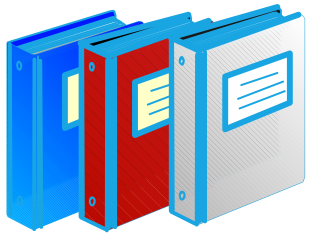
RACF & OS/390 UNIX SYSTEM SERVICES SECURITY OVERVIEW

Vanguard Enterprise Security Expo 2001
Session 110



Bruce R. Wells
RACF Development, IBM
845-435-7498
brwells@us.ibm.com

Disclaimer

Trademarks

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

Agenda

- z/OS UNIX Overview
- UNIX Identity Management
  - Users and UIDs, Groups and GIDs
  - Default OMVS user identity
  - Superusers
  - Changing identity
  - Auditing
- UNIX File System Security
  - HFS security data
  - File permissions
  - Auditing/Reporting
- ISHELL

What is OS/390 UNIX System Services?

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

What is it for?

- Makes application development easier
  - Standard (open) programming interface
  - Interoperability in networks
  - Portable programs
  - Portable data
- Required by some products

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

UNIX Identity Management

UNIX User definition

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

User Definition ...

```
ADDGROUP UNIXGRP OMVS(GID(100))
ALTUSER ADMIN OMVS(UID(1) HOME(/u/admin)
   PROGRAM(/bin/sh))
```

!!!! Note the mixed case !!!!

```
CONNECT ADMIN GROUP(UNIXGRP)
ADDUSER JOHN PASSW(xxxx) DFLTGRP(UNIXGRP)
   OMVS(UID(2) HOME(/u/john) PROGRAM(/bin/sh))
   TSO(ACCTNUM(12345) PROC(PROC01))
LISTUSER  JOHN OMVS NORACF
```

```
        USER=JOHN
        OMVS INFORMATION
        ----------------------------------
        UID = 0000000002
        HOME = /u/john
        PROGRAM = /bin/sh
```

