UNIX System Services
Introduction

Nat Stevenson
Washington Systems
Gaithersburg, MD

AGENDA

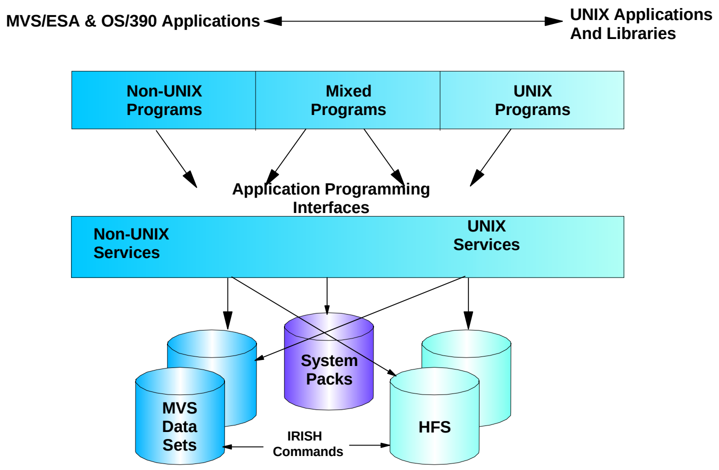
UNIX on z/OS Overview

HFS & ZFS Overview

AGENDA

UNIX on z/OS Overview

The Hidden Gem In OS/390



MVS/ESA & OS/390 Applications

UNIX Applications And Libraries

Non-UNIX Programs

Mixed Programs

UNIX Programs

Application Programming Interfaces

Non-UNIX Services

UNIX Services

MVS Data Sets

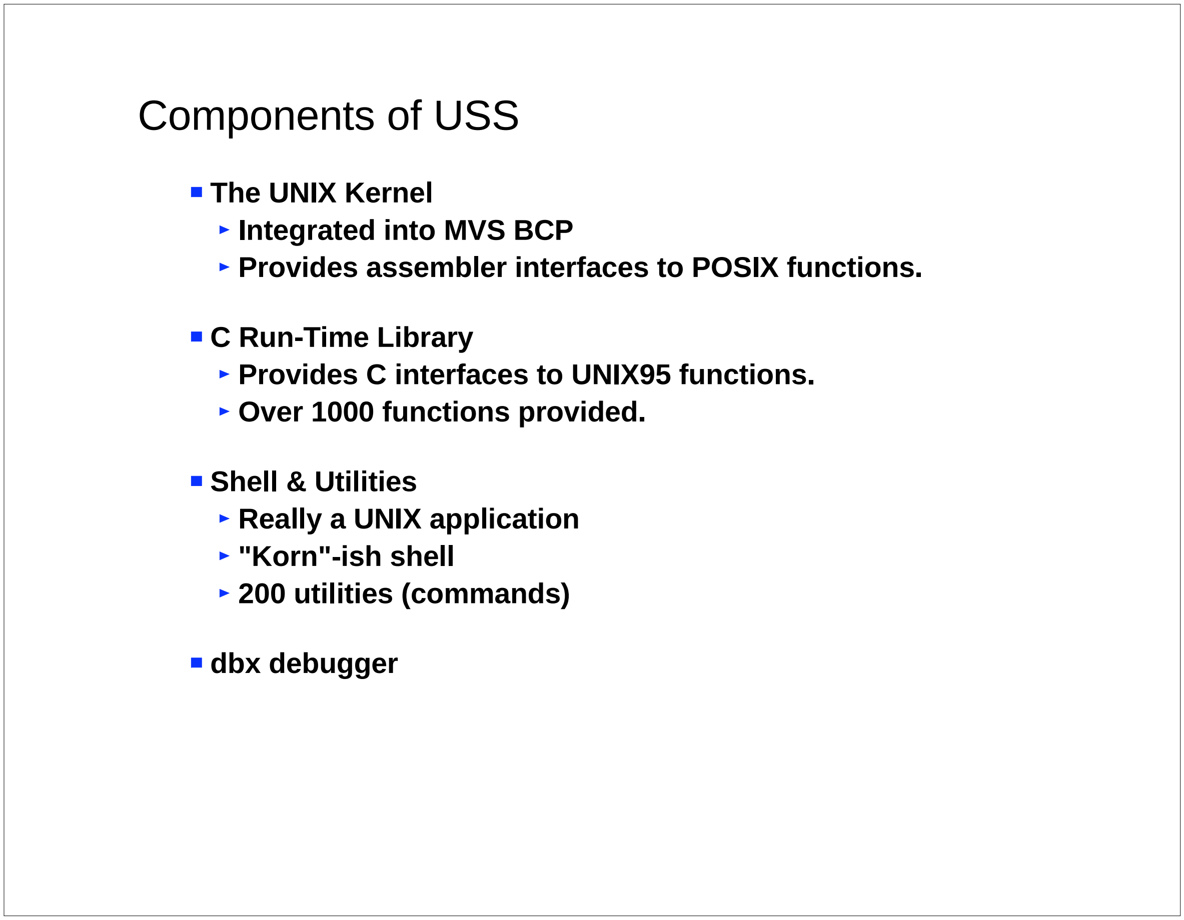
System Packs

HFS

IRISH Commands

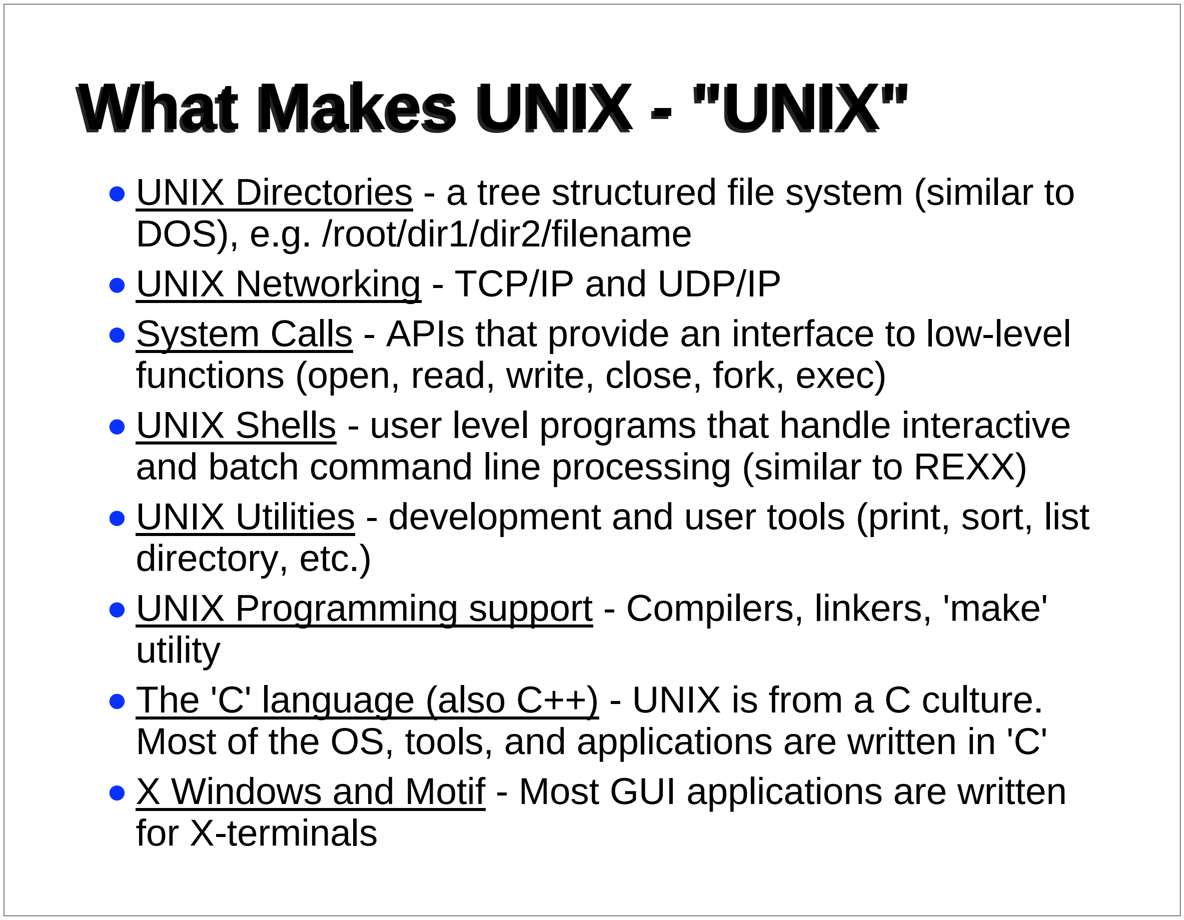
Application and skills portability

* UNIX functions defined by the Single UNIX Specification from X/Open



Components of USS

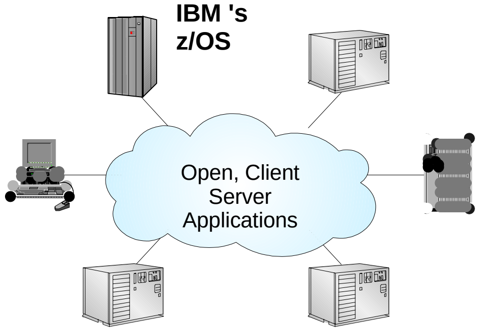
- The UNIX Kernel
  - Integrated into MVS BCP
  - Provides assembler interfaces to POSIX functions.
- C Run-Time Library
  - Provides C interfaces to UNIX95 functions.
  - Over 1000 functions provided.
- Shell & Utilities
  - Really a UNIX application
  - "Korn"-ish shell
  - 200 utilities (commands)
- dbx debugger



What Makes UNIX - "UNIX"

- UNIX Directories - a tree structured file system (similar to DOS), e.g. /root/dir1/dir2/filename
- UNIX Networking - TCP/IP and UDP/IP
- System Calls - APIs that provide an interface to low-level functions (open, read, write, close, fork, exec)
- UNIX Shells - user level programs that handle interactive and batch command line processing (similar to REXX)
- UNIX Utilities - development and user tools (print, sort, list directory, etc.)
