LA-7065-MS
Informal Report

UC-32

Issued: December 1977

Common File System Specifications

September 1977

P. M. Blood
R. D. Christman
M. W. Collins
E. W. Willbanks

los alamos
scientific laboratory
of the University of California
LOS ALAMOS, NEW MEXICO 87545

An Affirmative Action/Equal Opportunity Employer

UNITED STATES
DEPARTMENT OF ENERGY
CONTRACT W-7405-ENG. 36

Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

Microfiche	$ 3.00	126-150	7.25	251-275	10.75	376-400	13.00	501-525	15.25
001-025	4.00	151-175	8.00	276-300	11.00	401-425	13.25	526-550	15.50
026-050	4.50	176-200	9.00	301-325	11.75	426-450	14.00	551-575	16.25
051-075	5.25	201-225	9.25	326-350	12.00	451-475	14.50	576-600	16.50
076-100	6.00	226-250	9.50	351-375	12.50	476-500	15.00	601-up	--1
101-125	6.50								

1. Add $2.50 for each additional 100-page increment from 601 pages up.

CONTENTS

COMMON FILE SYSTEM SPECIFICATIONS

by

P. M. Blood
R. D. Christman
M. W. Collins
E. W. Willbanks

ABSTRACT

The Common File System (CFS) is a mass storage system for the Los Alamos Scientific Laboratory's (LASL) Integrated Computer Network. The function of the CFS is to allow users of computers in the network to store and retrieve large amounts of data. The CFS will have an on-line storage capacity of more than a trillion bits and an archival storage capacity of several trillion bits. This report describes the capabilities provided by the CFS. Particular attention is given to describing the hierarchical file system and the system functions available to utilize it.

1.0. PURPOSE

The Common File System (CFS) will be a node in the Los Alamos Scientific Laboratory's (LASL) Integrated Computer Network (ICN). The function of the CFS will be to allow users of the various worker machines and Input/Output stations in the ICN to store and retrieve data sets that can be shared by users of different computers and operating systems. At the same time it will provide a level of protection that is adequate from the standpoint of both privacy and security issues. The CFS will also provide the capabilities to accomplish an orderly and painless

phaseout of the present Hydra file system, which is inadequate in reliability, storage capacity, and response time. The highest possible priority will be given to the reliability and availability of the CFS.

2.0. CFS DESCRIPTION

2.1. User Interface to the CFS.

This document defines a standard set of functions that will be used by each worker computer to access the CFS. Each worker operating system will in turn use the standard CFS functions to provide a set of commands to be used by the user to access the CFS. These commands can be tailored to the particular needs of the users. This standard CFS interface will be independent of worker machine operating systems and file systems, and will also be independent of particular mass storage systems so that a stable interface can be provided as new mass storage systems are added.

2.2. User View of the CFS.

The user sees the CFS as a safe storage place to which his worker machine data sets are copied or staged. When he wants to use his data sets, he copies them from the CFS storage device to a storage device on the worker computer. Then the data sets are the same as any other worker machine data set. The CFS is basically an extension to the worker machine storage capacity.

Ideally the CFS would provide users with a simple means for the on-line storage of an essentially unlimited amount of data with data access and transmission capability sufficient to provide responses that do not degrade user or worker machine performance. From a practical standpoint, this ideal can best be achieved for a majority of the accesses by using a storage hierarchy in the CFS that is transparent to the user except for response time. This hierarchy will consist of off-line storage of several trillion bits, an on-line mass store of more than a trillion bits, and a much smaller, faster disk storage. The CFS will provide management and access to both on-line and off-line storage. In order to improve access response time for the more active data sets, the CFS will move data sets from one storage level to another according to size and usage. The user will also be allowed to move his data sets off-line or on-line, but not to the disk storage. The user will not be required to specify the storage device his data set resides on in order to access it. Also, the user does not need to be aware of the physical storage media or concerned with the management of data on it. Without any special user action, the system will put archival data sets on the proper storage media to be taken off-line.

The ultimate reliability goal of the CFS is that it be safe enough so that a user can always entrust his only copy of a data set to it for storage. In the real world, of course, this goal can only be approached, and even if it were achieved, not all users would believe it. For this reason facilities will be

provided for copying information. The system will also provide the user a limited capability to physically separate data sets.

The CFS will work on the basis of staging full and, possibly, partial data sets; it will not be capable of working on a record basis. If implemented, the accessing of partial data sets will be based on physical addressing (starting byte address and number of bytes to be transmitted). The CFS will have no knowledge of internal data set structures so no logical addressing, manipulation, or conversion of data sets will be performed. The user can use his data set from any appropriate worker but he has the responsibility of initiating any conversions that may be required.

With the trillion-bit devices now available for storage, the response time for data transfer of on-line data is tens of seconds. For off-line storage the response will be minutes and for disk storage it will be seconds. A priority scheme will be provided so that the worker machine can specify which requests from that machine should be processed first by the CFS. By multiplexing data transfers, most priority requests can be quickly processed. Simple requests for status or data set information can be satisfied in a few seconds.

The CFS will be a secure place to store classified information. The same system will store classified and unclassified data. The CFS appropriately protects all information and logs all legal and illegal transactions for classified data sets.

