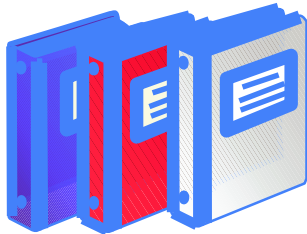


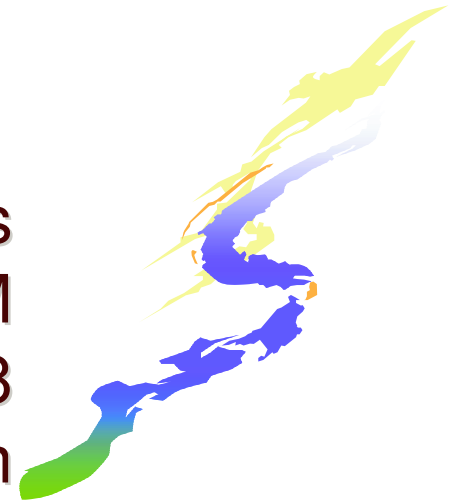


# RACF and z/OS UNIX Identities

Vanguard Enterprise Security Expo 2005  
Session G5



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:
  - 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
- UNIX vs. MVS identity
- Defining a UNIX user and group
  - ISHELL - ISPF interface to UNIX
  - Superusers
  - User resource limits
  - Default UNIX user/group identity
- Methods of changing identity
  - set-uid and set-gid files
  - the 'su' command
  - UNIX servers and daemons
- Auditing



# What is z/OS UNIX System Services?

---

- Standards-based, fully branded UNIX interface for 'MVS' providing
  - Hierarchical file system (directories and files)
  - Application Programming Interfaces
  - Shell and utilities (commands) for humans
- Services integrated with MVS. E.G.:
  - Invoke UNIX programs from TSO or BATCH
  - Issue TSO commands from UNIX shell
  - Manage file system from shell, TSO, console
  - Open any file/data set from any environment
- In short, it enables z/OS portability and interoperability in a heterogeneous world



# How does RACF support UNIX?

---

- User and group registry merged with RACF database
- User identification and authentication
- Protection of files and file systems
- Protection of services (su, chmod, chown, etc)
- Auditing of security events
- Protection above and beyond that required by the standard



# 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.USER

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

From OMVS segments of user's list of groups

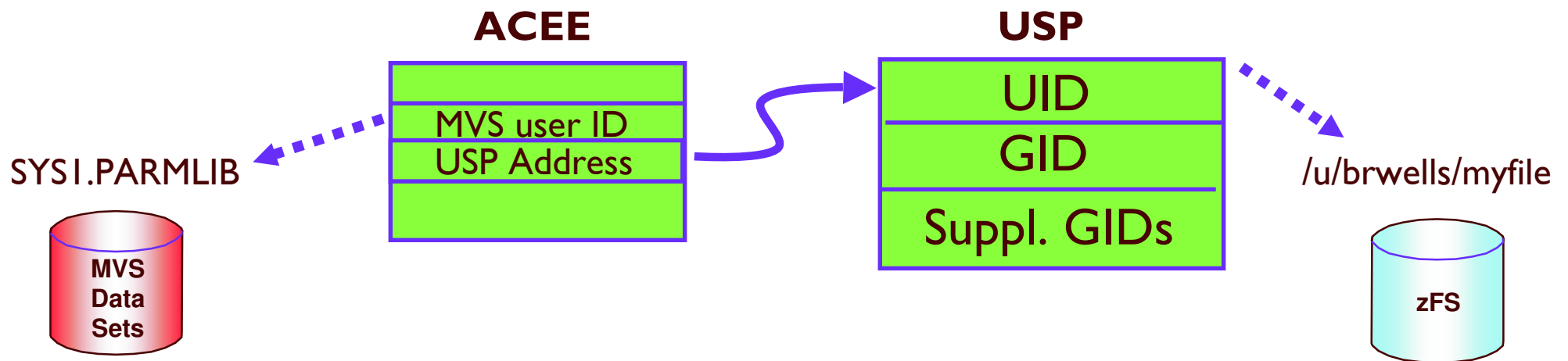
- **USP** created when first **UNIX** service is invoked
- use the **id** command to show user's **UNIX** identity

# id 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



# UNIX Group 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
  - 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...)



# UNIX User and Group 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
  - 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...)



