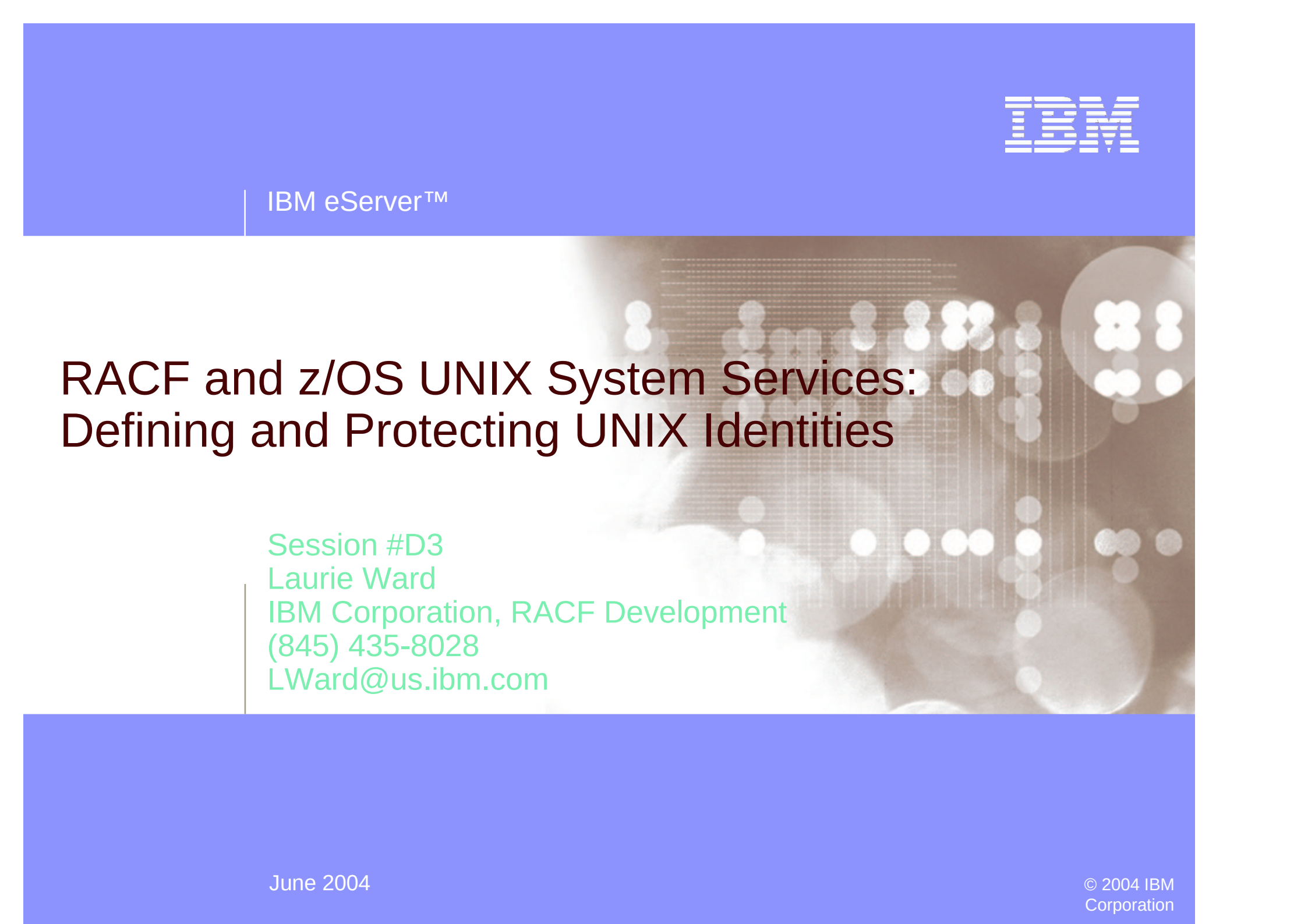


IBM eServer™

RACF and z/OS UNIX System Services: Defining and Protecting UNIX Identities

Session #D3
Laurie Ward
IBM Corporation, RACF Development
(845) 435-8028
LWard@us.ibm.com