- UNIX Programming support - Compilers, linkers, 'make' utility
- The 'C' language (also C++) - UNIX is from a C culture. Most of the OS, tools, and applications are written in 'C'
- X Windows and Motif - Most GUI applications are written for X-terminals

Why Integrate UNIX Into MVS?



Business requirements

- Application and skills portability
- Interoperability across heterogeneous systems and networks
- Transparent access and management of heterogeneous network databases
- Overall architecture for enterprise open systems computing
- Investment protection

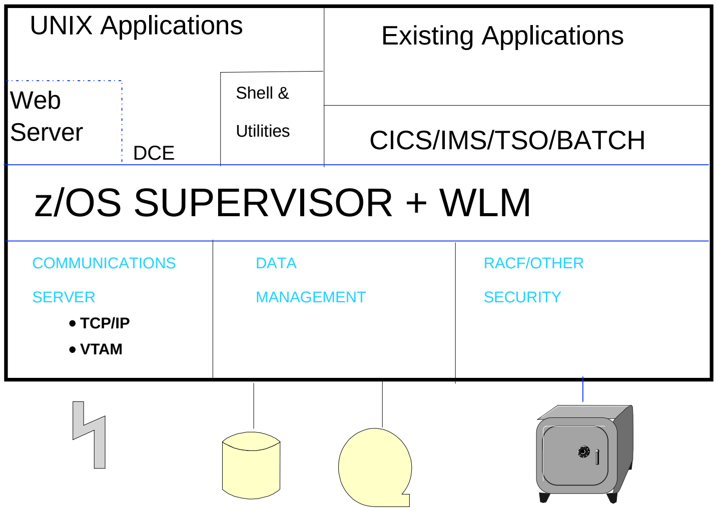
MVS participates in a multivendor environment with

- UNIX application portability
- Multivendor application interoperability

→ z/OS

Efficiency in a heterogeneous environment

z/OS UNIX System Environment



UNIX Applications

Existing Applications

Web Server

DCE

Shell & Utilities

CICS/IMS/TSO/BATCH

z/OS SUPERVISOR + WLM

COMMUNICATIONS SERVER
- TCP/IP
- VTAM

DATA MANAGEMENT

RACF/OTHER SECURITY

Unix Services are integrated at a low level It is NOT

– A Unix emulator

– A separate part of the system away from the TSO/batch Environment

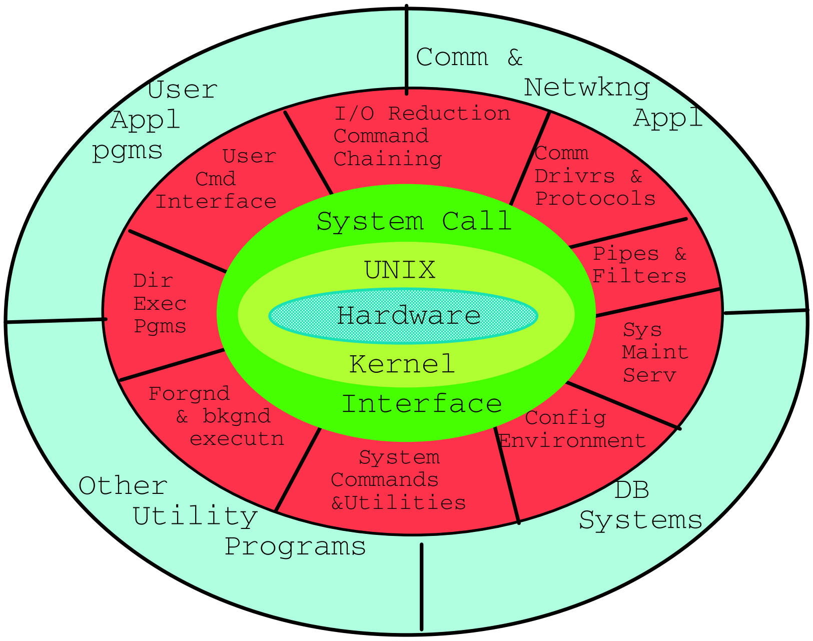
UNIX System Services on z/OS

- Required z/OS Subsystems
  - DFSMS (to handle UNIX file structure)
  - WLM (to handle UNIX process creation)
  - RACF/ACF2/TopSecret (to handle UNIX security)
- Communication
  - TCP/IP
- Misc z/OS Components
  - TSO - ISPF (ISHELL)
  - WLM (in Goal Mode)
  - ...