2.3. Directory Organization and Data Access.

2.3.1. Directory Structure. The CFS directory system structure will be based on the rules of tree structure. A tree is formally defined as a finite set T of one or more nodes such that:

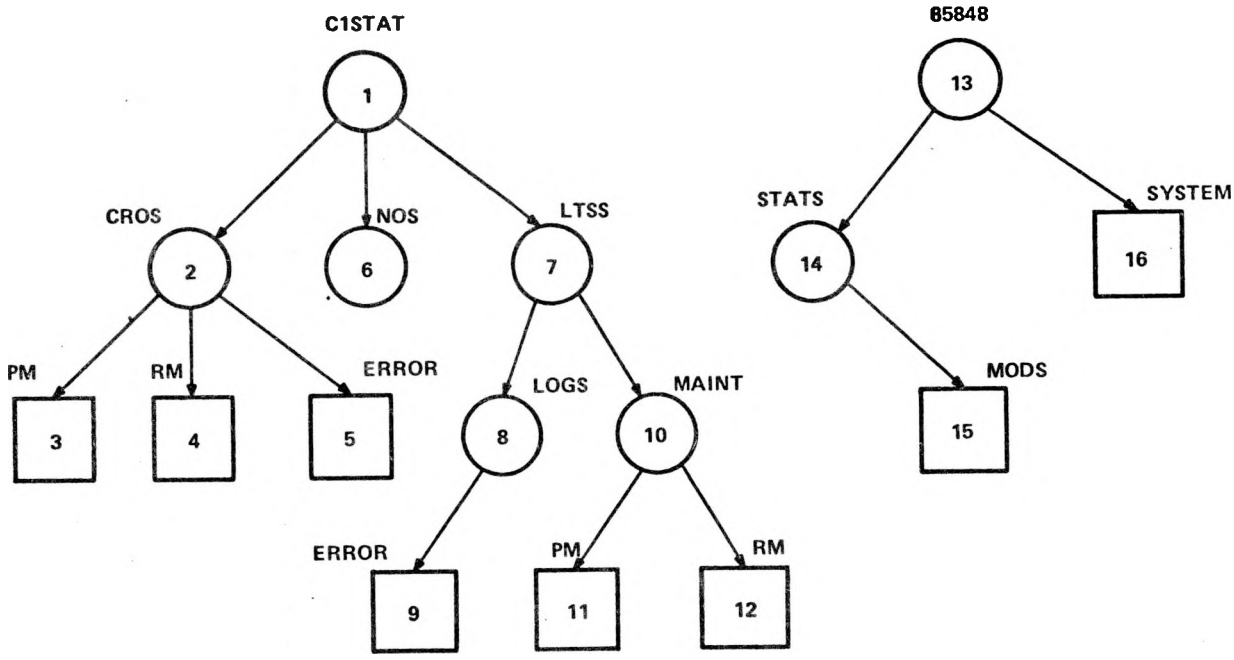
a. there is one specially designated node called the root of the tree,
b. the remaining nodes (excluding the root) are partitioned into $m \geq 0$ disjoint sets $T_1, T_2, \text{---}, T_m$ and each of these sets in turn is a tree. The trees $T_1, T_2, \text{---}, T_m$ are called the subtrees of the tree T.

The CFS directory system will be structured as a "forest" of trees where data sets for logical and functional partitions such as Laboratory divisions, groups, or projects can be organized in specific trees and subtrees. The nodes of a tree will be either directories or data set descriptors. A directory is a list of immediate descendant nodes and is the parent directory of these nodes. A data set descriptor is information pertaining to a particular data set such as location and usage. For each data set saved in the CFS, a corresponding descriptor node is formed.

The root of a tree is called the root directory for that tree. Trees that belong to individuals will typically have user numbers for names of root directories. This will facilitate

access to the tree; the user will not have to specify the root directory. It will also be possible to create root directory names having characters other than numeric. This will facilitate the sharing of data sets between individuals, projects, and groups. A directory that is not the root directory of the tree is referred to as a subdirectory. A subdirectory and its descendants form a subtree.

A tree is constructed by first creating the root directory and then creating subdirectories and/or saving data sets. Many directories or data set descriptors can occur as immediate descendants of any directory. A directory or data set descriptor is an immediate descendant of only one parent directory. A data set descriptor is always a terminal node (having no descendants) while a directory may or may not be a terminal node. Figure 1 shows a sample directory system of two trees. Directories are indicated by round nodes, and data set descriptors are indicated by square nodes. Nodes 1 and 13 are the root directories of the two trees. Arrows always point to immediate descendants.



NODE	PATH	NODE	PATH
1	C1STAT	9	C1STAT/LTSS/LOGS/ERROR
2	C1STAT/CROS	10	C1STAT/LTSS/MAINT
3	C1STAT/CROS/PM	11	C1STAT/LTSS/MAINT/PM
4	C1STAT/CROS/RM	12	C1STAT/LTSS/MAINT/RM
5	C1STAT/CROS/ERROR	13	85848
6	C1STAT/NOS	14	85848/STATS
7	C1STAT/LTSS	15	85848/STATS/MODS
8	C1STAT/LTSS/LOGS	16	85848/SYSTEM

Fig. 1
Sample directory system of two trees.

2.3.2. Path Concept. A path is the name by which any directory or data set descriptor node is specified. The path may also be used to identify the subtree having that node as a root. The path is the compound of all the names of the ancestors of the node plus the name of the node. The names are ordered from the oldest to the youngest. A path always consists of a root directory name, followed by the subdirectory names (if they exist), and ending with the node name. Directories and data sets can be given any name as long as the resulting path is unique. In Fig. 1, the path is given for each node.

2.3.3. Access. Access to directories and data sets is controlled by classification checking and user validation. Classification checking consists of verifying that the classification of the request is greater than or equal to the classification of the node to be accessed. User validation consists of verifying that the user making the request has the required access privilege (for the function requested), and that the correct password has been supplied for password-protected nodes.

The user validation verification is implemented through the placement of user validation entries at directory and data set descriptor nodes. Each user validation entry contains:

a. A user number.
b. The access privilege(s) allowed.
c. Password (optional except for classified nodes).