June 2004

Disclaimer

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

Trademarks

- **The following are trademarks or registered trademarks of the International Business Machines Corporation:**

  OS/390

  z/OS

  RACF

- **UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.**

Agenda

- z/OS UNIX Introduction
- UNIX vs. z/OS identity
- Defining a UNIX user and group
  - 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

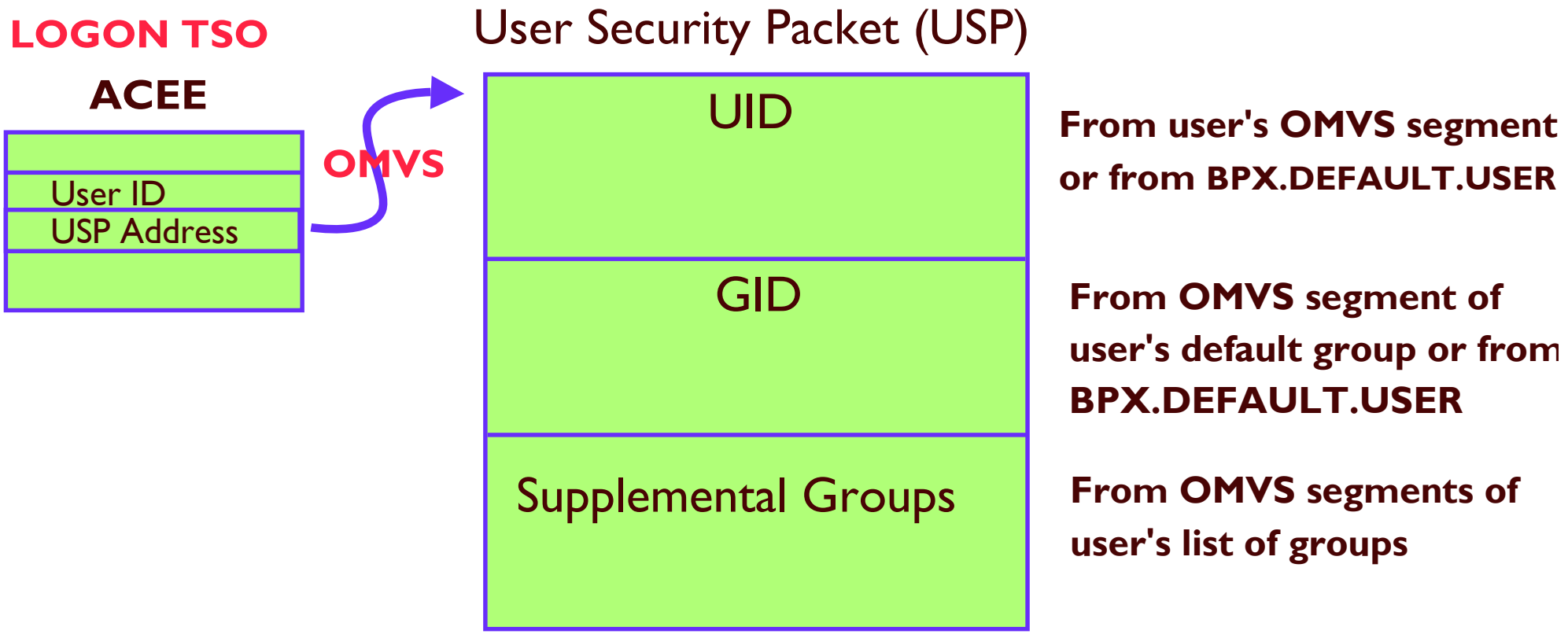
What is z/OS UNIX System Services?