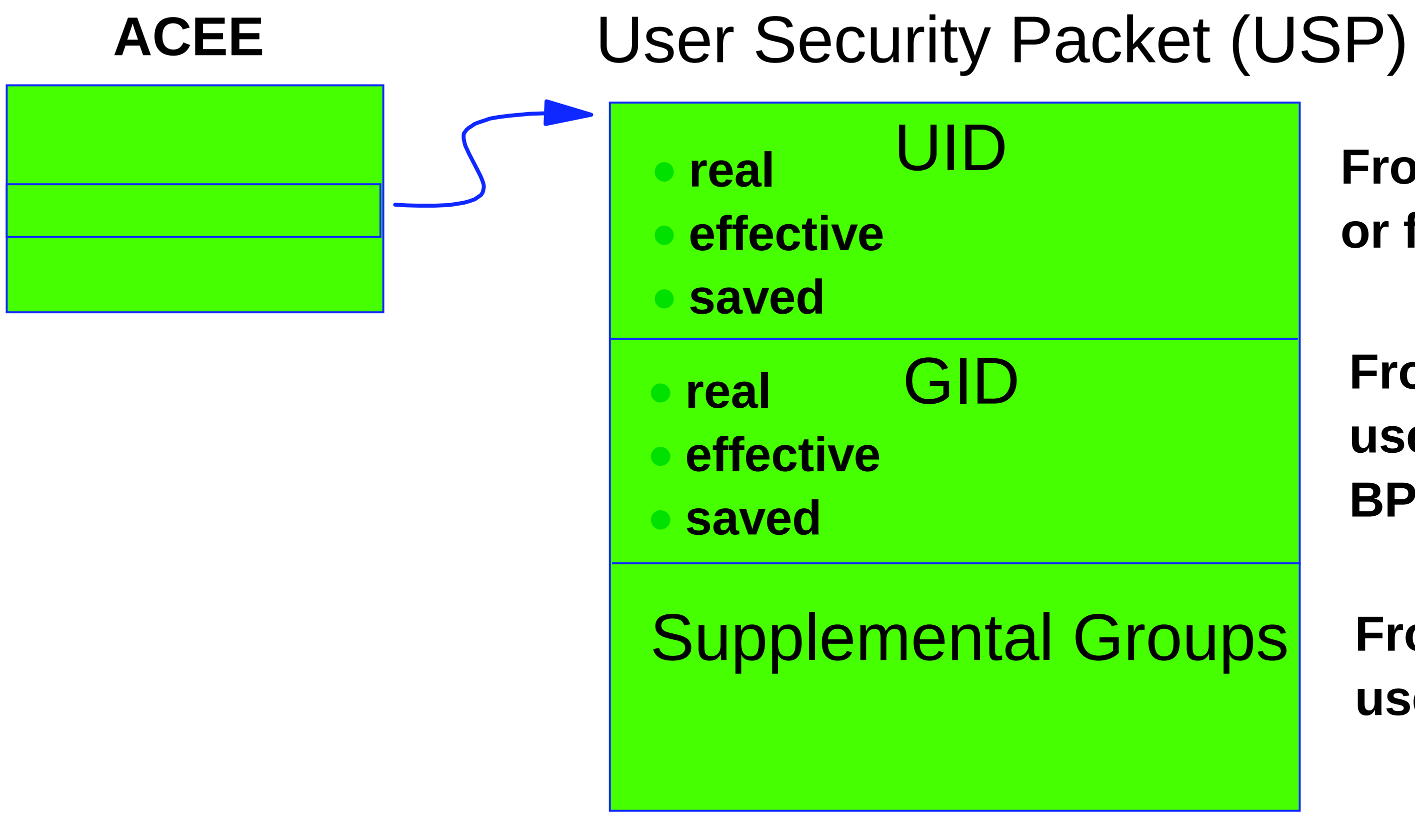
Default UNIX User and Group identity

- 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(... ... ...)**
  - **ADDGROUP DFTGROUP OMVS(GID(nnn))**
- Assigned during 'dub' when user/group doesn't have (complete) OMVS segment
- Can be overridden on a per-user basis
  - **ALTUSER BOB OMVS(NOUID)**
- Available on OS/390 V2R6 and up, or V2R4 + APAR OW26800
- Use of default identity is audited

User Definition ...

- RACF command and ISPF panels, or ISHELL can be used
  - **can use rac from RACF downloads page to issue RACF commands from the UNIX shell**
- Ensure uniqueness of UIDs and GIDs
  - **Use a value that's already unique (Serial Number)**
  - **Use the ISHELL**
  - **Use sample DBunload reports**
- Delegate with the FIELD class
  - **Allow an OS/390 UNIX administrator to assign UIDs and GIDs**
  - **Allow users to list their own info and change some of it (e.g. initial program)**
  - **See appendix for examples**

UNIX identity



From user's OMVS segme
or from BPX.DEFAULT.USE

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

From OMVS segments of
user's list of groups

- USP created when first UNIX service is invoked
- Effective UID/GID and supplemental groups are used to determine UNIX file authority (more on this later)
- user ID in ACEE is used to determine "MVS" authority
- use the id command to show user's UNIX identity

Ways of assuming another UNIX identity

- Various C language functions such as setuid(), setgid(), pthread_security_np()
  - Used by UNIX servers and daemons
- Executing a set-id file
  - changes effective UID/GID to that of file owner
- Issuing the su command
  - must have access to BPX.SUPERUSER in the FACILITY class to switch to superuser
  - must know the user's password, or have access to BPX.SRV.*userid* in the SURROGAT class
- Trojan Horse!
  - put an ls command in my home directory and do some social engineering

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
  - Change his identity to another UID
  - Use setrlimit to increase system limits
- Not used when accessing MVS resources
- No special meaning for GID 0

SUPERUSER Granularity: UNIXPRIV

- New with OS/390 V2R8: 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 HFS file
  - ability to change file ownership
  - ability to send signals to any process
  - ability to mount/unmount file systems

UNIXPRIV Resource Names

HFS File and Directory Access

Resource Name	Privilege	Access Req'd
SUPERUSER.FILESYS	**read any HFS file; read/search any HFS directory**	**READ**
SUPERUSER.FILESYS	**write any HFS file; also privileges of READ access**	**UPDATE**
SUPERUSER.FILESYS	**write any HFS directory; also privileges of UPDATE access**	**CONTROL**

See appendix for additional UNIXPRIV resources

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
- 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 (OS/390 V2R4 and higher with APAR OW42092, or OS/390 V2R10)**

UNIX Auditing ... The results

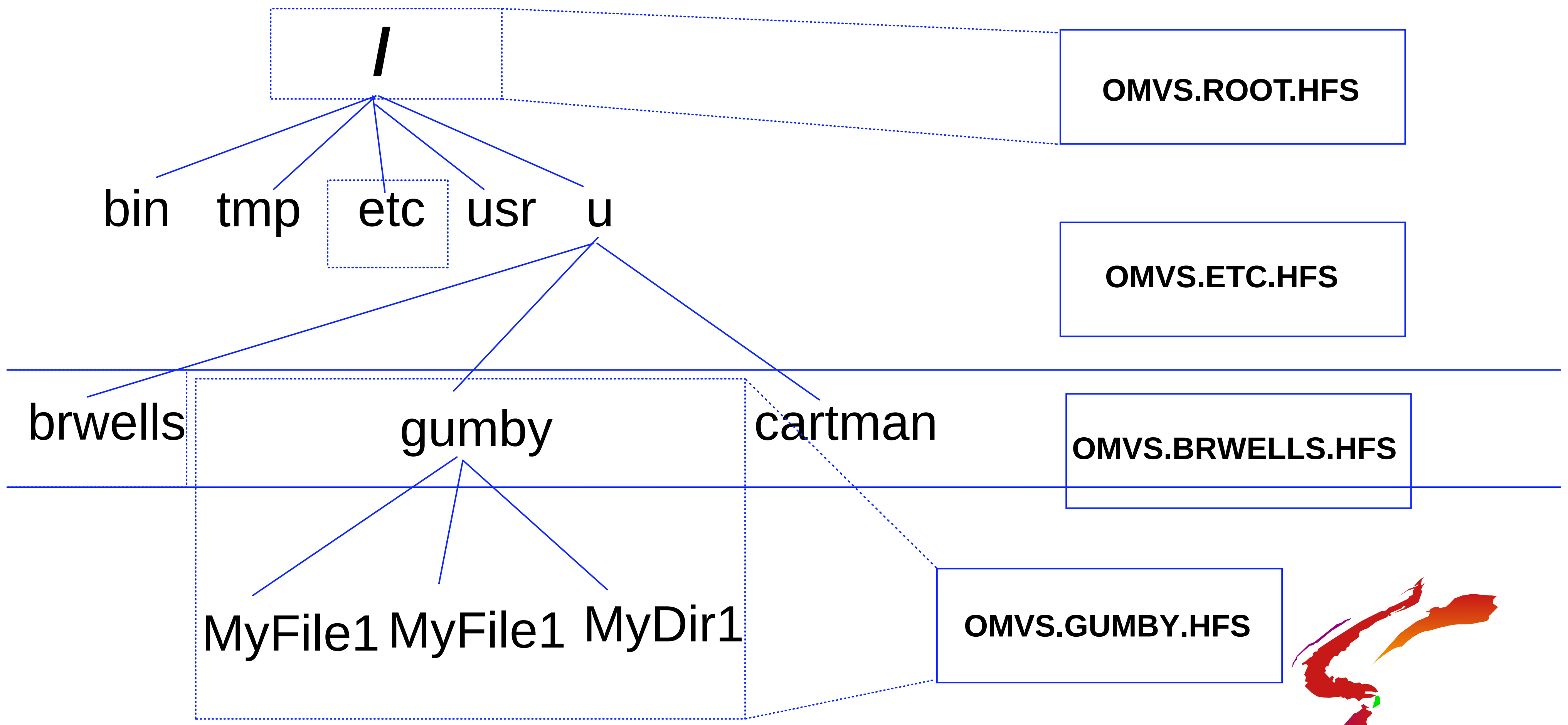
- Type 80 SMF records
- ICH408Is for resources and services