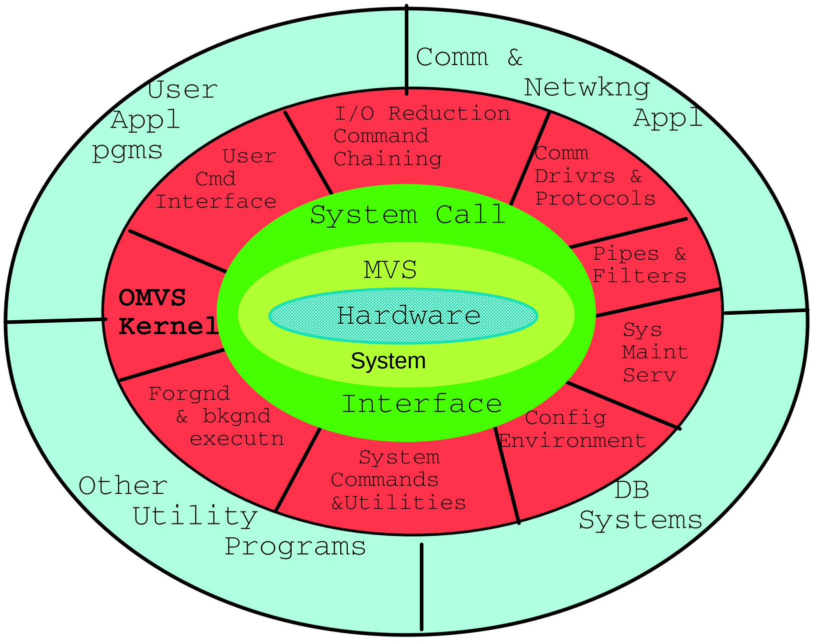
UNIX - More Than a Shell

- No wall between MVS and UNIX
  - Unix System Services
  - Language Environment
  - POSIX (bash) shell
- More than TCP/IP
  - Pipes
  - Inter Process Communication
  - Thread processing
- Rapid development with powerful constructs

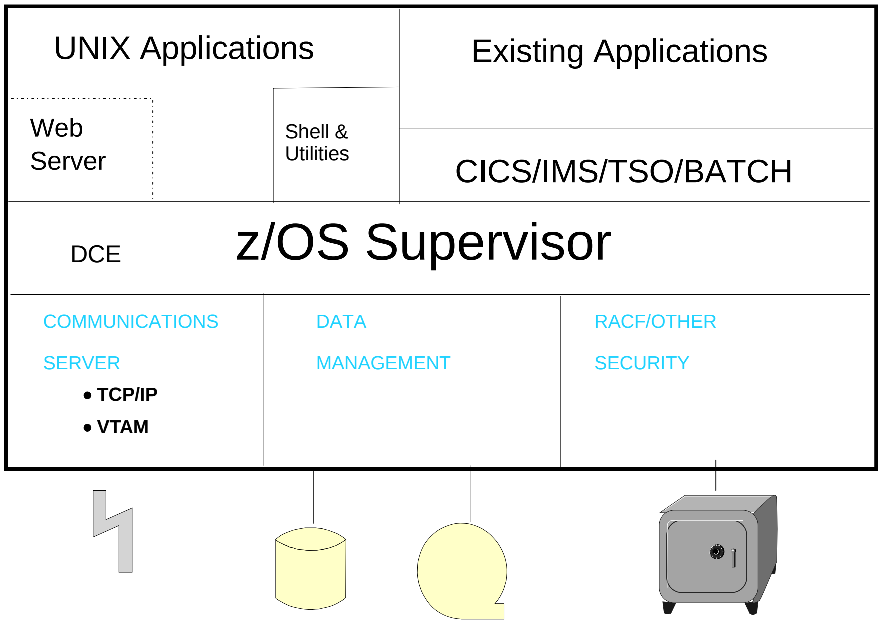
A UNIX System



A UNIX System



z/OS UNIX System Environment



UNIX Applications

Existing Applications

Web Server

Shell & Utilities

CICS/IMS/TSO/BATCH

DCE

z/OS Supervisor

COMMUNICATIONS SERVER

- TCP/IP
- VTAM

DATA MANAGEMENT

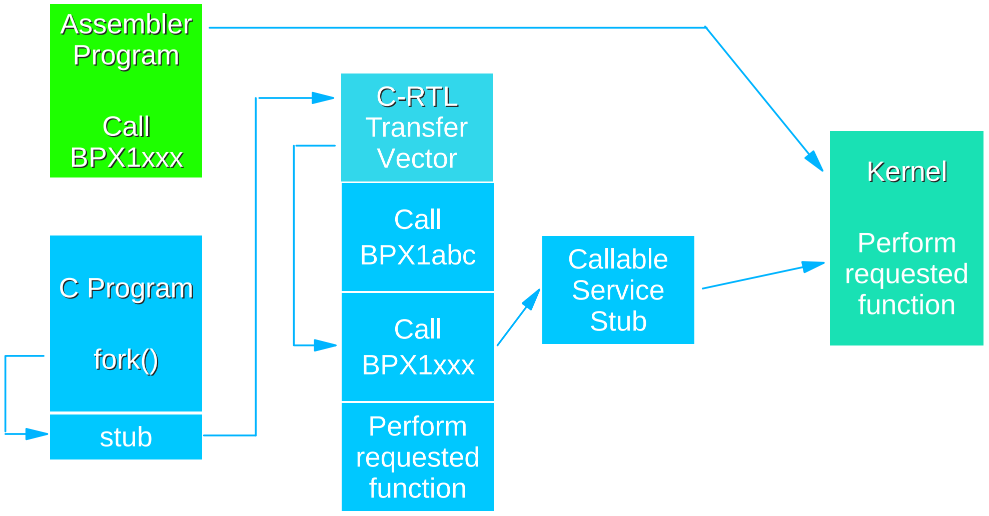
RACF/OTHER SECURITY

OEPG0094

USS / TSO Support

- Examples of using USS support
  - How the services are invoked
  - Programming in REXX
  - Using in a batch process
  - TSO Commands
  - Interprocess Communication
  - Required File Access

Using UNIX Services



Assembler Program

Call BPX1xxx

C Program

fork()

stub

C-RTL Transfer Vector

Call BPX1abc

Call BPX1xxx

Perform requested function

Callable Service Stub

Kernel

Perform requested function

REXX Support

TSO, BATCH

```
/* REXX */
ADDRESS SYSCALL
```

EXECIO - Access to HFS using file descriptors.

SPAWN - Create new USS process and exec a new program or REXX exec residing in the HFS.

"Traditional" REXX programs can now call UNIX services.

Shell

```
/* REXX */
```

ADDRESS SH - Access to shell & utilities. REXX support remaps file descriptors.

SAY 'Hello World' - Mapped to STDOUT (fd=1)

PARSE EXTERNAL - Mapped to STDIN (fd=0)

EXECIO - Access to HFS.

"Shell" REXX programs can mix shell commands and REXX, and pipe output between them.

