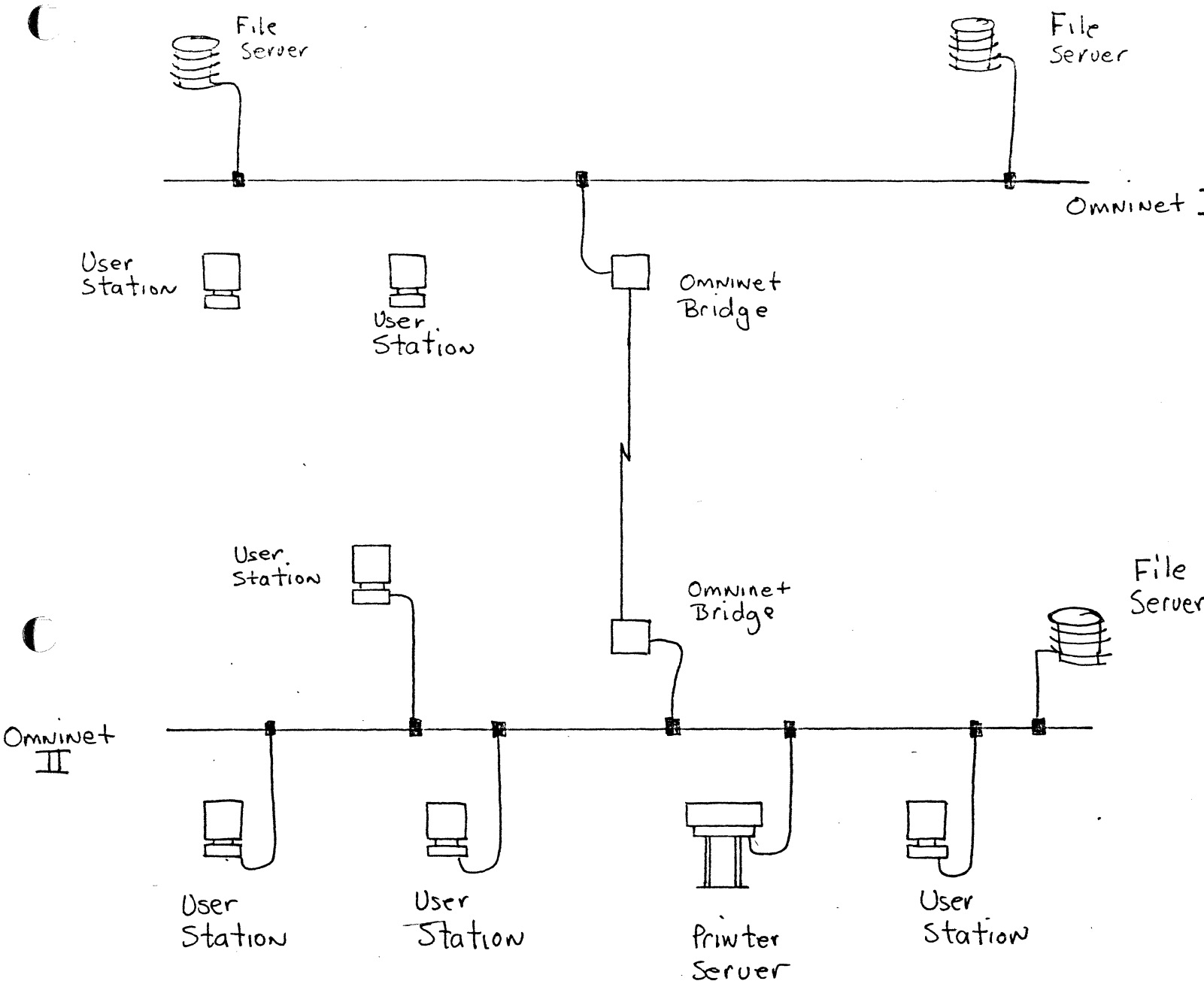
# What Is Constellation II

Constellation II is a mechanism for providing a heterogeneous group of micro computers the ability to share network resources. Access to these resources must be easy to use and provide consistent service and functionality to network users.


File Server
File Server
OMNINET I
User Station
User Station
OMNINET Bridge
User Station
OMNINET Bridge
File Server
OMNINET II
User Station
User Station
Printer Server
User Station

Constellation I weaknesses

1. Maximum 16 MB drive support
2. Maximum 16 MB volume support
3. Maximum 63 volumes per drive
4. Limited user name and password capability. Only 4 character user name and 2 character pass words.
5. Complicated installation procedures.
6. Maximum 128 users per network.
7. User and volume tables cannot be backed up.
8. Implementation very Apple II oriented. Difficult to add new processors and operation systems.
9. Single user and multiple user networks behave differently.

Constellation II features.

1. Designed to support many different personal computers on the same network.
2. Maximum 512 MB drive support
3. Maximum 512 MB volume per drive
4. Supports 512 users per network
5. Supports 512 volumes per drive
6. Extensive use of passwords for users, drives, servers, and other network resources.
7. Network configuration tables can be backed-up.
8. Eliminate need for logical drive for drives larger than 16 MB.
9. Supports multiple disk (file) servers on same network.
10. Network booting scheme allows many different processor types to be booted from network.
11. Localization of user and volume information allows network resources to be more transportable. Drives can now be moved from one network to another without loss of data.

Constellation II features

12. Single and multiple user systems operate in a consistent manner.

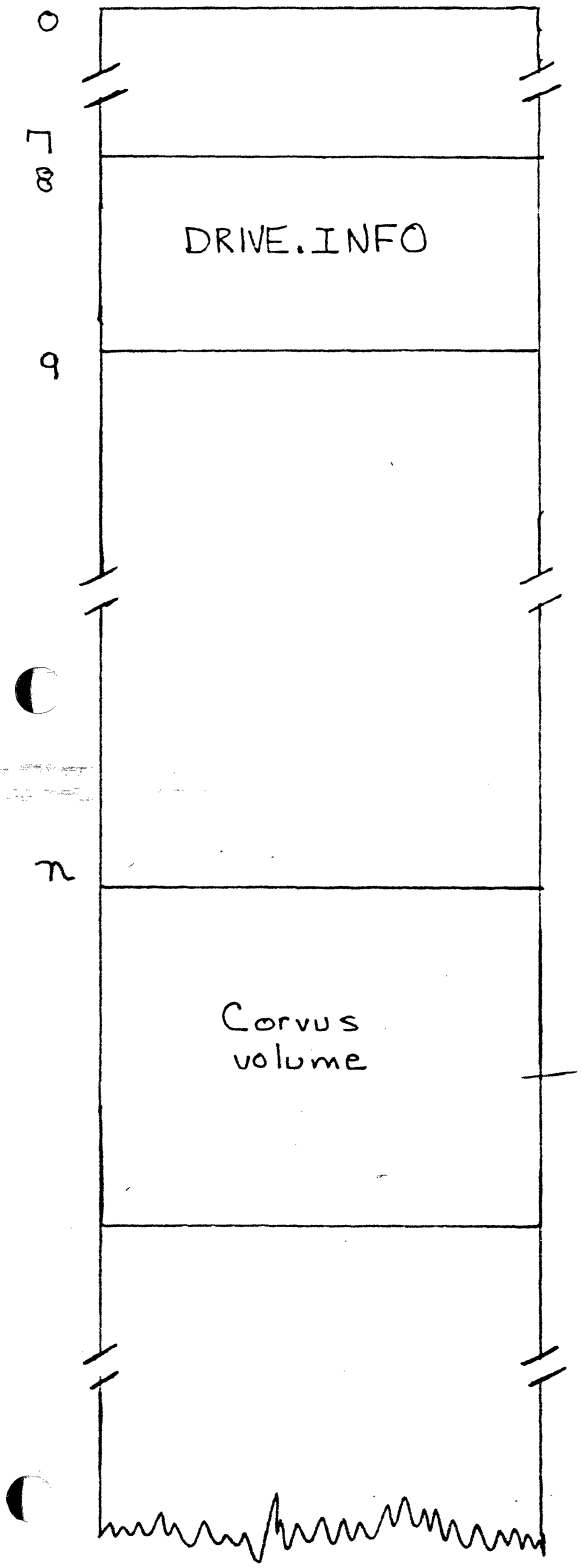
13. User written boot files allow user applications to be invoked directly during power-on for a completely turn-key system.