When a root node is "CREATED," a master user validation entry is automatically made at the root node that contains:

a. The user number of the request that created the node. This user number is considered to be the owner of the resulting tree.
b. All access privileges.
c. Password (optional except for classified node).

All other user validation entries must be explicitly added as desired.

Access privileges contained in a directory node affect that user's access to all the nodes below the directory in the following way. When a node is referenced in a request, the path from the root directory to the referenced node is traversed. The access privileges for that user at each node in the traversal are accumulated.

Accumulating the access privileges is subject to a modifier stored with each user validation entry. This modifier indicates how the access privileges in the current node should be combined with the previously accumulated privileges. There are three possible methods for combining the privileges:

1. The new access privileges are added to the previously accumulated privileges (a logical ØR operation).
2. The new access privileges specify a limit on the previously accumulated privileges (a logical AND operation).
3. The new access privileges replace the previously accumulated privileges (called a SET operation).

A significant aspect of the access method described is that an access privilege at a node applies to all nodes of its subtree unless explicitly modified by an entry in the subtree.

If the specified user does not have a validation entry in a node, a search is made for a zero (or Universal) user number entry. If the Universal entry is found, its access privileges are combined with the users previously accumulated privileges according to the modifier included with the Universal entry. The previously accumulated access privileges remain the same if the specified user does not have an entry or if there is no Universal entry.

The following access privileges are defined:

a. Read Access. Read access applied to a data set descriptor node allows:
   - All or part of the CFS data set to be transferred to a worker machine.
   - Activity and descriptive information for the data set to be read.

   Read access applied to a directory node allows:
   - List of the immediate descendant nodes to be read.
   - Activity and descriptive information for the directory to be read.

b. Execute Access. Execute access applied to a data set descriptor node allows all or part of the CFS data set to be transferred to a worker machine. It is the responsibility of the worker machine to prevent the user of the program from examining the contents of the program. Execute access does not apply to a directory node.

c. Write Access. Write access applied to a data set allows:
   - The CFS data set to be replaced by a data set from the worker machine.
   - Data set to be deleted.
   - Data set name, release date, archival status, and descriptive information to be changed.

   Write access applied to a directory allows:
   - The directory to be removed.
   - Immediate-descendant directories or data sets to be added.
   - Descriptive information to be changed.

d. Append Access. Append access applied to a data set descriptor node allows information to be added at the end of the CFS data set. Append access does not apply to a directory node.

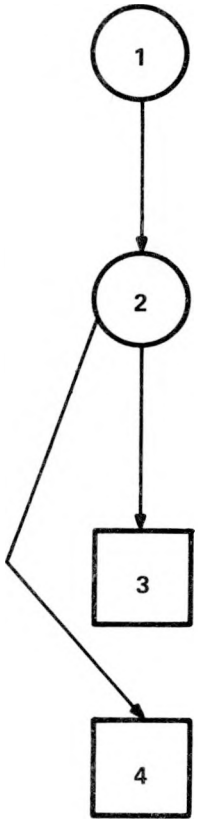
e. Insert Access. Insert access applied to a directory node allows immediate descendant directories or data sets to be added. Insert access does not apply to a data set descriptor node.
f. Bestow Access. Bestow access applied to a data set descriptor or directory node allows user validation entries to be added to the node. An added user validation entry can only have those access privileges that the bestowing user has.
g. Modify Access. Modify access applied to a data set descriptor node allows
   - Data set name, release date, archival status, and descriptive information to be changed.
   - User validation entries to be read, added, changed, and deleted.

Modify access applied to a directory node allows:
   - Descriptive information and classification to be changed.
   - User validation entries to be read, added, changed, and deleted.

Password protection for a node is provided by the password in the last applicable user validation entry encountered as the path is traversed from the root to the accessed node. If no password is contained in the last applicable entry, then no password is required even though passwords may be part of higher-level applicable entries. An applicable user validation entry is any entry along the path that contains the user number specified in the request. Any zero (Universal) user number along the path is also applicable as long as the same node does not contain an explicit entry for the user.

Figure 2 illustrates the use of access privileges and passwords. Note that

a. User 69258 has all access privileges to nodes 1, 2, 3, and 4. Must supply PASSWRD0 for nodes 1, 2, and 3. Must supply PASSWRD4 for node 4.
b. User 76543 has write, read, and bestow access to nodes 2 and 3, and read access to node 4. Must supply PASSWRD1 for nodes 2 and 3. No password required for node 4.
c. User 85848 has write and read access to nodes 2 and 3, and read access to node 4. Must supply PASSWRD2 for nodes 2 and 3. No password required for node 4.
d. User 75145 has read and insert access to nodes 2, 3, and 4. Must supply PASSWRD3 for nodes 2 and 3. Must supply PASSWRD4 for node 4.
e. User 82176 has write access to node 4.
f. All other users have read access to node 4. Must supply PASSWRD4.



	USER	ACCESS PRIVILEGES	PASSWORD	ACCUMULATION TYPE
1	69258	WRITE, READ, APPEND, EXECUTE, INSERT, BESTOW, MODIFY	PASSWRDO	OR
2	76543	WRITE, READ, BESTOW	PASSWRD1	OR
	85848	WRITE, READ	PASSWRD2	OR
	75145	READ, INSERT	PASSWRD3	OR
3				
4	82176	WRITE		OR
	85848	READ		SET
	76543	READ, APPEND		AND
	0	READ	PASSWRD4	OR

Fig. 2
Access privileges and password usage.

2.3.4. Linking. Linking is defined as the connecting of existing nodes in the same tree or different trees so that one node becomes the immediate descendant of the other. The descendant node remains connected to ancestors from previous creation and linking processes so that it now has two or more immediate ancestors and can be accessed by more than one path. A linked node still has only one parent directory node. Even though the linking destroys the tree structure, the concept is preserved since paths to nodes in a linked tree or subtree are as if the linked nodes belong only to the tree through which the access is made. Therefore, linking enables a tree or any of its subtrees to retain its original identity while becoming a subtree of any other tree.

