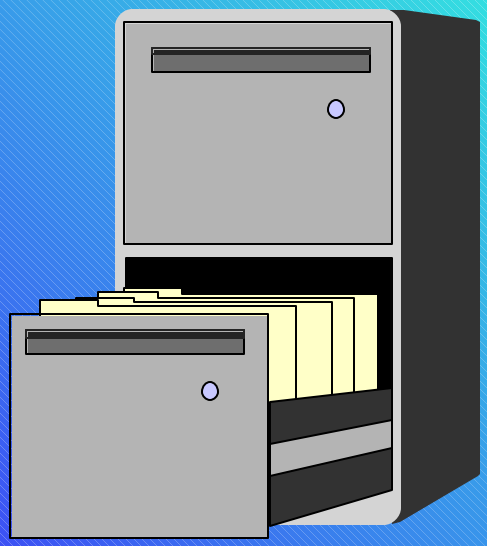
Session 9301

VM File Systems Using SFS and BFS



Alan Altmark
IBM Corporation
Endicott, New York

Disclaimer

This presentation provides a comparison and contrast of the various file systems available for use in CMS. Characteristics, administration, programming interfaces, and relative performance are discussed.

References to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe on any of the intellectual property rights of IBM may be used instead. The evaluation and verification of operation in conjunction with other products, except those expressly designed by IBM, are the responsibility of the user.

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

S/390 VM/ESA IBM AIX OS/390 z/VM z/OS

Other company, product, and service names, which may be denoted by double asterisks (**), may be trademarks or service marks of others.

LINUX is a registered trademark of Linux Torvalds.

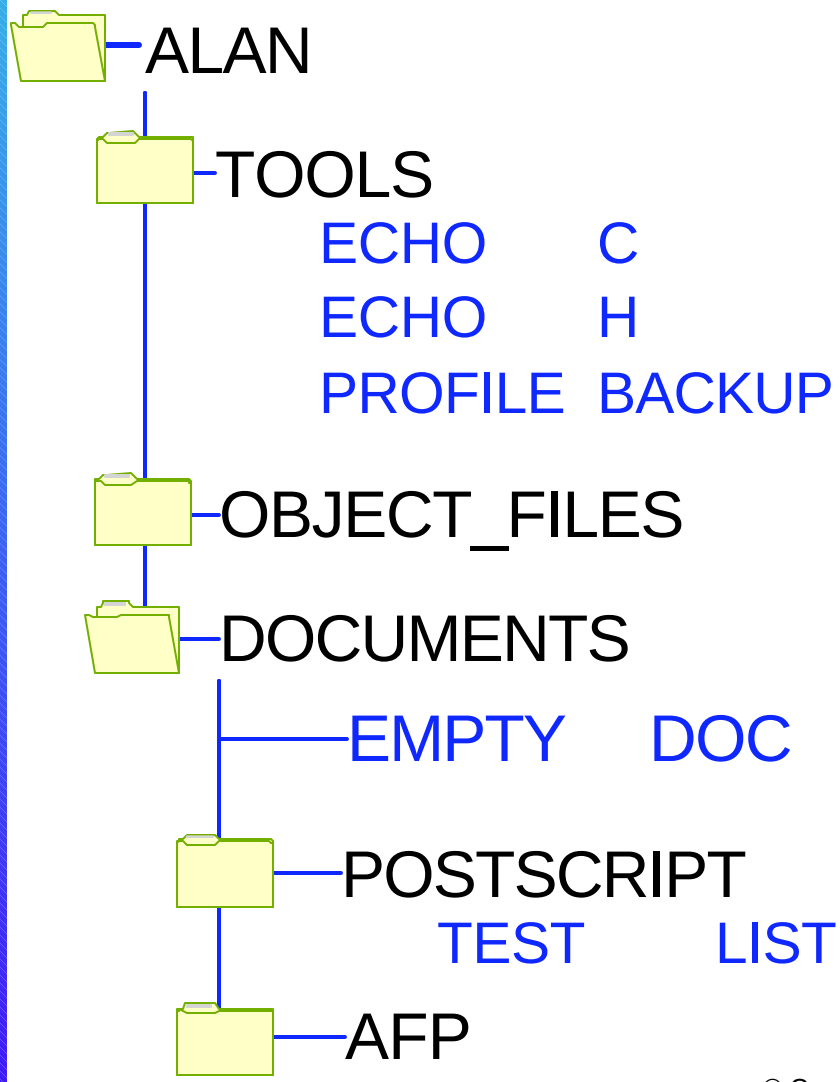
Windows is a registered trademark of Microsoft Corporation.