# Constellation II Utilities

| | |
|---|---|
| _SYSGEN | initializes network and drive tables on an individual drive. Installs cold boot files and Corvus Concept operating system on primary drive. |
| _BOOTMGR | manages cold boot file index blocks and cold boot files residing in Corvus volume. |
| _DRIVEMGR | manages creation and deletion of volumes on any drive |
| _USERMGR | manages network user information. Functions to add, delete, and list users as well as modify user attributes. |
| _ACCESSMGR | controls user access to volumes on the drive. Builds default mount table which identifies all volumes mounted during boot process. |
| _MOUNTMGR | cooperates with operating system to mount and unmount user volumes. |

# Constellation II drive layout

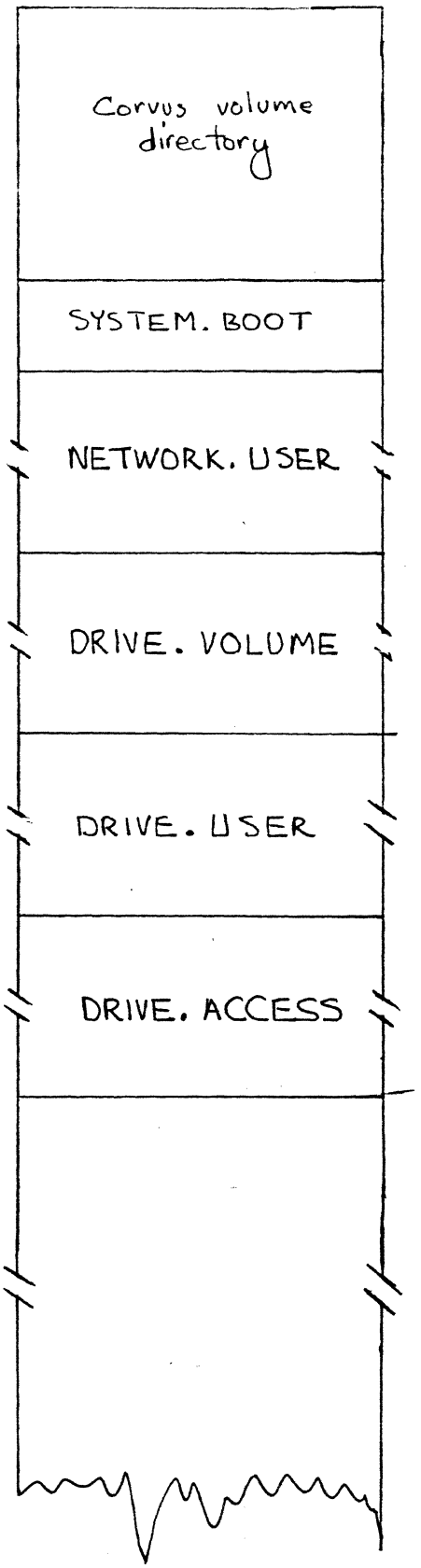
Block




Describes the location of the Corvus volume on the drive. Location of drive name/psw, disk server name/psw, and location of Corvus volume tables..... (1 block)

Contains all Constellation II tables, and boot files. Table/Volume layout is same as UCSD Pascal or Merlyn (Corvus Concept) operating system..... (32 - 6000+ blocks)

Corvus volume layout




The CORVUS volume directory describes each file contained in the Corvus volume. Each system table is represented by a file in the Corvus volume. (6 blocks)

This table contains the address of the cold boot file for each processor type. (2 blks)

Each user of the network is identified via an entry in this table

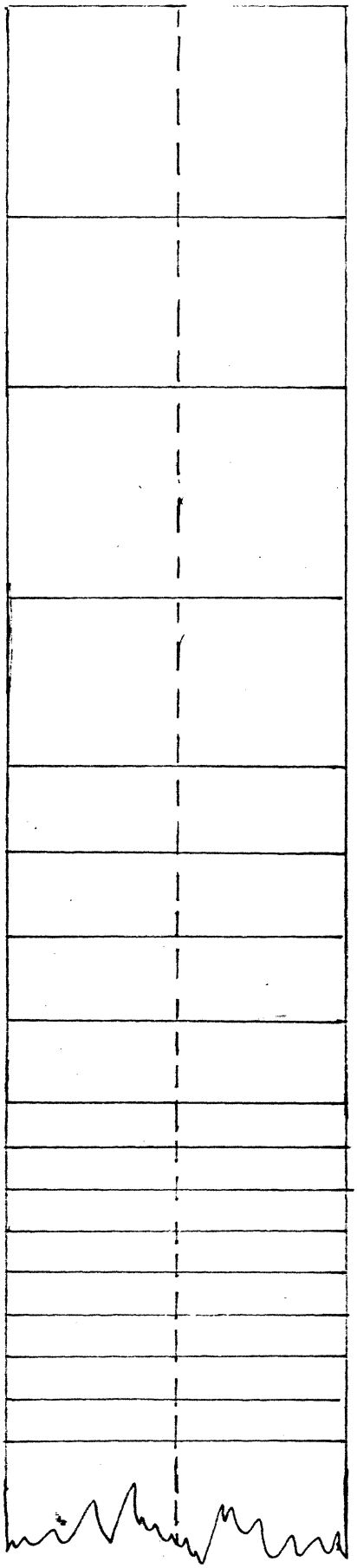
Describes each volume configured on this drive.

Describes each user having access to a volume on this drive.

Describes each volume an individual user has access to on this drive.

Contains cold boot files.

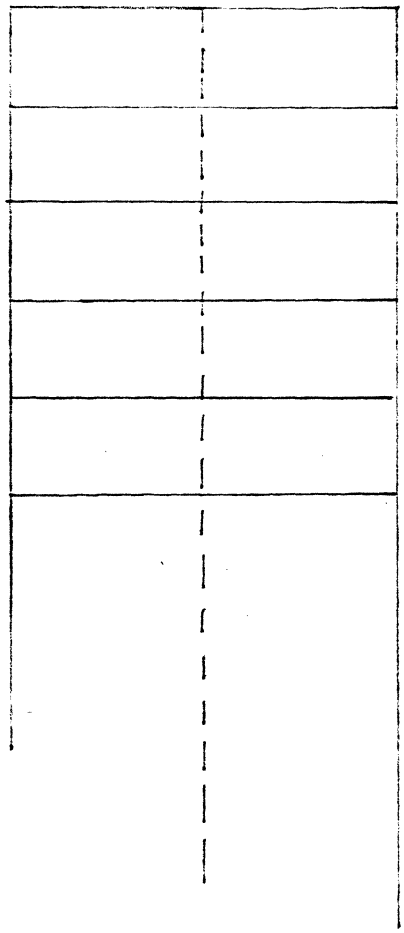
# DRIVE.INFO block layout



| | |
|---|---|
| 0 | Drive name |
| 10 | Drive password |
| 18 | Server name |
| 28 | Server password |
| 36 | Corvus volume address |
| 40 | Corvus volume size (no. of blocks) |
| 44 | Address of cold boot blocks |
| 48 | |
| 52 | Drive online indicator |
| 54 | Drive no. |
| 56 | Drive initialized indicator |
| 58 | NETWORK.USER table size (blocks) |
| 60 | DRIVE.VOLUME table size (blocks) |
| 62 | DRIVE.USER table size (blocks) |
| 64 | DRIVE.ACCESS table size (blocks) |
| 66 | SYSTEM.BOOT table size (blocks) |

512 bytes - area can be reserved for use by specific systems.

