IBM

ACADEMIC OPERATING SYSTEM 4.3

ACADEMIC OPERATING SYSTEM 4.3

VOLUME II

IBM

ACADEMIC OPERATING SYSTEM 4.3

ACADEMIC OPERATING SYSTEM 4.3

VOLUME II

IBM Academic Operating System 4.3

PRPQ #5799-WZQ
PRPQ #5799-PFF

Volume II

Academic Information Systems
International Business Machines Corporation
Palo Alto, CA

NOTICE

WARNING: Any shipment of Academic Operating System 4.3 (hereinafter referred to as "IBM/4.3") for the IBM RT PC or IBM 6152 Academic System to a country outside the United States requires a U.S. Government license.

First Edition (December 1987)

Changes will be made periodically to the information herein; these changes will be incorporated in new editions of this publication.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available outside the United States.

International Business Machines Corporation provides this software and documentation "as is", without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. No assurance of successful installation can be given. IBM may make improvements and/or changes in the product(s) and/or program(s) described in this manual at any time.

A reader's comment form has been provided at the back of this publication. If the form has been removed, address comments to Academic Information Systems; University Support, Dept. 6FR; IBM Corporation; P.O. Box 10500; Palo Alto, CA 94303.

IBM may use or distribute any of the information you supply in any way it deems appropriate without incurring any obligation to you.

TRADEMARKS

The following trademarks appear in this manual:

UNIX is a registered trademark of AT&T Bell Laboratories
DEC and VAX are trademarks of Digital Equipment Corporation
AIX, RT, and RT PC are trademarks of International Business Machines Corporation
MetaWare, High C, and Professional Pascal are trademarks of MetaWare Incorporated
Ethernet is a trademark of Xerox Corporation
Documenter's Workbench is a registered trademark of AT&T Technologies

December 1987

This page intentionally left blank.

Preface

This software and documentation is based in part on the 4.3 Berkeley Software Distribution under license from The Regents of the University of California. We gratefully acknowledge the following individuals and institutions for their role in its development: Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, Berkeley, California; Individual Computing Systems, IBM Research, Yorktown Heights, New York; The IRIS Group, Brown University, Providence, Rhode Island; ITC, Carnegie-Mellon University, Pittsburgh, Pennsylvania.

IBM
Academic Information Systems
Palo Alto, CA
July 1987

Preface to 4.3 Berkeley Software Distribution

This update to the 4.2 distribution of August 1983 provides substantially improved performance, reliability, and security, the addition of Xerox Network System (NS) to the set of networking domains, and partial support for the VAX 8600 and MICROVAXII.

We were greatly assisted by the DEC UNIX Engineering group who provided two full time employees, Miriam Amos and Kevin Dunlap, to work at Berkeley. They were responsible for developing and debugging the distributed domain based name server and integrating it into the mail system. Mt Xinu provided the bug list distribution service as well as donating their MICROVAXII port to 4.3BSD. Drivers for the MICROVAXII were done by Rick Macklem at the University of Guelph. Sam Leffler provided valuable assistance and advice with many projects. Keith Sklower coordinated with William Nesheim and J. Q. Johnson at Cornell, and Chris Torek and James O'Toole at the University of Maryland to do the Xerox Network Systems implementation. Robert Elz at the University of Melbourne contributed greatly to the performance work in the kernel. Donn Seeley and Jay Lepreau at the University of Utah relentlessly dealt with a miriad of details; Donn completed the unfinished performance work on Fortran 77 and fixed numerous C compiler bugs. Ralph Campbell handled innumerable questions and problem reports and had time left to write rdist. production systems long before we were confident enough to inflict it on our users. Bill Shannon at Sun Microsystems has been helpful in providing us with bug fixes and improvements. Tom Ferrin, in his capacity as Board Member of Usenix Association, handled the logistics of large-scale reproduction of the 4.2BSD and 4.3BSD manuals. Mark Seiden helped with the typesetting and indexing of the 4.3BSD manuals. Special mention goes to Bob Henry for keeping ucbvax running in spite of new and improved software and an ever increasing mail, news, and uucp load.

Numerous others contributed their time and energy in creating the user contributed software for the release. As always, we are grateful to the UNIX user community for encouragement and support.

Once again, the financial support of the Defense Advanced Research Projects Agency is gratefully acknowledged.

M. K. McKusick
M. J. Karels
J. M. Bloom

Preface to 4.2 Berkeley Software Distribution

This update to the 4.1 distribution of June 1981 provides support for the VAX 11/730, full networking and interprocess communication support, an entirely new file system, and many other new features. It is certainly the most ambitious release of software ever prepared here and represents many man-years of work. Bill Shannon (both at DEC and at Sun Microsystems) and Robert Elz of the University of Melbourne contributed greatly to this distribution through new device drivers and painful debugging episodes. Rob Gurwitz of BBN wrote the initial version of the code upon which the current networking support is based. Eric Allman of Britton-Lee donated countless hours to the mail system. Bill Croft (both at SRI and Sun Microsystems) aided in the debugging and development of the networking facilities. Dennis Ritchie of Bell Laboratories also contributed greatly to this distribution, providing valuable advise and guidance. Helge Skriverik worked on the device drivers which enabled the distribution to be delivered with a TU58 console cassette and RX01 console floppy disk, and rewrote major portions of the standalone i/o system to support formatting of non-DEC peripherals.

Numerous others contributed their time and energy in organizing the user software for release, while many groups of people on campus suffered patiently through the low spots of development. As always, we are grateful to the UNIX user community for encouragement and support.

Once again, the financial support of the Defense Advanced Research Projects Agency is gratefully acknowledged.

S. J. Leffler
W. N. Joy
M. K. McKusick

CONTENTS

VOLUME II. SUPPLEMENTARY DOCUMENTS

APPENDICES

This page intentionally left blank.

VOLUME II. SUPPLEMENTARY DOCUMENTS

This volume contains information about configuring and operating IBM/4.3, as well as information for the programmer. It contains new and revised material for the 4.3BSD *UNIX Programmer's Supplementary Documents (PS1)* and the *UNIX System Manager's Manual (SMM).*

- Updated articles from the 4.3BSD SMM:
  - **Installing and Operating Academic Operating System 4.3** describes how to install and operate the operating system.
  - **Building IBM/4.3 Systems with Config** describes *config*, a tool used in building IBM/4.3 system images, and provides information for using *config* on the IBM RT PC and IBM 6152 Academic System.
- **IBM RT PC New Model Series Upgrade Instructions** describes the procedure used to install the upgrade kit used to convert an IBM RT PC Model 015 or 025 to a 115 or 125.
- **The IBM 3812 Pageprinter** provides information for installing the IBM 3812 Pageprinter, and installing and converting fonts.
- **IBM/4.3 Console Emulators** explains the need for, and design of, emulators for IBM/4.3.
- **The Remote Virtual Disk System** describes a network service that provides a client computer with the appearance of removable-media disk drives and an unlimited number of disk packs.
- **The DMA Reference Manual** describes the utilities provided with the operating system for using the Direct Memory Access (DMA) channels.
- **Assembler Reference Manual for IBM/4.3** describes the usage and input syntax of *as*, the IBM/4.3 assembler for the IBM RT PC and IBM 6152 Academic System. This article replaces the *Assembler Reference Manual* found in the 4.3BSD PS1.
- **Floating Point Arithmetic** summarizes floating point arithmetic in IBM/4.3.
- **The C Subroutine Interface for the IBM Academic Information Systems Experimental Display** describes an interface with graphics routines for the IBM Academic Information Systems experimental display.[1]
- **Programmer's Notes** is a brief compendium of insights, suggestions, and notes gathered from the programmers who ported applications to IBM/4.3.
- **IBM/4.3 Linkage Convention** describes the calling sequence used in IBM/4.3.
- **Recompiling with High C** provides guidance to C programmers who recompile existing programs with High C.
- **Professional Pascal Differences** points out the major differences between Berkeley Pascal and Professional Pascal as an aid to programmers who recompile existing programs with Professional Pascal.
- **PIC -- A Graphics Language for Typsetting User Manual** describes how to use PIC, a language used to draw simple figures on a typesetter.

[1] Hereinafter referred to as "the experimental display."

- **The X Window System** describes how to use the X Windowing System as well as how to use the bitmap editor and how to change the system defaults.
- **Appendix A. Software Description** contains a list of 4.3BSD and IBM/4.3 functions supported in this distribution, as well as a list of unsupported functions.
- **Appendix B. Graphics Manual Pages** contains manual pages for the graphics routines provided with the C Subroutine Interface for the experimental display.
- **Appendix C. High C Programmer's Guide** contains a guide to programming with High C on the IBM RT PC and IBM 6152 Academic System. Note that revised and new manual pages for Volume 1 of the *UNIX Programmer's Manual* are found in Volume I of this manual.

Installing and Operating
Academic Operating System 4.3

ABSTRACT

This article is a revision of an article entitled "Installing and Operating 4.3BSD on the VAX," written in April 1986 by Michael J. Karels *et al.*, and found in the *UNIX System Manager's Manual.* The revisions include additions and changes appropriate to the IBM RT PC and IBM 6152 Academic System.

The article contains the following chapters:

1. **Introduction** provides information helpful in installing and operating IBM/4.3. It covers hardware supported, distribution format, IBM/4.3 device naming and IBM/4.3 block and raw devices.
2. **Saving and Restoring Local Modifications** describes how to upgrade a currently running release to an entirely new release.
3. **Installation Procedures** describes how to install the IBM/4.3 system.
4. **System Setup** describes the procedures used to set up an IBM/4.3 system.
5. **Network Setup** describes how to configure your system to use the networking support.
6. **System Operation** describes some typical IBM/4.3 operations on an IBM RT PC and IBM 6152 Academic System.

Appendix A. AIX and IBM/4.3 Co-residence describes installing and using AIX on a IBM/4.3 system.

Appendix B. MINIROOT Kernel Configured Devices describes the configurable devices supported by the MINIROOT kernel.

Appendix C. Building a Master/Server Machine describes how very small installations with a single IBM RT PC can combine functions usually shared by several RTs.

Note that throughout this article, prompts that appear on the screen are shown in *italics*, while values you are to type appear in **boldface**.

1. INTRODUCTION

This article explains how to install IBM/4.3. Most sites will have several IBM RT PCs and many IBM 6152s. However, in a very small installation, you might have a single IBM RT that functions as master for network installations, server for both the Andrew File System and RVD, and the uucp connection. See Appendix C of this article for guidance in creating such a machine.

The normal installation process differs slightly between the IBM RT PC and the IBM 6152 Academic System. For both, you begin by booting a system from diskette. For the IBM 6152 only, you copy a small root file system image onto a swap area, and boot this. For both, you then load the IBM/4.3 root, user and source file systems. Last, you reboot and then load the remainder of the distributed binaries and sources.

The technique for upgrading a 4.2/RT system is described in Chapter 2 of this article. As IBM/4.3 is upwardly-compatible with 4.2/RT, the upgrade procedure involves extracting a new set of system binaries onto new root and /usr file systems. The sources are then extracted, and local configuration files are merged into the new system. 4.2/RT user file systems may be upgraded in place, and 4.2/RT binaries may be used with IBM/4.3 in the course of conversion. It is desirable to recompile most local software after the conversion, as there are many changes and performance improvements in the compilers, standard libraries, and floating point linkage.

1.1. Software Supported

IBM/4.3 support all of 4.3BSD, excepting those items listed in the "Software Description" found in Appendix A of *IBM Academic Operating System 4.3*.

1.2. Hardware Supported

IBM/4.3 runs on the IBM 6152 Academic System and six models of the IBM RT PC: the IBM 6151 Model 010, 015 and 115 (desk models) and the IBM 6150 Model 020, 025, and 125 (floor models). This section describes the specific configurations supported.

Note: All configurations MUST have 2 MB of contiguous memory starting at address zero.

1.2.1. IBM 6152 Academic System

IBM Academic Operating System 4.3 supports the following configurations of this new system:

- A 6152 Academic System Unit and Enhanced Personal Computer Keyboard with one of the memory and fixed disk configurations listed below:
- 2MB of memory and
  - 1 IBM 20MB Fixed-Disk Drive (Model 022)
- 4MB of memory and one of the following:
  - 1 IBM 20MB Fixed-Disk Drive (Model 024)
  - 1 IBM 44MB Fixed-Disk Drive (Model 044)
  - 1 IBM 70MB Fixed-Disk Drive (Model 074)
- 8MB of memory and one of the following:
  - 1 IBM 44MB Fixed-Disk Drive (Model 048)
  - 1 IBM 70MB Fixed-Disk Drive (Model 078)

1.2.1.1. Additional Standard Features of the IBM 6152

- An MC68881 Floating Point unit
- A Serial Asynchronous port
- A Parallel port
- A Pointing Device port
- An IBM 8770 PS/2 Mouse
- A Video Graphics Array (VGA) and display port
- A Diskette Controller
- A 1.44MB (3.5 inch) Diskette Drive

1.2.1.2. Optional Features of the IBM 6152

- A second IBM 44MB Fixed-Disk Drive (Models 044 or 048)
- A second IBM 70MB Fixed-Disk Drive (Models 074 or 078)
- The IBM 8503 PS/2 Monochrome Display
- The IBM 8604 PS/2 Monochrome Display
- An IBM PS/2 Memory expansion for advanced color (#4081)
- The IBM 8513 PS/2 Color Display
- An IBM PS/2 Color Graphics Display Adapter 8514/A (#4054)
- The IBM 8514 PS/2 Color Display
- Up to two local area network adapters (one required; see "Local Area Networks" below)

1.2.2. IBM 6151 Model 010 Processor

IBM/4.3 for the Model 010 supports the following hardware:

- A 6151 Model 010 System Unit and Keyboard
- 2 MB of memory (but not as two separate 1 MB memory cards)
- An IBM PC AT Fixed-Disk and Diskette Drive Adapter (#3428) (standard)
- An IBM RT PC 40 MB Fixed-Disk Drive (#4735) (standard)
- An IBM PC AT High Capacity (1.2 MB) Diskette Drive (#0206) (standard)

1.2.3. IBM 6151 Model 015 Processor

IBM/4.3 for the Model 015 supports the following hardware:

- A 6151 Model 015 System Unit and Keyboard
- 2 MB of memory
- An IBM RT PC Enhanced Small Device Interface (ESDI) Magnetic Media Adapter (#6341) (standard)
- An IBM RT PC 70 MB ESDI Fixed-Disk Drive (#6941) (standard)
- An IBM PC AT High Capacity (1.2 MB) Diskette Drive (#0206) (standard)

1.2.4. IBM 6151 Model 115 Processor

IBM/4.3 for the Model 115 supports the following hardware:

- A 6151 Model 115 System Unit and Keyboard with 4 MB of memory
- Advanced Processor Card with built-in MC68881 floating-point unit
- An IBM RT PC Extended ESDI Magnetic Media Adapter
- An IBM RT PC 70 MB Extended ESDI Fixed-Disk Drive (#3988) (standard)
- An IBM PC AT High Capacity (1.2 MB) Diskette Drive (#0206) (standard)

1.2.5. IBM 6150 Model 020 Processor

IBM/4.3 for the Model 020 supports the following hardware:

- A 6150 Model 020 System Unit and Keyboard
- 2 MB of memory (but not as two separate 1 MB memory cards)
- An IBM PC AT Fixed-Disk and Diskette Drive Adapter (#3428) (standard)
- 2 Asynchronous (RS-232-C) Serial Ports
- An IBM RT PC 40 MB Fixed-Disk Drive (#4735) (standard)
- An IBM PC AT High Capacity (1.2 MB) Diskette Drive (#0206) (standard)

1.2.6. IBM 6150 Model 025 Processor

IBM/4.3 for the Model 025 supports the following hardware:

- A 6150 Model 025 System Unit and Keyboard
- 2 MB of memory
- An IBM RT PC ESDI Magnetic Media Adapter (#6341) (standard)
- 2 Asynchronous RS232C Serial Ports
- An IBM RT PC 70 MB ESDI Fixed-Disk Drive (#6941) (standard)
- An IBM PC AT High Capacity (1.2 MB) Diskette Drive (#0206) (standard)

1.2.7. IBM 6150 Model 125 Processor

IBM/4.3 for the Model 125 supports the following hardware:

- A 6150 Model 125 System Unit and Keyboard with 4 MB of memory
- Advanced Processor Card with built-in MC68881 floating-point unit
- An IBM RT PC Extended ESDI Magnetic Media Adapter
- 2 asynchronous RS232C Serial Ports
- An IBM RT PC 70 MB Extended ESDI Fixed-Disk Drive (#3988) (standard)
- An IBM PC AT High Capacity (1.2 MB) Diskette Drive (#0206) (standard)

1.2.8. Peripherals and Optional Features

The following peripherals and options are supported on all 0xx-series models:

- An IBM RT PC 1 MB Memory Expansion (#8222) or an IBM RT PC 2 MB Memory Expansion (#4739) or an IBM RT PC 4 MB Memory Expansion (#3156) (up to 8 MB total)

The following peripherals and options are supported on all 1xx-series models:

- An IBM RT PC 4 MB Fast Memory Expansion (#7004) or an IBM RT PC 8 MB Fast Memory Expansion (#7008) (up to 16 MB total)

The following peripherals and options are supported on all models of the 6150, except where noted:

- Up to 2 additional 40 MB Fixed-Disk Drives, 70 MB ESDI Fixed-Disk Drives, or 70 MB or 114 MB Extended ESDI Fixed-Disk Drives (Model 125 only).

  Note: When mixing drives of different types, separate adapters are needed. No more than two adapters can be installed on a given machine, hence no more than two types of drive can be installed on a given machine. Except in the case where all three drives are of the extended ESDI type, two adapters are needed to install a third drive. The Extended ESDI Fixed-Disk Drive is only supported on the 1xx-series models.

- A second diskette drive, which may be either the IBM PC AT High Capacity (1.2 MB) Diskette Drive or the IBM PC AT Dual-Sided (360 KB) Diskette Drive (#0207)

The following optional features are supported on all models of the IBM RT PC:

- An IBM RT PC Four-Port Asynchronous RS232C Adapter (#4763)
- An IBM PC AT Serial/Parallel Adapter (#0215)
- An IBM 6157 Streaming Tape Drive with adapter (#4797)
- Either of two Local Area Network Adapters (see "Local Area Networks" below)
- IBM All-Points-Addressable Displays (see "IBM All-Points-Addressable Displays" below)
- An IBM 5151 Monochrome Display attached to an IBM Monochrome Display and Printer Adapter (#4900)
- An IBM RT PC Floating Point Accelerator (#4758)
- An IBM RT PC Mouse (#8426)
- An IBM 3812 Pageprinter
- An IBM 4201 Proprinter or an IBM 5152 Graphics Printer (withdrawn from marketing)
- An IBM RT PC Small Computer System Interface Adapter (#7000), used to attach IBM 9332 DASD Models 240, 250, 440, and 450

1.2.8.1. Local Area Networks

You can install up to two LAN adapters in a single IBM 6150, IBM 6151, or IBM 6152 system unit. The two-adapter configuration can be either two IBM Token-Ring adapters, two Ethernet adapters, or one of each. At least one adapter must be installed in the IBM 6152.

1.2.8.1.1. IBM Token-Ring Network:

An IBM Token-Ring Network Adapter provides connection of the IBM 6150, 6151, or 6152 to the IBM Token-Ring Network. The network uses the IBM Cabling System for physical connection and a Token-Ring access protocol for network traffic control.

1.2.8.1.2. Ethernet:

The IBM RT PC Baseband Adapter (#6810) provides connection of the IBM 6150 or 6151 to an Ethernet network. The Ungermann-Bass Ethernet Adapter NICps/2 provides connection of the IBM 6152 to an Ethernet network. This adapter is not available from IBM; you must supply it.

Two Ethernet cards cannot be used in conjunction with a 8514/A in the 6152.

1.2.8.2. IBM All-Points-Addressable Displays

IBM/4.3 supports the following All-Points-Addressable displays and their associated adapters.

On the IBM RT PC:

(1) An IBM 6153 Advanced Monochrome Graphics Display

(2) An IBM 6154 Advanced Color Graphics Display

(3) An IBM 6155 Extended Monochrome Graphics Display

(4) The IBM Enhanced Graphics Adapter (EGA)

(5) The IBM 5154 Enhanced Color Display

(6) The IBM 5081 Display with MegaPel Adapter

(7) An IBM Academic Information Systems experimental display, which is no longer available.

On the IBM 6152 Academic System:

(1) The IBM 8604 PS/2 Monochrome Display

(2) The IBM 8514 PS/2 Color Display

1.2.8.3. Documentation

Each Customer Central Support Site should have a copy of the following manuals; if additional copies are required, they may be ordered from an IBM representative:

- IBM RT PC 6150 System Unit Hardware Maintenance and Service, SV21-8025
- IBM RT PC 6151 System Unit Hardware Maintenance and Service, SV21-8026
- IBM RT PC User Setup Guide, SV21-8020
- IBM RT PC Guide to Operations, SV21-8021
- IBM RT PC Problem Determination Guide, SV21-8022
- IBM RT PC Hardware Technical Reference, SV21-8024
- IBM RT PC Site Preparation Guide, GA23-1058
- IBM 6152 Academic System Hardware Maintenance Reference, SA 38-0000
- IBM 6152 Academic System Hardware Maintenance Service, SA38-0001
- IBM 6152 Academic System Technical Reference, SA38-0002
- IBM 6152 Academic System Quick Reference, SA38-0003
- IBM 6152 Academic System Setup Guide, SS68X-2209
- IBM PS/2 Hardware Maintenance Reference, S68X-2221
- IBM PS/2 Maintenance Service, S68X,2222

- IBM PS/2 Technical Reference, S68X-2224

1.2.9. Minimum Configuration Requirements

At least one IBM RT PC (the master) must have at least two 70 MB Fixed-Disk Drives (of any type) and an IBM 6157 Streaming Tape Drive with adapter for tape installation. All other machines may use either tape installation or have a network connection to the master machine for network installation.

1.3. Distribution Format

The distribution includes a cover letter and program directory, installation diskettes, distribution streaming tapes, and copies of the book, *IBM Academic Operating System 4.3*.

The distribution includes all 4.3BSD source files, whether ported or not. (User-contributed software is not provided. The source distribution is available from the University of California at Berkeley.) However, the distribution includes binary files for only those programs which have been ported and tested.

1.4. IBM/4.3 Device Naming

IBM/4.3 has a set of names for devices which are different from the RT hardware names, such as:

Fixed (Hard) Disks	hd
Diskettes (Floppies)	fd
6157 Streaming Tape	st

The normal standalone system, used to bootstrap the full IBM/4.3 system, uses device names of the form:

xx(y,z)

where *xx* is either **hd** or **fd**. The value *y* specifies the adapter to use and also the device. It is computed as

N * *adapter* + *device*

where *N* is either 2 or 3 (it is 3 only for fixed disks when adapter 0 is of the Extended ESDI type); *adapter* is the adapter to which the device is connected; and *device* is the device number on the adapter.

For example, the first disk (disk 0) on the first adapter (adapter 0) would have a *y* value of 0. The first disk on the second adapter would have a *y* value of either 2 or 3. The value *z* (in the range 0-7) is interpreted as a disk *partition* (in the range 0-7) for the hard disk, and is ignored for the floppy disk.

A IBM/4.3 physical (hard) disk is divided into eight logical disk partitions, each of which may occupy any consecutive cylinder range on the physical device. The cylinders occupied by the eight default partitions for each drive type are specified in the disk description file /etc/disktab (see *disktab*(5)). Non-standard partition sizes may be specified on a per disk basis (see *minidisk*(8R)). Each partition may be used either to store a IBM/4.3 file system or as a raw data area (such as a paging area). Typically the following is used:

- The first partition (partition 0 on drive 0) stores a root file system from which IBM/4.3 can be bootstrapped. The name of this partition is hd(0,0) for standalone programs, and /dev/hd0a for programs run under the kernel.
- The second partition -- hd(0,1) or /dev/hd0b -- is a paging area.
- The third partition -- hd(0,6) or /dev/hd0g -- is used for the /usr file system.

This partition can be used when making a backup copy. Such a backup must be restored onto the same disk from which it was backed up. Be extremely careful when you use this partition. The first cylinder contains bad sector forwarding and configuration information. If you overwrite these sectors, you will erase the bad block information.

It is a good idea to copy the first 4 tracks of each disk in case of accident. The copy should be placed on a separate disk or diskette so it can be recovered if needed. The standalone COPY command can be used to copy and restore this information.

The remaining five partitions (numbered 3 through 7) can be used for additional mounted file systems. Partition 6 -- hd(0,6) or /dev/hd0g -- is normally used for the /usr mounted file system.

1.5. IBM/4.3 Devices: Block and Raw

IBM/4.3 makes a distinction between "block" and "raw" (character) devices. Each disk has a block device interface that makes the device byte-addressable; you can write a single byte anywhere on the disk. The system reads the data from the disk sector, inserts the byte to be written, and writes the modified data. Names like /dev/hd0a indicate block devices. There are also raw devices available. These have names like /dev/rhd0a, the "r" indicating "raw." Raw devices bypass the buffer cache and perform I/O directly to/from the program's I/O buffers; they are normally restricted to full-sector transfers. The bootstrap procedure often uses the raw device interfaces because these tend to work faster in some cases. Raw devices are used when making new file systems, when checking unmounted file systems, or for copying quiescent file systems. The block devices are used to mount file systems or to operate on a mounted file system such as the root.

Be aware that it is often important which interface is used: the character device interface (for efficiency), or the block device interface (to write specific bytes within a sector). Do not indiscriminately change the installation instructions to use the alternate type of device interface.

2. SAVING AND RESTORING LOCAL MODIFICATIONS

This chapter is intended for those sites which:

(1) Are currently running a release of 4.2/RT or IBM/4.3,

AND

(2) Are about to install an entirely new release of IBM/4.3 to replace the existing system.

If you do not have a previous release of 4.2/RT or IBM/4.3 installed, skip this chapter and go to Chapter 3, "Installation Procedures."

If you are installing an entirely new release of IBM/4.3 to replace an existing 4.2/RT or IBM/4.3 system, you will probably want to back up all files as described in the section "Backing Up Your System." Then, after you have installed the new version of IBM/4.3, you can selectively restore locally modified files.

2.1. Backing Up Your System

It is highly recommended that you do a complete system backup so that you can selectively restore the files you want after installing the new release. On a standard system there are three file systems to back up: hd0a, hd0g and hd1g. If your site has a standard procedure for backing up these file systems, use that procedure instead of the following. (For example, you might use *rdump*(8) over the network to another RT with a streaming tape drive attached.)

2.1.1. Using Streaming Tapes for Backup

If your site does not have its own backup procedures, you can use the following instructions to back up your file systems to streaming tape. This method assumes you have a streaming tape drive on your RT. You need one blank streaming tape for each of the three file systems.

To back up your system, do the following:

(1) Erase each of the three streaming tapes. To erase a tape, insert it in the streaming tape drive and type:

```
mt -f /dev/rst0 erase  <Enter>
```

(2) Dump each file system onto tape. Begin by typing the superuser command (**su**) and supplying the superuser password.

(3) Insert an erased tape into the streaming tape drive and type:

```
dump 0sf 4400 /dev/st0 /dev/rhd0a  <Enter>
```

This tape will be referred to later as the "rhd0a" tape.

(4) Remove the first tape from the streaming tape drive, insert the second tape and type:

```
dump 0sf 4400 /dev/st0 /dev/rhd0g  <Enter>
```

This tape will be referred to later as the "rhd0g" tape.

(5) Remove the second tape from the streaming tape drive, insert the third tape and type:

```
dump 0f /dev/st0 /dev/rhd1g  <Enter>
```

This tape will be referred to later as the "rhd1g" tape.

2.1.2. Using Diskettes for Backup

If your site does not have its own backup procedures, you can use the following instructions to back up your file systems to diskette.

To back up your system, do the following:

(1) Determine the size of your file systems. To do so, type the following command:

```
df
```

The size in Kbytes is listed under the column headed "used." Multiply that number by 1024 before using it in the formula in the next step.

(2) Determine the approximate number of diskettes needed to back up each file system, using the following formula:

number of 1.2M diskettes = (file system size in bytes) / 1228800

Round fractions up, then total the number of diskettes needed for all file systems. Be sure to have enough usable diskettes before beginning.

(3) To format a 1.2M diskette, insert the diskette in the upper diskette drive and type the following command:

```
fdformat -h /dev/rfd0
Format /dev/rfd0 to high density? yes
```

Repeat this step for each diskette that needs formatting.

(4) Dump each file system onto diskette. Begin by typing the superuser command (**su**) and supplying the superuser password.

(5) Insert a formatted diskette into the upper diskette drive drive and type:

```
dump 0sf 70 /dev/rfd0 /dev/rhd0a <Enter>
```

As you remove each diskette during this step, be sure to label it appropriately, indicating both the file system (rhd0a) and which diskette (1st, 2nd, 3rd, etc.) it is in the sequence.

(6) Insert a formatted diskette into the upper diskette drive and type:

```
dump 0sf 70 /dev/rfd0 /dev/rhd0g <Enter>
```

As you remove each diskette during this step, be sure to label it appropriately, indicating both the file system (rhd0g) and which diskette (1st, 2nd, 3rd, etc.) it is in the sequence.

(7) Insert a formatted diskette into the upper diskette drive and type:

```
dump 0sf 70 /dev/rfd0 /dev/rhd1g <Enter>
```

As you remove each diskette during this step,
be sure to label it appropriately,
indicating both the file system (rhd1g)
and which diskette (1st, 2nd, 3rd, etc.) it is in the sequence.

NOTES:

(1) Both *dump*(8) and *restore*(8) use the raw diskette device.

(2) Be sure to label the diskettes appropriately during the backup to facilitate a restore from diskettes.

2.2. Converting from 4.2/RT to IBM/4.3

Begin by reading the article entitled "Bug Fixes and Changes in 4.3 BSD" in the *UNIX System Manager's Manual* and the "Summary of Amendments" (if any) in *IBM Academic*

Operating System 4.3 to see what has changed since the last release of 4.2/RT. If you have local system modifications to the kernel to install, look at the article entitled "Changes to the Kernel in 4.3 BSD" (in the *UNIX System Manager's Manual*) to learn how the system changes will affect your local modifications.

If you are running 4.2/RT, upgrading your system involves replacing your kernel and system utilities. Most binaries compiled under 4.2/RT will work without recompilation under IBM/4.3, though they may run faster if they are relinked. The easiest way to convert to IBM/4.3 (depending on your file system configuration) is to install new root and /usr file systems from the distribution ROOT/USER tape on an unused disk's a and g partitions, boot the new system, and then copy any local utilities from your old root and /usr file systems into the new ones. All user file systems and binaries can be retained unmodified, except that the new *fsck* should be run before they are mounted (see Section 2.3.1).

Section 2.3 lists the files to be saved as part of the conversion process. Section 2.4 describes the bootstrap process. Section 2.5 discusses the merger of the saved files back into the new system. Section 2.6 provides general hints on possible problems to be aware of when converting from 4.2/RT to IBM/4.3.

2.3. Files to Save

The easiest upgrade path from 4.2/RT is to install new root and /usr file systems on an unused disk's a and g partitions, then copy or merge site specific files into their corresponding files on the new system. The list on the next page enumerates the standard set of files you will want to save and suggests directories in which site-specific files should be present. This list will likely be augmented with non-standard files you have added to your system. If you do not have enough space to create parallel file systems, you should create a *tar* image of the following files before installing IBM/4.3.

/.cshrc	†	root csh startup script
/.login	†	root csh login script
/.profile	†	root sh startup script
/.rhosts	†	for trusted machines and users
/dev/MAKEDEV	‡	in case you added anything here
/dev/MAKEDEV.local	*	for making local devices
/etc/disktab	‡	in case you changed disk partition or block sizes
/etc/fstab	†	disk configuration data
/etc/ftpusers	†	for local additions
/etc/gateways	†	routing daemon database
/etc/gettytab	‡	getty database
/etc/group	*	group data base
/etc/hosts	†	for local host information
/etc/hosts.equiv	†	for local host equivalence information
/etc/networks	†	for local network information
/etc/passwd	*	user data base
/etc/printcap	†	line printer database
/etc/protocols	‡	in case you added any local protocols
/etc/rc	*	for any local additions
/etc/rc.local	*	site specific system startup commands
/etc/remote	†	auto-dialer configuration
/etc/services	‡	for local additions
/etc/syslog.conf	*	system logger configuration
/etc/securettys	*	for restricted list of ttys where root can log in
/etc/ttys	*	terminal line configuration data
/etc/ttytype	*	terminal line to terminal type mapping data
/etc/termcap	‡	for any local entries that may have been added
/lib	‡	for any locally developed language processors
/usr/dict/*	‡	for local additions to words and papers
/usr/hosts/MAKEHOSTS	†	for local changes
/usr/include/*	‡	for local additions
/usr/lib/aliases	†	mail forwarding data base
/usr/lib/crontab	*	cron daemon data base
/usr/lib/font/*	‡	for locally developed font libraries
/usr/lib/lib*.a	†	for locally developed libraries
/usr/lib/lint/*	‡	for locally developed lint libraries
/usr/lib/sendmail.cf	*	sendmail configuration
/usr/lib/tabset/*	‡	for locally developed tab setting files
/usr/lib/term/*	‡	for locally developed nroff drive tables
/usr/lib/tmac/*	‡	for locally developed troff/nroff macros
/usr/lib/uucp/*	†	for local uucp configuration files
/usr/man/manl	†	for manual pages for locally developed programs
/usr/msgs	†	for current msgs
/usr/spool/*	†	for current mail, news, uucp files, etc.
/usr/src/local	†	for source for locally developed programs
/sys/conf/HOST	†	configuration file for your machine
/sys/conf/files.HOST	†	list of special files in your kernel
/*/quotas	†	file system quota files

†Files that can be used from 4.2/RT without change. ‡Files that need local modifications merged into IBM/4.3 files. *Files that require special work to merge and are discussed below.

2.4. Building a IBM/4.3 System

There are two approaches to upgrading from 4.2/RT to IBM/4.3. In the first approach, one loads the IBM/4.3 distribution into unused partitions of an existing disk, boots IBM/4.3 from this disk, optionally builds a tailored kernel for the configuration and boots it, mounts the old (existing) 4.2/RT partitions onto the IBM/4.3 system, merges the old files into the new system, and frees up the old partitions. In the second approach, one backs up the existing system, loads the IBM/4.3 distribution as if bootstrapping a brand-new system, and restores and merges the old files into the new system.

The method chosen depends upon a number of factors, including the amount of available disk space and the expertise of the installer.

Users without a third disk or without much expertise might find it easier to use the second approach, while experienced users may find the first approach faster.

The first approach is described below. If you elect to follow the second approach, you should still read the rest of this chapter. You will return to these instructions after you install IBM/4.3 using the instructions in the following chapter entitled "Installation Procedures."

To build a working IBM/4.3 system, following the steps in the next chapter, specifying an unused disk as the root device as described in the section "Changing Installation Options." The root and /usr file system dump on the tape could also be extracted directly, although this will require an additional file system check after booting IBM/4.3 to convert the new root file system. The exact procedure chosen will depend on the disk used for IBM/4.3, but the following procedure demonstrates extraction onto hd2. After becoming ROOT, issue the following commands:

```
umount -a
newfs /dev/hd2a[1] <Enter>
fsck /dev/rhd2a <Enter>
newfs /dev/hd2g <Enter>
fsck /dev/rhd2g <Enter>
mount /dev/hd2a /mnt <Enter>
mkdir /mnt/usr <Enter>
mount /dev/hd2g /mnt/usr <Enter>
cd /mnt <Enter>
restore xvf /dev/st0 <Enter>
```

If there is insufficient space to load the new root and /usr file systems before reusing the existing 4.2/RT partitions, it is strongly advised that you make full dumps of each file system on streaming tapes before beginning. It is also desirable to run file system checks of all file systems to be converted to IBM/4.3 before shutting down 4.2/RT. In either case, this is an excellent time to review your disk configuration for possible tuning of the layout. The "Disk Configuration" section in the "System Setup" chapter is required reading.