Shared File System

Shared File System

- Hierarchical directory structure
  - File Control
    - File updates shown whenever file is used
  - Directory Control
    - Access-to-Release consistency for all files
    - Eligible for VM Data Spaces
- File sharing
  - one writer, many readers for a file
  - data integrity
- Each user has his or her own space in the filepool
  - owns all objects created in it, regardless of who created the object

Shared File System File Space



Standard CMS file names

Directories may be empty

Files may be empty!

Useful to preserve authorizations

Getting Started with SFS

1. Get enrolled
   - enroll user alan phantom (blocks 10000
   - This creates a top-level directory for user ALAN with a maximum of 10000 4K blocks

2. Set up defaults
   - SET FILEPOOL PHANTOM:

3. Create subdirectories if you want
   - CREATE DIRECTORY .tools
   - directories can have 8 elements, with 16 character per element
   - underscore and a few other special characters are OK

Getting Started with SFS *(continued)*

4. Access your directories
   - ACCESS . A
   - ACCESS .tools B
   - ACCESS vmsysu:hornet.kato C

5. Edit your files in the usual way

Aliases

- An alias a logical link to a file somewhere in the same file pool
  - Your own files
  - Someone else's

```
> access phantom:bob.tools b
> filelist * * b
    create alias / = = .tools.special
> create alias cool exec phantom:frank.goodies = = .tools
```

- Only original owner can grant authorization
- You have to have READ authority to the base file to create an alias to it
- You have to have WRITE authority to the directory in which you are placing the alias

Aliases

- Lets you combine files from multiple directories into a single directory
  - Can save on file modes
- Erasing an alias simply removes the alias; it does not erase the base file
- If your permission to the base file is removed, the alias remains, but your permission is put into "revoked" status

Sharing your files

- Files and directories are separately authorized
  - Someone can see files, but not the directory they're in
  - Someone can access directory, but can't see the content of any of the files
- GRANT and REVOKE AUTHORITY commands control access
  - GRANT AUTH *fn ft {dirid | fm}* TO *{userid* | PUBLIC}
    ([READ | WRITE]
  - GRANT AUTH *{dirid | fm}* TO *{userid* | PUBLIC}
    ([READ | WRITE] [NEWREAD | NEWWRITE]
- Only the owner can GRANT AUTHORITY

Levels of authority

- PUBLIC authority gives access to all enrolled users of the file pool
  - If administrator uses ENROLL PUBLIC, then any user on the system can access the file or directory
- READ means you can look at the contents of the object
  - Directories contain lists of files
  - Files contain data
- WRITE means you can modify the contents of an object
  - You can create and erase files
  - What you create is owned by files pace owner, not you.
  - Only the files pace owner can create or erase a directory

