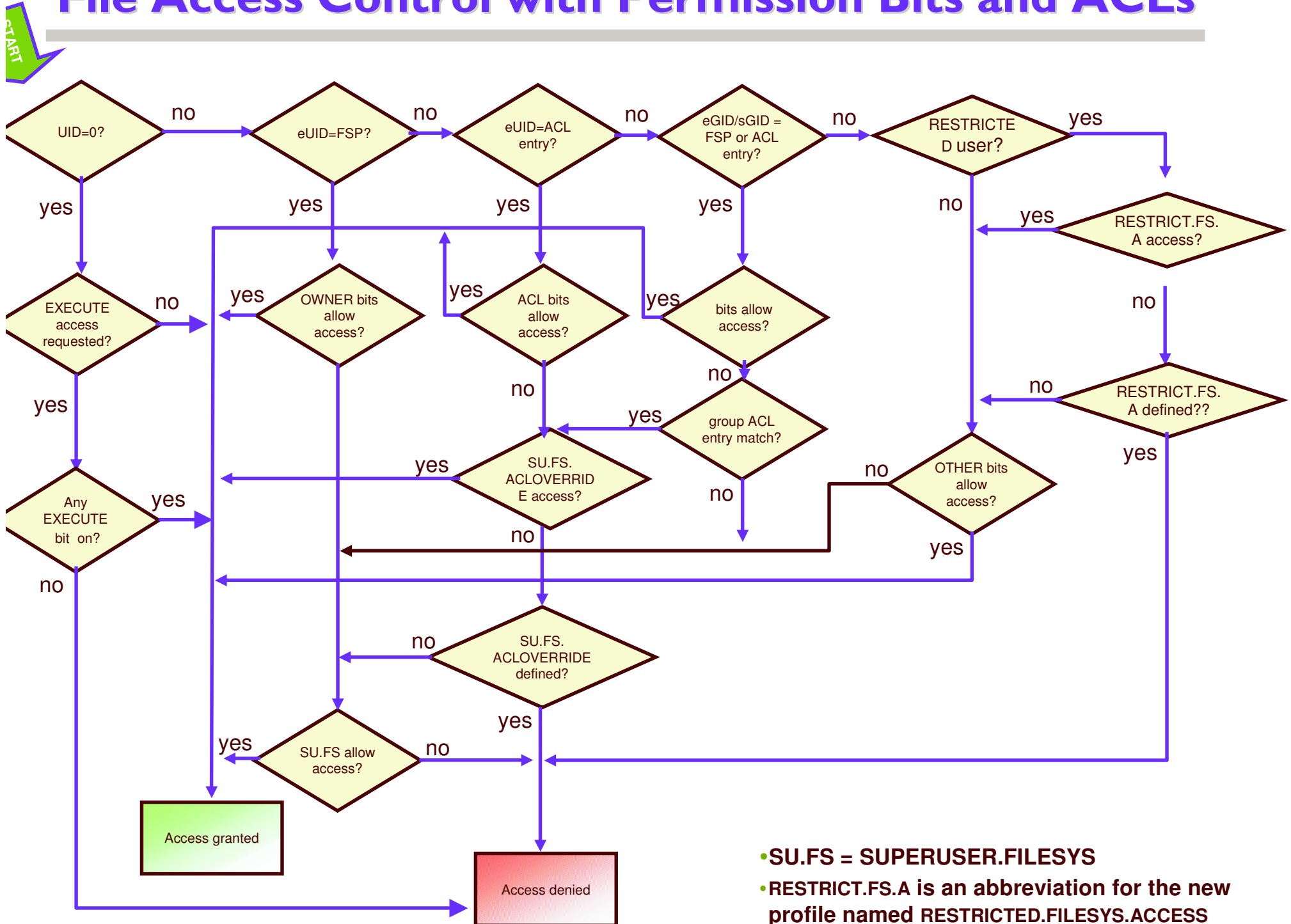
---

- 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





# File Access Control with Permission Bits and ACLs



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

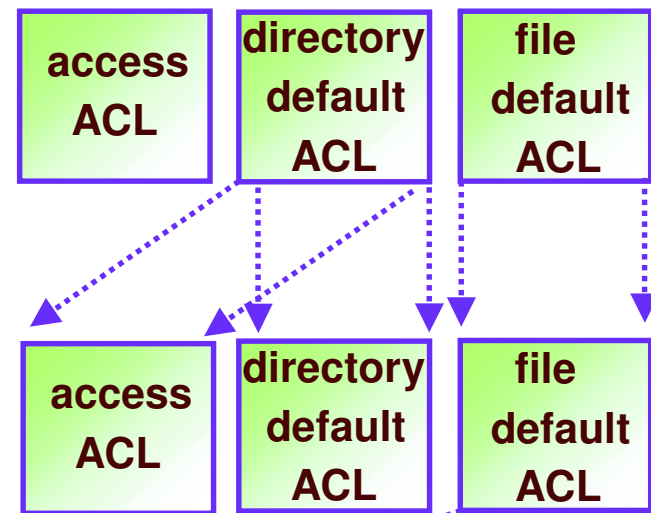
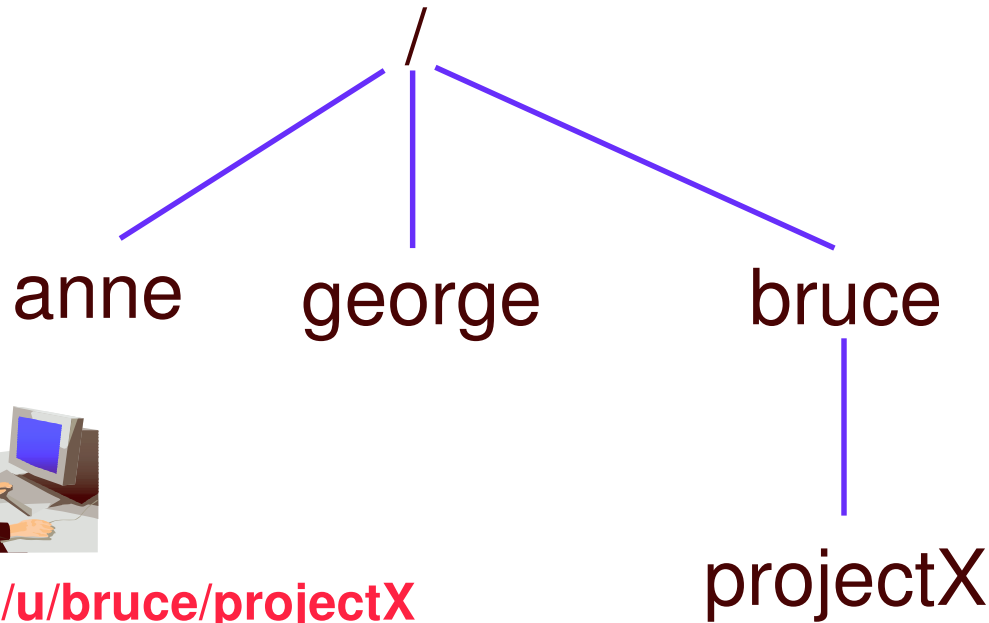
# ACL Inheritance

---

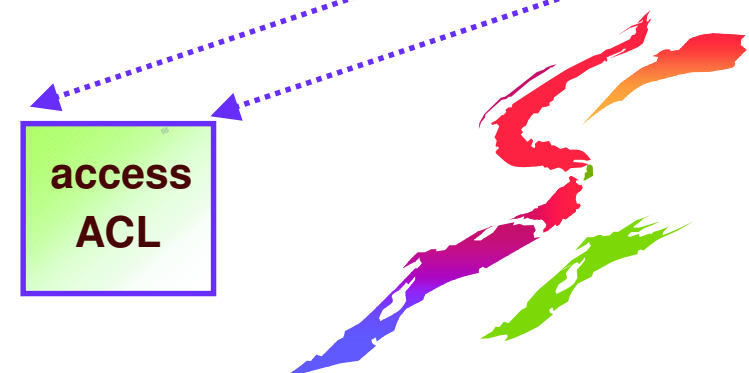
- Can establish default (or 'model') ACLs on a directory
- Get automatically applied to new files/directories created within the directory
- Separate default used for files and subdirectories
- Reduces administrative overhead



# ACL Inheritance ...



- status



**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

additional qualifier for directory default

```
#file: MyDir
#owner: BPXROOT
#group: SYS1
default:user:BRUCE:rwX
```



# 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

additional qualifier for  
file default

```
#file: MyDir  
#owner: BPXROOT  
#group: SYS1  
fdefault:user:BRUCE:rwx
```



# 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`

<u>Command</u>	<u>Permissio</u>
OPUT	600
touch	666
redirection ('>')	666
oedit	700
mkdir	777



# Multilevel Security

---

- Support is for zFS, starting on z/OS V1R5
- SECLABEL for new file system root copied from covering DATASET profile, or from creator
- SECLABEL propagated as new objects created
- SECLABEL cannot be changed once set
- For existing files, chlabel command can assign a SECLABEL - must be RACF SPECIAL
- SECLABEL existence enforced by SETROPTS MLFSOBJ
- SECLABEL checking enforced during file access