To ease the transition to new kernels, the IBM/4.3 bootstrap routines now pass the identity of the boot device through to the kernel. The kernel then uses that device as its root file system. Thus, for example, if you boot from /dev/hd1a, the kernel will use hd1a as its root file system. If /dev/hd1b is configured as a swap partition, it will be used as the initial swap area. Otherwise, the normal primary swap area (/dev/hd0b) will be used. The IBM/4.3 bootstrap is backward compatible with 4.2/RT, so you can replace your 4.2/RT bootstrap if you use it to boot your first IBM/4.3 kernel.

[1] You might need to change the free space percentage to four or five percent.

Once you have extracted the IBM/4.3 system and booted from it, you will have to build a kernel customized for your configuration. If you have any local device drivers, they will have to be incorporated into the new kernel. See the section "Building New System Images" in the chapter "System Setup" of this article, and the article "Building IBM/4.3 Systems with Config."

The disk partitions in IBM/4.3 are the same as those in 4.2/RT. 4.2/RT file systems may be converted in place. This is done by using the IBM/4.3 version of *fsck*(8) on each file system and allowing it to make the necessary corrections. The new version of *fsck* is more strict about the size of directories than the version supplied with 4.2/RT. Thus the first time that it is run on a 4.2/RT file system, it will produce messages of the form:

```
DIRECTORY ...: LENGTH xx NOT MULTIPLE OF 512 (ADJUSTED)
```

Length "xx" will be the size of the directory; it will be expanded to the next multiple of 512 bytes. Note that file systems are otherwise completely compatible between 4.2/RT and IBM/4.3, though running a IBM/4.3 file system under 4.2/RT may cause more of the above messages to be generated the next time it is *fsck*'ed on IBM/4.3.

2.5. Merging your files from 4.2/RT into IBM/4.3

When your system is booting reliably and you have the IBM/4.3 root and /usr file systems fully installed, you will be ready to continue with the next step in the conversion process, merging your old files into the new system.

If you saved the files on a *tar* tape, you can extract them into a scratch directory (say /usr/convert), as in the following example:

```
# mkdir /usr/convert <Enter>
# cd /usr/convert <Enter>
# tar xf /dev/rst0 <Enter>
```

If you used *dump*, see Section 3.7 below.

The data files marked in the previous table with a dagger (†) may be used without change from the previous system. Those data files marked with a double dagger (‡) have syntax changes or substantial enhancements. You should start with the IBM/4.3 version and carefully integrate any local changes into the new file. Usually these local modifications can be incorporated without conflict into the new file; some exceptions are noted below. The files marked with an asterisk (*) require particular attention and are discussed below.

If you have any homegrown device drivers in /dev/MAKEDEV.local that use major device numbers reserved by the system you will have to modify the commands used to create the devices or alter the system device configuration tables in /sys/ca/conf.c. Otherwise /dev/MAKEDEV.local can be used without change from 4.2/RT.

System security changes require adding several new "well-known" groups to /etc/group. The groups that are needed by the system as distributed are:

name	number
wheel	0
daemon	1
kmem	2
sys	3
tty	4
operator	5
staff	10

Only users in the "wheel" group are permitted to *su* to "root." Most programs that manage directories in /usr/spool now run set-group-id to "daemon" so that users cannot directly access the files in the spool directories. The special files that access kernel memory, /dev/kmem and /dev/mem, are made readable only by group "kmem." Standard system programs that require this access are made set-group-id to that group. The group "sys" is intended to control access to system sources, and other sources belong to group "staff." Rather than make users' terminals writable by all users, they are now placed in group "tty" and made only group writable. Programs that should legitimately have access to write on user's terminals such as *talk* and *write* now run set-group-id to "tty." The "operator" group controls access to disks. By default, disks are readable by group "operator," so that programs such as *df* can access the file system information without being set-user-id to "root."

Several new users have also been added to the group of "well-known" users in /etc/passwd. The current list is:

name	number
root	0
daemon	1
operator	2
uucp	66
nobody	32767

The "daemon" user is used for daemon processes that do not need root privileges. The "operator" user-id is used as an account for dumpers so that they can log in without having the root password. By placing them in the "operator" group, they can get read access to the disks. The "uucp" login has existed long before IBM/4.3, and is noted here just to provide a common user-id. The password entry "nobody" has been added to specify the user with least privilege.

After installing your updated password file, you must run *mkpasswd*(8) to create the *ndbm* password database. Note that *mkpasswd* is run whenever *vipw*(8) is run.

The format of the cron table, /usr/lib/crontab, has been changed to specify the user-id that should be used to run a process. The userid "nobody" is frequently useful for non-privileged programs.

Some of the commands previously in /etc/rc.local have been moved to /etc/rc; several new functions are now handled by /etc/rc.local. You should look closely at the prototype version of /etc/rc.local and read the manual pages for the commands contained in it before trying to merge your local copy. Note in particular that *ifconfig* has had many changes, and that host names are now fully specified as domain-style names (e.g, monet.Berkeley.EDU) for the benefit of the name server.

The C library and system binaries on the distribution tape are compiled with new versions of *gethostbyname* and *gethostbyaddr* which use the name server, *named*(8). If you have only a small network and are not connected to a large network, you can use the distributed library routines without any problems; they use a linear scan of the host table /etc/hosts if the name server is not running. If you are on the DARPA Internet or have a large local network, it is recommended that you set up and use the name server. For instructions on how to set up the necessary configuration files, refer to "Name Server Operations Guide for BIND." Several programs rely on the host name returned by *gethostname* to determine the local domain name.

If you want to compile your system to use the host table lookup routines instead of the name server, you will need to modify /usr/src/lib/libc/Makefile according to the instructions there and then recompile all of the system and local programs (see the section "Recompiling and Reinstalling System Software" in the chapter "System Operation" of this article). Next, you must run *mkhosts*(8) to create the *ndbm* host table database from /etc/hosts.

The format of /etc/ttys has changed, see *ttys*(5) for details. It now includes the terminal type and security options that were previously placed in /etc/ttytype and /etc/securettys.

There is a new version of *syslog* that uses a more generalized facility/priority scheme. This has changed the format of the syslog.conf file. See *syslogd*(8) for details. *Syslog* now logs kernel errors, allowing events such as soft disk errors, file-system-full messages, and other such error messages to be logged without slowing down the system while the messages print on the console. It is also used by many of the system daemons to monitor system problems more closely, for example network routing changes.

If you are using the name server, your *sendmail* configuration file will need some minor updates to accommodate it. See the "Sendmail Installation and Operation Guide" and the sample *sendmail* configuration files in /usr/src/usr.lib/sendmail/nscf. Be sure to regenerate your *sendmail* frozen configuration file after installation of your updated configuration file.

The *init*(8) utility will prompt for the root password before invoking a super-user shell on the console (if the file /etc/nosingle exists).

The spooling directories saved on tape may be restored in their eventual resting places without too much concern. Be sure to use the 'p' option to *tar* so that files are recreated with the same file modes:

```
# cd /usr <Enter>
# tar xfp /dev/rst0 msgs spool/mail spool/uucp spool/uucppublic spool/news <Enter>
```

The ownership and modes of two of these directories need to be changed from their 4.2/RT values. *at* now runs set-user-id "daemon" instead of root. Also, the uucp directory no longer needs to be publicly writable, as *tip* reverts to privileged status to remove its lock files. After copying your version of /usr/spool, you should do the following:

```
# chown −R daemon /usr/spool/at <Enter>
# chown −R root /usr/spool/uucp <Enter>
# chgrp −R daemon /usr/spool/uucp <Enter>
# chmod −R o−w /usr/spool/uucp <Enter>
```

Whatever else is left is likely to be site specific or require careful scrutiny before placing in its eventual resting place. Refer to the documentation and source code before arbitrarily overwriting a file.

2.6. Hints on converting from 4.2/RT to IBM/4.3

This section summarizes the most significant changes between 4.2/RT and IBM/4.3, particularly those that are likely to cause difficulty in doing the conversion. It does not include changes in the network; see Chapter 5 for information on setting up the network.

The mailbox locking protocol has changed; it now uses the advisory locking facility to avoid concurrent update of users' mail boxes. If you have your own mail interface, be sure to update its locking protocol.

The kernel's limit on the number of open files has been increased from 20 to 64. It is now possible to change this limit almost arbitrarily (there used to be a hard limit of 30). The standard I/O library autoconfigures to the kernel limit. Note that file ("_iob") entries may be allocated by *malloc* from *Ufopen*; this allocation has been known to cause problems with programs that use their own memory allocators. This does not occur until after 20 files have been opened by the standard I/O library.

Select can be used with more than 32 descriptors by using arrays of **ints** for the bit fields rather than single **ints**. Programs that used *getdtablesize* as their first argument to *select* will no longer work correctly. Usually the program can be modified to correctly specify the number of bits in an **int**. Alternatively the program can be modified to use an array of **ints**. There are a set of macros available in <sys/types.h> to simplify this. See *select*(2).

Old core files will not be intelligible by the current debuggers because of numerous changes to the user structure and because the kernel stack has been enlarged. The *a.out* header that was in the user structure is no longer present. Locally-written debuggers that try to check the magic number will need modification.

Find now has a database of file names, constructed once a week from *cron.* To find a file by name only, the command *find name* will look in the database for files that match the name. This is much faster than "find / −name name −print."

Files may not be deleted from directories having the "sticky" (ISVTX) bit set in their modes except by the owner of the file or of the directory, or by the superuser. This is primarily to protect users' files in publicly-writable directories such as /tmp and /usr/tmp. All publicly-writable directories should have their "sticky" bits set with "chmod +t."

The include file <time.h> has returned to /usr/include, and again contains the definitions for the C library time routines of *ctime*(3).

The *compact* and *uncompact* programs have been supplanted by the faster *compress.* If your user population has *compact*ed files, you will want to install *uncompact* found in /usr/src/old/compact.

The configuration of the virtual memory limits has been simplified. A MAXDSIZ option, specified in bytes in the machine configuration file, may be used to raise the maximum process region size from the default of 17 MB to 32 MB or 64 MB. The initial per-process limit is still 6 MB, but can be raised up to MAXDSIZ with the *csh limit* command.

Some IBM/4.3 binaries will not run with a 4.2/RT kernel because they take advantage of new functionality in IBM/4.3. One noticeable example of this problem is *csh.*

If you want to use *ps* after booting a new kernel, and before going multiuser, you must initialize its name list database by running "ps −U."

2.7. Restoring Selected Files

The interactive option of *restore* lets you traverse the directories on the dump tape and select the files you want to restore.

(1) To restore selected files from the "rhd0a" tape, begin by typing the superuser command (**su**) and supplying the superuser password. Make sure the file systems are mounted, e.g. you are in multi-user mode.

(2) Type:

```
cd / <Enter>
restore if /dev/st0 <Enter>
```

(3) Respond to the *restore* prompt by typing:

```
restore > add filenamex <Enter>
```

where *filenamex* is the pathname of the file you want restored. For example, to restore the etc/hosts and etc/passwd files, you would type:

```
restore > add etc/hosts <Enter>
restore > add etc/passwd <Enter>
```

(4) To extract the desired file(s) from tape, type:

```
restore > extract <Enter>
restore > quit <Enter>
```

If at any time you are prompted for a tape or volume number, type **1**.

To restore files from the "rhd0g" tape, repeat the preceding steps, except in Step 2, type the following:

```
cd /usr <Enter>
```

instead of:

```
cd / <Enter>
```

To restore files from the "rhd1g" tape, repeat the preceding steps, except in Step 2, type the following:

```
cd /usr/src <Enter>
```

instead of:

```
cd / <Enter>
```

For more information on either *dump*(8) or *restore*(8), see the appropriate man page.

Note: 4.2/RT notesfiles are not readable by IBM/4.3.

3. INSTALLATION PROCEDURES

This chapter, intended for the system manager, describes how to install IBM/4.3 on the IBM RT PC models 6150 and 6151 supported by this release and on the IBM 6152 Academic System.

3.1. Installation Procedures for the IBM RT PC

Instructions are provided for the following tasks:

- Changing disk partition sizes
- Installing IBM/4.3 on a model 6150 using streaming tape (**Note**: This requires two 70 MB disks)
- Installing IBM/4.3 on any model IBM RT PC from a model 6150 using a network connection

Upon successfully completing installation, a model 6150 with a system installed from streaming tape will contain a complete IBM/4.3 system, including all binary and source files. Any IBM RT PC model with a system installed across a network will contain the base system, which includes only binary files. If you want source files installed during a network installation, see the section "Changing Installation Options," later in this chapter.

3.1.1. Creating Disk Partition Sizes

The SAUTIL diskette includes the standalone utilities described in *sautil*(8r). Among the functions provided by these utilities are the ability to format and partition disks, change the default boot order, and list file systems.

The installation procedure will use default disk partition sizes. Some sites may wish to specify different partition sizes, especially for systems with a single 40-megabyte disk. If you wish to specify different sizes, you must modify the partition tables at the beginning of the installation process. See "Disk Configuration" in the "System Setup" chapter of this article.

Use the following table of approximate minimum sizes for system components to determine the minimum sizes of the disk partitions required for installation. For comparison, typical standard partition sizes are given. Note that the minimum sizes leave no space for user files, for temporary and spool files, or for recompiling the kernel and/or utilities. (Sizes are number of 512-byte sectors.)

	Minimum Sizes		Standard Sizes	
Partition	Tape	Network	40M	70M
hd0a	13166	13166	15884	15884
hd0g	76006	54859	59857	91476
hd1g	64514	---	59857	91476

To format the disks and create partition sizes, proceed as follows:

(1) Power on the display, if necessary.

(2) Insert the diskette labeled SAUTIL in the diskette drive. When inserting a diskette into a diskette drive, always hold the diskette with the label side up and the notch on the side of the diskette facing left. If your IBM RT PC has two diskette drives, insert the diskette in the upper drive.

(3) Power on the model 6150 IBM RT PC. If the system is already powered on, simply press and hold **<Ctrl>** - **<Alt>** - **<Pause>** to reboot from the diskette.

(4) After about 30 seconds, the LED display on the front panel of the IBM RT PC shows "29," indicating that the sautil program is loading.

(5) When the Standalone Utility Menu appears, respond to the *Choice?* prompt by typing **8** **<Enter>** to select the "fdisk" option.

(6) Respond to the prompt:

Device to format?

by typing:

hd(0,2) **<Enter>**

(7) When prompted to verify the format, type **yes** **<Enter>**.

(8) Press **<Enter>** to select the default *severe burnin* format pattern.

The default (severe burnin) pattern is more thorough, but takes more time. If you cannot afford the time for severe burnin, we recommend the zero pattern.

(9) When asked whether to change the default parameters, type **no** **<Enter>**.

(10) When asked whether to preserve data already on the disk, type either **yes** **<Enter>** (if you wish to preserve data on the disk) or **no** **<Enter>** (if you don't).

(11) When asked if the format process should start, type **yes** **<Enter>**.

For severe burnin, the system will take several hours to complete the format process.

(12) When the format is finished, press **<Enter>** to return to the main menu.

(13) If you wish to run AIX with IBM/4.3 or if you want non-standard partition sizes, you need to create minidisk partitions. Continue with this step to do so. Otherwise, skip to step 14 below.

(13a) At the *Choice?* prompt, type **10** **<Enter>** to select the "minidisk" option, which creates IBM/4.3 minidisk partitions.

(13b) When asked which disk to use, type **hd(0,2)** **<Enter>**.

During this step, you will be prompted for partition sizes. To allow adequate swap space on disks smaller than 40 M, select partition sizes by hand; do not use the *standard* command.

(13c) At the > prompt, type **init** **<Enter>**.

This initializes the minidisk table. The init program displays a warning message.

(13d) Reply to the warning message:

Confirm (all data will be lost) [y/n]

by typing **y** to indicate you do want to proceed.

(13e) You may use the command *standard* to create default hd0a, hd0b, and hd0g partitions, or you can create partitions manually.

For each minidisk entry you wish to create manually, use the *create* command. The system will prompt you for the *name*, *iodn*, *size*, and *type* of the minidisk entry.

Use the following information to develop the correct values for your system.

The *iodn* should start at 32736 for the first partition. Each succeeding *iodn* should be incremented by one.

To develop the *size* of each minidisk partition, you must first determine how much unallocated space remains. Use the *list* command to find the size of the pseudo-partition named "FREE." This is the total space available for you to divide among all the minidisk partitions on that disk.

The values for *type* are as follows. The boot partition is type 01, the hd0b (swap) partition is type 20, and all others are type 00.

(13f) Type **quit** **< Enter** to return to the main menu.

(14) Remove the SAUTIL diskette and insert the MINIROOT diskette in the upper drive. Then press **< Ctrl > - < Alt > - < Pause >**.

3.1.2. Installing IBM/4.3 using Streaming Tape

This section describes how to use streaming tapes to install IBM/4.3 on a model 6150 IBM RT PC.

3.1.2.1. Before You Start

You should have the following:

- An installation diskette, labeled MINIROOT
- Two streaming tape cartridges, labeled ROOT/USER and SOURCE
- A model 6150 IBM RT PC with at least two 70 MB hard disks

3.1.2.2. Tape Installation Overview

Installing IBM/4.3 on the IBM RT PC from tape takes about 65 minutes and involves five major tasks:

(1) Booting IBM/4.3 from the MINIROOT diskette

(2) Restoring the IBM/4.3 root (/) and user (/usr) file systems from the ROOT/USER streaming tape (about 30 minutes)

(3) Restoring the IBM/4.3 source file system (/usr/src) from the SOURCE streaming tape (about 30 minutes)

(4) Rebooting IBM/4.3 from the hard disk

(5) Making X-dependent special files

The next section provides step-by-step instructions for completing these tasks.

3.1.2.3. Tape Installation Steps

Information messages appear throughout the installation process. You may ignore all but those mentioned in these instructions.

If at any point in these installation steps an error should occur, see the section "Restarting After an Error," later in this chapter.

(1) Power on the tape drive and the display, if necessary.

(2) Insert the diskette labeled MINIROOT in the diskette drive. When inserting a diskette into a diskette drive, always hold the diskette with the label side up and the notch on the side of the diskette facing left. If your IBM RT PC has two diskette drives, insert the diskette in the upper drive.

(3) Power on the model 6150 IBM RT PC. If the system is already powered on, simply press and hold **<Ctrl>**-**<Alt>**-**<Pause>** to reboot from the diskette.

(4) After about 30 seconds, the LED display on the front panel of the IBM RT PC shows "29," indicating that the boot program is loading.

During the next two minutes, the IBM/4.3 kernel is read into memory and the system is initialized.

After initialization is complete, the screen will display instructions. Follow the instructions, which are paraphrased below.

Do **not** remove the MINIROOT diskette.

To correct typing errors, you can use the backspace key, located in the upper right corner of the typewriter keyboard and marked with a left-facing arrow.

(5) If this is the first time the system has been used, the time and date of the system clock may be incorrect. If so, the system clock will need to be reset before installing IBM/4.3.

Select the menu option to change the system clock. Note that the distributed system uses Pacific Time (Daylight or Standard, as appropriate). If your time zone is different, remember to allow for the difference until you configure a kernel for your time zone. For example, if the local time in New York is 11:05 am on 2 October 1987, one would type "8710020805". (8:05 is the Pacific Time corresponding to 11:05 Eastern Time.)

(6) Select the menu option to proceed with installation. You will be asked to specify tape or network installation. Select the menu option to specify tape installation. You will be asked if you wish to change the installation options.

If you plan to put the /usr/src file system in the Andrew File System, you need to change the installation options. For "select optional system components," you need to select all BUT utility. That is, you need to select kernel, doc, man, font, learn, and notes. See the "Changing Installation Options" section later in this chapter for more information. For normal installation you should not change the installation options.

The tape installation program loads, and prompts you to insert the ROOT/USER tape cartridge:

> > > *Insert 'root/user' tape and press <ENTER>*

(7) Insert the ROOT/USER tape cartridge in the tape drive and press **<Enter>**. When inserting a streaming tape cartridge into a tape drive,

always hold the cartridge with the label side up and the "safe" turnscrew facing right.

During the next 30 minutes, the root (/) and user (/usr) file systems are restored from tape. The name of each file is displayed as the file is restored.

Skip to setp 10 below if you did not select SOURCE as a system component. (See the section "Changing Installation Options" later in this chapter for more information.)

(8) The installation program prompts you to insert the SOURCE tape cartridge in the tape drive:

> > > *Insert 'source' tape and press < ENTER >*

Remove the ROOT/USER tape cartridge. Skip the next step if you wish to install your /src file system on the Andrew File System.

(9) Insert the SOURCE tape cartridge in the tape drive. Then press **< Enter >**.

During the next 30 minutes, the source file system (*/usr/src*) is restored from tape. Remove the SOURCE tape cartridge.

(10) Select the menu option to halt the system for reboot.

When the system LEDs go out, remove the MINIROOT diskette and press:

< Ctrl > - < Alt > - < Pause >

(11) When the boot prompt (:) appears, press **< Enter >**. IBM/4.3 is loaded from the hard disk and proceeds with normal initialization, including autoboot and fsck. After about five minutes, you will see the multi-user prompt:

```
Academic Operating System 4.3 (master) (console)

login:
```

(12) Log in as root and type the following commands:

```
cd /dev
MAKEDEV X
```

This will create X-dependent special files on the master installation machine for propogation to other machiens via network installation.

This completes the streaming tape installation. The model 6150 IBM RT PC contains all the binary files (and optionally all the source files) that comprise IBM/4.3.

3.1.2.4. Backing Up the Newly Installed System

Refer to the instructions in the section "Backing Up Your System" in Chapter 2, "Saving and Restoring Local Modifications."

3.1.2.5. Installation Verification

IBM/4.3 includes one guest account, named "guest," which has no password. To log in before your local accounts have been set up, respond to the login prompt with "guest":

```
Academic Operating System 4.3 (master) (console)

login: guest <Enter>
```

Respond to the prompt

```
TERM = (ibm6155)
```

If you have an IBM 6155, simply press <Enter>.

If you have an IBM 6153, type:

```
ibm6153 <Enter>
```

If you have an IBM 6154, type:

```
ibm6154 <Enter>
```

If you have an IBM ACIS experimental display, type:

```
ibmaed <Enter>
```

If you have an IBM 5151 PC Monochrome display, type:

```
ibmmono <Enter>
```

This completes the login process. The prompt *guest(n)* appears, where *(n)* is the chronological number of the prompt.

When you are logged in, you may run most of the commands found in Volume 1 of the *UNIX Programmer's Manual* or any of the sample programs supplied with this distribution.

After the *guest(n)* prompt appears, you may use the sample programs. To use the first sample program, type:

```
make sample1 <Enter>
```

When the *guest(n)* prompt reappears (in about one minute), type:

```
sample1 <Enter>
```

The message "Hello world" will appear on the screen, followed by the *guest(n)* prompt.

To use the second sample program, type:

```
make sample2 <Enter>
```

When the *guest(n)* prompt appears (in about one minute), type:

```
sample2 <Enter>
```

Wormlike patterns will appear to move around the screen. When you have seen enough, press <Ctrl>-C to terminate the program. To use the third sample program, type:

```
make sample3 <Enter>
```

When the *guest(n)* prompt appears (in about five minutes), type:

```
sample3 <Enter>
```

When the "$" prompt appears, you are in the Bourne shell, and you may issue commands such as "ls" or "date."

To exit the Bourne shell, press:

<Ctrl>-D

Note that, as distributed, neither the guest account nor the root account has any password security. Therefore, one of the first tasks of the system manager should be to establish user accounts and assign passwords. (See *passwd*(1).)

If you wish to log off the system at this time and do not intend to turn the power off, type:

logout <Enter>

This will return the *login* prompt. If you wish to turn the power off, see the section "Bootstrapping and Shutdown" in the chapter "System Operation," later in this document.

3.1.2.6. Customer Central Support Site Phone Numbers

When you have completed the system installation, and before proceeding to install the system across a network, you should add the telephone number of the designated customer support site to two files:

(1) Edit /etc/motd. (Its source is in /usr/src/etc/motd.) Replace nnn-nnnn with this telephone number, and delete the rest of the file.

(2) Edit /usr/ibm/support. (Its source is in /usr/src/ibm/support.sh.) Replace nnn-nnnn with this telephone number, and delete the rest of the file.

For further support information, see the article "Support Procedures" that accompanies the program directory.

3.1.2.7. Customizing Your System

Your system should now be customized, based on its intended use. During this process, you will be prompted for the type of each workstation on your system and, depending on type, for other information. The following are the workstation types and the additional information required for each:

Machine Type	Host Name	Serial Port Name	Modem Type	Long Distance Dial Digits	uucp Password	Network Device*
Master (m)	yes	no	no	no	no	yes
uucp connection (c)	yes	yes	yes	yes	yes	yes
Both master and connection (b)	yes	yes	yes	yes	yes	yes
Stand-alone or end use (s)	yes	no	no	no	no	yes

* Use un0 for Ethernet and lan0 for Token Ring.

We recommend you put your host names in the /etc/hosts file before you begin tailoring your system. Note also that if you specify a new host name, it will replace your old host name (if it exists) in the /etc/hosts file.

To tailor your system, first become the superuser by issuing the *su* command and entering the superuser password. Then use the command:

/etc/tailor <Enter>

3.1.3. Installing the Andrew File System

If you are installing the IBM Andrew File System, do so now. Follow the installation instructions in the document entitled *The IBM Andrew File System.* When completed, return here.

3.1.4. Installation Procedures for X and Andrew Toolkit

There are three tasks involved in installing X and Andrew Toolkit:

(1) Installing the X Window System

(2) Installing the Andrew Toolkit and application programs

(3) Installing the X and Andrew fonts

The rest of this section describes these tasks.

If you will be installing X on /usr, proceed to Section 3.1.4.1. If you will be installing X on another file system, proceed to Section 3.1.4.1.1.

3.1.4.1. Installing X on /usr

The X Window System is distributed on a single tape which contains both source (20 MB) and binaries (10 MB). We recommend you build the X Window System on a 70 MB file system. The installation involves copying the tape content to a special directory. This procedure will remove and replace all X directories (/usr/include/X11, /usr/lib/libX11.a, /usr/lib/libXtk11.a, /usr/lib/lib, and part of /usr/man/mann). You may wish to save these directories before proceeding.

To install X, proceed as follows:

(1) Become the superuser by issuing the *su* command and supplying the superuser password.

(2) Skip this step if you have done a netinstall. Otherwise, type the following commands:

```
cd /dev  <Enter>
MAKEDEV X   <Enter>
```

If you are installing on the master machine for network installation skip to the next step.

If you are going to run X on an RT PC, the following files must be removed:

```
rm -f /dev/*vga* /dev/*8514*
```

If you are going to run X on a 6152, the following files must be removed:

```
rm -f /dev/*mono* /dev/*aed* /dev/*ap* /dev/*mpel*
```

This is necessary because the RT PC and the 6152 console and mouse devices share minor device numbers. Only one set should exist on a given machine.

(3) Insert the tape labeled "X11" into your streaming tape drive.

(4) To install only binaries, type the following commands:

```
rm -rf /usr/include/X11 /usr/bin/X11
       /usr/lib/libX11.a /usr/lib/libXtk11.a /usr/lib/X11  <Enter>
cd / <Enter>
tar -xfp /dev/st0 usr/include usr/bin usr/lib usr/mann  <Enter>
```

To install only source, type the following commands:

```
rm -rf /usr/src/X11   <Enter>
cd /  <Enter>
tar -xfp /dev/st0 usr/src   <Enter>
```

To install BOTH source and binaries, type the following commands:

```
rm -rf /usr/include/X11 /usr/bin/X11 /usr/lib/libX11.a
      /usr/lib/libXtk11.a /usr/lib/X11 /usr/src/X11   <Enter>
cd /  <Enter>
tar -xfp /dev/st0   <Enter>
```

3.1.4.1.1. Installing X on Another File System

To conserve space or to isolate X from the rest of your system, you can install X on a file system other than /usr. We will use /space as an example. This procedure will remove and replace all X directories (/usr/include/X11, /usr/lib/libX11.a, /usr/lib/libXtk11.a, /usr/lib/X11, and part of /usr/man/mann). You may wish to save these directories before proceeding.

To install X on another file system, proceed as follows:

(1) Skip this step if you have done a netinstall. Become the superuser and type the following commands:

```
cd /dev   <Enter>
MAKEDEV X   <Enter>
```

If you are installing on the master machine for network installation skip to the next step.

If you are going to run X on an RT PC, the following files must be removed:

```
rm -f /dev/*vga* /dev/*8514*
```

If you are going to run X on a 6152, the following files must be removed:

```
rm -f /dev/*mono* /dev/*aed* /dev/*ap* /dev/*mpel*
```

This is necessary because the RT PC and the 6152 console and mouse devices share minor device numbers. Only one set should exist on a given machine.

(2) Become superuser.

(3) Insert the tape labeled "X11" into your streaming tape drive.

(4) To install only binaries, type the following commands:

```
rm -rf /usr/include/X11 /usr/bin/X11
      /usr/lib/libX11.a /usr/lib/libXtk11.a /usr/lib/X11   <Enter>
cd /space  <Enter>
tar -xfp /dev/st0 usr/include usr/bin usr/lib   <Enter>
ln -s /space/usr /include/X11 /usr/include/X11   <Enter>
ln -s /space/usr/bin/X11 /usr/bin/X11   <Enter>
ln -s /space/usr/lib/libX11.a /usr/lib/libX11.a   <Enter>
ln -s /space/usr/lib/libXtk11.a /usr/lib/libXtk11.a   <Enter>
ln -s /space/usr/lib/X11 /usr/lib/X11   <Enter>
```

To install only source, type the following commands:

```
rm -rf /usr/src/X11   <Enter>
cd /space   <Enter>
```

```
tar -xfp /dev/st0 usr/src <Enter>
ln -s /space/usr/src/X11 /usr/src/X11 <Enter>
```

To install BOTH source and binaries, type the following commands:

```
rm -rf /usr/include/X11 /usr/bin/X11
     /usr/lib/libX11.a /usr/lib/libXtk11.a /usr/lib/X11 <Enter>
cd /space <Enter>
tar -xfp /dev/st0 <Enter>
ln -s /space/usr/include/X11 /usr/include/X11 <Enter>
ln -s /space/usr/bin/X11 /usr/bin/X11 <Enter>
ln -s /space/usr/lib/libX11.a /usr/lib/libX11.a <Enter>
ln -s /space/usr/lib/libXtk11.a /usr/lib/libXtk11.a <Enter>
ln -s /space/usr/src/X11 /usr/src/X11 <Enter>
ln -s /space/usr/lib/X11 /usr/lib/X11 <Enter>
```

3.1.4.2. Installing the Andrew Toolkit and Application Programs

The Andrew Toolkit is distributed on a tape and a diskette. The diskette contains the *IBM Andrew Toolkit Programmer's Guide* and a README file describing how to print that document. The tape contains both source (24 MB) and binaries (16 MB). You must have installed X before installing the Andrew Toolkit and applications. Because the instructions in this section replace /usr/andrew, you may wish to save this directory before proceeding.

To install only binaries, proceed as follows:

(1) Become the superuser.

(2) If you have 16 MB of free space in your /usr file system and you wish to locate Andrew there, type the following command:

```
mkdir /usr/andrew <Enter>
```

(3) If you do not have the space in your /usr file system or you would like to locate Andrew on another file system, issue the *mkdir* command on that file system. For example, if you have 16 MB of free space on /space, type the following:

```
mkdir /space/andrew
```

Then make a symbolic link to /space/andrew from /usr/andrew with the following command:

```
ln -s /space/andrew /usr/andrew <Enter>
```

(4) Insert the tape labeled "Andrew Toolkit" in your streaming tape drive and type the following commands:

```
cd / <Enter>
tar -xfp /dev/st0 usr/andrew/X11fonts usr/andrew/bin usr/andrew/dlib
    usr/andrew/doc usr/andrew/etc usr/andrew/fonts usr/andrew/help
    usr/andrew/include usr/andrew/lib usr/andrew/man < Enter>
```

To install BOTH source and binaries requires a 40 MB file system. If you plan to build Andrew, it will require a 70 MB file system. If you do not have a file system with the needed free space, you must build and mount one.

To install source and binaries proceed as follows:

(1) Become the superuser by typing the *su* command and entering the superuser password.

(2) Insert the tape labeled "Andrew Toolkit" in your streaming tape drive and type the following commands:

```
cd /  <Enter>
tar -xfp /dev/st0  < Enter>
```

3.1.4.3. Operating Hints for Andrew

The following errata apply to the operation of Andrew.

3.1.4.3.1. andrew

The "andrew" shell script provided in the /usr/guest/guest/andrew directory assumes that none of the displays attached to your system is currently running either the X window system (xwindows) or the Andrew system (andrew).

3.1.4.3.2. console

Icons are used in "console." When the window containing console is too narrow, some of the information in the icons overlaps. Simply use your mouse to move the border, making the window wider so the icons can be drawn completely.

3.1.4.3.3. ega display

Because the ega display is smaller than the other supported displays in Andrew, the menus that are created with the default fonts do not fit on the ega screen. You can simply specify a smaller font size for the menus by adding these line to the .Xdefaults file:

```
cwm.PaneFont:        andysan8
cmenu.PaneFont:      andysan8
cwm.SelectionFont:   andysan8
cmenu.SelectionFont: andysan8
```

After restarting Andrew, the menus will be smaller and fit on your ega screen.

3.1.4.3.4. keyboard

Not all of the Andrew applications use the same keys in the same way. Read the help files for each application to avoid confusion.

3.1.4.3.5. preview

If you try "preview"but a blank page appears, you may have one of the following problems:

/usr/ibm/troff may have failed

you may have run out of temporary work space in /usr/tmp or /tmp (or both)

3.1.4.3.6. scroll bars

The scroll bars provided with most Andrew windows allow you to move quickly to other parts of the file on which you are working. For fine positioning within items on a screen, use mouse clicks.

3.1.4.3.7. vopcon

The Andrew File System console is called "vopcon." It is furnished only in source form. If you intend to use vopcon to monitor the Andrew File System, you will have to remake the Andrew Toolkit after establishing the correct environment, as described below.

First, install the Andrew File System. Second, compile a new Andrew Toolkit. Do this by typing the following commands:

```
cd /usr/andrew  <Enter>
make VICEDEFINES=-DVICE  <Enter>
make install  <Enter>
make ibmdoc  <Enter>
```

3.1.4.3.8. zip

Icons are used in "zip." When the window containing zip is too narrow, some of the information in the icons overlaps. Simply use your mouse to move the border, making the window wider so the icons can be drawn completely.

3.1.4.4. Operating Hints for X Window System

The following errata apply to the operation of X Window System.

xwindows

The "xwindows" shell script provided in the /usr/guest/guest/xwindows directory assumes that none of the displays attached to your system is currently running either the X window system (xwindows) or the Andrew system (andrew).

3.1.5. Installing IBM/4.3 across a Network

This section describes installing IBM/4.3 on any model IBM RT PC across a network.

