SUBJECT: PRIMOS REV. 19 ROAD SHOW

TIME : 2 Days

MATERIAL: Supplied

OBJECTIVES:

UPON COMPLETION OF THIS COURSE THE STUDENT SHOULD BE ABLE TO:

1. Install PRIMOS rev. 19 at customer sites
2. Describe and implement User profiles on a customer machine.
3. Describe and install ACCESS_CONTROL_LISTS on a customer machine.
4. Describe and install DISK QUOTAS on a customer machine.
5. Use the new FIX_DISK command to convert a disk to rev. 19.
6. Explain the new BADSPOT handling feature.
7. Explain the new BOOTSTRAP procedure.
8. Use the new FUTIL REPLACEMENT commands.
    a. COPY b. DELETE
    c. LIST_DIRECTORY d. RWLOCK
    e. PROTECT
9. Describe the new COMMAND PROCESSOR functionality
    a. Wildcarding b. Iteration
    c. Name Generation d. Tree-walking
10. Describe briefly, the INTERNAL modifications
11. Rebuild primos by invoking the supplied command files.
12. Describe the rev. 19 NPX features.
13. Explain the REMOTE JOB ENTRY enhancements
14. Define the changes to DBMS.
15. Describe the remaining changes to PRIMOS rev. 19.

REV19 ROAD SHOW OUTLINE

1. INTRODUCTION
   - A. NEW FEATURES
2. USER PROFILES
   - A. TERMS
   - B. ACCESS CONTROL LISTS
3. DISK QUOTAS
4. BADSPOT HANDLING
5. FIX_DISK
6. NEW BOOT STRAP PROCEDURE
7. FUTIL REPLACEMENT COMMANDS
   - A. COPY
   - B. DELETE
   - C. LIST_DIRECTORY
   - D. RWLOCK
   - E. PROTECT
8. COMMAND PROCESSOR
   - A. WILD CARDING
   - B. ITERATION
   - C. NAME GENERATION
   - D. Treewalking
9. INTERNAL MODIFICATIONS
   - A. QUIT$
   - B. LOGOUT NOTIFICATION
   - C. EPFs
   - D. LOGIN CHANGES
10. RESOURCE EXTENSIONS
11. PRIMOS BUILD CPL FILES
12. CONVERTING TO PRIMOS REV. 19
13. NPX
14. RJE
15. DBMS
16. REMAINING REV19 PRODUCTS

REV. 19 ROAD SHOW DAILY TOPIC MAP

	DAY 1	DAY 2
AM	INTRODUCTION	CONVERSION
	USER PROFILES	"
	DISK QUOTAS	"
	BADSPOT HNDLG	"
	FIX_DISK	"
	LUNCH	
PM	BOOTSTRAPPING	CONVERSION
	FUTIL REPLACEMENT	NPX
	COMMAND PROCESS.	RJE
	PRIMOS BUILD	DBMS

P R I M O S - Revision 19.0

SECURITY

USER PROFILES

ACCESS CONTROL LISTS

DISK QUOTAS

EASE of USE

COMMAND PROCESSOR ENHANCEMENTS

FILE UTILITY COMMANDS

INTEGRITY

FIX_DISK

IMPROVED BADSPOT HANDLING

MISCELLANEOUS ENHANCEMENTS

INTERNAL CHANGES

PRIMOS REV. 19 NEW FEATURES

ACCESS CONTROL LISTS

ASSIGNABLE AMLC LINE IMPROVEMENTS

BADSPOT HANDLING

COMMAND PROCESSOR EXTENSIONS

CPL PHANTOMS

CROSS PROCESS SIGNALLING (Internal use only)

DISK QUOTAS

EXECUTABLE PROGRAM FORMAT (Internal use only)

FILE SYSTEM UTILITY

FIX DISK

ENHANCED FORCEW PRIMITIVE

PRIMOS REV. 19 NEW FEATURES

THE HELP COMMAND

STATIC ON UNITS (Internal use only)

STATUS COMMAND CHANGES

MPC4 SUPPORT

USER PROFILES

NEW ERROR CODES

PRIMOS INTERNAL LOGIC MODIFICATIONS

CORRECTED REV. 18 POLERS

NEW FEATURES FOR APPLICATIONS PROGRAMS

Asynchronous Signals

- Rev 18:

quit$	user break, control-p
cpu_timer$	cpu seconds watchdog timer (see limit$)
alarm$	real time seconds watchdog (see limit$)

- New signals at rev 19:

logout$	1-2 minute warning before force logout
ph_logo$	a user initiated phantom has logged out
cps	cross process signals (not released)

- Break$ routine only disables quit$.

- To insure atomic code, create condition handlers for all asynchronous signals, or use SW$INT (documented below).

NEW FEATURES FOR APPLICATIONS PROGRAMS

SW$INT (KEY, SELECTION, VALUE, ERCODE [, OUTER_RING])

NOTE: This is an unreleased call. The calling sequence and/or fuctionality is subject to change. It is documented here to provide a simple mechanism to control asynchronous signals by third-party sub-system software.

SW$INT is used to control the enable/disable status of the software interrupts. It does this by setting/resetting the enable bit(s) of the software interrupt ring control words located in pudcom. (For terminal quit, BREAK$ is called to enable/disable.) The format of this word is:

```
dcl 1 b_swityp based,
      2 mbz bit(10),
      2 logout bit(1),          /* logout condition */
      2 cps bit(1),             /* cross process signalling */
      2 cpu_time bit(1),        /* cpu timer */
      2 alarm bit(1),           /* real time timer */
      2 lon bit(1),             /* phantom logout */
      2 terminal bit(1);        /* terminal quit */
```

SW$INT (continued)

SW$INT is able to set on, set off, or simply read the status of the interrupt type chosen by its caller. The type is chosen by either setting a bit(s) on in the input argument, SELECTION, or using one of the "all" keys. For read, the current setting is returned in the argument, VALUE.

The valid keys for specific bit selection(s) are:

- k$on - turn interrupt(s) on
- k$off - turn interrupt(s) off
- k$rdon - read present status then turn interrupt(s) on
- k$rdof - read present status then turn interrupt(s) off
- k$read - read present status

The valid key for non-specific bit selection are:

- k$alon - turn on all interrupts
- k$alof - turn off all interrupts
- k$raon - read all status then turn on all interrupts
- k$raof - read all status then turn off all interrupts
- k$rdal - read present status of all interrupts

SW$INT (continued)

A user may enable/disable any interrupt(s) in an outer ring by including the optional OUTER_RING argument. If it is included it's value currently must be 3. Software interrupts are normally on in the outer rings.

Abnormal conditions: Bad key. Bad parameter. Buffer too small.

NEW FEATURES FOR APPLICATIONS PROGRAMS

Command Processor

- Static mode programs all features enabled, no_verify for wildcard selections is the default. for static mode programs

- CPL programs only simple iteration enabled.

- NX$___ only simple iteration enabled.

- NW$___ only treewalking enabled.

- Special command processor arguments:

  Wildcarding: -before, -after, -file, -directory, -acat
  -segment_directory, -verify, -no_verify
  Treewalking: -walk_from, -walk_to, -bottom_up

NEW FEATURES FOR APPLICATIONS PROGRAMS

Attach-Scan

- All local disks are searched before all remote disks (in ldev order).

- If user does not have Use access to the MFD, then that partition is not searched.

- If the user does not have List access to the MFD and does not have sufficient access to the UFD, then the search continues (becuase 'no information' is returned).

- Search stops on any 'bad password' and 'insufficient access rights'

- New error code 'Top-level UFD inaccessible or not found'. 'Not found' will never be returned by an attach scan at rev 19.

Disk Quotas

- Quota checking can be performed by the program, and appropriate actions taken (i.e. closing files, trying other UFDs).

- New error code 'Maximum quota exceeded'.

NEW FEATURES FOR EXTERNAL LOGIN

- During login, CMDNCO>LOGIN is resumed. (.SAVE suffix not allowed)

- During logout, CMDNCO>LOGOUT is resumed if it exists,
  else CMDNCO>LOGIN.

- Login-over-Login is defined as:

  Rev 18: change of user id (no accounting meters reset)
* Rev 19: logout followed by login

- New features at rev 19:

  Project Name: prjid$ returns name of user's login project
  ACL Groups: getid$ returns ACL groups (and user_id)
  Disk Quotas: check for free records before writing accounting
  or metering files
  Disk Usage: q$read returns record-time-product meter to account
  for disk usage over time within a quota sub-tree.

- LOGIN.@ (CPL, COMI, SAVE) is resumed from user's initial attach
  point following execution of external login program.

SUBJECT: USER PROFILES and ACLS

TIME : 2 and 1/2 HOURS

MATERIAL: SUPPLIED

OBJECTIVES:

UPON COMPLETION OF THIS SUBJECT THE STUDENT WILL BE ABLE TO:

1. DESCRIBE AN ACCESS CONTROL LIST
    a. ACCESS CATEGORY b. SPECIFIC ACL
    c. DEFAULT ACL
2. USE THE SET_ACCESS COMMAND TO PROVIDE ACL PROTECTION ON THE FILE SYSTEM.
3. DEFINE AND USE THE PRIORITY ACL
4. EXPLAIN HOW TO CONVERT TO AN ACL DIRECTORY.
5. DESCRIBE THE USER PROFILE MECHANISM
6. DESCRIBE THE CONCEPT OF PROJECTS
7. INVOKE AND USE THE EDIT_PROFILE UTILITY TO ADD USERS, PROJECTS AND ACL GROUPS TO THE SYSTEM.
8. DESCRIBE A DEFAULT PROJECT
9. DEFINE THE ROLES OF A SYSTEM ADMINISTRATOR, PROJECT ADMINISTRATOR.
10. EXPLAIN WHAT A PROFILE IS AND HOW THEY CAN BE USED AT EACH SITE.
11. ACTIVATE REMOTE USERS OF THE SYSTEM BY REGISTERING REMOTE IDs.

DEFINITIONS

ACCESS CONTROL LISTS

ACCESS GROUP NAMES

USER PROFILES

USER ID

USER REGISTRATION

PROJECT

INITIAL ATTACH POINT

ACCESS CONTROL LISTS - MOTIVATION

PASSWORDS ARE ACTIVE AT EACH DIR ATTACH
ACL PASSIVE

- To improve file system security.

- To provide an easy to use interface for user's and programs to set and modify access.

- To interact with user profiles to provide common access for specified groups of users under administrative control.

- To provide an extensible system for the specification/ enforcement of access rights.

- Passive protection mechanism versus passwords which must be specified by users/programs.

ACCESS CONTROL LISTS

- New protection mechanism to control access to files and directories. Alternative to the current password scheme.

- An Access Control List (ACL) is a list of users and access rights to one or more files/directories.

Romeo :	r	(Romeo may read files)
Juliet :	rw	(Juliet may read and write files)
.Muppet:	l	(The group of users '.muppet' may list the contents of the ufd)
$rest :	none	(All other users have no access)

  ACLs may be specified for a single file or a set of files.

- ACLs are enabled on a per-mfd basis. Password directories can be subordinate to ACL directories not vice versa.

- Like user profiles, ACLs default to closed. A user must specifically be given access - NONE is default.

- Requires rev 19 disk format.

ACCESS CONTROL LISTS

An access pair consists of:

<identifier> : <access_rights>

Identifier

The identifier may be

- A user_id contained in the user profile data base
- An ACL group, which is a special name that begins with a '.' and specifies a group of users which share common access rights.
  For example,

  .MUPPET specifies KERMIT
  MISS_PIGGY
  FOZZIE_BEAR

  (ACL groups are specified in a user's profile.)
- The special identifier '$rest' which signifies all other users.

ACCESS CONTROL LISTS

Access Rights

RIGHT	APPLIES TO	MEANING
Protect	Directories	Accesses and attributes may be changed.
Delete	Directories	Entries may be deleted from the dir.
Add	Directories	Entries may be added to the dir.
List	Directories	The contents of the dir may be read.
Use	Directories	The dir may be attached to.
Read	Files	The file contents may be read.
Write	Files	The file contents may be changed.
ALL	Both	PDALURWX.
NONE	Both	Explicitly deny all access.

'LD' - LIST WITH ATTACH

ACCESS CONTROL LISTS

Useful Combinations of Access Rights

u: Required for the user to access anything at this directory level or below. Allows the user to attach.

lu: Allows the user to attach and display the names of file system objects.

lur: A good combination for trusted users. Allows user to peruse a directory and its contents, and to make copies.

lura: Allows the user to also create new files and directories in a non-destructive manner. Existing files cannot be changed.

lurwad: A good combination where access is controlled by an administrator, but allows the user to do 'everything' - read and write files, add and delete entries. The user cannot change the protection on any object.

all: A standard combination for the system administrator or 'owner' of an object. Allows protection to be changed, and disk quotas to be set.

none: Prevents all access.

ACCESS CONTROL LISTS - Calculating Access

WHEN IS ACL ACCESS CHECKED?

- During an attach operation
- During a file open operation

ACL modifications not reflected until attach again.

ACCESS CONTROL LISTS - Calculating Access

HOW IS ACL ACCESS CALCULATED?

- Password owner/non-owner access rights are mapped to ACL rights:

Owner:	=	PDALU
Non-owner:	=	LU
Read:	=	R
Write:	=	W
Delete:	=	D

- Calculate access as follows:

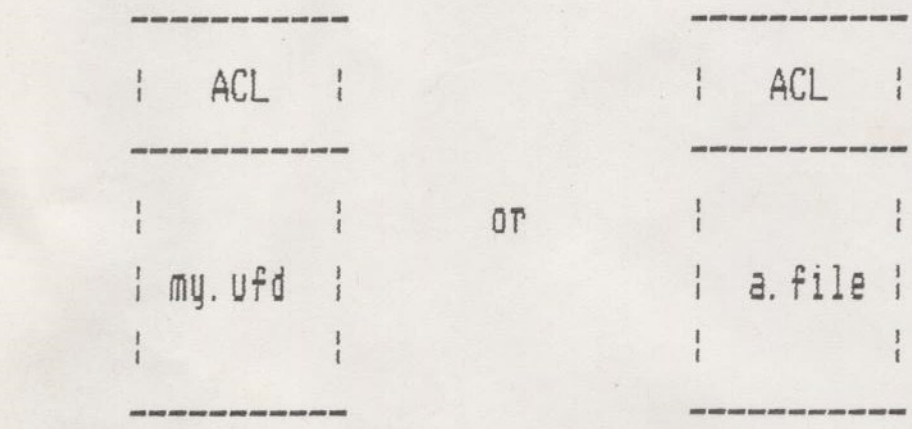
```
Priority Access:  if priority_acl then
                      if user_in_pacl then
                          get access from pacl
User Id:          else if user_id_in_acl then
                          get access from acl
ACL Groups:           else if user_member_of_group(s) then
                              get access for each member_group
                              logical-or these accesses together

$Rest:                    else if $rest then
                                  get access from $rest pair
                              else no access
```

ACCESS CONTROL LISTS

There are two kinds of ACLs: specific and category.

