z/OS

Session 2914 z/OS UNIX Advanced Topics

March 4, 2005

Garth Godfrey
z/OS UNIX System Services Development
IBM Poughkeepsie, NY
ggodfrey@us.ibm.com
http://www.ibm.com/servers/eserver/zseries/zos/unix
http://www.ibm.com/servers/eserver/zseries/zos/unix/port_tools.html

Trademarks

- **The following are trademarks of the IBM Corporation in the United States or other countries or both:**

  IBM

  InfoPrint

  Language Environment

  z/OS

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

Session Objectives

z/OS UNIX System Services provides callable services, command interfaces, and facilities to enable UNIX applications to run on z/OS

At the completion of this session, you will be able to identify z/OS UNIX capabilities:

- UNIX shell commands
- System Management
- Security

z/OS UNIX

Need Addressed	Solution
UNIX shell commands	skulker su fuser uptime
System Management	BPXPRMxx Syntax Checker Monitoring BPXPRMxx limits Remount
Security	Superuser Granularity ACLs Managing UID / GID assignment OpenSSH

skulker OS/390

- skulker [-irw] [-l *logfile*] *directory days_old*
- removes files in directory older than specified number of days
- shell script in **/samples**
  - copy to /bin/skulker or /usr/sbin/skulker
  - can be modified by installation
  - **Protect it from hackers!** (make it non-writable)
- *e.g.* skulker /tmp/ 100
  - deletes files in /tmp older than 100 days
  - trailing slash follows a /tmp symlink to another directory
- use **cron** to schedule it to run regularly

su login shell option z/OS 1.5

- Prior to v1r5,
  - **su** command starts a child shell with new user (UID) and groups, but
  - maintains current shell environment

- Using su login shell options, you can
  - Start the new user's default login shell
  - Run the new user's login profiles
  - Pass arguments to the child shell

- Advantages:
  - Provides UNIX function
  - Facilitates automation

su login shell option **examples**

➢ **su [-] [-s] [userid [arg ...]]**

example 1: su - admin

- starts a child shell with login environment of **admin** userid
  - admin's default shell
  - admin's HOME directory
  - runs /etc/profile and admin's .profile
  - environment variables

example 2: su admin /usr/lib/backup

- runs the /usr/lib/backup shell script under the **admin** userid
- returns to the invoker when the shell script ends

fuser, uptime

- **fuser** [-cfku] file ... *OS/390*
  - List process IDs of processes with open files
  - Useful for finding the current users of a file, or a filesystem (e.g. before unmount)
  - *e.g.* `fuser -cu /usr/lpp/dfs` *shows who is using the containing filesystem*

- **uptime** *z/OS 1.6*
  - Display how long the system has been IPLed
  - *e.g.* **uptime**

```
01:02PM   up 14 day(s), 01:15,   58 users,   load average: 0.00, 0.00, 0.00
```

current time

how long since system IPL (OMVS init)

logged in to z/OS UNIX

not available on z/OS

z/OS UNIX

Need Addressed	Solution
UNIX shell commands	skulker su fuser uptime
System Management	BPXPRMxx Syntax Checker Monitoring BPXPRMxx limits Remount
Security	Superuser Granularity ACLs Managing UID / GID assignment OpenSSH

BPXPRMxx Syntax Checker OS/390

- Option on the SETOMVS operator command to syntax check a BPXPRMxx parmlib member **before IPL**
  - Avoids OMVS initialization in minimum-mode for syntax errors

SETOMVS SYNTAXCHECK=(xx)

- Runs the same logic used at IPL or via SETOMVS
- Checks whether HFS / zFS data sets exist
- Any errors cause messages to be written to the system log
  - Same messages as IPL

Plus:

```
BPXO039I SETOMVS SYNTAXCHECK COMMAND SUCCESSFUL.

BPXO023I THE PARMLIB MEMBER BPXPRMXX CONTAINS SYNTAX ERRORS.
REFER TO HARD COPY LOG FOR MESSAGES.
```