# User Definition ...

---

```
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 = 0000000008
```

```
HOME = /u/antoine
```

```
PROGRAM = /bin/sh
```

```
CPUTIMEMAX= NONE
```

```
ASSIZEMAX= NONE
```

```
FILEPROCMAx= NONE
```

```
---
```



# User definition ...

## System Resource Limits

---

- **UNIX System Services provides global resource limits on a per user basis in BPXPRMxx:**
  - **MAXASSIZE:** address space region size
  - **MAXCPU TIME:** cpu time
  - **MAXFILEPROC:** open files per process
  - **MAXPROCUSER:** processes per UID
  - **MAXTHREADS:** threads per process
  - etc
- **Can be reset by SETOMVS or SET OMVS**
- **SUPERUSER can choose to exceed these limits**



# User definition ...

## System Resource Limits

---

- OMVS segment keywords on ADDUSER and ALTUSER allow specific limits for individual users:
  - ASSIZEMAX(address-space-size)
  - CPUTIMEMAX(cpu-time)
  - FILEPROCMAX(files-per-process)
  - PROCUSERMAX(processes-per-UID)
  - THREADSMAX(threads-per-process)
  - etc
- Listed via LISTUSER
- Limits not specified in segment taken from BPXPRMxx.



# 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 MVS resources, but a superuser **may** be able to assume any MVS user ID



# User Definition ... SUPERUSER!

---

- A superuser essentially has **SPECIAL** and **OPERATIONS!!!!**
- To the best of your ability, you should avoid assigning UID(0) to human administrators
  - use UNIXPRIV class or BPX.SUPERUSER (more later ...)
- UID(0) for started task users, and UNIX servers and daemons, is generally OK
  - use the NOPASSWORD attribute to prevent these from being logged onto



# 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 HFS 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





# UNIXPRIV Resource Names

---

## Example: File and Directory Access

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



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

# 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 during 'dub' when user/group doesn't have an OMVS segment
- Can be overridden on a per-user basis
  - **ALTUSER BOB OMVS(NOUID)**
- Use of default identity is audited
- Should have only limited use
  - **TCP/IP from MVS to MVS, or, just getting your feet wet with UNIX System Services**



# Prevention of shared IDs

---

- 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
- Requires AIM stage 2 or 3 (See IRRIRA00 utility to implement Application Identity Mapping)



# Prevention of shared IDs

---

- Does not affect pre-existing shared IDs
  - Must clean those up separately, if desired
    - Not a pre-req for using the new support
  - Can use supplied IRRICE reports to find shared UIDs and GIDs

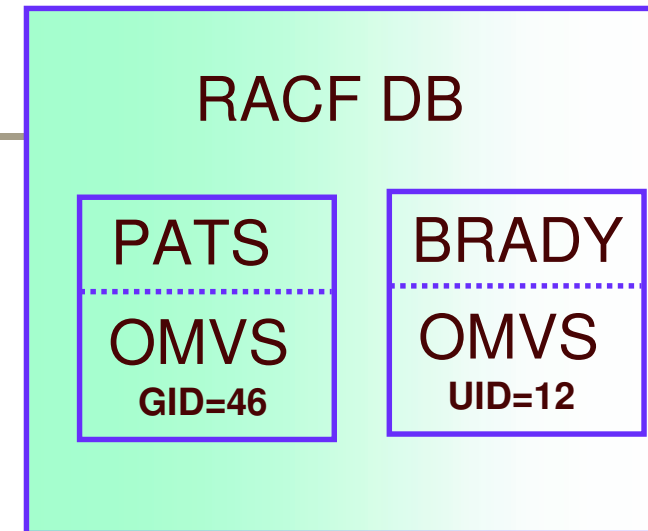


# Prevention of shared IDs ...

## Example

---

- `RDEFINE UNIXPRIV SHARED.IDS UACC(NONE)`
- `SETROPTS RACLIST(UNIXPRIV) REFRESH`



- `ADDUSER MARCY OMVS(UID(12))`  
`IRR52174I Incorrect UID 12. This value is already in use by BRADY.`
- `ADDGROUP ADK OMVS(GID(46))`  
`IRR52174I Incorrect GID 46. This value is already in use by PATS.`