Specific Protection

There is a unique acl associated with a single file/directory.
The acl is accessed through the name of the object it protects.



set_access my.ufd <acl>

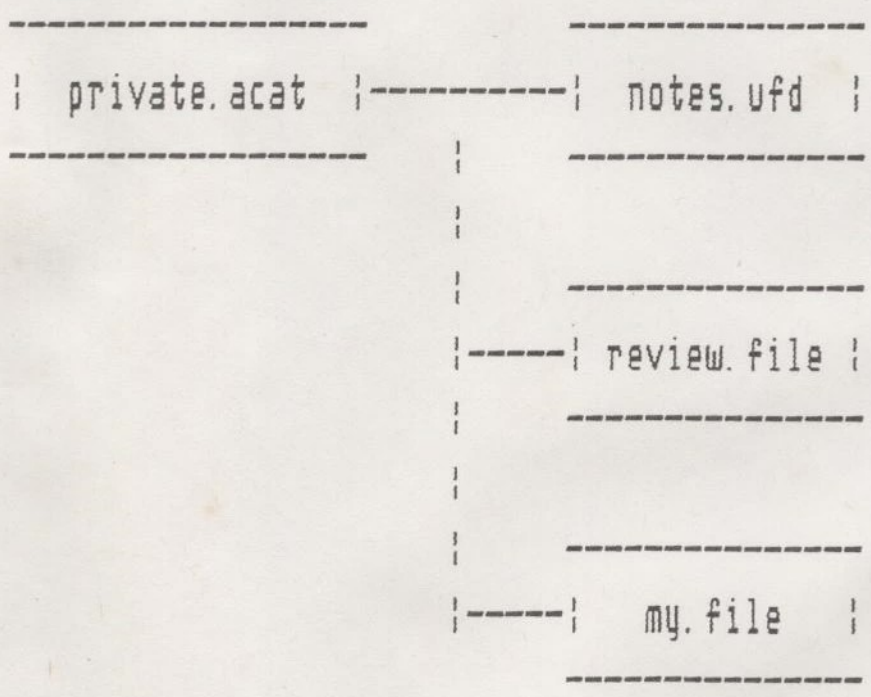
set_access a.file <acl>

'sac'

ACCESS CONTROL LISTS

Category Protection

There is a file system object called an access category that protects one or more files/directories. The acl is accessed by the name of the access category. When the access category is modified, the access rights for all the files/directories protected by the acl are changed.

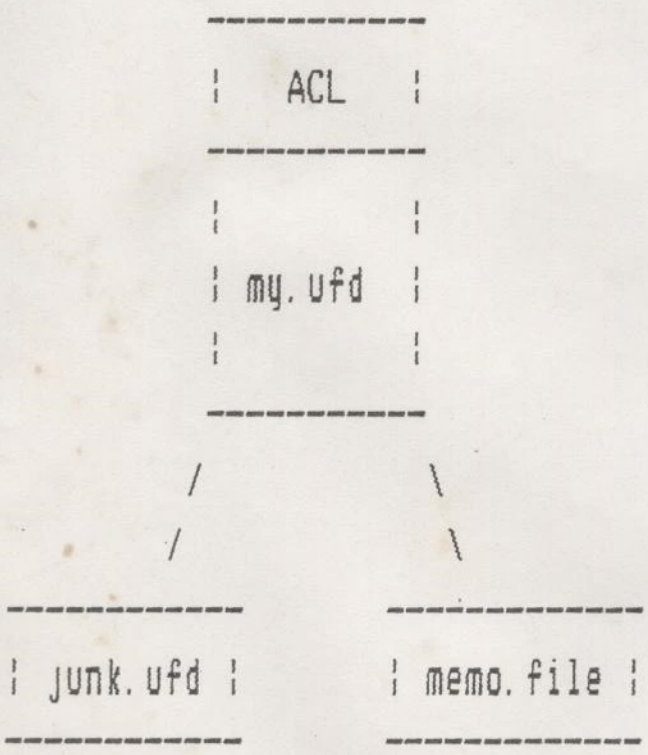


set_access private.acat <acl>

ACCESS CONTROL LISTS

Default Protection

If no specific or category protection is specified, then the file/directory is protected by the acl associated with the parent directory. New files are protected by default acl.



set_access my.ufd <acl>

ACCESS CONTROL LISTS
NEW AND MODIFIED COMMANDS

SET_ACCESS, SAC <target_pathname>
<target_pathname> <access_control_list>
<target_pathname> -LIKE <reference> for specific ACL's
<target_pathname> -CATegory <access_category>

<target_pathname>
Pathname of file, directory or access category to protect.

<access_control_list>
A list of access pairs - <identifier>:<access_rights>.

<reference>
Pathname of a file, directory or access_category.

<access_category>
Name of an access_category. Supported suffix is '.acat'.

- Used to set ACL protection for <target_pathname>.
- If <target_pathname> is a password directory, convert to an acl directory.

ACCESS CONTROL LISTS
NEW AND MODIFIED COMMANDS

- If only <target_pathname> is specified, then default access is inherited. In this case, target may not be a mfd.
- Protect access is required for the directory; or the directory containing the file or access_category.
- If not otherwise specified, $rest:none is implicit in every acl.

EDIT_ACCESS, EDAC <target_pathname> <access_control_list>

- Used to modify/create an acl.
- The access pair for each new identifier in the is added to the target's acl.
- Each existing identifier has its access changed in the target's acl to be the specified access pair.
- If an access pair is specified with no access rights, that access pair will be deleted from the target's acl.

ACCESS CONTROL LISTS
NEW AND MODIFIED COMMANDS

LIST_ACCESS, LAC [<target_pathname>]

- Used to list the acl protecting <target_pathname>.
- If <target_pathname> is omitted, then the acl protecting the current attach point is listed.
- If a priority acl is in effect, then it is listed first.
- List access is required to the directory that contains the protected target.

REVERT_PASSWORD

- Used to convert an acl directory back to a password directory.
- Converts the current directory back to a password directory.
- Protect access is required.

ACCESS CONTROL LISTS
NEW AND MODIFIED COMMANDS

SET_DELETE, SDL <pathname> {-PROTect | -NoPROTect}

<pathname>

Name of file or directory to protect.

-PROTect

Set delete switch to prevent deletion.

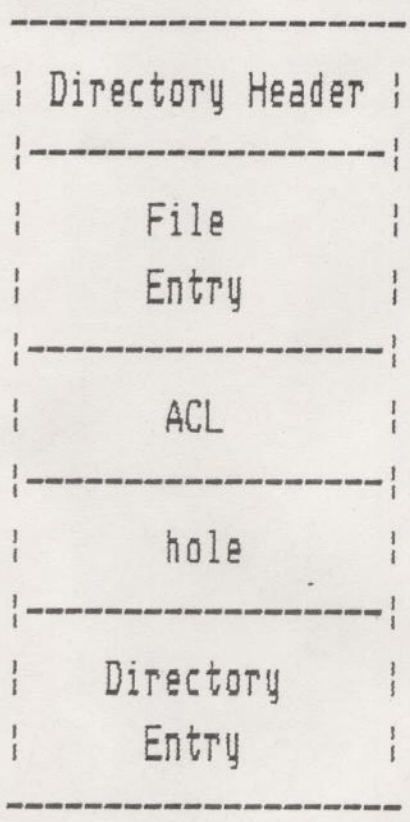
-NoPROTect

Permit deletion.

- Used to protect <pathname> from accidental deletion.
- Delete access is required to set the delete switch.
- The switch cannot be used on access_categories.

ACCESS CONTROL LISTS - Directory Structure

- A directory is a header followed by a bunch of entries.
- ACLs are embedded in the directory itself.



ACCESS CONTROL LISTS - Directory Structure

DIRECTORY ENTRY TYPES:

- Directory Header
- Vacant Entry: Unused hole in the directory.
- Normal Entry: Describes a file: SAM
  DAM
  SEGSAM
  SEGDAM

  or a directory: ACL
  Password
- ACL Entry: Set of access pairs.
- Access Category: Named ACL. Always points to an ACL entry.

ACCESS CONTROL LISTS - PRIORITY ACLS

- Mechanism to allow the system administrator or operations staff to set special overriding access on the file system, e.g. for backups.
- A priority acl may be specified from the system console for any partition on the system.
- The priority acl is checked first when computing a user's access rights.
- The $rest:none access pair is not implied in a priority acl.

ACCESS CONTROL LISTS - PRIORITY ACLS

SET_PRIORITY_ACCESS, SPAC <partition_name> <access_control_list>

- Sets a priority ACL on a partition.

- If $rest identifier is given as part of <access_control_list> then the access rights given by that id override any other access control in effect on the partition.

- Can only be executed from the system console or by the system administrator.

LIST_PRIORITY_ACCESS, LPAC <partition_name>

- Lists any priority acl on the partition. Should only be used when List_ACcess cannot be used.

REMOVE_PRIORITY_ACCESS, RPAC <partition_name>

- Removes the priority acl on the partition.

- Can only be executed from the system console or by the system administrator.

STRATEGY FOR PROFILES, ACLS, QUOTAS

Utilizing ACLs

- Protect files and directories

- Allow the setting of disk quotas. Protect access to the parent directory allows imposing quotas on subdirs.

  SA may control top-level quotas.
  PA may control quotas on sub-directories.

- Allow installation of new commands/libraries. Add access is required to CMDNCO or LIB to add new files. Note, delete access is also required to COPY in a new version of an existing file.

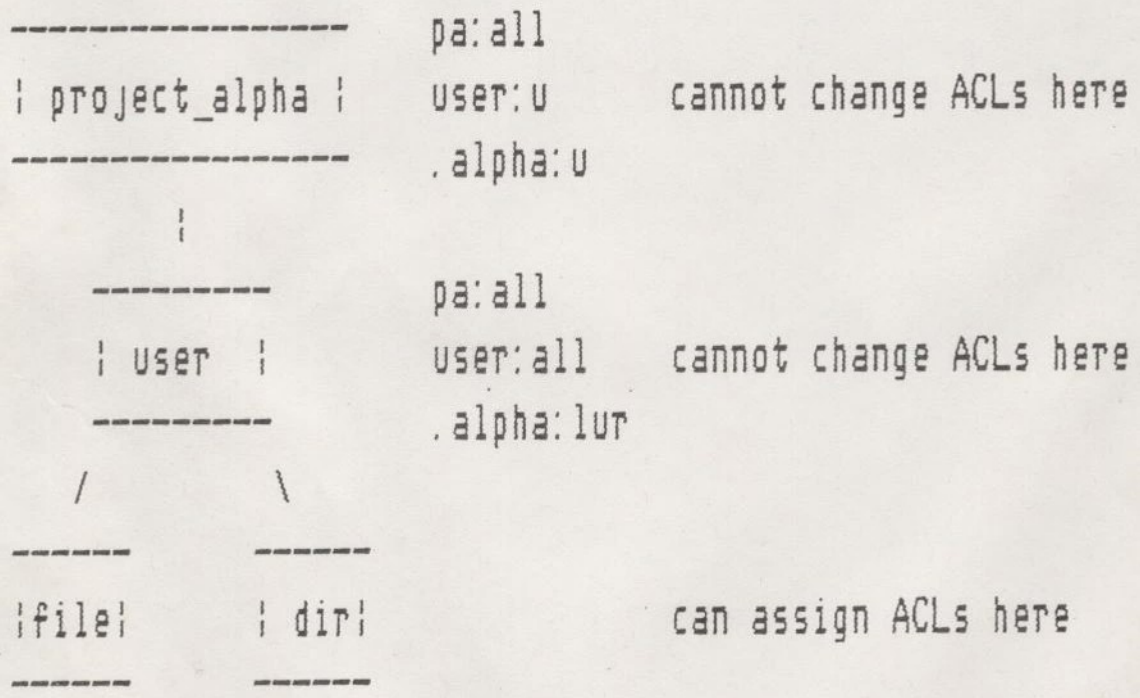
- May prevent execution of certain external commands. Any command in CMDNCO can be ACL'd to prevent read access to any user or group of users. This prevents execution of the command.

- Allow reversion of a password directory to an ACL directory requires protect access to the ufd.

STRATEGY FOR PROFILES, ACLS, QUOTAS

Utilizing ACLs

- The SA/PA may want to prevent a user from changing the ACL access to his/her initial attach point, but still allow the user to specify ACLs for any sub-directories of the iap. This is useful to force some minimal access to a user's ufd say for the PA or project members.



- If a user does not have protect access to his/her ufd, then no ACLs can be assigned to any files or sub-directories. This is useful in a tightly controlled system where the SA or PA wants full control (at a cost of more administrative work).

STRATEGY FOR PROFILES, ACLS, QUOTAS

Useful Combinations of Access Rights

u: Required for the user to access anything at this directory level or below. Allows the user to attach.

REMEMBER MUST BE 'U' IN SUPERIOR DIR TO ALLOW USER TO SET OWN ACL'S.

lu: Allows the user to attach and display the names of file system objects.

lur: A good combination for trusted users. Allows user to peruse a directory and its contents, and to make copies.

lura: Allows the user to also create new files and directories in a non-destructive manner. Existing files cannot be changed.

lurwad: A good combination where access is controlled by an administrator, but allows the user to do 'everything' - read and write files, add and delete entries. The user cannot change the protection on any object.

all: A standard combination for the system administrator or 'owner' of an object. Allows protection to be changed, and disk quotas to be set.

none: Prevents all access.

STRATEGY FOR PROFILES, ACLS, QUOTAS

Utilizing ACL Groups

- Can be specified in a user's system profile and/or project profile.

- A user_id becomes a member of the ACL groups in his/her system profile during every login, regardless of the project_id. These are often used for global system access. For example:
  .Super_user = could give ALL access to system ufds

- A user_id becomes a member of the ACL groups in his/her project profile only when logging into that project_id. Oftentimes, a project_id will have a corresponding ACL group. For example:
  Project = Operations ACL Group = .Operations

- ACL group access rights are additive. If a user is a member of multiple groups which are specified in a single ACL, the user obtains the sum of the access rights for each group. For example:
  .Project_Leaders:pd
  .Project_Members:alurw
  Any user who is a both a leader and a member is granted all access.

- Note: Specifying the usr_id can increase or decrease ACL access.

STRATEGY FOR PROFILES, ACLS, QUOTAS

Choosing a System Administrator

- Most trusted person on the system (network).

- Has full access to every file in every directory.

- Must be able to understand user profile database structure, and execute EDIT_PROFILE to manage it.

- Responsible for registering all users in the SAD, creating projects, assigning PAs and defining valid ACL groups.

- Should be available to solve any system problems/emergencies.

- The SA is a user_id. To have multiple SA's create an id and give multiple people the password.

- KNOWLEDGE OF CUSTOMER ENVIRONMENT

STRATEGY FOR PROFILES, ACLS, QUOTAS

- The SAD (System Administrator's Directory) is a database that defines all processes/users who LOGIN to the system.

- The purpose of user profiles is two-fold:

    control access of users entering the system
    define access rights of users on the system

- A user entering the system must specify:

    the system the user wishes to run on
    the user_id s/he is to assume on that system
    any password validating the user_id (may be optional)
    a project_id affiliation (may be optional)

- A user running on the system is identified by:

    a user_id
    a project_id
    an initial attach point in the file system
    a set of ACL groups

STRATEGY FOR PROFILES, ACLS, QUOTAS

Choosing Project Administrators

- Aids the SA in managing the SAD.

- Must be able to understand project structure and execute a subset of EDIT_PROFILE commands to manage it.

- Can grant project membership to any user_id registered in the SAD by the SA.

- Can grant ACL group access to any member of a project s/he administrates. (WITHIN GROUP LIMITS SET BY SAD)

- PA is often an administrative assistent, group leader, teacher, etc.

- Multiple persons managing the same project can be achieved by creating a separate user_id, and giving out the password.

STRATEGY FOR PROFILES, ACLS, QUOTAS

Utilizing the Initial Attach Point

- Is the one ufd on the system the user must have some access to (Use minimum).

- Setting an iap two levels below the MFD makes it easier to limit access (i.e. not a top-level UFD).

- Multiple users can share the same iap, hence the same: login.@ and abbrevs.

STRATEGY FOR PROFILES, ACLS, QUOTAS

Projects

- Projects are a convenient mechanism to group together users who have the similar file system access.

- They provide an accounting entity for external login programs.

- When a PA registers a user into a project, the following are specified:

    a set of project level ACL groups (optional)
    an initial attach point

- Note, every user must be a member of at least one project.

- Project 'Default' is easy to administrate.

*NOTE LIMITED TO 20-30 PROJECTS

STRATEGY FOR PROFILES, ACLS, QUOTAS

Projects

- Projects are a convenient mechanism to group together users who have the similar file system access.