Make Authorization Simple

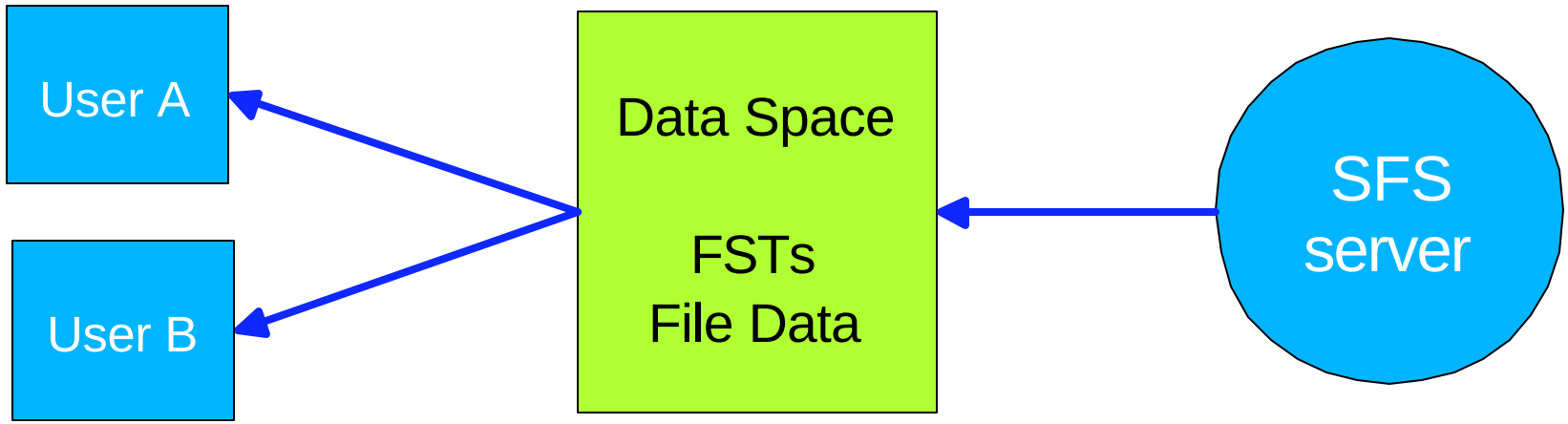
- NEWREAD and NEWWRITE automatically confer READ or WRITE authority to a file when it is created
- NEWREAD and NEWWRITE does not apply to files which already exist in the directory
- It is easiest if you do things in this order
  1. CREATE DIR .SHARE
  2. GRANT AUTH .SHARE PUBLIC (READ NEWREAD
  3. Then populate the directory

Directory Control Directories

- CREATE DIRECTORY .TOOLS.TCPIP (DIRCONTROL
- Provide access to release consistency
- Changes are visible only when directory is release and re-accessed
  - Similar to minidisks, only better: No "Error 3 reading file" errors
- If updated rarely, but read often, it can be made eligible by sysadmin to be placed in a **data space**

Directory Control Directories & Data Spaces

- If updated rarely, but read often, it can be made eligible by sysadmin to be placed in a **data space**



- Each update to the directory causes a new data space to be created
- When last user of an old data space releases directory, the data space is destroyed