# Prevention of shared IDs ... New SHARED keyword

---

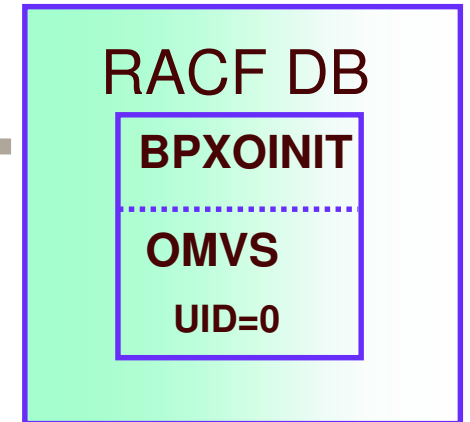
- There are valid reasons to assign a non-unique UID/GID
  - E.G. Assigning UID(0) to started task user IDs
- Do so using the new SHARED keyword in the OMVS segment of the ADDUSER, ALTUSER, ADDGROUP, and ALTGROUP commands
- SHARED requires SPECIAL, or at least READ authority to SHARED.IDS
  - Profile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



# Prevention of shared IDs ...

## Example

---

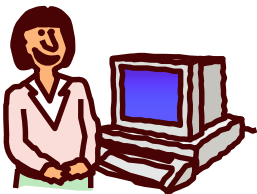


- PERMIT SHARED.IDS CLASS(UNIXPRIV) ID(UNIXGUY) ACCESS(READ)

- SETROPTS RACLIST(UNIXPRIV) REFRESH



AU OMVSKERN OMVS(UID(0) SHARED)  
AG (G1 G2 G3) OMVS(GID(9) SHARED)



AU MYBUDDY OMVS(UID(0) SHARED)

IRR52175I You are not authorized to specify the SHARED keyword.

# So you don't want to convert to AIM???

---

- Prior to OS/290 V2R10, profiles in the UNIXMAP class were used to map UIDs to user IDs and GIDs to group names
- UNIXMAP profiles automatically maintained by RACF commands
- RLIST UNIXMAP Unnn ALL shows all users with UID(nnn)
- In OS/390 V2R10, customers migrating to AIM lost that capability
- Not anymore!!!

Next Page





# 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



# Automatic UID/GID Assignment

---

- AUTOUID keyword in the OMVS segment of the ADDUSER and ALTUSER commands
- AUTOGID keyword in the OMVS segment of the ADDGROUP and ALTGROUP commands
- Derived values are guaranteed to be unique



**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.**

# 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 consists of 2 qualifiers separated by a forward slash ("/")
  - left qualifier specifies starting UID value, or range of UID values
  - right qualifier specifies starting GID value, or range of GID values
  - qualifiers can be null, or specified as 'NOAUTO', to prevent automatic assignment of UIDs or GIDs



# Automatic UID/GID Assignment ...

## APPLDATA syntax

---

- Examples

- RDEFINE FACILITY BPX.NEXT.USER  
APPLDATA(*'data'*)

- good *data*

- I/0
- I-50000/I-50000
- NOAUTO/I00000
- /I00000
- I0000-20000/NOAUTO
- I0000-20000/



# Automatic UID/GID Assignment ...

## APPLDATA

---

- When AUTOUID or AUTOGID is issued, RACF
  - 1 extracts the APPLDATA from BPX.NEXT.USER
  - 2 parses out the starting value
  - 3 checks to see if it is already in use
    - If so, the value is incremented and checked again until an unused value is found
  - 4 assigns the value to the user or group
  - 5 replaces the APPLDATA with the new starting value
- The administrator can change the APPLDATA at any time using RALTER



