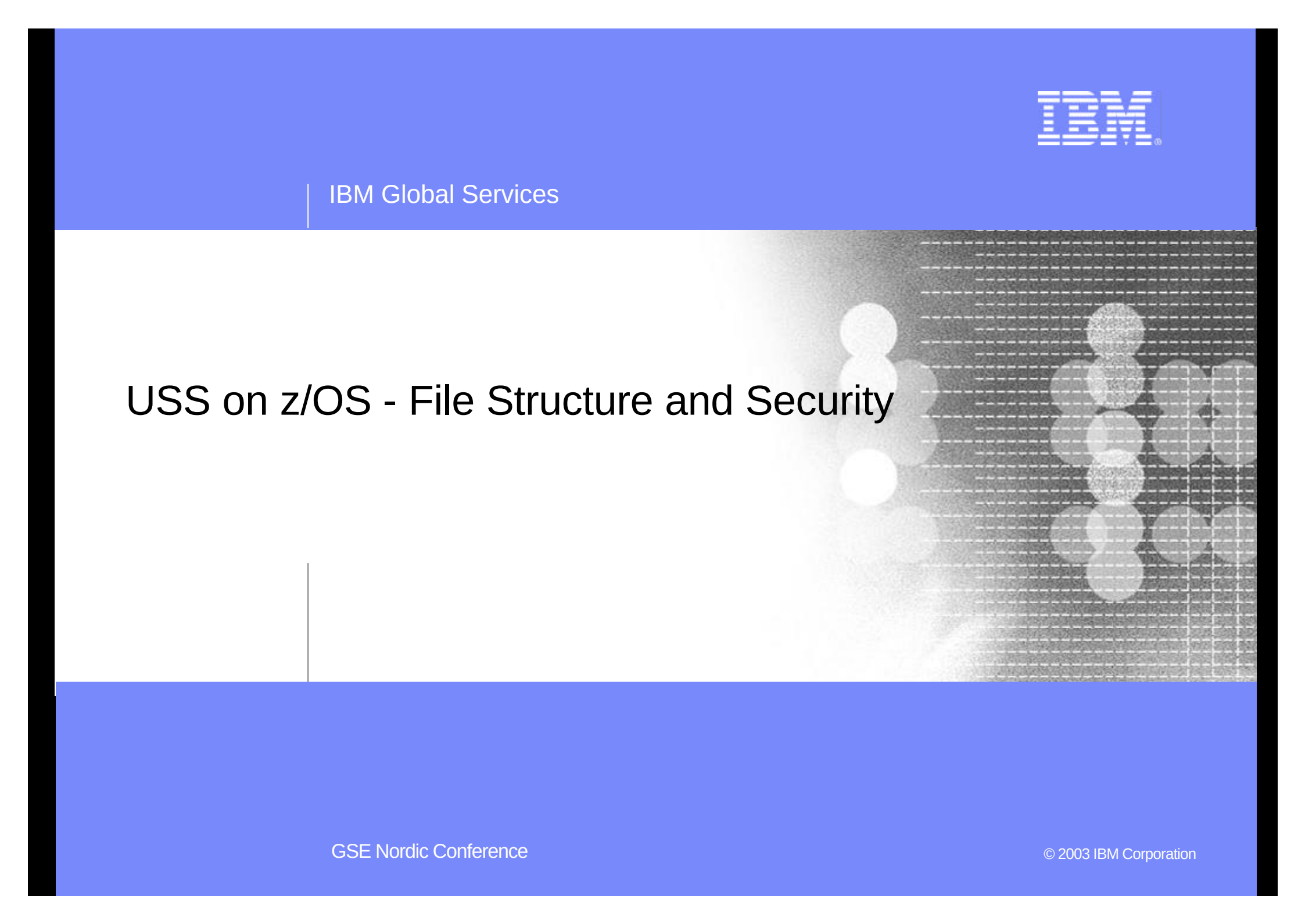