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

- Settings can cause excessive records SETR LOGOPTIONS(ALWAYS(...))
- No write-to-programmers are issued
- RACFRW information is incomplete
- Use SMF Data Unload utility (IRRADU00)

UNIX File System Security

Hierarchical File System (HFS) is a Collection of MVS Data Sets



TSO MOUNT FILESYSTEM(OMVS.BRWELLS.HFS)
MOUNTPOINT('/u/brwells') TYPE(HFS)

UNIX File Security

- UNIX invokes RACF through SAF callable services
- Access checking is automatic
- No profiles in RACF database
- Access control by permission bits
  - read, write, execute (non-hierarchical)
  - one set each for owner, group, other
- File Security Packets stored with file contain file security attributes
  - owning UID and GID
  - permission bits and flags
  - audit settings

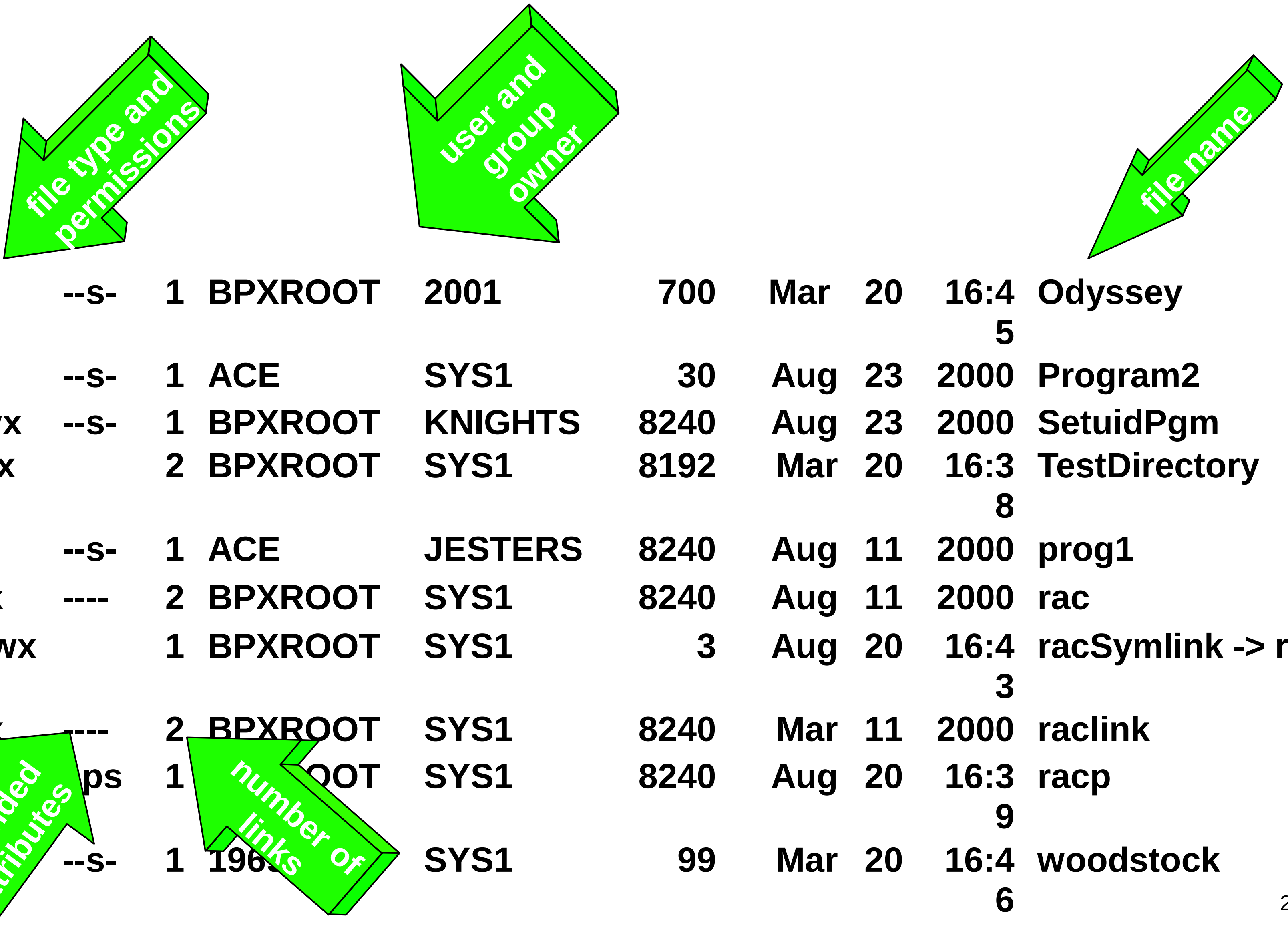
UNIX File Security Packet (FSP)