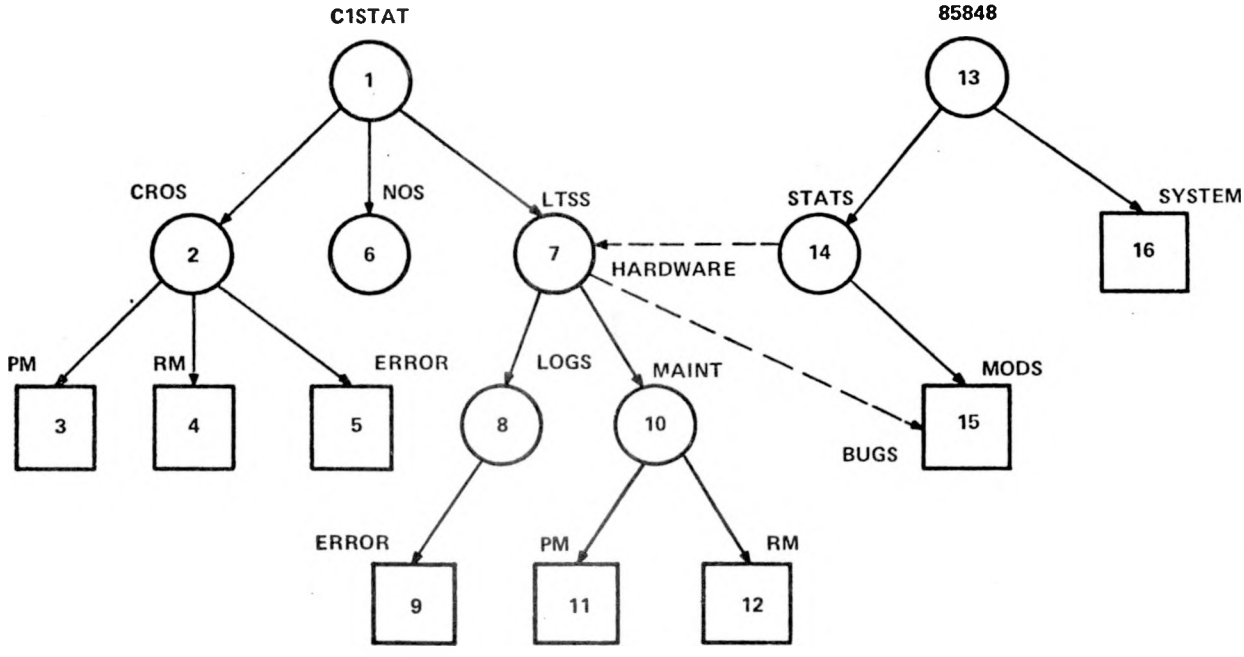
The path definition is modified when links to a node exist such that the path is determined by treating each node as though it had only one immediate ancestor (the ancestor on the desired path).

In the linking of two nodes, the descendant node may be given a different name (an alias). This alias must be used if and only if a path uses this link.

In Fig. 3, the tree structure shown in Fig. 1 is given along with the representation of the following two linking operations:

(a) Node 7 (descendant) is linked to node 14 with "alias" name HARDWARE given to node 7.
(b) Node 15 (descendant) is linked to node 7 with "alias" name BUGS given to node 15.

The subtree C1STAT/LTSS also becomes the subtree 85848/STATS/HARDWARE. The subtree 85848/STATS/MODS (which consists only of node 15) also becomes the subtree C1STAT/LTSS/BUGS. Nodes having multiple paths are listed in Fig. 3. The nodes listed are also roots of subtrees that have multiple paths.



NODE	PATHS
7	C1STAT/LTSS 85848/STATS/HARDWARE
8	C1STAT/LTSS/LOGS 85848/STATS/HARDWARE/LOGS
9	C1STAT/LTSS/LOGS/ERROR 85848/STATS/HARDWARE/LOGS/ERROR
10	C1STAT/LTSS/MAINT 85848/STATS/HARDWARE/MAINT

NODE	PATHS
11	C1STAT/LTSS/MAINT/PM 85848/STATS/HARDWARE/MAINT/PM
12	C1STAT/LTSS/MAINT/RM 85848/STATS/HARDWARE/MAINT/RM
15	85848/STATS/MODS 85848/STATS/HARDWARE/BUGS C1STAT/LTSS/BUGS

Fig. 3
Sample directory system of two trees showing linking.

Access privileges are not accumulated across links. The system converts a linked path to the true path. The true path does not contain any links. In Fig. 3, the system would replace the linked path C1STAT/LTSS/BUGS with the true path 85848/STATS/MODS. Access privileges are accumulated along the true path.

It is likely that initial versions of the CFS will only allow linking from a directory node to a data set descriptor node.

2.3.5. Implications. The reasons for choosing a "tree-like" directory structure are as follows:

a. Data Sets can be organized in a logical hierarchical manner. This provides a convenient means for users to manage, access, and share data sets.
b. Functions may be applied to one or more data sets in a subtree. This allows the logical partitions of a large data set to be stored as individual data sets in a subtree where one or more of the partitions can be accessed by a single operation.

The directory structure for a given application can be very simple or very complex. A minimum directory structure to store a single data set in the CFS must consist of a root directory and one data set descriptor. The length (number of characters) of any given path is the only restriction to the number of levels in a tree.

A version capability can be implemented by the user or worker machine utilizing the tree structure. A directory node would be added having as the path name the desired data set name. Versions of the data set would be saved as data sets in this directory using the version number or other appropriate version I.D. The user or worker machine has the responsibility for accessing, deleting, etc., the desired version.