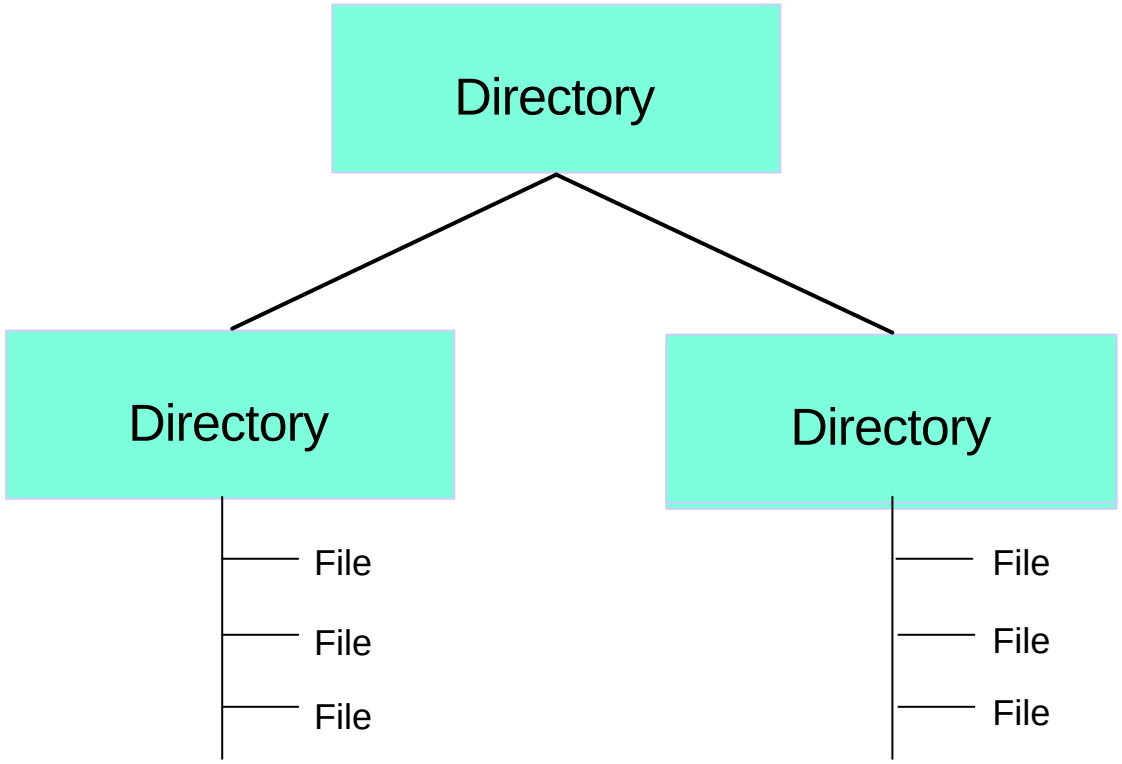
IBM Global Services