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

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

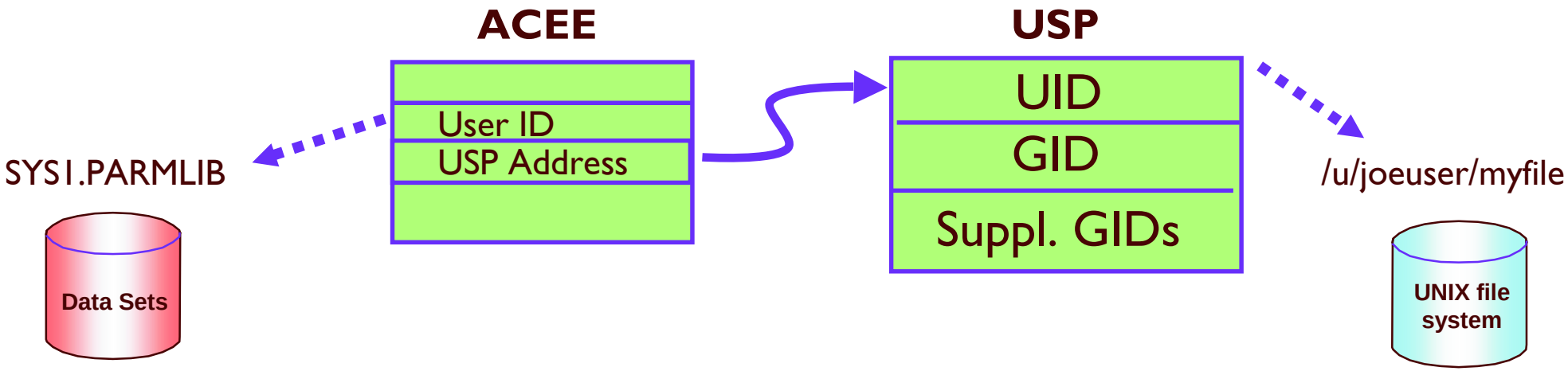
UNIX Identity



- **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

UNIX User definition

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

User Definition Example

ADDGROUP UNIXGRP OMVS(GID(100))

ADDUSER ANTOINE PASSW(xxxx) DFLTGRP(UNIXGRP)

OMVS(UID(8) HOME(/u/antoine) PROGRAM(/bin/sh))

TSO(ACCTNUM(12345) PROC(PROC01))

LISTUSER ANTOINE OMVS NORACF

USER=ANTOINE
OMVS INFORMATION

UID = 0000000008
HOME = /u/antoine
PROGRAM = /bin/sh
CPUTIMEMAX= NONE
ASSIZEMAX= NONE
FILEPROCMAX= NONE
...

User Definition ... SUPERUSER!

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

User Definition ... SUPERUSER!

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

SUPERUSER Granularity: UNIXPRIV Class

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

UNIXPRIV Resource Names

Examples:

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

See z/OS UNIX System Services Planning for complete list of UNIXPRIV resources

User definition ...

System Resource Limits

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

User definition ...

System Resource Limits

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

Example:
ALTUSER SERVER1 OMVS(THREADSMAX(400))

Default UNIX User and Group Identity

- BPX.DEFAULT.USER in the FACILITY class can be used to assign default OMVS segment data
  - **RDEFINE FACILITY BPX.DEFAULT.USER APPLDATA('DFTUSER/DFTGROUP')**
  - **ADDUSER DFTUSER OMVS(... ... ...) NOPASSWORD**
  - **ADDGROUP DFTGROUP OMVS(GID(nnn))**

- Assigned when user/group doesn't have an OMVS segment
- Can be overridden on a per-user basis
  - **ALTUSER BOB OMVS(NOUID)**
- Use of default identity is noted in any audit record produced
- Should have only limited use
  - TCP/IP from z/OS to z/OS, or, just getting your feet wet with UNIX System Services