DRIVE.INFO block layout (continued)




Address of NETWORK.USER table

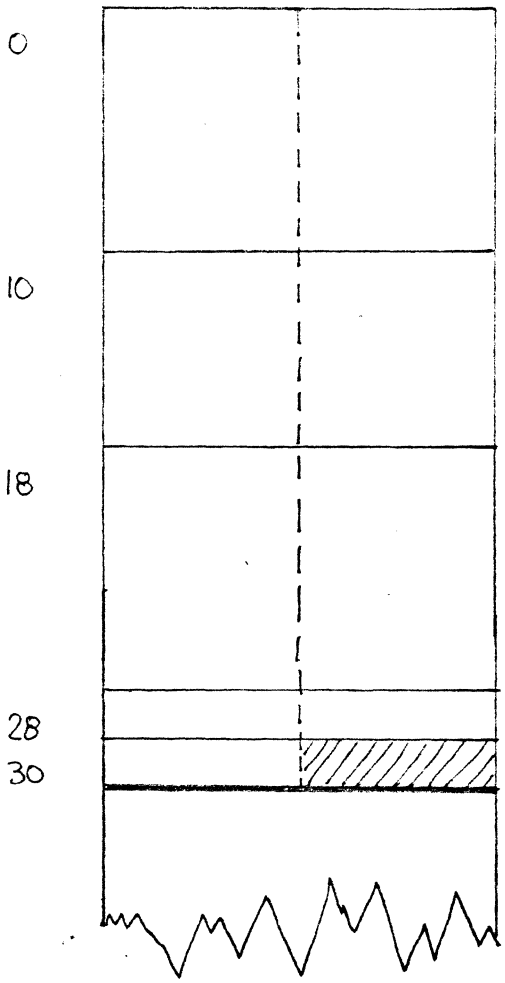
Address of DRIVE.VOLUME table

Address of DRIVE.USER table

Address of DRIVE.ACCESS table

Address of SYSTEM.BOOT table

NETWORK.USER table entry



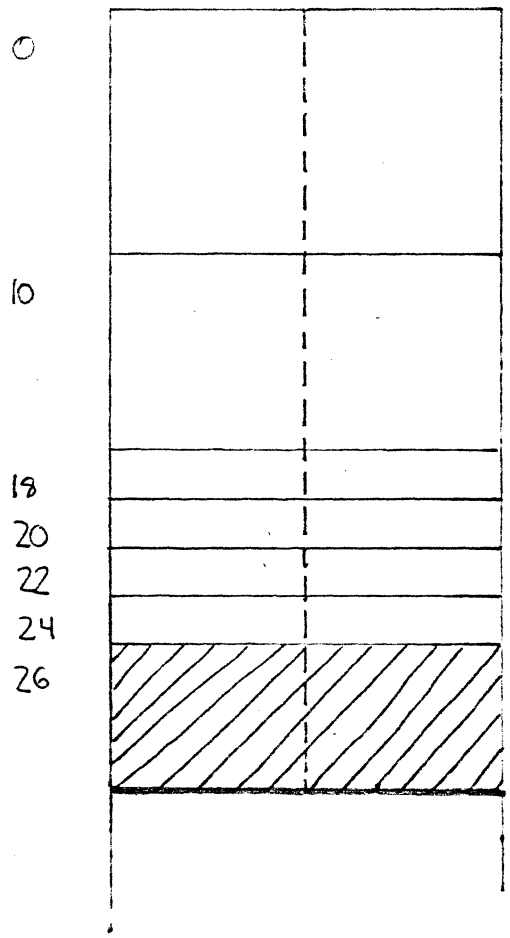

User name

User password

User home disk server name

Operating System type
Host station type

# DRIVE.USER table entry

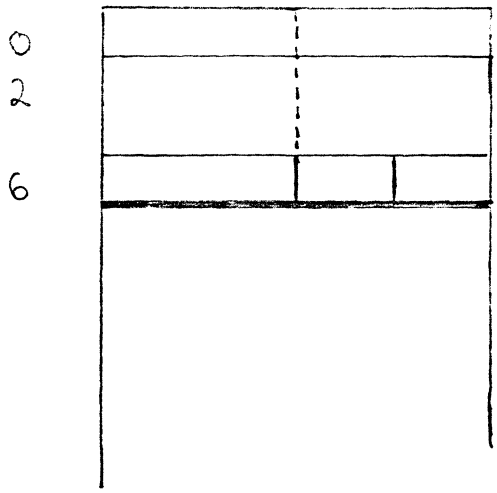



Drive user name

Drive user password

Drive user identifier
No. of volumes mounted on drive
No. of volumes accessible on drive
Boot volume indicator

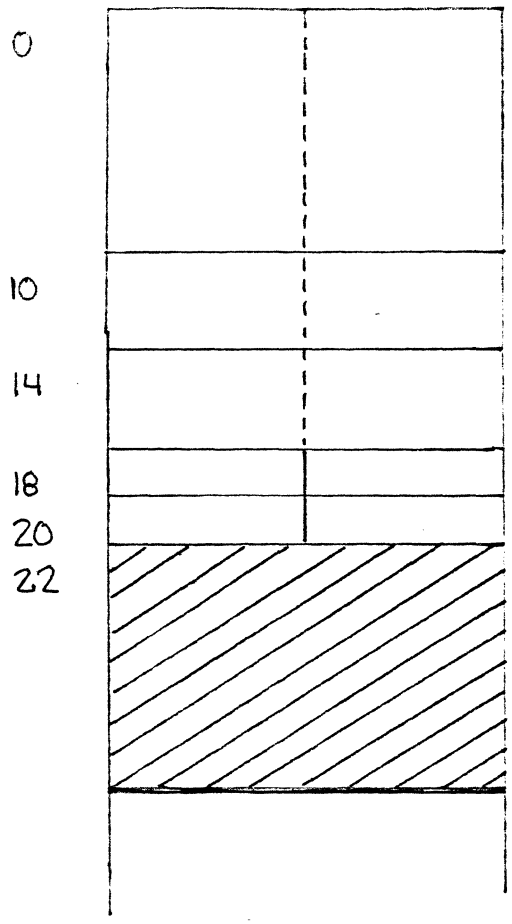
# DRIVE.ACCESS table entry




Drive user identifier
Volume starting address

Operating system dependent mount info/
READONLY indicator/
O.S. boot volume indicator