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 owner r w x · group r w x · other r w x	chmod command
set-id bits off, sticky	Flags Directory, set-uid, set-gid, sticky bit	chmod command
bit specified by fn read, write, and execute failures	Owner audit options read · write · execute	chaudit command
no auditing	Auditor audit options read · write · execute	chaudit -a comman
SHAREAS bit on for executable files	Extended attributes	extattr command

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

Security Field	Required authority
Owning UID	● UID 0 ● File owner if CHOWN.UNRESTRICTED is defined in the UNIXPRIV class ● READ access to UNIXPRIV profile SUPERUSER.FILESYS.CHOWN
Owning GID	● UID 0 ● Owner, if a member of new group ● File owner if CHOWN.UNRESTRICTED is defined in the UNIXPRIV class ● READ access to UNIXPRIV profile SUPERUSER.FILESYS.CHOWN
File mode (permisions and flags)	● UID 0 ● File owner
Owner audit options	● UID 0 ● File owner
Auditor audit options	● RACF AUDITOR
Extended attributes	READ access to FACILITY class profile named: ● APF - BPX.FILEATTR.APF ● Program control - BPX.FILEATTR.PROGCTL ● shared library - BPX.FILEATTR.SHARELIB

Output of ls (list files) Command



```
# ls -E
total 192
-rw-r--r--     --s-   1  BPXROOT    2001         700    Mar   20   16:4  Odyssey
                                                                     5
--wx--S---     --s-   1  ACE        SYS1          30    Aug   23   2000  Program2
-r-srwxrwx     --s-   1  BPXROOT    KNIGHTS     8240    Aug   23   2000  SetuidPgm
drwxr-xr-x            2  BPXROOT    SYS1        8192     Mar  20   16:3  TestDirectory
                                                                     8
-rwxr----t     --s-   1  ACE        JESTERS     8240    Aug   11   2000  prog1
-rwxr-x--x     ----   2  BPXROOT    SYS1        8240    Aug   11   2000  rac
lrwxrwxrwx            1  BPXROOT    SYS1           3    Aug   20   16:4  racSymlink -> ra
                                                                     3
-rwxr-x--x     ----   2  BPXROOT    SYS1        8240     Mar  11   2000  raclink
-rwxr-x-[illegible]  [illegible]ps   1  [illegible]OOT   SYS1  8240    Aug   20   16:3  racp
                                                                     9
-rw-r[illegible]   --s-   1  196[illegible]   SYS1      99     Mar  20   16:4  woodstock
                                                                     6
```

chown Command - Change File Owner

- Change owning user and group of a file
  - chown flem:snopes FaulknerFile
- 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 bill's files to george
  - find /u -user bill -exec chown george {} \;
- 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 all read bits on for all files in a directory and its subdirectories using relative perms
  - chmod -R a+r MyDirectory
- Turn on the set-uid bit for a program
  - chmod u+s MyProgram
- Turn on the sticky bit for a program
  - chmod +t MyProgram

Other File-related Commands

- Display security information (including extended attributes) for files within the current directory
  - ls -E
- Display umask in symbolic form
  - umask -S
- Set umask so group and other write bits cannot be set during file creation
  - umask g-w,o-w
- Turn on APF and program control bits for a program
  - extattr +ap MyProgram