- They provide an accounting entity for external login programs.

- When a PA registers a user into a project, the following are specified:

    a set of project level ACL groups (optional)
    an initial attach point

- Note, every user must be a member of at least one project.

- Project 'Default' is easy to administrate.

*NOTE LIMITED TO 20-30 PROJECTS

STRATEGY FOR PROFILES, ACLS, QUOTAS

Utilizing the Project id

- Projects are most useful for grouping users together:

  for accounting purposes
  for functional organization
  for common file system access

- Project level ACL groups are useful for granting access to certain sets of files for a certain task.

Utilizing the Default Project

- Allowing a user to log into a default project provides the convenience of not typing the project_id.

- Not specifying a project_id removes a level of security.

- Currently cannot change project_id without logout.

STRATEGY FOR PROFILES, ACLS, QUOTAS

Decision: To use projects or not to use projects?

Projects:

- Requires pre-planning for the initial organization.

- Allows a project_id to be 'charged' by the external login program for every user session.

- For systems with a large user community, allows management of the user profile data base and file system access to be delegated to a set of project administrators.

No Projects:

- Much easier to maintain the single default project.

- File access can still be granted to groups of users without separate project affiliations.

STRATEGY FOR PROFILES, ACLS, QUOTAS

Users

- Users are phantoms, terminal or remote processes that log into and use the system.

- Note: NETMAN, SYSTEM never login or have their user_id changed during login, and need not be registered. FTP (FILE TRANSFER PROCESS) needs SYSTEM registered (changes user_id after login).

- When the SA registers a user, the following must be specified:

    a user_id
    a password (optional)
    a set of system level ACL groups (optional)
    a default login project (optional)

STRATEGY FOR PROFILES, ACLS, QUOTAS

Utilizing the User id

- Certain site-dependent information can be encoded in the user_id such as employee number, class level, etc.
  HASH ALGORITHM WORKS BETTER WITH ID's of differing length
- In a networked environment it is easier if each user has the same user_id across all systems.

- System level ACL groups are a function of the user_id. The user obtains this access regardless of the project logged into. They are useful for granting access to system resources.

Utilizing the Password

- Password validates the user_id.

- The SA can force that each user has a password. by using NO_NULL switch in Edit_Password

- No one can see a password, only SA and user can change it.

USER PROFILES

There are two mechanisms for controlling/monitoring a user:

ACL Groups
Projects

ACL Groups

- Mechanism to group users for file system access purposes.
- Very general, ACL group is a set of user_ids.
- Can be used in a hierarchical manner, e.g.
  - .student:alur
  - .teacher:dalurw
  - .department:all
- Convenient mechanism to give a new user certain access to the system. Simply add that user to the ACL group.

USER PROFILES

PROFILE: "Parameters of a user's operating characteristics which create a unique environment for that user"

System Attributes

User_Id (32 CHAR)
User Login Password
Default Project (if none specified at LOGIN)
Maximum 16 ACL Groups
Date/Time of Last Login

(SYSTEM ADMINISTRATORS DATABASE)

- All users must be registered in the SAD (user profile data base) before they can login.
- Each user id is unique. The system administrator must insure uniqueness between 'friendly' systems. * NOTE NETWORK IMPLICATION
- The system administrator may specify that each user_id must have a non-null password associated with it.
- The system administrator maintains the data base with the edit_profile command.
- Defaults to a CLOSED system which ensures all users must be registered before login can occur

USER PROFILES

Projects

- Mechanism to group users for accounting purposes.
- Very general, project is a set of user_ids.
- Allows system administration task to be sub-divided.
- Users must belong to at least one project, but may belong to many.

USER PROFILES

Project Attributes

Maximum 16 ACL Groups
Initial Attach Point

- Project limits are set by SA. This consists of a list of ACL groups the PA may specify for its users.
- PA cannot add a new user to the system, only to a project.
- The project administrator maintains the project data base with the edit_profile command. The PA need not be a member of the projects s/he manages.

USER PROFILES

Decision: To use projects or not to use projects?

Projects:

- Requires pre-planning for the initial organization.
- Allows a project_id to be 'charged' by the external login program for every user session.
- Project_id can be based on:
  Accounting Entity: department, branch, class
  Session Activities: administrative, programming, data entry
- For systems with a large user community, allows management of the user profile data base to be delegated to a set of project administrators.

No Projects:

- Much easier to maintain the single default project.
- File access can still be granted to groups of users without separate project affiliations.

USER PROFILES

Decision: How to use ACL groups?

- Can be specified in a user's system profile and/or project profile.
- A user_id becomes a member of the ACL groups in his/her system profile during every login, regardless of the project_id.
- A user_id becomes a member of the ACL groups in his/her project profile only when logging into that project_id.
- System based ACL groups are an attribute of the user_id.
- Project based ACL groups are an attribute of the project_id.

USER PROFILES
NEW AND MODIFIED COMMANDS

LOGIN <user_id> [<password>] [-ON <system>] [-PROJect <project_id>]

<user_id> <password>

If either is missing, they are prompted for.

<project_id>

If missing, then the default project for the user_id is used, otherwise it is prompted for.

LOGOUT

- CMDNCO>LOGOUT is resumed if it exists, else CMDNCO>LOGIN.

LIST_GROUP, LG

- List ACL groups the current user belongs to.

EDIT_PROFILE CAN SPECIFY FORCE OF PASSWD ON PROMPT LINE INSTEAD OF LOGIN LINE

USER PROFILES
NEW AND MODIFIED COMMANDS

ORIGIN, OR

- Attach to initial attach point for project user is logged into.

CHANGE_PASSWORD, CPW <old_password>

- Change login password for my user_id. New password is requested.

USER PROFILES - GLOBAL OPTIONS

Some global options may be selected when the SAD is initially created.

Non-ACL SAD

A SAD may be created on a non-ACL disk.
There is little protection of a non-ACL sad.
Only the special project 'DEFAULT' may be created.

Projects

Projects need not be created.
If no projects created then special project 'DEFAULT' must be.
All users will belong to project DEFAULT.

ACL groups

A group may be associated with a user:
Regardless of project (system-wide groups),
Because of project (project based groups),
Or both.

USER PROFILES - EDIT_PROFILE

Utility used to maniplulate SAD.

Runs in three modes:

- Initialization mode,
- System administrator mode,
- Project administrator mode.

Many commands with many options.

Allows 'rebuild' to compress/extend files.

USER PROFILES - EDIT PROFILE

The following table lists the commands which the profile editor accepts, along with a list of their respective arguments and option names. Capital letters in the names show the abbreviations, e.g. "AU" is the abbreviation for "Add_User." For more detailed information about each command, type "HELP <command_name>."

Command name	Argument	Options
Add_Project	project	-PA, -CReate_pa, -SIZE -No_Query, -LIKE
Add_User	user	-LIKE, -PROJect, -PROFile, -No_Query -SYStem, -DeFauLT -PassWord, -Verify_NS
ATtach_Project	project	none
Change_Project	project	-PROFile, -SIZE, -LIST -PA, -LIMits
Change_System_Administrator	SA name	-ADD
Change_User	user	-PROJect -LIST -SYStem -PassWord
ConVert_ACL	none	-SYStem, -PROJect, -BOTH
Delete_Project	project	none

Delete_User	user	-PROJect
DeTach_Project	project	none
Force_PassWord	none	-ON, -OFF
HELP	command	none
List_Project	project	-PROFile, -USER, -ALL -OUTput, -TTY, -APPend
List_System	none	-USers, -GRoups, -PROJects, -ALL -OUTput, -TTY, -APPend -DETail
List_User	user	-PROJect, -ALL
No_Null_PassWord	none	-ON, -OFF
REbuild	none	-PROJect, -SIZE
Set_Project_Groups	none	-ON, -OFF
Set_System_Groups	none	-ON, -OFF
Verify_User	user	-ALL

SET_DEFAULT_PROTECTION none

EDIT PROFILE

```
Profile editor [rev 19.0] in initialization mode 02 Aug 82 09:35:56.
SAD does not exist.  Create it? YES
Do you want SYSTEM-wide groups, PROJECT-based groups, or BOTH? BOTH
*** Creating User Validation File.  Projected number of users: 64
System administrator = "SYSTEM".

Create project "DEFAULT"? YES

Set system-wide attributes for user "SYSTEM":
    Password: ADMIN
    Groups: .ADMIN
*** New group added to system: ".ADMIN".
```

User Validation File created 02 Aug 82 09:36:40
    92 entries in prime area; file is 5 records long.

Master Project File created 02 Aug 82 09:36:40

Master Group File created 02 Aug 82 09:36:40

```
Set limits for project "DEFAULT":
    Groups: .USERS
*** New group added to system: ".USERS".

Set attributes for user "SYSTEM" in project "DEFAULT":
    Groups: .USERS
*** New group added to project: ".USERS".
    Initial attach point: <STAFF>USERS

Set profile attributes for project "DEFAULT":
    Groups: .USERS
    Initial attach point: <STAFF>USERS
Project "DEFAULT" created.
    92 entries in prime area; file is 5 records long.
Check entry? YES
```

```
**********************************************************************
*****
Project: DEFAULT                          Administrator: SYSTEM
    One entry in use out of 92.

Master project limits:
    Groups: .USERS

Project profile:
    Groups: .USERS
    Initial attach point: <STAFF>USERS
**********************************************************************
*****
Change entry? N
```

```
> AU DOUG -PASSWORD SPORT -VERIFY_NS

Set system-wide attributes for user "DOUG":
    Groups: .USERS

User "DOUG" added to system.
Check entry? YES

********************************************************************************
*****
System-wide attributes for user "DOUG":
    Groups: .USERS
    Default login project: DEFAULT

Attributes for user "DOUG" in project "DEFAULT":
    Groups: <none>
    Initial attach point: <none>
********************************************************************************
*****
Change entry? AU DALE -PASSWORD COMM
Please answer YES or NO? NO
> AU DALE -PASSWORD COMM

Set system-wide attributes for user "DALE":
    Groups: .USERS
```

```
User "DALE" added to system.
Check entry? NO
> AP DEVELOPMENT -PA DOUG -CREATE_PA

Set limits for project "DEVELOPMENT":
    Groups: .PROGRAM
*** New group added to system: ".PROGRAM".

Set attributes for user "DOUG" in project "DEVELOPMENT":
    Groups: .PROGRAM
*** New group added to project: ".PROGRAM".
    Initial attach point: <STAFF>DEVELOPMENT
Project "DEVELOPMENT" created.
    20 entries in prime area; file is 1 record long.
Check entry? YES

**************************************************************************
*****
Project: DEVELOPMENT                          Administrator: DOUG
    One entry in use out of 20.

Master project limits:
    Groups: .PROGRAM

Project profile:
    Groups: <none>
```

```
    Initial attach point: <none>
*************************************************************************
*****
Change entry? NO
> ATP DEFAULT
> LS -ALL

*************************************************************************
*****
System                          Administrator: SYSTEM
   3 entries in use out of 92.
   System-wide groups enabled.
   Project-based groups enabled.
   Non-DEFAULT projects exist.

                  ******************************
                  *                            *
                  *     Project section        *
                  *                            *
                  ******************************

Project: DEFAULT                          Administrator: SYSTEM
   3 entries in use out of 92.

Master project limits:
   Groups: .USERS
```

Project profile:

Groups: .USERS

Initial attach point: <STAFF>USERS

* *
* * *

Project: DEVELOPMENT Administrator: DOUG

One entry in use out of 20.

Master project limits:

Groups: .PROGRAM

Project profile:

Groups: <none>

Initial attach point: <none>

```
******************************
*                            *
*       Group section        *
*                            *
******************************
```

Group: .ADMIN

Group: .USERS

Group: .PROGRAM

```
                    ******************************
                    *                            *
                    *        User section        *
                    *                            *
                    ******************************
```

System-wide attributes for user "DOUG":
    Groups: .USERS
    Default login project: DEFAULT

System-wide attributes for user "SYSTEM":
    Groups: .ADMIN
    Default login project: DEFAULT

System-wide attributes for user "DALE":
    Groups: .USERS
    Default login project: DEFAULT

```
************************************************************************
*****
```

> DTP

> FPW -ON

> HELP

The following table lists the commands which the

profile editor accepts, along with a list of their respective arguments and option names. Capital letters in the names show the abbreviations, e.g. "AU" is the abbreviation for "Add_User." For more detailed information about each command, type "HELP <command_name>."

Command name	Argument	Options
Add_Project	project	-PA, -CReate_pa, -SIZE -No_Query, -LIKE
Add_User	user	-LIKE, -PROJect, -PROFile, -No_Query -SYStem, -DeFauLT -PassWord, -Verify_NS
ATtach_Project	project	none
Change_Project	project	-PROFile, -SIZE, -LIST -PA, -LIMits
Change_System_Administrator	SA name	-ADD
Change_User	user	-PROJect -LIST -SYStem -PassWord
ConVert_ACL	none	-SYStem, -PROJect, -BOTH
Delete_Project	project	none
Delete_User	user	-PROJect
DeTach_Project	project	none
Force_PassWord	none	-ON, -OFF
HELP	command	none

```
List_Project          project     -PROFile, -USER, -ALL
                                  -OUTput, -TTY, -APPend
List_System           none        -USers, -GRoups, -PROJects, -ALL
                                  -OUTput, -TTY, -APPend
                                  -DETail
List_User             user        -PROJect, -ALL
No_Null_PassWord      none        -ON, -OFF
REbuild               none        -PROJect, -SIZE
Set_Project_Groups    none        -ON, -OFF
Set_System_Groups     none        -ON, -OFF
Verify_User           user        -ALL

> NNPW -ON
> ATP DEVELOPMENT
> LS -ALL

**********************************************************************
*****
System                              Administrator: SYSTEM
    3 entries in use out of 92.
    System-wide groups enabled.
    Project-based groups enabled.
    Non-DEFAULT projects exist.
    Null passwords not allowed.
    Passwords always requested at login.
```

```
*****************************
*                           *
*      Project section      *
*                           *
*****************************
```

Project: DEFAULT   Administrator: SYSTEM

3 entries in use out of 92.

Master project limits:

Groups: .USERS

Project profile:

Groups: .USERS

Initial attach point: <STAFF>USERS

* *
* * *

Project: DEVELOPMENT   Administrator: DOUG

One entry in use out of 20.

Master project limits:

Groups: .PROGRAM

Project profile:

```
    Groups: <none>
    Initial attach point: <none>

                    ******************************
                    *                            *
                    *       Group section        *
                    *                            *
                    ******************************

Group: .ADMIN
Group: .USERS
Group: .PROGRAM

                    ******************************
                    *                            *
                    *        User section        *
                    *                            *
                    ******************************

System-wide attributes for user "DOUG":
    Groups: .USERS
    Default login project: DEFAULT

System-wide attributes for user "SYSTEM":
    Groups: .ADMIN
```

```
   · Default login project: DEFAULT

System-wide attributes for user "DALE":
    Groups: .USERS
    Default login project: DEFAULT
**********************************************************************
*****
> QUIT
```

DEFINITIONS:

ACCESS CONTROL LISTS (ACL)

Provides an alternative to passwords as a means of controlling the use of the file systems. It is a list of users (or sets of users) along with the corresponding access modes associated with that user.

FILE ACCESS MODES
- R - Read Access
- W - Write Access

DIRECTORY ACCESS MODES
- L - List Access
- P - Protect Access
- D - Delete Access
- U - Use Access
- A - Add Access

ALL - RWLPDUA

NONE - Deny all access

ACL GROUP NAMES

One or more users grouped together because of their common access needs to the file system. Group names begin with a period. (Example: .USERS)

PROJECT

Is defined to be a group of users with similiar attributes and system usage. Such as: Persons doing an accounting application in one project, while the group doing program development is in another. It is also a method of allowing the system administrator to delegate responsibility of controlling the users of a PROJECT to the PROJECT ADMINISTRATOR.
Note: Project names do not begin with a period.