The Network Security Controller will require a password for user number validation. By controlling directory and data set access with user numbers, the CFS provides another level of protection (in addition to the use of CFS passwords).

2.3.6. Partial List of Capabilities Not Provided by the CFS. Some capabilities not provided by the CFS are:

a. A common user interface. Each worker-machine system has the responsibility for providing a CFS/user interface. Hopefully the worker machines will attempt to provide a common user interface.
b. Access to or update of logical parts of a data set. The CFS will not support any logical structure. Data sets will be stored as a string of 8-bit bytes. If implemented, partial data set access will be based on specifying the relative starting address in bytes, and the number of bytes to be transmitted.
c. Data set conversion. The CFS will not support any data set conversion.

2.4. Functions Available to the Worker Machines.

2.4.1. CREATE a Root Directory. A tree (root only) is formed with the user number or a special name as the root directory. The tree is identified as belonging to the specified division, group, and user (or equivalent) for accounting purposes.

Function parameters

1. User number.
2. Classification of request.
3. Directory classification.
4. Master password for root directory (optional except for classified roots).
5. User validation entries (optional).
6. Special root directory name (optional).
7. Owner division, group, and name (or equivalent).
8. Other descriptive information (optional).

Notes:

1. User validation entries must include passwords if root is classified.
2. Request classification must be greater than or equal to directory classification.
3. A special root directory name will be rejected if the name is already the name of another root node.

2.4.2. ADD a New Subdirectory Node. A new subdirectory node is formed and the name of the directory is entered in the parent directory. Write or insert access to the parent directory is required.

Function parameters:

1. User number.
2. Classification of request.
3. Path of new directory.
4. Password of parent directory if required.
5 New directory classification.
6. User validation entries (optional).
7. Other descriptive information (optional).

Notes:

1. Path to the parent directory is determined from path of the new directory.
2. Request classification must be greater than or equal to parent directory classification.
3. New directory classification must be less than or equal to parent directory classification.
4. User validation entries must include passwords if node is classified.

2.4.3. SAVE a New Data Set. A new data set descriptor node is formed and the name of the data set is entered in the parent directory. Space is allocated and the data set is copied

to the storage system. Write or insert access to the parent directory is required.

Function parameters
1. User number.
2. Classification of request.
3. Path of new data set (descriptor).
4. Password of parent directory if required.
5. Data set classification.
6. User validation entries (optional).
7. Other descriptive information (optional).
8. Data set size.
9. Release date.

Options:
1. The data set can be specified as archival, in which case it will be moved off-line.
2. The physical storage units will be separated into at least two groups. The user will have the option of specifying which group the data set will be stored into. The group cannot be changed by a MODIFY.

Notes:
1. User validation entries must include passwords for classified data set.
2. Path to parent directory is determined from path of new data set (descriptor).
3. Request classification must be greater than or equal to parent directory classification.
4. The data set classification must be less than or equal to the parent directory classification.
5. The data set is effectively released when the release date occurs.
6. A SAVE function will not be treated as a REPLACE function if the data set already exists. The worker machine can provide this capability if desired.

2.4.4. REPLACE an Existing Data Set. New data is saved and then the old data is deleted. Write access to the data set (descriptor) is required.

Function parameters:
1. User number.
2. Classification of request.
3. Path to data set (descriptor).
4. Password for data set if required.
5. Data classification.
6. Other descriptive information (optional).
7. Data set size.

Notes:
1. A REPLACE function will not be treated as a SAVE function if the data set does not exist. The worker machine can provide this capability if desired.

2. Classification of request must be greater than or equal to classification of data set.
3. Data classification can be lower than classification of data set. The CFS will raise the data classification to the data set classification.

2.4.5. APPEND to an Existing Data Set. Data from the worker machine is appended to (added to the end of) the data set. Write or append access to the data set (descriptor) is required.

Function parameters:
1. User number.
2. Classification of request.
3. Path to data set (descriptor).
4. Password for data set if required.
5. Data classification.
6. Other descriptive information (optional).
7. Append size.

Notes:
1. This function will not be implemented for initial versions of the CFS.
2. Classification of request must be greater than or equal to classification of data set.
3. Data classification can be lower than classification of data set. The CFS will raise the data classification to the data set classification.
4. If a data set does not end on a byte (8-bit) boundary, the CFS will pad it to a byte boundary before adding the append data.

2.4.6. GET a Data Set. Requested data is sent to the worker machine. Read or execute access to data set (descriptor) is required.

Function parameters:
1. User number.
2. Classification of request.
3. Path to data set (descriptor).
4. Password to data set if required.

Future implementation:
1. Get part of a data set (starting byte address and number of bytes to be transmitted).
2. Get all data sets in a subtree.

Notes:
1. The implementation of the GET subtree function might better be done at the worker machines.
2. Request classification must be greater than or equal to data set classification.
3. The worker machine must prevent a data set that was read with execute access from being examined by the user.

2.4.7. MODIFY Directory or Data Set Descriptor Node. Information that may be modified includes:

a. Classification (directory node only). Modify access is required.
b. Data set release date (expiration will cause data set deletion). Modify or write access required.
c. Archival status. Modify access is required.
d. User validation entries. Bestow or modify access is required to add entries. Modify access is required to change or delete entries.
e. Owner division, group, and name (root directory only). Modify access is required.
f. Descriptive information. Modify or write access required.
g. Data set name. Modify or write access is required.

Function parameters:
1. User number.
2. Classification of request.
3. Path to node.
4. Password to node if required.
5. Modifications.

Future Implementation: Modify all nodes of a subtree.