BPXPRMxx Limit Management OS/390

- Monitor and manage Unix System Services parmlib values through operator messages and commands

- Console messages are issued
  - as the usage reaches 85%, 90%, 95% and 100% of the current limit
  - as the usage decreases and when it drops below 85%

Managing BPXPRMxx Parmlib Values

- Display command options

D OMVS,L	to display the **system** wide parmlib limits
D OMVS,L,PID=*nnnnnnnn*	to display the specific limits for a **process**
D OMVS,PFS	to display the high water mark for each **sockets** PFS

- commands to set the limit values

SETOMVS / SET OMVS

- the parmlib values take effect immediately

SETOMVS PID=nnnnnnnn

- to change the limit for a specific process

BPXPRMxx parmlib limits monitored

- SYSTEM level limits:
  - MAXPROCSYS
  - MAXUIDS
  - MAXPTYS
  - MAXMMAPAREA
  - MAXSHAREPAGES
  - IPCMSGNIDS
  - IPCSEMNIDS
  - IPCSHMNIDS
  - IPCSHMSPAGES
  - IPCMSGQBYTES
  - IPCMSGQMNUM
  - IPCSHMMPAGES
  - SHRLIBRGNSIZE
  - SHRLIBMAXPAGES

- PROCESS level limits:
  - MAXFILEPROC
  - MAXFILESIZE
  - MAXPROCUSER
  - MAXQUEUEDSIGS
  - MAXTHREADS
  - MAXTHREADTASKS
  - IPCSHMNSEGS
  - MAXCORESIZE
  - MAXMEMLIMIT *z/OS 1.6*

- SOCKETS Address Family level limit:
  - MAXSOCKETS

UNIX User Limits

- Stored in **OMVS** segment of user profile
  - CPUTIMEMAX
  - ASSIZEMAX
  - FILEPROCMAX
  - PROCUSERMAX
  - THREADSMAX
  - MMAPAREAMAX
  - MEMLIMIT *z/OS 1.6*

ADDUSER ... OMVS(CPU(100) ASSIZEMAX(200M) ...)

Monitoring Message controls

```
SETOMVS LIMMSG=NONE
               SYSTEM
               ALL
```

- **LIMMSG=NONE** (default)
  - **No** console messages issued for any of the limits.
- **LIMMSG=SYSTEM**
  - Console messages will be issued for
    - SYSTEM level limits
    - PROCESS level limits for a process if limit
      - is defined in the user's OMVS segment
      - was changed via the SETOMVS PID= command
- **LIMMSG=ALL**
  - Console messages issued for all SYSTEM and PROCESS level limits

Remount for Shared FS Environment z/OS 1.5

- Remount now supported in the Shared FS Environment
  - Switch between **read-only** and **read-write** mode
    - e.g. to apply maintenance
    - without unmounting filesystems mounted under it

- For use when all systems at:
  - V1R5 or V1R4 with APAR OA02584

- Supported through:
  - existing remount interfaces

    TSO UNMOUNT ... REMOUNT(rdwr)

    ISHELL ... File_systems pulldown
  - new chmount options -r -w

    chmount -w ***pathname***

z/OS UNIX

Need Addressed	Solution
UNIX shell commands	skulker su fuser uptime
System Management	BPXPRMxx Syntax Checker Monitoring BPXPRMxx limits Remount
Security	Superuser Granularity ACLs Managing UID / GID assignment OpenSSH

Superuser Granularity OS/390

- To minimize the users with BPX.SUPERUSER . . . or UID 0
- **UNIXPRIV** class Resource Names Supported in RACF:
  - CHOWN.UNRESTRICTED
  - FILE.GROUPOWNER.SETGID
  - RESTRICTED.FILESYS.ACCESS
  - SHARED.IDS
  - SUPERUSER.FILESYS.ACLOVERRIDE
  - SUPERUSER.FILESYS
  - SUPERUSER.FILESYS.CHANGEPERMS
  - SUPERUSER.FILESYS.CHOWN
  - SUPERUSER.FILESYS.MOUNT
  - SUPERUSER.FILESYS.QUIESCE
  - SUPERUSER.FILESYS.PFSCTL
  - SUPERUSER.FILESYS.VREGISTER
  - SUPERUSER.IPC.RMID
  - SUPERUSER.PROCESS.GETPSENT
  - SUPERUSER.PROCESS.KILL
  - SUPERUSER.PROCESS.PTRACE
  - SUPERUSER.SETPRIORITY