USS on z/OS - File Structure and Security

HFS Dataset



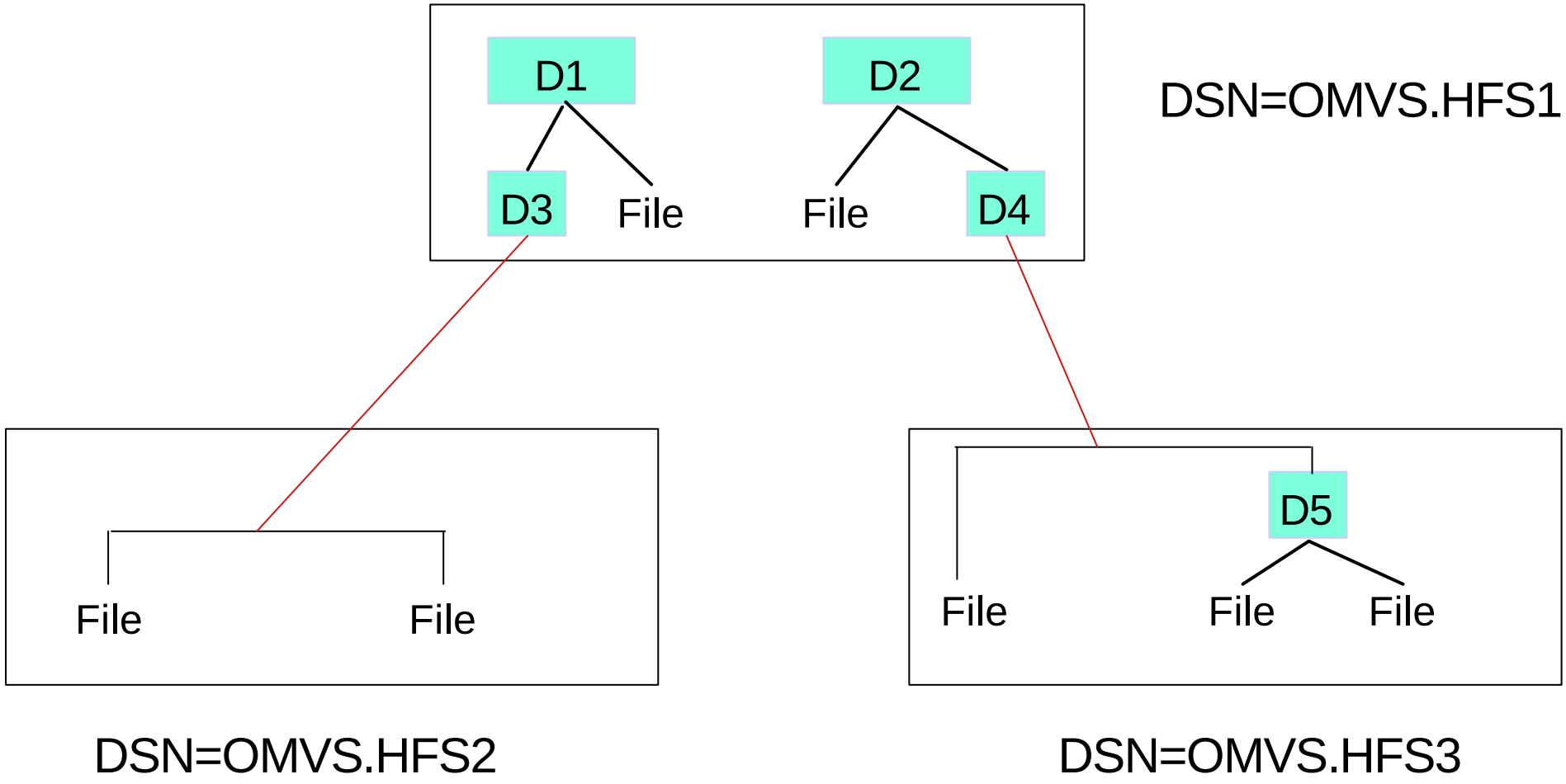
HFS dataset - characteristics

- One - and only one - root point
- Designated by /
- Contains Directories and/or files
- Pathnames and Filenames are case sensitive
- Max. filename is 256 characters
- One logical HFS may be divided into several physical datasets

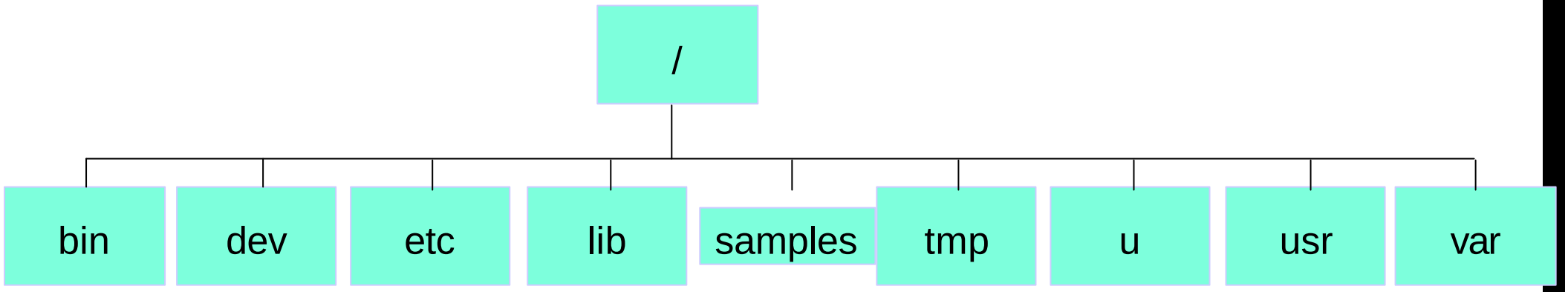
HFS dataset - File types

- Directories
- Regular files (text, executables etc.)
- Named pipes (FIFO)
- Character Special Files (Device drivers)
- Symbolic Links (points to other files)

HFS dataset - logical and physical view



Root System - recommended layout




```
bin     : USS executables
dev     : Device drivers
etc     : Parameter files etc.
lib     : USS support libs
samples: Sample parameter files
tmp     : Temporary work files
u       : Mount point for user HFSs
usr     : Application Services + lpp
var     : Contains log files
```

Bold directories should be contained in seperate HFS datasets

File security in the HFS

- Implemented via FSP (File Security Packet)
  - XPG conforming
- FSP is communicating with External Security i.e. Security Server RACF
- Several RACF classes and profiles involved in USS security.