Command useful with SFS

- FILELIST * * dirid
- QUERY AUTH
- QUERY ACCESSED
- SET FILEPOOL
- ALIALIST
- DIRLIST

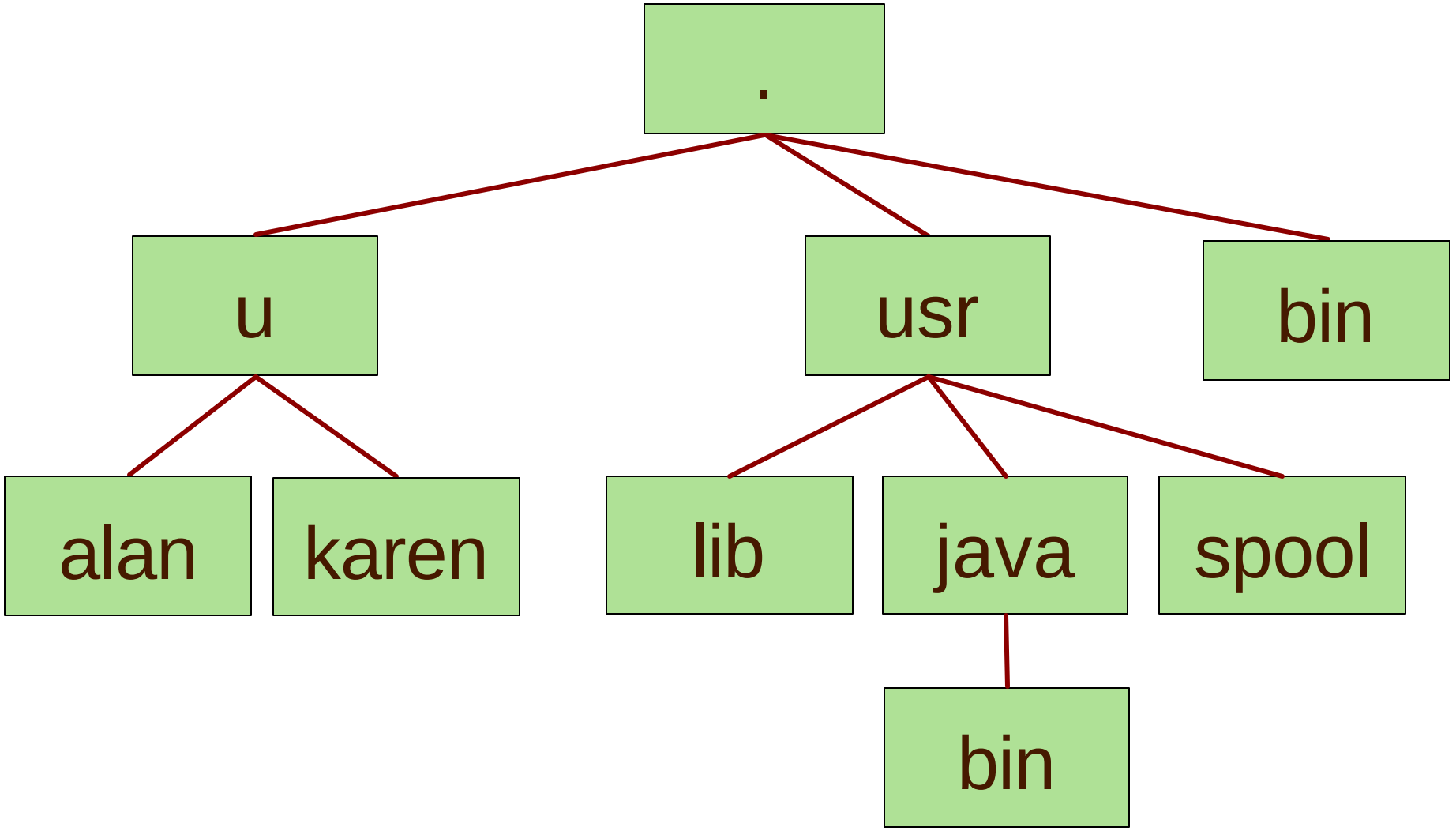
- Note that the output from QUERY DISK is different. Consider moving to QUERY ACCESSED instead