USER ID

User Id's are the same as the Rev. 19 login name but they are not neccessarily tied to a UFD Id. The user name can now be a maximum of 32 characters.

INITIAL ATTACH POINT (ORIGIN)

Is a UFD or SUB-UFD in the file system to which the user is attached when they log in. This is a project based attribute. Up to 16 levels deep within the file system and currently restricted to local partitions. (no remote disks for local logins.)

USER REGISTRATION

A method for the system administrator to create a list of users with their personal passwords, ACL group names and projects. It will also provide user verification as part of the login protocol.

USER PROFILES

Loosely defined as those system wide parameters of a users operating characteristics which create a unique environment for that user. This includes the following information:

1. User Id - up to 32 characters
2. User Login Password - up to 16 characters
3. Attributes
   a. project - may be more then one
   b. ACL groups - as many as 16 groups
   c. Initial attach point

ACCESS CONTROL LISTS

Access Control Lists (ACLs) are a way of protecting file system objects (files and directories) from unauthorized access. They provide a passive (requiring no intervention by the accessing user) mechanism for effecting this protection, as opposed to the active mechanism currently provided by passwords. ACLs are simply lists of ordered pairs (<identifier>:<access rights>) which determine what users and groups of users are accorded rights to files and directories.

When a user thinks of protecting objects under his control, there are three basic areas of protection which come to mind: First, the user may want to protect all of his objects in a certain way without taking any specific action. This is known as default protection. Second, the user may want to protect a certain class or category of objects in a common way, adding new objects to the category as they are created and changing access of all objects in a category. We call this protection by access category. Finally, objects may be protected individually according to their specific needs. This is known as specific protection. Together, categoric and specific protection provide explicit protection of objects.

The new ACL system provides the ability to use all three types of protection. Default protection is provided through the directory hierarchy; that is, the access on a directory "trickles down" to all files and subdirectories contained in it (unless they are explicitly protected themselves). Protection by access category is provided by 'NAMED' ACLs. These ACLs may be created and edited independently of the objects which they protect, and when the ACL for a category is changed, the access changes for all the objects in that category simultaneously. For purposes of clarity, in this document named ACLs are always referred to as "access categories." Specific protection is provided by ACLs which are, in effect, simply attributes of the object being protected. They may be manipulated only through identification with the object they protect.

ACCESS CONTROL LISTS

The ACL access rights

ACLs provide access control by associating identifiers with lists of access rights. The rights available and their meanings are as follows:

Right	Applies to	Primary Meaning
Protect	Directories	Accesses may be set and modified.
Delete	Directories	Entries may be deleted from the directory.
Add	Directories	Entries may be added to the directory.
List.	Directories	The contents of the directory may be read.
Use	Directories	The directory may be attached to.
Read	Files	The contents of the file may be read.
Write	Files	The contents of the file may be changed.
ALL	Both	PDALURWX.
NONE	Both	Explicitly deny all access.

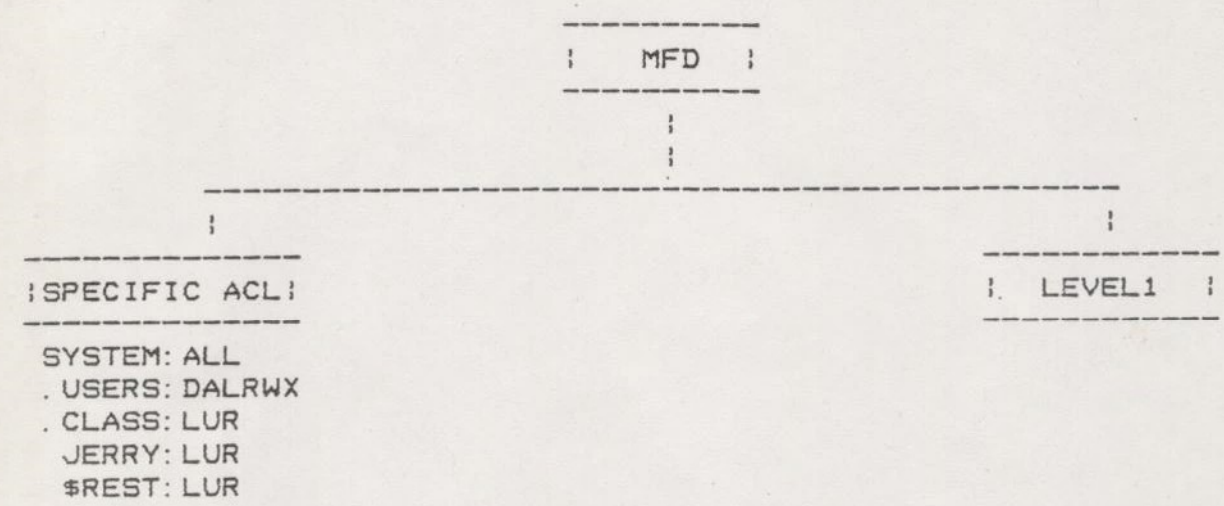
Identifiers

Identifiers in ACL pairs identify either a single user, a group of users, or all users who do not fall into the above two categories. Individual users are identified by their user id. Groups of users are identified by group names, which always begin with a dot (".."). Groups are assigned by system and project administrators and set up at login time. Any user not listed either by name or in a group may be covered by the special identifier "$REST," which is essentially a "catch-all" group.

Access pairs

Throughout this document we will be referring to "access pairs." An access pair is defined as a pair "<id>:<access>," where the <id> must be a legal identifier, the colon must be present, and <access> is a list of access rights as defined above (which may be null). Syntactically, we will always refer to this as an <access_pair>. We will also use the term <access_control_list>, which is simply a list of <access_pair>s separated by spaces.

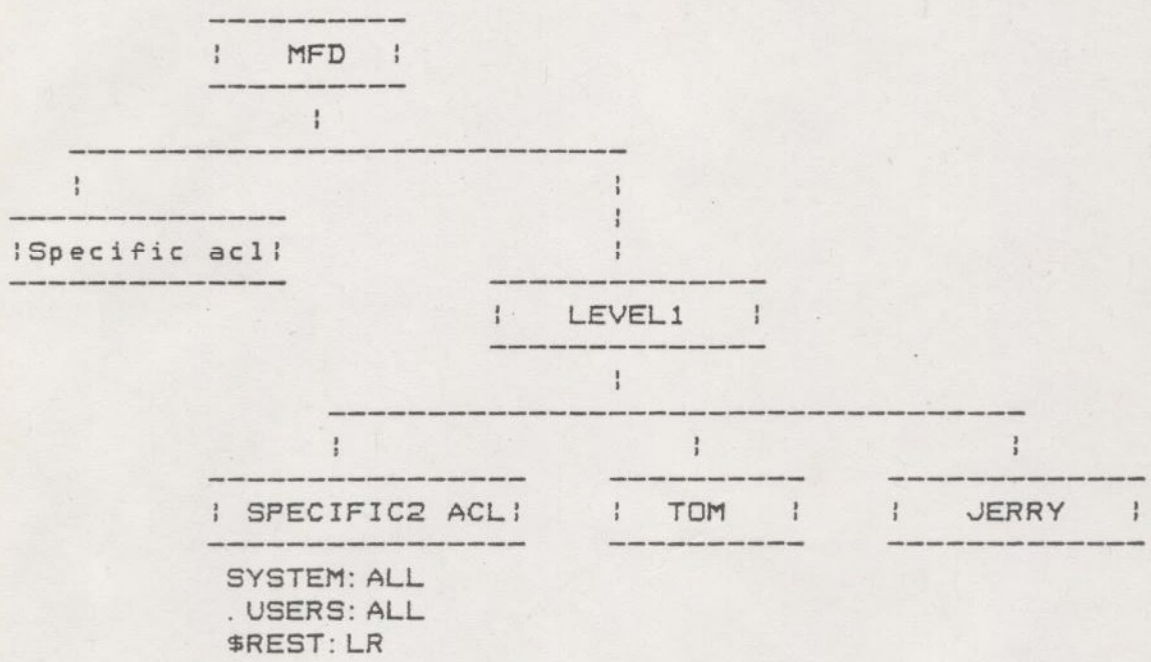
The following example is an example of a default ACL:



The MFD is normally protected by a specific ACL. The MFD is the only directory on the system that cannot have a parent. Once it has been converted to an ACL directory attempts to convert to default protection or to delete an access category will be rejected. In the above example a specific ACL was set to allow the user 'SYSTEM' all access rights, the acl group '.USERS' everything but protect, and all others list, use, and read. This is because of the special identifier '$REST'. As we create new directories on the MFD, these new additions will automatically be protected by this "SPECIFIC ACL". Once the ACL is enabled the system will check the list each time the file or directory is OPENED for access. The file system will compare each entry in the ACL for a match on the "USER_ID" or an "ACL GROUP" name.

In the above example, the "SPECIFIC ACL", which is the default acl of the MFD, allows the user "SYSTEM" full access to the mfd. The users in the ACL GROUP ".USERS" can delete, add, list, read, write, and execute only. The ACL GROUP ".CLASS" and the user "JERRY" can only list, use, and read. For the purpose of this paper "JERRY" will be a part of the ACL GROUP ".USERS". By creating the ACL with JERRY's name listed explicitly, we have limited his access to only LUR. This way we can separate a user from an ACL GROUP for specific purposes. In rev. 18 If a user had the owner password they had complete access to the directory and its files. It was nearly impossible to control just one user. At Rev. 19 we can now specifically eliminate one user by User name. In the above example the RESERVED name "$REST" is used when we would like to reference all users of the file system.

The next example is one level deeper in the tree and is meant to show that an ACL at the next deeper level could increase a user's access rights.

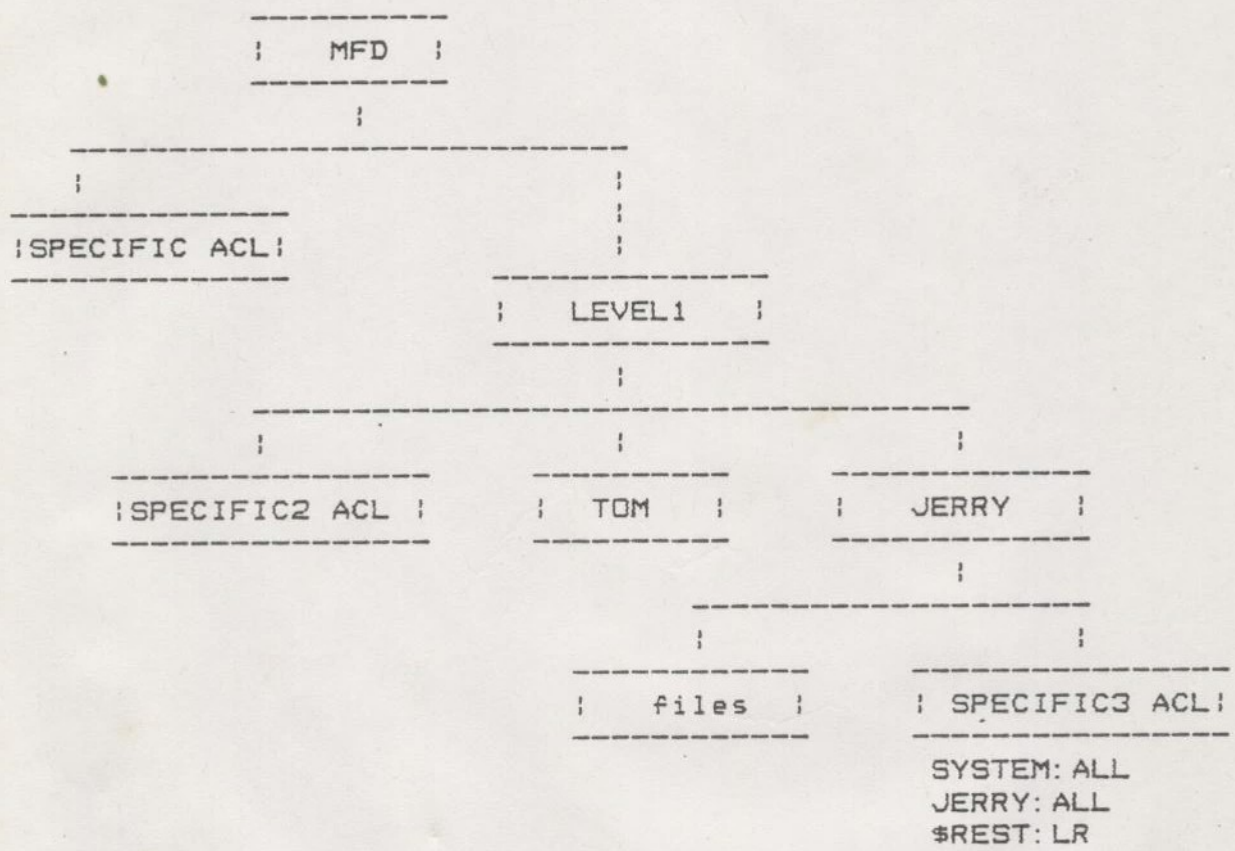


The above example shows an ACL that will protect all files and directories subordinate to LEVEL1. The user "JERRY" will have complete access below this level since he is not explicitly mentioned and he is a part of the ACL GROUP ". USERS".

Each level of directories within the file system could have an ACL. If not otherwise set explicitly, the DEFAULT ACL at the MFD will cover the entire volume. With the two previous examples, the user "JERRY" was allowed only list and read access at the MFD level but at the directory LEVEL1 the new ACL increased the user "JERRY'S" access rights for this part of the file structure.

If a user were to attach to the MFD the DEFAULT ACL would provide the protection. When that same user attached to the directory "LEVEL1" the specific acl at that level would automatically become the one the file system used. After the SPECIFIC ACL is created it essentially becomes the default acl for the files and directories at this level of the file system.

The next example is again one level deeper in the tree and shows how the user "JERRY" has created a "CATEGORY ACL" to protect his files and directories. With this ACL he has allowed the SYSTEM administrator and himself complete access but only allowed the rest of the users read and list access



There are 3 commands that control the Access Control Lists at rev. 19. They are SET_ACCESS, EDIT_ACCESS, and LIST_ACCESS.

Setting access

The SET_ACCESS command is used to specify the complete set of access rights for either a category or a specific object. Its syntax has the following forms:

```
SET_ACCESS <target>
SET_ACCESS <target> <access_control_list>
SET_ACCESS <target> -LIKE <reference>
SET_ACCESS <target> -CATEGORY <category_name>
```

For purposes of this discussion, files and directories are considered to be the same. The NO_QUERY option may be used to suppress any of the questions indicated below. In all cases, if the <target> is a password

directory whose parent is an ACL directory, the <target> is converted to an ACL directory.

* When only a <target> is given
  The <target>, which must be a file, is set to use the default ACL for the directory.

* When a set of access pairs is given
  The action in this case depends on whether or not the <target> exists, what its type is, and how it is currently protected.

  A.) <target> is a file
  The ACL for the file is set as specified. If no specific ACL currently exists for the file, one is created. Note that if the file was category-protected the category will not be changed.

  B.) <target> is an access category
  The old contents of the category's ACL are lost, and are replaced by the specified list of access pairs.

  C.) <target> does not exist
  A new access category is created with the specified list of access pairs.

When the LIKE option is given
In this case, both the <target> and <reference> must be files. If no specific ACL exists for the <target>, one is created. The <target>'s ACL is set to be identical to that of the <reference>. Again, if the <target> was category-protected, the <target> is removed from the category and the category is not changed.

When the CATEGORY option is given
The <target> must be a file and the <category_name> must specify an existing access category. The <target> is added to the access category.