Notes:
1. Request classification must be greater than or equal to the node classification.
2. Classification of a data set can be changed only by deleting the data set and then saving it with the new classification.
3. Directory classification cannot be raised above request classification or parent directory classification. Directory classification cannot be lowered below highest classification of immediate descendant nodes.

2.4.8. DELETE a Data Set. Data set descriptor node is deleted and data set space is released. The data set name is removed from the parent directory node. Write access to the data set (descriptor) node is required.

Function parameters:
1. User number.
2. Classification of request.
3. Path to data set (descriptor).
4. Password to data set if required.

Notes:
1. Path to parent directory is determined from path of data set.
2. The request classification must be greater than or equal to data set classification.

2.4.9. REMOVE a Directory Node. The directory is deleted and its name is removed from its parent directory node if it has one. Write access to the directory node is required.

Function parameters:
1. User number.
2. Classification of request.
3. Path to node.
4. Password to node if required.

Notes:
1. Directory must be empty.
2. Path to parent directory is determined from path of directory to be removed.
3. The request classification must be greater than or equal to directory classification.

2.4.10. LINK a Directory or Data Set Descriptor Node. A path is formed from a directory node (ancestor) to a directory or data set descriptor node (descendant) in the same or a different tree. Write or insert access to the ancestor node is required.

Function parameters:
1. User number.
2. Classification of request.
3. Existing path to descendant node.
4. "New linked" path of descendant node.
5. Password to ancestor node if required.

Notes:
1. It is likely that initial versions of the CFS will only allow linking from a directory node to a data set descriptor node.
2. A link to a node only defines an alternate path to the node. It does not guarantee access.
3. An "alias" name may be given for the descendant node on the "new linked" path.
4. Path to ancestor directory is determined from "new linked" path of descendant node.
5. Any existing pointer to a data set (descriptor) or directory node will not be removed by DELETE or REMOVE functions. For this reason, the appearance of linked data sets or directories in the listing of an ancestor directory does not guarantee that the linked data sets or directories exist. Only the UNLINK or DESTROY function can cancel the effect of a LINK function.
6. If the descendant node of a link is deleted and later a new node with the same path is inserted, the link will point to the new node.
7. The request classification must be greater than or equal to ancestor node classification.
8. The descendant node classification must be less than or equal to the ancestor node classification.

2.4.11. UNLINK a Directory or Data Set Descriptor Node. A linked path (created by LINK function) to a directory or data set descriptor node is removed. Write access to the immediate ancestor node on the linked path is required.

Function parameters:
1. User number.
2. Classification of request.
3. Linked path to node (descendant).
4. Password to immediate ancestor node if required.

Notes:
1. The MODIFY function is required to remove any access validation entries that may exist for the "unlinked" path.
2. Path to immediate ancestor node determined from linked path to node (descendant).
3. The request classification must be greater than or equal to ancestor node classification.

2.4.12. LIST a Directory or Data Set Descriptor Node. Listed information will include:

a. Descendant node names (if directory).
b. Status, activity, size, etc., (if data set descriptor).
c. Descriptive information.

Read access to the node is required.

Function parameters:
1. User number.
2. Classification of request.
3. Path to node.
4. Password of node if required.

Options:
1. Obtain list at all nodes in subtree.

Notes:
1. List of directory may also include description of descendant nodes.
2. The request classification must be greater than or equal to node classification.
3. Information will only be returned for nodes of the subtree for which the user has read access.

2.4.13. DESCRIBE a Directory or Data Set Descriptor Node. Will include all information from LIST function as well as information such as classification and user validation entries. Modify access to the node is required.

Function parameters:
1. User number.
2. Classification of request.
3. Path to node.
4. Password of node if required.

Options:
1. Obtain description of all nodes in subtree.

Notes:
1. The request classification must be greater than or equal to node classification.
2. Information will only be returned for nodes of the subtree for which the user has modify access.

2.4.14. COPY a Data Set. Information in the source data set is copied to a new data set. The source data set is left intact. Read access to the source data set (descriptor) is required. Write or insert access to the parent directory of the new data set is required.

Function parameters:
1. User number.
2. Classification of request.
3. Path of source data set (descriptor).
4. Password of source data set if required.
5. Password of new data set parent directory if required.
6. Path of new data set (descriptor).
7. Data set classification.
8. User validation entries (optional).
9. Other descriptive information.
10. Release date.

Options:
1. The new data set can be specified as archival (see 2.4.3., Option 1).
2. The physical storage group of the new data set may be specified (see 2.4.3., Option 2).

Notes:
1. Classification of new data set must be greater than or equal to source data set.
2. Path of parent directory node is determined from path of new data set.
3. Request classification must be greater than or equal to classification of source data set and parent directory of destination data set.
4. New data set classification must be less than or equal to its parent directory classification.

2.4.15. DESTROY a Subtree. All data sets and directories in the subtree are deleted. Write access to each node of the subtree is required.

Function parameters:
1. User number.
2. Classification of request.
3. Path to subtree.
4. Password to nodes if required.

Notes:
1. Path to parent of subtree root node is determined from path of subtree.
2. The request classification must be greater than or equal to classification at each node of subtree.
3. If the user does not have write access to a node of the subtree, all nodes along that node's path will remain in the subtree. Nodes along paths from other nodes may still be deleted.
4. This function will not be available when system first goes into production.

2.4.16. STATUS of the CFS. The status of the CFS and its devices is given along with the status of any incomplete requests for the user number. The status will also include current operator comments.

Function parameters:
1. User number.
2. Classification of request.

2.4.17. ABORT Request. The specified request is aborted.

Function parameters:
1. User number.
2. Request identification.
3. Classification of request.