3.1.5.1. Before You Start

You should have the following:

- An installation diskette, labeled MINIROOT
- A model 6150 IBM RT PC (with IBM/4.3 installed, both binary and source files) as the source machine
- Any model IBM RT PC as the target machine
- *Either*
  IBM Ethernet Adapters installed in both the source and target machines
  or
  IBM Token-Ring Adapters installed in both the source and target machines

- A network cable connecting the source and target machines

3.1.5.2. Network Installation Overview

To install IBM/4.3 on the IBM RT PC across a network takes about 30 minutes. The procedure requires that a model 6150 IBM RT PC act as a source machine to a target receiving IBM RT PC.

Installing IBM/4.3 on the IBM RT PC across a network involves four major tasks:

(1) Booting IBM/4.3 from the MINIROOT diskette on the target IBM RT PC

(2) Verifying the network connection between the source and the target machines

(3) Restoring the IBM/4.3 file systems from the source machine

(4) Rebooting IBM/4.3 from the target hard disk

Note that the network installation procedure installs only the base system by default. To include other system components, see the section, "Changing Installation Options," later in this chapter.

The next section provides step-by-step instructions for completing these tasks.

3.1.5.3. Network Installation Steps

This section describes how to install IBM/4.3 on any model IBM RT PC using a network connection. The IBM Token-Ring Adapter and the IBM Token Ring Network connection, or the IBM Ethernet Adapter and the two-station Ethernet connection, must be installed as described in the chapter, "Installing Network Adapters," in this article.

If at any point in these installation steps an error should occur, see the section "Restarting After an Error," later in this chapter.

These steps assume that the source IBM RT PC is powered on, and that the files /etc/hosts and ~operator/.rhosts on the source IBM RT PC include appropriate entries for the target system. All actions occur on the target system.

(1) Power on the display, if necessary.

(2) Insert the diskette labeled MINIROOT in the upper diskette drive.

(3) Power on the IBM RT PC.

No pattern will appear on the screen until the system unit is turned on. After about 30 seconds, the LED display on the front of the IBM RT PC shows "29," indicating that the boot program has successfully loaded from diskette.

During the next two minutes, the IBM/4.3 kernel is read into memory and the system is initialized. Information messages appear throughout the installation process. You may ignore all but those mentioned in these instructions.

After initialization is complete, the screen will display instructions. Follow the instructions, which are paraphrased below.

Do **not** remove the MINIROOT diskette.

To correct typing errors, use the backspace key, located in the upper right corner of the typewriter keyboard and marked with a left-facing arrow.

(4) If this is the first time the system has been used, the time and date of the system clock may be incorrect. If so, the system clock will need to be reset before installing IBM/4.3.

Select the menu option to change the system clock. Note that the distributed system uses Pacific Time (Daylight or Standard, as appropriate). If your time zone is different, remember to allow for the difference until you configure a kernel for your time zone. For example, if the local time in New York is 11:05 am on 2 October 1986, one would type "8610020805". (8:05 is the Pacific Time corresponding to 11:05 Eastern Time.)

(5) Select the menu option to proceed with installation. You will be asked to specify tape or network installation. Select the menu option to specify network installation. You will be asked if you wish to change the installation options. See the "Changing Installation Options" section later in this chapter for more information. For normal installation you should not change the installation options.

The network installation program loads, and verifies that the target system can connect to the source system via the network connection. The system displays the following message:

```
> > > establishing network connection from 'slave' to 'master'
```

Once a connection is established, the installation program will begin to restore the root and user file systems (without kernel or documentation source files). This procedure will take approximately 25 minutes. If error messages occur during this procedure, the system may have failed to establish a network connection. See the section, "Restarting after an Error," later in this chapter.

(6) This completes network installation. The target IBM RT PC contains all the binary files that comprise IBM/4.3.

Select the "Halt the system before rebooting from hard disk" menu option.

When the system LEDs go out, remove the MINIROOT diskette. To start the reboot, press:

< Ctrl > - < Alt > - < Pause >

(7) When the boot prompt (:) appears, press **< Enter >**.

IBM/4.3 is loaded from the hard disk and proceeds with normal initialization, including autoboot and fsck. After about five minutes, you will see the multi-user prompt:

```
Academic Operating System 4.3 (master) (console)

login:
```

3.1.5.4. Installation Verification

See Section 3.1.2.5, "Installation Verification," above for instructions.

3.1.5.5. Customizing Your System

Your system should now be customized, based on its intended use as a standalone/end-user machine. Use the command:

```
/etc/tailor < Enter >
```

You will be prompted for the workstation type (s for standalone/end-user machine) and network device (un0 for Ethernet and lan0 for Token Ring).

3.1.6. Restarting after an Error

If you were just beginning to install from tape when the error occurred, be sure that the streaming tape adapter is not in slot 5. That slot on a model 6150 does not support the streaming tape adapter.

If you were in the process of installing from tape when the error occurred, you will be prompted to fix the problem and restart the installation.

Experienced users may wish to avoid repeated installation of the ROOT/USER tape if the error occurred while installing the SOURCE tape. Refer to the section "Notes for Experienced Users" for directions on using *restore.tape*(8R) directly, and type:

restore.tape prompt=root/user hd1g:source <Enter>

3.1.7. Changing Installation Options

Users may choose to modify the default installation options, which are described in this section.

Help This option displays memory-jogging information based on the rest of this section.

Select configuration
This option, which applies only to the 6152, may be set to "full" (the default), "reduced", or "minimal". A "full" configuration has the root and user partitions on the local hard disk. A "reduced" configuration has the root partition on the local hard disk and the Andrew File System is used for the user partition. A "minimal" configuration uses the Remote Virtual Disk (RVD) system for the root partition and the Andrew File System for the user partition.

Select root device
This option specifies which hard disk to use as the root device for IBM/4.3. The default is hd0. For the 6152, hd1 is also supported. For the RT, hd1 and hd2 are supported. Installing IBM/4.3 on and RT's hd1 or hd2 can be useful when it will co-reside with AIX. (Please refer to Appendix A of this article for more information on co-residence.) Note however that using hd2 on an RT for the root device prevents installing the source for utilities.

Change output
This option specifies either verbose or terse output. The default is verbose. Note that terse output results in long periods of time with no activity on the screen.

Select optional system components
In addition to the base IBM/4.3 system, the optional system components selected will be installed. The default is "all" for tape installation and "none" for network installation. One or more of the following may be selected:

Name	Description
kernel	The kernel source in /usr/sys. This is not available on a 6152.
utility	The utility source in /usr/src. This requires an extra disk. If the root partition is hd0, the utility source is installed on hd1g. If root is on hd1, utility source is on hd2g. If root is on hd2 (available only on an RT), utility source cannot be installed.
doc	The documentation source in /usr/doc.
man	The online man pages in /usr/man.
font	The 3812 font libraries in /usr/lib/font.
learn	The *learn*(1) database in /usr/lib/learn.
notes	The *notes*(1) system in in /usr/spool/notes. This is not optional (it is part of the base system) for tape installation.
all	All of the above.
none	None of the above.

NOTE: For tape installation, /sys/support is not optional (it is part of the base system). It is not available for network installation.

The following table of component sizes should help you determine the minimum file system size necessary for the components you wish to install.

		Minimum Sizes		Standard Sizes	
Component	Partition	Tape	Network	40M	70M
base system	hd0a	13166	13166	15884	15884
	hd0g	55953	54859	59857	91476
kernel	hd0g	9627	9545	---	---
utility	hd1g	64514	64514	59857	91476
doc	hd0g	10426	10426	---	---
notes	hd0g	---	1094	---	---

Note that the "notes" component is only optional for network installation. Also, the support tools in /sys/support are not installed by network installation.

Change network preference

This option specifies which network adapter the installation procedure attempts to configure first. The option toggles between Ethernet (the default) and IBM Token Ring. If your system has only one network adapter, this option is irrelevant. This option applies only to network installations.

Select source host

This option specifies the system from which IBM/4.3 is to be downloaded. The value of the option may be a hostname (if the diskette was created with the site's local host table). Otherwise, the value is an internet address. The default is "master." This option applies only to network installations.

Select target host

This option specifies the system onto which IBM/4.3 is to be installed. The value of the option may be a hostname (if the diskette was created with the site's local host table). Otherwise, the value is an internet address. The default is "slave." This option applies only to network installations.

If the host table on the MINIROOT diskette has been updated to include local hostnames, they may be used. Internet addresses may be used regardless of the state of the MINIROOT diskette's host table.

Proceed with installation

Select this option when you are ready to proceed. You will be given an opportunity to verify your work before installation actually proceeds.

Abort installation

This option returns you to the main installation menu, where you can set the date, halt the system, invoke the Bourne shell, or initiate installation.

3.1.8. Notes for Experienced Users

The *restore.tape*(8R) and *restore.net*(8R) commands can also be used directly, by invoking an interactive Bourne shell from the main installation menu. This step replaces the use of the installation menu interface. When finished, leave the Bourne shell by typing **< Ctrl > -D**, and halt the system.

Non-standard file systems may be installed from *dump*(8) format tapes with *restore.tape*, and non-standard file systems may be installed over the network with *restore.net*. The user is referred to the appropriate man pages. Normal dump tapes of the root, user, and source file systems (but *not* the distribution tapes) may be installed with the command

```
restore.tape prompt=root hd0a prompt=user hd0g prompt=source hd1g <Enter>
```

Distribution tapes may be installed with the command

```
restore.tape prompt=root/user hd0a hd0g prompt=source hd1g <Enter>
```

3.1.9. When root is not hd0

You must take special steps to boot IBM/4.3 when the root partition is not */dev/hd0a*. You must explicitly give the kernel name to the boot program; also, you must give the generic (distribution) kernel the location of the root partition at boot time.

For example, assume that we have installed IBM/4.3 on */dev/hd2a*. The following dialogue would take place when booting a generic kernel. System messages are in *italic*; user responses are in **bold**.

```
4.2 BSD UNIX Standalone Boot Program $Revision: X.Y$

Default: hd(0,0)vmunix (just press Enter or wait ~30 seconds)

: hd(2,0)vmunix
. . . (many messages)
root device? hd2
. . . (more messages)
#
```

You will need to preen the disks manually (issue *fsck -p*), and then enter < Ctrl > -D to bring the system up in multiuser mode. Once you have a built a non-GENERIC kernel with hd2 as its root device, you will no longer see the *root device?* question, although you will have to provide the kernel name to boot. For these reasons, you will eventually want to move the root partition to hd0 (after building a kernel to support root on hd0).

3.2. Installation Procedures for the IBM 6152 Academic System

You install IBM/4.3 onto an IBM 6152 across a network from a host IBM RT PC with the operating system installed. However, before you begin, you must install the diskette IBM 6152 Kernel on the IBM RT PC host.

3.2.1. Setting Up the Master RT PC for 6152 Installation

To install the IBM 6152 Kernel on the IBM RT PC, proceed as follows:

(1) Become the super user by issuing the *su* command and supplying the super user password.

(2) Place the IBM 6152 Kernel diskette in the RT PC diskette drive and close the drive door.

(3) Type the following commands:

```
# cd /  <Enter>
# dd if=/dev/rfd0 bs=30b | uncompress | tar xvfp -  <Enter>
```

(4) To create a minimal root filesystem (which is necessary before installing on reduced configurations), type the following commands. Note that **hd1a** is used as an example. You can change this name as necessary for your configuration. However, this partition must be at least 2 MB.

```
# mkdir /minroot.atr  <Enter>
# newfs /dev/hd1a  <Enter>
# mount /dev/hd1a /minroot.atr  <Enter>
# cd /usr/sys/dist_atr  <Enter>
# make.minimal  <Enter>
```

(5) Edit the /etc/fstab file and add the following line:

```
/dev/hd1a:/minroot.atr:rw:0:3
```

(6) Finally, type the following:

```
# exit  <Enter>
```

You may now begin setting up the 6152 workstations.

3.2.2. Hardware Setup

These instructions assume you have set up the IBM 6152 hardware, as described in the manual *IBM 6152 Academic System Setup Guide*, (S68X-2209). After you have set up the system, be sure to run the reference diskette to initialize the configuration and date.

3.2.3. Keyboard Notes

As you install IBM/4.3, keep these keyboard characteristics in mind:

- Use the Backspace key to correct errors, not the cursor control keys.
- These instructions will use the convention <Ctrl> to mean the *Control* key.

3.2.4. System Configurations for the IBM 6152

The three available 6152 configurations are described below:

minimal:

The root partition is a bare-bones root. The real root partition is from RVD. The user partition (and any others) come from the Andrew File System.

reduced:

There is a normal root partition. The user partition (and others) come from the Andrew File System.

full: There are normal root and user partitions, as on an IBM RT installed from the network.

3.2.5. Configuring the Disk Partitions

You must determine the amount of space to allocate to each hard disk partition, using the information contained in this section. The partition sizes are in 512-byte blocks.

The following table lists the number of cylinders available on each type of disk, as well as the number of cylinders that comprise one megabyte (1 MB):

	20 MB	40 MB (type 31)	40 MB (type 32)	70 MB
cylinders available	614	614	1022	70
cylinders/1MB	32	16	26	1

The following table lists the minimum sizes required for the 4.3/6152 partitions:

partition name	minimal system	default 20 MB
boot		
hd0a	6 MB	7 MB
hd0b	8 MB	5 MB
hd0g	---	*

* The rest of the UNIX partition after subtracting the amount used for the a and b partitions and DOS partition.

On a minimal system, we have installed and used IBM/4.3 successfully using the following minidisk entries:

Name	Iodn	Size	Type
boot	32736	68	01
hd0a	32737	20116	00
hd0b	32738	18000	20

On a reduced system, we have installed and used IBM/4.3 successfully using the following minidisk entries:

Name	Iodn	Size	Type
boot	32736	68	01
hd0a	32737	15895	00
hd0b	32738	41390*	20
hd0g	32738	26000*	00

* Note that hd0b can be smaller if you wish to allocate more space to DOS. However, hd0b must be at least 18000. The minimum partition size for DOS is 1712128 bytes (40 or 50 cylinders, depending upon disk type).

On a minimal system, the minidisk partitioning requires a minimal root (hd0a) partition of 6 MB and a minimal swap (hd0b) partition of 8 MB. This leaves 6 MB that can be added to either of the above, or used for DOS. We suggest allocating 9 MB to hd0a and hd0b, and 2 MB to DOS.

3.2.6. Installing DOS on a Hard Disk Partition

Skip this section if the entire disk is to be used for an IBM/4.3 filesystem. This will require that you **always** boot from diskette. To proceed with this section, you must have a current DOS diskette.

(1) Follow the steps in the IBM DOS manual for installing DOS on a fixed disk. Use DOS to format your fixed disk.

(2) When the "A>" prompt returns, remove the DOS diskette and insert the IBM 6152 Academic System installation diskette.

(3) To copy all the files from the installation diskette to the hard disk, type the following command:

copy *.* c: <Enter>

IBM/4.3 will start automatically whenever you boot the system. If this is what you want, skip the next step. If you do not want IBM/4.3 to start automatically (that is, if you wish to remain in DOS), proceed with the next step.

(4) If you wish to remain in DOS whenever you reboot, type the following command at the DOS prompt:

erase c:autoexec.bat <Enter>

This completes the steps necessary to install DOS on a hard disk partition, and prepare to install IBM/4.3. Continue now with the next section to install IBM/4.3.

3.2.7. Installing IBM/4.3 on an IBM 6152

Follow these steps to install IBM/4.3 on an IBM 6152.

(1) Insert the IBM 6152 Academic Operating Systems 4.3 installation diskette in the diskette drive (the upper one, if there are two).

(2) Turn on the power. If it is already on, press <Ctrl>-<Alt>-<Delete>. (Press and hold the <Ctrl> key and the <Alt> key. Then press the <Delete> key.)

(3) When the Standalone Utility Menu appears, respond to the *Choice?* prompt by typing **8 <Enter>** to select the "fdisk" option. **Note:** If you do not respond to the Standalone Utility Menu within 30 seconds, the system will time out and the boot prompt will appear. If this occurs, press <Ctrl>-C to return to the Standalone Utility Menu.

You will now create the hard disk partitions for IBM/4.3.

(4) Reply to the prompt:

Typical disk names are hd(unit,7) where unit = 0 or 1

by typing the following:

hd(0,7) <Enter>

(5) At the > prompt, type:

create <Enter>

This will allocate a disk partition to IBM/4.3.

(6) Respond to the prompt:

Starting Cylinder?

by typing the starting cylinder number for the IBM/4.3 partition. If you created a DOS partition in the preceding section, the starting cylinder number will be the same as the length of the DOS partition. Otherwise, the starting cylinder will be 0.

(7) Reply to the prompt:

(nnn maximum) Partition Length?

by typing the length of this partition.

The partition length will be the maximum partition length displayed in the prompt.

(8) At the > prompt, type **quit** **<Enter>** to return to the main menu.

(9) At the *Choice?* prompt, type **2** **<Enter>** to select the option to format this new partition.

(10) Respond to the prompt:

Device to format?

by typing:

hd(0,2) **<Enter>**

(11) When prompted to verify the format, type **yes** **<Enter>**.

(12) Press **<Enter>** to select the default *severe burnin* format pattern.

The default (severe burnin) pattern is more thorough, but takes more time. If you cannot afford the time for severe burnin, we recommend the zero pattern.

(13) When asked whether to change the default parameters, type **no** **<Enter>**.

(14) When asked whether to preserve data already on the disk, type either **yes** **<Enter>** (if you wish to preserve data on the disk) or **no** **<Enter>** (if you don't).

(15) When asked if the format process should start, type **yes** **<Enter>**.

For severe burnin, the system will take several hours to complete the format process.

(16) When the format is finished, press **<Enter>** to return to the main menu.

(17) At the *Choice?* prompt, type **10** **<Enter>** to select the "minidisk" option, which creates IBM/4.3 minidisk partitions.

(18) When asked which disk to use, type **hd(0,2)** **<Enter>**.

During this step, you will be prompted for partition sizes. To allow adequate swap space on disks smaller than 40 M, select partition sizes by hand; do not use the *standard* command.

(19) At the > prompt, type **init** **<Enter>**.

This initializes the minidisk table. The init command displays a warning message.

(20) Reply to the warning message:

Confirm (all data will be lost) [y/n]

by typing y to indicate you do want to proceed.

(21) You may use the command *standard* to create default hd0a, hd0b, and hd0g partitions, or you can create partitions manually.

For each minidisk entry you wish to create manually, use the *create* command. The system will prompt you for the *name*, *iodn*, *size*, and *type* of the minidisk entry.

Use the following information to develop the correct values for your system.

The *iodn* should start at 32736 for the first partition. Each succeeding *iodn* should be incremented by one.

To develop the *size* of each minidisk partition, you must first determine how much unallocated space remains. Use the *list* command to find the size of the pseudo-partition named "FREE." This is the total space available for you to divide among all the minidisk partitions on that disk.

The values for *type* are as follows. The boot partition is type 01, the hd0b (swap) partition is type 20, and all others are type 00.

(22) Type **quit** <**Enter** to return to the main menu.

(23) Reply to the *Choice?* prompt, by typing **9** <**Enter**> to select the "install" option. This copies the MINIROOT onto the hd0b (swap) partition and boots it.

(24) When the prompt to insert the installation MINIROOT diskette appears, simply press <**Enter**>. (The diskette is already in the drive.)

(25) When the boot prompt (:) appears, either press <**Enter**>, or simply wait about 30 seconds.

At this point the standard installation miniroot menu appears. Go to the section entitled "Installing IBM 4.3 across a Network" earlier in this chapter. The instructions apply to both and IBM RT PC and an IBM 6152.

When you have completed the steps in that section, you will have successfully installed IBM/4.3 on your IBM 6152.

4. SYSTEM SETUP

This chapter describes procedures used to set up a IBM/4.3 system. Use these procedures after you first install your system or when your system configuration changes. Procedures for normal system operation are described in Chapter 6.

4.1. Kernel Configuration

This section briefly describes the layout of the kernel code and how files for devices are made. For a full discussion of configuring and building system images, consult the article "Building IBM/4.3 Systems with Config" in the manual.

4.1.1. Kernel Organization

As distributed, the kernel source is in /usr/sys. The source may be physically located anywhere within any file system as long as a symbolic link to the location is created for the file /sys. (Many files in /usr/include are normally symbolic links relative to /sys.) In further discussions of the system source, all path names will be given relative to /sys.

The directory /sys/sys contains the mainline machine-independent operating system code. Files within this directory are conventionally named with the following prefixes:

init_	system initialization
kern_	kernel (authentication, process management, etc.)
quota_	disk quotas
sys_	system calls and similar
tty_	terminal handling
ufs_	file system
uipc_	interprocess communication
vm_	virtual memory

The remaining directories are organized as follows.

/sys/h	machine independent include files
/sys/conf	site configuration files and basic templates
/sys/net	network independent, but network related code
/sys/netinet	DARPA Internet code
/sys/netimp	IMP support code
/sys/netns	Xerox NS support code
/sys/ca	RT specific mainline code
/sys/caif	RT network interface code
/sys/caio	RT device drivers and related code
/sys/cacons	RT console device drivers and related code

Many of these directories are referenced through /usr/include with symbolic links. For example, /usr/include/sys is a symbolic link to /sys/h. The system code, as distributed, is totally independent of the include files in /usr/include. This allows the system to be recompiled from scratch without the /usr file system mounted.

4.1.2. Devices and Device Drivers

Devices supported by IBM/4.3 are implemented in the kernel by drivers whose source is kept in /sys/ca, /sys/caio, and /sys/cacons. These drivers are loaded into the system when included in a cpu-specific configuration file kept in the conf directory. Devices are accessed through special files in the file system, made by the *mknod*(8) program, and normally kept in the /dev directory. For devices supported by the distribution system, files are created in /dev by the /dev/MAKEDEV shell script.

Determine the set of devices that you have and create a new /dev directory by running the MAKEDEV script. First create a new directory /newdev; copy MAKEDEV into it; edit the file MAKEDEV.local to provide an entry for local needs; run it to generate a /newdev directory. For instance, if your machine has one hard disk and a diskette, you would type the following:

```
#cd <Enter>
#mkdir newdev <Enter>
#cp dev/MAKEDEV newdev/MAKEDEV <Enter>
#cd newdev <Enter>
#MAKEDEV hd0 fd0 std local <Enter>
```

Note the "std" argument causes standard devices such as /dev/console (the machine console) to be created.

You can then type:

```
#cd <Enter>
#mv dev olddev ; mv newdev dev <Enter>
#sync <Enter>
```

to install the new device directory.

4.1.3. Building New System Images

The kernel configuration of each IBM/4.3 system is described by a single configuration file, stored in the /sys/conf directory. To learn about the format of this file and the procedure used to build system images, you should:

- Read "Building IBM/4.3 Systems with Config" in *IBM Academic Operating System 4.3.*
- Study the manual pages for the devices you have (See *IBM Academic Operating System 4.3* or the *UNIX Programmer's Reference Manual*).
- Review the sample configuration file in the /sys/conf directory.

The configured system image "vmunix"[2] should be copied to the root and then booted to try it out. It is best to name it /newvmunix so as not to destroy the working system until you are sure it does work:

```
# cp vmunix /newvmunix <Enter>
# sync <Enter>
```

It is also a good idea to save the old system under some other name. In particular, we recommend that you save the generic distribution version of the system permanently as /genvmunix for use in emergencies.

To boot the new version of the system, power on the IBM RT PC. If it's already on, you can perform a hardware boot by using the *reboot*(8) command. Alternatively, you can use the *sync*(1) command, and then press and hold down the following keys:

<Ctrl>-<Alt>-<Pause>

After having booted and tested the new system, it should be installed as /vmunix before going into multiuser operation. A systematic scheme for numbering and saving old

[2] A system configured with the debugger is called "vmunix.ws".

versions of the system may be useful.

4.2. Disk Configuration

This section describes how to lay out file systems to make use of the available space and to balance disk load for improved system performance.

4.2.1. Initializing /etc/fstab

Change into the directory /etc and copy the appropriate file from:

fstab.hd.1	(for a one-disk, desk model system)
fstab.hd.3	(for a two- or three-disk, floor model system)

to the file /etc/fstab, i.e.:

```
# cd /etc  <Enter>
# cp fstab.hd.x fstab  <Enter>
```

where *x* is either 1 or 3.

This will set up the initial information about the usage of disk partitions. Note that the installation procedures attempt to do this automatically during installation.

4.2.2. Disk Naming and Divisions

Each physical disk drive can be divided into up to eight partitions; IBM/4.3 typically uses only three or four partitions. The first partition (hd0a) stores a root file system from which IBM/4.3 can be bootstrapped. The second partition (hd0b) is used for paging and swapping. The third partition (hd0g) is used for the /usr file system.

The disk partition sizes for a drive are based on a set of four default partition tables. See *diskpart*(8). The particular table used depends on the size of the drive. The "a" partition is the same size across all drives, 15884 sectors. The "b" partition is 33440 sectors on 70-megabyte disks, and 10032 sectors on 40-megabyte disks. The "c" partition is large enough to access the entire disk, including the space at the front of the disk reserved for the bad sector forwarding table, and the space at the end of the disk containing the pool of replacement sectors.

Non-standard partition sizes may be specified on a per disk basis (see *minidisk*(8R)).

4.2.3. Space Available

The space available (in sectors) in the default disk partitions is listed in the following table.

Name	40MB	70MB
hd?a	15884	15884
hd?b	10032	33440
hd?c	87040*	141372*
hd?d	15884	15884
hd?e	-----**	55936
hd?f	43826	19404
hd?g	59721	91476

* Note that a file system on the "c" partition can only be 138040 blocks on a 70-megabyte disk, or 86275 blocks on a 40-megabyte disk, to allow for the reserved space at the end of the disk.

** Partition "e" is not available on a 40-megabyte disk.

Be aware that the disks sizes are measured in disk sectors (512 bytes), while the IBM/4.3 file system blocks are variably sized. User programs report disk space in kilobytes, and disk sizes in sectors. The /etc/disktab file used in making file systems specifies disk partition sizes in sectors. The default sector size of 512 bytes may be overridden with the "se" attribute. Note that the only sector size currently supported is DEV_BSIZE as defined in /sys/h/param.h.

4.2.4. Layout Considerations

There are several considerations in deciding how to arrange your disks. Two major considerations are adequate space and adequate throughput. Paging space is an important parameter. The system, as distributed, sizes the configured paging areas each time the system is booted. Further, multiple paging areas of different size may be interleaved. Drives smaller than 400 megabytes have swap partitions of 16 megabytes, while drives larger than 400 megabytes have 32 megabytes. These values may be changed to get more paging space by changing the appropriate minidisk table on the disk.

Many common system programs (C, the editor, the assembler, etc.) create intermediate files in the /tmp directory, so the file system where /tmp is stored should be large enough to accommodate most high-water marks. If you have several disks, mount /tmp in a root (i.e. first partition) file system on another disk. All programs that create files in /tmp also delete them but may leave dregs. Examine the directory periodically and delete old files.

On a single-disk system, there may not be sufficient free space on the root file system for /tmp. You can replace /tmp with a symbolic link to /usr/tmp on the hd0g partition, which should have sufficient space.

The efficiency with which IBM/4.3 is able to use the CPU is often strongly affected by the configuration of disk controllers. For general time-sharing applications, the best strategy is to try to split the root file system (/), system binaries (/usr), the temporary files (/tmp), and the user files among several disk arms, and to interleave the paging activity among several arms.

It is critical for good performance to balance disk load. There are at least five components of disk load that you can divide between available disks:

1. The root (/) file system.
2. The /tmp file system.
3. The /usr file system.
4. The user files.
5. The paging activity.

The following possibilities are ones Berkeley has used when they have had two or three disks:

	disks	
what	2	3
/	0	0
tmp	1	2
usr	1	1
paging	0+1	0+2
users	0	0+2

You should try to even out the disk load as much as possible by locating on separate arms those file systems between which heavy copying occurs. Note that long-term balancing of the load is not important; it is much more important to balance the load properly for when the system is busy.

Intelligent experimentation with a few file system arrangements can pay off in much improved performance. It is particularly easy to move the root, the /tmp file system and the paging areas. Place the user files and the /usr directory as space dictates, and experiment with the other, more easily-moved file systems.

4.2.5. File System Parameters

Each file system has associated parameters describing its block size, fragment size, and the geometry characteristics of the disk on which it resides. Inaccurate specification of the disk characteristics or haphazard choice of the file system parameters can cause substantial throughput degradation or significant wasted disk space. As distributed, file systems are configured according to the following table.

File system	Block size	Fragment size
/	8 Kbytes	1 Kbyte
usr	4 Kbytes	1 Kbyte
users	4 Kbytes	1 Kbyte

The root file system block size is made large to optimize bandwidth to the associated disk. This large block size is important because the /tmp directory is normally part of the root file or a similar file system. The large block size is also important because many of the most heavily used programs are demand-paged out of the /bin directory. The fragment size of 1 Kbyte is a "nominal" value to use with a file system. With a 1-Kbyte fragment size, disk space utilization is approximately the same as with earlier versions of the file system.

The /usr file system uses a 4-Kbyte block size with 1-Kbyte fragment size to achieve high performance.

File systems for users have a 4-Kbyte block size with 1-Kbyte fragment size. These parameters have been selected based on the performance of Berkeley's user file systems. The 4-Kbyte block size provides adequate bandwidth while the 1-Kbyte fragment size provides acceptable space conservation and disk fragmentation.

You may choose other parameters in constructing file systems, but the factors involved in block size and fragment size are many and interact in complex ways. Larger block sizes result in better throughput to large files in the file system, because larger I/O requests can be performed. However, you should consider the average file sizes found in a file system and the performance of the internal system buffer cache. The system provides space in the inode for 12 direct block pointers, one single indirect block pointer, and one double indirect block pointer.[3] If a file uses only direct blocks, you can optimize access time to it by maximizing the block size. If a file spills over into an indirect block, increasing the block size of the file system may decrease the amount of space used (by eliminating the need to allocate an indirect block). However, if you increase the block size, and an indirect block is still required, the file will use more disk space (because indirect blocks are allocated according to the block size of the file system).

[3] A triple indirect block pointer is also reserved, but not supported.

In selecting a fragment size for a file system, you must consider at least two things. The major performance tradeoffs are between an 8-Kbyte block file system and a 4-Kbyte block file system. Because of implementation constraints, the ratio of block size to fragment size cannot be greater than 8. An 8-Kbyte file system will always have a fragment size of at least 1 Kbyte. If a file system is created with a 4-Kbyte block size and a 1-Kbyte fragment size, and then upgraded to an 8-Kbyte block size and 1-Kbyte fragment size, identical space conservation occurs. However, if a file system has a 4-Kbyte block size and 512-byte fragment size, converting it to an 8K/1K file system causes significantly more space to be used. A 4-Kbyte block file system which might be upgraded to 8-Kbyte blocks for higher performance should use fragment sizes of at least 1 Kbyte to minimize the amount of work required in conversion.

A second, more important, consideration when selecting the fragment size for a file system is the level of fragmentation on the disk. With a 512-byte fragment size, storage fragmentation occurs much sooner, particularly with a busy file system running near full capacity. By comparison, the level of fragmentation in a 1-Kbyte fragment file system is much less severe. On file systems where many files are created and deleted, the 512-byte fragment size is more likely to result in apparent space exhaustion because of fragmentation. That is, when the file system is nearly full, file expansion that requires locating a contiguous area of disk space is more likely to fail on a 512-byte file system than on a 1-Kbyte file system. To minimize fragmentation problems of this sort, a parameter in the super block specifies a minimum acceptable free space threshold. When anyone but the super-user attempts to allocate disk space and the free space threshold is exceeded, the user is returned an error as if the file system were actually full. This parameter is nominally set to 10%, and can be changed by supplying a parameter to *newfs*, or by updating the super block of an existing file system using *tunefs*(8).

In general, unless a file system is to be used for a special purpose application (for example, storing image processing data), Berkeley recommends using the default values supplied. Remember that the current implementation limits the block size to at most 8 Kbytes and the ratio of block size to fragment size must be 1, 2, 4, or 8.

The disk geometry information used by the file system affects the block layout policies employed. The file /etc/disktab, as supplied, contains the data for drives supported by the system. When constructing a file system you should use the *newfs*(8) program and specify the type of disk on which the file system resides. This file also contains the default file system partition sizes, and default block and fragment sizes. To override any of the default values you can modify the file or use an option of *newfs*.

4.2.6. Implementing a Layout

To put a chosen disk layout into effect, use *newfs*(8) to create each new file system, and add its name to the file /etc/fstab. The system will check and mount each file system found in /etc/fstab when the system is bootstrapped.

Consider a system with 70-megabyte drives. On the first drive, (hd0), we put the root file system in hd0a and the /usr file system in hd0g. The /tmp directory is part of the root file system because no file system is mounted on /tmp. On a one-drive model, we put user files in the hd0g partition with the system binaries.

On a three-drive model, we create a file system in hd1g and put user files there, calling the file system /mnt. We interleave the paging between the first and second drives. To do this, we build a system configuration that specifies:

```
config   vmunix root on hd0      swap on hd0 and hd1
```

to get the swap interleaved. /etc/fstab would then contain:

```
/dev/hd0a:/:rw:1:1
/dev/hd0b::sw::
/dev/hd0g:/usr:rw:1:2
/dev/hd1b::sw::
/dev/hd1g:/mnt:rw:1:2
```

We keep a backup copy of the root file system in the hd1a disk partition.

To make the /mnt file system, we use the following commands:

```
#cd /dev <Enter>
#MAKEDEV hd1 <Enter>
#newfs hd1g <Enter>
```

After information about the file system prints, we type the following commands:

```
#mkdir /mnt <Enter>
#mount /dev/hd1g /mnt <Enter>
```

4.3. Setup for Remote Virtual Disks (RVD)

For information regarding installing and setting up remote virtual disks, see the "The Remote Virtual Disk System" article.

4.4. Configuring Terminals

If IBM/4.3 is to support simultaneous access from multiple terminals, you must edit the file /etc/ttys. (See *ttys*(5).)

Terminals connected to the system via RS232 ports are conventionally named **ttyxx** where xx identifies the specific line. The lines on port RS232 cards are named /dev/tty00, /dev/tty01, . . . , /dev/tty15 (up to four cards may be installed). The planar serial ports are known as /dev/ttys0 and /dev/ttys1. The asynchronous communications cards are known as /dev/ttyc0 and /dev/ttyc1.

To add a new terminal device, be sure the device is configured into the system and that the special files for the device have been made by /dev/MAKEDEV. Then, enable the appropriate lines of /etc/ttys by setting the "status" field to **on** (or add new lines to /etc/ttys). Note that lines in /etc/ttys are one-for-one with entries in the file of current users (/etc/utmp), and therefore it is best to make changes while running in single-user mode and to add all of the entries for a new device at once.

The format of the /etc/ttys file is completely new in IBM/4.3. Each line in the file is broken into four tab separated fields (comments are shown by a '#' character and extend to the end of the line). For each terminal line the four fields are: the device (without a leading /dev), the program /etc/init should startup to service the line (or **none** if the line is to be left alone), the terminal type (found in /etc/termcap), and optional status information describing if the terminal is enabled or not and if it is "secure" (i.e. the super user should be allowed to login on the line). All fields are character strings with entries requiring embedded white space enclosed in double quotes. Thus a newly added terminal /dev/tty00 could be added as

```
tty00    "/etc/getty std.9600"    apa16   on secure        # mike's office
```

The std.9600 parameter provided to /etc/getty is used in searching the file /etc/gettytab; it specifies a terminal's characteristics (such as baud rate). To make custom terminal types, consult *gettytab*(5) before modifying /etc/gettytab.