Users and Groups within USS

Any user, who wants to do USS type of work MUST have a UID defined in OMVS-segment in RACF

Any user, who wants to do USS type of work MUST belong to a default RACF-group, which has a GID defined in OMVS-segment in RACF

UID and GID are numeric values

User types

- Two user types:
  - Superuser - UID = 0
  - Any other user (UID > 0)

- GID has no significance

Superusers have unlimited access to all files in any HFS

Security on the physical dataset level

Since all i/o to and from HFS files is controlled by the OMVS Kernel, only the region owner of the Kernel needs UPDATE access to the HFS datasets.

UACC should be NONE

File Security Packet

- Three levels of access rules
  - Rules for the file owner
  - Rules for any user in the same group as the fileowner
  - Rules for any other user

Every file / directory associated with a UID and GID identifying the file owner

Access rules

- Read
  - Read, print and copy of file allowed
- Write
  - Update and DELETE of file allowed
- Excute
  - Execute allowed

Execute rule applies to all files, but only makes sense when the file is an executable

Filelist - example

```
CCKNUDV:/home/ccknudv: >ls -l

drwxr-xr-x  13 9911312  root         8192 Jan 22  2001 pc-drive
drwxr-x---   2 9911312  root         8192 Aug 28  2001 sandbox
-rwxr-x--x   1 STCMVSG  root        17193 Aug 18  2000 sdsf
```

```
-   : Means that this is a regular file
rwx: Means that file owner has full access
r-x: Means that other members of file owners group may read
     and execute
--x: Means that any other user may execute
```

Permission bits - octal representation

Value	Meaning
0	No access(---)
1	Execute(--x)
2	Write(-w-)
3	Write execute(-wx)
4	Read (r--)
5	Read execute (r-x)
6	Read write (rw-)
7	Read write execute (rwx)

Sticky bit

- On an executable file:
  - Look in LPA or LINKLIST for the module first
- On a directory:
  - Only the directory owner may delete the directory even if others have write access

FSP - extended attributes

Extended attributes are unique to the z/OS implementation of Unix.

Used to mark an executable APF authorized and/or to mark it program controlled

Important RACF profiles in the FACILITY class

- BPX.SUPERUSER
  - Switch between standard user and superuser with the SU command
- BPX.DAEMON
  - Allows daemons to switch to superuser when required
- BPX.FILEATTR.APF
  - allows user to mark an executable APF authorized
- BPX.FILEATTR.PROGCTL
  - allows user to mark an executable Program Controlled

The UNIXPRIV class

SUPERUSER.FILESYS	Allows user to read any HFS file, and to read or search any HFS directory	READ
SUPERUSER.FILESYS	Allows user to write to any HFS file, and includes privileges of READ access	UPDATE
SUPERUSER.FILESYS	Allows user to write to any HFS directory, and includes privileges of UPDATE access	CONTROL
SUPERUSER.QUIESCE	Allows user to quiesce and unquiesce a file system with the nosetuid option	READ
SUPERUSER.QUIESCE	Allows user to quiesce and unquiesce a file system with the setuid option	UPDATE
SUPERUSER.MOUNT	Allows user to mount and unmount a file system with the nosetuid option	READ
SUPERUSER.MOUNT	Allows user to mount and unmount a file system with the setuid option	UPDATE

Backup/recovery strategies

- DFSMS Backup/restore on the physical dataset level
  - Copy operations are not possible
- Tivoli Storage Manger (TSM) is recommended for backup/restore at the file/directory level
- CPIO, TAR and PAX are Unix programs used to build backups of files/directories
  - Similar to the ZIP on the Windows platform

Symbolic links - important to understand

A symbolic link is like an ALIAS. It is a file pointing to another file

Example: Symbolic link /u/user1/myfile points to /u/user2/text.file

When user1 edits myfile he/she actually edits user2's text.file

Permissions are granted from the real file.