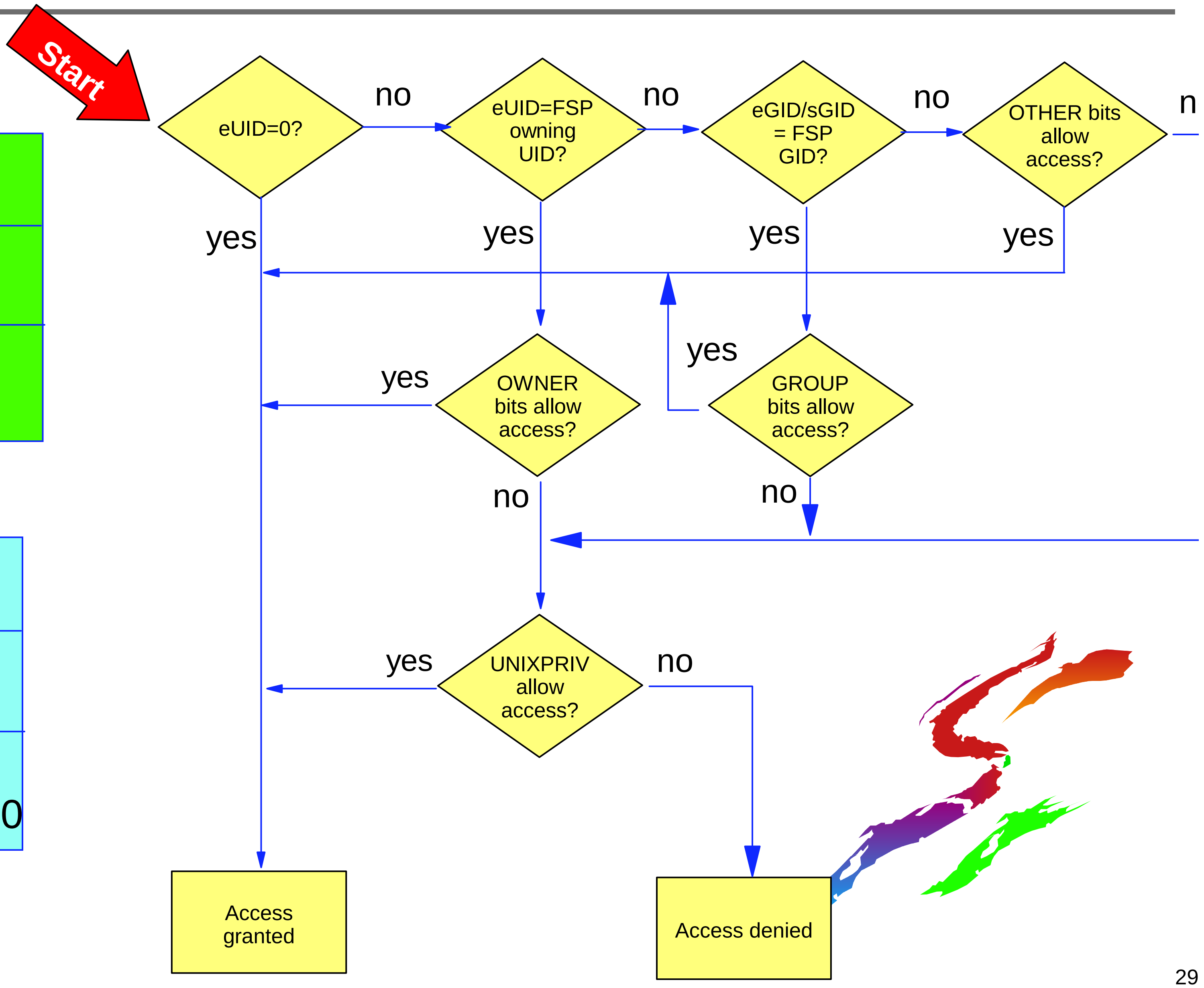
UNIX File Access Algorithm

FSP

owning UID 50
owning GID 100
Permission bits

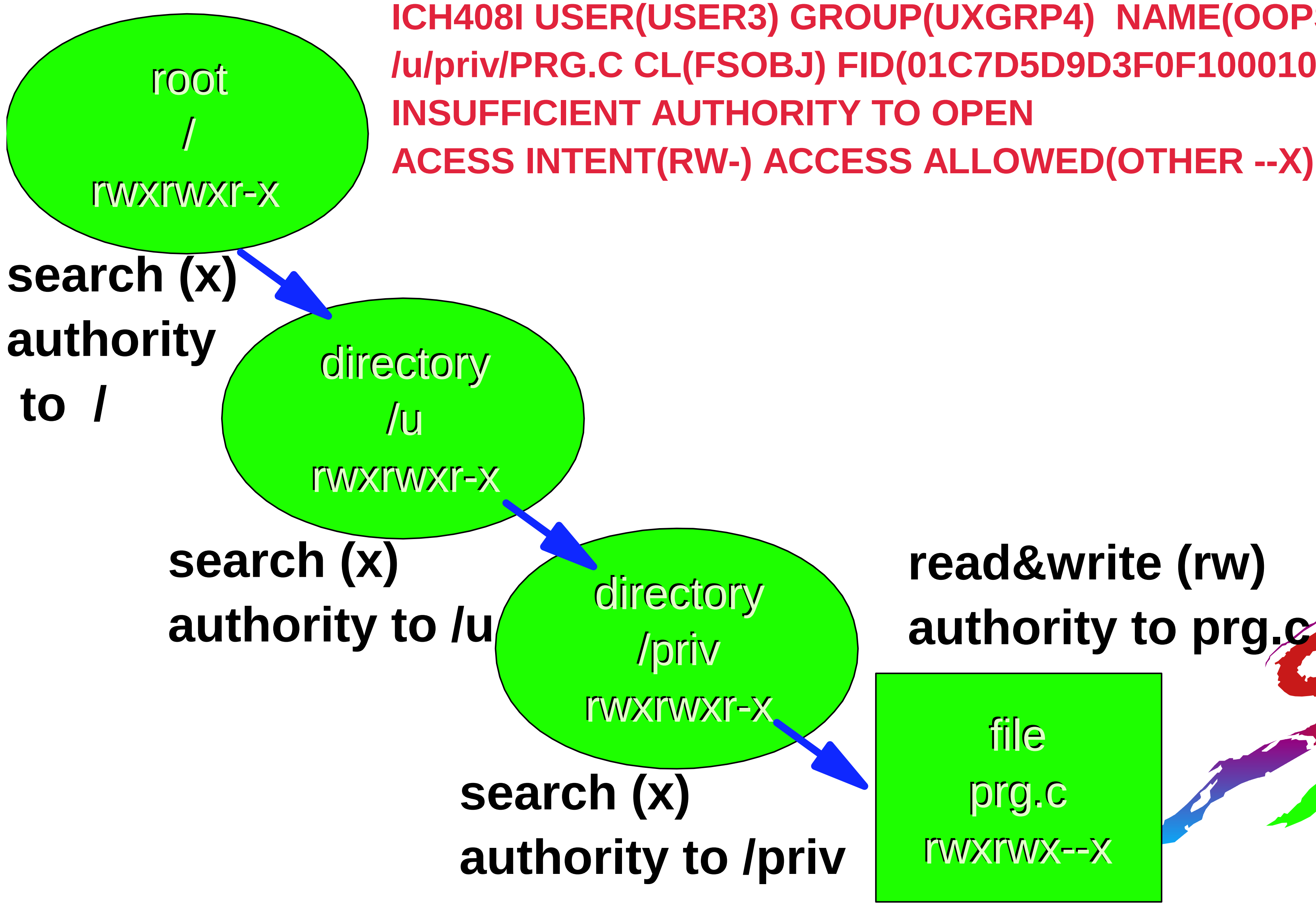
USP

effective UID 77
effective GID 999
Supplemental Groups 100 200 300



Protecting Files ...

```
ICH408I USER(USER3) GROUP(UXGRP4)  NAME(OOPS)
/u/priv/PRG.C CL(FSOBJ) FID(01C7D5D9D3F0F100010400007000)
INSUFFICIENT AUTHORITY TO OPEN
ACESS INTENT(RW-) ACCESS ALLOWED(OTHER --X)
```



Protecting Files ... ICH408I Violation

```
ICH408I USER(USER3) GROUP(UXGRP4)  NAME(OOPS)
/u/priv/PRG.C CL(FSOBJ) FID(01C7D5D9D3F0F1000010400007000)
INSUFFICIENT AUTHORITY TO OPEN
ACCESS INTENT(RW-) ACCESS ALLOWED(OTHER --X)
```

- USER3 tried to open this file for READ and WRITE acces
- The owner of this file wasn't USER3
- UXGRP4 wasn't the owning group for this file
- USER3 didn't belong to the group that owns this file
- The OTHER permissions only allow execute access
- Auditing of failures was set for this file,
  or with SETROPTS for class FSOBJ

UNIX File Auditing

- Controlled by audit classes
  - SETR LOGOPTIONS, SETR AUDIT
    - DIRSRCH,DIRACC,FSOBJ,FSSEC
  - CLASSACT/NOCLASSACT has no effect
- And by file-level audit options
  - Similar to AUDIT() and GLOBALAUDIT()
  - Set with chaudit, not ALTDSD or RALT
- RACF UAUDIT attribute honored
- Failing mounts/unmounts always audited

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 contols 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 option of ALTDSD)	File-level auditing can be specified by file owner (chaudit command)