# DRIVE.VOLUME table entry



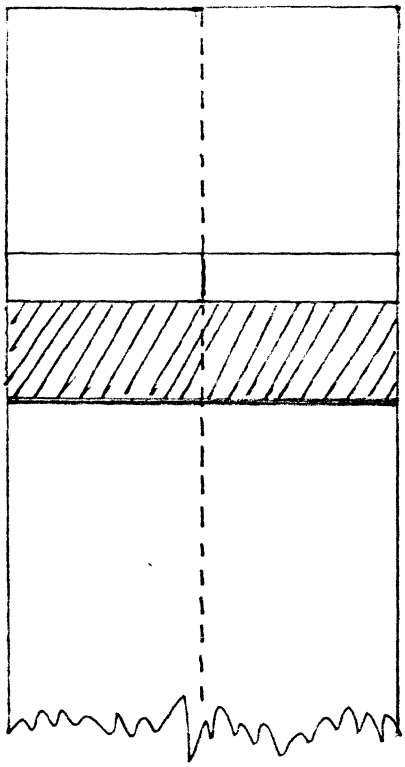

Volume name

Starting address of volume

Ending address of volume

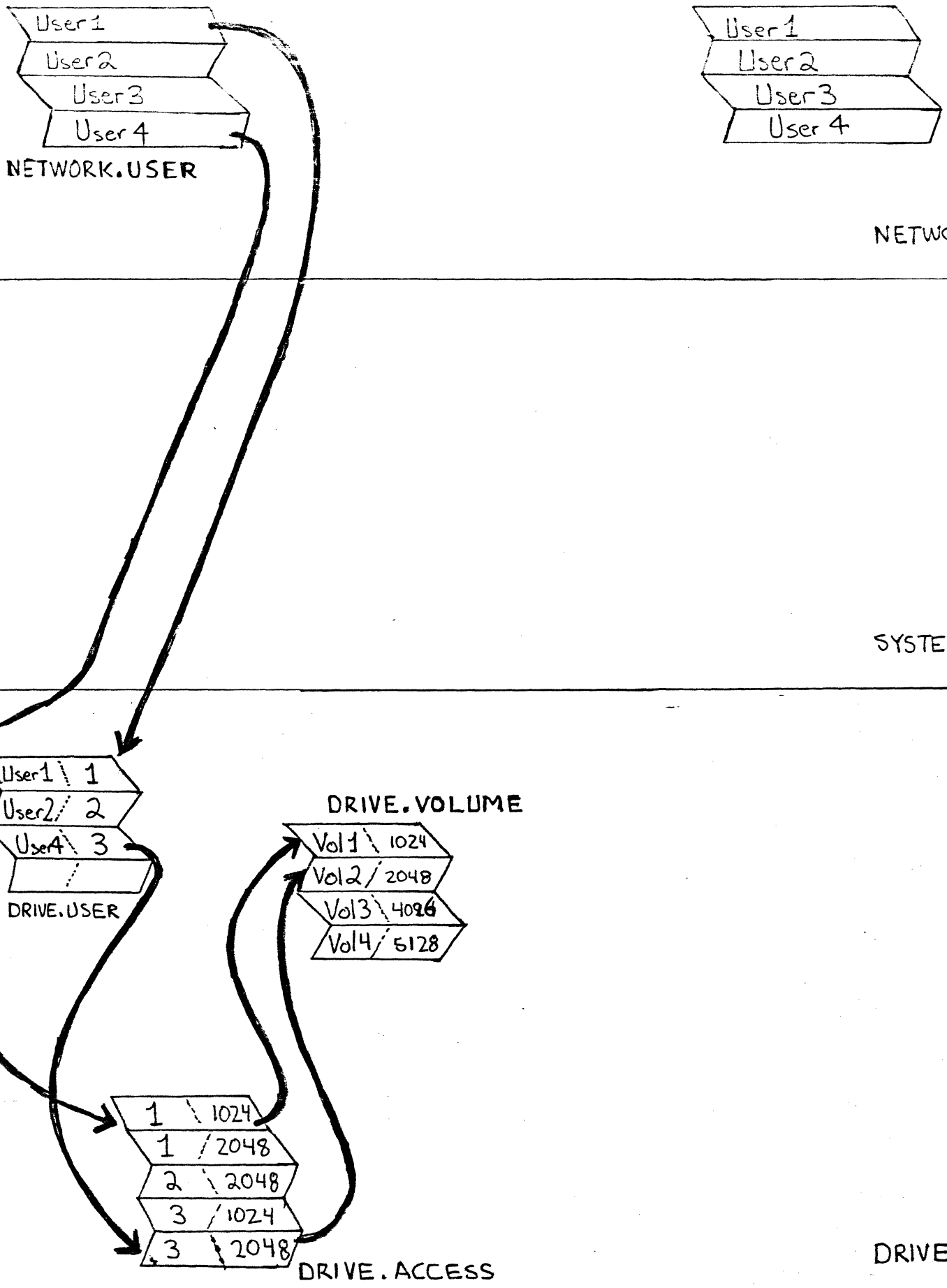
Volume type / READONLY indicator
Global access indicator / Volume offset

NETWORK.ACTIVE table entry



Host user/station name

Host Omninet station address/ Host type


User 1
User 2
User 3
User 4
NETWORK.USER
User 1
User 2
User 3
User 4
NETWORK
SYSTEM
User1 1
User2 2
User4 3
DRIVE.USER
DRIVE.VOLUME
Vol 1 1024
Vol 2 2048
Vol 3 4096
Vol 4 5128
1 1024
1 2048
2 2048
3 1024
3 2048
DRIVE.ACCESS
DRIVE

From: Denise Pitsch

February 15, 1982

To: Distribution
R&D Staff

Subject: Proposal for Constellation II software

This document is an update to the one dated January 22, 1982. Changes are marked in the margin.

(*$

permanent*)

# PROPOSAL FOR CONSTELLATION SOFTWARE

We at Corvus have long been planning to implement "Constellation software" on all of the systems that we support. By "Constellation software", we mean the set of programs which support multiple computers sharing a Corvus system. Currently, this set of programs has only been implemented on the Apple.

I feel that the current way of doing volume and user management is fine: its conceptually easy to understand and use. However, I would like to briefly describe two other schemes which were considered.

In the first scheme, all volumes are assigned two passwords when they are created: a read password and a write password. The user must supply the proper password whenever he/she runs the mount program. The advantages of this scheme are 1) the system manager does not have to individually grant volume access to a bunch of users when a volume is created. The disadvantages are 1) a user may have to remember several passwords; 2) every user sees every volume during the list function of the mount program.

In the second scheme, each volume is assigned an owner when it is created. Each owner is a user who belongs to a group. The owner of a volume can specify group and system access to the volume. For example, if I, user DENISE, own volume DENISE, and I belong to the RD group, I can specify that everyone in the RD group has read access to volume DENISE. The advantage of this scheme is 1) the system manager does not have to individually grant volume access to a bunch of users when a volume is created. The disadvantage is 1) users and volumes can belong to only one group, so a user will still have difficulty accessing a volume which is not in his/her particular group.

I have rejected both of these alternatives, and have instead chosen to stay with the known problems of our current scheme. To address the problem of gaining temporary access to another user's volume, a secured/released flag has been added at the user level. Each user may specify for each volume in his/her access table whether that volume is secured or released. Any other user may mount any volume which is released, as long as the user can specify a valid user name and the volume name. For instance, suppose PHIL has a volume in his access table called PHILB, and that PHILB is released. As user DENISE, I can specify that I want to mount PHILB(PHIL). This mount cannot be made permanent and read-only access is the only access allowed.

Before implementing Constellation II software, we should consider the shortcoming of the current Constellation software, as well as the implications of supporting multiple operating systems on one network.

Assumptions

1) No more single-user systems will be supported.

2) Any disks currently in use must be upgradeable to the new layout without loss of data. (This doesn't mean that the upgrade will be painless.)

Inadequacies of the current system:

1) Maximum 16MB drive. We should be able to support at least a 40MB drive, and preferably a 200MB (i.e., the Bank) drive. The virtual drive table now in use was implemented because current software could only handle 16MB drives. Its use is very confusing to our users, and it should be dropped. The current drive firmware supports a 20-bit address (256k 128-byte sectors, or a 128MB drive). The mirror commands can handle 64k blocks maximum (a 32MB drive), so the program to backup/restore an entire drive becomes more complicated.

2) Maximum 16MB per volume. This should be equal to the maximum drive size. Note that a given operating system may impose some lesser limit on volume size.

3) Maximum 63 volumes per drive. This number should be increased, and left variable, if possible. If it is not possible to make it variable, then 512 is probably a reasonable limitation. This would allow volumes of approximately 150 kbytes on an 80MB drive.

4) Four character user name, and 2 character password. These numbers should be upped to at least ten and seven, respectively. Also, all names should be encrypted when stored on the drive to prevent unscrupulous users from reading them easily.

5) Complicated installation procedure. The installation procedure should be made as turnkey as possible.

6) Maximum 128 users. Many of our educational installations complain that this number is much too small. Ideally, this number could be variable, but at least greater than 255.

7) User and volume tables cannot be backed up and restored. This can be fixed by making a Corvus volume, which can then be backed up and restored with the MIRROR program, or by a program which writes the various tables to a file.

Implications of supporting multiple processor types

1) A boot area is needed for each processor type. The firmware "boot command" needs to be modified to support this capability.

Implications of supporting multiple operating systems

1) Volume information must include a Corvus volume name (independent of any operating system volume name), and operating system dependent information.

2) The boot code for each processor must be able to decide which operating system to boot. For instance, the Apple II boot must know whether to boot DOS, Pascal, or CP/M.

OMNINET implications

1) Multiple disk servers on one network are possible. Therefore, the operating system drivers should include support for a disk server number in the mount table, and there must be a way to mount a volume located on a system other that the default.

2) There must be a convention for a processor to choose which system to get its boot code and user table from.

3) User names must be unique across the network, which is another reason to increase the limit on the numbe of user names.

4) It is possible to detect whether a user is still active on OMNINET. We could use this feature to try and do something about the problem of two people trying to access the same area at the same time.

Other wish lists

1) Unattended boot. If a user name is not entered within a certain amount of time, some default user will automatically be booted. On OMNINET, the default chosen could be based on transporter number.