The symbolic link will always have RWX on all levels

Setting up shared HFSs in a //SYSPLEX

- Create BPX Couple Dataset
- Update COUPLExx member in PARMLIB
- Update BPXPRMxx member in PARMLIB
- Allocation of HFS datasets
- Definition of symbolic links
- Create MOUNT statements

Create BPX Couple Dataset

```
//STEP1    EXEC PGM=IXCL1DSU
//STEPLIB  DD  DSN=SYS1.MIGLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
     DEFINEDS SYSPLEX(TOYPLEX)
              DSN(TOYPLEX.HFSCOUP.DS1)
              VOLSER(SHAR00)
              MAXSYSTEM(8)
              CATALOG
            DATA TYPE(BPXMCDS)
                ITEM NAME(MOUNTS)   NUMBER(500)
                ITEM NAME(AMTRULES) NUMBER(50)
```

Update Couple-member in Parmlib

```
DATA    TYPE(BPXMCDS)
        PCOUPLE(&SYSPLEX..HFSCOUP.DS1,SHAR00)
        ACOUPLE(&SYSPLEX..HFSCOUP.DS2,SHAR00)
```

Update BPXPRM member in Parmlib

SYSPLEX(YES)

- Can NOT be changed dynamically

Version(ZOS14)

- Can NOT be changed dynamically

Symbolic variables

ÅSYSNAME:

- Replaced by SYSNAME from IEASYSxx

ÅVERSION:

- Replaced by name from BPXPRMxx VERSION(....) statement

Allocation of HFSs

- SYSPLEX-wide root file system
  - Small (1 cylinder) - contains only symbolic links and mountpoints
- Image-specific file system
  - Small (1 cylinder) - contains only symbolic links and mountpoints
- Version-specific root file system
  - Large - contains everything delivered in ServerPack

SYSPLEX-wide root

```
/bin      ----> /ÅVERSION/bin
/lib      ----> /ÅVERSION/lib
/samples  ----> /ÅVERSION/samples
/usr      ----> /ÅVERSION/usr
/dev      ----> /ÅSYSNAME/dev
/etc      ----> /ÅSYSNAME/etc
/tmp      ----> /ÅSYSNAME/tmp
/var      ----> /ÅSYSNAME/var
```

SYSPLEX-wide root

```
/u          ---> Mountpoint for user HFSs
/ÅSYSNAME   ---> Mountpoint for image-
                 specific HFS
/ÅVERSION   ---> Mountpoint for Version
                 root

ÅSYSNAME will be replaced by SYSNAME
and ÅVERSION will be replaced by
VERSION when OMVS starts
```

Image-specific HFS

```
/etc      ----> Mountpoint for etc HFS
/dev      ----> Mountpoint for dev HFS
/tmp      ----> Mountpoint for tmp HFS
/var      ----> Mountpoint for var HFS
/bin      ----> /ÅVERSION/bin
/lib      ----> /ÅVERSION/lib
/samples  ----> /ÅVERSION/samples
/usr      ----> /ÅVERSION/usr
```

Version specific root

```
/etc      ----> /ÅSYSNAME/etc
/dev      ----> /ÅSYSNAME/dev
/tmp      ----> /ÅSYSNAME/tmp
/var      ----> /ÅSYSNAME/var
/bin      ----> Contains USS files & dirs
/lib      ----> Contains USS files & dirs
/samples  ----> Contains USS files & dirs
/usr      ----> Contains USS files & dirs
```

Automove in a SYSPLEX

When a member in a SYSPLEX is brought down it is possible for another member of the SYSPLEX to take over the mounted HFS.

Specify AUTOMOVE on the MOUNT command
The SYSPLEX member that is taking over is selected randomly

The NOAUTOMOVE parameter

- Specifying the NOAUTOMOVE paremter on the MOUNT command prevents taking over
- Should be specified for image specific HFS

Parameter files in /etc, temporary files in /tmp and logs in /var are considered image specific.