# Automatic UID/GID Assignment ...

## Miscellany

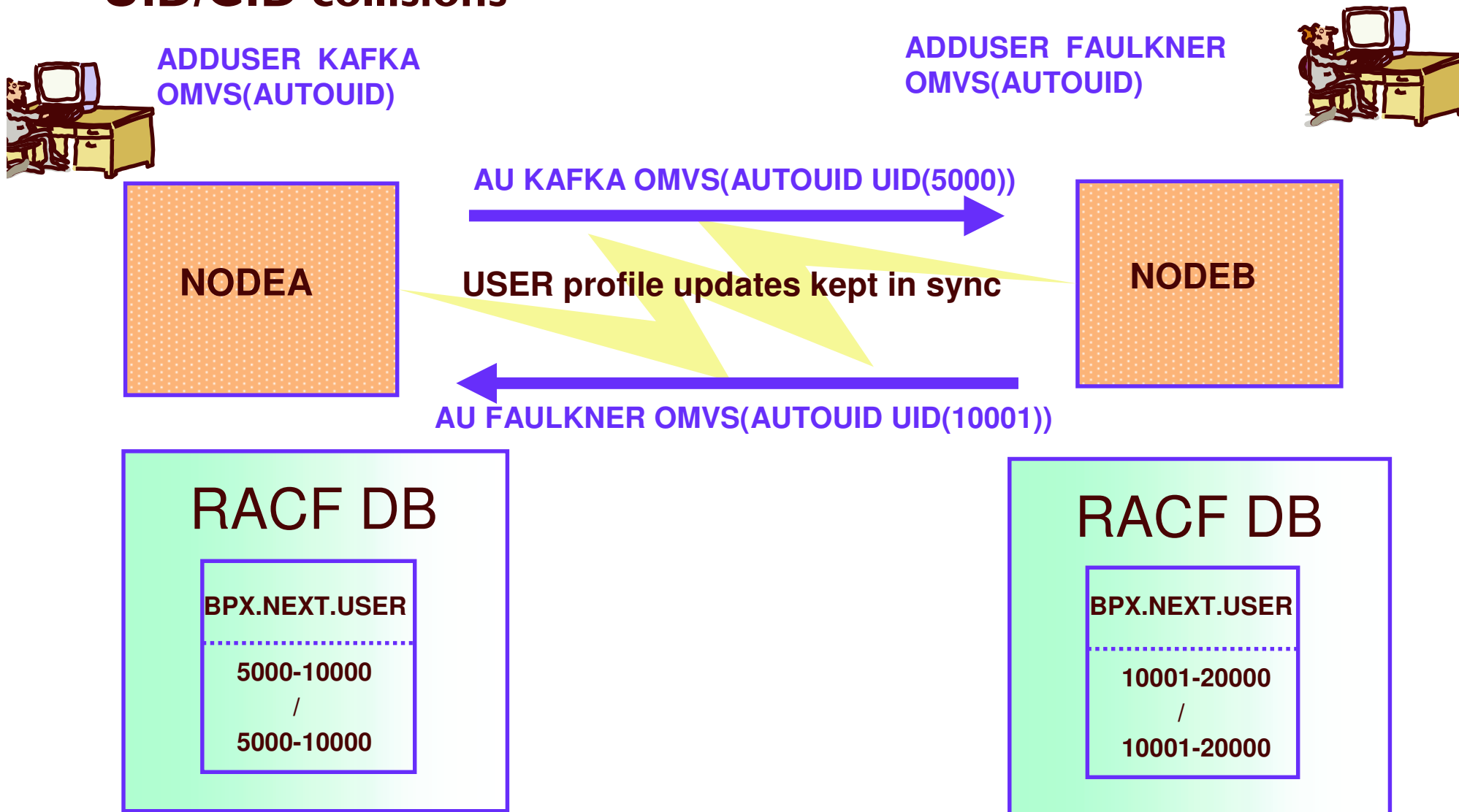
---

- Must be enforcing uniqueness with SHARED.IDS in order to use AUTOUID/AUTOGID
- Which in turn requires AIM stage 2 or 3
- Want an easy way to assign a unique GID to all your groups?
  - SEARCH CLASS(GROUP) NOLIST  
CLIST('ALTGROU ' ' OMVS(AUTOGID)')
  - EX EXEC.RACF.CLIST