Definition of terms used

network - an OMNINET network, with 1 or more disk servers

system - a set of daisy-chained drives, with a disk server or MUX

drive - one drive in a set of daisy-chained drives

volume - a contiguous area on a drive

mount - associate a volume with a driver in the host operating system

unmount - clear any association with a driver

The operating system types for users are:

| | | |
|---|---|---|
| 1 | Apple Pascal | APUCSD |
| 2 | Apple DOS 3.3 | APDOS33 |
| 3 | UCSD II.0 | UCSDII |
| 4 | IBM DOS | IBMDOS |
| 5 | Apple SOS | A3SOS |
| 6 | Apple run-time | APRT |
| 7 | CP/M | CPM |
| 8 | RT-11 | RT11 |
| 9 | RSX-11 | RSX11 |
| 10 | Pet | ? |
| 11 | NEWDOS | NEWDOS |
| 12 | NEWDOS-80 | NEWDOS80 |
| 13 | Atari DOS 2.0 | ATDOS20 |
| 14 | UNIX | UNIX |
| 15 | UCSD IV.0 | UCSDIV |

The directory types for volumes are:

| | | |
|---|---|---|
| 1 | UCSD | UCSD |
| 2 | CP/M | CPM |
| 3 | MDOS | MDOS |
| 4 | Pet | Pet |

| | | |
|---|---|---|
| 5 | Apple DOS 3.3 | A2DOS33 |
| 6 | Atari DOS 2.0 | ATDOS20 |
| 7 | RT-11 | RT11 |
| 8 | RSX-11 | RSX11 |
| 9 | NEWDOS | NEWDOS |
| 10 | NEWDOS-80 | NEWDOS-80 |
| 11 | UNIX | UNIX |
| 12 | Apple III SOS | A3SOS |

The processor names for boot files are:

| boot blocks | processor type | boot file name |
|---|---|---|
| 0, 1, 2, 3 | Apple II | BOOTx.APPLE2 |
| 4, 5, 6, 7 | LSI-11 | BOOTx.LSI11 |
| 8, 9, 10, 11 | IBM | BOOTx.IBMPC |
| 12, 13, 14, 15 | Xerox 820 | BOOTx.XR820 |
| 16, 17, 18, 19 | Zenith H89 | BOOTx.HZ89 |
| 20, 21, 22, 23 | NEC PC8000 | BOOTx.NC |
| 24, 25, 26, 27 | Pet | BOOTx.PET |
| 28, 29, 30, 31 | Atari | BOOTx.ATARI800 |
| 32, 33, 34, 35 | TRS-80 MOD I | BOOTx.TRSI |
| 36, 37, 38, 39 | TRS-80 MOD III | BOOTx.TRSIII |

x is L for flat cable boot and O for omninet boot.

(*$

permanent*)

# OVERVIEW OF CONSTELLATION II PROGRAMS

1. Installation program
   Initializes drive 1. Writes a default system password. Writes any boot information necessary. After running the installation program, the system should be usable by one user.

2. Drive Manager - system password required
   a. Initialize drive. Formats drive directory.
   b. List drives on line. List the drive number and capacity of all drives on line. Indicate if they are not initialized. Indicate controller ROM version and firmware version.
   c. List volumes. List the name, address, length, and operating system type of all volumes on the specified drive. Also list all the unused spaces on the drive.
   d. Add a volume to a drive.
   e. Remove a volume from a drive.
   f. List freespace. The unused space on all drives is displayed.
   g. Protect. Specify system-wide access: none/read-only/read-write.

3. User Manager -- requires system password (user information is updated on all systems, but transparently)
   a. List user information including user name, password, boot operating system type, and home system.
   b. Remove user from user directory.
   c. Add user to user directory.
   d. Change user information.

4. Access Manager - requires system password
   a. Specify user access to several volumes.
   b. Specify volume access for several users.

5. Boot manager -- requires system password.
   a. List filenames and cpu types of all boot files.
   b. Add boot file to volume.
   c. Remove boot file from volume.

6. Mount Manager
   a. List drives on line. Same as Drive Manager function.
   b. List volumes. Lists volumes accessible to this user.
   c. Mount volume. Mounts a volume.
   d. Unmount volume. Unmounts a volume.
   e. Save mount table. Saves current mount table as default.
   f. Change password. Changes user's password.
   g. Protect. Specify individual access: read-only/read-write
   h. Security. Specify secure/release.

7. Library procedures
   a. Mount/unmount a volume.
   b. Get user name.

8. Recovery programs
   a. Backup to a file the volume information, user information, boot information.
   b. Restore from a file the volume information, user information, boot information.
   c. Rewrite system name and password.

permanent*)
GENERAL USER INTERFACE CONSIDERATIONS

1. User will be able to press the <ESC> key at any time when he is prompted for input. This will cause an immediate exit to the current menu.

2. When user is prompted for a single character response, the user must press only those keys which correspond to correct input, return if there is a default, or <ESC>.

3. When user is prompted for a numeric input the first character may be a '+', '-', '!', '$', or a digit. The '!' or '$' signals that the input will be in hex (base 16). The backspace key may be used. The input must be terminated with a return, return with no input for the default, or <ESC>. Any characters typed by the user that are not valid for the current character position will be ignored. Result will be converted to an integer and truncated to +/- maxint if neccessary.

4. When the user is prompted for a string input, the backspace key may be used. Any characters in the string not in the required set will be ignored. Input must be terminated by return, return with no input for default, or <ESC>.

Boot/link sequence

In order to gain access to the network, the user must either boot directly from the Corvus, or run a link program. In either case, the user must specify a user name and password. The user name determines where the user's boot system is and which operating system will be booted/linked to.

Example

Enter your user name: <enter response>

Enter your password: <enter response-NOT echoed>

ERROR***
User <name> was not found. Reboot.
Incorrect password. Reboot.