Changing access

The EDIT_ACCESS command is used to modify existing ACLs. Its syntax is:

EDIT_ACCESS <target> <access_control_list>

The <target>, which may be a specifically-protected file or an access category, has its ACL modified to include each of the new <id>s. If an <id> already exists, its <access> is changed. A null <access> indicates that the <id> should be removed from the list. EDIT_ACCESS should not be used on files which are currently default or category-protected, but if it is the user will be queried to determine whether or not he wishes to create a new specific ACL. Example: We want to add a new user to the USER "JERRY'S" category. The command would look like this:

EDAC CATEGORY.ACL NEW.USER:ALRW

If we wanted to change the access rights of the group "$REST", the command would look like this:

EDAC CATEGORY.ACL $REST:NONE

To eliminate the User completely use the command:

EDAC CATEGORY.ACL OLD.USER:

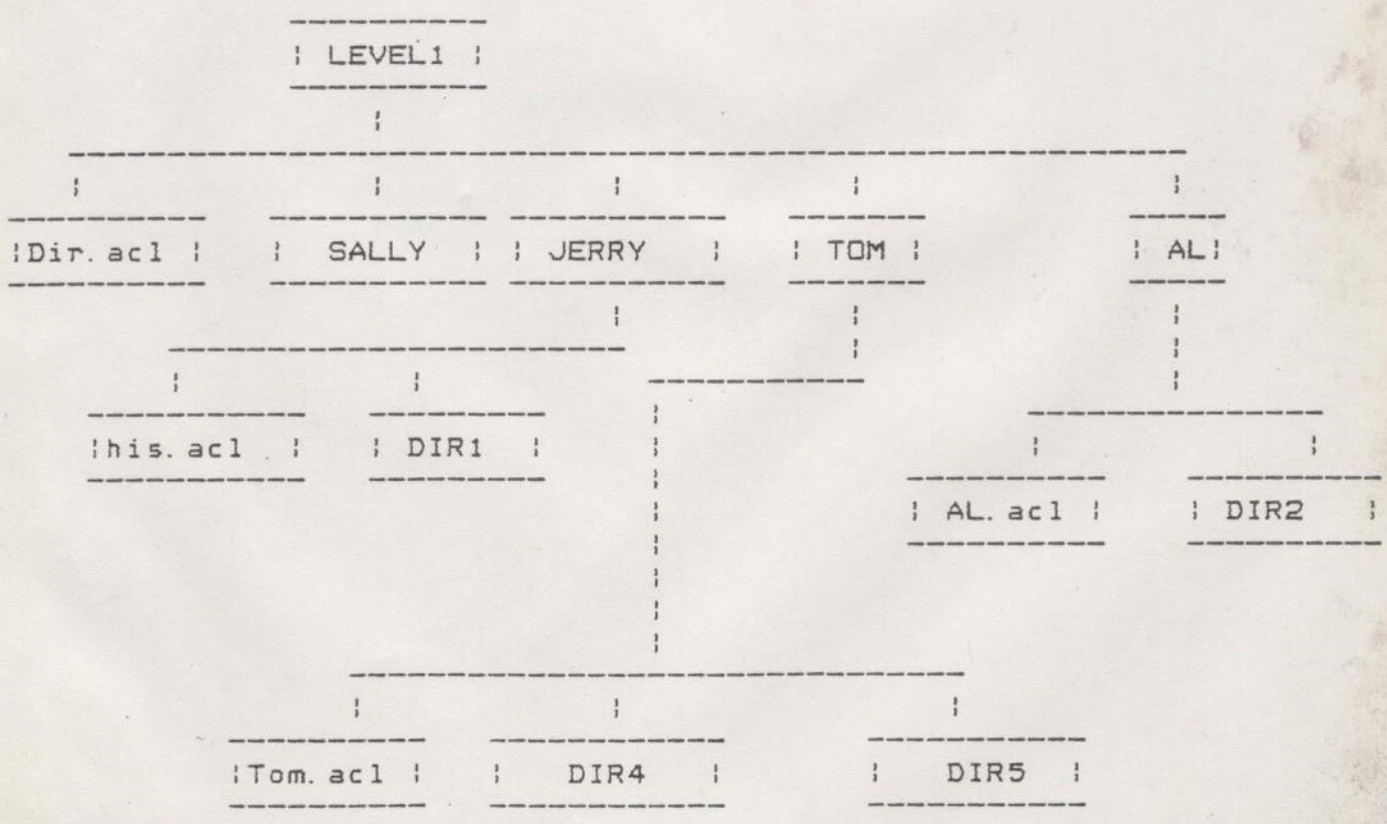
the <NULL> access rights will remove the user from the list.

Examining access

The LIST_ACCESS command allows users to examine the access rights for any file system object. Its syntax is:

LIST_ACCESS [<object>]

If the <object> is omitted, the access rights for the current directory are given. If the <object> is an access category, its ACL is displayed. Otherwise, the ACL protecting the <object> is listed.



In the above example the ACL "DIR.ACL" beneath the directory

"LEVEL1" could become the DEFAULT PROTECTION of this entire subtree with the following command:

```
SAC LEVEL1 -CATEGORY DIR. ACL
```

The user "JERRY" could protect specific files using the command SAC like such:

```
SAC SPECIFIC. FILE JERRY: RWX $REST: NONE
```

The users "TOM" and "AL" could do likewise. The DEFAULT ACL is a global one and would be used to automatically protect any new files and directories that would be created on a day by day basis. If a User needed special protection for a specific file or files they would create their own ACLs to provide it. A DEFAULT acl could protect all entries within a directory or an individual acl could protect a single file or directory with specific access rights. An example might be if the user "JERRY" had a category acl allowing the ACL GROUP ". USERS" read access only but had one file that the user "AL" needed to be able to write. An ACL could be created to allow the user "AL" write access.

In summary, a default acl at the mfd level could protect the entire volume. But in actual practice as we traverse deeper into the tree a users access rights could increase or decrease depending on their individual needs. With the implementation of access control lists this is now possible, just by defining new ACLs at lower levels. We are much more flexible about who can have access and who cannot.

KEN Hopper
INDY

8-3-82

DOUG THOMAS MARK REES

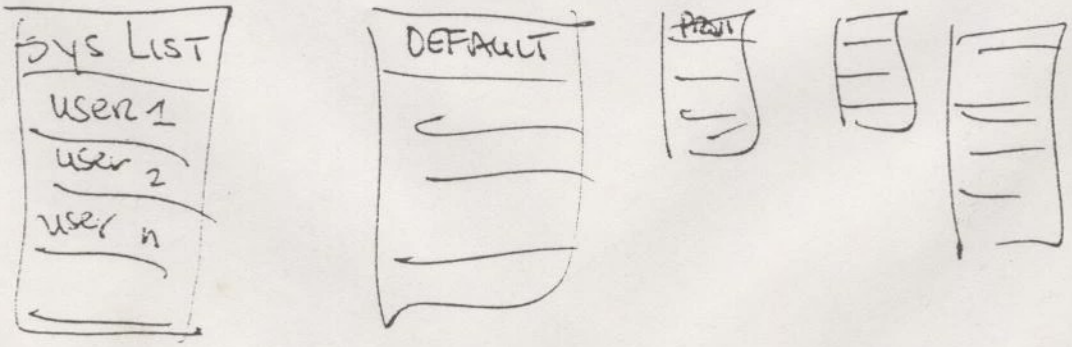
FULL
C DEMO PA
&

USER-ID up to 32 char
not top-level UFD

SPECIFIC · ACL's are unique - change to one not in other
CATEGORY ACL's are universal - change applies to all

SAD

Projects contain < initial attach point (IAP)
ACL Group association



User must be in SYS LIST and a project

Projects Login accountability,
offloads SA functions to PA

SUBJECT: DISK QUOTAS

TIME : 15 MINUTES

MATERIAL: SUPPLIED

OBJECTIVES:

UPON COMPLETION OF THIS SUBJECT THE STUDENT WILL ABLE TO:

1. DESCRIBE THE DISK QUOTA FEATURE OF PRIMOS
2. USE THE COMMAND: SET_QUOTA
3. USE THE COMMAND: LIST_QUOTA
4. DESCRIBE THE DISK HIERARCHIES

DISK QUOTAS

- What are disk quotas?

  - Provides administrative control over disk usage.
  - Quota limits the number of records a single directory or directory sub-tree can use.
  - Units are physical disk records (2kb).
  - May be specifed on a per-ufd basis.
  - Quota of zero means unlimited record usage is allowed.
  - Quota may not be set on an mfd.
  - Requires rev 19 disk format.

Note: No temporary file allowance, nor login/out quota.

STRATEGY FOR PROFILES, ACLS, QUOTAS

Utilizing Quotas

- The ability to impose quotas is controlled by ACLs.

- For a particular sub-tree or partition, quotas may be:

  conserved
  overcommited
  undercommited
  unregulated

- Conserved means that the sum of all the quotas equals the amount of disk space available.

- Overcommited means that the sum of all the quotas exceeds available disk space, and that a 'disk full' condition can occur.

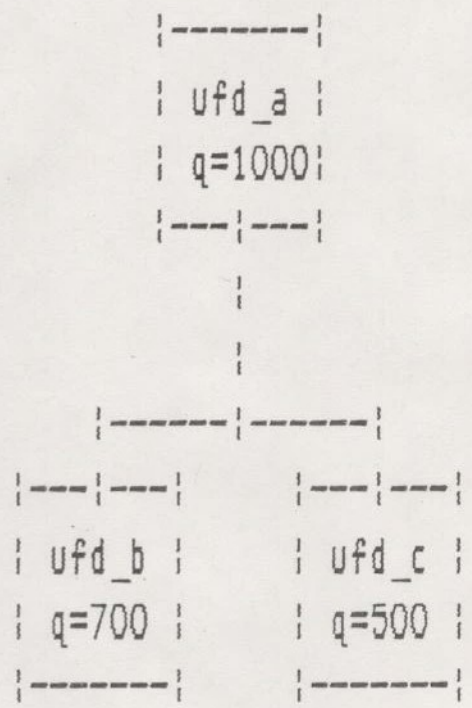
- Undercommited means that the sum of all the quotas is less than the available disk space, thus reserving some amount of space for future needs. Only the 'maximum quota exceeded' condition can occur.

- Unregulated means that one or more UFDs have no quota.

DISK QUOTAS

- When are quotas useful?
    - To meter disk record usage
    - To size directories
    - To administer to file system more effectively

Example:



- If the quota set on ufd_b is 700 records and the quota set on ufd_c is 500 records, and the parent directory ufd_a has a quota of 1000 records, then the total records that can be used by the entire sub-tree (ufd_a, ufd_b and ufd_c) is 1000.

NEW COMMANDS

SET QUOTA

USAGE: SQ <PATHNAME> -MAX <RECORDS>

<pathname>

Name of directory to impose quota on.

<records>

Number of 2kb records allowed for use by this sub-tree.

- Sets a quota on <pathname>.
- Must be owner or have Protect access to superior directory.
- If records is specified as zero, there is no quota.

NEW COMMANDS

LIST QUOTA

USAGE: LQ [<PATHNAME>]

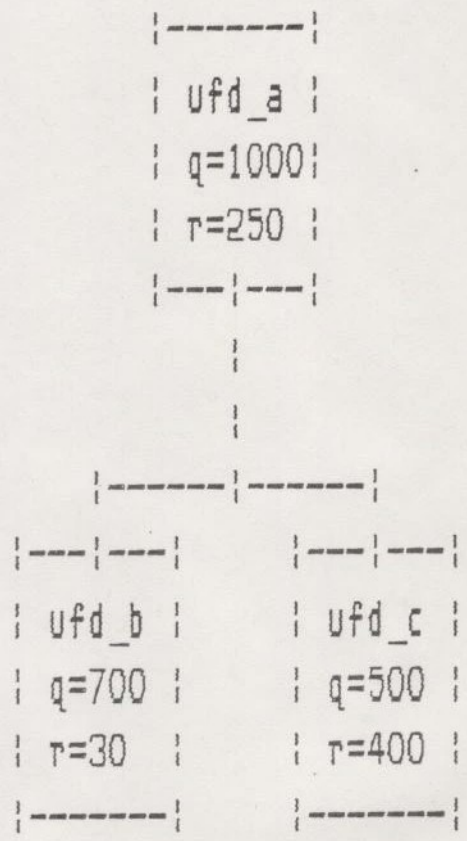
- Reports:

  Maximum quota for the directory and its subdirectories. Total number of records currently used by this subtree. The number of records used in this directory only. (not including subdirectories)

- List access is required.

DISK QUOTAS

Example:



EXAMPLE: LQ UFD_A

Maximum records allowed = 1000

Total records used = 680

Records used in this directory = 250

1 DISK QUOTAS

1.1 When Are Quotas Useful

Quotas are useful to limit the number of records that a directory may use. They may also be used to meter disk record usage and to size directories.

1.2 What Are Quotas

Quotas are limits placed on directory size. The limits are in disk record units. No directory with a quota is permitted to obtain records causing it to exceed its quota. This restriction is enforced on the entire subtree. If multiple quotas are in effect at various levels of the subtree, then the most restrictive quota is enforced.

A quota is always a positive integer value. No quota (quota = 0) allows unlimitted usage. Negative quotas are not allowed.

1.3 Commands

1.3.1 List Quota [<pathname>] (list quota information)

List quota information of directory <pathname>. If <pathname> is omitted, the current attach point is used.

The abbreviation LQ may be used.

Output is of the form:

Maximum records allowed = X
Total records used = Y
Records used in this directory = Z,

where X is the quota max for the directory, Y is the number of records used in the directory tree and Z is the number of records used in this directory level.

Quotas are enforced by never allowing Total records used to exceed Maximum records allowed. Also neither Total records used nor Records used in this directory may be less than one.

1.3.2 Set Quota <pathname> -max <n> (set maximum quota)

Set the maximum quota on directory <pathname> to the value <n> records. A restriction on usage is that the user of the command must have owner or protect access to the parent directory for <pathname>.

The abbreviation SQ may be used for Set_Quota and -m for -max.

Exception condition 'file in use' may be raised when setting a quota on a directory without a current quota. The exception will occur if there are active users of the directory or its subtrees at the time the quota change from zero is requested. Note that this will occur on CMDNCO when the system console is attached there. Also note that this only occurs when changing a quota from a current value of zero to a positive value. If the directory already has a non-zero quota this exception will not occur.

1.4 Using Quotas

Disk record quota are an optional feature for each directory. If there is a maximum quota on a directory, then the system uses the quota restriction. Disk usage meters are recorded regardless.

1.4.1 Maximum Quota

Maximum quota is used to restrict a user to an amount of disk storage specified by the system administrator. This is accomplished by setting the MAX quota on the user's UFD. The subtree for the UFD will not be able to use more records than that maximum.

The maximum quota is an arbitrary value. The sum of the MAX quotas on all top level UFDs can exceed that which is available on the logical disk. In this case, the maximum quota does not guarantee the availability of records in the future. The user still competes with other users for available records, but the competition is controlled.

Inferior directories can also have their MAX quota set. The setting is the same as that for top level directories. The Owner of the immediately superior directory merely sets the value of MAX quota of the target directory. The previous value, if any, is changed to the new value. The MAX quota of the current directory is unchanged. In this way the MAX quota is

non-conserved.

With a non-conserved system the project administrator has a privilege similar to that of the system administrator, namely that he can over-commit his allocated disk records. It should be noted, however, that he is not allowed to actually use more records than he is allocated.

The reason for this arbitrary maximum is to allow a user extra records on a short time basis for listings and other temporary files. Most users will periodically cleanup their directories in order to keep far enough below their maximum to allow ease in working. If this assumption is correct, the administrator can give out more records in maximum quota than actually exist because users will not use their allocated maximums at the same time. This gives better utilization of the disk, since users are sharing records for temporary use. Of course, if a site finds that most users tend to keep close to their maximum quotas and the disk full condition continues to occur, it can set the maximum quotas so that the total is equal to the size of the logical disk.