# Automatic Assignment in an RRSF Configuration

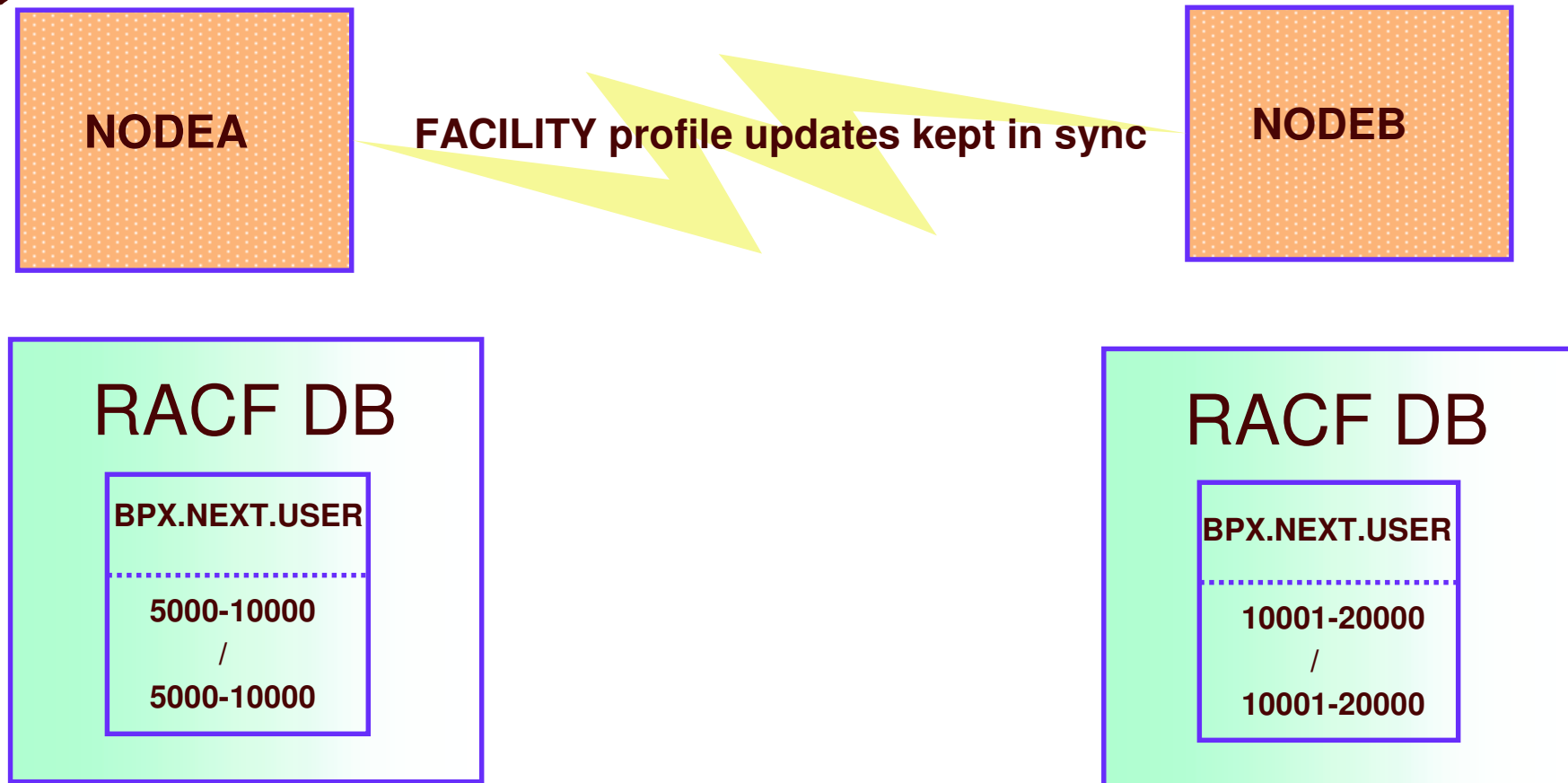
- Use non-overlapping APPLDATA ranges to avoid UID/GID collisions



# Automatic Assignment in an RRSF Configuration

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

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





# Agenda

---

- z/OS UNIX Introduction
- UNIX vs. MVS identity
- Defining a UNIX user and group
  - ISHELL - ISPF interface to UNIX
  - Superusers
  - User resource limits
  - Default UNIX user/group identity
- Methods of changing identity
  - set-uid and set-gid files
  - the 'su' command
  - UNIX servers and daemons
- Auditing



# 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 APIs such as `setuid()`, `setgid()`, `pthread_security_np`, etc
  - Used by UNIX servers and daemons



# 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`

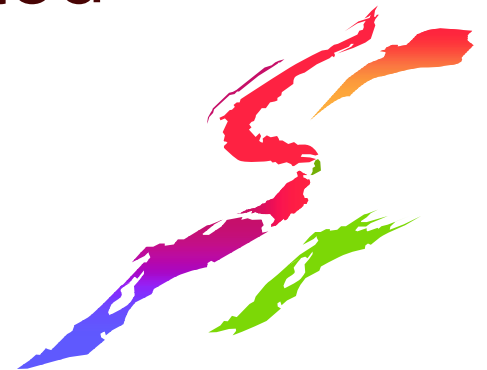


```
-rwsr-s--x  2 BRWFHLS DEPTD60  8192 Feb  8 10:51 mvnprogram
```