Mount statements in BPXPRMxx

```
ROOT FILESYSTEM('OMVS.&SYSPLEX..ROOT')
     TYPE(HFS)
     MODE(RDWR)

MOUNT FILESYSTEM('OMVS.&SYSNAME..HFS')
      TYPE(HFS)
      MOUNTPOINT('/&SYSNAME.')
      MODE(RDWR)
      NOAUTOMOVE

MOUNT FILESYSTEM('OMVS.&VERSION..ROOT')
      TYPE(HFS)
      MOUNTPOINT('/&VERSION.')
      MODE(READ)
```

Other exploiters of USS

- Other non-z/OS Serverpack exploiters of USS:
  - CICS Transaction Server
  - DB2
  - WebsPhere Application Server
  - Lotus Domino
  - On Demand
  - Enterprise compilers (Cobol, PL/1)
  - and many others .....

Mountpoint /usr/lpp

All these exploiters will typically provide an HFS during installation.

The default mountpoint is:

- /usr/lpp/cics
- /usr/lpp/db2
- /usr/lpp/lotus

etc.

/usr/lpp is the focal point. lpp means Licensed Program Products

Mountpoint /SERVICE

- To service an HFS (i.e. by SMP/E)
  - Create /SERVICE mountpoint on root
  - Mount a copy of the HFS on /SERVICE
  - Point DDDEFs to /SERVICE/.....

JCL support for HFS

- DD PATH=
  - Points to an HFS file or directory
- PATHOPTS=
  - Specifies the open options (read, write, r/w)
- PATHMODE=
  - Permissions given to a new file/directory

SMP/E supports PATH in DDDEF

JCL support for HFS - an example

```
//TESTUNIX EXEC PGM=BPXBATCH,PARM='SH /u/cckvr/testscript.sh'
//STDOUT   DD  PATH='/u/cckvr/testrun.output',
//             PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//             PATHMODE=SIRWXU
//STDERR   DD  PATH='/u/cckvr/testrun.error',
//             PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//             PATHMODE=SIRWXU
//
//STDENV   DD  PATH='/u/cckvr/.profile',PATHOPTS=ORDONLY
//
```

ZFS - the new file system

- Why use ZFS ?
  - Better i/o performance for files > 8K
  - Asynchronous access
  - Logs for recovering data
  - Space sharing
  - Cloning

What is ZFS ?

- Similar to HFS in the way you access files from USS
- Not a replacement for HFS.
  - Root filesystem must still be HFS
- Introduced in OS/390 2.10 by a PTF
- Physically defined as a VSAM Linear Dataset

Defining a ZFS

- A VSAM Linear dataset is defined using IDCAMS
- VSAM dataset formatted using IOEAGFMT
- ZFS(s) are defined within the VSAM dataset using /usr/lpp/dfs/global/bin/zfsadm

If one ZFS is defined in a VSAM dataset having the same name the ZFS is a nonaggregate

If several ZFSs are defined in a VSAM dataset the ZFS is an aggregate

Space sharing in a ZFS aggregate

Defining several ZFSs in an aggregate VSAM enables space sharing between the ZFSs.

Space freed in one ZFS (by file deletion) becomes available to the others.

Upper limit for space consumption set by parameter

ZFS definition - an example

```
//STEP01   EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
         DEL   OMVS.MVSL.ZFS CL PRG

         SET   MAXCC = 0

         DEF   CL                        -
               (                         -
               NAME(OMVS.MVSL.ZFS)       -
               LINEAR                    -
               CYL(500 0)                -
               SHR(2,3)                  -
               )                         -
           DATA                          -
               (                         -
               NAME(OMVS.MVSL.ZFS.DATA)  -
               )
/*
//* ------------------------------------------
//STEP02   EXEC PGM=IOEAGFMT,
//             PARM=('-aggregate OMVS.MVSL.ZFS')
//SYSPRINT DD  SYSOUT=*
//STDOUT   DD  SYSOUT=*
//STDERR   DD  SYSOUT=*
```

ZFS definition - an example

```
//DEFZFS   EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//*
//SYSTSIN  DD  *
         bpxbatch SH                                          +
               /usr/lpp/dfs/global/bin/zfsadm create          +
                      -filesystem OMVS.MVSL.ZFS1              +
                      -aggregate  OMVS.MVSL.ZFS               +
                      -size 22500
         bpxbatch SH                                          +
               /usr/lpp/dfs/global/bin/zfsadm create          +
                      -filesystem OMVS.MVSL.ZFS2              +
                      -aggregate  OMVS.MVSL.ZFS               +
                      -size 22400
/*
```

Size is calculated in 8K blocks

Cloning

- Possible to establish a readonly clone of a ZFS within the same aggregate
- Fast - only metadata is copied
- Will not work in a //SYSPLEX environment