# 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 'RACF and z/OS UNIX Identities' (session G5)



# 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 RALT
  - RACF AUDITOR can read and search any directory
- RACF UAUDIT attribute honored
- Failing mounts/unmounts always audited
- Always:



# Auditing UNIX Files: compared with data sets

---

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

# Auditing UNIX Files: compared with data sets ...

---

<u>DATASET auditing</u>	<u>UNIX file auditing</u>
<b>LOGOPTIONS with ALWAYS and NEVER overrides profile settings</b>	same for file settings
<b>LOGPTIONS with SUCCESSES or FAILURES merged with profile-level settings</b>	same for file settings
<b>LOGOPTIONS with DEFAULT uses the profile-level settings</b>	same for file settings
<b>Default profile setting is READ failures for owner options, and no settings for auditor options (implies UPDATE, CONTROL, and ALTER failures too)</b>	<b>Default is read, write, and execute failures for owner settings (note that UNIX permissions are not hierarchical - these are separate settings for each access type)</b>
<b>Display profile options with LISTDSD</b>	<b>Display file options with ls -W</b>


# 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 -lW
total 192
-rw-r--r--    --- ---    1 BPXROOT    2001    ...    Odyssey
--wx--S---    --- ---    1 ACE        SYSI    ...    Program2
-r-srwxrwx    -aa ---    1 BPXROOT    KNIGHTS ...    SetuidPgm
drwxr-xr-x    fff ---    2 BPXROOT    SYSI    ...    TestDirectory
-rwxr----t    --- --a    1 ACE        JESTERS ...    prog1
-rwxr-x--x    --- ---    2 BPXROOT    SYSI    ...    rac
lrwxrwxrwx    fff ---    1 BPXROOT    SYSI    ...    racSymlink -> rac
-rwxr-x--x    --- ---    2 BPXROOT    SYSI    ...    raclink
-rwxr-x---    --- ---    1 BPXROOT    SYSI    ...    racp
-rw-r--r--    -s- ---    1 1969      SYSI    ...    woodstock
```

f = failures

s = successes

a = all (successes and failures)



auditor audit settings

# File System Security Reporting - HFS Unload!!!

---

- irrhfsu command available on <http://www-1.ibm.com/servers/eserver/zseries/zos/racf/goodies.html>
- Reports on HFS security data like IRRDBU00 reports c 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 //BRWELLS.HFSU.OUTPUT /u/brwells/dir1 dir2/subdir





# HFS Unload (continued)

---

- UIDs mapped to user IDs and GIDs mapped to group names
  - Implement the UNIXMAP class or AIM\*, or modify the source code!!!

0900	file name	i-node	uid	user id	gid	group name	set uid	set gid	sticky bit	owner read	owner write	owner execute	group read	etc ...
------	-----------	--------	-----	---------	-----	------------	---------	---------	------------	------------	-------------	---------------	------------	---------

Get it at: <http://www-1.ibm.com/servers/eserver/zseries/zos/racf/goodies.h>

\* AIM - Application Identity Mapping. Activated using the IRRIRA00 utility.



# HFS Unload (continued)

---

- Integrate it with current IRRDBU00 procedure

```
//BRWELLSL JOB '577018,B0011038','B.R.WELLS',  
// CLASS=2,NOTIFY=BRWELLS,MSGLEVEL=(1,1),  
// MSGCLASS=H  
//*****  
//HFSUNLD EXEC PGM=BPXBATCH,  
// PARM='PGM irrhfsu -f //SYS1.IRRDBU00.OUTPUT /'  
//STDERR DD PATH='/u/brwells/hfsuerr',  
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),  
// PATHMODE=SIRWXU
```



# Good Sources of Information

---

- UNIX System Services web site, at <http://www-1.ibm.com/servers/eserver/zseries/zos/unix/>  
Check out the Tools and Toys page
- UNIX System Services Planning manual SC28-1890 (for your release)
- UNIX System Services Command Reference
  - Available online at <http://www-1.ibm.com/servers/s390/os390/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-1.ibm.com/servers/eserver/zseries/zos/racf/>  
See Downloads page for HFS Unload
- RACF Security Administrator's Guide (UNIX chapter)
- RACF Auditor's Guide
  - Available online at <http://www-1.ibm.com/servers/s390/os390/bkserv/>
- racf-l mailing list (see the front-matter in any RACF book for information)



# 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
- File mgmt and access are controlled by the kernel and file system through calls to RACF via SAF services
- 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)