Batch

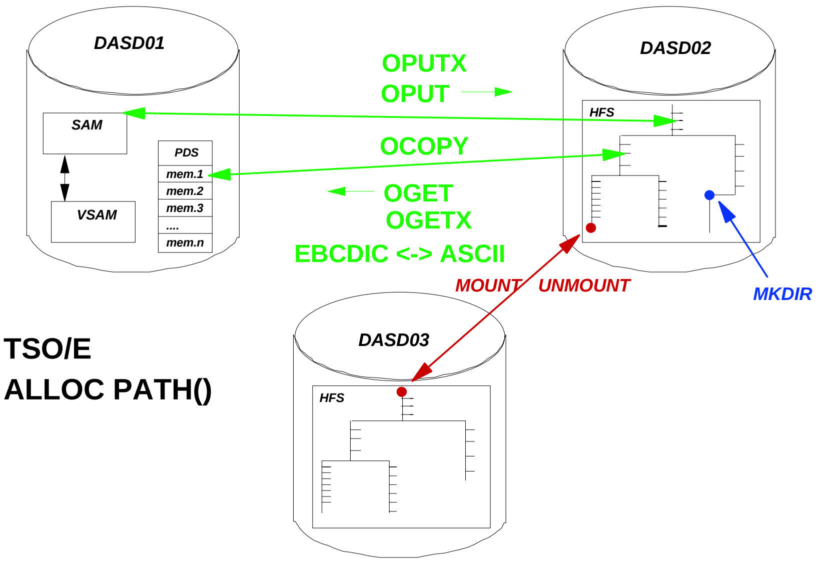
BPXBATCH -- Execute a program in the HFS. Invoked via:

- JCL EXEC statement.

```
//jobname JOB ...
//stepname EXEC PGM=BPXBATCH,PARM='SH ps
-ej'
//STDOUT DD PATH='stdout file path'
//STDERR DD PATH='stderr file path'
```

- TSO CALL command.
- As a TSO command. `BPXBATCH SH ps -ej`
- CLISTS or REXX Execs.
- OSHELL command. `oshell ps -ej`

TSO COMMANDS



TSO/E
ALLOC PATH()

OGETX/OPUTX REXX execs do PDS and directories

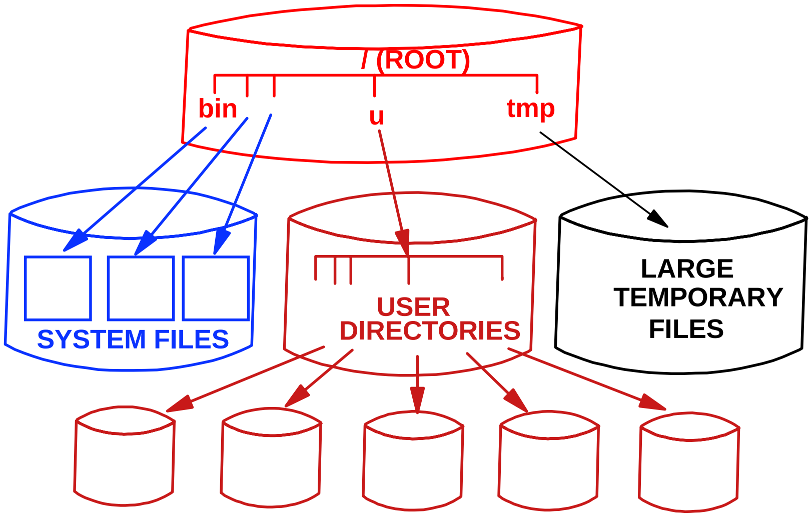
Interprocess Communication

- Shared Memory
  - Ideal for large data
  - Avoids data movement
  - ESQA overhead to map storage
- Semaphores
  - Generally used to serialize shared memory
- Named Pipes
  - First-in first-out Queue
  - Intended for point to point
  - Can have multiple readers and writers
- Message Queues
  - Strength in n to 1 or n to m communications
  - Ideal for small messages feeding a server
- Signals
  - Basis for error handling
  - Can be from system events or application programming
- Local Sockets
  - Used by servers that have the option of local or remote clients

AGENDA

HFS & ZFS Overview

HFS DATA SET ORGANIZATION



SEPARATE HFS FILES FOR USERS OR GROUPS

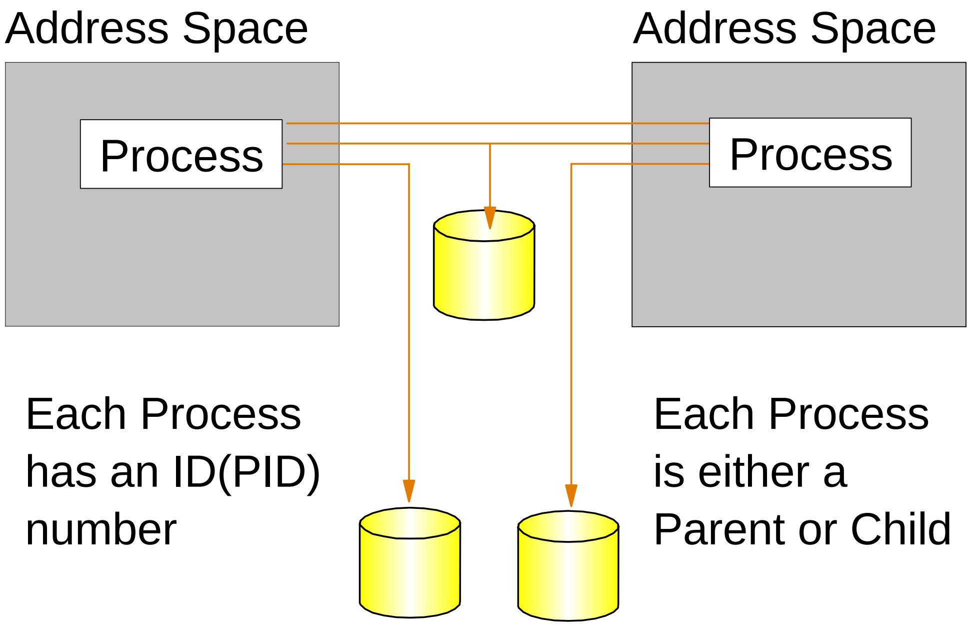
USE AUTOMOUNT TO MOUNT ON DEMAND

HFS Data Set Considerations

- ★ Must be SMS managed
- ★ Allocated like a PDSE using DSNTYPE=HFS
- ★ Up to 123 extents (keep to 1 extent)
- ★ Single Volume
- ★ May be shared by multiple read-only systems
- ★ Maximum file size is 2GB
- ★ Cannot be mounted at multiple mount points

alloc da('hfs.filesys') dsntype(hfs) space(10,10) dsorg(po) cyl

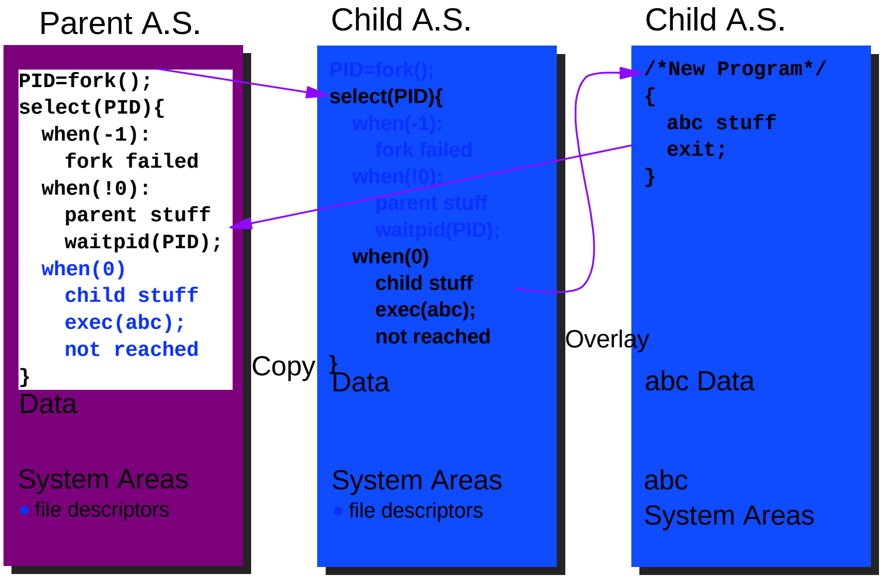
UNIX Execution Control



Each Process has an ID(PID) number

Each Process is either a Parent or Child

Fork and Exec



```
PID=fork();
select(PID){
  when(-1):
    fork failed
  when(!0):
    parent stuff
    waitpid(PID);
  when(0)
    child stuff
    exec(abc);
    not reached
}
```