Profile-level auditing can be specified by auditor (GLOBALAUDIT option 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
LOGPTIONS 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, and no settings for auditor options (implies UPDATE, CONTROL, and ALTER failures too)	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)
Display profile options with LISTDSD	Display file options with ls -W

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

owner audit settings

```
# ls -W
total 192
-rw-r--r--   --- ---   1  BPXROOT   2001      ...   Odyssey
--wx--S---   --- ---   1  ACE       SYS1      ...   Program2
-r-srwxrwx   -aa ---   1  BPXROOT   KNIGHTS   ...   SetuidPgm
drwxr-xr-x   fff ---   2  BPXROOT   SYS1      ...   TestDirectory
-rwxr----t   --- --a   1  ACE       JESTERS   ...   prog1
-rwxr-x--x   --- ---   2  BPXROOT   SYS1      ...   rac
lrwxrwxrwx   fff ---   1  BPXROOT   SYS1      ...   racSymlink -> rac
-rwxr-x--x   --- ---   2  BPXROOT   SYS1      ...   raclink
-rwxr-x---   --- ---   1  BPXROOT   SYS1      ...   racp
-rw-r--r--   -s- ---      1969      SYS1      ...   woodstock
```

auditor audit settings

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

File System Security Reporting - HFS Unload!!!

- irrhfsu command available on http://www.s390.ibm.com/products/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 (OK with automount)
- 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	grp name	set uid	set gid	sticky bit	owner read	owner write	owner execute	group read	et ...