Prevention of Shared IDs

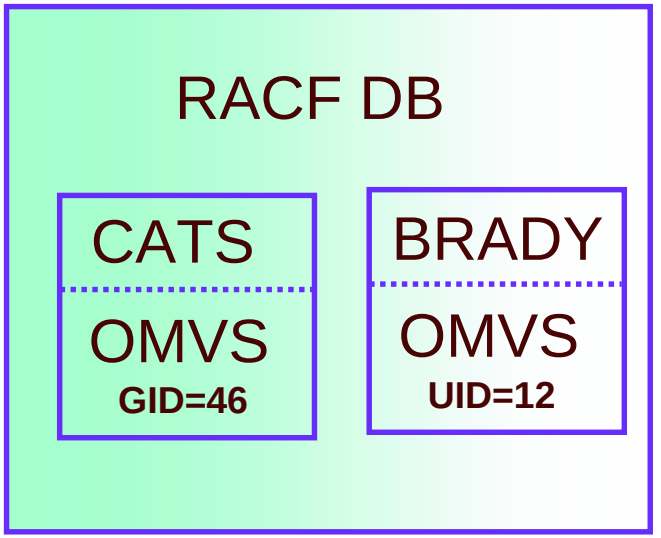
- New SHARED.IDS profile in the UNIXPRIV class
- Acts as a system-wide switch to prevent assignment of an ID which is already in use
- No generic characters allowed in name: discrete profile name must be used
- APAR OW52135
  - OS/390 2.10: UW89970 - also applies to z/OS 1.1
  - z/OS 1.2: UW89971
  -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

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

Prevention of shared IDs

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

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 DOGS OMVS(GID(46))

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

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

NEW!

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

- SETROPTS RACLIST(UNIXPRIV) REFRESH

RACF DB

BPXOINIT

OMVS

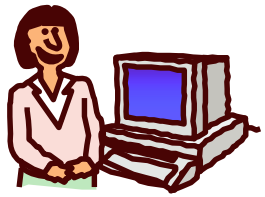
UID=0

UNIXGUY

AU OMVSKERN OMVS(UID(0) SHARED)

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

OK!



MVSGAL

AU MYBUDDY OMVS(UID(0) SHARED)

IRR52175I You are not authorized to specify the SHARED keyword

SEARCH Enhancement to Map UIDs and GIDs

- SEARCH CLASS(USER) UID(0)
  - **OMVSKERN**
  - **BPXOINIT**
  - **SUPERGUY**
- SEARCH CLASS(GROUP) GID(99)
  - **RACFDEV**
- SEARCH CLASS(USER) UID(1234567)

ICH31005I NO ENTRIES MEET SEARCH CRITERIA

- Available with OW52135

Automatic UID/GID Assignment

NEW!

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

ADDUSER MELVILLE OMVS(AUTOUID)

IRR52177I User MELVILLE was assigned an OMVS UID value of 4646

ADDGROUP WHALES OMVS(AUTOGID)

IRR52177I Group WHALES was assigned an OMVS GID value of 105.

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

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

Automatic UID/GID Assignment ... APPLDATA syntax

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

Ways of Assuming Another UNIX Identity

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

set-UID and set-GID files

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

```
-rwsr-s--x   2 JILL    DEPTD60    8192 Feb  8 10:51 myprogram
```

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

- Changing file ownership (chown command), or writing to the file, resets set-uid and set-gid bits

- Consider mounting remote/untrusted file systems with the NOSETUID option

su Command

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

Controlling Daemons

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

Controlling Servers

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

Auditing Users and Processes

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

UNIX Auditing ...

The results

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

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

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

See: z/OS Security Server RACF Auditor's Guide and z/OS Security Server Macros and Interfaces

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.ibm.com/servers/eserver/zseries/zos/unix/
- UNIX System Services Planning manual
  - Available online