Access Control Lists z/OS 1.3

- UNIX files are protected with POSIX permission bits

User			Group			Other		
read	write	execute	read	write	execute	read	write	execute

- Can only specify permissions for file owner (user), group owner, and everybody else

- **Access Control Lists** permit/restrict access to specific users and groups

Access Control Lists (ACLs) Overview

- Traditional UNIX approach
- Contained within the file system
  - File security is portable
  - Deleted automatically if the file is removed
- Not protected by RACF profiles
- Managed using new UNIX shell commands, or ISHELL
- Supports inheritance for new files and subdirectories

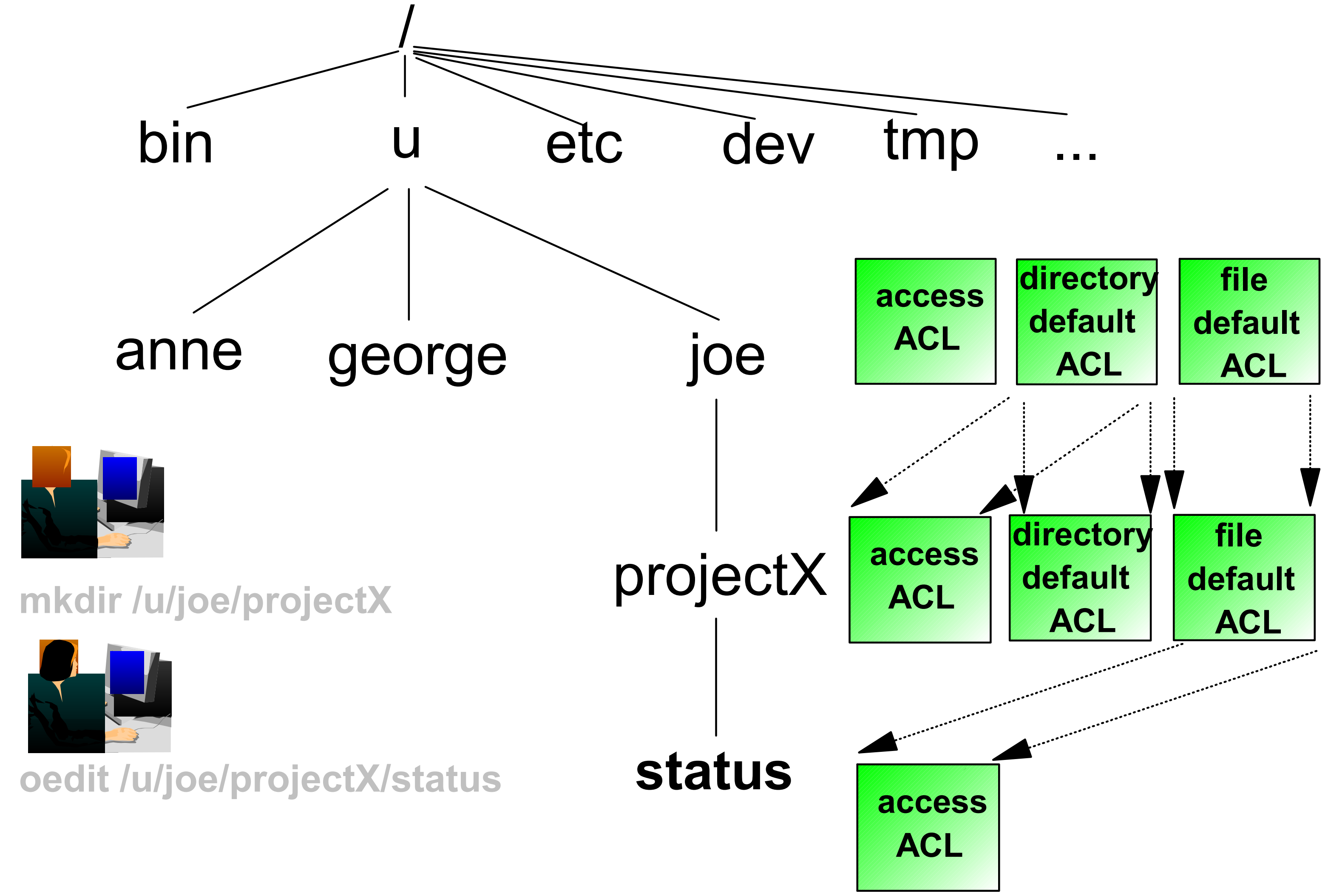
Participating File Systems

- HFS - Hierarchical File System
  - component of z/OS DFSMS 1.3
- zFS - z-Series File System
  - component of z/OS 1.3 Distributed File Service
- TFS - Temporary File System
  - supported in z/OS 1.5

ACL Inheritance

- Can establish default (or 'model') ACLs on a directory
- They will get automatically applied to new files/directories created within the directory
- Separate default ACL used for files and (sub)directories
- Can reduce administrative overhead

ACL Inheritance example



shell commands

- **setfacl** *set, remove, modify ACL entries*
  - Allowed by file owner

    or
  - superuser
    - UID 0

      or
    - READ access to

      SUPERUSER.FILESYS.CHANGEPERMS

- **getfacl** *display owner, group, ACL entries*
  - Allowed by anyone with directory search access

New terms

- **base ACL entries** = permission bits
  - user::*rwx*
  - group::*rwx*
  - other::*rwx*

- **extended ACL entries**