Home system is <name>.
Operating system type is <type>.
(*$

```
 permanent*)

        PROGRAM DRVMGR  (* DRIVE MANAGER *)

Enter system name: <enter response>

ERROR***
  System <name> not found.  Reenter.

Enter system password: <enter response>

ERROR***
  Incorrect password.  Reenter.
  Program currently in use.  Program aborted.

DRIVE MANAGER [Vx.x] : I(nit drive     D(rives online     L)ist volumes
                       A(dd volume     R(emove volume     C)hange volume
                       P(rotect        F(ree space        Q(uit


DRIVE MANAGER - INIT DRIVE FUNCTION

Initialize drive volume table

  Eligible drives are 1-n.
  Enter the drive number of the drive to be initialized. <enter response>

  WARNING***
    Drive <number> has already been initialized, all volumes on
    the drive will be lost.  Do you still want drive <number>
    initialized? (y/n) <enter response>

  Initializing volume directory of drive <number>.


DRIVE MANAGER - DRIVES ON LINE FUNCTION

Drives online

  Drive Capacity Type    ROM   Firmware
    1    21220    10MB/B  22    V17.3 Corvus Systems   11/20/81
    2    38400    20MB/B  22    V17.3 Corvus Systems   11/20/81
   *3    11220     5MB/B  61    V17.3 Corvus Systems   11/20/81

   *Uninitialized drive


DRIVE MANAGER - ADD VOLUME FUNCTION

Add volume to drive

  Default volume size is <default> blocks.
  Enter the number of blocks for the new volume. <enter response>

  ERROR***
    The largest available space is nnnnn blocks.  Reenter.
```

```
  The new volume will be allocated on drive <number> at block
  address nnnnn.  Do you wish to change the volume
  location? (y/n) <enter response>

  Default drive is <default number>
  Enter drive number for new volume. <enter response>

  Enter the block address for the new volume. <enter response>

  ERROR***
    Space is not available at block address nnnnn.

  Enter the name for the new volume. <enter response>

  ERROR***
    Volume <name> already exists.
    <name> is an illegal volume name.
    <name> is a reserved volume name.

  Enter directory type: <enter response>

  ERROR***
    Unknown directory type.  Reenter.


DRIVE MANAGER - LIST DRIVE FUNCTION

List drive volume table

  The default drive is <default number>.
  Enter the number of the drive to be listed. <enter response>

  The default listing device is <default device>.
  Enter the device to send the listing to. <enter response>

  ERROR***
    Lising device <name> cannot be opened.  Reenter.

Volumes and unused space on drive <number> of system <system name>.
<date if available>

  Prot  Volume       Block address  Length  Dir type
  RO    APPLE3           nnnnn       nnnnn  UCSD
        <UNUSED>         nnnnn       nnnnn

  Prot: RO read-only, RW read-write, NA not accessible

DRIVE MANAGER - REMOVE VOLUME FUNCTION

Remove volume from drive

  Enter the name of the volume to be removed. <enter response>

  Enter the number of the drive the volume is on,
  press return if the drive is not known. <enter response>

  Searching for volume on drive <number>.

  ERROR***
```

```
    Volume <name> was not found.

  Volume <name> on drive <number> at block address nnnnn will
  be removed.  Are you sure? (y/n)

  Removing volume <name> and updating access tables ...


DRIVE MANAGER - CHANGE VOLUME

Change volume name.

  Enter name of volume to be changed: <enter response>

  Enter the number of the drive the volume is on,
  press return if the drive is not known. <enter response>

  Searching for volume on drive <number>.

  ERROR***
    Volume <name> was not found.  Reenter.

  Volume <name> found on drive <number>.
  Enter new volume name: <enter response>

  ERROR***
    Volume <name> already exists.  Reenter.


DRIVE MANAGER - FREE SPACE

Free space online

    Drive       Address  Length
      x         nnnnn    nnnnn
      x         nnnnn    nnnnn
      x         nnnnn    nnnnn


DRIVE MANAGER - PROTECT

  Enter the name of the volume to be protected: <enter response>

  ERROR***
    Volume <name> was not found.  Reenter.

  Volume <name> is currently (not accessible/read-only/read-write).
  Enter protection code: <enter response>


(*$
```

```
permanent*)

-
-     PROGRAM USERMGR  (* USER MANAGER *)
-

Enter system name:  <enter response>

ERROR***
  System <name> not found.  Reenter.

Enter system password:  <enter response>

ERROR***
  Incorrect password.  Reenter.
  Program currently in use.  Program aborted.


USER MANAGER [Vx.x]:  L(ist users     A(dd user        R(emove user
                      C(hange user    Q(uit


USER MANAGER - LIST USERS

List user information

  The default listing device is <default device>.
  Enter the device to send the listing to. <enter response>

  ERROR***
    Listing device <name> cannot be accessed.  Reenter.

  User information for system <system name>.   <date if available>

        User name    Password    Boot type   Boot system
    1   DENISEU      XYZ         UCSD        R&D
    2   BRUCE        X           UCSDIV      R&D
    3   KEITH                    CPM         R&D
        ...


USER MANAGER - ADD USER

Add a user to the system.

  Enter user name: <enter response>

  ERROR***
    User <name> already exists.  Reenter.
    Invalid user name.  Legal characters are A..Z,0..9.  Reenter.

  Enter password: <enter response-NOT echoed>

  Enter boot system name: <enter response>

  WARNING***
    System <name> is not currently active.  Do you wish to
    respecify the system name? <enter response>
    If y, then reenter.
```

```
  User <name> added.

USER MANAGER - REMOVE USER

Remove a user from the system.

  Enter user name to remove: <enter response>

  ERROR***
    User <name> was not found.  Reenter.

  User <name> removed.


USER MANAGER - CHANGE USER

Change the user information.

  Enter user name to be changed: <enter response>

  ERROR***
    User <name> was not found.  Reenter.

  Enter new user name: <enter response>

  ERROR***
    User <name> already exists.  Reenter.

  Enter new password: <enter response>

  The current boot system is <system name>.
  Enter boot system name: <enter response>

  WARNING***
    System <name> is not currently active.  Do you wish to
    respecify the system name? <enter response>
    If y, then reenter.

  Information for user <name> has been updated.


(*$
```

```
 permanent*)

      PROGRAM ACCMGR    (* ACCESS MANAGER *)

Enter system name:  <enter response>

ERROR***
  System <name> not found.  Reenter.

Enter system password: <enter response>

ERROR***
  Incorrect password.  Reenter.
  Program currently in use.  Program aborted.

ACCESS MANAGER [Vx.x]:  U(ser access     V(olume access


ACCESS MANAGER - USER ACCESS

Grant user access to several volumes.

  Enter user name:  <enter response>

  ERROR***
    User <name> was not found.  Reenter.

  Supervising user <name>.
  L(ist volumes  A(dd volume    R(emove volume    P(rotect
  M(ount         U(nmount       D(uplicate        Q(uit


ACCESS MANAGER - VOLUME ACCESS

Grant volume access to several users.

  Enter volume name: <enter response>

  ERROR***
    Volume <name> was not found.  Reenter.

  Supervising volume <name>.
  L(ist user     A(dd user       R(emove user   D(uplicate
  Q(uit


(*$
```

```
permanent*)

    PROGRAM BOOTMGR  (* BOOT MANAGER *)

BOOT MANAGER [Vx.x]:  L(ist boot files     A(dd boot file
                      R(emove boot file    U(pdate boot block
                      Q(uit

BOOT VOLUME MANAGER - LIST BOOT FILES

List the current boot files.

  The default listing device is <name>.
  Enter the device to send the listing to. <enter response>

  ERROR***
    Listing device <name> cannot be accessed.  Reenter.

  Boot file name   Address    Length
  BOOTL.APPLE      30         10
  BOOTO.SUPERB     40         5
  BOOTL.H89        45         5
  ...


BOOT VOLUME MANAGER - ADD BOOT FILE

Add a new file to the boot set.

  Enter processor type: <enter response>
  Enter type of boot: <enter response>

  ERROR***
    Unknown processor type <response>.  Reenter.
    Unknown boot type <response>.  Reenter.
  WARNING***
    Boot file for <processor/type> already exists.
    Do you wish to replace it? (y/n) <enter response>

  Enter source file name: <enter response>

  ERROR***
    File <name> not found.  Reenter.
    No room for boot file.

  Boot file <name> created.


BOOT MANAGER - REMOVE BOOT FILE

Remove a boot file.

  Enter name of file to be removed: <enter response>

  ERROR***
    File <name> not found.  Reenter.
```

File <name> removed.

BOOT MANAGER - UPDATE BOOT BLOCK

Updates the boot block.

Do you wish to recreate the boot block? <enter response>

Boot block rewritten.

(*$

permanent*)

```
    PROGRAM MOUNT  (* MOUNT/UNMOUNT MANAGER *)

MOUNT MANAGER [Vx.x] : D(rives online   L(ist volumes   M(ount volume
                       U(nmount volume  P(rotect        S(ecure
                       R(emove volume   Q(uit

MOUNT MANAGER - DRIVES ON LINE FUNCTION

(same as DRIVE MANAGER)


MOUNT MANAGER - LIST VOLUMES FUNCTION

List user volume table

  The default listing device is <default device>.
  Enter the device to send the listing to. <enter response>

  ERROR***
    Listing device <name> cannot be accessed.  Reenter.

Volume table for user <user name> on system <system name>.
<date if available>

  Prot Sec Mount Volume      Block address  Length   Drive
  RO       #4    APPLE3          nnnnn       nnnnn     1
       S         PRIVATE         nnnnn       nnnnn     4

  Prot: RO read-only, NA non accessible, blank is read-write
  Sec: S secured, blank is released


MOUNT MANAGER - MOUNT VOLUME FUNCTION

Mount volume

  Enter the name of the volume to be mounted: <enter response>

  ERROR***
    Volume <name> was not found.  Reenter.
    Volume <name> is currently not accessible.  Reenter.

  Enter unit to mount volume on. <cpu specific entry>

  ERROR***
    Invalid unit. Re-enter.

  <volume name> has been mounted on unit <unit mounted on> with
  <type of access> access.

To mount a volume from another system in the network, specify volume
name as <volume name@system name>.  To mount a released volume from
another user, specify volume name as <volume name (user name)>.  System
name may also be specified if necessary.
```

Volumes from another system, and volumes not in your access table, may not be permanently mounted.

MOUNT MANAGER - UNMOUNT FUNCTION

Unmount volume

Enter the unit to be unmounted. <enter response>

ERROR***
Invalid unit number. Reenter.

Unit <unit number> unmounted.

MOUNT MANAGER - PROTECT

Write protect/unprotect at the user level.

Enter the name of the volume to be protected: <enter response>

ERROR***
Volume <name> was not found. Reenter.
You have read-only access to volume <name>.
Volume <name> is currently not accessible.

Volume <name> is currently (read-only/read-write).
Enter protection code: <enter response>

MOUNT MANAGER - SECURE

Sets secured/released flag.

Enter the name of the volume to be secured: <enter response>

ERROR***
Volume <name> was not found. Reenter.