3.0. OPERATOR VIEW OF CFS

3.1. Overview.

The operator sees the CFS as several storage devices connected to a control computer. This complex in turn is connected to a message and data transport device called a File Transport or Distributor/Collector that has connections to each of the worker computers (Fig. 4).

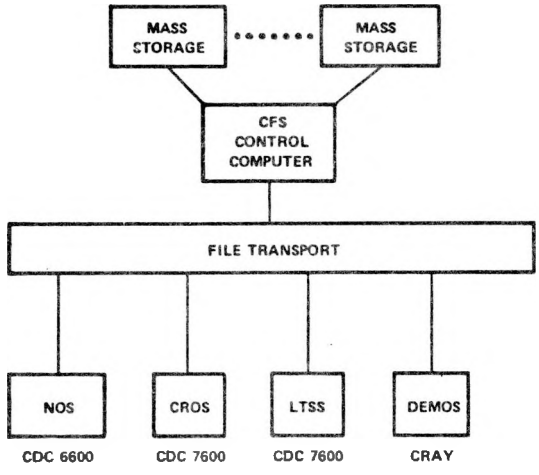


Fig. 4
Common File System configuration.

The major difference between the CFS and Hydra is that there are no direct connections between the various computers and the CFS. Hydra provides a common file function similar to that which will be provided much more capably by the CFS.

In general the operator will have enough information to determine when everything is working properly and enough control to clear up problems and to restart tasks that have hung up. The operator will not have nor need much direct knowledge about the state of the various worker computers. He will need to know the status of the File Transport, however, since it is the only connection between the CFS and the worker computers. The operator console will consist of a CRT device(s) for dynamic displays, a keyboard device(s) for input, and a printer device(s) for hardcopy output. Display and printout for critical problems will be accompanied by an audible alarm. Input commands and output messages will be descriptive English text to the maximum possible extent. Messages will be prefixed with a code referencing a section in the operator's manual that gives a more complete description of the problem.

A "HELP" display will be provided and will list all operator commands and their functions.

3.2. Operator Functions.

General functions available to the operator are:

1. Command descriptions. A list of operator commands and functions will be displayed or printed.
2. CFS device status. Each physical device will have a display adequate to determine if it is operating properly.
3. File Transport status. This display will consist of counts of messages sent, messages received, data blocks sent, data blocks received, and errors.
4. Operator comment. This comment is included in the status messages sent to users of the CFS.
5. Task lists. The list or queue of incomplete tasks can be displayed. Optionally, tasks for only certain machines, users, or devices can be displayed.
6. Unhang or abort tasks or hardware. The operator will be able to restart or abort tasks or hardware operations that have hung up.
7. Change status of a storage device. Storage devices can be turned on or off, initialized, or put into a hold state (CFS accepts requests but does not process them until the device is reactivated).
8. System log dumps. The system logs may be dumped to hard copy, mass storage, or magnetic tape.
9. Active task status. The tasks currently being processed are displayed with enough information so the operator can determine if they are progressing.
10. Commands and functions display. All the commands and functions available for the operating system of the CFS control computer can be displayed.

11. Backup dump initiation. Certain critical information may be copied and saved by initiating backup dumps to alternate storage media.
12. Reload backup dumps.

4.0. OPERATIONS MANAGEMENT VIEW OF THE CFS

4.1. Overview.

The Operations people who manage and control the use of the CFS view it as a valuable resource and, as such, attempt to make sure it is used properly. The general things Operations needs to do are to purge off-line data sets, control data set migration in the storage hierarchy, monitor space and access requirements, generate reports of CFS usage and charging information, monitor CFS performance, and modify the CFS internal scheduling. Logs will be kept containing cost, performance, usage, and activity information. The logs can be copied to hard copy or mass storage. Log entries are neatly formatted for manual analysis and uniquely identified so they can be analyzed by computer programs that generate reports and summaries.

Operations will interface to the CFS by using either the operator console or by special batch jobs executed directly in the CFS control computer. The programs to generate various reports and summaries would probably be written and maintained by the Operations Analysis Section. Operations will also be responsible for maintenance and backup of all system data sets required by the CFS. These include data sets used by the Mass Storage devices and the directory system.

4.2. Available Functions.

The following functions are available to Operations Management.

1. Control data set migration between the various storage systems.
2. Purge off-line data sets.
3. Modify CFS scheduling parameters. This allows operations to change the characteristics of the CFS that affect the response time of user's requests.
4. Modify CFS accounting parameters. This allows the accounting procedures for CFS services to be changed.
5. Analyze performance, usage, activity, and accounting logs.
6. Generate reports. These are reports needed for analyzing performance and for summarizing usage information.

5.0. MAINTENANCE ENGINEER VIEW OF THE CFS

5.1. Overview.

The maintenance engineer sees the CFS as a valuable resource used throughout the ICN. As such it must be kept

running essentially all the time, in a degraded mode if necessary. If a device fails, it must be checked on a stand-alone basis, or on the CFS control computer with background programs. The control computer cannot be dedicated to device repair and checkout except in emergencies.

Logs of equipment malfunctions will be available to indicate hardware degradation and to give the engineers more information about hardware problems. Diagnostic programs written by the maintenance people will be allowed to run in the background while the CFS is processing network requests.

6.0. SYSTEM PROGRAMMER VIEW OF THE CFS

6.1. Overview.

The system programming staff has the responsibility for keeping the CFS running, adding enhancements, updating systems, and resolving hardware/software problems. These responsibilities require that the system programmers can use all facilities that are available to all other classes of users plus additional facilities that allow a more intimate involvement with the operating system. They also need an adequate debug facility so that most system changes and enhancements can be fully checked out without using the CFS for dedicated system time. This should include the checkout required for a new worker machine or storage device to interface to the CFS.