  - user:***uid***:*rwx*
  - group:***gid***:*rwx*

  - default:user:***uid***:*rwx*
  - default:group:***gid***:*rwx*

  - fdefault:user:***uid***:*rwx*
  - fdefault:group:***gid***:*rwx*

setfacl command

- setfacl -s ***entries*** [***path*** ...]
  - set (replace) entire ACL
  - must include base ACL entries (permission bits)

- setfacl -S ***file*** [***path*** ...]
  - set (replace) entire ACL from file
  - must include base ACL entries (permission bits)

- setfacl -D ***type*** ... [***path*** ...]
  - delete extended ACL entries of matching type

- setfacl -m|M|x|X ***EntryOrFile*** [***path*** ...]
  - modify or delete extended ACL entries

setfacl command . . .

- An ACL can be set from contents of a file
  - `setfacl -S ~/acls/ateam  rel4dir`

    where ~/acls/ateam contains an entire ACL (e.g.):

```
u::rwx
g::r-x
o::---
g:shut:rwx
g:testers:r-x
```

- Allows use of "named ACLs"

- An ACL can be set from stdin, and thus piped in from a getfacl command
  - `getfacl YourFile | setfacl -S - MyFile`

getfacl Display ACL contents

- getfacl MyFile
  - Displays file name, user owner, and group owner
  - Displays base POSIX permissions in "ACL format"
  - Displays access ACL entries

```
#file:  MyFile
#owner: BRUCE
#group: RACFDEV
user::rwx
group::r--
other::r--
user:GARTH:rwx
group:RACFDEV:r-x
```

ls command

list file / directory attributes

- ls command indicates existence of extended ACL entries

```
ls -l MyFile
-rwxrwxr-x+  1 GODFREY SHUT    44 Apr   3 14:49 MyFile
```

find *find files with matching criteria*

- `find` *path* `-acl a|d|f`
  - find all files with an ACL of a given type, or types