Data

System Areas
- file descriptors

```
PID=fork();
select(PID){
  when(-1):
    fork failed
  when(!0):
    parent stuff
    waitpid(PID);
  when(0)
    child stuff
    exec(abc);
    not reached
}
```

Data

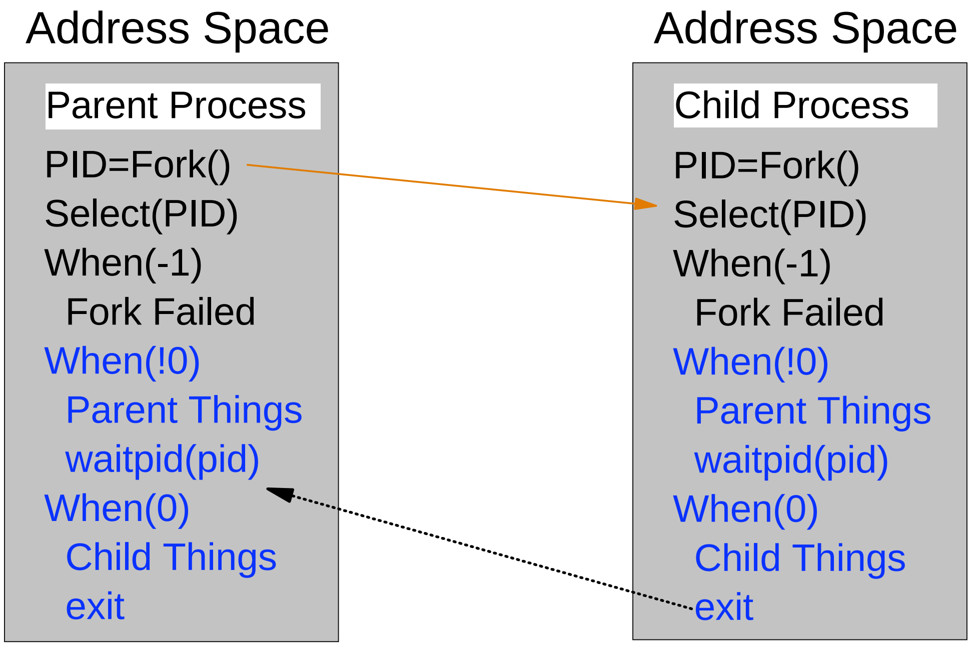
System Areas
- file descriptors

```
/*New Program*/
{
  abc stuff
  exit;
}
```

abc Data

abc
System Areas

Fork: Creating a New Process



Address Space

Parent Process

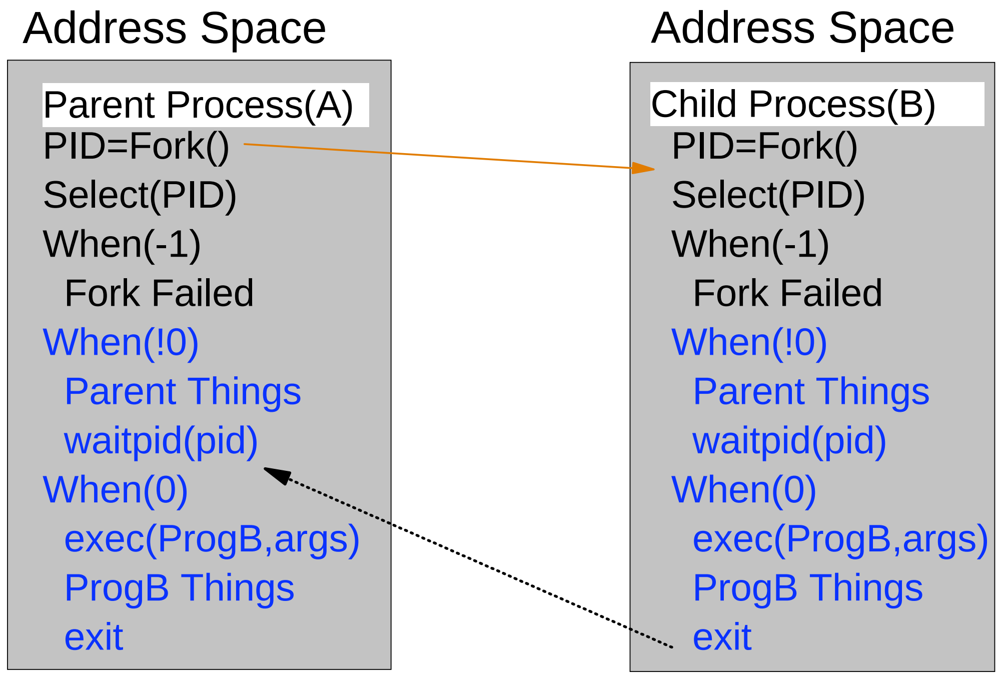
PID=Fork()
Select(PID)
When(-1)
  Fork Failed
When(!0)
  Parent Things
  waitpid(pid)
When(0)
  Child Things
  exit

Address Space

Child Process

PID=Fork()
Select(PID)
When(-1)
  Fork Failed
When(!0)
  Parent Things
  waitpid(pid)
When(0)
  Child Things
  exit

Fork: Executing A New Program



Address Space

Parent Process(A)

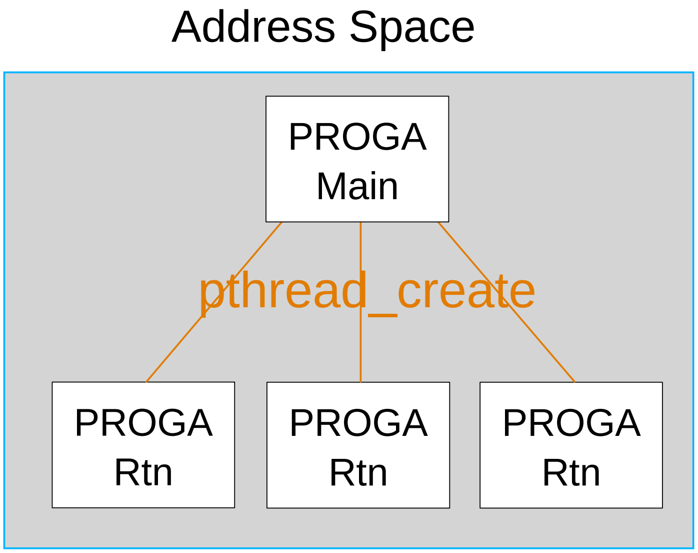
PID=Fork()
Select(PID)
When(-1)
  Fork Failed
When(!0)
  Parent Things
  waitpid(pid)
When(0)
  exec(ProgB,args)
  ProgB Things
  exit