1.4.2 Quota Hierarchies

A logical disk can be made to use the quota feature by first modifying it with the new FIX_DISK. FIX_DISK will fill in the records used field in UFD headers. The owner of the MFD would then set the MAX quota on any top level UFDs he wishes. In this way he could restrict the number of records used by those UFDs. A directory only becomes a quota directory when its max quota is set.

When a file is created or extended the quota system will check all superior directories to insure that a MAX quota is not exceeded. In the example below B and D are non-quota directories and are therefore not checked. Let us take the example of adding a record to directory D. D has no quota so its quota is not checked. Its parent C has a quota of 100 records and a total records used of 60 records leaving a difference 40 records. 40 records minus the one we are adding is 39 which is greater than zero so we pass the test for directory C. Directory B has no quota so we go on to directory A. A has a quota of 4000 records and a total records used of 4000 leaving 0. Subtracting the record we wish to add leaves -1 records which is negative and we fail the quota test for directory A. Therefore, the record will not be allocated and a error will be returned.

```
Max          4000      A
Dir used     1500      ;
Total used   4000      ;
                       ;
Max             0      B
Dir used     2440      ;
Total used   2500      ;
                       ;
Max           100      C
Dir used       15      ;
Total used     60      ;
                       ;
Max             0      D
Dir used       45
Total used     45
```

SUBJECT: BADSPOT HANDLING

TIME : 15 Mihutes

MATERIAL: Supplied

OBJECTIVES:

UPON COMPLETION OF THIS SUBJECT THE STUDENT WILL BE ABLE TO:

1. Describe the New BADSPOT handling features.
2. Use the new BADSPOT handling feature effectively.
3. Present the new features to customers.

IMPROVED BADSPOT HANDLING

- New badspot file format allows single record badspots, instead of mapping out a whole track. Contains errors to a smaller space.

- Allows remapping of a bad record to a good one.

- COPY_DISK and PHYRST do not understand file system structures. If a badspot is encountered they can create an 'equivalence' block to a goodspot.

- FIX_DISK understands file system structures. It examines mapped entries in badspot file and adjusts file system pointers to include mapped record. This must be done prior to adding the partition.

- PRIMOS does not create badspot entries, nor remap badspots.

- Available only on 19 format disks.

NEW PHYSICAL DISK FORMAT

- Access Control Lists, Disk Quotas, Badspot Handling Require a new physical disk format.

- Starting at 19.0 each partition contains a revision stamp.

- Paritition is converted to 19 format by FIX_DISK or rev 19 MAKE.

- Must use FIX_DISK not FIXRAT on 19 format partitions.

- FIX_DISK will fix any rev partition, and will do a better job than FIXRAT.

- Pre-19 format disks may be run under rev 19, but must be converted before above features can be used.

- Pre-19 format disks can be moved freely between rev 18 and rev 19 systems (without running FIX_DISK or a conversion utility).

- Rev 19 format disks will NOT run under earlier revisions.

- FIXRAT, rev 17 or 18 will break a rev 19 disk.

- Rev 19 disks can be reverted to rev 18 format via a CPL program. (Note, should only be done in case of emergency.)

- New DOS for rev 19.0.

1 BADSPOT HANDLING

1.1 Background

Currently there is no viable scheme to handle badspots on disks in either hardware or software, other than MAKE marking out the entire track as unavailable. This is not a problem for any product except for COPY_DISK/PHYRST, specially in light of new disks with high error rates. A general badspot handling scheme applicable to file system disks is required.

1.2 Problems With Current Badspot Handling

1. Details of disk badspots are currently saved in the BADSPT file as head number, track number pairs. Therefore, even if only one record of a track is bad, all nine records of the track are considered to be bad. The loss of good records hasn't been a problem yet because the disks we are currently selling contain very few badspots. Future disks are expected to have many more badspots.

2. Another problem would have arisen with the introduction of the FIX_DISK utility. If FIX_DISK detects a bad disk record, it will truncate the file or UFD to which the record belongs. If the other records in the track are good, the files and UFDs to which these records belong will be left intact. This is in order to minimize the loss of information due to one bad record. However, as the current BADSPT file can contain only head number and track number of the bad record, the whole track (all nine records) will be marked as bad in the BADSPT file. Because the file system does not look at the BADSPT file, it will be able to run using the good records in the track. Nevertheless, this is an inconsistent condition. COPY_DISK and PHYSAV use the BADSPT file of the source disk to avoid reading bad records. These tracks are not written to the target disk (COPY_DISK) or to magnetic tape (PHYSAV). The new partition will not contain any information in the good records of the bad tracks.

3. When COPY_DISK or PHYRST is restoring a partition, they may not be able to write a record due to a badspot on the new partition. The record which cannot be written at its correct address is lost.

1.3 Solution

The solution selected involves mapping new physical records in place of bad ones and changing all file system pointers to this new record. Essentially COPY_DISK/PHYRST would create the mapping and FIX_DISK will change the file pointers. A new format for the BADSPT file is required. All this is explained in detail in the following sections.

1.3.1 BADSPT file format

1.3.1.1 Old format

Currently, the BADSPT file is a save memory image. The file may be examined and modified by restoring it and referencing it with PSD/VPSD. BADSPT is restored into consecutive memory locations starting at location '1000 and ending at '1000 + 2 * N - 1 where N is the number of bad tracks in the partition. Each word pair in the BADSPT file contains the track and head numbers of a defective track on the disk. The BADSPT file is created by MAKE and is used by FIXRAT/FIX_DISK, COPY_DISK, PHYSAV, PHYRST and AINIT.

1.3.1.2 New Format

```
dcl 1 badspt_file_header,
      2 bad_blk_off fixed bin,   /* offset of the 1st badspt blk */
      2 MBZ fixed bin,           /* must be zero */
      2 file_size fixed bin,     /* size of the badspt file*/
      2 reserve(5) fixed bin;

dcl 1 badspt_blk_header,
      2 bcw,                 /* block control word */
        3 type bit(4),       /* type of this block (badspt blk type =
        3 length bit(12),    /* length of this block */
      2 badspt_blk((badspt_blk_header.bcw.length-1)/2)
        3 track fixed bin,   /* track number */
        3 sector bit(8),     /* sector number+1, (0 means whole track
        3 head bit(8);       /* head number */

dcl 1 eqv_blk_header,
      2 bcw,                 /* block control word */
        3 type bit(4),       /* type of this block (eqv blktype = 1)
        3 length bit(12),    /* length of this block */
```

```
2 eqv_blk((eqv_blk_header.bcw.length-1)/2)
  3 bad_track fixed bin,   /* bad track number */
  3 bad_sector bit(8),     /* bad sector number+1 */
  3 bad_head bit(8),       /* bad head number */
  3 eqv_track fixed bin,   /* equivlant track number*/
  3 eqv_sector bit(8),     /* equivlant sector number+1 */
  3 eqv_head bit(8);       /* equivlant head number *
```

1.3.2 PHYSAV, PHYRST, COPY DISK, and FIX DISK

1.3.2.1 INTRODUCTION

Previously, the way the physical copy utilities handled badspots on a source partition was by reading a file in the MFD called BADSPT which was created by MAKE if a partition was found to contain bad records when it was initialised. This BADSPT file provided information which enabled PHYSAV and COPY_DISK to avoid reading a track which contained a bad record. All records in this track were also marked 'in-use' in the DSKRAT file so that the File Management System was not able to access the bad record.

The new format BADSPT file has two major enhancements over the previous format.

1) Single bad records are marked, rather than whole tracks.

2) An EQUIVALENCE block has been defined, which enables software using the BADSPT file to indicate that it has been able to avoid writing to a badspot by writing the record elsewhere.

Starting from Rev 19, each partition will have a Rev Stamp. The new BADSPT file format will be allowed only on a Rev 19 style partition.

1.3.2.2 BADSPOT HANDLING

1.3.2.2.1 Source Badspot Handling

PHYSAV and COPY_DISK were previously only able to avoid reading bad tracks as marked in the BADSPT files of the source partitions. Therefore, these two utilities have been enhanced to avoid reading individual bad records as marked in the new format BADSPT files.

1.3.2.2.2 Target Badspot Handling

PHYRST and COPY_DISK previously had no notion at all of badspots on the target partitions. Target disk badspot handling has therefore been added to them.
Badspot handling for a target disk involves more than just avoiding bad records on the target disk, as the record that would have fallen on the badspot needs to be written elsewhere, i.e. it needs to be mapped to an available free record. The only way that PHYRST and COPY_DISK can know whether a particular record is free is by checking the DSKRAT file. Therefore, for each badspot entry in the target BADSPT file, a free record must be found from the source DSKRAT file, and the entry that says to which record address the badspot has been mapped to must be stored in the BADSPT file. This is achieved by adding four-word entries to the EQUIVALENCE block of the BADSPT file.

During the restore or disk copy, the programs can then access the EQUIVALENCE block of the target BADSPT file to:

1) map records that would have fallen on a badspot,

2) avoid overwriting those records that have had badspots mapped onto them

N.B. The records that have been mapped contain exactly the same information as the original record (except for the CRA).

1.3.2.2.3 FIX DISK

Badspot handling for FIX_DISK involves fixing the file pointers associated with the bad records to point to the remapping records using the remapped information that is contained in the EQUIVALENCE block of the new format BADSPT file. It also makes available the good records on the target disk which correspond to bad records on the source disk. After FIX_DISK is run, the EQUIVALENCE block of the new format BADSPT file will be removed.

1.3.2.3 INITIAL STATE OF PARTITIONS

As a rule, the format of the target partition will be dictated by the source partition. Also, the badspot handling feature will be available only for rev 19 format partitions. Because the target badspot handling involves using the DSKRAT file of the source partition to find free records, then the DSKRAT file must be correct. If you cannot be sure this is so, then FIX_DISK should be run on the source partition.

1.3.2.4 FINAL STATE OF PARTITIONS

PHYSAV and COPY_DISK will leave source partitions exactly as they were.

PHYRST and COPY_DISK will only leave target partitions as exact copies of the original source partitions if the command line option -NOBADS is used, or if the source partition was a pre rev19 partition.

Otherwise:

1) If the target disk originally had a BADSPT file then afterwards it will contain that BADSPT file with the appended EQUIVALENCE block. Records will have been remapped as indicated by the EQUIVALENCE block.

2) If the target disk did not originally have a BADSPT file then afterwards it will still not contain a BADSPT file.

The exception to these rules is if badspot handling has been turned off by the program - for example if no free records were available on the partition for bad records to be mapped onto. In this case there will be no BADSPT file left on the target disk.

1.3.2.5 Compatibility

COPY_DISK, PHYSAV/RST will handle pre rev 19 partitions exactly as before. In other words, the target partition will be an exact copy of the source partition, and no badspot handling will be provided.
In this case, the message:

WARNING - SOURCE PARTITION IS PRE REV 19
NO BADSPOT HANDLING WILL OCCUR ON PARTITION pdev

will be issued.

1.3.2.6 USER INTERFACE

If badspot handling has taken place during PHYRST or COPY_DISK, then for each affected partition the message:

BADSPOTS HANDLED ON PARTITION pdev

will be put to the terminal at the end. FIX_DISK must be run on that partition before it is used for any reason other than as a target disk for PHYRST or COPY_DISK.

If the situation occurs where PHYRST or COPY_DISK are attempting to map a record round a badspot and there are no free records available, the message:

NO FREE RECORDS AVAILABLE ON PARTITION pdev
OK TO WRITE TO IT WITHOUT BADSPOT HANDLING (YES/NO)?

will be issued to the terminal. If the user types YES then the partition will be copied to without badspot handling, otherwise the program will exit, to allow the user to copy to a different partition with fewer badspots.

Upon finding a BADSPT file (on source or target partitions) which is in some way inconsistent, the message:

BAD BADSPT FILE ON PARTITION pdev - IGNORED

is issued.
If the BADSPT file of a source partition contains an EQUIVALENCE block, then the program will abort with the error message:

BADSPT FILE ON PARTITION pdev HAS AN EQUIVALENCE BLOCK
PLEASE RUN FIX_DISK

N.B. A BADSPT file not marked as special in the MFD is completely ignored.

SUBJECT: FIX_DISK Command

TIME : 20 Minutes

MATERIAL: Supplied

OBJECTIVES:

UPON COMPLETION OF THIS SUBJECT THE STUDENT WILL BE ABLE TO:

1. Describe the FIX_DISK command and its purpose
2. Use the FIX_DISK command to convert a rev. 18 disk to rev. 19 format.
3. Present the new command features to customers.

FIX DISK

- Fix_Disk replaces fixrat for rev 19 format disks.

- Fix_Disk provides all fixrat features plus:

  Checks acl integrity
  Checks disk quota integrity
  Supports new badspot mechanism

≤ Rev 17
- Fixrat must NOT be run on a rev 19 format disk.

- Fix_Disk will not run under Primos II. This may require a change in backup procedures for sites that are currently doing backups under Primos II, and they include running fixrat (recommended).

FIX DISK

FIX_DISK -DISK <physical_device> <control_arguments>

Control Arguments

-FIX	fixes inconsistencies
-ufd_CoMPRession	compresses ufd's (must have -fix)
-COMmand_DEVice	allows comdev to fixed from system console; other users are logged out
-No_Quota	turn off quota checking
-CONVERT_19	convert to 19 format
-LEVEL <number>	directory level for print output
-FILE	print all file names
-MAX_nested_level <number>	maximum depth allowed for nesting
-Auto_Truncation	truncate nesting greater than MAX
-INTeractive	allows reconstruction of bad RAT
-DUFE	delete unknown file entries

FIX DISK

FILE SYSTEM INCONSISTENCIES REPAIRED:
(without deleting or truncating)

- Bad backward pointer in record header.
  If the record really belongs to the file the pointer will be fixed, else the file will be trucated at that point.

- Bad forward or backward pointers in the data level of a DAM file.
  These can generally be fixed if the index pointers are okay.

- Bad record address in index level of a DAM file.
  These can generally be fixed if the data pointers are okay.

- DAM index is too short.
  The index will be extended if there is space in the index record. If not, the file will be truncated.

- DAM index is too long.
  The index should be truncated leaving the file intact.

FIX DISK

- The index level in a record is wrong.
  The correct value should be set.

- The special bit is not set for DSKRAT or BOOT.
  The bit will be forced on.

- The BRA in the header of a record is wrong.
  It will be set to the right value.

- The CRA in the header of a record is wrong.
  It will be set to the right value.

- The word count in the header of a record is wrong.
  It will be set to the right value.

- The UFD header in a directory has the wrong length.
  The entry control word will have the correct length set.

- Quota information that is wrong will be set to the correct value.

1 Overview of FIX DISK

FIX_DISK reads every physical record in every file, UFD, and segment directory, and checks that the information in each record header is consistent with the UFD that contains the record. If the current UFD is a Quota UFD, FIX_DISK also checks the consistency of its quota information. If any inconsistency exists, an error message is generated.

FIX_DISK builds its own record availability table (RAT) while it is traversing the existing file structure and compares its RAT with the DSKRAT file. If discrepancies are found, an error message is generated.

If requested, FIX_DISK will attempt to repair mismatched pointers, correct quota information, truncate/delete defective files, and replace the defective DSKRAT file. The disk will then be in a consistent state.

If requested, FIX_DISK will convert a pre-rev 19 partition into a rev 19 partition. It involves initializing the quota information, changing the BADSPT file to the new format, and creating a rev stamp. If the current partition is a rev 19 partition and a equivlence section exisit in the BADSPT file, FIX_DISK will map the bad records into their equivlence records and fixes the file system pointers to point to the equivlence records. When FIX_DISK has completely traversing the file system structure, the equivlence section of the BADSPT file will be deleted from the BADSPT file. If a badspot is encountered in a rev 19 partition, it will be added to the BADSPT file. If the BADSPT file does not exist, one will be created.