- `find` *path*
  ```
  -acl_user userid
  -acl_group groupid
  -acl_entry acl_text
  ```
  - find files with ACL entries for a specific user/group

- `find` *path* `-acl_count number`
  - find files with (more than) number ACL entries

find Command substitution

- Useful in command substitution
  - Permit group ALPHA to search every directory under /u/godfrey/tools

```
setfacl -m g:ALPHA:r-x $(find /u/godfrey/tools -type d)
```

  - Remove user TED from all ACL entries

```
setfacl -qx u:TED,d:u:TED,f:u:TED $(find / -acl_user TED)
```

  - Add the group ALPHA to every access list in /u/shr/ which contains an entry for UNIXGRP:

```
setfacl -m  g:ALPHA:rwx $(find /u/shr -acl_entry UNIXGRP)
```

Application Programming Interfaces

- Language Environment (LE) provides C services to manipulate ACLs
- REXX provides similar functions
- Low level Logical File System (LFS) interface also available

RACF Access Checking with ACLs

- Takes into account base POSIX permissions and access ACLs

- ACLs only used if the **FSSEC** class is active
  - **SETROPTS CLASSACT(FSSEC)**
    will activate use of ACLs in Unix file authority checks

  - Make sure that FSSEC is **not active** until you are ready to use ACLs
    - The class need not be active to create ACLs

    - Recommendation:
      Migrate **all SYSPLEX nodes** to z/OS V1R3 or later before using ACLs

- **setfacl** can be used to create ACLs at any time

RACF UID / GID management z/OS 1.4

- Prevent shared ID assignment
  - **SHARED.IDS** profile in the UNIXPRIV class (system-wide switch)
  - override with SHARED keyword (e.g. on ADDUSER)
    - RACF admin only

- SEARCH for users with UID / GID

```
SEARCH CLASS(USER) UID(0)
    OMVSKERN
    BPXOINIT
    ELVIS
```

- Automatic UID / GID assignment
  - **AUTOUID** keyword on ADDUSER, ALTUSER

    ```
    ADDUSER MICHELLE OMVS(AUTOUID)
    ```

  - **AUTOGID** keyword on ADDGROUP, ALTGROUP

    ```
    ADDGROUP TESTER OMVS(AUTOGID)
    ```

OpenSSH Secure Shell

- Program Product: **IBM Ported Tools for z/OS**
  - Available May 2004 for installation on z/OS 1.4 and later
  - Non-priced
  - Open Source Software
    - ported, tested, and packaged for z/OS

OpenSSH
Secure Login Across the Wild Wild Web

- OpenSSH **Network connectivity tools** that provide **secure** communications between untrusted hosts over an **insecure** network
  - OpenSSH is common on all major UNIX platforms

OpenSSH Utilities shell commands & daemons

Function	OpenSSH Utility	An alternative to…
Secure remote login	ssh, sshd	rlogin, rsh
Secure file transfer	sftp, sftp-server, scp	rcp

OpenSSH additionally provides these utilities:	
Key management	ssh-keygen, ssh-agent, ssh-add, ssh-keyscan

Session Summary

Hope you learned some useful stuff!

- UNIX shell commands
- System Management Capabilities
- Security

Appendix / References

- z/OS V1R6.0 UNIX System Services Command Reference (SA22-7802-05)
- z/OS V1R6.0 UNIX System Services User's Guide (SA22-7801-05)
- z/OS V1R6.0 UNIX System Services Planning (SA22-7800-05)
- z/OS V1R6.0 UNIX System Services Programming: Assembler Callable Services Reference (SC28-7803-05)
- z/OS V1R6.0 UNIX System Services Messages and Codes (SA22-7807-05)
- z/OS V1R6.0 UNIX System Services Programming Tools (SA22-7805-05)
- z/OS V1R6.0 MVS System Commands (SA22-7627-10)

- From http://www.ibm.com/servers/eserver/zseries/zos/unix/port_tools.html
  - IBM Ported Tools for z/OS Program Directory
  - IBM Ported Tools for z/OS User's Guide