Address Space

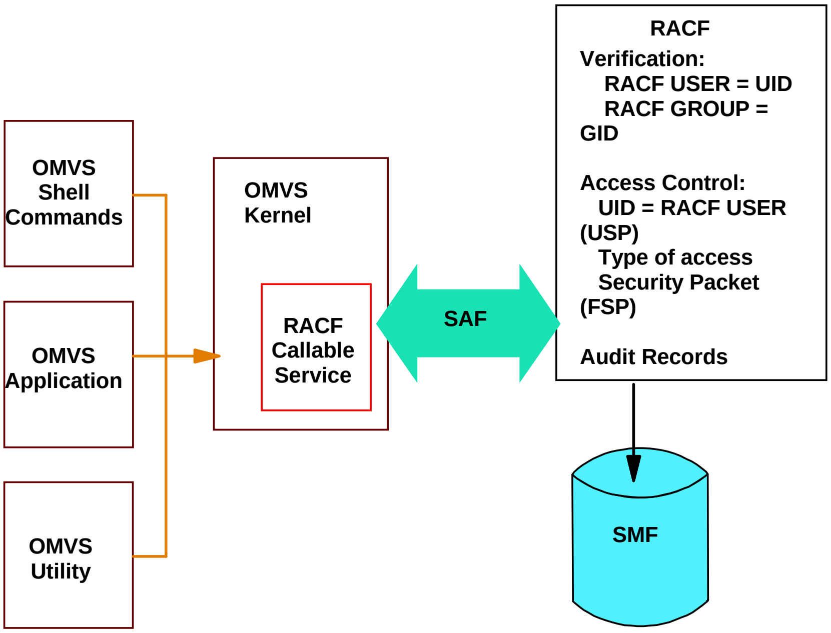
Child Process(B)

PID=Fork()
Select(PID)
When(-1)
  Fork Failed
When(!0)
  Parent Things
  waitpid(pid)
When(0)
  exec(ProgB,args)
  ProgB Things
  exit

UNIX System Services Thread Model



RACF - Unix System Services Interface



RACF

Verification:
RACF USER = UID
RACF GROUP = GID

Access Control:
UID = RACF USER (USP)
Type of access
Security Packet (FSP)

Audit Records

User / File Security

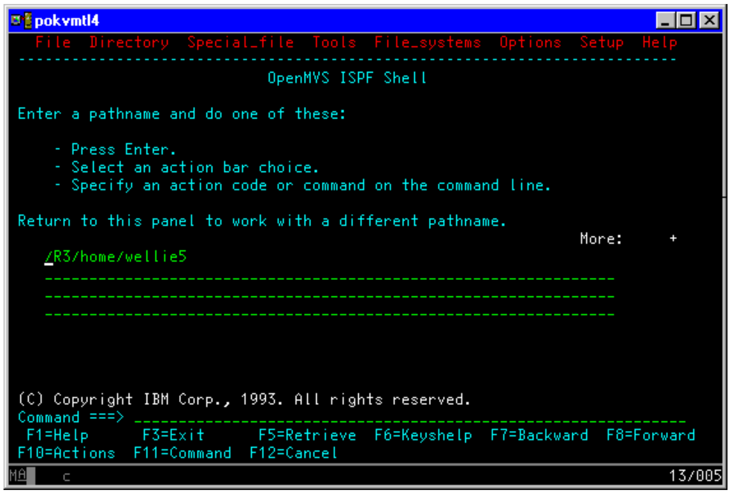
Superuser is able to access any file in the system. UID=0

User is able to access own and authorized files. UID>0

User within Group is able to access authorized files. GID

Using ISHELL to ease into the Shell

ISHELL is not the shell. It is a TSO program that gives you an ISPF interface to work with HFS files.

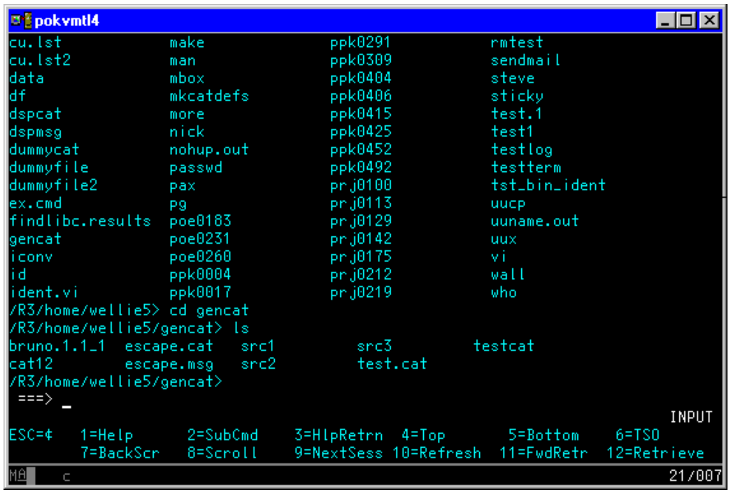


pokvmtl4
File Directory Special_file Tools File_systems Options Setup Help

```
                            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:     +
   /R3/home/wellie5
   ________________________________________________________________
   ________________________________________________________________
   ________________________________________________________________




(C) Copyright IBM Corp., 1993. All rights reserved.
Command ===> ______________________________________________________________
 F1=Help      F3=Exit      F5=Retrieve  F6=Keyshelp  F7=Backward  F8=Forward
F10=Actions  F11=Command  F12=Cancel
MA      c                                                             13/005
```

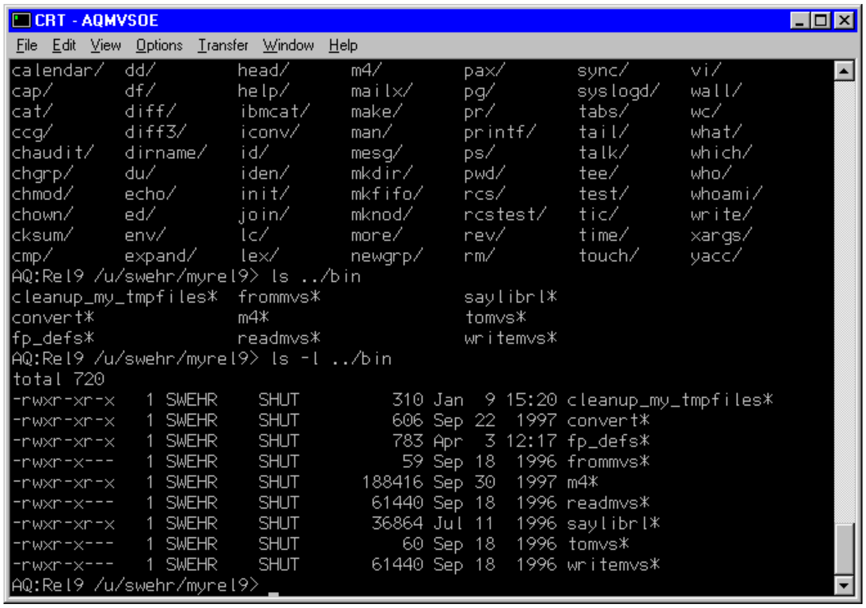
How do I get to the real shell

OMVS - familiar ISPF interface.