6.2. System Programmer Functions.

The system programmer uses the functions available to all other classes of users. It is important that the vendor-furnished operating system has adequate tools for program and system maintenance, including an adequate high-level system programming language. Additional functions needed are shown below:

1. Background debug facility. The background mode allows a new version of the CFS software to be checked out at the same time the production software is operating. Functions performed in the debug mode include software repair, adding new storage devices, and interfacing new worker operating systems to the CFS.
2. Formatted/unformatted snapshot dumps. These dumps to storage or hard copy can be initiated by a program or as the result of a console command while the system is in debug or operational status.
3. Message/data recording. Messages and data entering and leaving the system at some software level can be dumped to storage or hard copy. Information to be recorded is selected for all sources, for specified worker machine(s), for a specified user or job, for a specified data set, or for a specified physical device.
4. General debug tools are available such as memory display, memory change, data and instruction breakpoints, instruction trace, instruction step mode, request step mode, system freeze, and memory dumps.

5. Privileged functions. Read any directory or data set, read an arbitrary area on any storage device, perform system consistency checks, and step through an I/O sequence.

The special facilities and functions available to the system programmers will be adequately protected; no user programs will be allowed to execute in the CFS, access to the CFS will be only in the CCF, and use of the special functions will be controlled.

7.0. CFS STORAGE AND PERFORMANCE REQUIREMENTS

7.1. Storage Requirements.

Current CCF mass storage requirements for the various LASL programs are over 40 billion words (60-bit words). Of this about 26 billion words are classified as archival. It is estimated that in a few years these requirements will total 105 billion words. Of this about 81 billion words are classified as archival, giving an on-line storage requirement of about 24 billion words. This projected on-line mass storage requirement of 24 billion words is only for the next two years, but a Mass Storage System having a capacity of 25 billion words would probably satisfy the LASL CCF storage requirements for the next four or five years as a greater percentage of the data could reside off-line.

7.2. Prime Shift File Transmission Requirements.

The prime shift file transmission requirements for a peak hour are:

- File accesses - 1,500 per hour
- Data transmitted - 150 million words per hour
- Average file size - 100 thousand words
- Average data transmission rate - 2.5 million words per minute

The file access and data transmission rate required of a Mass Storage System and the associated File Transport System is dependent upon the desired response and the expected peak loading for periods less than an hour.

7.3. Production File Transmission Requirements.

An upper bound on the production file transmission requirements consists of:

- File accesses - Minimal
- Data transmitted - 300 million words per hour
- Average file size - Greater than one million words
- Average data transmission rate - 5 million words per minute

8.0. INTEGRATING THE CFS INTO THE ICN AND PHASING OUT HYDRA

Adding the CFS to the ICN and phasing out Hydra with a minimum impact upon the user community is extremely important. A key factor in this process is that a File Transport System to which both Hydra and the CFS are connected must be available. In addition to allowing users data set access to both Hydra and the CFS, a service will be provided that allows Hydra data sets to be moved to the CFS prior to Hydra's demise. It is probable that Hydra will not be phased out until approximately a year after the CFS is available.

The data transport technique used by Hydra and the worker machines for the FT/X interface is being viewed as a prototype for the interface that will be used with the CFS. This will allow the data transport technique to be evaluated and refined, and should decrease changes necessary for the CFS/worker machine interface.

9.0. GLOSSARY

ALIAS: The different name that may be given to the descendant node in a linking process.

ACCESS PRIVILEGE: The type of access allowed at a node: execute, read, insert, append, write, modify, bestow.

ACCESS VALIDATION: Classification checking and user validation to verify access to a node.

CFS: Common File System.

CLASSIFICATION: The security designation assigned to a node or a user request.

DATA SET: Major unit of data storage and retrieval. Equivalent to a NOS or LTSS file and a CROS fileset. Identified by a path.

DATA SET DESCRIPTOR: Node in tree corresponding to a specific data set stored in the CFS. Contains information about the data set: user validation entries, statistics, physical location, etc. Identified by a path.

DIRECTORY: Node in tree. Contains a list of all immediate descendant nodes. Also contains user validation entries and classification.

ICN: Integrated Computer Network. The interconnection of worker machines, storage systems, I/O stations, and terminal systems in the LASL Central Computing Facility.

LINKING: The connecting of existing nodes in the same tree or different trees so that one node becomes the descendant of the other. This allows for data set sharing that could not be accomplished using the same path.

MASTER ACCESS ENTRY: Entry made in user validation list of root directory when the node is created.

MASTER PASSWORD: Password associated with the master access entry (optional except for classified node).

NODE: A directory or data set descriptor element of a tree.

PARENT DIRECTORY: The immediate ancestor directory node that a newly created node is connected to. All nodes have at most one parent directory. (However, a node may have more than one immediate ancestor due to linking processes.)

PATH: The name by which any directory or data set descriptor node is specified. The path may also be used to identify the subtree having that node as a root. The path is the compound of all the names of the ancestors of the node plus the name of the node. The names are ordered from the oldest to the youngest.

ROOT DIRECTORY: The root node of a tree. A root node is the ancestor of all other nodes in the tree. The root directory is associated with a Division (or the equivalent) for accounting and space allocation purposes.

SUBDIRECTORY: Any directory node in a tree other than the root directory.

SUBTREE: A partition of a tree that when considered alone is a tree. May be a single node. Identified by path to its "root."

TERMINAL NODE: A node of a tree having no descendants.

TREE: A grouping of directory nodes and data set descriptor nodes having a defined structure.

USER VALIDATION: Verification that a user validation entry (for a node) can be found that satisfies the access privilege and password requirements.

WORKER COMPUTER: A computer in the ICN that executes user jobs or that manages terminals and will make requests to the CFS for data sets or information about data sets or directories.