Dialup terminals should be wired so that carrier is asserted only when the phone line is dialed up. For non-dialup terminals from which modem control is not available, you must either wire back the signals so that the carrier appears to always be present, or show in the system configuration that carrier is to be assumed to be present with *flags* for each terminal device. See *asy*(4) and *psp*(4) for details.

For network terminals (i.e. pseudo terminals), no program should be started up on the lines. Thus, the normal entry in /etc/ttys would look like

```
ttyp0   none   network
```

(Note the fourth field is not needed for these entries.)

When the system is running multi-user, all terminals that are listed in /etc/ttys as **on** have their line are enabled. If, during normal operations, it is desired to disable a terminal line, you can edit the file /etc/ttys to change the terminal's status to **off** and then send a hangup signal to the *init* process, using the following command:

```
# kill −1 1 <Enter>
```

Terminals can similarly be enabled by changing the status field from **off** to **on** and sending a hangup signal to *init*.

Note that if a special file is inaccessible when *init* tries to create a process for it, *init* will log a message to the system error logging process (/etc/syslogd) and try to reopen the terminal every minute, reprinting the warning message every 10 minutes. Messages of this sort are normally printed on the console, though other actions may occur depending on the configuration information found in /etc/syslog.conf.

Finally note that you should change the names of any dialup terminals to ttyd? where ? is in [0-9a-zA-Z], as some programs use this property of the names to determine if a terminal is a dialup. Shell commands to do this should be put in the /dev/MAKEDEV.local script.

While it is possible to use truly arbitrary strings for terminal names, the *ps*(1) command makes good use of the convention that tty names (by default, and also after dialups are named as suggested above) are distinct in the last 2 characters. We don't recommend you change this; the heuristics *ps*(1) uses that are based on these conventions may break down and *ps* will run MUCH slower.

4.5. Adding Users

You can add new users to the system by adding a line to the password file /etc/passwd. The procedure for adding a new user is described in *adduser*(8). You should add accounts for the initial user community, give each a directory and a password, and put users who wish to share software in the same group.

4.6. Site Tailoring

All programs that require the site name or some similar characteristic obtain the information through system calls or from files located in /etc. To supply a site name, edit the file /etc/rc.config. A line in this file,

hostname = *hostname*

defines the value returned by the *gethostname* system call. If you are running the name server, your site name should be your fully qualified domain name. Programs such as *getty*(8), *mail*(1), *wall*(1), and *who*(1) use this system call so that the binary images are site-independent. IBM/4.3 provides the *tailor*(8) command to facilitate site tailoring.

4.7. Setting Up the Line Printer System

The line printer system consists of at least the following files and commands:

/usr/ucb/lpq	spooling queue examination program
/usr/ucb/lprm	program to delete jobs from a queue
/usr/ucb/lpr	program to enter a job in a printer queue
/etc/printcap	printer configuration and capability data base
/usr/lib/lpd	line printer daemon, scans spooling queues
/etc/lpc	line printer control program
/etc/hosts.lpd	list of hosts allowed to use the printers

The file /etc/printcap is a master data base describing both line printers directly attached to a machine and printers accessible across a network. The manual page *printcap*(5) describes the format of this data base and shows the default values for such things as the directory in which spooling is performed. The line printer system handles multiple printers, multiple spooling queues, local and remote printers, and printers attached via serial lines that require line initialization such as the baud rate.

Remote spooling via the network is handled with two spooling queues, one on the local machine and one on the remote machine. When a remote printer job is initiated with *lpr*, the job is queued locally and a daemon process is created to oversee the transfer of the job to the remote machine. If the destination machine is unreachable, the job will remain queued until it is possible to transfer the files to the spooling queue on the remote machine. The *lpq* program shows the contents of spool queues on both the local and remote machines.

To configure your line printers, consult the *printcap*(5) man page and the article entitled "4.3BSD Line Printer Spooler Manual" in the *UNIX System Manager's Manual.* Include a call to *lpd*(8) in /etc/rc. (See also *ibm38l2*(8) and *ppt*(8).)

4.8. Setting Up the Mail System

The mail system consists of the following commands:

/bin/mail	old standard mail program, *binmail*(1)
/usr/ucb/mail	UCB mail program, described in *mail*(1)
/usr/lib/sendmail	mail routing program
/usr/spool/mail	mail spooling directory
/usr/spool/secretmail	secure mail directory
/usr/bin/xsend	secure mail sender
/usr/bin/xget	secure mail receiver
/usr/lib/aliases	mail forwarding information
/usr/ucb/newaliases	command to rebuild binary forwarding database
/usr/ucb/biff	mail notification enabler
/etc/comsat	mail notification daemon

Normally, you use the *mail*(1) command to send and receive mail. This command provides a front end to edit the messages sent and received, and passes the messages to *sendmail*(8) for routing. To process each piece of mail, the routing algorithm uses knowledge of the network name syntax, aliasing and forwarding information, and network topology, as defined in the configuration file /usr/lib/sendmail.cf. The program /bin/mail delivers local mail by adding it to the mailboxes in the directory /usr/spool/mail/*username*, using a locking protocol to avoid problems with simultaneous updates. After mail is delivered, the local mail delivery daemon /etc/comsat is notified, which in turn notifies users who have issued a "biff y" command that mail has arrived.

Normally, mail queued in the directory /usr/spool/mail can be read only by the recipient. To send mail that is secure against any possible perusal (except by a code-breaker), you should use the secret mail facility, which encrypts the mail so that no one can read it.

To set up the mail facility, read the instructions in the file READ_ME in the directory /usr/src/usr.lib/sendmail. Then adjust the necessary configuration files. You should also set up the file /usr/lib/aliases for your installation, creating mail groups as appropriate. Documents describing *sendmail*'s operation and installation are also included in the distribution.

4.8.1. Setting Up a Uucp Connection

The version of *uucp* included in IBM/4.3 is an enhanced version of that originally distributed with 32/V.[4] The enhancements include:

- support for many auto call units
- breakup of the spooling area into multiple subdirectories
- addition of an *L.cmds* file to control the set of commands that may be executed by a remote site
- enhanced "expect-send" sequence capabilities when logging in to a remote site
- new commands used to poll sites and obtain snapshots of *uucp* activity
- additional protocols for different communication media

This section gives a brief overview of *uucp* and points out the most important steps in its installation.

To connect two IBM/4.3 machines with a *uucp* network link using modems, one site must have an automatic call unit and the other must have a dialup port. It is best if both sites have both.

You should first read the article "Uucp Implementation Description" in the *UNIX System Manager's Manual.* It describes in detail the file formats and conventions, and will give you a little context. In addition, the document setup.tblms, located in the directory /usr/src/usr.bin/uucp/UUAIDS, may be of use in tailoring the software to your needs.

The *uucp* support is located in three major directories: /usr/bin, /usr/lib/uucp, and /usr/spool/uucp. User commands are kept in /usr/bin; operational commands are in /usr/lib/uucp; and /usr/spool/uucp is used as a spooling area. The commands in /usr/bin are:

/usr/bin/uucp	file copy command
/usr/bin/uux	remote execution command
/usr/bin/uusend	binary file transfer using mail
/usr/bin/uuencode	binary file encoder (for *uusend*)
/usr/bin/uudecode	binary file decoder (for *uusend*)
/usr/bin/uulog	scans session log files
/usr/bin/uusnap	gives a snapshot of *uucp* activity
/usr/bin/uupoll	polls remote system until an answer is received
/usr/bin/uuname	prints a list of known uucp hosts
/usr/bin/uuq	gives information about the queue

[4] The *uucp* included in this distribution is the result of work by many people; we gratefully acknowledge their contributions, but refrain from mentioning names in the interest of keeping this document current.

The important files and commands in /usr/lib/uucp are:

/usr/lib/uucp/L-devices	list of dialers and hardwired lines
/usr/lib/uucp/L-dialcodes	dialcode abbreviations
/usr/lib/uucp/L.aliases	hostname aliases
/usr/lib/uucp/L.cmds	commands remote sites may execute
/usr/lib/uucp/L.sys	systems to communicate with, how to connect, and when
/usr/lib/uucp/SEQF	sequence numbering control file
/usr/lib/uucp/USERFILE	remote site pathname access specifications
/usr/lib/uucp/uucico	*uucp* protocol daemon
/usr/lib/uucp/uuclean	cleans up garbage files in spool area
/usr/lib/uucp/uuxqt	*uucp* remote execution server

while the spooling area contains the following important files and directories:

/usr/spool/uucp/C.	directory for command (C.) files
/usr/spool/uucp/D.	directory for data (D.) files
/usr/spool/uucp/X.	directory for command execution (X.) files
/usr/spool/uucp/D.*machine*	directory for local D. files
/usr/spool/uucp/D.*machine*X	directory for local X. files
/usr/spool/uucp/TM.	directory for temporary (TM.) files
/usr/spool/uucp/LOGFILE	log file of *uucp* activity
/usr/spool/uucp/SYSLOG	log file of *uucp* file transfers

To install *uucp* on your system, start by selecting a site name (shorter than 14 characters). Next, create a *uucp* account in the /etc/passwd file and set up a password. Then, create the appropriate spooling directories with mode 755 and owned by user *uucp*, group *daemon*.

If you have an auto-call unit, create the L.sys, L-dialcodes, and L-devices files. The L.sys file should contain the phone numbers and login sequences required to establish a connection with a *uucp* daemon on another machine. For example, the L.sys file looks something like:

```
ibmsupt Any ACU 1200 out0123456789- ogin-EOT-ogin uucp
cbosg Never Slave 300
cbosgd Never Slave 300
chico Never Slave 1200 out2010123456
```

The first field is the name of a site; the second tells when the machine may be called; the third specifies how the host is connected (through an ACU, a hardwired line, etc.); the fourth is the baud rate; the fifth is the phone number to use in connecting through an auto-call unit; and the sixth is a login sequence. The phone number may contain common abbreviations that are defined in the L-dialcodes file. The device specification should refer to devices found in the L-devices file. Using only ACU causes the *uucp* daemon, *uucico*, to search for any available auto-call unit in L-devices. Berkeley's L-dialcodes file is of the form:

```
ucb 2
out 9%
```

while their L-devices file is:

```
ACU tty00 unused 1200 ventel
```

Refer to the README file in the *uucp* source directory for more information about installation.

As *uucp* operates, it creates (and removes) many small files in the directories underneath /usr/spool/uucp. Sometimes files are left undeleted; purge them with the *uuclean* program. The log files can grow without bound unless trimmed back; use *uulog* to maintain these files. Many useful aids in maintaining your *uucp* installation are included in a subdirectory UUAIDS beneath /usr/src/usr.bin/uucp. Peruse this directory, and read the "setup" instructions also located there.

The *tailor*(8) command may be used to set up the necessary uucp entries for connecting a uucp-connection work station to the IBM support network.

5. NETWORK SETUP

IBM/4.3 provides support for the DARPA standard Internet protocols IP, ICMP, TCP, and UDP. These protocols may be used on top of a variety of hardware devices ranging from the IMP's (PSN's) used in the ARPANET to local area network controllers for the Ethernet or IBM Token-Ring local area network. Network services are split between the kernel (communication protocols) and user programs (user services such as TELNET and FTP). This section describes how to configure your system to use the Internet networking support. IBM/4.3 also supports the Xerox Network Systems (NS) protocols. IDP and SPP are implemented in the kernel, and other protocols such as Courier run at the user level.

5.1. System Configuration

To configure the kernel to include the Internet communication protocols, define the INET option. Xerox NS support is enabled with the NS option. In either case, include the pseudo-devices "pty", and "loop" in your machine's configuration file. The "pty" pseudo-device forces the pseudo terminal device driver to be configured into the system, see *pty*(4), while the "loop" pseudo-device forces inclusion of the software loopback interface driver. The loop driver is used in network testing and also by the error logging system.

If you are planning to use the Internet network facilities on a 10 MB/s Ethernet or on the IBM Token-Ring local area network, the pseudo-device "ether" should also be included in the configuration; this forces inclusion of the Address Resolution Protocol module used in mapping between 48-bit Ethernet and 32-bit Internet addresses. Also, if you have an IMP connection, you will need to include the pseudo-device "imp."

Before configuring the appropriate networking hardware, you should consult the manual pages in section 4 of the *UNIX Programmer's Manual.* Software support exists for the device "un," the IBM Baseband Adapter for use with the Ethernet network, and for the device "lan," the IBM Token-Ring local area network interface.

All network interface drivers including the loopback interface, require that their host address(es) be defined at boot time. This is done with *ifconfig*(8C) commands included in the /etc/rc.local file. Interfaces that are able to dynamically deduce the host part of an address may check that the host part of the address is correct. The manual page for each network interface describes the method used to establish a host's address. *Ifconfig*(8) can also be used to set options for the interface at boot time. Options are set independently for each interface, and apply to all packets sent using that interface. These options include disabling the use of the Address Resolution Protocol; this may be useful if a network is shared with hosts running software that does not yet provide this function. Alternatively, translations for such hosts may be set in advance or "published" by a IBM/4.3 host by use of the *arp*(8c) command. Note that the use of trailer link-level is now negotiated between IBM/4.3 hosts using ARP, and it is thus no longer necessary to disable the use of trailers with *ifconfig*.

To use the pseudo terminals just configured, device entries must be created in the /dev directory. To create 32 pseudo terminals (plenty, unless you have a heavy network load) execute the following commands.

```
# cd /dev <Enter>
# MAKEDEV pty0 pty1 <Enter>
```

More pseudo terminals may be made by specifying *pty2*, *pty3*, etc. The kernel normally includes support for 32 pseudo terminals unless the configuration file specifies a different number. Each pseudo terminal really consists of two files in /dev: a master and a slave. The master pseudo terminal file is named /dev/ptyp?, while the slave side is /dev/ttyp?.

Pseudo terminals are also used by several programs not related to the network. In addition to creating the pseudo terminals, be sure to install them in the /etc/ttys file (with a 'none' in the second column so no *getty* is started).

5.2. Local Subnetworks

In IBM/4.3 the DARPA Internet support includes the notion of "subnetworks." This is a mechanism by which multiple local networks may appear as a single Internet network to off-site hosts. Subnetworks are useful because they allow a site to hide their local topology, requiring only a single route in external gateways; it also means that local network numbers may be locally administered. The standard describing this change in Internet addressing is RFC-950.

To set up local subnetworks one must first decide how the available address space (the Internet "host part" of the 32-bit address) is to be partitioned. Sites with a class A network number have a 24-bit address space with which to work, sites with a class B network number have a 16-bit address space, while sites with a class C network number have an 8-bit address space[5]. To define local subnets you must steal some bits from the local host address space for use in extending the network portion of the Internet address. This reinterpretation of Internet addresses is done only for local networks; i.e. it is not visible to hosts off-site. For example, if your site has a class B network number, hosts on this network have an Internet address that contains the network number, 16 bits, and the host number, another 16 bits. To define 254 local subnets, each possessing at most 255 hosts, 8 bits may be taken from the local part. (The use of subnets 0 and all-1's, 255 in this example, is discouraged to avoid confusion about broadcast addresses.) These new network numbers are then constructed by concatenating the original 16-bit network number with the extra 8 bits containing the local subnetwork number.

The existence of local subnetworks is communicated to the system at the time a network interface is configured with the *netmask* option to the *ifconfig* program. A "network mask" is specified to define the portion of the Internet address that is to be considered the network part for that network. This mask normally contains the bits corresponding to the standard network part as well as the portion of the local part that has been assigned to subnets. If no mask is specified when the address is set, it will be set according to the class of the network. For example, **at Berkeley** (class B network 128.32) 8 bits of the local part have been reserved for defining subnetworks; consequently the /etc/rc.local file contains lines of the form

```
/etc/ifconfig en0 netmask 0xffffff00 128.32.1.7
```

This specifies that for interface "en0," the upper 24 bits of the Internet address should be used in calculating network numbers (netmask 0xffffff00), and the interface's Internet address is "128.32.1.7" (host 7 on network 128.32.1). Hosts *m* on sub-network *n* of this network would then have addresses of the form "128.32.*n.m*;" for example, host 99 on network 129 would have an address "128.32.129.99." For hosts with multiple interfaces, the network mask should be set for each interface, although in practice only the mask of the first interface on each network is actually used.

5.3. Internet Broadcast Addresses

The address defined as the broadcast address for Internet networks according to RFC-919 is the address with a host part of all 1's. The address used by 4.2/RT was the address with

[5] If you are unfamiliar with the Internet addressing structure, consult "Address Mappings", Internet RFC-796, J. Postel; available from the Internet Network Information Center at SRI.

a host part of 0. IBM/4.3 uses the standard broadcast address (all 1's) by default, but allows the broadcast address to be set (with *ifconfig*) for each interface. This allows networks consisting of both 4.2/RT and IBM/4.3 hosts to coexist while the upgrade process proceeds. In the presence of subnets, the broadcast address uses the subnet field as for normal host addresses, with the remaining host part set to 1's (or 0's, on a network that has not yet been converted). IBM/4.3 hosts recognize and accept packets sent to the logical-network broadcast address as well as those sent to the subnet broadcast address, and when using an all-1's broadcast, also recognize and receive packets sent to host 0 as a broadcast.

5.4. Routing

If your environment allows access to networks not directly attached to your host you will need to set up routing information to allow packets to be properly routed. Two schemes are supported by the system. The first scheme employs the routing table management daemon /etc/routed to maintain the system routing tables. The routing daemon uses a variant of the Xerox Routing Information Protocol to maintain up-to-date routing tables in a cluster of local area networks. By using the /etc/gateways file created by *htable*(8), the routing daemon can also be used to initialize static routes to distant networks (see the next section for further discussion). When the routing daemon is started up (usually from /etc/rc.local) it reads /etc/gateways if it exists and installs those routes defined there, then broadcasts on each local network to which the host is attached to find other instances of the routing daemon. If any responses are received, the routing daemons cooperate in maintaining a globally consistent view of routing in the local environment. This view can be extended to include remote sites also running the routing daemon by setting up suitable entries in /etc/gateways; consult *routed*(8C) for a more thorough discussion.

The second approach is to define a default or wildcard route to a smart gateway and depend on the gateway to provide ICMP routing redirect information to dynamically create a routing data base. This is done by adding an entry of the form

/etc/route add default *smart-gateway* 1

to /etc/rc.local; see *route*(8C) for more information. The default route will be used by the system as a "last resort" in routing packets to their destination. Assuming the gateway to which packets are directed is able to generate the proper routing redirect messages, the system will then add routing table entries based on the information supplied. This approach has certain advantages over the routing daemon, but is unsuitable in an environment where there are only bridges (i.e. pseudo gateways that, for instance, do not generate routing redirect messages). Further, if the smart gateway goes down there is no alternative, save manual alteration of the routing table entry, to maintaining service.

The system always listens, and processes, routing redirect information, so it is possible to combine both of the above facilities. For example, the routing table management process might be used to maintain up-to-date information about routes to geographically local networks, while employing the wildcard routing techniques for "distant" networks. The *netstat*(1) program may be used to display routing table contents as well as various routing oriented statistics. For example,

```
#netstat −r <Enter>
```

will display the contents of the routing tables, while

```
#netstat −r −s <Enter>
```

will show the number of routing table entries dynamically created as a result of routing redirect messages, etc.

5.5. Use of IBM/4.3 Machines as Gateways

Several changes have been made in IBM/4.3 in the area of gateway support (or packet forwarding, if one prefers). A new configuration option, GATEWAY, is used when configuring a machine to be used as a gateway. This option increases the size of the routing hash tables in the kernel. Unless configured with that option, hosts with only a single non-loopback interface never attempt to forward packets or to respond with ICMP error messages to misdirected packets. This change reduces the problems that may occur when different hosts on a network disagree as to the network number or broadcast address. Another change is that IBM/4.3 machines that forward packets back through the same interface on which they arrived will send ICMP redirects to the source host if it is on the same network. This improves the interaction of IBM/4.3 gateways with hosts that configure their routes via default gateways and redirects. The generation of redirects may be disabled with the configuration option IPSENDREDIRECTS=0 in environments where it may cause difficulties.

Local area routing within a group of interconnected Ethernets and other such networks may be handled by *routed*(8c). Gateways between the Arpanet or Milnet and one or more local networks require an additional routing protocol, the Exterior Gateway Protocol (EGP), to inform the core gateways of their presence and to acquire routing information from the core. An EGP implementation for 4.2BSD was done by Paul Kirton while visiting ISI, and any sites requiring such support that have not already obtained a copy should contact Joyce Reynolds (JKReynolds@usc-isif.arpa) for information. That implementation works with IBM/4.3 without kernel modifications. The Kirton implementation must be modified, as packets from the ICMP raw socket include the IP header like other raw sockets in IBM/4.3. If necessary, contact the Berkeley Computer Systems Research Group for assistance.

5.6. Network Servers

In IBM/4.3 most of the server programs are started up by a "super server," the Internet daemon. The Internet daemon, /etc/inetd, acts as a master server for programs specified in its configuration file, /etc/inetd.conf, listening for service requests for these servers, and starting up the appropriate program whenever a request is received. The configuration file contains lines containing a service name (as found in /etc/services), the type of socket the server expects (e.g. stream or dgram), the protocol to be used with the socket (as found in /etc/protocols), whether to wait for each server to complete before starting up another, the user name as which the server should run, the server program's name, and at most five arguments to pass to the server program. Some trivial services are implemented internally in *inetd*, and their servers are listed as "internal." For example, an entry for the file transfer protocol server would appear as

```
ftp     stream tcp     nowait root     /etc/ftpd       ftpd
```

Consult *inetd*(8c) for more detail on the format of the configuration file and the operation of the Internet daemon.

5.7. Network Data Bases

Several data files are used by the network library routines and server programs. Most of these files are host independent and updated only rarely.

File	Manual reference	Use
/etc/hosts	*hosts*(5)	host names

/etc/networks	*networks*(5)	network names
/etc/services	*services*(5)	list of known services
/etc/protocols	*protocols*(5)	protocol names
/etc/hosts.equiv	*rshd*(8C)	list of "trusted" hosts
/etc/rc.local	*rc*(8)	command script for starting servers
/etc/ftpusers	*ftpd*(8C)	list of "unwelcome" ftp users
/etc/hosts.lpd	*lpd*(8C)	list of hosts allowed to access printers
/etc/inetd.conf	*inetd*(8)	list of servers started by *inetd*

The files distributed are set up for ARPANET or other Internet hosts. Local networks and hosts should be added to describe the local configuration; the Berkeley entries may serve as examples (see also the next section). Network numbers will have to be chosen for each Ethernet and for each Token-Ring network. For sites not connected to the Internet, these can be chosen more or less arbitrarily, otherwise the normal channels should be used for allocation of network numbers.

5.7.1. Regenerating /etc/hosts and /etc/networks

When using the host address routines that use the Internet name server, the file /etc/hosts is only used for setting interface addresses and at other times that the server is not running, and therefore it need only contain addresses for local hosts. There is no equivalent service for network names yet. The full host and network name data bases are normally derived from a file retrieved from the Internet Network Information Center at SRI. To do this you should use the program /etc/gettable to retrieve the NIC host data base, and the program *htable*(8) to convert it to the format used by the libraries. You should change to the directory where you maintain your local additions to the host table and execute the following commands.

```
# /etc/gettable sri-nic.arpa <Enter>
Connection to sri-nic.arpa opened.
Host table received.
Connection to sri-nic.arpa closed.
# /etc/htable hosts.txt <Enter>
Warning, no localgateways file.
#
```

The *htable* program generates three files in the local directory: hosts, networks, and gateways. If a file "localhosts" is present in the working directory its contents are first copied to the output file. Similarly, a "localnetworks" file may be prepended to the output created by *htable*, and 'localgateways" will be prepended to gateways. It is usually wise to run *diff*(1) on the new host and network data bases before installing them in /etc. If you are using the host table for host name and address mapping, you should run *mkhosts*(8) after installing /etc/hosts. If you are using the name server for the host name and address mapping, you only need to install networks and a small copy of hosts describing your local machines. The full host table in this case might be placed somewhere else for reference by users. The gateways file may be installed in /etc/gateways if you use *routed*(8c) for local routing and wish to have static external routes installed when *routed* is started. This procedure is essentially obsolete, however, except for individual hosts that are on the Arpanet or Milnet and do not forward packets from a local network. Other situations require the use of an EGP server.

If you are connected to the DARPA Internet, it is highly recommended that you use the name server for your host name and address mapping, as this provides access to a much larger set of hosts than are provided in the host table. Many large organization on the network, currently have only a small percentage of their hosts listed in the host table retrieved from NIC.

5.7.2. /etc/hosts.equiv

The remote login and shell servers use an authentication scheme based on trusted hosts. The hosts.equiv file contains a list of hosts that are considered trusted and, under a single administrative control. When a user contacts a remote login or shell server requesting service, the client process passes the user's name and the official name of the host on which the client is located. In the simple case, if the host's name is located in hosts.equiv and the user has an account on the server's machine, then service is rendered (i.e. the user is allowed to log in, or the command is executed). Users may expand this "equivalence" of machines by installing a .rhosts file in their login directory. The root login is handled specially, bypassing the hosts.equiv file, and using only the /.rhosts file.

Thus, to create a class of equivalent machines, the hosts.equiv file should contain the official names for those machines. If you are running the name server, you may omit the domain part of the host name for machines in your local domain. For example, several machines on Berkeley's local network are considered trusted, so the hosts.equiv file is of the form:

```
ucbarpa
calder
dali
ernie
kim
matisse
monet
ucbvax
miro
degas
```

5.7.3. /etc/rc.local

Most network servers are automatically started up at boot time by the command file /etc/rc (if they are installed in their presumed locations) or by the Internet daemon (see above). These include the following:

Program	Server	Started by
/etc/rshd	shell server	inetd
/etc/rexecd	exec server	inetd
/etc/rlogind	login server	inetd
/etc/telnetd	TELNET server	inetd
/etc/ftpd	FTP server	inetd
/etc/fingerd	Finger server	inetd
/etc/tftpd	TFTP server	inetd
/etc/rwhod	system status daemon	/etc/rc
/etc/syslogd	error logging server	/etc/rc
/usr/lib/sendmail	SMTP server	/etc/rc
/etc/routed	routing table management daemon	/etc/rc
/etc/landump	IBM Token-Ring diagnostic daemon	/etc/rc.local

Consult the manual pages and accompanying documentation (particularly for sendmail) for details about their operation.

To have other network servers started up as well, the appropriate line should be added to the Internet daemon's configuration file /etc/inetd.conf, or commands of the following sort should be placed in the site-dependent file /etc/rc.local.

```
if [ -f /etc/routed ]; then
            /etc/routed & echo -n ' routed'          > /dev/console
fi
```

5.7.4. /etc/ftpusers

The FTP server included in the system provides support for an anonymous FTP account. Because of the inherent security problems with such a facility you should read this section carefully if you consider providing such a service.

An anonymous account is enabled by creating a user *ftp*. When a client uses the anonymous account a *chroot*(2) system call is performed by the server to restrict the client from moving outside that part of the file system where the user ftp home directory is located. Because a *chroot* call is used, certain programs and files used by the server process must be placed in the ftp home directory. Further, one must be sure that all directories and executable images are unwritable. The following directory setup is recommended.

```
# cd ~ftp <Enter>
# chmod 555 .; chown ftp .; chgrp ftp . <Enter>
# mkdir bin etc pub <Enter>
# chown root bin etc <Enter>
# chmod 555 bin etc <Enter>
# chown ftp pub <Enter>
# chmod 777 pub <Enter>
# cd bin <Enter>
# cp /bin/sh /bin/ls . <Enter>
# chmod 111 sh ls <Enter>
# cd ../etc <Enter>
# cp /etc/passwd /etc/group . <Enter>
# chmod 444 passwd group <Enter>
```

When local users wish to place files in the anonymous area, they must be placed in a subdirectory. In the setup here, the directory ~ftp/pub is used.

Another issue to consider is the copy of /etc/passwd placed here. It may be copied by users who use the anonymous account. They may then try to break the passwords of users on your machine for further access. A good choice of users to include in this copy might be root, daemon, uucp, and the ftp user. All passwords here should probably be "*".

Aside from the problems of directory modes and such, the ftp server may provide a loophole for interlopers if certain user accounts are allowed. The file /etc/ftpusers is checked on each connection. If the requested user name is located in the file, the request for service is denied. This file normally has the following names on our systems.

```
uucp
root
```

Accounts with nonstandard shells should be listed in this file. Accounts without passwords need not be listed in this file, the ftp server will not service these users.

6. SYSTEM OPERATION

This section describes some typical IBM/4.3 operations on an IBM RT PC, including:

- Bootstrapping and shutdown
- Diskettes
- LED numbers
- Keyboard
- Screen Status Line
- Checking system and device error logs
- Checking file systems and performing backups
- Moving file systems
- Monitoring system performance
- Recompiling and reinstalling system software
- Making local modifications
- Accounting for connect time and process resources
- Controlling resources
- Network troubleshooting
- Monitoring specific files

Procedures described here are used periodically to reboot the system, analyze error messages from devices, do disk backups, monitor system performance, recompile system software and control local changes.

6.1. Bootstrapping and Shutdown

During a normal reboot, the system checks the disks and comes up in multi-user mode without intervention at the console. To bring the system up in single-user mode, press and hold down **<Ctrl>-<C>** as soon as the system prints the date. This interrupts the boot with only the console terminal active. It is also possible to allow the file system checks to complete and then to return to single-user mode by signaling *fsck* with a QUIT signal (^).

To boot from the console, press and hold down the following keys:

<Ctrl>-<Alt>-<Pause>

The system tries to boot from a diskette, then from the hard disk.

You can also boot in single-user mode by explicitly typing the system name in response to the boot prompt:

```
:hd(0,0)vmunix <Enter>
```

To bring the system up to a multi-user mode from single-user mode, press and hold down **<Ctrl>-<D>** on the console. The system executes /etc/rc (a multi-user restart script) and /etc/rc.local, and comes up on the terminals listed as active in the file /etc/ttys. See *init*(8) and *ttys*(5). Note, however, that this does not do a file system check. If the previous shutdown was not clean, you should run "fsck −p" or force a reboot with *reboot*(8) to check the disks.

When the system is in multi-user mode, you can take it to single-user mode with either:

```
# kill 1 <Enter>
```

or the *shutdown*(8) command. The latter is much more polite if there are other users logged in. Either command will kill all processes and give you a shell on the console, as if you had just booted. File systems remain mounted after the system becomes single-user. To change to multi-user mode again, use the following commands:

```
# cd /  <Enter>
# /etc/umount -a  <Enter>
#  <Ctrl>-<D>
```

Note that the file /usr/adm/shutdownlog records each system shutdown, crash, processor halt, and reboot, with its associated cause.

6.2. Diskettes

Diskettes normally load after the LED display reaches "22." If the LED remains at "22" and the diskette drive light flashes on and off, the diskette does not contain a boot record. Be sure you have a bootable diskette in the drive.

6.3. LED Numbers

The LED displays show the following numbers during a boot of a IBM/4.3 system:

- **00,01,03,09,10,14,15,16,17** - part of the internal power-on sequence.
- **22-29** - booting from diskette.
- **22, with diskette drive light flashing** - not an IBM RT PC boot diskette, no diskette in drive, or drive door not closed.
- **94** - kernel stack overflow.
- **96** - unsupported memory configuration.
- **98** - /boot not found on hd0a.
- **99** - key is in locked position.

If no numbers are displayed, the system has been halted by /etc/halt.

When IBM/4.3 is running, the LED display shows the "load average" for the system. A lightly loaded system will display numbers ranging between "00" and "15" (meaning 0.0 and 1.5). A heavily loaded system will display numbers greater than "50" (5.0).

6.4. Keyboard

The following key combination reboots the entire system (causing a warm IPL, similar to <Ctrl>-<Alt>-<Del> on an IBM PC). Hold each key down in sequence.

<Ctrl>-<Alt>-<Pause>

The following key combination forces the system to panic, creating a coredump after rebooting. Hold each key down in sequence.

<Ctrl>-<Alt>-<Scroll Lock>

Note that you should NOT use this when the kernel is running, except in extreme emergencies. Instead, see "Bootstrapping and Shutdown" above.

Some users may wish to have their keyboards map to an IBM PC keyboard. The single change required is to swap the functions of the Caps Lock and Ctrl keys on the IBM RT PC keyboard. To do this, see *pf*(1) and *kbdemul*(4).

6.5. Screen Status Line

While IBM/4.3 is running, the last line of the screen is used to display keyboard status, such as CAPS for < Caps Lock > key pressed; SHIFT for < Shift > key pressed; ALT for < Alt > key pressed; CRTL for < Ctrl > key pressed; and ACTION for < Action > key pressed.

6.6. Checking System and Device Error Logs

When serious errors occur on peripherals or in the system, the system displays a warning diagnostic on the console. These messages are collected by the system error logging process *syslogd*(8) and written into a system error log file /usr/adm/messages. Less serious errors are sent directly to *syslogd*, which may log them on the console. The error priorities that are logged and the locations to which they are logged are controlled by /etc/syslog.conf. See *syslogd*(8) for details.

Error messages issued by the devices in the system are described with the drivers for the devices in Volume I, Section 4, of this manual. If errors occur suggesting hardware problems, you should contact your hardware support group. You should check the error log file regularly, using the command:

```
tail −r /usr/adm/messages <Enter>
```

6.7. Checking File Systems and Performing Backups

You should periodically check all file systems for consistency. Use the *fsck*(1) command weekly in the absence of problems, and always (usually automatically) after a crash. You can use the procedures of *reboot*(8) to put the system in a state where a file system check can be performed manually or automatically.

You should also back up file systems regularly. Use *dump*(8) for both complete and incremental dumps. Berkeley recommends a towers-of-hanoi dump sequence with full dumps taken every month.

Use three sets of dump media (streaming tape): daily, weekly, and monthly. Perform daily dumps circularly on the daily set with sequence '3 2 5 4 7 6 9 8 9 9 9 . . .' Each weekly is a level 1; daily dump sequence levels restart after each weekly dump. Full dumps are level 0; daily sequence levels also restart after each full dump.

Thus a typical dump sequence would be:

Dump ID	Level Number	Date	Opr	Size
FULL	0	Nov 24, 1984	sy	137K
D1	3	Nov 28, 1984	sy	29K
D2	2	Nov 29, 1984	ac	34K
D3	5	Nov 30, 1984	ac	19K
D4	4	Dec 1, 1984	ac	22K
W1	1	Dec 2, 1984	sy	40K
D5	3	Dec 4, 1984	ac	15K
D6	2	Dec 5, 1984	sy	25K
D7	5	Dec 6, 1984	sy	15K
D8	4	Dec 7, 1984	ac	19K
W2	1	Dec 9, 1984	sy	118K
D9	3	Dec 11, 1984	ac	15K
D10	2	Dec 12, 1984	sy	26K
D1	5	Dec 15, 1984	ac	14K
W3	1	Dec 17, 1984	sy	71K

D2	3	Dec 18, 1984	sy	13K
FULL	0	Dec 22, 1984	ac	135K

Take weekly dumps often enough that daily dumps always fit on one streaming tape, and never get to the sequence of 9's in the daily level numbers.

Operators can execute **/etc/dump** w at login to learn what needs to be dumped (based on the /etc/fstab information). Be sure to create a group "operator" in the file /etc/group so that *dump*(8) can notify logged-in operators when it needs help.

Dumping files by name is best done with *tar*(1) but the amount of data moved is limited to a single tape. If there are enough drives, you can copy entire disks with *dd*(1) using the raw special files and an appropriate blocking factor. The number of sectors per track is usually a good value to use; consult /etc/disktab.