FIX_DISK determines whether a UFD is a quota UFD by examing the maximum quota word in the UFD header. If it is not zero, it is a quota UFD. If the MFD of a partition is a quota UFD, that partition is a quota partition. Otherwise it is not a quota partition and quota information fields are ignored. When FIX_DISK has finished traversing all the subtrees of a quota UFD, the quota information is checked against the records used determined by FIX_DISK. If any inconsistency exists, an error message is generated. If requested, the incorrect quota information is fixed unless the quota used is greater than the maximum quota. Because FIX_DISK cannot and should not decide which records to release to correct the problem, it just marks the quota system as in an inconsistent state. Since the records used of this quota UFD has exceeded its quota, it cannot draw any addition records. The user must delete records or increase the directory's quota to resolve this conflict. FIX_DISK determines whether a UFD is an ACL UFD by the file type field for the UFD in the UFD entry of it's parent. FIX_DISK will verify that for an ACL UFD file entries point to valid ACLs or Access Categories or default, Access Categories point to valid ACLs, and ACLs point back to the same object that points to them. If there is an error and fixing has been requested, for file entries with bad ACL pointers, it will set the ACL pointer of the file entries to the default value. Access Categories or ACLs with errors will be deleted.

FIX_DISK should be run on a regular schedule or whenever there is reason to expect that the file structure or the quota system is damaged.

1.1 Usage:

FIX_DISK -DISK <physical disk> [control arguments]

<physical disk> is the the physical disk number on which FIX_DISK is to be run. The disk MUST be assigned first, unless the -comdev option is being used.

The control arguments are optional. They may be selected in any order from the list below. If no control argument is selected, FIX_DISK only generates error messages if errors are detected.

-fix

Besides printing file structure error messages, FIX_DISK corrects quota information, truncates or deletes defective files, generates a corrected DSKRAT if the current one is bad, and maps the badspot records to the BADSPT file if -fix is specified. If omitted, FIX_DISK will not perform any disk modifications.

-ufd_compression {-cmpr}

If specified along with -fix, FIX_DISK compresses UFDs, eliminates entries flagged as being deleted files or directories.

-command_device {-comdev}

If specified, the disk being fixed is the command disk and FIX_DISK must be invoked via the system console. FIX_DISK will be the only user in the system. If there is any other users, they will be logged out automatically.

-no_quota {-nq}

If specified, it assumes that the partition is not a quota partition and the quota checking mechanism in FIX_DISK will be turned off.

-convert_19

If specified, the current partition will be converted into a rev 19 style disk. If a BADSPT file has already existed, it will be converted into the new format. All quota information is initialized, and all warning/error message related to quota will not be printed. A rev stamp will be created. This option must be used with -fix option.

-level [n]

If specified, the decimal number n that follows is the lowest level in the tree structure in which directory names are to be printed. If omitted, FIX_DISK will print up to level 2 directories (MFD and all directories in MFD file).

-file

If specified, the file names in all directories are printed.

-max_nested_level {-max}

If specified, the decimal number that follows is the maximum depth that directories are allowed to be nested. If omitted, the maximum depth is set to 100. (see -auto_truncation)

-auto_truncation {-at}

If specified, FIX_DISK automatically truncates directories that are nested too deeply in a directory tree. If omitted, FIX_DISK will abort if the maximum depth is reached.

-interactive {-int}

If specified, and the current DSKRAT is bad or missing, questions will be asked so that FIX_DISK can reconstruct a consistent DSKRAT. If omitted and the current DSKRAT is bad or missing, FIX_DISK will abort.

The motivation of implementating this feature is to allow users to replace a bad or missing RAT. FIX_DISK computes the number of records in the partition from the disk number. In case of ambiquity, FIX_DISK asks resolving questions, answerable by either YES or NO.

-dufe (delete unknown file entry)

If specified, all unknown file entries are eliminated. If omitted, all unknown file entries are left untouched, no compressions are performed on the UFDs in which the unknown file entries reside and the DSKRAT will not be altered except in the case of the DSKRAT indicates a particular record is free but that record is actually in use.

The motivation of implementing this feature is to avoid accidental deletion of valid file entries by running the wrong version of FIX_DISK. (e.g. an older version that does not recognize the new file types has to be run.) However there is a drawback of not deleting unknown file entries. The File System advances to the next file entry by using the length field of the current file entry. If the current file entry is garbage, the File System may bypass good file entries by using its length field.

1.2 Description of Error Messages

The backward pointer is bad. It should be YY instead of XX

The backward pointer of a record does not point back to the previous record of the file. In the case of the first record of a file, its back pointer is not zero. If -fix option is specified, the back pointer is fixed to point to the previous record if the BRA word of this record matches the first record address of this file. The file is truncated if the BRA word of this record does not match the first record address of the file.

The Beginning Record Address (BRA) pointer is bad. It should be YY instead of XX

The beginning record address word of the records within the file except the first record should point to the first record of the file. If -fix option is specified, the BRA pointer is fixed.

System file is bad, ignored

An error, which would normally cause deletion of a file, has been found in one of the special files BOOT, MFD, or DSKRAT in the MFD. FIX_DISK aborts.

The current record address (CRA) is bad. It should be YY is XX

The current record address word of this record does not match the current address. This message may be preceded by ten disk error messages because this problem could indicate a disk drive problem. If -fix option is specified, the file is truncated.

UFD nesting exceeds maximum specified

Directories may be nested to a depth of N levels. (default N = 100) FIX_DISK cannot follow the directory tree because the user has nested directories to more than N levels. FIX_DISK ignores this directory unless -at option is specified in which case directories that are nested too deeply in the directory tree will be truncated.

The record header of DSKRAT file is bad

The number of heads is different. It should be YY is XX
The physical record size is different. It should be YY is XX
The DSKRAT header has wrong length. It should be YY is XX

The information contained in the DSKRAT header does not correspond to the information computed from the disk number. Either the disk number is incorrect or the DSKRAT header contains incorrect information. If -int option is omitted, FIX_DISK aborts. Otherwise FIX_DISK asks:

FIX DSKRAT?

A NO response causes FIX_DISK to abort.

The file structure of DSKRAT is bad

This message is obtained if the DSKRAT file contains any bad record pointers, or contains inconsistent information. If either -int or -fix is omitted, FIX_DISK aborts. Otherwise FIX_DISK attempts to reconstruct the DSKRAT file. FIX_DISK computes the number of records in the partition from the disk number. In case of ambiquity, FIX_DISK asks resolving questions, answerable by YES or NO, such as: 40 MB storage module?

FIX_DISK then asks

Split partition?

If part of the disk is to be used for paging then answer YES, otherwise answer NO. If the answer is YES, FIX_DISK then asks

Paging records (decimal)?

The user should type in the number of records to be used for paging.

FIX_DISK then prints the disk number, file records, and paging records.

Partition XX File-records XX Paging-records XX

and asks:

Parameters OK?

If the numbers are incorrect, answer NO and FIX_DISK will attempt to recompute the numbers again.

The father pointer is bad. It should be YY is XX

The father record address word of the first record of a file does not point to the beginning record address of the file in which this file is entered (its father). If -fix option is specified, the father pointer is fixed to point to the BRA of its father.

The forward pointer of the top level index record

of a DAM file is not zero The top level index must only be one record long, therefore the forward pointer of this record must be zero.

The index level of this DAM file is incorrect. It should be YY instead of XX

The index level word of this record is incorrect. It should be zero for SAM files or one less than the previous level for DAM files. If -fix option is specified, the index level word is fixed.

The DAM index is too long to represent the DAM file

The data records of a DAM file are shorter than its index indicates. If -fix option is specified, the index is truncated.

The index of this DAM file is too short to represent the data records

The data records of a DAM file is longer than its index indicates. If -fix option is specified, the index is fixed.

The tree used count is bad. It should be YY instead of XX

The tree used word of this quota UFD does not match the quota used that is calculated by FIX_DISK. If -fix option is specified, the tree used is fixed.

The directory used count is bad. It should be YY instead of XX

The directory used count word for this directory (all the files and nonquota UFDs belong to this directory and the directory file itself) does not match the directory used count that is calculated by FIX_DISK. If -fix option is specified, the directory used count is fixed.

The next index does not match the forward pointer of the current data record

The pointers of the index section and the data section do not agree. If -fix option is specified, the following actions will be taken. The back pointer of the record that is pointed to by the DAM index and the back pointer of the record that is pointed by the forward pointe of the current data record are examined. The record with the back pointer points to the previous data

record will be chosen. If neither back pointer points to the previous record or both back pointer point to the previous record, the file is truncated.

Inconsistent entry. Record = XX, Word = YY

Information in a file entry in a UFD is not self-consistent and cannot be reconciled. If -fix option is specified, the entry of this file is changed to vacant.

Disk read/write error. Record = XX Track = YY Head = ZZ

An error occured while reading/writing record XX. If -fix option is specified, the file is truncated and this badspot record is added to the BADSPT file.

EOF occurs in the middle of an entry

A directory ends in the middle of the last UFD entry. If -fix option is specified, the entry will be deleted.

The Quota system may be incorrect

This message is issued if the partition was changed under DOS. Since DOS doesn't support quotas, there may be directories on this partition with incorrect quota information.

Partition not shutdown correctly during the previous session

This message is issued if the partition was not shutdown with the SHUTDOWN command under Primos. If the system crashed or the disk drive was spun down instead, this message will result.

The word count of record XX is bad

The data word count of a record is not reasonable. For every record except the last record, the data word count should equal the record data size. The data word count of the last record should be between zero and the record data size. If -fix option is specified, the word count is set to record data size.

Physical Device number {-DISK} is missing

The physical device number is not specified in the command line.

Bad physical device number

The physical device number that is specified in the command line is bad.

2 files point to the same record

Two or more files on this partition use the same record. If -fix, the second or later file to reference the record will be deleted.

The Directory/Segdir is longer than 64K!

The maximum size of a UFD/SEGDIR is 64K words. If one exceeds this limit, it will be truncated if -fix is specified.

The BADSPT file is bad, ignored

The BADSPT file that is found by FIXDISK is bad, this file will be treated just like an ordinary file instead of a special BADSPT file.

File entry at word XX does not reference an ACL or Access Category

The ACL pointer of a file entry doesn't point to a valid ACL or Access Category. If -fix, it is changed to the default value.

Access Category at word XX does not reference an ACL

The ACL pointer of an Access Category doesn't point to a valid ACL. If -fix, it is deleted.

Access category at word XX is not pointed at by ACL it points to

The ACL pointer of an Access Category points to an ACL which doesn't point back to it. If -fix, it is deleted.

File entry at word XX is not pointed at by ACL it points to

The ACL pointer of a file entry points to an ACL which doesn't point back to it. If -fix, it is set to the default value.

ACL at word XX does not point to a file entry or Access Category

The owner pointer of an ACL doesn't point to a file entry or Access Category. If -fix, the ACL is deleted.

ACL at word XX is not pointed at by object it points to

The owner pointer of an ACL points to an object which doesn't point back to it. If -fix, it is deleted.

Cannot allocate segment for XX

Fix_disk tried to dynamically allocate a segment for XX and failed. Fix_disk will abort.

SUBJECT: BOOTSTRAP PROCEDURES

TIME : 15 MINUTES

MATERIAL: SUPPLIED

OBJECTIVES:

UPON COMPLETION OF THIS SUBJECT THE STUDENT WILL BE ABLE TO:

1. USE THE NEW BOOTSTRAP PROCEDURE TO INVOKE PRIMOS
2. DESCRIBE THE DIFFERENT OPTIONS OF THE BOOTSTRAP FEATURE
    a. SWITCH 4.
    b. SWITCH 5.

NEW BOOTSTRAP PROCEDURE

- A MORE AUTOMATED BOOT PROCEDURE

- USES SWITCH SETTINGS 4 AND 5

- NEW COMMAND INSTALLED iN CMDNCO: PRIMOS

- SWITCH 4 ON AND 5 OFF

  Do not prompt for 'Physical device='

EXAMPLE:

```
CP> SYSCLR
CP> BOOT 10114
OK:                                  Now at PRIMOS II
'PRIMOS ,PRIRUN'
```

- SWITCH 4 AND 5 BOTH ON

  Fully automatic

  Primos is brought up to the SET_DATE command

EXAMPLE 2

```
CP> SYSCLR
CP> BOOT 14114
OK,
```

0.1 BOOTSTRAP PROCEDURE

At REV 19, Primos may be coldstarted using a procedure that takes the system from depressing the start switch to Primos in one step. This procedure uses additional front panel switch settings (switches 4 and 5) and a new command in CMDNCO (PRIMOS).

0.1.1 Introduction

At REV 18, three software systems are used during coldstart. They are Boot, Primos II and Primos. At REV 19 a fourth system has been introduced, the PRIMOS command. It is installed in CMDNCO and is instrumental in simplifying the coldstart procedure. Subsequent sections of this document specify in detail the software required and procedures to be followed to perform a simplified coldstart at REV 19.

0.1.2 Software Required

1 Boot - must be from a REV 19 Master Disk or created by a REV 19 MAKE.

2 Primos II - must be REV 19 dated 11/18/80 or later. Must be installed in DOS>*DOS64.

3 PRIMOS command installed in CMDNCO.

4 Primos runfiles installed in a directory on the partition to be coldstarted.

0.1.3 Use of Front Panel Switches 4 and 5

1 Switch 4 down, switch 5 down. No change from REV 18 procedure.

2 Switch 4 up, switch 5 down. Do not prompt for 'Physical Device ='.

2.1 Front panel switches are interrogated by software and the device is automatically started up. For example, if coldstarting from physical device 60, switch setting 10114 will startup disk 460, 1060, etc. e.g. you cannot start up disk 20060 this way. Switch setting 10134 will start up disk 660, 1260, etc. Note this may be used only with the top (head 0) partition on a disk.

2.2 When the system prompts 'OK: ', it is running Primos II. At this point the PRIMOS command is used to bring up Primos. The command is issued as PRIMOS <pathname>, where <pathname>

is the pathname of the directory containing the run files for Primos. The Primos command remembers the pathname so the next time typing just PRIMOS is sufficient. Initially the pathname defaults to PRIRUN.

3 Switch 4 up, Switch 5 up. Fully automatic.

Physical device is automatically started up from front panel switch setting as above.

Primos is then automatically brought up from the pathname saved in the PRIMOS command.

0.1.4 Example of Coldstart using Device 460

Assume Primos runfiles are in a directory called OPSYS.

- Power on.
- Turn rotary selector to Stop/Step
- Master clear.
- Turn Address/Data switch to Address.
- Set '10114 in the sense switches (switches 4, 10, 13, 14 up).
- Turn selector to load.
- Press Start.
- Turn selector to Run.
- Type PRIMOS OPSYS on the system console.

To reboot the system:

- Turn rotary selector to Stop/Step
- Master Clear
- Turn Address/Data switch to Address.
- Set '14114 in the sense switches (switches 4, 5, 10, 13, 14 up).
- Turn selector to Load.
- Press Start.
- Turn selector to Run.

SUBJECT: FUTIL REPLACEMENT COMMANDS

TIME : 20 minutes

MATERIAL: Supplied

OBJECTIVES:

UPON COMPLETION OF THIS SUBJECT THE STUDENT WILL BE ABLE TO:

1. Effectively use the FUTIL replacement commands to copy and delete files, directories, and segmented directories from the file system.
2. Present the new features to prospective customers
3. Describe the new commands
    a. COPY
    b. DELETE
    c. List_directory
    d. RWLOCK
    e. PROTECT

NEW FILE UTILITY COMMANDS - COPY

COPY <source_pathname> [<target_pathname>] [<control_args>]

<source_pathname>

A standard treename.

<target_pathname>

A standard treename. If omitted, directory is current; filename is from <source_pathname>.

<control_args>

-Query, -No_Query (distinguish from VERIFY, No_VERIFY)

-LeVel <decimal_number>

-RePorT

-DeLete

-DAM, -SAM

-FORCE

-INCremental

-REPLACE (only copies those that exist)

NEW FILE UTILITY COMMANDS - MOTIVATION

- To provide an easier to use set of file utility commands.
- To realize increased user productivity.
- To provide support for new features such as acls and quotas.
- Implemented to replace futil functions, and support new features.
- Futil and Listf continue to work, but not with acls or quotas.
- Implemented as EPFs.