New ZFS Address Space

All i/o to HFSs is managed by OMVS kernel

- but

a new ZFS Address Space (default name ZFS) is managing i/o to all ZFSs

ZFS is a restartable task

ZFS File System Characteristics

- **zFS Multi-filesystem aggregates**
  - zFS can put more than one mountable file system in a dataset. These datasets are called "**aggregates**"
    - Aggregate must be Attached (zFSAdmin command)
      - zFS opens the dataset and learns the names of all file systems that it contains.
      - These lists of names are used during mounts to find the aggregate
    - MOUNT FILESYSTEM() name can be different from the dataset name (always the same name for HFS)
    - Due to the characteristics of multi-filesystem aggregates, there are currently some restrictions

ZFS File System Characteristics

- **zFS HFS Compatible filesystems** - Only one file system defined inside the aggregate and the file system name is same as the aggregate name
  - Can be mounted without being explicitly attached, since dataset name = file system name, and zFS implicitly attaches during mount
  - Use of HFS Compatible filesystems removes most of the restrictions that apply to multi-filesystem aggregates
  - Documentation will advise HFS Compatibility Mode until there is more support for Sysplex built into zFS

ZFS - //SYSPLEX restrictions

- Sysplex restrictions
  - All filesystems in an aggregate must be mounted from the same system because attach (R/W) can only be done one system at a time. Mounts from other systems will fail.
  - Move is unsupported
  - If the owning system dies, the file system cannot be AUTOMOVEd because the aggregate will not have been attached on any of the other systems. They will become unowned.

News in z/OS 1.4

- AUTOMOVE system list
- ZFS enhancements
- Automount enhancements

Automove System List Overview

- **Problem**:
  - In Shared HFS environment, AUTOMOVE(YES) on MOUNT will move the ownership of the filesystem to some other system in the sysplex if the current server system for that filesystem is brought down
  - Which system will become the new server is random
  - Customers have requested the capability to specify which system or systems in a sysplex will takeover as server for a filesystem.

Automove System List Overview

- **Solution:**
  - A system list will be allowed on the AUTOMOVE parameter for MOUNT
    - List will begin with either i or e (indicating include or exclude), followed by a list of system names.
    - If include, the list of systems is in priority order.
    - If none of the systems specified in the list can takeover as server, the filesystem will be unmounted

Automove System List Overview

- Using Automove System List, the customer can:
  - Provide predictability in which systems will takeover a given filesystem.
  - Be able to exclude certain systems from taking over a given filesystem.
- Advantages:
  - Permits predictability, if desired, for workload balancing/performance.

Manipulating Automove System List

- AUTOMOVE system list can be specified on all methods of mounting a filesystem: parmlib, tso command, shell command, ishell, C program, assembler and REXX.
- The AUTOMOVE option or system list can be changed for a filesystem after mounted using:
  - chmount shell command
  - SETOMVS system command
- Display commands that display the system list:
  - DISPLAY OMVS,FILESYSTEM
  - df -v
  - MODIFY BPXOINIT,FILESYS=DISPLAY,ALL

ZFS enhancements - ISHELL support

ISHELL

A new option will be available under the File_systems pull-down to create a new zFS HFS compatible filesystem:

File_systems Options Setup

1. Mount table...
2. New HFS ...
3. Mount(O)...
4. **New ZFS ...**

ZFS enhancements - ISHELL support

ISHELL panel

File Directory Special_file Tools File_systems Options Setup Help

Create a zFS File System

Enter the fields as required then press Enter.

File system name . . . . _____
Owning User . . . . . _____ (Number or user name)
Owning Group . . . . . _____ (Number or group name)
Permissions . . . . . ___ (3 digits, each 0-7)
Primary cylinders . . . . _____
Secondary cylinders . . . _____
Storage class . . . . . _____
Management class . . . . _____
Data class . . . . . _____
Volume names . . . . . _____ _____ _____ _____ _____
_____ _____ _____ _____ _____
_____ _____ _____ _____ _____

F1=Help F3=Exit F6=Keyshelp F12=Cancel

AUTOMOUNT enhancements

Symbol substition is enabled.

Like in PARMLIB:

- &SYSPLEX
- &SYSNAME
- &SYSCLONE

etc.....