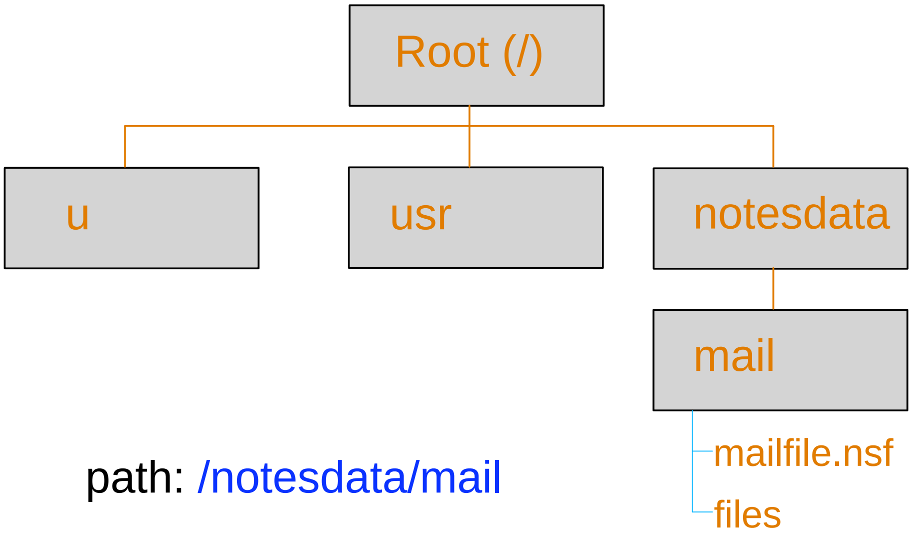
```
cu.lst              make                ppk0291             rmtest
cu.lst2             man                 ppk0309             sendmail
data                mbox                ppk0404             steve
df                  mkcatdefs           ppk0406             sticky
dspcat              more                ppk0415             test.1
dspmsg              nick                ppk0425             test1
dummycat            nohup.out           ppk0452             testlog
dummyfile           passwd              ppk0492             testterm
dummyfile2          pax                 prj0100             tst_bin_ident
ex.cmd              pg                  prj0113             uucp
findlibc.results    poe0183             prj0129             uuname.out
gencat              poe0231             prj0142             uux
iconv               poe0260             prj0175             vi
id                  ppk0004             prj0212             wall
ident.vi            ppk0017             prj0219             who
/R3/home/wellie5> cd gencat
/R3/home/wellie5/gencat> ls
bruno.1.1_1  escape.cat  src1          src3          testcat
cat12        escape.msg  src2          test.cat
/R3/home/wellie5/gencat>
 ===> _
                                                                       INPUT
ESC=¢   1=Help      2=SubCmd    3=HlpRetrn  4=Top       5=Bottom    6=TSO
        7=BackScr   8=Scroll    9=NextSess 10=Refresh  11=FwdRetr  12=Retrieve
MA     c                                                                21/007
```

How do I get to the real shell



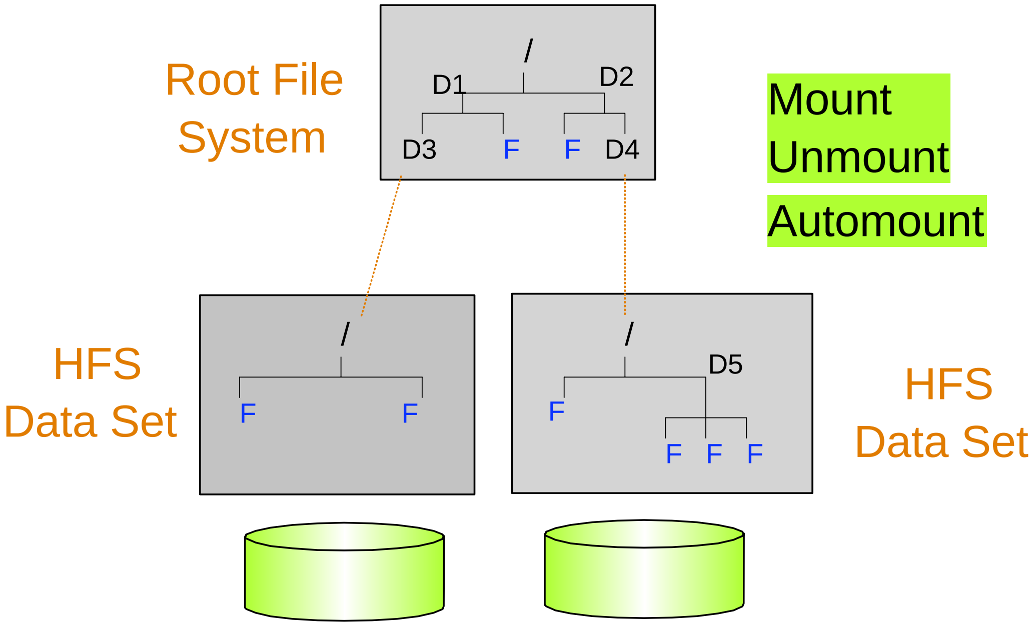
```
calendar/  dd/        head/      m4/        pax/       sync/      vi/
cap/       df/        help/      mailx/     pg/        syslogd/   wall/
cat/       diff/      ibmcat/    make/      pr/        tabs/      wc/
ccg/       diff3/     iconv/     man/       printf/    tail/      what/
chaudit/   dirname/   id/        mesg/      ps/        talk/      which/
chgrp/     du/        iden/      mkdir/     pwd/       tee/       who/
chmod/     echo/      init/      mkfifo/    rcs/       test/      whoami/
chown/     ed/        join/      mknod/     rcstest/   tic/       write/
cksum/     env/       lc/        more/      rev/       time/      xargs/
cmp/       expand/    lex/       newgrp/    rm/        touch/     yacc/
AQ:Rel9 /u/swehr/myrel9> ls ../bin
cleanup_my_tmpfiles*  frommvs*              saylibrl*
convert*              m4*                   tomvs*
fp_defs*              readmvs*              writemvs*
AQ:Rel9 /u/swehr/myrel9> ls -l ../bin
total 720
-rwxr-xr-x   1 SWEHR    SHUT         310 Jan  9 15:20 cleanup_my_tmpfiles*
-rwxr-xr-x   1 SWEHR    SHUT         606 Sep 22  1997 convert*
-rwxr-xr-x   1 SWEHR    SHUT         783 Apr  3 12:17 fp_defs*
-rwxr-x---   1 SWEHR    SHUT          59 Sep 18  1996 frommvs*
-rwxr-xr-x   1 SWEHR    SHUT      188416 Sep 30  1997 m4*
-rwxr-x---   1 SWEHR    SHUT       61440 Sep 18  1996 readmvs*
-rwxr-xr-x   1 SWEHR    SHUT       36864 Jul 11  1996 saylibrl*
-rwxr-x---   1 SWEHR    SHUT          60 Sep 18  1996 tomvs*
-rwxr-x---   1 SWEHR    SHUT       61440 Sep 18  1996 writemvs*
AQ:Rel9 /u/swehr/myrel9> _
```