Byte File System

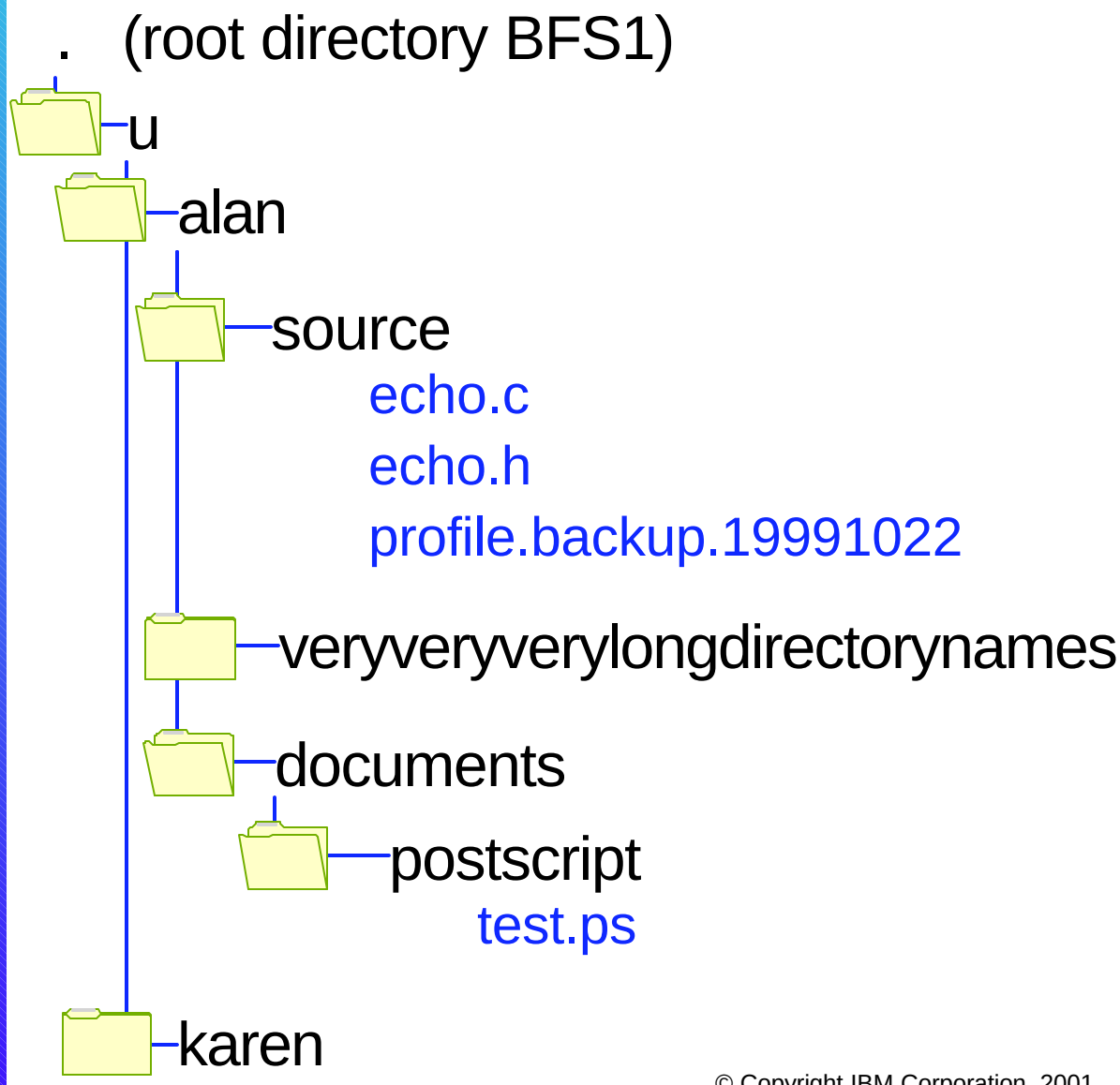
Byte File System (BFS)

- UNIX** file system
  - hierarchical directories
  - streams instead of records
  - advisory locking
  - multiple users can write to the same file
  - no shadow updates - all updates occur instantly
    - All users see same file image
- A BFS is simply an enrolled user in an SFS file pool
  - Enrolled with the "BFS" option
  - UNIX-style permissions determine access rights, not VM user ID

Byte File System File Space



Hierarchical BFS Directory



Accessing BFS

- As with SFS, the server moderates access
  - BFS is just another kind of file space in an SFS filepool
- OPENVM MOUNT
  - Cannot use ACCESS command
  - easiest to use with Shell & Utilities Feature (S&U)
- S&U mount
  - openvm mount /../vmbfs:phantom:bfs1/ /
  - openvm shell
  - > cd /u/alan
  - > mkdir /mnt
  - > cms openvm mount /../vmbfs:phantom:bfs2/u/billybob/shr ./mnt