You should also make full dumps of the root file system on a regular schedule. This is especially true on a system with only one disk. If the root file system is damaged by a hardware or software failure, you can rebuild a workable disk by restoring the dump, using the MINIROOT diskette.

Exhaustion of user file space is certain to occur now and then. You can impose disk quotas, or you might use the programs *du*(1), *df*(1), and *quot*(8) combined with messages of the day and personal letters.

6.8. Moving File Systems

If you have a streaming tape, the best way to move a file system is to dump it to tape using *dump*(8), create a new file system using *newfs*(8), and restore the tape using *restore*(8). If you do not have tape, *dump* accepts an argument telling where to put the dump; you might use another disk. Filesystems may also be moved by piping the output of *dump* to *restore*. The restore program uses an "in-place" algorithm that allows file system dumps to be restored without concern for the original size of the file system. Further, portions of a file system may be selectively restored in a manner similar to the tape archive program.

To merge a file system into an existing one, use *tar*(1).

To shrink a file system, dump the original and restore it onto the new file system. To shrink the root file system with only one disk drive, the procedure is more complicated:

(1) Dump the root file system to a remote streaming tape using *rdump*(8).

(2) Bring the system down.

(3) Use the MINIROOT diskette and *restore.tape*(8) to install the new root file system.

(4) Boot normally using the newly-created disk file system.

Note that if you add new disk drivers, you must modify the default disk partition tables in /etc/disktab and add the drivers to the standalone system in /sys/standca.

6.9. Monitoring System Performance

The *systat* program provided with the system is designed to be an aid to monitoring system-wide activity. The default "pigs" mode shows a dynamic "ps." By running *systat* when the system is active you can judge the system activity in several dimensions: job distribution, virtual memory load, paging and swapping activity, device interrupts, and disk and cpu utilization. Ideally, there should be few blocked (b) jobs; little paging or swapping activity; available bandwidth on the disk devices (most single arms peak out at 20-30 tps in practice); and high (above 50%) user cpu utilization (us).

If the system is busy, then the count of active jobs may be large, and several of these jobs may often be blocked (b). If the virtual memory is active, then the paging daemon will be running (sr will be non-zero). It is healthy for the paging daemon to free pages when the virtual memory gets active; it is triggered by the amount of free memory dropping below a threshold and increases its pace as free memory goes to zero.

If you run in the "vmstat" mode when the system is busy, you can find imbalances by noting abnormal job distributions. If many processes are blocked (b), then the disk subsystem is overloaded or imbalanced. If you have several non-dma devices or open teletype lines that are "ringing," or user programs that are doing high-speed non-buffered input/output, then the system time may go high (60-70% or higher). It is often possible to pin down the cause of high system time by seeing if there is excessive context switching (cs) and per-device interrupt counts, interrupt activity (in) or system call activity (sy).

If the system is heavily loaded or if you have little memory for your load (2 megabyte is little in most any case), then the system may be forced to swap. This is likely to be accompanied by a noticeable reduction in system performance and pauses when interactive jobs such as editors swap out. If you expect to be in a memory-poor environment for an extended period, you might consider administratively limiting system load.

6.10. Recompiling and Reinstalling System Software

It is easy to regenerate the system, and it is a good idea to try rebuilding pieces of the system to build confidence in the procedures. The system consists of two major parts: the kernel itself (/sys) and the user programs (/usr/src and subdirectories). The major part of this is /usr/src.

The major library is the C library in /usr/src/lib/libc. The library is remade by changing into the correct directory and typing:

```
# make <Enter>
```

and then installed by typing:

```
# make install <Enter>
```

Similarly, typing:

```
# make clean <Enter>
```

cleans up.

The source for all other libraries is kept in subdirectories of /usr/src/usr.lib; each has a makefile and can be recompiled by the above recipe.

NOTE: The code to support IEEE floating point emulation is distributed only in object form on the system. If the system is rebuilt, be very careful not to delete the /usr/src/usr.lib/libfp/emulfp/*.o modules. (They will not be removed by a *make clean.*) It is strongly recommended that you *tar* these modules to a diskette before starting a rebuild.

If you look at /usr/src/Makefile, you will see that you can recompile the entire system source with one command. To recompile a specific program, find out where the source resides with the *whereis*(1) command, then change to that directory and remake it with the makefile present in the directory. For instance, to recompile "date," all one has to type is:

```
# whereis date <Enter>
date: /usr/src/bin/date.c /bin/date /usr/man/man1/date.1 <Enter>
# cd /usr/src/bin <Enter>
# make date <Enter>
```

This will create an unstripped version of the binary of "date" in the current directory. To install the binary image, use the install command:

```
# install −s date /bin/date <Enter>
```

The −s option will insure the installed version of date has its symbol table stripped. The install command should be used instead of mv or cp as it understands how to install programs even when the program is currently in use.

If you wish to recompile and install all programs in a particular target area, you can override the default target by typing:

```
# make <Enter>
# make DESTDIR=pathname install <Enter>
```

To regenerate all the system source you can type:

```
# cd /usr/src <Enter>
# make <Enter>
```

If you modify the C library (perhaps to change a system call) and want to rebuild and install everything from scratch, you have to be careful. You must insure the libraries are installed before the remainder of the source; otherwise the loaded images will not contain the new routine from the library. The following sequence will accomplish this:

```
# cd /usr/src <Enter>
# make clean <Enter>
# make build <Enter>
# make installsrc <Enter>
```

The first *make* removes any existing binaries in the source trees to ensure that everything is reloaded. The next *make* compiles and installs the libraries and compilers, then compiles the remainder of the sources. The final line installs all of the commands not installed in the first phase.

6.11. Making Local Modifications

To keep track of changes to system source, Berkeley migrates changed versions of commands in /usr/src/bin, /usr/src/usr.bin, and /usr/src/ucb in through the directory /usr/src/new and out of the original directory into /usr/src/old for a time before removing them. (Berkeley also uses /usr/new for the programs that constitute the contributed software portion of the 4.3BSD distribution.) Locally written commands that aren't distributed are kept in /usr/src/local and their binaries are kept in /usr/local. This allows /usr/bin, /usr/ucb, and /bin to correspond to the distribution tape (and to the manuals that people can buy). People wishing to use /usr/local commands are made aware that they aren't in the base manual. As manual updates incorporate these commands, they are moved to /usr/ucb.

A directory /usr/junk to throw garbage into, as well as binary directories /usr/old and /usr/new, are useful. The man command supports manual directories such as /usr/man/mano for old and /usr/man/manl for local to make this or something similar practical.

6.12. Accounting for Connect Time and Process Resources

IBM/4.3 optionally records two kinds of accounting information: connect time accounting and process resource accounting. The connect time accounting information is stored in the file /usr/adm/wtmp, and is summarized by the program *ac*(8). The process time accounting information is stored in the file /usr/adm/acct after it is enabled by *accton*(8), which is analyzed and summarized by the program *sa*(8).

If you need to charge for computing time, you can implement procedures based on the information provided by these commands. A convenient way to do this is to give commands to the clock daemon /etc/cron to be executed every day at a specified time. This is done by adding lines to /usr/lib/crontab; see *cron*(8) for details.

6.13. Controlling Resources

Resource control in the current version of IBM/4.3 is elaborate compared to most UNIX operating systems. The disk quota facilities developed at the University of Melbourne have been incorporated in the system and allow control over the number of files and amount of disk space each user may use on each file system. In addition, the resources consumed by any single process can be limited by the mechanisms of *setrlimit*(2). As distributed, the latter mechanism is voluntary, though sites may choose to modify the login mechanism to impose limits not covered with disk quotas.

To use the disk quota facilities, the system must be configured with "options QUOTA." Then place file systems under the quota mechanism by creating a null file *quotas* at the root of the file system, running *quotacheck*(8), and modifying /etc/fstab to indicate the file system is read-write with disk quotas (an "rq" type field). Then run the program *quotaon*(8) to enable quotas.

Apply individual quotas using the quota editor *edquota*(8). Users may view their quotas (but not those of other users) with the *quota*(1) program. Use the *repquota*(8) program to summarize the quotas and current space usage on a particular file system or file systems.

You can enforce quotas with *soft* and *hard* limits. When a user first reaches a soft limit on a resource, a message appears on his/her terminal. If the user fails to lower the resource usage below the soft limit, the next *login* causes a warning about excessive usage. Should three login sessions go by with the soft limit breached, the system then treats the soft limit as a *hard* limit and disallows any allocations until enough space is reclaimed to bring the user back below the soft limit. Hard limits are strictly enforced, resulting in errors when a user tries to create or write a file. Each time a hard limit is exceeded the system will generate a message on the user's terminal.

Consult the document "Disc Quotas in a UNIX Environment" in the *UNIX System Manager's Manual* and the related manual pages for more information.

6.14. Network Troubleshooting

If you have anything more than a trivial network configuration, from time to time you are bound to run into problems. Before blaming the software, first check your network connections. On networks such as the Ethernet, a loose cable tap or misplaced power cable can result in severely deteriorated service. The *netstat*(1) program may be of aid in tracking down hardware malfunctions. In particular, look at the −i and −s options in the manual page.

Should you believe a communication protocol problem exists, consult the protocol specifications, and attempt to isolate the problem in a packet trace. The SO_DEBUG option may be supplied before establishing a connection on a socket, in which case the system will trace all traffic and internal actions (such as timers expiring) in a circular trace buffer. This buffer may then be printed out with the *trpt*(8C) program. Most servers distributed with the system accept a −d option that forces all sockets to be created with debugging turned on. Consult the appropriate manual pages for more information.

6.15. Monitoring Specific Files

As part of normal system operations, you should periodically review the following files (some of which are system-specific):

/etc/fstab	how disk partitions are used
/etc/disktab	disk partition sizes
/etc/printcap	printer data base
/etc/gettytab	terminal type definitions
/etc/remote	names and phone numbers of remote machines for *tip*(1)
/etc/group	group memberships
/etc/motd	message of the day
/etc/passwd	password file; each account has a line
/etc/rc.local	local system restart script; runs reboot; starts daemons
/etc/inetd.conf	local internet servers
/etc/hosts	host name data base
/etc/networks	network name data base
/etc/services	network services data base
/etc/hosts.equiv	hosts under same administrative control
/etc/syslog.conf	error log configuration for *syslog*(8)
/etc/ttys	enables/disables ports
/usr/lib/crontab	commands that are run periodically
/usr/lib/aliases	mail forwarding and distribution groups
/usr/adm/acct	raw process account data
/usr/adm/lpd-errs	line printer daemon error log
/usr/adm/messages	system error log
/usr/adm/ppd-errs	page printer error log
/usr/adm/shutdownlog	log of system reboots
/usr/adm/wtmp	login session accounting

Appendix A. Advanced Interactive Executive (AIX) and IBM/4.3 Co-residence

It is possible to have AIX and IBM/4.3 systems on the same machine. Each system should have its own disk or disks. For the two systems to co-reside, the following steps must be taken (a two-disk system is described; a three-disk system is similar):

1. Installing AIX on an Existing IBM/4.3 System

AIX normally expects to be booted from drive 0, so move the IBM/4.3 system to drive 1 or drive 2. This can be done by copying the file systems (with dump/restore) or by physically moving the disks. To move the disks, see *IBM RT PC User Setup Guide*, SV21-8020.

1.1. Creating a Minidisk (partition) Table on Existing IBM/4.3 Disks

For the AIX installation procedure not to use the IBM/4.3 disk (or disks), a minidisk (partition) table must be established on each IBM/4.3 disk that uses all the available space. This is done by the *minidisk*(8R) utility, part of the *sautil*(8R) standalone utility. This procedure assumes there is no existing IBM/4.3 minidisk (partition) table.

The procedure is:

(1) Boot up the *sautil* utility. It is located in /usr/stand/sautil on an installed system, and on the standalone SAUTIL diskette.

(2) Select the "minidisk" menu item.

(3) Initialize the minidisk directory - this makes all the space on the disk available.

(4) Create the standard partition tables by using the *standard* command (this corresponds to the normal a, b, and g partitions of a IBM/4.3 disk.) It also creates a boot partition to hold the bootstrap, since this exists before the "a" partition.

2. Installing AIX and IBM/4.3 on a New Machine

Install IBM/4.3 first, then AIX as follows:

- Create a IBM/4.3 minidisk (partition) table on each IBM/4.3 disk
- Install the IBM/4.3 root and /usr file systems onto an available disk (AIX uses drive 0 and possibly drive 1; IBM/4.3 uses drives after AIX.) See the section "Changing Installation Options," of the chapter "Installation Procedures," in this article.
- Install AIX -- it should only use the drives set aside for it, leaving the IBM/4.3 drives alone because they have valid minidisk tables using all the available space.

3. Booting AIX

After AIX is installed, it puts its bootstrap into the boot block of drive zero. This means when the system is booted, AIX runs (since the default boot order is f0, f1, d0, d1, d2).

4. Booting IBM/4.3

There are two alternatives here. The simplest solution is to use a IBM/4.3 boot diskette to boot IBM/4.3 from drive 1. This requires manual intervention (or a non-standard boot diskette) since the the standard boot diskette attempts to boot hd(0,0)vmunix. Since the IBM/4.3 root is on drive 1 (or drive 2), you should boot IBM/4.3 from hd(1,0)vmunix or hd(2,0)vmunix as appropriate. A generic kernel then asks for the root disk (which is either hd1 or hd2 depending upon which drive is used for the root).

The other alternative is to change the boot order in non-volatile ram so the boot order is f0, f1, d1, d0, d2. This should be done in those cases where AIX is not used frequently. The "iplsource" option of *sautil*(8R) describes how to do this. In this case, a non-generic kernel must be used. See the section "Building New System Images" in the chapter "System Setup" of this article for information on building kernels. A generic kernel will attempt to use hd0a as its root device and then panic as there is not a proper superblock there.

5. Creating a Minidisk Partition Table on a New IBM/4.3 Disk

There are two reasons for creating a partition table on a disk before the IBM/4.3 installation:

- To prepare for eventual AIX installation (this is not critical as the partition table can always be added later)
- To use non-standard partitions. This is often done because the standard partitions do not fit every situation. In particular, it is often the case on small (40Mb) disks, the swap partition is insufficient to run large applications (such as window managers) and more swap space must be allocated. In other cases, it is desirable to create more partitions, or to have only a swap area and a large filesystem (on a second drive for example) rather than the standard three partitions.

(WARNING: changing the size or location of a partition containing a filesystem effectively destroys all the contents of that filesystem. You must do a dump/restore to change a filesystem's size and keep the contents). The procedure for creating a non-standard minidisk (partition) table is:

(1) Boot up *minidisk*(8R) utility (as described earlier).

(2) Initialize the minidisk directory by using the *initialization* command to make all the space available.

(3) Create standard partitions by using the *standard* command.

(4) Delete the partitions not needed (but keep the partition named "boot" as this is required to align the IBM/4.3 minidisks on cylinder boundaries).

(5) Create (or recreate) the new partitions. Usually one makes the 'b' (swap) partition bigger, and the 'g' (usr) partition smaller.

6. Installing IBM/4.3 on an Existing AIX Machine

As AIX automatically uses all the available disks, this is only feasible in two cases:

- A new disk can be added. In this case, create the IBM/4.3 minidisk (partition) table and install IBM/4.3 on the new disk.
- The AIX system must be dumped to tape or diskette. Then IBM/4.3 is installed as described above and AIX is re-installed from the dumped tape or diskette.

7. Shared Swap Partitions

IBM/4.3 can use the AIX swap partition, but a non-generic kernel must be configured to do so. Assume AIX on drive 0, IBM/4.3 on drive 1. This kernel could have a configuration like:

```
config   vmunix          root on hd1 swap on hd0 and hd1
```

8. Generating a Sautil Diskette

A bootable standalone diskette is generated by taking the program (such as */boot*) and writing it onto a diskette with *doswrite* (see *dosread*(1)) with the appropriate options.

To generate a standalone bootable sautil diskette, do the following:

```
cd /sys/standca  < Enter >
make boot.out  < Enter >
doswrite -i -b -v boot.out  < Enter >
```

Appendix B. MINIROOT Kernel Configured Devices

This appendix lists the configurable devices supported by the MINIROOT kernel. For those adapters with selectable addresses, the supported addresses are listed.

Device	Maximum	Description	Address
Displays:			
apa16	1	IBM 6155 Extended Monochrome Graphics Display	
apa8c	1	IBM 6154 Advanced Color Graphics Display	
apa8	1	IBM 6153 Advanced Monochrome Graphics Display	
aed	1	IBM ACIS experimental display	
mono	1	IBM 5151 PC Monochrome Display	
Networks:			
lan	1	IBM Token-Ring Network Adapter	f00001c0
un	1	IBM Baseband Adapter for Ethernet	f4080000 f4088000 f4090000 f4098000
Disks:			
hdc	2	IBM PC/AT or ESDI Controller (hard disk function)	f00001f0 f0000170
hd	3	Hard disk drives	
fdc	1	IBM PC/AT or ESDI Controller (diskette function)	f00003f2 f0000372
fd	2	Diskette drives	
Tape:			
stc	1	Streaming tape controller	f00001e8
st	1	Streaming tape drive	

This page intentionally left blank.

Appendix C. Building a Master/Server Machine

This appendix describes how, in a very small installation, you can make a single IBM RT PC function as a master for network installations and a server for both the Andrew File System and RVD for a network of IBM 6152 workstations.

When you have completed these instructions, the RT will have the following:

- the IBM 4.3 base system root and user partitions
- a "minimal root" partition (used during installation of 6152 workstations)
- an RVD "root" pack (used during the operation of "minimal" configuration 6152 workstations)
- an Andrew File System "usr" pack (used during the operation of "minimal" and "reduced" configuration 6152 workstations)

Disk Partitioning

The RT needs the following partitions:

- A standard root (hd0a) partition
- A standard user (hd0g) partition
- A 70 MB scratch partition (we used hd1g, which was all of hd1)
- A 7 MB partition for the RVD root pack (we used sc0a)
- A 2 MB partition for the minimal root partition "/minroot.atr" (we used sc0d)
- A large partition for the Andrew File System (we used sc0e, which covered the rest of a 200 MB SCSI disk)

Use the *minidisk* function of the SAUTIL diskette to create the hard disk partitions. (You create the SCSI disk partitions after you install the base system.)

The remainder of this appendix provides the installation steps required.

1. **Perform a tape installation of IBM/4.3 on the RT.** Follow the instructions for installing an RT from streaming tape, found in Chapter 3 of this article, with one significant change:

   Change the value for "optional system components" from the default ("all") to "none."

2. **Partition any SCSI disks.**

   (1) Use the /etc/minidisk command to partition your SCSI disks. (See *minidisk*(8).)

   (2) Reboot the system so that the new minidisk tables are read by the kernel.

3. **Edit the /etc/hosts file.**

   (1) Insert the hostname "vice1" *before* the hostname "master," thus:

   ```
   46.0.0.1   vice1  master
   ```

   (2) Change *46.0.0.1* to your machine's network address.

   (3) Run the tailor command, and change the hostname to vice1. (See *tailor*(8).)

   (4) Reboot the system.

4. Convert the RT into a VICE server and client.

(1) Login as root and type the root password.

(2) Type the following commands to mount the scratch partition:

```
# newfs  /dev/hd1g  <Enter>
# mount  /dev/hd1g  /mnt  <Enter>
# cd  /mnt  <Enter>
```

(3) Insert the Andrew File System (AFS) tape into the streaming tape drive.

(4) Issue the following commands to extract the server and client binaries from the tape and install them on the RT:

```
# tar  xfp /dev/st0  usr/andrew/fsserver  usr/andrew/fsclient  <Enter>
# cd /mnt/usr/andrew/fsserver  <Enter>
# tar  cf  -  . | (cd  /; tar xfp -)  <Enter>
# cd  /mnt/usr/andrew/fsclient  <Enter>
# tar  cf  -  . | (cd  /; tar xfp -)  <Enter>
# cd /  <Enter>
# umount  /dev/hd1g  <Enter>
```

5. Set up the VICE server.

(1) Type the following commands:

```
# newfs  /dev/sc0e  <Enter>
# mount  /dev/sc0e  /vicepa  <Enter>
```

(2) Edit /etc/fstab to add the following line directly after the line defining the root partition:

```
/dev/sc0e:/vicepa:rw:1:2
```

(3) Type the following commands to add the admin account to your password database:

```
# cat  /etc/passwd.adm  > >  /etc/passwd  <Enter>
# vipw  <Enter>
```

This will start the *vi*(1) editor, allowing you to edit the password file.

(4) Simply exit by typing:

```
:q!  <Enter>
```

The password database will be updated automatically.

(5) Type the following commands to initialize the Andrew File System user database:

```
# cd  /vice/db  <Enter>
# /vice/bin/pwd2pdb  -p  /etc/passwd  -g  /usr/admin/groups  >  vice.pdb  <Enter>
# /vice/bin/pcfgen  vice.pdb  <Enter>
# echo  00000000  >  /vice/vol/maxvolid  <Enter>
```

Note that in the previous command, it is important that the string of zeroes contain eight (8) zeroes.

(6) Type the following command:

```
# /vice/bin/startup  <Enter>
```

Wait for a syslog message that says "File server has started." Do not confuse this with messages that say "File server is starting."

6. Create the root volume.

(1) Type the following command:

```
# /vice/bin/vol-create  /vicepa  root  <Enter>
```

Note that vol-create is used to create only the first AFS volume. Afterwards the createvol command is used.

(2) Build the volume database with the following command:

```
# /vice/bin/bldvldb.sh  <Enter>
```

7. Start the client process.

Type the following command:

```
# /etc/viced  <Enter>
```

8. Create andrew and usr volumes.

(1) Type the following commands:

```
# /vice/bin/createvol  andrew  vice1  /vicepa  <Enter>
# /vice/bin/createvol  usr  vice1  /vicepa  <Enter>
```

(2) Type the following commands to give the admin account full privileges to the andrew and usr volumes, which will have no quotas:

```
# sed  '/^wheel/s/$/,admin/g'  /etc/group  >  /etc/group.new   <Enter>
# mv  /etc/group.new   /etc/group   <Enter>
# login  admin   <Enter>
Password: install   <Enter>
% cd  /andrew   <Enter>
% /usr/andrew/bin/fs  sa  .  admin all   <Enter>
% /usr/andrew/bin/fs  mkmount  andrew  andrew   <Enter>
% /usr/andrew/bin/fs  sv  andrew  -a  0  <Enter>
% /usr/andrew/bin/fs  sa  andrew  admin  all  <Enter>
% /usr/andrew/bin/fs  mkmount  usr  usr   <Enter>
% /usr/andrew/bin/fs  sv  usr  -a  0  <Enter>
% /usr/andrew/bin/fs  sa  usr  admin  all  <Enter>
```

9. Load the andrew volume.

(1) Type the following commands:

```
% /usr/andrew/bin/loadafs  /usr/andrew  /andrew  ""  <Enter>
```

(2) Become the superuser again by typing the **su** command and supplying the superuser password.

(3) Type the following commands:

```
# rm  -rf  /usr/andrew   <Enter>
# ln  -s  /andrew/andrew  /usr/andrew   <Enter>
```

10. Load the BE2 volume.

(1) Insert the BE2 tape in the streaming tape drive.

(2) Type the following commands:

```
# newfs  /dev/hd1g  <Enter>
# mount  /dev/hd1g /mnt   <Enter>
# cd  /mnt   <Enter>
# tar  -xfp  /dev/st0  usr/andrew/X11fonts  usr/andrew/bin  usr/andrew/dlib

      usr/andrew/doc  usr/andrew/etc  usr/andrew/fonts  usr/andrew/help
      usr/andrew/include  usr/andrew/lib  usr/andrew/man   <Enter>
# exit   <Enter>
% /usr/andrew/bin/loadafs  /mnt/usr/andrew  /andrew  ""   <Enter>
% cd /   <Enter>
% su   <Enter>
# umount  /dev/hd1g   <Enter>
```

11. **Load the usr volume.**

(1) Type the following commands:

```
# newfs  /dev/hd1g   <Enter>
# mount  /dev/hd1g  /mnt   <Enter>
# mkdir  /mnt/usr   <Enter>
# cd  /mnt/usr   <Enter>
```

(2) Insert the ROOT/USER tape into the streaming tape drive.

(3) Type the following commands:

```
# mt  -f  /dev/rst0  rewind   <Enter>
# mt  -f  /dev/nrst0  fsf   <Enter>
# restore  xvf  /dev/st0   <Enter>
# ln  -s  /andrew/andrew  /mnt/usr/andrew   <Enter>
# cd  /mnt   <Enter>
```

(4) Follow the normal instructions for installing the other tapes and diskettes (if you have them) with one exception: replace the "cd /" to "cd /mnt" in ALL cases. This applies to the following tapes and diskettes:

- the X11 tape
- the 6152 Academic System 5-1/4 inch diskette (but do not create the minimal root file system; you will do this later.)
- the Professional Pascal diskette

(5) Type the following commands to install the /mnt filesystem into the usr volume:

```
# cd  /mnt   <Enter>
# mv  vmunix.atr  /vmunix.atr   <Enter>
# /mnt/usr/sys/dist_atr/make.site  -r  -u  /mnt/usr   <Enter>
# exit   <Enter>
% /usr/andrew/bin/loadafs  /mnt/usr  /andrew  ""   <Enter>
```

12. **Make the RT an RVD server.**

(1) Become the superuser again by typing the **su** command and supplying the superuser password.

(2) Type the following command to create the root RVD pack:

```
# newfs /dev/sc0a   <Enter>
```

(3) Note the number of sectors displayed by the newfs command. You will need it later. In our case, there were 16128 sectors.

(4) Type the following commands:

```
# cd /etc/rvd  <Enter>
# ./rvd.mkserver  <Enter>
```

The rvd.mkserver program will set up the system as an RVD server.

(5) Respond to the following prompts as shown:

```
next available link is vdsrv0 - make link? (y/n)  y  <Enter>

which device to link to?  (... CR)  sc0a  <Enter>

next available link is vdsrv1 - make link? (y/n) n  <Enter>

calling vddb (see vddb(8)).
would you like to see an example first? (y/n)
```

To this prompt, respond **y** if you would like to see the example, or **n** if you do not.

(6) Continue with the following responses:

```
hit return key to begin vddb session: <Enter>

Password: <Enter>

Ready
> add physical  <Enter>
Physical disk file name: /dev/vdsrv0  <Enter>
Size in 512-byte blocks: 16128  <Enter>
```

(This is the number of sectors displayed by the newfs command above.)

```
Are you sure (y or n)? y  <Enter>

Ready
> add virtual  <Enter>
Virtual disk name: root  <Enter>
Virtual disk uid: 0  <Enter>
Owner: root  <Enter>
Read-only password: <Enter>
Exclusive password:  <password>  <Enter>
```

We recommend you enter a password here. This password must be used in order to spin up the root pack in exclusive (read/write) mode. Remember this password.

```
Shared password: <Enter>
Disk size in 512-byte blocks: 16128  <Enter>
```

Again, this is the number of sectors displayed by the newfs command above.

```
Allowable modes: 5  <Enter>
Owning host (<CR> for none): <Enter>
Physical disk name (<CR> for any): /dev/vdsrv0  <Enter>

Ready
> quit  <Enter>
```

setting rvd command authorization password

```
enter new password:
```

At this point, enter the RVD command authorization password. This password may be used in place of the other three passwords above, so it should not be a null password.

Note: This password WILL be displayed on the screen!

(7) The RVD server program starts automatically. When you are prompted for the RVD system message that appears whenever an RVD pack is spun up, continue thus:

```
enter any rvd system message; end with ^D
< Your RVD system message >
< CTRL > -D
Password: < Your RVD command authorization password >
```

The password will not be displayed on the screen this time.

```
# cd  /etc/rvd   < Enter >
# cp  rvdstart.client  rvdstart   < Enter >
```

13. Make the system an RVD client

(1) Type the following commands:

```
# cd  /dev   < Enter >
# ./MAKEDEV  rvd   < Enter >
# cd  /etc/rvd   < Enter >
# mkdir  /root   < Enter >
# ./rvd.mkclient   < Enter >
```

(2) Respond to the following prompts:

```
enter a new pack into the database? (y/n) y   < Enter >

what will the pack's name be? root   < Enter >

is this pack to be mounted by default (d) or is it
absolutely-must-be-mounted (a)?  (default is no mount)
enter a, d, or CR:  < ENTER >

what spinup mode, read-only (r), exclusive read-write (x),
or both (rx), do you want?  (default is read-only) rx   < Enter >

on what server(s) does this pack reside? vice1   < Enter >

what drive number (0-9)? (no default) 0   < Enter >

where do you want to mount this file system?  (default is /usr for
usr pack, /rvdusr for rvdusr, and /usr/src for src)  /root   < Enter >

enter password for this pack, if any:  < Enter >

any comment to associate with this pack in the database entry? < ENTER >

enter a new pack into the database? (y/n) n   < Enter >
```

14. Load the root pack.

(1) Type the following commands:

```
# up -x root <Enter>
Password: <Your exclusive password for the root pack>
# cd /root <Enter>
# (cd /; tar cf - bin lib etc ) | tar xfp - <Enter>
# cd /root/etc/rvd <Enter>
# rm rvddb rvdtab rvdauthor <Enter>
```

Note: These files contain the RVD passwords and must not be served via RVD!

(2) Type the following commands:

```
# cd / <Enter>
# down root <Enter>
```

15. Create the minimal root partition.

(1) Type the following commands:

```
# cd /andrew/usr/sys/dist_atr <Enter>
# newfs /dev/sc0d <Enter>
# mkdir /minroot.atr <Enter>
# mount /dev/sc0d /minroot.atr <Enter>
```

(2) Edit the /etc/fstab file and add the following line:

```
/dev/sc0d:/minroot.atr:rw:1:2
```

(3) Type the following commands:

```
# cd /andrew/usr/sys/dist_atr <Enter>
# ./make.minimal <Enter>
```

This command will take approximately 5 minutes to run.

16. Miscellaneous

Type the following commands to set up symbolic links in /usr to the AFS usr volume:

```
# cd /usr <Enter>
# rm -rf man doc sys <Enter>
# ln -s /andrew/usr/man /andrew/usr/doc /andrew/usr/sys . <Enter>
# ln -s /andrew/usr/include/X11 /usr/include <Enter>
# ln -s /andrew/usr/bin/X11 /usr/bin/X11 <Enter>
# ln -s /andrew/usr/lib/libX11.a /usr/lib/libX11.a <Enter>
# ln -s /andrew/usr/lib/libXtk11.a /usr/lib/libXtk11.a <Enter>
# ln -s /andrew/usr/lib/X11 /usr/lib/X11 <Enter>
```

This completes the special installation steps.

This page intentionally left blank.

Building IBM/4.3 Systems with Config

ABSTRACT

This article is an updated version of an article entitled "Building Berkeley UNIX Kernels with Config," written by Samuel J. Leffler and found in the *UNIX Systems Manager Manual.* The updates include additions and changes appropriate to the IBM RT PC. The article contains six chapters and four appendices:

1. **Introduction** describes *config* and its uses.
2. **Configuration File Contents** defines each element of a configuration file.
3. **System Building Process** describes the steps that build a bootable system image.
4. **Configuration File Syntax** describes the rules for writing a configuration file.
5. **Sample Configuration File** illustrates how to configure a sample IBM RT PC.
6. **Adding New System Software** describes some of the inner workings of the configuration process.

Appendix A. Configuration File Grammar is a compressed form of the actual *yacc*(1) grammar used by *config.*

Appendix B. Rules for Defaulting System Devices describes how *config* arrives at default values for device parameters.

Appendix C. Sample Configuration File lists the complete sample configuration file developed in Chapter 5.

Appendix D. Kernel Data Structure Sizing Rules describes the rules used at compile time and boot time to size certain system data structures.

Appendix E. Network Configuration Options describes changes that can be made to customize network behavior so it conforms to local restrictions.

1. INTRODUCTION

Config is a tool used in building IBM/4.3 system images. It reads a file describing a system's tunable parameters and hardware support, and generates a collection of files used to build a copy of IBM/4.3 appropriate to that configuration. *Config* simplifies system maintenance by isolating system dependencies in a single, easy-to-understand file.

This article describes how to use *config*(8) to configure and create bootable IBM/4.3 system images.

Summary of Changes for the IBM RT PC

Significant changes to the original article are in the following sections:

- 4.1: Global Configuration Parameters
- 4.3: Device Specifications
- Appendix C: Sample Configuration File

2. CONFIGURATION FILE CONTENTS

A system configuration must include at least the following pieces of information:

- machine type
- cpu type
- system identification
- time zone
- maximum users
- location of the root file system
- available hardware

Config allows multiple system images to be generated from a single configuration description. Each system image is configured for identical hardware, but may have different locations for the root file system and, possibly, other system devices.

2.1. Machine Type

The *machine type* identifies the machine on which IBM/4.3 will operate. The machine type is used to locate certain machine-specific data files and to select rules for constructing the configuration files.

2.2. Cpu Type

The *cpu type* identifies on which cpus IBM/4.3 will operate. Specifying more than one cpu type implies IBM/4.3 should be configured to run on all the cpus specified. For those machines on which this is not possible, *config* prints a diagnostic message.

2.3. System Identification

The *system identification* is a name attached to the system, and often the machine the system is to run on. The system identification is used to create a global C "#define" that in turn is used to isolate system-dependent code in the kernel.

The system identifier "GENERIC" is given to a system that will run on any cpu of a particular machine type; it should not otherwise be used for a system identifier. A "GENERIC" system must also have the "swap generic" clause specified. (See "System Image Parameters" below.)

2.4. Time Zone

The *timezone* in which the system will run affects the information returned by the *gettimeofday*(2) system call. This value is specified as the number of hours east or west of GMT. Negative numbers indicate a value east of GMT. The timezone specification may also indicate the type of daylight savings time rules to be applied.

2.5. Maximum Number of Users

The system allocates system data structures at boot time based on the maximum number of users the system will support. This number, *maxusers*, is normally between 4 and 16, depending on the hardware and expected job mix. The rules used to calculate system data structures are discussed in Appendix D of this article.

2.6. Root File System Location

When the system boots it must know the location of the root of the file system tree. This location and the part(s) of the disk(s) to be used for paging and swapping must be specified to create a complete configuration description. You use the keyword *config* to specify these values. *Config* uses rules to calculate default locations for these items. These rules are described in Appendix B of this article.

When a generic system is configured, the root file system is left undefined until the system is booted. Therefore, the root file system need not be specified. You need only specify that the system is a generic system.

2.7. Hardware Devices

Part of the boot process is an *autoconfiguration* phase, when the system searches for those hardware devices the system builder has indicated might be present. This probing sequence requires certain pieces of information such as register addresses. A system's hardware may be configured with considerable flexibility or without any flexibility whatsoever. Most people do not configure hardware devices into the system unless:

- The devices are currently present on the machine
- The devices are due soon
- The devices are a safeguard against a hardware failure somewhere else at the site

  (Berkeley recommends configuring extra disks if an emergency requires moving one from a machine with hardware problems).

The bulk of the configuration file is usually devoted to hardware device specifications. Much of this article explains these specifications. Section 6.3 describes the autoconfiguration process for those planning to write new, or modify existing, device drivers.

2.8. Pseudo Devices

Several system facilities are configured in a manner like that used for hardware devices although they are not associated with specific hardware. These system options are configured as *pseudo-devices.* Some pseudo devices allow an optional parameter that sets the limit on the number of instances of the device that are active simultaneously.