File System Structure



path: /notesdata/mail

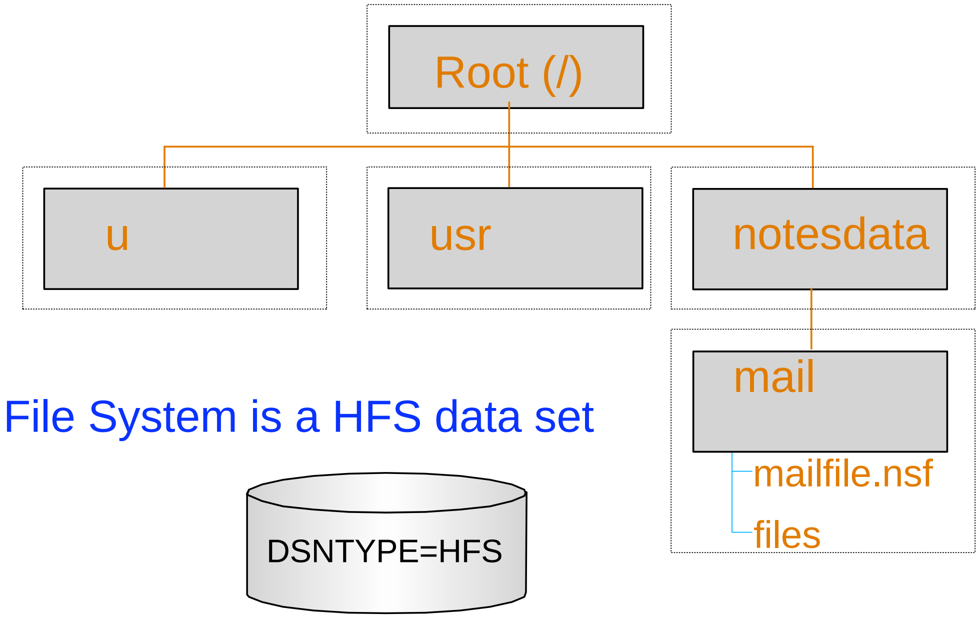
File System Mount Points



Mount
Unmount

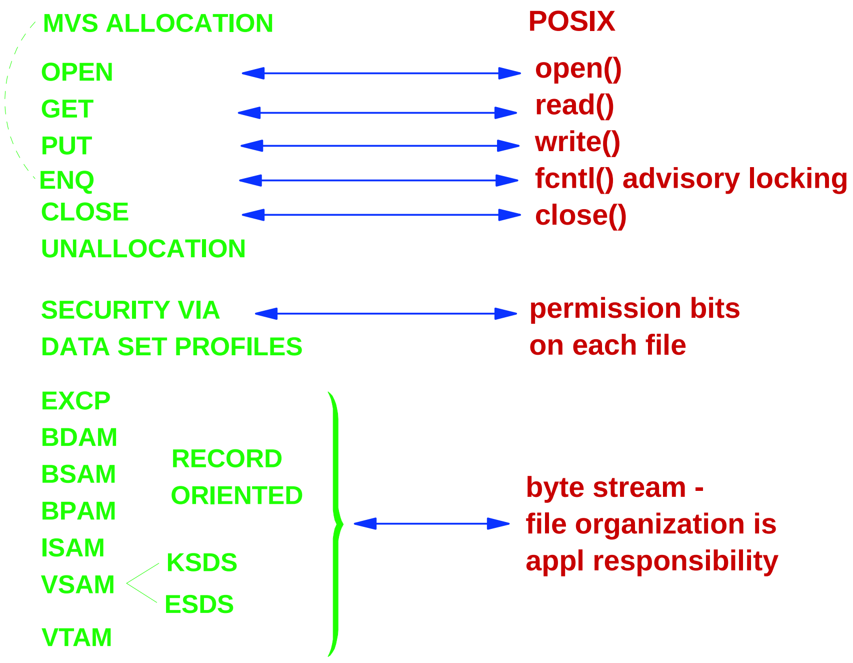
Automount

File System Data Sets



File System is a HFS data set

FILE ACCESS


MVS ALLOCATION

OPEN ↔ open()

GET ↔ read()

PUT ↔ write()

ENQ ↔ fcntl() advisory locking

CLOSE ↔ close()

UNALLOCATION

POSIX

SECURITY VIA DATA SET PROFILES ↔ permission bits on each file

EXCP
BDAM
BSAM
BPAM
ISAM
VSAM
VTAM

RECORD ORIENTED

KSDS
ESDS

↔ byte stream - file organization is appl responsibility

zSeries File System (zFS)

z/OS zSeries File System

- Overview
  - A new UNIX file system to meet the changing needs of new workloads
  - Complementary to the existing Hierarchical File System (HFS)
  - In many environments we have seen significant performance improvements over HFS.
  - Based on the DCE Local File System used by DFS
  - Improved crash recovery
  - Underlying architecture supports additional function
  - zFS file systems can be shared in a sysplex

z/OS zSeries File System

- Overview
  - zFS is a Physical File System (PFS) that runs in a Colony Address Space
  - zFS needs a FILESYSTYPE entry in BPXPRMxx
  - FILESYSTYPE TYPE(ZFS) ENTRYPOINT(IOEFSCM) ASNAME(ZFS)
  - zFS requires a JCL PROC in PROCLIB
  - zFS file systems are mounted just like HFS file systems except the TYPE is ZFS
  - zFS first became available in z/OS V1R2
  - Available for OS/390 V2R10 and z/OS V1R1 via APAR OW51780

z/OS zSeries File System

- Administration
  - Format a zFS aggregate
  - IOEAGFMT
  - Manage a zFS aggregate / file system
  - zfsadm command suite
  - pfsctl API
  - Define configuration options for aggregates
  - IOEFSPRM parmlib member
  - Define zFS to UNIX System Services
  - add FILESYSTYPE entry in BPXPRMxx parmlib member
  - Define zFS PROC

z/OS zSeries File System

- Crash Recovery
  - zFS is a logging file system. It logs metadata updates. If a system crash occurs, the log is replayed to bring the file system to a consistent state.
  - I/O requests are started immediately asynchronously so that if a system crash occurs, most data is already on disk
- Backup / Restore
  - A zFS aggregate can be backed up and restored using IDCAMS REPRO
  - A zFS aggregate can be backed up and restored using DFSMSdss
  - The aggregate must be quiesced before the backup

z/OS zSeries File System

- POSIX Removal (V1R3)
  - Problem
    - zFS is a POSIX process which causes some unwanted restrictions
    - zFS mount commands must be placed in /etc/rc
    - zFS does not support OMVS restart
  - Solution
    - Remove POSIX calls
    - Replace with basic MVS calls
    - zFS mounts can be put in BPXPRMxx
    - OMVS restart supported for zFS
  - Migration
    - Mount commands can be moved to BPXPRMxx or left in /etc/rc

z/OS zSeries File System

- UNIXPRIV Support (V1R3)
  - Problem
    - zFS Adminstration should not require UID=0
  - Solution
    - Support the UNIXPRIV class profile
    - READ authority to SUPERUSER.FILESYS.PFSCTL instead of UID=0 to manage zFS

z/OS zSeries File System

- Publications
  - z/OS Distributed File Service zSeries File System Administration (SC24-5989-01)
  - Available online by choosing Library from the z/OS home page
  - http://www.ibm.com/servers/eserver/zseries/zos