Volume <name> is currently (secured/released).
Enter security code: <enter response>

MOUNT MANAGER - QUIT

Do you wish to make these changes permanent? (y/n) <enter response>

Mount table saved.

(*$

permanent*)

## LIMITATIONS

There are certain functions which are not provided by the scheme described here. These are:

1) Users cannot jump from OS to OS (as they can now from Apple Pascal to Apple DOS). A user who wants to access two OS's must have 2 user id's.

# CORVUS CONSTELLATION II Table

## Reference Document

15-Feb-82

Corvus System Incorporated
2029 O'Toole Avenue
San Jose, California
95129

{$

This document describes the proposed formats of the Corvus Constellaton II tables:

The proposed tables included are:

(1) Drive Information Table
(2) Active User Table
(3) User Name Table
(4) User Access Table
(5) User Logon Table
(6) Cold Boot Table
(7) Volume Table
(8) Corvus File System Tables

Table name: Drive Information Table

Location: Block 1

The Drive Information table is located in block 1 on each drive.

File name: N/A

Entry size: 512 bytes (1 block)

Table size: 1 blocks

Description: The DRIVE.INFO table occupies a fixed location on each disk on a Corvus network. The primary purpose of the table is to provide global access to vital system information. The address of the Corvus volume is located in the table. Keeping the volume address in a common location allows the volume to be moved on the disk when volume expansion is necessary. The table also holds the drive name and password, and the disk server name and password. The drive initialized flag, disk size, and drive number are also contained in this table.

Access: All Corvus utilities accessing the Corvus volume make use of this table since the addres of the Corvus volume is located here. The DRIVE MANAGER utility modifies the address of the Corvus volume whenever the volume is relocated or expanded. The drive initialization utility updates the drive and server identification field, sets the drive init field, drive size, and drive number.

{$

```
Layout:         type
                   DRIVENAME    =  packed array[1..10] of char;
                   DRIVEPSW     =  packed array[1..8] of BYTE;
                   SERVERNAME   =  packed array[1..10] of char;
                   SERVERPSW    =  packed array[1..8] of BYTE;
                   ABSDISKADDR  =  packed array[ABSORDER] of byte;
                   DRIVESIZE    =  ABSDISKADDR;
                   ABSORDER     =  (HI, MID, LO );
                   ABSDISKADDR  =  packed array[ABSORDER] of byte;

                   DRIVEINFO    =  record
                                   DRIVE:          DRIVENAME;
                                   DRIVEPASS:      DRIVEPSW;
                                   SERVER:         SERVERNAME;
                                   SERVERPSW:      SERVERPSW;
                                   CRVSADDR:       ABSDISKADDR;
                                   SIZE:           DRIVESIZE;
                                   DRIVEINIT:      BOOLEAN;
                                   DRIVENO:        INTEGER;
                                   ONLINE:         Boolean;
                                   BOOTADDR;       ABSDISKADDR;
                                   ACTIVEADDR:     ABSDISKADDR;
                                   RESERVED:       packed array[460] of byte;
                                   end;
{$
```

Table name: Active Station Table

Location: Blocks 2-5

Blocks 2 thru 5 are allocated for the active user table on each drive of the network although only the copy on the primary drive of the disk server is actually used.

File name: NA

Entry size: 16 bytes/entry
32 entries/block

Table size: 2 blocks

Organization: The Active station table is ordered by station name.

Description: The Active Station table (NETWORK.ACTIVE) records all stations active on the network. Each new user of the network is logged into the table during cold boot or via the special server logon protocol. An active user table entry consists of the station name, station type ( User station, disk server, printer server , etc. ), and the host number of the station. This table is maintained by the disk server firmware but can be read by Corvus utilities for hints as to the station address of various nodes on the network. This table is used by the cold boot code to find the station address of the home disk server.

Access: This table is initialized and modified exclusively by the disc server. The table is read by the cold boot code to find the station address of the home disk server.

Layout:

```
type
   STATIONNAME =  packed array[1..10] of char;

   USERENTRY   =  record
                  STATIONNAME:  STATIONNAME;
                  HOSTNO:       byte;
                  HOSTTYPE:     byte;
                  reserved:     array[1..4] of byte;
{$
```

Table name: User Name Table

Location: Corvus Volume

The DRIVE.USER table is located on each drive of the network in the corvus volume area. The table is unique on each drive.

File name: DRIVE.USER

Entry size: 32 bytes/entry
16 entries/block

Table size: 4 - 32 blocks (maximum of 512 entries requires 32 blocks)

Organization: The User Name table is organized in alphabetical order.

Description: The DRIVE.USER table is used to map user names to volumes accessible on a particular drive. Each user with access to a volume on the drive has an entry in the DRIVE.USER table. The USERID field of each entry corresponds to the USERID field in the USER ACCESS Table. Each entry of this table contains the user name, user password, and user id field. An optional field identifies the number of volumes the user has currently mounted on this drive and the total number of volumes to which the user has access.

All volume mount operations whether during cold boot or via the MOUNT MANAGER require the use of this table. An entry in the USER ACCESS table, DRIVE.ACCESS, associates a drive unique USERID to a volume on that drive. When resolving volume access for a particular user, the USER NAME table (DRIVE.NAME) is searched inorder to determine the user's USERID. This ID can then be used to locate that user's volumes on that drive.

Access: All table additions/deletions are made via the Corvus ACCESS MANAGER utility. The MOUNT MANAGER may update table entries for record keeping purposes.

{$

```
Layout:         type
                   USERNAME      =  packed array[1..10] of char;
                   USERPSW       =  packed array[1..8] of byte;


                   USNAMEENTRY   =  record
                                    NAME:         USERNAE;
                                    PASSWORD:     USERPSW;
                                    USERID:       integer;
                                    MOUNTED:      integer;
                                    ACCESS:       integer;
                                    RESERVED:     packed array[1..8] of char;
                                    end;


                   USNAMETABLE   =  packed array[1..512] of USNAMEENTRY;
```

{$

Table name: User Access Table

Location: Corvus Volume

The DRIVE.ACCESS table is a drive specific table and is located on each drive on the network. The table contents are unique on each drive of the network.

File name: DRIVE.ACCESS

Entry size: 8 bytes/entry
64 entries/block

Table size: 4 - 256 blocks ( 512 users with access to a maximum of 256 volumes on a single drive requires 256 blocks. )

Organization: Table is ordered in ascending order by USERID.

Description: The USER ACCESS table identifies each volume a user has access to on the drive. Each entry contains the USERID of a user identified in the DRIVE.NAME table, the physical disk address of the volume, and information concerning volume mount condition, read/write access, and boot information. The DRIVE.ACCESS table is ordered by USERIDs. The mount access information MNTINFO is operating system dependent.

Access: The DRIVE.ACCESS table is modified by the Corvus ACCESS MANAGER. The Corvus MOUNT MANAGER, boot process, and VOLUME MANAGER also access this table.

Layout:

```
type
   ABSORDER     =  ( HI, MID, LO );
   ABSDISKADDR =  packed array[ABSORDER] of byte;

   USACCENTRY   =  record
                   USERID:       integer;
                   VOLADDR:      ABSDISKADDR;
                   MNTINFO:      byte;
                   READONLY:     byte;
                   BOOTFLAG:     byte;
                   end;


   USACCTABLE   =  packed array[1..165361 of USACCENTRY;
```

{$

Table name: User Logon Table

Location: Corvus Volume.

The NETWORK.USER table is located in the Corvus volume area of the primary drive of the disk server. The primary drive is the first drive on the disk server. Each primary drive's Corvus volume contains an entry for each user the network. The user logon table is stored redundantly on each drive.

File name: NETWORK.USER

Entry size: 32 bytes/entry
16 entries/block

Table size: 4 - 32 blocks ( a maximum of 512 users requires 32 blocks)

Orgainzation: Table organized in alphabetical order.

Description: The User logon table identifies each user of the network. The table contents include the user name, logon password, the user's disk server name, and operating system type. Each valid user of the network has an entry in the user table which is kept in the Corvus volume located on the first drive of the disk server. The entry is replicated on each primary Corvus volume on each disk server.

Access: User entries can be modified by the USER MANAGER exclusively.