2.9. Optional Items

In addition to the mandatory pieces of information described above, you can include various optional system facilities. For example, you can include support for monitoring disk quotas, and for tracing the performance of the virtual memory subsystem. You use the configuration file to specify any optional facilities to be configured into the system. The resultant files generated by *config* will automatically include the necessary pieces of the system.

3. SYSTEM BUILDING PROCESS

This section describes the steps necessary to build a bootable system image. We assume the system source is located in the /sys directory and that, initially, the system is being configured from source code.

Under normal circumstances there are five steps in building a system:

(1) Create a configuration file for the system.

(2) Make a directory in which to construct the system.

(3) Run *config* on the configuration file to generate the files required for compiling and loading the system image.

(4) Construct the source code dependency rules for the configured system.

(5) Compile and load the system with *make*(1).

Steps 1 and 2 are usually done only once. When a system configuration changes, you usually just run *config* on the modified configuration file, rebuild the source code dependencies, and remake the system. Sometimes, however, configuration dependencies may not be noticed. Then it is necessary to clean out the relocatable object files saved in the system's directory. This is discussed later.

3.1. Creating a Configuration File

Configuration files normally reside in the /sys/conf directory. It is easiest to construct a configuration file by copying an existing configuration file and modifying it. This distribution includes a sample configuration file.

The configuration file must have the same name as the directory in which the configured system is to be built. Further, *config* assumes this directory is located in the parent directory of the directory in which *config* is run. For example, the generic system has a configuration file named /sys/conf/GENERIC, and an accompanying directory named /sys/GENERIC. In general it is unwise to move your configuration directories out of /sys as most of the system code and the files created by *config* use pathnames of the . . / form. If you are running out of space on the file system where the configuration directories are located, there is a mechanism for sharing relocatable object files between systems. This is described later.

When building your configuration file, be sure to include the items described in Chapter 2. In particular, you must specify machine type, cpu type, time zone, system identifier, maximum users, and root device. Specifying the hardware present may take a bit of work, particularly if your hardware is configured at non-standard places (e.g. device registers located at unexpected places, or devices not supported by the system). Chapters 4, 5, and 6 of this article should be of help to you. If the devices to be configured are not described in the sample configuration file, you should check the manual pages in Volume I, Section 4, of this manual or Volume I, Section 4, of the *UNIX Programmer's Manual.* For each supported device, the manual page synopsis entry gives a sample configuration line.

Once the configuration file is complete, run it through *config* and look for any errors. Don't try to use a system that *config* has complained about; the results are unpredictable. For the most part, *config*'s error diagnostics are self explanatory; sometimes the line numbers given with the error messages are off by one.

A successful run of *config* on your configuration file will generate several files in the configuration directory. These files are:

- A file to be used by *make*(1) in compiling and loading the system
- One file for each possible system image for your machine, which describes where swapping areas, the root file system, and other miscellaneous system devices are located
- A collection of header files, one per possible device the system supports, which defines the hardware configured
- A file containing the I/O configuration tables used by the system during its *autoconfiguration* phase

Unless you have reason to doubt *config* or are curious how the system's autoconfiguration scheme works, you should never have to look at any of these files.

3.2. Constructing Source Code Dependencies

When *config* finishes generating the files needed to compile and link your system, it terminates with a message of the form:

> *Don't forget to run make depend.*

This message is a reminder that you should change to the configuration directory for the system just configured and type:

```
make depend
```

This step builds the rules used by *make* to recognize interdependencies in the system source code, and insures that any changes to system source code will result in the proper modules being recompiled the next time *make* is run.

This step is particularly important if your site makes changes to the system include files. The rules specify which source code files are dependent on which include files. Without these rules, *make* will not recognize when it must rebuild modules because a system header file has been modified.

3.3. Building the System

The makefile constructed by *config* allows a new system to be rebuilt by simply typing:

```
make image-name
```

For example, if you have named your bootable system image "vmunix", then **make vmunix** will generate a bootable image named "vmunix". You use a different system image name if the root file system location and/or swapping configuration differ from those in the bootable system image. The makefile that *config* creates has entry points for each system image defined in the configuration file. Thus, if you have configured "vmunix" to be a system with the root file system on "hd0" and "hd1vmunix" to be a system with the root file system on "hd1," then **make vmunix hd1vmunix** will generate binary images for each.

Note that the name of a bootable image is different from the system identifier. All bootable images are configured for the same system; only the information about the root file system and paging devices differ. (This is described in more detail in Chapter 4.)

The last step in the system building process is to rearrange certain commonly used symbols in the system image symbol table. The makefile generated by *config* does this automatically for you. This approach is advantageous for programs such as *ps*(1) and *vmstat*(1), that run much faster when the symbols they need are located at the front of the symbol table. Remember also that many programs expect the currently executing system to be named /vmunix. If you install a new system and name it something other than /vmunix, many programs are likely to give strange results.

If you are making a kernel that contains the debugger, the makefile target is:

```
make image_name.ws
```

For example, if the bootable image would normally be vmunix, then vmunix.ws is the kernel with the debugger and should be specified on the *make* command line. It should be installed as /vmunix, of course.

3.4. Sharing Object Modules

If you have many systems that are all built on a single machine there are at least two approaches to saving time in building system images. The best way is to have a single system which is run on all machines. This is attractive since it minimizes disk space used and time required to rebuild systems after making changes. However, it is often true that one or more systems will require a separately configured system image. This may be because of limited memory (building a system with many unused device drivers can be expensive), or configuration requirements (one machine may be a development machine where disk quotas are not needed, while another is a production machine where they are). In these cases it is possible for common systems to share relocatable object modules that are not configuration-dependent. Most of the modules in the directory /sys/sys are of this sort.

To share object modules across systems, you should first build a generic system. Then for each system, configure the system as before, but before recompiling and linking the system, type:

```
make links
```

This step searches the system for source modules that are safe to share between systems, and generates symbolic links in the current directory to the appropriate object modules in the ../GENERIC directory. This request also generates a shell script, "makelinks", which you may want to check for accuracy. The file /sys/conf/defines contains a list of symbols that Berkeley feels are safe to ignore when checking the source code for modules to be shared. Note that this list includes the definitions used to compile in the virtual memory tracing facilities and the trace point support used only rarely (even at Berkeley). It may be necessary to modify this file to reflect local needs. Note further that, as already mentioned, interdependencies that are not directly visible in the source code are not caught. This means that if you place per-system dependencies in an include file, they will not be recognized and the shared code may be selected in an unexpected fashion.

3.5. Building Profiled Systems

It is simple to configure a system that automatically collects profiling information as it operates. The profiling data can be collected with *kgmon*(8) and processed with *gprof*(1) to obtain information regarding the system's operation. Profiled systems maintain histograms of the program counter as well as the number of invocations of each routine. The *gprof*(1) command also generates a dynamic call graph of the executing system, and propagates time spent in each routine along the arcs of the call graph. (Consult the gprof documentation for more information.) The program counter sampling can be driven by the system clock or a real time clock (if you have one). The latter is highly recommended; using the system clock results in statistical anomalies, and time spent in the clock routine will not be accounted for correctly.

To configure a profiled system, the **−p** option should be supplied to *config*. A profiled system is about 5-10% larger in its text space because of the calls to count the subroutine invocations. When the system executes, the profiling data is stored in a buffer that is 1.2 times the size of the text space. The overhead for running a profiled system varies; under normal load Berkeley sees 5-25% of the system time spent in the profiling code.

Note that systems configured for profiling should not be shared as described above unless all the other shared systems are also to be profiled.

3.6. Building a System with pcc

In building a system with *pcc* rather than *hc*, be aware of two problems caused by the larger kernels *pcc* generates. The first problem is caused by the kernel debugger becoming larger than the kernel assumes an *hc*-compiled debugger will be. (The assumption is set in the file /sys/ca/rdb.h.) If the debugger exceeds that size, the make of rdb.ws will fail, generating an error message advising you to increase the value of **RDB_END**.

The second problem is that kernels having several options and pseudo devices defined may become too large to boot, when compiled by *pcc*. This will be true if the total kernel size -- the size of the kernel text plus data -- is larger than 1 Mb.

4. CONFIGURATION FILE SYNTAX

This section describes the specific rules used in writing a configuration file. A complete grammar for the input language is in Appendix A, and may be useful for resolving syntax errors.

A configuration file contains three logical pieces of information:

- parameters global to all system images
- parameters specific to each system image to be generated
- device specifications

4.1. Global Configuration Parameters

The global configuration parameters are machine type, cpu types, options, time zone, system identifier, and maximum users. Each is specified with a separate line in the configuration file.

machine *type*

The system runs on the machine type specified. No more than one machine type can appear in the configuration file. For the IBM RT PC, **ca** is the legal value.

cpu *"type"*

This system runs on the cpu type specified. More than one cpu type specification can appear in a configuration file. "**IBMRTPC**" is the legal value on the IBM RT PC. Note that the quotation marks are required.

options *optionlist*

The listed optional code is compiled into the system. Options in this list are separated by commas. Possible options are listed at the top of the generic makefile. A line of the form "options DEBUG" generates a define of the form −DDEBUG in the resulting makefile. A line of the form "options ROROOTDEV=0x0200" generates a line of the form −DROROOTDEV=0x0200. An option may be given a value by following its name with " = " and the value enclosed in (double) quotes. None of the standard options use such a value. Some useful options appear in the following table. See also Appendices D and E of the original 4.3BSD article.

Option	Effect
BLACK_ON_WHITE	
DEBUG	Compiles various debugging code into the kernel
SHOW_LOAD	Shows load average in front panel LED displays
QUOTA	Compiles code for disk quotas
INET	Compiles code for Internet protocols
NS	Compiles code for Xerox NS protocols
LF_DELAY=n	Specifies a line-feed delay for console monochrome output, making kernel debugging messages easier to read
LP_LOG=n	If n! = 0 then log console messages on the printer
GPROF	Includes kernel profiling code
SGP	Allows running an old (non-APC) processor

Option	Effect
CLOCKDEBUG	Includes clock debugging code
IODEBUG	Includes SLIH tracing IO debugging code
SYSCALLTRACE	Traces system calls and their arguments
RDB	Specifies that the kernel will interface with the debugger
AEDDEBUG	Includes debugging code for the experimental display
PGINPROF	Profiles VM usage
ROROOTDEV = 0xmmnn	Causes the root disk to be mounted read-only if it's major/minor
XWM	Must be defined to use a pseudo-device xemul
SECURE	Allows *kbdlock*(1) to lock the console keyboard
DUALCALL	Allows running a.outs with either calling sequence

Additional options associated with certain peripheral devices are listed in the Synopsis section of the manual page for the devices.

timezone *number* [**dst** [*number*]]

This specifies the timezone you are in: the number of hours your time zone is west of GMT. EST is five hours west of GMT; PST, eight. Negative numbers indicate hours east of GMT. If you specify **dst**, the system will operate under daylight savings time. An optional integer or floating point number can be included to specify a particular daylight saving time correction algorithm. The default value is 1, for the United States. Other values are: 2 (Australia), 3 (Western Europe), 4 (Middle Europe), and 5 (Eastern Europe). See *gettimeofday*(2) and *ctime*(3) for more information.

ident *name*

This system is known as *name*. The sample configuration file uses the name **SAMPLE**. On the IBM 6152 Academic System, this field must be **ATR**.

maxusers *number*

This is the maximum number of simultaneously active users expected on the system, and is used to size several system data structures. On the IBM RT PC, the minimum value for maxusers is 4.

4.2. System Image Parameters

Multiple bootable images may be specified in a single configuration file. The systems will have the same global configuration parameters and devices, but the location of the root file system and other system specific devices may be different. You specify a system image using a "config" line:

config *sysname config-clauses*

The *sysname* field is the name given to the loaded system image; almost everyone names their standard system image "vmunix". The configuration clauses are one or more specifications showing where the root file system is located, how many paging devices there are, and where they go. The device used by the system to process argument lists during *execve*(2) calls can also be specified, though in practice this is almost always done by *config* using one of its rules for selecting default locations for system devices.

A configuration clause is one of the following

root [**on**] *root-device*
swap [**on**] *swap-device* [**and** *swap-device*]
dumps [**on**] *dump-device*
args [**on**] *arg-device*

(The "on" is optional.) Multiple configuration clauses are separated by white space; *config* allows specifications to be continued across multiple lines by beginning the continuation line with a tab character. The "root" clause specifies where the root file system is located, the "swap" clause specifies swapping and paging area(s), the "dumps" clause can be used to force system dumps to a particular device, and the "args" clause can be used to force argument list processing for *execve* to a particular disk.

The device names supplied in the clauses may be fully specified as a device, unit, and file system partition; or underspecified, in which case *config* will use its own rules to select default unit numbers and file system partitions. The defaulting rules are a bit complicated, as they are dependent on the overall system configuration. For example, the swap area need not be specified at all if the root device is specified; the swap area is placed in the "b" partition of the disk where the root file system resides. Appendix B contains a complete list of the defaulting rules used in selecting system configuration devices.

The device names are translated to the appropriate major and minor device numbers on a per-machine basis. A file, /sys/conf/devices.machine (where "machine" is the machine type specified in the configuration file), is used to map a device name to its major block device number. The minor device number is calculated using the standard disk partitioning rules: on unit 0, partition "a" is minor device 0, partition "b" is minor device 1, and so on; for units other than 0, add 8 times the unit number to get the minor device.

If the default mapping of device name to major/minor device number is incorrect for your configuration, it can be replaced by an explicit specification of the major/minor device. You do this by substituting

> **major** *x* **minor** *y*

where the device name would normally be found.

Normally, the areas configured for swap space are sized by the system at boot time. If a non-standard partition size is to be used for one or more swap areas, you can add a "size" specification to the device name for the swap area. For example,

> **config vmunix root on hd0 swap on hd0b size 1200**

would force swapping to be done in partition "b" of "hd0" and the swap partition size would be set to 1200 sectors. A swap area sized larger than the associated disk partition is trimmed to the partition size.

To create a generic configuration, only the clause "swap generic" should be specified; any extra clauses will cause an error. The "swap generic" clause can only be specified with the system identifier "GENERIC" or if options GENERIC is specified.

4.3. Device Specifications

You must specify to *config* each device attached to a machine, so that the generated system will know to probe for it during the autoconfiguration process at boot time. Hardware specified in the configuration file need not actually be present on the machine where the generated system is run. Only the hardware actually found at boot time will be used by the system.

The specification of hardware devices in the configuration file parallels the interconnection hierarchy of the machine to be configured. A configuration file must identify what adapters are present. A device description can provide a complete definition of the possible configuration parameters, or can leave certain parameters undefined; at boot time the system probes for these missing values. The latter approach allows a single device configuration list to match many possible physical configurations. This approach, termed *wildcarding*, provides more flexibility in the physical configuration of a system. If a disk must be moved for some reason, the system will still locate it at the alternate location.

For the IBM RT PC, a device specification takes one of the following forms:

controller *device-name device-info*
device *device-name device-info*
disk *device-name device-info*
tape *device-name device-info*

A *controller* is an adapter that controls one or more disks or tapes or uses DMA; everything else is a *device*.

The *device-name* is one of the standard device names, concatenated with the *logical* unit number assigned to the device. (The *logical* unit number may differ from the *physical* unit number shown on the front of a device like a disk; the *logical* unit number refers to the IBM/4.3 device, not the physical unit number). Standard device names for the IBM RT PC are documented in Volume I,, Section 4, of this manual.

The *device-info* clause specifies how the hardware is connected in the interconnection hierarchy. The beginning of the hierarchy is defined as follows:

```
controller          iocc0          at nexus ?
```

The remaining legal interconnections are:

- A controller can be connected to another controller
- A disk or tape is always attached to a controller
- Devices are always attached to controllers

On the IBM RT PC, the controller specification takes the form:

```
controller          xxcn           at iocc0 csr addr priority irq
```

where

"xx"	is the two- or three-letter name of the device
"n"	is the controller number (0, 1, 2, 3, etc.)
"addr"	is the controller adapter base address
"irq"	is the PC/AT IRQ level for the adapter

For example, the line:

```
controller          hdc0           at iocc0 csr 0xf00001f0 priority 14
```

specifies that the hard disk controller (adapter) is at iocc0 (required); its csr address is 0xf00001f0; and its device interrupt request level is 14. Note that interrupt service routines are not specified here. Each adapter has only one interrupt service routine, specified in the iocc_driver structure within the device driver.

For a slave device on the IBM RT PC, the specification takes one of the following forms:

```
disk                xxn    at xxcy drive z
tape                xxn    at xxcy drive z
```

where

"xx"	is the two- or three-letter name of the device
"n"	is the unit number
"y"	is the controller number
"z"	is the slave unit number

For example, the following is a specification for a hard disk:

```
disk                hd0            at hdc0 drive 0
```

For a non-slave device on the IBM RT PC, the specification takes the form:

```
device              xxn    at iocc0 csr addr priority irq
```

where

"xx" is the two- or three-letter name of the device
"n" is the adapter unit number (0, 1, 2, 3, etc.)
"addr" is the device adapter base address
"irq" is the PC/AT IRQ level for the adapter

For example, the following is a specification for the four-line serial card:

device asy0 **at** iocc0 csr 0xf0001230 priority 9

Any piece of hardware that can be connected to a specific controller can also be wildcarded across multiple controllers, by specifying the controller number as "?".

The final piece of information needed by the system to configure devices is some indication of where or how a device will interrupt. On the IBM RT PC, interrupt routines are specified in the driver rather than in the configuration file.

Certain device drivers require extra information passed to them at boot time to tailor their operation to the actual hardware present. The drivers for the terminal multiplexors need to know which lines are attached to modem lines so that no one will be allowed to use them unless a connection is present. Therefore, one last parameter may be specified for a device, a *flags* field. It has the syntax

flags *number*

and is usually placed after the *csr* specification. The *number* is passed directly to the associated driver. You should consult the manual pages in Volume I, Section 4, of this manual to determine how each driver uses this value (if at all). Communications interface drivers commonly use the flags to indicate whether modem control signals are in use.

The exact syntax for each specific device is given in the Synopsis section of its manual page in Volume I, Section 4, of this manual.

4.4. Pseudo-Devices

Some drivers and software subsystems are treated like device drivers without any associated hardware. To include any of these pieces, a "pseudo-device" specification must be used. A specification for a pseudo-device takes the form

pseudo-device *device-name* [*howmany*]

Examples of pseudo-devices are **bk**, the Berknet line discipline; **pty**, the pseudo terminal driver (where the optional *howmany* value indicates the number of pseudo terminals to configure, with 32 as the default); and **inet**, the DARPA Internet protocols (one must also specify INET in the "options" statement). Other pseudo-devices for the network include **loop**, the software loopback interface; **imp** (required when a CSS or ACC imp is configured); and **ether** (used by the Address Resolution Protocol on 10 Mb/sec Ethernets and IBM Token-Ring Networks). More information on configuring each of these can also be found in Section 4 of the *UNIX Programmer's Manual.*

Other pseudo-devices specific to the IBM RT PC are the following:

ms for the mouse

mono for the monochrome display

aed for the IBM Academic Information Systems experimental display

apasixteen for the IBM 6155 Extended Monochrome Graphics Display

apaeightc for the IBM 6154 Advanced Color Graphics Display

apaeight for the IBM 6153 Advanced Monochrome Graphics Display

ega for the IBM 5154 Enhanced Color Graphics Display
xemul for the X support
ap for the IBM 3812 Pageprinter.

5. SAMPLE CONFIGURATION FILES

This chapter illustrates how to configure a sample IBM 6150 Model 25 (floor model) that will run in a networking environment.

5.1. Floor Model

The following table lists the hardware to be configured.

Item	Connection	Name	Reference
cpu	"IBMRTPC"		
controller	nexus ?	iocc0	
U-B Ethernet card	iocc0	un0	un(4)
disk controller	iocc0	hdc0	hd(4)
disk controller	iocc0	hdc1	
disk controller	iocc0	scc0	sc(4)
disk controller	iocc0	scc1	
diskette controller	iocc0	fdc0	fd(4)
disk	scc0	sc0	sc(4)
disk	scc0	sc1	
disk	scc0	sc2	
disk	scc0	sc3	
disk	scc0	sc4	
disk	scc0	sc5	
disk	scc0	sc6	
disk	scc1	sc7	
disk	scc1	sc8	
disk	scc1	sc9	
disk	scc1	sc10	
disk	scc1	sc11	
disk	scc1	sc12	
disk	scc1	sc13	
disk	hdc0	hd0	hd(4)
disk	hdc0	hd1	
disk	hdc?	hd2	
async controller	iocc0	asy0	asy(4)
async controller	iocc0	asy1	asy(4)
async controller	iocc0	asy4	asy(4)
async controller	iocc0	psp0	psp(4)
enhanced color graphics display	iocc0	ega	ibm5154(4)
experimental display		aed	ibmaed(4)
advanced monochrome display		apaeight	ibm6153(4)
extended monochrome display		apa1sixteen	ibm6155(4)
monochrome display		mono	ibm5151(4)
printer controller	iocc0	lp0	lp(4)
diskette	fdc0	fd0	fd(4)
tape controller	iocc0	stc0	st(4)
tape	stc0	st0	st(4)
token ring	iocc0	lanc0	lan(4)
token ring adapter	lanc0	lan0	lan(4)

The following steps illustrate how to build the configuration file for this system.

(1) Fill in the global configuration parameters.

The machine is an IBM RT PC, with a *machine type* of **ca**. This system is to run only on this one processor; the *cpu type* is **"IBMRTPC"**. We will use the INET option, because we plan to use the DARPA standard Internet protocols. The system identifier is **SAMPLE**. The maximum users we plan to support is about 16. Thus the beginning of the configuration file looks like this:

```
#
# Sample Configuration File for the IBM RT PC
#
machine         ca
cpu             "IBMRTPC"
ident           SAMPLE
timezone        8 dst
maxusers        16
options         INET
```

(2) Add the specification for a single system image.

Our standard system has the root on "hd0" and swapping on both "hd0" and "hd1".

```
config          vmunix          root on hd0 swap on hd0 and hd1
```

(3) Specify the hardware.

Transcribe the information from the preceding table:

```
controller  iocc0   at nexus ?
device      un0     at iocc0 csr 0xf4080000 priority 6
controller  hdc0    at iocc0 csr 0xf00001f0 priority ?
controller  hdc1    at iocc0 csr 0xf0000170 priority ?
controller  fdc0    at iocc0 csr 0xf00003f2 priority 6
controller  scc0    at iocc0 csr 0xf0000d52 priority 11
controller  scc1    at iocc0 csr 0xf0000952 priority 12
disk        sc0     at scc0 drive 0
disk        sc1     at scc0 drive 1
disk        sc2     at scc0 drive 2
disk        sc3     at scc0 drive 3
disk        sc4     at scc0 drive 4
disk        sc5     at scc0 drive 5
disk        sc6     at scc0 drive 6
disk        sc7     at scc1 drive 0
disk        sc8     at scc1 drive 1
disk        sc9     at scc1 drive 2
disk        sc10    at scc1 drive3
disk        sc11    at scc1 drive 4
disk        sc12    at scc1 drive 5
disk        sc13    at scc1 drive 6
disk        hd0     at hdc0 drive 0
disk        hd1     at hdc0 drive 1
disk        hd2     at hdc? drive ?
```

```
device      asy0    at iocc0 csr 0xf0001230 priority 9 flags 0x0f
device      asy1    at iocc0 csr 0xf0002230 priority 10
device      asy4    at iocc0 csr 0xf00003f8 priority 4
device      lp0     at iocc0 csr 0xf00003bc priority 7
device      fd0     at fdc0 drive 0
controller  stc0    at iocc0 csr 0xf00001e8 priority 12
tape        st0     at stc0 drive 0
device      psp0    at iocc0 csr 0xf0008000 priority 2 flags 0x03
controller  lanc0   at iocc0 csr 0xf00001c0 priority 12
device      lan0    at lanc0 drive 0
```

(4) Add the required pseudo-devices:

```
pseudo-device       mono
pseudo-device       pty
pseudo-device       loop
pseudo-device       inet
pseudo-device       ether
pseudo-device       ms
pseudo-device       aed
pseudo-device       apaeight
pseudo-device       apasixteen
pseudo-device       ega
```

This completes the sample configuration file. It appears in its entirety in Appendix C.

5.2. Miscellaneous Comments

Note that the sample system does not use either disk quotas or 4.1BSD compatibility mode. To use these optional facilities or others, Berkeley recommends cleaning out your current configuration, reconfiguring the system, then recompiling and relinking the system image(s). You could avoid this, of course, by figuring out which relocatable object files are affected by the reconfiguration, and then reconfiguring and recompiling only the affected files. Use this technique carefully.

6. ADDING NEW SYSTEM SOFTWARE

This chapter is not for the novice. It has four sections:

- how to modify system code
- how to add a device driver to IBM/4.3
- how device drivers are autoconfigured under IBM/4.3
- how to add non-standard system facilities to IBM/4.3

6.1. How to Modify System Code

To make site-specific modifications to the system, it is best to bracket them with

```
#ifdef SITENAME
...
#endif
```

This allows your source to be distributed to others easily, and also simplifies *diff*(1) listings. If you choose not to use a source code control system (e.g. SCCS, RCS), and perhaps even if you do, it is recommended that you save the old code with something of the form:

```
#ifndef SITENAME
...
#endif
```

Berkeley tries to isolate site-dependent code in individual files that may be configured with pseudo-device specifications.

Identify machine-specific code with "#ifdef ibm032".

6.2. How to Add Device Drivers to IBM/4.3

The I/O system and *config* have been designed so you can add new device support easily. The system source directories are organized as follows:

/sys/h	machine independent include files
/sys/sys	machine independent system source files
/sys/conf	site configuration files and basic templates
/sys/net	network independent, but network related code
/sys/netimp	IMP support code
/sys/netinet	DARPA Internet code
/sys/netns	XEROX NS code
/sys/ca	IBM RT PC specific mainline code
/sys/caif	IBM RT PC network interface code
/sys/caio	IBM RT PC device drivers and related code
/sys/cacons	IBM RT PC console device drivers and related code

Existing block and character device drivers for the IBM RT PC reside in "/sys/ca" and "/sys/caio". Network interface drivers reside in "/sys/caif". Any new device drivers should be placed in the appropriate source code directory and named so as not to conflict with existing devices. Normally, definitions for things like device registers are placed in a separate file in the same directory. For example, "psp.c" is the name of the "psp" device driver, and "pspreg.h" is the name of its associated include file.

Once the source for the device driver has been placed in a directory, you should modify /sys/conf/files.machine and, possibly, /sys/conf/devices.machine. The two "files" files in the conf directory contain a line for each source or binary-only file in the system.

Machine-independent files are located in /sys/conf/files, while machine-specific files are in /sys/conf/files.machine. The devices.machine file is used to map device names to major block device numbers. If the device driver being added provides support for a new disk, you will want to modify this file. (The format is obvious.) Note that the *.machine* suffix refers to the specific machine on which you're working. On the IBM RT PC, *.machine* becomes *.ca.*

The format of these two files has grown somewhat complex over time. Entries are normally of the form:

caio/foo.c **optional** foo **device-driver**

where the keyword *optional* indicates that to compile the "foo" driver into the system, it must be specified in the configuration file. If instead the driver is specified as *standard,* the file will be loaded no matter what configuration is requested. This is not normally done with device drivers.

Aside from including the driver in the appropriate "files" file, it must also be added to the device configuration tables. These are located in the /sys/ca/conf.c file. If you don't understand what to add to this file, you should study an entry for an existing driver. Remember that the position in the block device table specifies what the major block device driver number is; this number is needed in the "devices.machine" files if the device is a disk.

With the configuration information in place, your configuration file appropriately modified, and a system reconfigured and rebooted, you should incorporate the shell commands needed to install the special files in the file system to the /dev/MAKEDEV or /dev/MAKEDEV.local file. This is discussed in the article "Operating Academic Operating System 4.3."

6.3. Adding Non-Standard System Facilities

This section describes the work needed to augment *config*'s data base files for non-standard system facilities.

For *config*, non-standard facilities fall into two categories, those for kernel-profiling and those that are configuration-dependent. Files used for kernel profiling appear in the "files" files with a *profiling-routine* keyword. For example, the current profiling subroutines are found in a separate file with the following entry:

sys/subr_mcount.c **optional profiling-routine**

The *profiling-routine* keyword prohibits *config* from compiling the source file with the **−pg** option.

The keyword for the second category is *config-dependent.* This makes *config* compile the appropriate module with the global configuration parameters. This allows certain modules such as *machdep.c* to size system data structures based on the maximum users configured for the system.

6.4. Autoconfiguration on the IBM RT PC

IBM/4.3 requires all device drivers to conform to a set of rules that allow the system to:

(1) Support system configuration at boot time, and

(2) Manage resources so as not to crash when devices request unavailable resources.

The IBM RT PC I/O control channel (IOCC) uses a set of translation control word (TCW) registers to convert from the PC/AT I/O bus address space into the IBM RT PC address space when using direct memory access (DMA). There is a structure of type *struct iocc_hd* in the system per DMA channel used to manage these resources. This structure also contains a linked list where devices waiting for resources to complete DMA activity have requests waiting.

There are three central structures used to write drivers for controllers:

struct iocc_ctlr -- the controller structure

struct iocc_driver -- the driver structure

struct iocc_device -- the device structure

These are defined in the . . . /caio/ioccvar.h file.

The elements are analogous to the VAX structures, except for the following:

ic_irq	is the irq level specified as 'priority' for a controller.
iod_irq	is the irq level specified as 'priority' for a device.
idr_intr	specifies the interrupt service routine for this driver.
idr_flags	are any necessary driver-specific flags.
idr_csr	is the offset to a read/write register that can be interrogated for the existence of the device. Values at addr[csr + 0] and addr[csr + 1] will be tested for a non-existent device (compared to the contents of a "standard" non-existent device).
ic_dmachannel	is the channel number for dma transfers.
ic_dmaflags	are flags based to dma code (see dmavar.L).
ic_dmabuf	is the pointer to the buffer structure for the dma code to transfer.
ic_dmaforw	is the forward pointer chain used by dma code internally.

Devices that do not do DMA I/O can often use only two of these structures (iocc_driver and iocc_device). Each *controller* specified in the config file has an associated line in *struct iocc_ctlr iocc_cinit[ ]* and each device or slave has an entry in *struct iocc_device iocc_dinit[ ]* generated automatically in ioconf.c in the config directory. The *iocc_ctlr* and *iocc_device* structures are in one-to-one correspondence with the definitions of controllers and devices in the system configuration. Each driver has a *struct iocc_driver* structure specifying an internal interface to the rest of the system.

The specification:

```
controller hdc0 at iocc0 csr 0xf00001f0
```

would cause a *struct iocc_ctlr* to be declared and initialized in the file *ioconf.c* for the system configured from this description. Similarly specifying:

```
disk hd0 at hdc0 drive 0
```

would declare a related *iocc_device* in the same file. The *hd.c* driver which implements this driver specifies in declarations:

```
int hdprobe(), hdslave(), hdattach(), hdint();
int hdwstart, hdwatch();                                    /* watch routine

/* hddinfo contains pointers to the slaves (drives) */
struct iocc_device *hddinfo[NHD];

struct iocc_ctlr *hdminfo[NHDC];
struct iocc_driver hdcdriver = {
                hdprobe, hdslave, hdattach,
/*    dgo      addr   dname   dinfo   mname   minfo   intr   csr     */
               0, hdstd, "hd", hddinfo, "hdc", hdminfo, hdint, 2
```

which initializes the *iocc_driver* structure. The driver will support some number of controllers named *hdc0, hdc1*, etc, and some number of drives named *hd0, hd1*, etc. where the drives may be on any of the controllers (that is, there is a single linear name space for devices, separate from the controllers.)

We now explain the fields in the various structures. It may help to look at a copy of *caio/ioccreg.h*, *caio/ioccvar.h* and drivers such as *hd.c* and *asy.c* while reading the descriptions of the various structure fields.

6.4.1.1. iocc_driver structure

One of these structures exists per driver. It is initialized in the driver and contains functions used by the configuration program and by the DMA resource routines. The fields of the structure are:

idr_probe

The probe routine is given the adapter base address and should return one of the following values:

PROBE_BAD (0) The device is bad, didn't really exist, etc.

PROBE_NOINT (1) The device is there, but either cannot interrupt easily or the driver isn't smart enough to do so. Use the irq from the configuration information.

PROBE_OK (2) The device is there, is healthy, and should have interrupted. If it did not actually interrupt, then a message will be printed and the device ignored.

The PROBE_DELAY(n) macro can be used when waiting for an interrupt to happen in the probe routine. It behaves exactly like DELAY(n), which delays for *n* microseconds, but will return as soon as an interrupt has happened.

idr_slave

This routine is called with a *iocc_device* structure (yet to be described) and the address of the device controller. It should determine whether a particular slave device of a controller is present, returning 1 if it is and 0 if it is not.

idr_attach

The attach routine is called after the autoconfigure code and the driver concur that a peripheral exists attached to a controller. This is the routine where internal driver state about the peripheral can be initialized.

The attach routine performs a number of functions. The first time any drive is attached to the controller it starts the timeout routine which watches the disk drives to make sure that interrupts aren't lost. It also initializes, for devices which have been assigned *iostat* numbers (when iod->iod_dk >= 0), the transfer rate of the device in the array *dk_mspw*, the fraction of a second it takes to transfer a 16-bit word. It increments the count of the number of devices on this controller, so that search commands can later be avoided if the count is exactly 1.

idr_dgo

Is the routine which is called by the DMA resource management routines when an operation is ready to be started (because the required resources have been allocated). It is not used by programmed I/O routines, such as hd.c.

idr_addr

Are the conventional addresses for the device control registers. This information is used by the system to look for instances of the device supported by the driver. When the system probes for the device it first checks for a control-status register located at the address indicated in the configuration file (if supplied), then uses the list of conventional addresses pointed to be *idr_addr*.

idr_dname

Is the name of a *device* supported by this controller; thus the disks on a fixed disk controller are called *hd0*, *hd1*, etc. That is because this field contains *hd*.

idr_dinfo

Is an array of back pointers to the *iocc_device* structures for each device attached to the controller. Each driver defines a set of controllers and a set of devices. The device address space is always one-dimensional, so that the presence of extra controllers may be masked away (e.g. by pattern matching) to take advantage of hardware redundancy. This field is filled in by the configuration program, and used by the driver.

idr_mname

The name of a controller, e.g. *hdc* for the *hd.c* driver. The first controller is called *hdc0*, etc.

idr_minfo

The backpointer array to the structures for the controllers.

idr_intr

The interrupt routine is called after the device receives an interrupt. It returns 0 if the interrupt really was for that device. Otherwise, it returns 1 so that multiple devices may share the same interrupt level.

idr_chanrelse

The routine that signals a device driver to release a channel.

idr_csr

The offset from idr_addr that PROBE uses to see if a device exists; idr_csr must be the offset to a read-write location that is safe for PROBE to read-write during autoconfig.

idr_flags

Are for any necessary driver-specific flags. In IBM/4.3, one such flag is **EARLY_INT**, which causes the interrupt handler to be called on the initial (probe) interrupt. The function int_8259() will do the shared interrupt reset.