BFS Security

- As with UNIX, access is based on POSIX UID and GID
  - UID (user id) and GID (group id) are simply integers
  - Kept in CP directory or ESM for each user
  - CP provides UID/GID values when clients connect to the file pool - user can't fake it
- permissions: owner, group, other
- authorizations: read, write, execute
- UID 0 is a superuser
  - can do anything to any file or directory

BFS Permisssions

- Creator's UID and GID stored when object is created
  - Superuser can use OPENVM OWNER or chown to change UID and/or GID
  - Owner can change only GID
- Accessor's UID and GID are compared to object's stored UID and GID
  - if UID matches stored UID, then owner permissions apply
  - if GID matches stored GID, then group permissions apply
  - otherwise, "other" permissions apply
- Changing VM user ID doesn't affect stored UIDs and GIDs

Managing BFS Permisssions

- openvm permit /u/alan/sample.txt rw- rw- r-- (replace
  - owner can read and write
  - group members can read and write
  - everyone else can read
- chmod u+x,g=x,o-rwx /u/alan/myprog
  - owner can run program in addition to current setting
  - group can run the program
  - no one else has access
- Default is controlled by umask or OPENVM SET MASK command

Creating a BFS from scratch

enroll user bfs1 phantom (blocks 200000 bfs
openvm mount /../VMBFS:PHANTOM:BFS1/ /
openvm create directory /u

openvm create directory /u/alan
openvm owner /u/alan STAFF ALAN

openvm create directory /u/karen
openvm owner /u/karen STUDENT KAREN

Creating a BFS from scratch (redeux)

enroll user bfs1 phantom (blocks 200000 bfs
openvm mount /../VMBFS:PHANTOM:BFS1/ /
openvm shell

> mkdir /u/alan
> chown alan:staff /u/alan

> mkdir /u/karen
> chown karen:student /u/karen

Command useful with BFS

- OPENVM LISTFILE
- OPENVM GETBFS and OPENVM PUTBFS
  - Moves CMS files to/from BFS
- XEDIT
- CMS PIPELINEs
  - bfs stages available to work against BFS files
    - mounted or not

Xedit

- xedit /dir/file.ext (nametype bfs bfsline NL
  - *bfsline* value is inserted in data stream at the end of each line when the file is written
  - When opening a file, XEDIT uses *bfsline* value to decide where each line ends
  - *bfsline* default is NL (0x15)
  - Similar to RECFM V
- xedit /dir/file.ext (nametype bfs bfsline 80
  - chunks up file into fixed-length 80 byte "records"
- Not useful to edit binary files
  - Use NFS and a binary editor on your PC instead

NFS Client in z/VM

- OPENVM MOUNT ...
  - `/../nfs:host.ibm.com/home/alan  /home/alan/mnt ( userid ALAN password XXXXXX`
  - `/../nfs:vmhost/sfs1:alan.tools /home/alan/tools`
  - `/../nfs:vmhost/maint.193  /home/alan/vmtools ( anonymous`
  - `/../nfs:vmhost/../vmbfs:sfs1:root/u/alan /home/alan`
- You can set up NETRC DATA file to contain user IDs and passwords
- Use XEDIT to edit files on your PC!

File System References

- Usage
  - CMS User's Guide
  - OpenExtension User's Guide
  - OpenExtension Command Reference
  - HELP OSHELL MENU
  - HELP OPENVM MENU
- Programming
  - CMS Application Development References
  - CMS Application Multitasking
  - CMS Application Development Guide
  - OpenExtension Callable Services Reference
  - IBM C for VM/ESA Library Reference

Contact Information

- By e-mail: Alan_Altmark@us.ibm.com
- In person: USA 607.752.6027
- The Web: http://www.VM.ibm.com/devpages/altmarka
- Mailing lists: VMESA-L@listserv.uark.edu