Get it at: http://www.s390.ibm.com/products/racf/goodies.html

* AIM - Application Identity Mapping. Available on OS/390 V2R10

HFS Unload *(continued)*

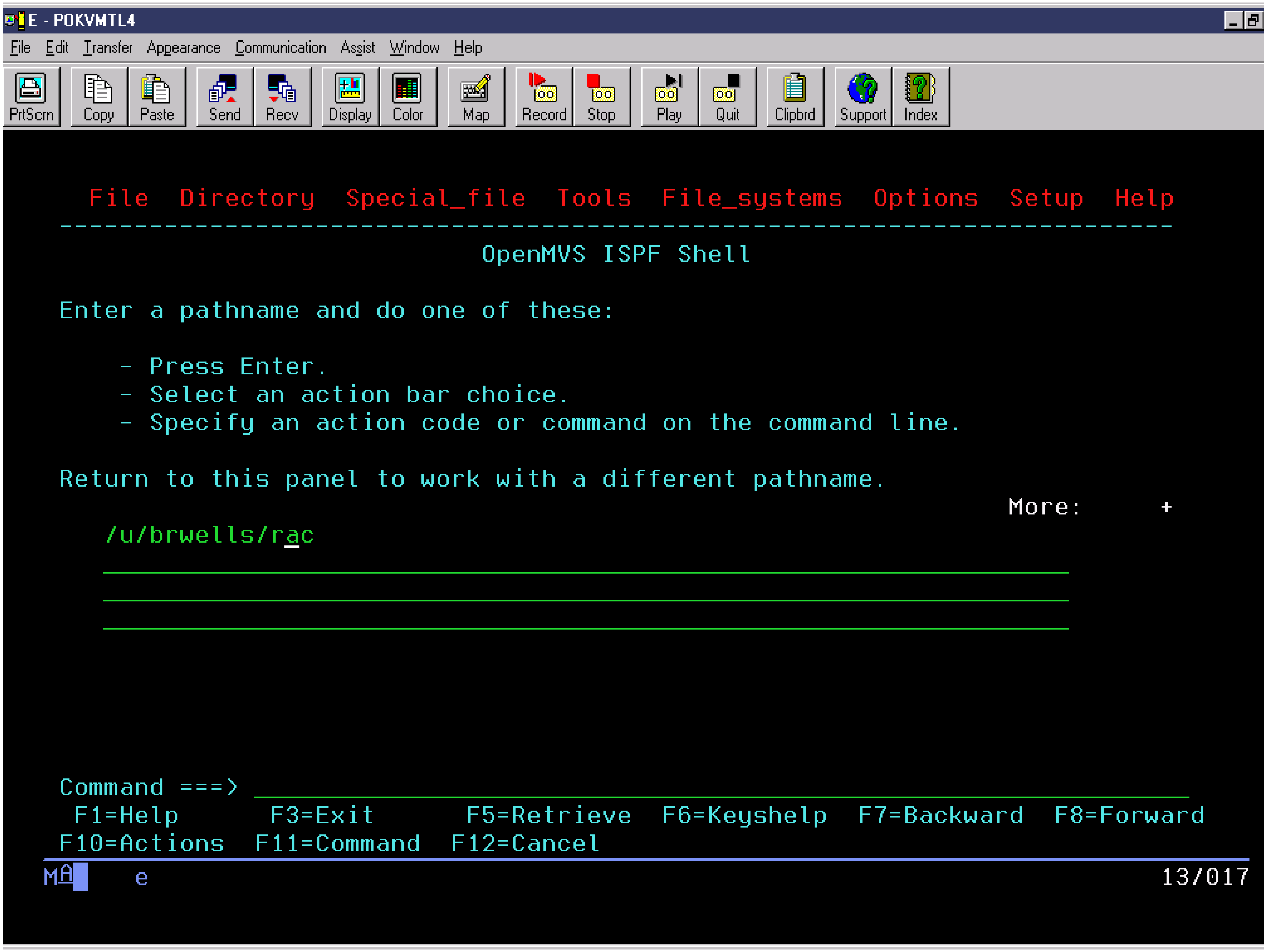
- Integrate it with current IRRDBU00 procedure

```
//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
```

UNIX ISPF Shell (ISHELL)

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


E - POKVMTL4
File Edit Transfer Appearance Communication Assist Window Help
PrtScrn Copy Paste Send Recv Display Color Map Record Stop Play Quit Clipbrd Support Index

```
  File  Directory  Special_file  Tools  File_systems  Options  Setup  Help
 --------------------------------------------------------------------------
                             OpenMVS ISPF Shell

 Enter a pathname and do one of these:

     - Press Enter.
     - Select an action bar choice.
     - Specify an action code or command on the command line.

 Return to this panel to work with a different pathname.
                                                               More:     +
    /u/brwells/rac
    ________________________________________________________________
    ________________________________________________________________
    ________________________________________________________________






 Command ===> ______________________________________________________________
  F1=Help      F3=Exit      F5=Retrieve  F6=Keyshelp  F7=Backward  F8=Forward
 F10=Actions  F11=Command  F12=Cancel
MA      e                                                            13/017
```

What do we need to remember?