6.4.1.2. iocc_ctlr structure

One of these structures exists per controller. The fields link the controller to its adapter and contain the state information about the devices on the controller. The fields are:

ic_driver

A pointer to the *struct iocc_driver* for this driver, which has fields as defined above.

ic_ctlr
The controller number for this controller, e.g. the 0 in *hdc0*.

ic_alive
Set to 1 if the controller is considered alive; currently, always set for any structure encountered during normal operation. That is, the driver will have a handle on a *iocc_ctlr* structure only if the configuration routines set this field to a 1 and entered it into the driver tables.

ic_tab
This buffer structure is a place where the driver hangs the device structures which are ready to transfer. Each driver allocates a buf structure for each device (e.g. *hddtab* in the *hd.c* driver) for this purpose. You can think of this structure as a device-control-block, and the buf structures linked to it as the unit-control-blocks. The code for dealing with this structure is stylized; see the *fd.c* or *hd.c* driver for the details. If the *dmago* routine is to be used, the structure attached to this *buf* structure must be:

- A chain of *buf* structures for each waiting device on this controller.
- On each waiting *buf* structure another *buf* structure which is the one containing the parameters of the I/O operation.

ic_addr
Address of the device in I/O space.

ic_irq
The interrupt request level for this device.

ic_channel
The device's dma channel (dma devices only).

ic_dmabuf
Buffer describing the DMA transfer.

ic_party
The device's dma party type. It must be set to DM_THIRDPARTY (dma devices only).

ic_dmaflags
Transfer flags for dma (dma devices only).

6.4.1.3. iocc_device structure

One of these structures exist for each device. Devices which are not attached to controllers or which perform no DMA I/O may have only a device structure. Thus *asy* and *lp* devices have only *iocc_device* structures. The fields are:

iod_driver
A pointer to the *struct iocc_driver* structure for this device type.

iod_unit
The unit number of this device, e.g. 0 in **hd0**, or 1 in **asy1**.

iod_ctlr
The number of the controller on which this device is attached, or −1 if this device is not on a controller.

iod_slave
The slave number of this device on the controller which it is attached to, or −1 if the device is not a slave. Thus a disk which was unit 0 on a disk adapter would have *iod_slave* 0. It might or might not be *hd0*, that depends on the system configuration specification.

iod_irq
The interrupt request level for this device.

iod_addr
The control-status register address of this device.

iod_dk
The iostat number assigned to this device. Numbers are assigned to disks only, and are small positive integers which index the various *dk_** arrays in <*sys/dk.h*>.

iod_flags
The optional "flags *xxx*" parameter from the configuration specification was copied to this field, to be interpreted by the driver. If *flags* was not specified, then this field will contain a 0.

iod_alive
The device is really there. Presently set to 1 when a device is determined to be alive, and left 1.

iod_type
The device type, to be used by the driver internally.

iod_physaddr
The physical memory address of the device control-status register. This is used in the device dump routines typically.

iod_mi
A *struct iocc_ctlr* pointer to the controller (if any) on which this device resides.

iod_hd
A *struct iocc_hd* pointer to the DMA channel this device uses.

6.4.1.4. DMA Resource Management Routines

DMA drivers are supported by a collection of utility routines which manage DMA resources. If a driver attempts to bypass the DMA routines, other drivers may not operate properly. The major routines are: *dma_setup* to allocate DMA resources, *dma_done* to release previously allocated resources, and *dma_go* to initiate DMA.

For more information, consult the "DMA Reference Manual" article.

6.4.2. Autoconfiguration Requirements

Basically all you have to do is write a *idr_probe* and a *idr_attach* routine for the controller. It suffices to have a *idr_probe* routine which just returns PROBE_NOINT, and a *idr_attach* routine which does nothing. Making the device fully configurable requires, of course, more work, but is worth it if you expect the device to be in common usage and want to share it with others.

If you managed to create all the needed hooks, then make sure you include the necessary header files; the ones included by *caio/lp.c* are nearly minimal. Order is important here, don't be surprised at undefined structure complaints if you order the includes wrongly. Finally if you get the device configured in, you can try bootstrapping and see if configuration messages print out about your device. It is a good idea to have some messages in the probe routine so that you can see that you are getting called and what is going on. If you do not get called, then you probably have the control-status register address wrong in your system configuration. The autoconfigure code notices that the device doesn't exist in this case and you will never get called.

Assuming that your probe routine works and you manage to generate an interrupt, then you are basically back to where you would have been under older versions of UNIX operating systems. Just be sure to use the *iod_ctlr* field of the *iocc_device* structures to address the device; compiling in funny constants will make your driver less portable.

APPENDIX A. CONFIGURATION FILE GRAMMAR

The following grammar is a compressed form of the actual *yacc*(1) grammar used by *config* to parse configuration files. Terminal symbols are shown all in upper case, literals are emboldened; optional clauses are enclosed in brackets "[" and "]"; zero or more instantiations are denoted with "*".

Configuration ::= [Spec ;]*

Spec ::= Config_spec
| Device_spec
| **trace**
| /* lambda */

/* configuration specifications */

Config_spec ::= **machine** ID
| **cpu** ID
| **options** Opt_list
| **ident** ID
| System_spec
| **timezone** [−] NUMBER [**dst** [NUMBER]]
| **timezone** [−] FPNUMBER [**dst** [NUMBER]]
| **maxusers** NUMBER

/* system configuration specifications */

System_spec ::= **config** ID System_parameter [System_parameter]*

System_parameter ::= swap_spec | root_spec | dump_spec | arg_spec

swap_spec ::= **swap** [**on**] swap_dev [**and** swap_dev]*

swap_dev ::= dev_spec [**size** NUMBER]

root_spec ::= **root** [**on**] dev_spec

dump_spec ::= **dumps** [**on**] dev_spec

arg_spec ::= **args** [**on**] dev_spec

dev_spec ::= dev_name | major_minor

major_minor ::= **major** NUMBER **minor** NUMBER

dev_name ::= ID [NUMBER [ID]]

/* option specifications */

Opt_list ::= Option [, Option]*

Option ::= ID [= Opt_value]

Opt_value ::= ID | NUMBER

```
/* device specifications */

Device_spec ::= device Dev_name Dev_info Int_spec
                | master Dev_name Dev_info
                | disk Dev_name Dev_info
                | tape Dev_name Dev_info
                | controller Dev_name Dev_info [ Int_spec ]
                | pseudo-device Dev [ NUMBER ]

Dev_name ::=  Dev NUMBER

Dev ::=  uba | mba | ID

Dev_info ::=  Con_info [ Info ]*

Con_info ::=  at Dev NUMBER
                | at nexus NUMBER

Info ::=  csr NUMBER
                | drive NUMBER
                | slave NUMBER
                | flags NUMBER

Int_spec ::=  vector ID [ ID ]*
                | priority NUMBER
```

Lexical Conventions

The terminal symbols are loosely defined as:

ID

One or more alphabetics, either upper or lower case, and underscore.

NUMBER

Similar to the C language specification for an integer number. That is, a leading "0x" indicates a hexadecimal value, a leading "0" indicates an octal value, otherwise the number is expected to be a decimal value. Hexadecimal numbers may use either upper or lower case alphabetics.

FPNUMBER

A floating point number without exponent. That is a number of the form "nnn.ddd", where the fractional component is optional.

In special instances a question mark (?), can be substituted for a "NUMBER" token. This is used for wildcarding in device interconnection specifications.

Comments in configuration files being with a "#" character; the remainder of the line is discarded.

A specification is interpreted as a continuation of the previous line if the first character of the line is tab.

APPENDIX B. RULES FOR DEFAULTING SYSTEM DEVICES

When *config* processes a "config" rule that does not fully specify the location of the root file system, paging area(s), device for system dumps, and device for argument list processing it applies a set of rules to define those values left unspecified. The following list of rules are used in defaulting system devices.

(1) If a root device is not specified, the swap specification must indicate a "generic" system is to be built.

(2) If the root device does not specify a unit number, it defaults to unit 0.

(3) If the root device does not include a partition specification, it defaults to the "a" partition.

(4) If no swap area is specified, it defaults to the "b" partition of the root device.

(5) If no device is specified for processing argument lists, the first swap partition is selected.

(6) If no device is chosen for system dumps, the first swap partition is selected (see below to find out where dumps are placed within the partition).

The following table summarizes the default partitions selected when a device specification is incomplete (e.g. "hd0").

Type	Partition
root	"a"
swap	"b"
args	"b"
dumps	"b"

Multiple swap/paging areas

When multiple swap partitions are specified, the system treats the first specified as a "primary" swap area that is always used. The remaining partitions are then interleaved into the paging system at the time a *swapon*(2) system call is made. This is normally done at boot time with a call to *swapon*(8) from the /etc/rc file.

System dumps

System dumps are automatically taken after a system crash, provided the device driver for the "dumps" device supports this. The dump contains the contents of memory, but not the swap areas. Normally the dump device is a disk; the information is copied to a location near the back of the partition. The dump is placed in the back of the partition because the primary swap and dump device are commonly the same device and this allows the system to be rebooted without immediately overwriting the saved information. When a dump has occurred, the system variable *dumpsize* is set to a non-zero value indicating the size (in bytes) of the dump. The *savecore*(8) program then copies the information from the dump partition to a file in a "crash" directory and also makes a copy of the system that was running at the time of the crash (usually "/vmunix"). The offset to the system dump is defined in the system variable *dumplo* (a sector offset from the front of the dump partition). The *savecore* program operates by reading the contents of *dumplo*, *dumpdev*, and *dumpmagic* from /dev/kmem, then comparing the value of *dumpmagic* read from /dev/kmem to that located in corresponding location in the dump area of the dump partition. If a match is found, *savecore* assumes a crash occurred and reads *dumpsize* from the dump area of the dump partition. This value is then used in copying the system dump. Refer to *savecore*(8) for more information about its operation.

The value *dumplo* is calculated to be

> *dumpdev-size* − memsize

where *dumpdev-size* is the size of the disk partition where system dumps are to be placed, and *memsize* is the size of physical memory. If the disk partition is not large enough to hold a 4-megabyte dump, *dumplo* is set to 0 (the front of the partition).

APPENDIX C. SAMPLE CONFIGURATION FILE

The following sample configuration file is developed in Chapter 5; it is included here for completeness.

```
#
# Sample Configuration File for the IBM RT PC
#
machine         ca
cpu             "IBMRTPC"
ident           SAMPLE
timezone        8 dst
maxusers        16
options         INET

config          vmunix      root on hd0 swap on hd0 and hd1

controller      iocc0       at nexus ?
device          un0         at iocc0 csr 0xf4080000  priority 6
controller      hdc0        at iocc0 csr 0xf00001f0 priority ?
controller      hdc1        at iocc0 csr 0xf0000170 priority ?
controller      scc0        at iocc0 csr 0xf0000d52 priority 11
controller      scc1        at iocc0 csr 0xf0000952 priority 12
controller      fdc0        at iocc0 csr 0xf00003f2 priority 6
controller      stc0        at iocc0 csr 0xf00001e8 priority 12
controller      lanc0       at iocc0 csr 0xf00001c0 priority 12
disk            hd0         at hdc0 drive 0
disk            hd1         at hdc0 drive 1
disk            sc0         at scc0 drive 0
disk            sc1         at scc0 drive 1
disk            sc2         at scc0 drive 2
disk            sc3         at scc0 drive 3
disk            sc4         at scc0 drive 4
disk            sc5         at scc0 drive 5
disk            sc6         at scc0 drive 6
disk            sc7         at scc1 drive 0
disk            sc8         at scc1 drive 1
disk            sc9         at scc1 drive 2
disk            sc1         at scc1 drive3
disk            sc1         at scc1 drive 4
disk            sc12        at scc1 drive 5
disk            sc13        at scc1 drive 6
disk            hd2         at hdc? drive ?
device          asy0        at iocc0 csr 0xf0001230 priority 9
device          asy1        at iocc0 csr 0xf0002230  priority 10
device          asy4        at iocc0 csr 0xf00003f8  priority 4
device          lp0         at iocc0 csr 0xf00003bc  priority 7
device          fd0         at fdc0 drive 0
tape            st0         at stc0 drive 0
device          lan0        at lanc0 drive 0
pseudo-device   mono
pseudo-device   pty
pseudo-device   loop
```

```
pseudo-device        inet
pseudo-device        ether
pseudo-device        ms
pseudo-device        aed
pseudo-device        apaeight
pseudo-device        apasixteen
pseudo-device        ega
```

APPENDIX D. KERNEL DATA STRUCTURE SIZING RULES

Certain system data structures are sized at compile time according to the maximum simultaneous users expected, while others are calculated at boot time based on the physical resources present. This appendix lists both sets of rules and also includes some hints on changing built-in limitations on certain data structures.

Compile time rules

The file /sys/conf/param.c contains the definitions of almost all data structures sized at compile time. This file is copied into the directory of each configured system to allow configuration-dependent rules and values to be maintained. The rules implied by its contents are summarized below (here MAXUSERS refers to the value defined in the configuration file in the "maxusers" rule).

nproc

The maximum number of processes which may be running at any time. It is defined to be 20 + 8 * MAXUSERS and referred to in other calculations as NPROC.

ntext

The maximum number of active shared text segments. The constant is intended to allow for netwrok servers and common commands that remain in the table. It is defined as 36 + MAXUSERS.

ninode

The maximum number of files in the file system that may be active at any time. This includes files in use by users, as well as directory files being read or written by the system and files associated with bound sockets in the IBM/4.3 ipc domain. This number is defined as (NPROC + 16 + MAXUSERS) + 32.

nfile

The number of "file table" structures. One file table structure is used for each open, unshared, file descriptor. Multiple file descriptors may reference a single file table entry when they are created through a *dup* call, or as the result of a *fork*. This value is defined to be

16 * (NPROC + 16 + MAXUSERS) / 10 + 32

ncallout

The number of "callout" structures. One callout structure is used per internal system event handled with a timeout. Timeouts are used for terminal delays, watchdog routines in device drivers, protocol timeout processing, etc. This number is defined as 16 + NPROC.

nclist

The number of "c-list" structures. C-list structures are used in terminal I/O, and currently each holds 60 characters. Their number is defined as 60 + 12 * MAXUSERS.

nmbclusters

The maximum number of pages that may be allocated by the network. This is defined as 256 (a quarter megabyte of memory) in /sys/h/mbuf.h. In practice, the network rarely uses this much memory. It starts off by allocating 8 kilobytes of memory, and then requests more as required. This value represents an upper bound.

nquota

The number of "quota" structures allocated. Quota structures are present only when disk quotas are configured in the system. One quota structure is kept per user. This number is defined to be (MAXUSERS * 9) / 7 + 3.

ndquot

The number of "dquot" structures allocated. Dquot structures are present only when disk quotas are configured in the system. One dquot structure is required per user, per active file system quota. That is, when a user manipulates a file on a file system on which quotas are enabled, the information regarding the user's quotas on that file system must be in-core. This information is cached, so that not all information must be present in-core all the time. Dquot is defined as NINODE + (MAXUSERS * NMOUNT) / 4, where NMOUNT is the maximum number of mountable file systems.

In addition to the above values, the system page tables (used to map virtual memory in the kernel's address space) are sized at compile time by the SYSPTSIZE definition. This is defined to be 20 + MAXUSERS pages of page tables. Its definition affects the size of many data structures allocated at boot time because it constrains the amount of virtual memory which may be addressed by the running system. This is often the limiting factor in the size of the buffer cache, in which case a message is printed when the system configures at boot time.

Run-time calculations

The most important data structures sized at run-time are those used in the buffer cache. Allocation is done by swiping physical memory (and the associated virtual memory) immediately after the system has been started up; look in the file /sys/ca/machdep.c.

The buffer cache is comprised of several "buffer headers" and a pool of pages attached to these headers. Buffer headers are divided into two categories, those used for swapping and paging and those used for normal file I/O. The system tries to allocate 10% of available physical memory for the buffer cache (where *available* does not count that space occupied by the system's text and data segments). If the result is fewer than 16 pages of memory allocated, then 16 pages are allocated. This value is kept in the initialized variable *bufpages* so that it may be patched in the binary image (to allow tuning without recompiling the system). Adequate file I/O buffer headers are then allocated to allow each to hold 2 pages each, and half as many swap I/O buffer headers are then allocated. The number of swap I/O buffer headers is constrained to be no more than 256.

System size limitations

As distributed, the sum of the virtual sizes of the core-resident processes is limited to 512M bytes. The size of the text, and data segments of a single process are currently limited to 16M bytes each, and the stack segment size is limited to 512K bytes as a soft, user-changeable limit, and may be increased to 16M with the *setrlimit*(2) system call. If these are insufficient, they can be increased by changing the constants MAXTSIZ, MAXDSIZ and MAXSSIZ in the file /sys/ca/vmparam.h, as well as changing the definitions in /sys/h/dmap.h and /sys/h/text.h. In making this change, be sure you have adequate paging space. As normally configured, the system has only 16M bytes per paging area. The best way to get more space is to provide multiple, thereby interleaved, paging areas.

To increase the amount of resident virtual space possible, you can alter the constant USRPTSIZE (in /sys/ca/vmparam.h). To allow 1 gigabyte of resident virtual space one would change the 1 to a 2.

Because the file system block numbers are stored in page table *pg_blkno* entries, the maximum size of a file system is limited to 2^19 2048 byte blocks. Thus no file system can be larger than 1 gigabyte.

The number of mountable file systems is set at 20 by the definition of NMOUNT in /sys/h/parm.h. This should be sufficient; if not, the value can be increased up to 255. If you have many disks, it makes sense to make some of them single file systems; the paging areas don't count in this total.

The limit to the number of files that a process may have open simultaneously is set to 64. This limit is set by the NOFILE definition in /sys/h/parm.h. It may be increased arbitrarily, with the caveat that the user structure expands by 5 bytes for each file, and thus UPAGES must be increased accordingly.

APPENDIX E. NETWORK CONFIGURATION OPTIONS

The network support in the kernel is self-configuring according to the protocol support options (INET and NS) and the network hardware discovered during autoconfiguration. There are several changes that may be made to customize network behavior due to local restrictions. Within the Internet protocol routines, the following options set in the system configuration file are supported:

GATEWAY

The machine is to be used as a gateway. This option currently makes only minor changes. First, the size of the network routing hash table is increased. Secondly, machines that have only a single hardware network interface will not forward IP packets; without this option, they will also refrain from sending any error indication to the source of unforwardable packets. Gateways with only a single interface are assumed to have missing or broken interfaces, and will return ICMP unreachable errors to hosts sending them packets to be forwarded.

TCP_COMPAT_42

This option forces the system to limit its initial TCP sequence numbers to positive numbers. Without this option, 4.3BSD systems may have problems with TCP connections to 4.2BSD systems that connect but never transfer data. The problem is a bug in the 4.2BSD TCP; this option should be used during the period of conversion to 4.3BSD.

IPFORWARDING

Normally, 4.3BSD machines with multiple network interfaces will forward IP packets received that should be resent to another host. If the line "options IPFORWARDING="0"" is in the system configuration file, IP packet forwarding will be disabled.

IPSENDREDIRECTS

When forwarding IP packets, 4.3BSD IP will note when a packet is forwarded using the same interface on which it arrived. When this is noted, if the source machine is on the directly-attached network, an ICMP redirect is sent to the source host. If the packet was forwarded using a route to a host or to a subnet, a host redirect is sent, otherwise a network redirect is sent. The generation of redirects may be inhibited with the configuration option "options IPSENDREDIRECTS="0"."

SUBNETSARELOCAL

TCP calculates a maximum segment size to use for each connection, and sends no datagrams larger than that size. This size will be no larger than that supported on the outgoing interface. Furthermore, if the destination is not on the local network, the size will be no larger than 576 bytes. For this test, other subnets of a directly-connected subnetted network are considered to be local unless the line "options SUBNETSARELOCAL="0"" is used in the system configuration file.

COMPAT_42

This option, intended as a catchall for 4.2BSD compatibility options, has only a single function thus far. It disables the checking of UDP input packet checksums. As the calculation of UDP packet checksums was incorrect in 4.2BSD, this option allows a 4.3BSD system to receive UDP packets from a 4.2BSD system.

The following options are supported by the Xerox NS protocols:

NSIP

This option allows NS IDP datagrams to be encapsulated in Internet IP packets for transmission to a collaborating NSIP host. This may be used to pass IDP packets through IP-only link layer networks. See *nsip*(4P) for details.

THREEWAYSHAKE

The NS Sequenced Packet Protocol does not require a three-way handshake before considering a connection to be in the established state. (A three-way handshake consists of a connection request, an acknowledgement of the request along with a symmetrical opening indication, and then an acknowledgement of the reciprocal opening packet.) This option forces a three-way handshake before data may be transmitted on Sequenced Packet sockets.

IBM RT PC New Model Series Upgrade Instructions for IBM/4.3

NOTICE

> **These instructions are only for IBM RT PCs that will run IBM Academic Operating System 4.3. If your RT is running AIX, use the instructions that accompany the Upgrade Kit.**

ABSTRACT

This article describes the procedures used to install the New Model Series Upgrade Kit on an IBM RT[1] Personal Computer, Models 10, 15, 20, or 25 that will be running IBM Academic Operating System 4.3 (IBM/4.3). Note that for Models 10 and 20, the user for purchasing and installing E70 disks, if desired, as they are not standard.

The installation procedures involve both hardware and software instructions. This article contains the following sections:

Charts contains a complete set of the charts found in this article. This set is provided so that you can easily make a copy for each of the system units you will be upgrading.

Installation Checklist contains a list you should use as you complete the the installation of the upgrade kit.

Step 1. Complete the Upgrade Kit Contents Checklist describes how to verify that all the needed pieces of the upgrade kit are available.

Step 2. Complete the Prerequisites Checklist shows how to determine if the system hardware and software are at the correct levels and available before continuing installation.

Step 3. Back Up the Existing System provides instructions for backing up the current system.

Step 4. Record Minidisk Information describes how to display and record minidisk sizes and locations.

Step 5. Install the New Hardware provides step-by-step instructions for installing the advanced processor card and the extended ESDI adapter.

Step 6. Run Diagnostics describes how to run the system checkout diagnostics.

Step 7. Run the IBM/4.3 Standalone Formatter explains how to reformat the disks and rewrite the bad block tables.

Step 8. Recreate Minidisks describes how to use the information captured in Step 4 to recreate minidisks.

Step 9. Install IBM/4.3 points you to "Installing and Operating IBM/4.3" in Volume II of *IBM Academic Operating System 4.3*. When all eight steps in this article are completed, you are ready to install IBM/4.3.

[1] RT Personal Computer, RT, and AIX are trademarks of International Business Machines Corporation.

UNIX, where it appears in this document, is a registered trademark of AT&T Bell Laboratories.

This page intentionally left blank.

CHARTS

This section contains a complete set of the charts found throughout this article. Use the set as a master to produce copies for each of the system units you will be upgrading. When performing an upgrade, you should mark your entries on one of the copies wherever a chart appears in the article.

INSTALLATION CHECKLIST

The checklist below includes each step in the upgrade process in the order it is to be performed, along with the approximate time required. Check off each step as it is completed. Do not proceed to the next step until you have completed the current step. Read each step carefully and complete all actions described in each step.

	STEP	GO TO PAGE	TIME REQUIRED
□	1. Complete the Upgrade Kit Contents Checklist	9	3 minutes
□	2. Complete Prerequisites Checklist	10	5 minutes
□	3. Back Up the Existing System	11	35 minutes*
□	4. Record Minidisk Information	12	10 minutes
□	5. Install the New Hardware	15	60 minutes
□	6. Run Diagnostics	33	20 minutes
□	7. Run the IBM/4.3 Standalone Formatter	36	100 minutes/disk
□	8. Recreate Minidisks	38	5 minutes/disk
□	9. Install IBM/4.3	40	90 minutes

* This is the time required to back up the root, user, and source file systems.

UPGRADE KIT CONTENTS

UPGRADE KIT CONTENTS

□ 1. IBM RT PC Advanced Processor Card (APC)

or

IBM RT PC Advanced Processor Card (APC)[2] plus 4MB Fast Memory Card

□ 2. IBM RT PC Extended ESDI Magnetic Media Adapter

□ 3. Type E70 Fixed Disk Drive Labels

□ 4. IBM RT PC Extended ESDI Cable Bundle
(Not used by the IBM 6151)

□ 5. IBM RT ESDI to Extended ESDI Disk Conversion Utility
Diskette (Note that you will NOT use this diskette. Instead,
you will use the SAUTIL diskette that comes with IBM/4.3.)

[2] The APC may or may not have onboard memory. To determine if it does, hold the card so that the two 32-bit connectors are pointing down, and the front (where the chips are mounted) is facing you. An APC with onboard memory has an 8x5 array of "silver" chips in the upper left corner whereas an APC without onboard memory has no such array. If you have an APC without onboard memory, you should also have a 4MB Fast Memory card.

PREREQUISITES CHECKLIST

		Description	Action
□	1.	Any necessary preinstallation hardware changes already completed (Required for all IBM 6151 system units)	Call your service representative.
□	2.	IBM RT PC Diagnostic Diskettes Version 2.1.1 available	If not in your new *Problem Determination Guide*, call your service representative.
□	3.	Streaming Tapes available for backup	Tapes are supplied by the customer (see Note).
□	4.	IBM/4.3 available and ready to install	Obtain the distribution tapes or the necessary network connection to an RT with IBM/4.3 already installed.

NOTE: The number of tapes needed for backup depends upon the amount of data to be backed up. At least one streaming tape is required for each file system to be backed up.

HD70R DISKS

Disk	hd70r
hd0:	
hd1:	
hd2:	

DISK IODN, NAME, SIZE, TYPE

hd0:

Iodn	Name	Size	Type

hd1:

Iodn	Name	Size	Type

hd2:

Iodn	Name	Size	Type

INSTALLED OPTIONS LIST

The following options are installed on your machine.

ADAPTER/DEVICE	SLOT

This page intentionally left blank.

INSTALLATION CHECKLIST

The checklist below includes each step in the upgrade process in the order it is to be performed, along with the approximate time required. Check off each step as it is completed. Do not proceed to the next step until you have completed the current step. Read each step carefully and complete all actions described in each step.

	STEP	GO TO PAGE	TIME REQUIRED
□	1. Complete the Upgrade Kit Contents Checklist	9	3 minutes
□	2. Complete Prerequisites Checklist	10	5 minutes
□	3. Back Up the Existing System	11	35 minutes*
□	4. Record Minidisk Information	12	10 minutes
□	5. Install the New Hardware	15	60 minutes
□	6. Run Diagnostics	33	20 minutes
□	7. Run the IBM/4.3 Standalone Formatter	36	100 minutes/disk
□	8. Recreate Minidisks	38	5 minutes/disk
□	9. Install IBM/4.3	40	90 minutes

* This is the time required to back up the root, user, and source file systems.

STEP 1. COMPLETE THE UPGRADE KIT CONTENTS CHECKLIST

Before you begin the Upgrade Kit installation, you must determine the following:

- Are all the materials for the upgrade available?
- Is the system hardware at the correct level?
- Are all the materials for installing IBM Academic Operating System 4.3 available?

The checklists contained in this and the next section will provide information to answer these three questions.

Complete the checklists according to the instructions provided.

Using a copy of this chart, check off each item in the upgrade kit.

UPGRADE KIT CONTENTS

□ 1. IBM RT PC Advanced Processor Card (APC)

or

IBM RT PC Advanced Processor Card (APC)[3] plus 4MB Fast Memory Card

□ 2. IBM RT PC Extended ESDI Magnetic Media Adapter

□ 3. Type E70 Fixed Disk Drive Labels

□ 4. IBM RT PC Extended ESDI Cable Bundle
(Not used by the IBM 6151)

□ 5. IBM RT ESDI to Extended ESDI Disk Conversion Utility Diskette (Note that you will NOT use this diskette. Instead, you will use the SAUTIL diskette that comes with IBM/4.3.)

If any of first four items is missing, contact your sales representative.

If the first four items are present, go to STEP 2.

Note that users of 4.2/RT or IBM/4.3 should ignore the document entitled "IBM RT PC New Model Series Upgrade Kit Instructions" which comes with the Upgrade Kit. Use these instructions instead.

[3] The APC may or may not have onboard memory. To determine if it does, hold the card so that the two 32-bit connectors are pointing down, and the front (where the chips are mounted) is facing you. An APC with onboard memory has an 8x5 array of "silver" chips in the upper left corner whereas an APC without onboard memory has no such array. If you have an APC without onboard memory, you should also have a 4MB Fast Memory card.

STEP 2. COMPLETE THE PREREQUISITES CHECKLIST

The following checklist determines if the system hardware and software are at levels necessary for installing the Upgrade Kit. If you can check off **all** the items on your copy of the list, continue with the installation.

See the **Action** column for further instructions if you cannot check off an item.

		Description	Action
□	1.	Any necessary preinstallation hardware changes already completed (Required for all IBM 6151 system units)	Call your IBM service representative.
□	2.	IBM RT PC Diagnostic Diskettes Version 2.1.1 available	If not in your new *Problem Determination Guide*, call your IBM service representative.
□	3.	Streaming Tapes available for backup	Tapes are supplied by the customer (see Note).
□	4.	IBM/4.3 available and ready to install	Obtain the distribution tapes or the necessary network connection to an RT with IBM/4.3 already installed.

NOTE: The number of tapes needed for backup depends upon the amount of data to be backed up. At least one streaming tape is required for each file system to be backed up.

Adapters Needing Upgrading

If you have the following adapters installed in the system unit, the service representative must upgrade these adapters to work properly with the new adapters installed with the Upgrade Kit. If you have not had the service representative upgrade these cards before beginning this installation, call your service representative now.

Device or Adapter	Action
IBM RT PC Advanced Monochrome Graphics Adapter (for the 6153)	This adapter may require an upgrade in order to operate correctly with the Advanced Processor Card. Call your IBM service representative.
IBM RT PC Advanced Color Graphics Adapter (for the 6154)	This adapter may require an upgrade in order to operate correctly with the Advanced Processor Card. Call your IBM service representative.

STEP 3. BACK UP THE EXISTING SYSTEM

If your IBM RT PC is NOT currently running 4.2/RT or IBM/4.3, skip both this step and STEP 4; go to STEP 5.

If your IBM RT PC is presently running 4.2/RT or IBM/4.3, back up that system before proceeding. Use one of the following three methods (listed in order of preferred use):

(1) If your site has a standard backup procedure, use it now. When you are done, proceed with the instructions below for recording the locations of any hd70r disks.

(2) If your site has no standard backup procedure and you have access to a streaming tape drive, use the instructions for backing up the root (/), user (/usr), and source (/src) file systems provided in "Installing and Operating IBM/4.3" in Volume II of *IBM Academic Operating System 4.3*. Go there now and follow the instructions provided in the section entitled "Saving Modified Files" to perform the backup. When you are done, proceed with the instructions below for recording the locations of any hd70r disks.

(3) If you wish to back up your system to diskettes, use the instructions provided in the article entitled "Installing and Operating IBM/4.3" in Volume II of *IBM Academic Operating System 4.3*. Go there now and follow the instructions provided in the section entitled "Saving Modified Files" to perform the backup. When you are done, proceed with the instructions below for recording the locations of any hd70r disks.

Now record below the locations of any hd70r disks on your current system. To do this, enter the following command:

```
/etc/dmesg | grep hd70r
```

If you do not obtain the information you need, you must reboot your system and then re-enter the command.

In the chart below, put a check beside the disk unit if it is a hd70r disk. You will use this information in STEP 4.

Disk	hd70r
hd0:	
hd1:	
hd2:	

When you have backed up your system and recorded the locations of your hd70r disks, go to STEP 4.

STEP 4. RECORD MINIDISK INFORMATION

If your system *does not have* hd70r disks, you can skip STEPS 4, 7, and 8. Those steps deal with converting hd70r disks to hd70e disks. Go now to STEP 5 if your system has no hd70r disks.

These instructions are used to determine the status of minidisks on your hd70r disk(s). The three possibilities are:

- No minidisks exist on your disk(s).
- Standard minidisks exist on your disk(s).
- Non-standard minidisks exist on your disk(s).

In this step, you will display and record the current minidisk information. This information is necessary so that you can create the same minidisk configuration after the upgrade is complete.

How to Display Minidisk Information

To display minidisk information, do the following. For more guidance, see *minidisk*(8R) in Volume I of *IBM Academic Operating System 4.3*.

(1) Reboot the system by typing the super user command (su), supplying the superuser password, and typing the following:

```
/etc/shutdown -r now <Enter>
```

After a few seconds, the system displays the following:

```
4.X BSD UNIX Standalone Boot Program $Revision: X.X $
Default: hd(0,0)vmunix (just press Enter or wait ~30 seconds)
:
```

(2) To invoke the standalone utilities, type the following command, ignoring the "would overlay boot" warning message:

```
hd(0,6)stand/sautil
```

The following display appears.

```
4.X BSD UNIX Standalone Maintenance Program $Revision:  X.X$

Choice     Description
1           boot - boot standalone program or kernel
2          format - format hard disk
3           dump - display disk or floppy (hex)
4           cat - display a file contents (ASCII)
5           ls - print directory of UNIX filesystem
6           copy - copy all/part of a disk or floppy
7           debugger - display memory etc.
8           iplsource - set boot order in nvram
9           minidisk - display/change minidisk directory
10          dosboot - boot standalone program or kernel from DOS diskette

Enter the menu choice desired then press Enter

Choice ?
```

(3) Type 9 and press < **Enter** > to display the minidisk directory. Information such as the following appears. You will type hd(unit,2) for each hd70r disk you recorded in the table in STEP 3. A sample response is shown in **bold**.

```
Typical disk names are hd(unit,2) where unit=0, 1, 2
disk  hd(0,2)
```

(4) If you see information like the following, then you have no minidisks on that disk. Respond with **n** to the re-initialize prompt and write the words "no minidisks" across the chart for that disk.

```
          hd(0,2):
          number= -32834071  level= -32834071  unused= -5011  first= -5011  last= -5011  bad_l
          =-328340371  bad_size= -328340371

index   iodn   name    date   start   size     type
  0      0     FREE            140    141450    00

          minidisk directory corrupted
          re-initialize minidisk directory [y/n] n
```

(5) If you see information like the following, then you have standard minidisks on that disk. Write the words "standard" across the chart for that disk.

```
          hd(0,2):

          number=5 level=1 unused=5 first=0 last=4 bad_block=141379 bad_size=1000

index   iodn    name              date                  start     size    type

  0     32746   boot    Wed Mar  18 06:53:20 1987         144      108    01 ipl
  1     32747   hd2a    Wed Mar  18 06:53:20 1987         252    16128    00
  2     32748   hd2b    Wed Mar  18 06:53:20 1987       16380    33516    20 swap
  3     32749   hd2g    Wed Mar  18 06:53:20 1987       49896    91476    00
  4         0   FREE                                   141372        8    00
```

(6) If you have non-standard minidisks on a disk, record the displayed information (iodn, name, size and type) on the chart for that disk.

hd0:

Iodn	Name	Size	Type

hd1:

Iodn	Name	Size	Type

hd2:

Iodn	Name	Size	Type

(7) Type **quit** <**Enter**>. Then press <**Enter**> again to return to the menu.

(8) Repeat instructions 3 through 7 for each hd70r disk.

(9) Turn off the machine.

When you have recorded the minidisk information for up to three hd70r disks, go to STEP 5.

STEP 5. INSTALL THE NEW HARDWARE

In this step you actually install the new hardware in the IBM RT PC system units. These procedures tell you how to:

- Load the Diagnostic Utilities from the diagnostic diskettes
- Use the SHOW INSTALLED OPTIONS utility to record adapters, options, and drive types installed in your system unit
- Remove and replace the covers of your system unit
- Remove and replace adapters, options, and cables

  WARNING: If you have not completed Checklist Steps 1 through 4, DO NOT continue with these instructions. Data may be lost if the steps are not followed in strict sequence.

Loading the Diagnostic Utilities

Use the following steps to load the hardware diagnostic diskette:

(1) Remove the diagnostic diskettes from the back of the *Problem Determination Guide*. Be sure they are Version 2.1.1.