# set-UID and set-GID files (continued)

---

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



# su Command

---

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



# Some new terms we will hear

---

- process - a program using kernel services running in an address space
- thread - a task doing the same
- daemon - a process that changes identities
- fork - creation of a child process in a new address space
- spawn - creation and starting of a child process that runs a named program (fork plus exec)
- server - a process that does work for clients
- client - a user



# What is a Daemon?

---



- A program that starts at initialization time (a started task or cataloged procedure)
- A long-lived process, running unattended
- A service provider
- An authorized, superuser process
- Daemons perform work for users by:
  - Verifying the user requesting the service
  - Creating a new process to do the work
  - Giving the new process the user's identity



# What is a Daemon?

---

- Daemons supplied with z/OS UNIX:
  - inetd - the internet daemon
  - rlogind - the remote login daemon
  - cron - the batch scheduler
  - lm - the Communications Server login monitor
  - uucpd - the UUCP (UNIX-to-UNIX Copy Program) daemon
- Daemons supplied with IBM Communications Server for z/OS
  - svslor - message routing daemon





# How Does a Daemon Change Identity?

---

- Daemon programs call identity changing services to alter the UID and RACF user ID of an address space
  - `seteuid()`
  - `setuid()`
  - `spawn()` with a user ID
- Daemons can become any user that has an OMVS segment defined (any user if you have `BPX.DEFAULT.USER` setup)



# Controlling Daemons ...

## Selecting your level of security

---

- Typical UNIX-level security
  - A superuser is equivalent to a daemon
    - Superusers can change identities
      - ▶ **UNIX and MVS!**
    - Superusers can access all resources
      - ▶ **UNIX and MVS!**
  - A superuser is equivalent to a system programmer
    - essentially has **SPECIAL and OPERATIONS!!!**



# Controlling Daemons ...

## Selecting your level of security

---

- z/OS UNIX-level security
  - A superuser is not equivalent to a daemon
    - Superusers cannot directly change MVS identity
    - Superusers cannot directly access MVS resources
  - A superuser maintains the file systems



# Controlling Daemons ...

## z/OS UNIX-Level Security

---

- Activated by defining FACILITY BPX.DAEMON
- Restricts the use of identity changing services
- Only trusted daemons should be given authority
  
- The daemon address space must be kept clean
  - If a program that is NOT a controlled program is loaded, the address space is marked dirty and cannot perform daemon activities
- Clean environment ensures daemons perform their intended function



# Controlling Daemons ...

## z/OS UNIX-Level Security

---

- All programs loaded must be controlled
  - PROGRAM profiles covering all programs from MVS 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 *noasetuid* 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



# What is a Server?

---



- A program that starts at initialization time (a started task or cataloged procedure)
- A long-lived process, running unattended
- A service provider
- Not necessarily authorized or superuser
- Servers perform work for users by:
  - Verifying or **trusting** the user requesting the service
  - Creating a new **thread** to do the work
  - Giving the new **thread** the user's identity



# How Does a Server Do 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
  - Associates the client's user ID with the thread
    - `pthread_security_np` (BPX1TLS)
  - Checks the client's authority when accessing z/OS resources
    - `_check_resource_auth_np` (BPX1ACK)
- Not all servers are well-behaved



# Controlling Servers ...

## Selecting your level of security

---

- Typical UNIX-level vs. z/OS UNIX level security applies
  - Activated by defining FACILITY BPX.SERVER
  - Restricts use of *pthread\_security\_np* and *auth\_check\_resource\_np*
  - UPDATE for trustworthy servers
    - Only the client's authority is checked
  - READ for servers that need added control
    - Client's and server's authority checked
    - SURROGAT allows server to represent client
    - Support for anonymous users
  - Clean address space is required





# Agenda

---

- z/OS UNIX Introduction
- UNIX vs. MVS identity
- Defining a UNIX user and group
  - ISHELL - ISPF interface to UNIX
  - Superusers
  - User resource limits
  - Default UNIX user/group identity
- Methods of changing identity
  - set-uid and set-gid files
  - the 'su' command
  - UNIX servers and daemons
- Auditing



# 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



# 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
```


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

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



# 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
- 

# Good Sources of Information

---

- UNIX System Services web site, at <http://www-1.ibm.com/servers/eserver/zseries/zos/unix/>
  - UNIX System Services Planning manual SC28-1890 (for your release)
    - 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)
  - Check program product documentation for daemon or server security setup
- 