NEW FILE UTILITY COMMANDS - COPY

<control_args> for attribute copying

-DTM	Preserve original date/time
-PROtect	Preserve acl protection
-QUOTA	Copy maximum quota
-RWLock	Preserve rwlock setting
-Copy_All	Preserve all of the above -DTM, -PRO, -QUOTA, -RWL

- Cannot use COPY on MFD, BOOT or DSKRAT.

- Usage under password directories requires owner access to <target_pathname>. If attributes are to be copied they are protection keys and passwords (directories only). Owner access to <source_pathname> is required if -DL is specified, or if a password protected directory is copied.

NEW FILE UTILITY COMMANDS

- COPY

  Copy files/directories

- DELETE

  Delete files/directories

- LD

  List directory contents

- RWLOCK

  Set read-write lock for a file/segment directory

- PROTECT only for old fashioned Password type dirs

  Set protection for owner/non-owner on files/directories

NEW FILE UTILITY COMMANDS - DELETE

DELETE <target_pathname> [<control_args>]

<control_args>

-Query, -No_Query

-FORCE

-RePorT

- DELETE will not delete MFD, BOOT, DSKRAT.

- Wildcard expansion is controlled by command processor, will query unless -no_verify is specified.

NEW FILE UTILITY COMMANDS - LD

LD [<target_object> [<wild_cards>...]] [<control_args>]

<target_object>

Specifies directory pathname, plus first wildcard name.

<wild_cards>

Additional wild cards.

<control_args>

-No_SORT, -SORT_Dtm, -SORT_Name

-ReVerse

-SinGLe_COLumn

-CATegory_Protected [<category_name>]

-DeFaulT_Protected, -SPECific_Protected

-DETail

-PROtect, -DTM, -SIZE

NEW FILE UTILITY COMMANDS - RWLOCK

RWLOCK <target_pathname> <lock> -RePorT

<lock>

Specifies concurrency lock to be set:

SYS	Use system read/write lock (default)
EXCL	N readers OR 1 writer
UPDT	N readers AND 1 writer
NONE	N readers AND N writers

- Only applies to files and segment directories.

- Note: not compatible with SRWLOC.

NEW FILE UTILITY COMMANDS - PROTECT

PROTECT <target_pathname> <owner_access> <non-owner_access> -RePort

<owner_access>, <non-owner_access>

NIL
R
W
D
RW
RD
WD
RWD

- Only useful on password directories, or acl directories that are converted back.
- Replaces the current PROTEC command. Note: not compatible.

NEW FILE UTILITY COMMANDS EPFs

- All the new futil replacement commands are implemented as EPF's (Executable Program Format).

- EPF's are built with a new loader called BIND, which produces a new type of run file (suffix is .RUN) which may be resumed.

- EPF's do not contain absolute addresses. They are recursive and can be shared or relocated easily.

- Execution takes place in segments 4360...4377 of each user's address space. Maximum value for NUSEG is now '357.

- RLS releases the current static mode program. EPF's must be released separately. Breaking an EPF causes the file to remain open until it is RLS'd. Beware of segment consumption.

- For internal use only. Beta test begins (at rev 19.1.) Full release is not commited at this time.

1 FILE SYSTEM UTILITY COMMANDS

1.1 INTRODUCTION

The following sections are intended to provide a complete description of the file system utility commands. The commands are designed to perform the following basic functions:

- File, segment directory, directory, and access category copying.
- File, segment directory, directory, and access category deletion.
- Setting the read/write lock for files and segment directories.
- Displaying the contents of a directory.
- Setting the protection keys for files and segment directories.

These commands are intended to replace, but are not compatible with, the current FUTIL subsystem. See the section, New Command Processor Features, for information about applying these commands to multiple files or directories in a simple manner.

Document conventions

- Lower case text enclosed in angle brackets ("<" and ">") represents an object whose actual value should be substituted, upper case text indicates a literal value. For example, "<date>" means substitute a calendar date and "ALL" would mean use the literal value "ALL".
- Text enclosed in square brackets ("[" and "]") represents optional objects. Two or more objects separated by spaces represent optional choices, two or more objects separated by vertical bars, "|", represent a choice of mutually exclusive options.
- objects followed by "..." represent multiple occurrences of such objects.

1.2 COPY

COPY will copy files, directories, segment directories, and access categories.

Usage: COPY <source_object> [<target_object>] [control_arguments...]

source object

A standard treename specifying the location and name of the object to be copied. Read (R) access is required on this object.

target object

A standard treename specifying the destination and name of the target object. If the target_object is omitted, the target directory is assumed to be the current directory, and the source object name is used for the target name. Append (A) access is required on the directory containing the target object. Delete (D) access is required on the directory containing the target object if the target object already exists.

1.2.1 control arguments

Zero or more control arguments specified in any order from the the following list:

-QUERY, -Q

Specifies that COPY is to request that the user resolve unexpected or potentially dangerous situations. This is the default mode of operation.

-NO_QUERY, -NQ

Specifies that COPY is NOT to request the user's permission but to attempt to resolve those situations in the most intuitive fashion.

-LEVELS, -LV [<dec>]

Specifies that COPY is only to copy down to the level specified by "dec" when copying a directory tree. "dec" is a decimal integer from 0 to 999. If "-LEVELS" is omitted, the default is to copy the entire tree; if "dec" is omitted, the default is 0 (only copy the top level, the directory entry itself and none of its subentries).

-REPORT, -RPT

Specifies that COPY is to report the results of each

successful copy operation.

-DELETE, -DL

Specifies that COPY is to delete the source object once it has been copied. The default is no deletion. This option requires delete (D) access on the source directory.

-DAM

Specifies that all SAM files copied are to be converted to DAM files. The default is to preserve the original file type.

-SAM

Specifies that all DAM files copied are to be converted to SAM files. The default is to preserve the original file type.

-FORCE

Specifies that COPY is to force delete rights for all delete-protected objects selected to be deleted. This includes both a target object that already exists and the source object if "-DELETE" is selected. This argument is most useful when overwriting a directory tree that may contain delete protected objects. This option requires protect (P) access on the appropriate directory.

The default is to request the user's permission to force delete an object, unless "-NO_QUERY" was specified, in which case the protected object(s) will NOT be deleted.

-INCREMENTAL, -INC

Specifies that COPY is only to copy those objects whose dump bit is off (= 0). (I.E. those files that have NOT been dumped to tape.) The default is to copy objects regardless of the dump bit setting.

This argument is intended to provide functionality similar to that provided by the MAGSAV INCREMENTAL command.

Note that if a directory is the object of the command, all entries within that directory are copied, regardless of their dump bit setting.

-REPLACE

Specifies that COPY is to only copy those objects which exist in the target directory.

1.2.2 Attribute copying arguments

The following arguments specify which attributes of selected objects are to be preserved or reset by COPY. If none are specified, the default is to use the system default. If one or more are specified, only those attributes are preserved, the rest will be reset to the system default.

The use of any of these arguments requires protect (P) access on the appropriate directory.

-DTM

Specifies that COPY is to preserve the date/time modified stamp of all source objects copied. The system default is to reset the date/time modified to the current date/time.

When a directory is copied, the use of this argument will cause the date/time modified stamp of each subentry in the directory to be preserved.

-PROTECT, -PRO

Specifies that COPY is to preserve the protection attributes of all source objects copied. This is done by protecting the target object with a specific access control list. The default is to use the default access in the target directory.

-QUOTA

Specifies that when a directory is copied the maximum quota information associated with it and any of its subdirectories is to be copied also. The system default for maximum quota information is no limit, i.e., there is no restriction on the maximum directory size.

-RWLOCK, -RWL

Specifies that COPY is to preserve the read/write locks of the source object. The default is to set the read/write locks to the system default.

Note that only files (i.e., DAM and SAM) and segment directories have user alterable locks, for all other file system types copied the read/write locks will have the system default.

-COPY_ALL, -CA

Specifies that COPY is to preserve all the attributes. It is the same as specifying "-DTM -PROTECT -QUOTA -RWLOCK".

1.2.3 Restrictions

- COPY will not allow the MFD, BOOT, or DSKRAT files of a MFD to be overwritten. In order to copy a boot file to a MFD the user should first RESTore the new boot to memory and then SAve it with the name "BOOT". Note that this restriction does not apply when these files exist in other than a MFD.

1.2.4 Usage under password directories

Under password directories the requirement for access is different. In all cases owner access is needed on the target directory. Delete access is need on the appropriate file if COPY is going to delete it (source if "-DELETE" and/or target if it exists). If "-PROTECT" is specified then all the password parts of protection are copied. Protection attributes include protection keys (files, directories, and segment directories), and passwords (directories only).

The system default for protection keys is rwd nil (owner has all rights, nonowner has none); for passwords owner is blank, nonowner is null.

Copying the passwords of a directory requires owner rights in the source directory, if that directory is a password directory. If the user does not have owner rights COPY will request the user's permission to copy the directory. If "-NO_QUERY" was specified, the directory will be copied without requesting the user's permission. If the directory is copied, it will acquire the system default passwords.

1.3 DELETE

DELETE will delete files, directories, segment directories, and access categories.

Usage: DELETE <target_object> [control_arguments...]

target object

A standard treename specifying the location and name of the object to be deleted. Delete (D) access is required on the target directory.

1.3.1 control arguments

Zero or more control arguments specified in any order from the following list:

-QUERY, -Q

Specifies that DELETE is to request that the user resolve unexpected or potentially dangerous situations. This is the default mode of operation.

-NO_QUERY, -NQ Specifies that DELETE is NOT to request the user's permission but to attempt to resolve those situations in the most intuitive fashion.

-REPORT, -RPT

Specifies that DELETE is to report the results of each successful deletion.

-FORCE

Specifies that DELETE is to force delete rights for all delete-protected objects selected. This argument is most useful when deleting a directory tree that may contain delete protected objects. This option requires protect (P) access on the appropriate directory.

The default is to request the user's permission to force delete an object, unless "-NO_QUERY" was specified.

1.3.2 Restrictions

o DELETE will not delete the MFD, BOOT, or DSKRAT files in a MFD. Note that DELETE may be used to delete these files if they exist in other than a MFD.

1.3.3 Implications

o Query will always be requested for directory and access category deletion, unless "-NO_QUERY" was specified.

o Verification is requested of the wildcards handled by the command processor. The command processor option "-NO_VERIFY" will suppress this.

1.3.4 Usage under password directories

Under password directories the requirement for access is different. In all cases delete access is needed on the target object. If the file does not have delete then owner access is needed on the target directory.

1.4 LD - List Directory

LD displays a directory and, optionally, the various attributes of entries in the directory. The user may select entries based on all the standard command processor ways and also by how the object is protected.

Usage: LD [<target_object> [<wild_cards>...]] [control_arguments...]

target object

Specifies both the directory to be listed, and the first wildcard name. For example, "a>b>@.list" would specify entries in the directory A>B whose names match "@.LIST". If pathname is omitted, "@@" is assumed; that is, all entries in the current directory are selected.

wild cards

Specify additional wildcard names. An entry is selected if it matches either the entryname part of pathname or one of the wild_cards.

1.4.1 control arguments

Zero or more control arguments specified in any order from the following list:

-NO_HEADER, -NHE

specifies that the header line is not to be output. The header line contains the pathname of the directory listed, the access rights (in parentheses), the records used by this directory if available, and the quota used if this is a quota directory.

-SPECIFIC_PROTECTED, -SPECP

specifies that those entries that are specific protected will be selected.

-DEFAULT_PROTECTED, -DFTP

specifies that those entries that are default protected will

be selected.

-CATEGORY_PROTECTED, -CATP [<cat_name>]

specifies that those entries that are protected by the access category "cat_name" will be selected. If "cat_name" is missing then all entries that are protected by access categories will be selected.

-NO_SORT, -NSORT

Specifies that the entries listed not be sorted. The default is to sort by ascending NAME within TYPE. TYPEs are always sorted according to the order: file, segment directory, directory, access category.

-SORT_DTM, -SORTD

specifies that the entries be sorted by descending DTM within TYPE. -SORT_NAME must not also be specified.

-SORT_NAME, -SORTN

Specifies that the entries be sorted by ascending NAME only (not within TYPE). -SORT_DTM must not also be specified.

-REVERSE, -RV

Specifies that the sort order be reversed from its default. Note that the sort order of TYPEs is never affected.

-DETAIL, -DET

Specifies that all attributes be displayed for each entry selected. From left to right these are:

access rights available to this user (for password directories, the protection keys are displayed).

size of entry in physical disk records.

quota of entry in physical disk records (directories only).

type of entry.

setting of concurrency lock on entry (" " for system, "excl" for N readers or 1 writer, "updt" for N readers and 1 writer, and "none" for N readers and N writers).

incremental dump switch ("dmp" if the entry has been dumped).

delete-protection switch ("pr" if protected).

date-time modified.

name of entry.

and type of protection (name of access category protecting entry, or (Specific) for specific protected, or blank for protected by default);

The default output format is to list only the name of each entry, four across. To print a subset of "detail" format information, use one or more of the following options.

-PROTECT, -PRO

Specifies that protection information (access mode, delete-protect switch, and type of protection) be printed for each entry.

-DTM

Specifies that date-time-modified be printed for each entry.

-SIZE

Specifies that size information (size of entry, quota for directories only) be printed for each non-access category entry. A size of -1 will be reported for any entry for which the user does not have R (or L) permission.

-SINGLE_COLUMN, -SGLCOL

Is useful only if the default (names only) format is used. In this case, specifies that names are to be printed one per line instead of four per line.

1.4.2 Usage under password directories

Under password directories the access listed is the protection keys as owner nonowner.

1.5 RWLOCK

RWLOCK will set the read/write concurrency locks for files and segment directories.

Usage: RWLOCK <target_object> [<lock>] [control_arguments...]

target object

A standard treename specifying the object whose read/write lock is

to be modified. This command required protect (P) access on the target directory.

lock

Read/write lock, may be one of the following:

SYS - use system read/write lock (default)
EXCL - N readers OR 1 writer (exclusive OR)
UPDT - N readers AND 1 writer
NONE - N readers AND N writers

1.5.1 control arguments

Zero or more control arguments specified in any order from the following list:

-REPORT, -RPT

Specifies that RWLOCK is to report the results of each successful lock change operation.

1.5.2 Restrictions

o Only files and segment directories currently have user alterable read/write locks. If a wildcard name is specified with no file type selection arguments, only files and segment directories will be selected.

1.5.3 Usage under password directories

Under password directories the requirement for access is different. In all cases owner access is needed on the target directory.

1.6 PROTECT

Set protection keys for files, directories, and segment directories. This command is useful only in password directories.

Usage: PROTECT <target_object> [<owner> [<nonowner>]]
[control_arguments...]

target object

Standard treename specifying the object to be protected. Owner access is needed on the target directory.

owner, nonowner

Protection keys, must be selected from the following list:

nil - no access (default)	rw - read/write access
r - read access	rd - read/delete access
w - write access	wd - write/delete access
d - delete access	rwd - read/write/delete access

Note: the order of letters is not important. I.E. "wd" is the same as "dw".

If either owner or nonowner is omitted, the default is nil - no access.

1.6.1 control arguments

Zero or more control arguments specified in any order from the following list:

-REPORT, -RPT

Specifies that PROTECT is to report the results of each successful protection operation.

1.6.2 Restrictions

o PROTECT requires protect (P) access in the target directory.

1.6.3 Implications

o Although the PROTECT command may be used to modify the protection keys of objects in ACL directories, the keys are ignored when accessing those objects. But if the directory were converted back to a password directory, the changed protection keys would be in effect.

o If a wildcard name is specified with no file type selection arguments, the default will be to select files, directories, and segment directories.