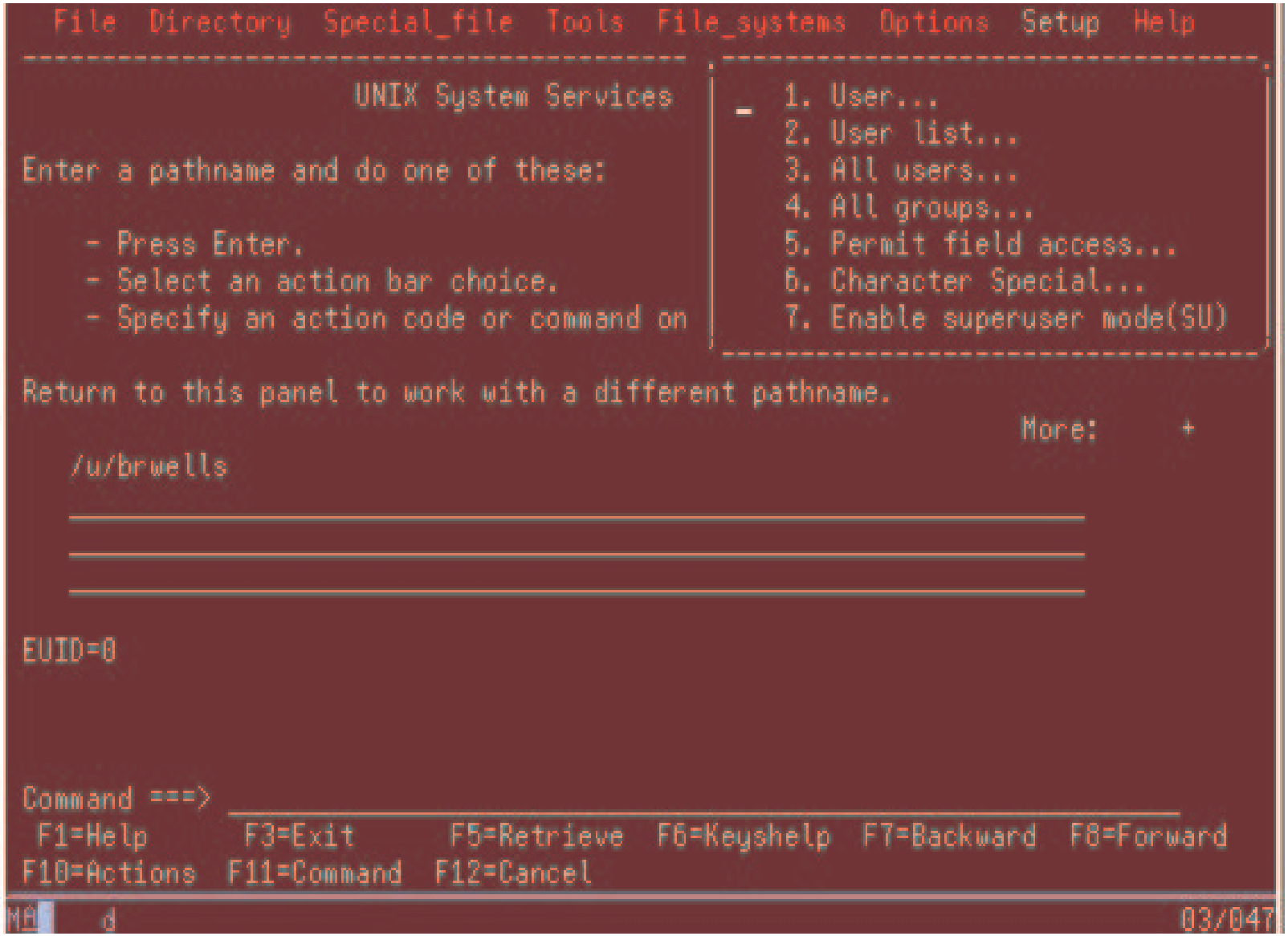
  For OS/390: http://www.ibm.com/servers/s390/os390/bkserv/

  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)
- Check program product documentation for daemon or server security setup

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

The ISPF Shell ... A Panel Interface



Automatic Assignment in an RRSF Configuration

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

ADDUSER JACK OMVS(AUTOUID)

ADDUSER JILL OMVS(AUTOUID)

AU JACK OMVS(AUTOUID UID(5000))

NODEA

USER profile updates kept in sync

NODEB

AU JILL OMVS(AUTOUID UID(10001))

RACF DB

BPX.NEXT.USER

5000-10000
/
5000-10000

RACF DB

BPX.NEXT.USER

10001-20000
/
10001-20000

Automatic Assignment in an RRSF Configuration

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

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