- Read the security chapters of the z/OS UNIX System Services Planning manual for YOUR release level (SC28-1890)
- To set up security for a z/OS UNIX application, read its documentation for recommendations
- Use the ISPF Shell (ISHELL) if you don't like that UNIX feel
- It's still RACF and MVS under the surface

Good Sources of Information

- UNIX System Services Planning manual SC28-1890 (for your release)
  - Available online at http://www-1.ibm.com/servers/s390/os390/bkserv/
- UNIX System Services Command Reference
- UNIX System Services web site, at http://www-1.ibm.com/servers/eserver/zseries/zos/unix/
- mvs-oe mailing list (see the Forums link at the previous web site for information)

Appendix

Using the FIELD class to delegate OMVS administration

```
ADDUSER UXADM PASSW(yyyyyy) DFLTGRP(UNIXGRP)
   OMVS(UID(1000)) TSO(...)
SETR CLASSACT(FIELD) GENERIC(FIELD)
RDEF FIELD USER.OMVS.* UACC(NONE)
RDEF FIELD USER.OMVS.PROGRAM  UACC(NONE)
PE USER.OMVS.* CL(FIELD) ID(UXADM) ACC(UPDATE)
PE USER.OMVS.PROGRAM CL(FIELD) ID(UXADM)
ACC(UPDATE)
PE USER.OMVS.*  CL(FIELD) ID(&RACUID) ACC(READ)
PE USER.OMVS.PROGRAM  CL(FIELD) ID(&RACUID)
ACC(UPDATE)
SETR RACLIST(FIELD)
```

Defining the kernel and initialization proc

```
ADDGROUP OMVSGRP OMVS(GID(1))
AU OMVSKERN DFLTGRP(OMVSGRP) PASSWORD(xyz)
OMVS(UID(0) HOME('/') PROGRAM('/bin/sh'))
        NAME('OMVS KERNEL')
SETR GENERIC(STARTED)
RDEF STARTED OMVS.* STDATA(USER(OMVSKERN)
        GROUP(OMVSGRP) TRUSTED(YES))
RDEF STARTED BPXOINIT.*  STDATA(USER(OMVSKERN)
        GROUP(OMVSGRP) TRUSTED(NO))
SETR CLASSACT(STARTED) RACLIST(STARTED)
```

- Define default OMVS user/group
- Define BPX.SUPERUSER profile

```
AG  UXDFLTG OMVS(GID(999))
AU  UXDFLTU  DFLTGRP(UXDFLTG)  NOPASSWORD
    OMVS(UID(999)) NAME('DEFAULT UNIX USER')
RDEF  FACILITY  BPX.DEFAULT.USER
    APPLDATA('UXDFLTU/UXDFLTG')

RDEF   FACILITY  BPX.SUPERUSER UACC(NONE)
AG SUPERUSE OMVS(GID(3))
PERMIT BPX.SUPERUSER CLASS(FACILITY)  ID(SUPERUSE)
        ACCESS(READ)
SETR CLASSACT(FACILITY) RACLIST(FACILITY)
                              ...OR...
SETR RACLIST(FACILITY) REFRESH
```

UNIXPRIV Resource Names

Mount and Quiesce File Systems

Resource Name	Privilege	Access Req'd
SUPERUSER.FILESYS.MOUNT	**mount or unmount file system with nosetuid attribute**	**READ**
SUPERUSER.FILESYS.MOUNT	**mount or unmount file system with setuid attribute**	**UPDATE**
SUPERUSER.FILESYS.QUIESCE	**quiesce or unquiesce a file system mounted with nosetuid**	**READ**
SUPERUSER.FILESYS.QUIESCE	**quiesce or unquiesce a file system mounted with setuid**	**UPDATE**

UNIXPRIV Resource Names

Other File System Resources

Resource Name	Privilege	Access Req'd
SUPERUSER.FILESYS.CHOWN	**change owner of any file using chown**	**READ**
SUPERUSER.FILESYS.PFSCTL	**allows use of the pfsctl() service**	**READ**
SUPERUSER.FILESYS.VREGISTER	**allows use of vreg() service to register as a VFS file server**	**READ**

UNIXPRIV Resource Names

Process Manipulation

Resource Name	Privilege	Access Req'd
SUPERUSER.PROCESS.GETPSENT	**allows use of w_getpsent() service to retrieve info for any process**	**READ**
SUPERUSER.FILESYS.KILL	**allows user to send signals to any process**	**READ**
SUPERUSER.FILESYS.PTRACE	**allows use of dbx debugger against any process ***	**READ**

* For APF authorized or BPX.SERVER processes, also need FACILITY BPX.DEBUG in order to debug using SUPERUSER.FILESYS.PTRACE

UNIXPRIV Resource Names

Miscellaneous Resources

Resource Name	Privilege	Access Req'd
SUPERUSER.IPC.RMID	**allows user to release IPC resources**	**READ**
SUPERUSER.SETPRIORITY	**allows user to increase own priority**	**READ**
CHOWN.UNRESTRICTED	**if this profile exists, users can change ownership of files they own**	**N/A**

A Sample of BPX profiles available in the FACILITY class

- BPX.DAEMON - restricts the use of sensitive services
- BPX.DEBUG - allows debugging of authorized programs
- BPX.FILEATTR.APF - controls marking files authorized
- BPX.FILEATTR.PROGCTL - controls marking files program controlled
- BPX.SERVER - restricts the use of sensitive services
- BPX.SMF - allows the writing of SMF records
- BPX.STOR.SWAP - controls making address spaces non-swappable
- BPX.WLMSERVER - controls access to WLM interface
- BPX.SAFFASTPATH - improves performace but prevents auditing of successful events