Layout:

```
type
   USERNAME     =  packed array[1..10] of char;
   USERPSW      =  packed array[1..8] of char;
   SERVERNAME   =  packed array[1..10] of char;

   USERENTRY    =  record
                   NAME:          USERNAME;
                   PASSWORD:      USERPSW;
                   SERVER:        SERVERNAME;
                   OPSYS:         byte;
                   RESERVED:      packed array[1..3] of byte;
                   end;


   USERTABLE    =  packed array[1..512] of USERENTRY;
```

{$

Table name: Cold Boot Table

Location: Corvus Volume

The SYSTEM.BOOT table is located on each drive of the network. The contents of each table may differ depending on the boot code files residing on the particular volume.

File name: SYSTEM.BOOT

Entry size: 8 bytes/ processor or system type
(allows maximum of 128 processor types)

Table size: 2 blocks (table size is constant )

Description: A Corvus Cold Boot Table entry contains the disk address of a block of the cold boot code file for each bootable system on the network. Any system utilizing the Constellation logon process is considered a bootable system. This includes systems with special boot ROMs and those requiring user intervention (i.e. CLINK program of CP/M).

The cold boot table file is the first file of the Corvus volume. The entrys are arranged in pairs of two. The first two-byte entry is the address of the first 512 byte block of the processor's OMNINET cold boot file. The second 2-byte entry id the disk address of the second 512 byte block of the OMNINET cold boot file. The second pair of entries contains the addresses of the first and second blocks of the MUX or flat cable cold boot. The assigned processor type number is the index into the table used by the cold boot firmware to locate the processor boot cold. The disk addresses used are relative to the start of the Corvus volume. A zero entry indicates an uninitialized entry.

Access: The Cold Boot Table is accessible via the controller boot command and processor's cold boot code. The table is manipulated by the Corvus BOOT MANAGER utility.

{$

```
Layout:         type
                   RELORDER     =   ( HI, LO );
                   RELDISKADDR =    packed array[RELORDER] of byte;


                   BOOTENTRY    =   record
                                    OMNIBLK1:      RELDISKADDR;
                                    OMNIBLK2:      RELDISKADDR;
                                    MUXBLK1:       RELDISKADDR;
                                    MUXBLK2:       RELDISKADDR;
                                    end;


                   BOOTTABLE    =   record
                                    case integer of
                                    1:(entry:
                                       packed array[1..64] of BOOTENTRY);
                                    2:(index:
                                       packed array[1..256] of RELDISKADDR);
                                    end;

{$
```

Table name: Corvus Volume Table

Location: Corvus Volume

Each drive on the network contains a unique volume table that identifies each volume configured on that drive.

File name: DRIVE.VOLUME

Entry size: 32 bytes
16 entries/block

Table size: 4 - 64 blocks ( maximum allows 512 volumes/drive)

Organization: Table is organized in ascending order by starting disk address of volume. This scheme allows free space to be calculated easily.

Description: The Corvus Volume Table contains a single entry for each volume residing on the drive. Each entry identifies the volume by name, the absolute starting and ending disk block addresses, the volume type (i.e. UCSD, ATARI, etc ), and 2 volume access flags. The "GLOBAL ACCESS FLAG" is used to restrict access to the volume for all users. All operating system dependent information is considered part of the volume, the optional volume offset field points to the start of the actual volume contents. The volume table contains an entry for the Corvus Volume also.

Access: All modifications to the volume table are performed by the Corvus volume manager utility. The Corvus Mount manager and cold boot also access this table.

{$

```
Layout:         type
                   VOLUMENAME   =  packed array[1..10] of char;
                   ABSORDER     =  (HI, MID, LO );
                   ABSDISKADDR  =  packed array[ABSORDER] of byte;


                   VOLUMEENTRY  =  record
                                   NAME:            VOLUMENAME;
                                   STARTBLK:        ABSDISKADDR;
                                   ENDBLK:          ABSDISKADDR;
                                   VOLTYPE:         byte;
                                   WRITEENABLE:     byte;
                                   GLBACCESS:       byte;
                                   VOLOFFSET:       byte;
                                   RESERVED:        packed array[12] of char;
                                   end;             { VOLUMETABLE }


                   VOLUMETABLE  =  packed array[1..512] of VOLUMEENTRY;

{$
```

Table name: Corvus Volume File Entry

Location: Corvus Volume Directory

A Corvus volume file entry can occupy any of 77 file entry locations in blocks 0 thru 5 of the Corvus volume directory.

File name: N/A

Entry size: 26 bytes

Table size: N/A

Descripton: A Corvus volume file entry describes the characteristics of a file residing in the area of the Corvus volume. The first entry of a Corvus volume directory is the volume entry. All remaining entries describe the volume's files. Entry information includes file name, file size, starting and ending block numbers, and other file information. A Corvus directory volume entry is identical to a Merlin file system volume entry.

Access: All manipulation of files in the Corvus volume is through the file system defined by the Corvus volume constructs. Each Corvus utility is responsible for maintaining the integrity of the volume and the volume directory entries. Routines designed to access files in the Corvus directory must adhere to restrictions imposed by the directory structure. Since all Corvus tables are also Corvus volume files, the directory mechanism should be used when accessing any Corvus tables.

Layout:

```
type
   RELORDER     =  (HI, LO);
   RELDISKADDR =  packed array[ RELORDER ] of byte;

   fileentry    =  record
                   STARTBLOCK:  RELDISKADDR;
                   ENDBLOCK:    RELDISKADDR;
                   FILETYPE:    byte;
                   RESERVED1:   byte;
                   CRVSFLELEN:  byte;
                   CRVSFLENAME: packed array[1..15] of char;
                   LASTBYTES:   integer;
                   LASTDATE:    integer;
                   end;
```

{$

Table name: Corvus Volume Directory Entry

Location: Corvus Volume Directory

A Corvus volume directory entry occupies the first 26 bytes of the Corvus volume. The first 6 blocks of the volume make up the Corvus volume directory. The first entry in the Corvus volume directory is an entry for the Corvus volume.

File name: N/A

Entry size: 26 bytes

Table size: N/A

Description: The Corvus volume directory entry describes the characteristics of the corvus volume and the characteristics of all files collectively in the Corvus volume. The Corvus volume directory and all entries mimick exactly the UCSD and Merlin fle structures. Included in the volume directory entry are the volume name, starting and ending relative block addresses, total count of all files in the volume, and other maintenance data. There is but one volume entry in the Corvus volume directory.

Access: All manipulation of files in the Corvus volume is through the file system defined by the Corvus volume constructs. Each Corvus utility is responsible for maintaining the integrity of the volume and the volume directory entries. Routines designed to access files in the Corvus directory must adhere to restrictions imposed by the directory structure. Since all Corvus tables are also Corvus volume files, the directory mechanism should be used when accessing any Corvus tables.

Layout:

```
type
   RELORDER     =  (HI, LO);
   RELDISKADDR =  packed array[ RELORDER ] of byte;

   volumeentry =  record
                  STARTBLOCK:   RELDISKADDR;
                  ENDBLOCK:     RELDISKADDR;
                  ENTRYTYPE:    byte;
                  RESERVED1:    byte;
                  CRVSVOLLEN:   byte;
                  CRVSVOLNAME:  packed array[1..7] of char;
                  LASTBLOCK:    integer;
                  FILECOUNT:    integer;
                  RESERVED2:    integer;
                  LASTDATE:     integer;
                  RESERVED3:    integer;
                  RESERVED4:    integer;
                  end;
```

Table name: Corvus Volume

Location: The Corvus volume is located on each drive of the network.

File name: N/A

Entry size: N/A

Volume size: 64 - 1024 blocks.

Description: The Corvus volume contain the following tables:

(1) User Name table
(2) User Access table
(3) User Logon table
(4) Volume table
(5) Cold Boot table

Access: For CORVUS Concept utility implementation, the Merlin PASCAL file system interface is sufficient for all file operation. Maintaining directory and file integrity is of utmost importance.

Layout:

```
type
   BLOCKSIZE     =  512;
   BLOCK         =  packed array[1..512] of byte;


   CRVSDIRECTORY=   record
                    VOLINFO     volumeentry;
                    FILEINFO:   array[MAXFILE] of fileentry;
                    end;

   CORVUSVOLUME =   record
                    DIRECTORY:   Crvsdirectory;
                    FILESPACE:   packed array[1018] of BLOCK;
                    end;
```