Note: If you do not have Version 2.1.1 Diagnostic Diskettes, contact your sales representative before continuing with the Upgrade Installation.

(2) Remove the write protect tab, if present, from the diskettes.

(3) Insert **DIAG-1** into diskette drive 0 (or A) and close the drive. Then turn the system unit on.

When inserting a diskette into a diskette drive, always hold the diskette with the label side up and the notch on the side of the diskette facing left. If your IBM RT PC has two diskette drives, insert the diskette in the upper drive.

Note: Do not operate or move any device, unless instructed to do so. Also note that the diagnostic diskettes do not support the experimental display.

After about two minutes, the IBM logo screen will appear while the diagnostics are loading.

(4) Wait until the DIAGNOSTICS OPERATING INSTRUCTIONS appear. (The system may not appear active until this time.)

This screen describes the function keys used when running diagnostics. The function keys and their uses are as follows:

Keys	Usage
Enter	Continues the procedure or performs an action.
End	Stops a test or an action.
F10	Returns to the previous menu and cancels any changes.
Esc	Resets the procedures and returns to the Diagnostic Operating Instructions.
Page Down	Displays the following page of information.

Page Up	Displays the preceding page of information.
Backspace	Moves the cursor back one position; used for correcting errors.

Running Diagnostics

Before you install the new hardware, run diagnostics to check for any hardware problems with the devices and adapters presently installed in your system.

Use the following instructions to run diagnostics.

(1) After you have read the DIAGNOSTIC OPERATING INSTRUCTION screen, press **<Enter>**.

The FUNCTION SELECTION menu appears.

(2) Press **<Enter>** to select *Diagnostic Routines.*

(3) Insert diagnostic diskette **DIAG-2** when prompted and press **<Enter>**.

The DIAGNOSTIC SELECTION menu appears.

(4) Type **1** and press **<Enter>** to select *System Checkout.*

The TEST METHOD SELECTION menu appears.

(5) Press **<Enter>** to select *Run Test One Time.*

(6) Follow the instructions as they appear on the display to test the system hardware.

When the test completes, the TESTING COMPLETE screen appears if no problems were found.

(7) Press **<Enter>** to return to the DIAGNOSTIC SELECTION menu.

(8) Type **99** and press **<Enter>** to return to the FUNCTION SELECTION menu. Do not remove the diskette.

Note: If there is a hardware problem with the machine, contact your service representative.

Running Utilities

Use the following instructions to use the Show Installed Options Utility.

(1) On the FUNCTION SELECTION menu, type **2** and press **<Enter>** to select *Utilities.*

The UTILITY SELECTION menu appears.

(2) Type **1** and press **<Enter>** to select *Show Installed Options Utility.*

This utility displays which options are installed on your machine.

(3) Record the displayed information on the blank INSTALLED OPTIONS LIST below.

Note: Use the **Page Up** and **Page Down** keys to display all the items on the list.

(4) Save the information you recorded. You will use it to configure the system after the new hardware is installed.

INSTALLED OPTIONS LIST

The following options are installed on your machine.

ADAPTER/DEVICE	SLOT

Exiting the Utility

After you have recorded your system configuration, exit the utility.

(1) Press <**Enter**> to return to the UTIL:ITY SELECTION menu.

(2) Press <**Esc**> to return to the DIAGNOSTICS OPERATING INSTRUCTIONS screen.

(3) Remove the diskette from the diskette drive.

(4) Replace the diagnostics diskettes in the back of the *Problem Determination Guide.*

To help you install the new hardware, the rest of STEP 5 is divided into two parts:

- STEP 5A, "IBM 6150 Hardware Installation" on the next page is for floor model IBM RT PCs.
- STEP 5B, "IBM 6151 Hardware Installation" on page 28 is for desk model IBM RT PCs.

To begin the hardware installation, go to the appropriate page for your system unit. You will need a medium-sized, flat-bladed screwdriver to remove and install some components.

STEP 5A. IBM 6150 HARDWARE INSTALLATION

This section is intended only for users of the IBM 6150 (floor model) system unit. If you have an IBM 6151 (desk model) system unit, go to STEP 5B.

Removing the System Unit Covers

Remove the covers from the system unit. Do not operate the system unit with the covers removed. Operating with the covers on ensures adequate cooling for the components. You do not need any tools to remove the covers of the system unit. You will probably find this and the following tasks easier to complete if you put the system unit on a sturdy table.

CAUTION: This device is heavy, weighing up to 84 pounds (38 kg). Two or more people should lift it.

(1) Turn the system and any connected devices off.

(2) Unplug the power cable from the electrical outlet.

(3) Firmly grasp the bottom of the rear cover and snap the cover free.

(4) Rotate the cover outward until the hooks are clear of the top of the unit.

(5) Unplug the display power cable and the system unit power cable. Label any cables you unplug from the rear of the system.

(6) Unlock the keylock on the front cover and remove the key.

(7) Push up on the front cover tab until it is free from the hole in the bottom of the unit.

(8) Rotate the cover out and away from the bottom of the unit.

(9) Lower the cover until the hooks are clear of the top of the unit.

(10) Find the tab under one outside edge of the left side cover.

(11) Push up on the tab until it is free of the hole in the cover.

(12) Do the same to the tab on the other end.

(13) Rotate the cover out and away from the bottom of the unit.

(14) Lift the side cover up until the hooks are free of the loops in the top of the unit.

You have completed removing those covers necessary for installing the new hardware.

Removing and Replacing Adapters and Options

> **WARNING:** Adapters can be damaged by static electricity discharge. The adapters are wrapped in antistatic bags to prevent this damage. Take the following precautions when handling devices sensitive to static electricity discharge:
>
> 1. Hold the device (still in the antistatic bag) in one hand, and with the other hand, touch the metal on the system unit frame.
>
> *Note:* When handling devices or adapters, hold them by the corners or brackets only. Do not touch the components, modules, or connections.
>
> 2. Immediately remove the device from the antistatic bag.

Installing the Advanced Processor Card

The following instructions assume that:

- The system unit is on a table.
- The front of the unit is toward your right hand.
- You are facing the left side of the unit with all three covers removed.

These instructions guide you through removing the card installed in Slot A and installing the Advanced Processor Card in its place.

(1) Hold the card in Slot A by the edges and carefully remove it.

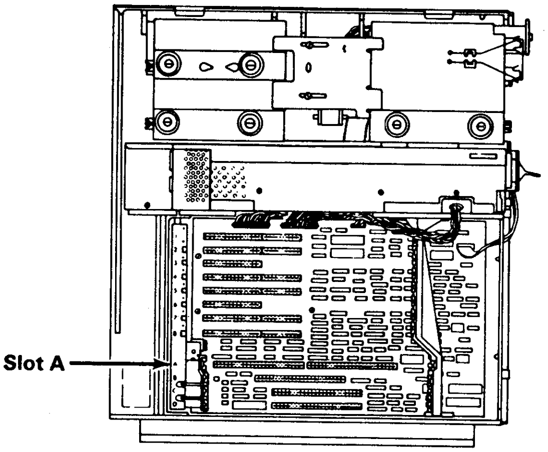


FIGURE 1. SLOT A

(2) Unpack the APC adapter, using the instructions for handling static-sensitive devices on the previous page.

(3) Holding the card by the edges, point the tab toward Slot A on the system board and press the card into place.

(4) Ensure that the right edge is in the plastic guides and the card is completely in position.

(5) Remove the memory cards installed in Slots C and D. Hold each card by the edges and carefully remove it.

(6) If you have the 4MB Fast Memory card, install it in Slot C. Hold the card by the edges and press it into place.

Note: Do not remove the Floating Point Accelerator (if any) installed in Slot B.

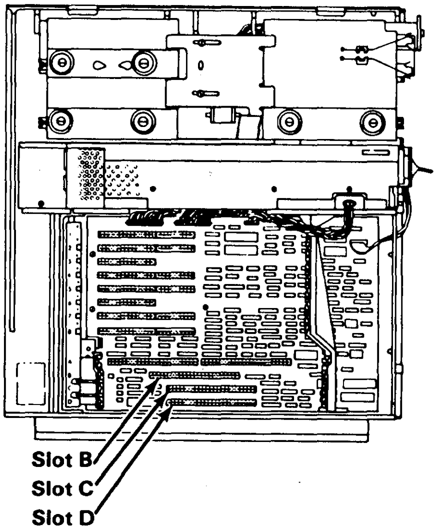


FIGURE 2. SLOTS B, C, AND D

If you *are not* installing the new Extended ESDI disk adapter, you have completed the necessary hardware installation. You should now go to "Replacing the System Unit Covers" on page 27.

Locating the New Fixed Disk Adapter Slot

If your system unit has hd70r fixed disks, you can replace the IBM RT PC ESDI Magnetic Media Adapter with the new fixed disk adapter shipped in the upgrade kit. The new adapter, the IBM RT PC Extended ESDI Magnetic Media Adapter and the new cable bundle will support the existing diskette drives and up to three hd70e fixed disks. Note that if your system unit has other non-hd70r fixed disks installed, you cannot remove the IBM Personal Computer AT Fixed Disk and Diskette Drive Adapter to which they are attached. Only hd70r fixed disks will operate with the new adapter and cable. Also note that combining hd70r and non-hd70r drives requires use of the old cables, as shown in Figure 4.

WARNING: Do not change the physical location of any hd70r fixed disks. Moving disks now will cause problems later. If you need to change the physical location of a fixed disk, do so after you have completed these steps.

Use Figure 3 on the next page to determine the correct fixed disk positions. Use the accompanying table to determine the correct slot for the new adapter.

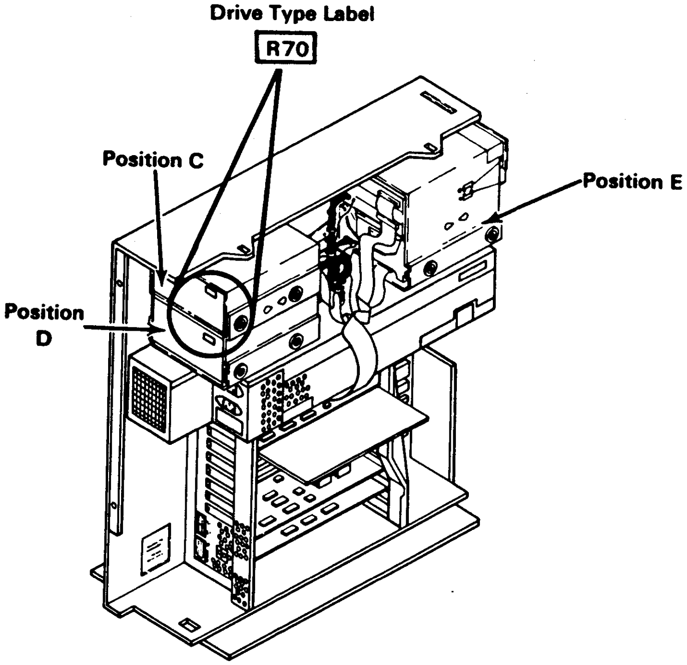


FIGURE 3. POSITIONS C, D, AND E

If the system has disk(s) installed in:			Remove the adapter in:	Install the new adapter in:	And use these cables:
C	D	E			
non-R70			N/A	N/A	N/A
non-R70	non-R70		N/A	N/A	N/A
non-R70		R70	Slot 2	Slot 2	old
non-R70	non-R70	non-R70	N/A	N/A	N/A
non-R70	non-R70	R70	Slot 2	Slot 2	old
R70			Slot 1	Slot 1	new
R70		non-R70	Slot 1	Slot 1	old
R70	R70		Slot 1	Slot 1	new
R70	R70	non-R70	Slot 1	Slot 1	old
R70	R70	R70	Slots 1 and 2	Slot 1	new

Note 1: "Non-R70" includes M40, M70, R40 and C70 disks.

Note 2: Remove only the adapters indicated by the above table. Some systems will have two different fixed disk adapters installed.

Note 3: Using the old cables may require removing the blocking pins from locations C-AB, C-CD, C-C1, and C-D1 (on Cable 1), and locations C-E and C-E1 (on Cable 2).

Removing Fixed Disk Adapters

You will remove the old disk adapter(s) next, but will not install the new adapter until later in this section.

(1) To remove an adapter from Slot 1 or 2, remove the screw holding the adapter in place. Save the screw to install the new adapter later.

(2) Holding the edges of the adapter, slide the adapter out of the slot.

(3) Disconnect the 34-pin connector(s) and the 20-pin connector(s).

Do not install the new adapter at this time.

Removing and Installing Cables

NOTE: If you have both non-hd70r and hd70r fixed disks installed in the system unit, **DO NOT** install the new cable bundle. Skip this step and go to "Applying the New E70 Labels" below.

If your system has three fixed disks, there are two separate cable bundles in the system:

- Cable 1 connects the diskette drives and the first two fixed disks to the adapter in Slot 1.
- Cable 2 connects the third fixed disk to the adapter in Slot 2.

If you have only hd70r fixed disks, you will install the new cable bundle in the next step. The cables should not be connected to the ESDI adapter(s) at this time. You should have already removed the ESDI adapters and disconnected the cables.

The new cable bundle can connect two diskette drives and up to three hd70e fixed disks.

Before you can remove the old fixed disk cables, you must remove the cooling fan assembly.

(1) Loosen the mounting screws of the cooling fan assembly and slide the fan bracket to the right.

(2) Remove the cooling fan assembly.

(3) Disconnect the cooling fan power connector.

(4) Disconnect the 34-pin and the 20-pin connectors from the rear of the fixed disks.

(5) Remove the cables from the system unit.

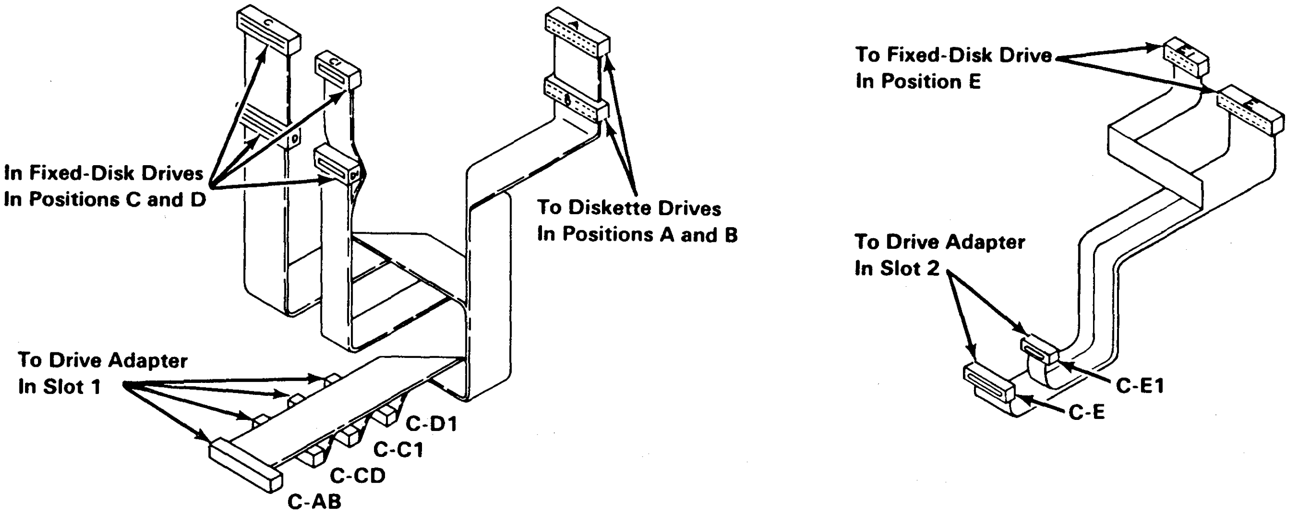


FIGURE 4. OLD CABLES

Installing the New ESDI Cable

(1) Carefully route the new cable as shown in the picture below.

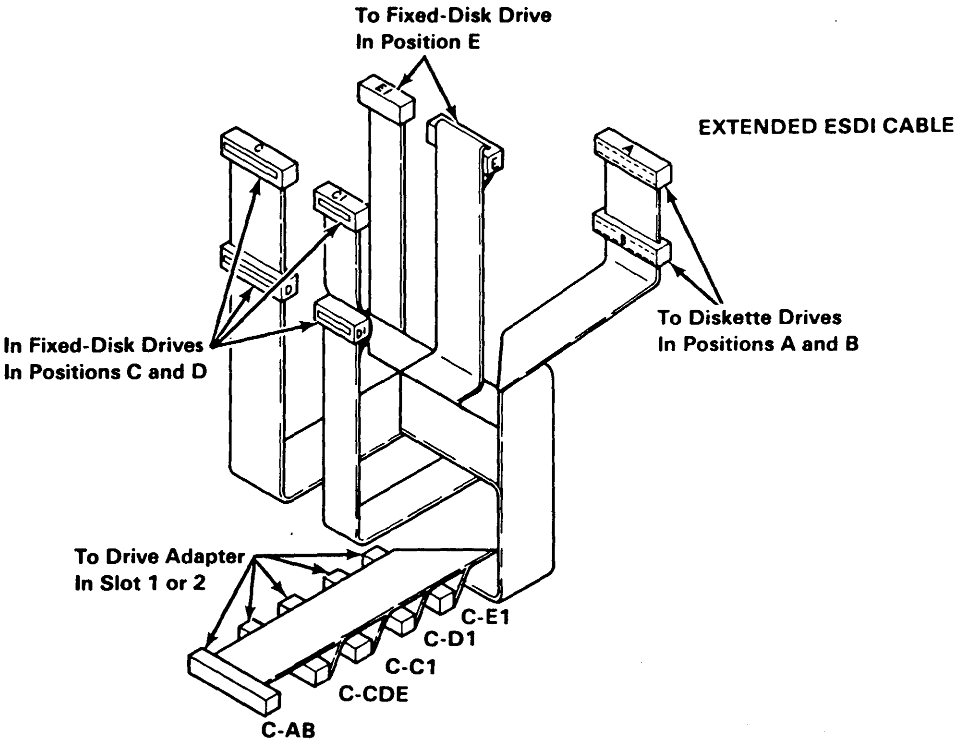


FIGURE 5. NEW CABLE

(2) Connect the 34-pin connectors as follows:

- A to Diskette Drive in Position A
- B to Diskette Drive in Position B
- C to Fixed Disk in Position C
- D to Fixed Disk in Position D (if installed)
- E to Fixed Disk in Position E (if installed)

(3) Connect the 20-pin connectors as follows:

- C1 to Fixed Disk in Position C
- D1 to Fixed Disk in Position D (if installed)
- E1 to Fixed Disk in Position E (if installed)

Applying the New E70 Labels

Before you install the new adapter, apply the new drive type labels (E70) over the R70 labels on your fixed disks. Three E70 labels were shipped with the Upgrade Kit.

(1) Locate the front side of the fixed disk.

(2) Wipe off any dirt or dust on the existing R70 label.

(3) Carefully peel the backing off the new E70 label, and place the new label on top of the R70 label.

Repeat these instructions for each R70 drive.

Installing the New Adapter

The following instructions describe how to install the Extended ESDI Magnetic Media Adapter in Slot 1 or 2 and how to connect the cables to the adapter.

To Install the Adapter in Slot 1

(1) Hold the adapter by the bracket and edges.

(2) Set the adapter jumper for Slot 1. Use the two left-most positions for Slot 1. See Figure 6.

(3) Connect the 20-pin connectors C-C1, C-D1, and (if present) C-E1 to the 20-pin connectors on the adapter. See Figure 7.

(4) Connect the 34-pin connectors C-AB and C-CDE to the 34-pin connectors on the adapter. See Figure 7.

(5) Slide the adapter in Slot 1 and replace the screw you removed earlier.

Replace the fixed-disk adapter in Slot 2 if needed, using the instructions that follow.

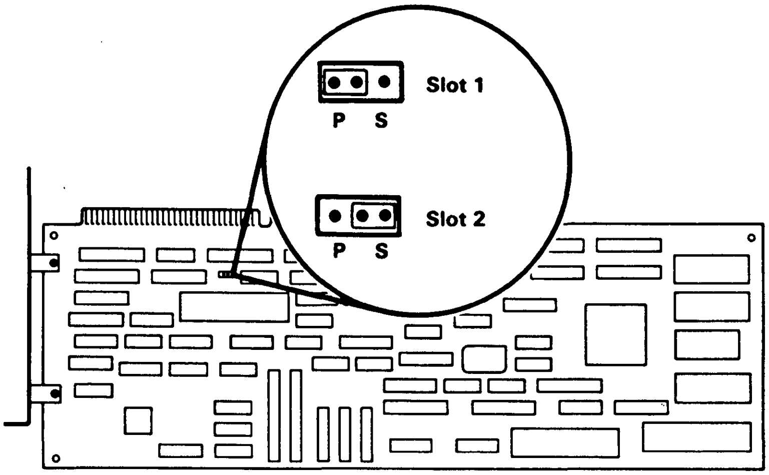


FIGURE 6. ADAPTER JUMPER

To Install the Adapter in Slot 2

(1) Hold the adapter by the bracket and edges.

(2) Set the adapter jumper for Slot 2. Use the two right-most positions for Slot 2. See Figure 6.

(3) Connect the 20-pin connector C-E1 to the 20-pin connector on the adapter. See Figure 7.

(4) Connect the 34-pin connectors C-CDE to the 34-pin connector on the adapter. See Figure 7.

(5) Slide the adapter in Slot 2 and replace the screw you removed earlier.

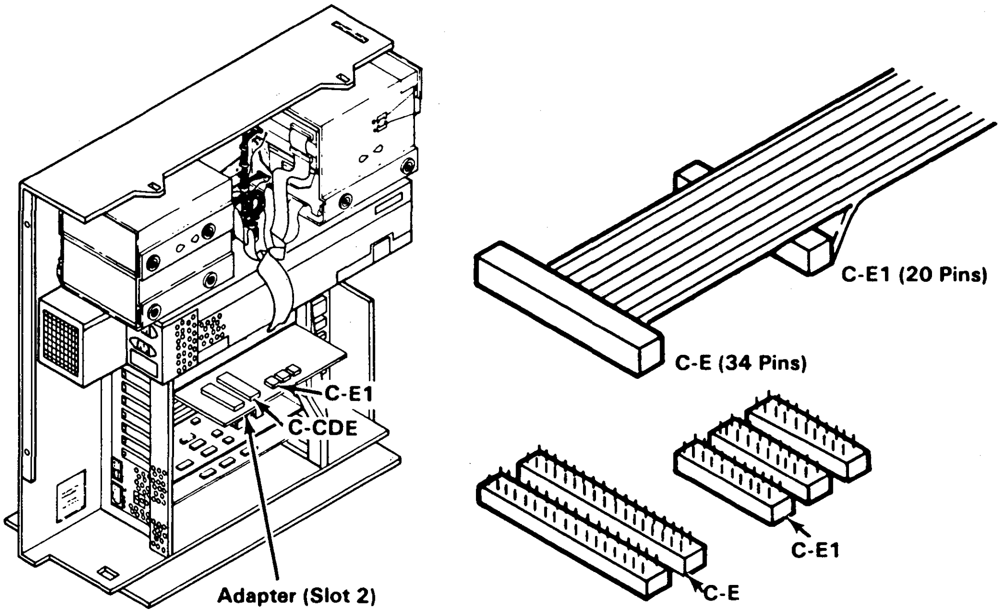


FIGURE 7. ADAPTERS AND CABLE CONNECTORS

To Finish Installing the Adapter

Skip this section if you have both R70 and non-R70 disks in your system.

(1) Replace the cooling fan, if necessary, by reconnecting the fan power connector.

(2) Insert the right end of the cooling fan assembly onto the front cage, then slide the fan bracket to the left behind the rear cage until the alignment holes between the two screws align.

(3) Tighten the mounting screws.

Replacing the System Unit Covers

(1) Replace the left side cover. Insert the top hooks in the loops in the top of the frame. Rotate the bottom of the cover inward and snap the cover in place.

(2) Replace the front cover by inserting the hooks under the top edge cover.

(3) Replace the rear cover, first reconnecting any cables you disconnected earlier and arranging them so that you can fit the rear cover in place.

(4) Plug the power cables into the electrical outlet.

When you have completed installing the new hardware, go to STEP 6.

STEP 5B. IBM 6151 HARDWARE INSTALLATION

This section is intended only for users of the IBM 6151 (desk model) system unit. If you have an IBM 6150 (floor model) system unit, go to STEP 5A.

Removing the System Unit Covers

Remove the covers from the system unit. Do not operate the system unit with the covers removed. Operating with the covers on ensures adequate cooling for the components. You do not need any tools to remove the covers of the system unit. You will probably find this and the following tasks easier to complete if you put the system unit on a sturdy table.

CAUTION: This device is heavy, weighing up to 51 pounds (23 kg). Two or more people should lift it.

(1) Turn the system and any connected devices off.

(2) Unplug all power cables from electrical outlets.

(3) Push down on the rear cover, directly over the two loops.

(4) Remove the loops from the top hooks.

(5) Lift the cover off the bottom hooks.

(6) Unplug the display power cable and the system unit power cable.

(7) Disconnect and label all cables from the rear of the system.

(8) Unlock the keylock on the top cover and remove the key.

(9) Remove the four screws holding the top cover in place. Save these screws as you will need them later.

(10) Pull forward on the cover and slide it off the system unit.

Removing and Replacing Adapters and Options

> **WARNING:** Adapters can be damaged by static electricity discharge. The adapters are wrapped in antistatic bags to prevent this damage. Take the following precautions when handling devices sensitive to static electricity discharge:
>
> 1. Hold the device (still in the antistatic bag) in one hand, and with the other hand, touch the metal on the system unit frame.
>
> *Note:* When handling devices or adapters, hold them by the corners or brackets only. Do not touch the components, modules, or connections.
>
> 2. Immediately remove the device from the antistatic bag.

Installing the Advanced Processor Card

Before installing the Advanced Processor Card, you must remove the adapter installed in Slot 6 and disconnect the fixed disk and diskette cables from the adapter. After this adapter is removed, you can remove the processor card installed in Slot A and the memory cards installed in Slots C and D.

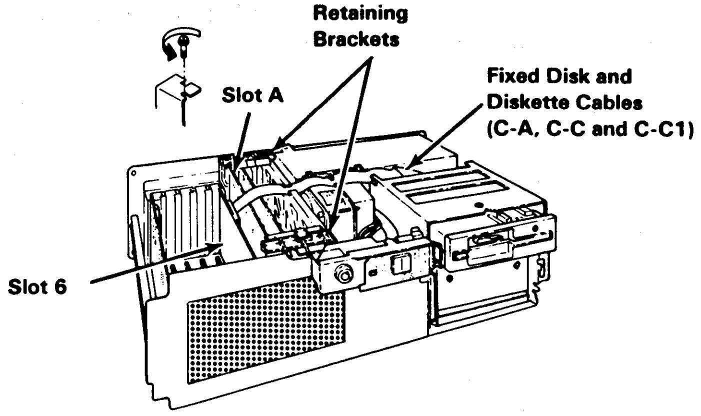


FIGURE 8. SLOTS A, C, D, AND 6

(1) Remove the screw holding the fixed disk adapter in Slot 6. Save the screw for later use.

(2) Hold the card in Slot 6 by the bracket and carefully slide it out of the slot.

(3) Disconnect the 34-pin connectors C-A and C-C and the 20-pin connector C-C1 from the adapter.

(4) Carefully move the cable so that it no longer covers Slot A.

(5) To remove the card installed in Slot A, open the retaining brackets.

(6) Hold the card by the edges and carefully remove it from Slot A.

(7) Remove the memory cards installed in Slots C and D. Hold each card by the edges and carefully remove it. (The memory cards must be removed because they do not work with the APC adapter.)

(8) If you have the 4MB Fast Memory card, install it in Slot C. Hold the card by the edges and press it into place.

Note: Do not remove the Floating Point Accelerator (if any) installed in Slot B.

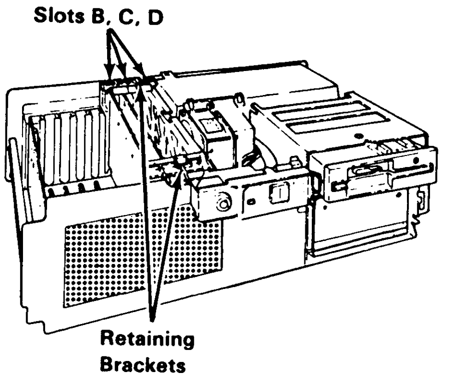


FIGURE 9. SLOTS B, C, AND D

(1) Unpack the APC adapter, using the instructions for handling static-sensitive devices on page 18.

(2) Holding the card by the edges, point the tab toward Slot A on the system board and press the card into place.

(3) Ensure that the right edge is in the plastic guides and the card is completely in position.

(4) Close the retaining brackets.

If you have a hd70r disk and are installing the Extended ESDI Magnetic Media Adapter, go to "Installing the Extended ESDI Magnetic Media Adapter" on the next page.

If you do not have hd70r disks or *are not* installing the new fixed disk adapter, continue with the instructions below.

Reinstalling the Fixed Disk Adapter

In this step, you will reinstall the adapter removed from Slot 6.

(1) Connect the 34-pin connectors C-A and C-C and the 20-pin connector C-C1 to the adapter.

(2) Hold the adapter by the bracket and edges. Point the tab toward Slot 6 and press the adapter into place.

(3) Ensure that the right edge is in the plastic guides and the adapter is completely in position.

(4) Align the screw holes and install the screw you removed earlier.

Now go to "Replacing the System Unit Covers" on page 32.

Installing the Extended ESDI Magnetic Media Adapter

The following instructions describe how to set the jumper on the new fixed disk adapter, how to connect the cables previously removed, how to replace the fixed disk type label, and how to install the new adapter.

Note: You must have hd70r disks in order to use the new Extended ESDI Adapter.

Applying the E70 Labels

Before you install the new adapter, apply the new drive type label (E70) over the R70 label on your fixed disk. An E70 label was shipped with the Upgrade Kit.

(1) Locate the front side of the fixed disk at the front of the system unit.

(2) Wipe off any dirt or dust on the existing R70 label.

(3) Carefully peel the backing off the new E70 label, and place the new label on top of the R70 label.

Setting the Adapter Jumper

Holding the adapter by the bracket and edges, set the adapter jumper. Use the two left-most position for the Slot 1 setting.

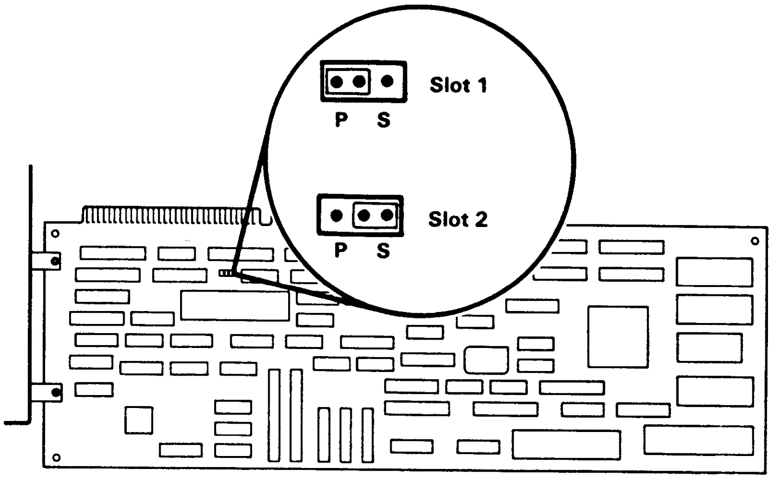


FIGURE 10. ADAPTER JUMPER

Installing the New Adapter

(1) Connect the 20-pin cable connector to the 20-pin connector on the adapter. See Figure 11.

(2) Connect the 34-pin cable connectors to the 34-pin connectors on the adapter. See Figure 11.

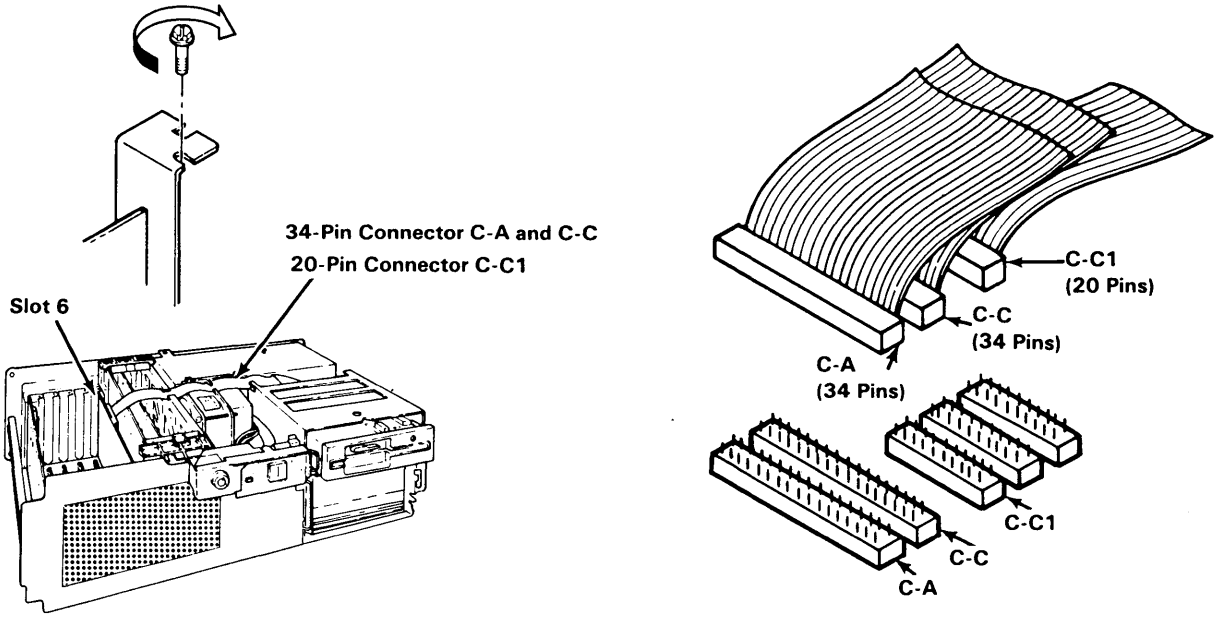


FIGURE 11. CABLES AND PIN CONNECTORS

(3) Hold the adapter by the bracket and edges. Point the tab toward Slot 6 and press the adapter in place.

(4) Ensure that the right edge is in the plastic guides and the adapter is completely in position.

(5) Align the screw holes and ensure that the adapter is even with the frame.

(6) Install the screw you removed earlier.

Replacing the System Unit Covers

(1) Replace the top cover by facing the system unit and sliding the cover over the unit. Ensure that all the cables are safely out of the way.

(2) Fit the rear cover plate in place and tighten the screws.

(3) Connect any cables you disconnected earlier.

(4) Slide the cables into the slots in the back cover.

(5) Insert the bottom hooks into the holes.

(6) Push down on the cover.

(7) Insert the top loops into the hooks.

(8) Plug the power cables into the electrical outlet.

When you have completed installing the new hardware, go to STEP 6.

STEP 6. RUN DIAGNOSTICS

In this step, you will do the following:

- Update the system's Diagnostics Test List
- Run the System Checkout procedure.

If you have questions as you run the diagnostics, refer to the *Problem Determination Guide.*

(1) Load the hardware diagnostics diskette as follows:

a. Remove the Version 2.1.1 diagnostic diskettes from the back of the *Problem Determination Guide.*

b. Remove the write protect tab, if one is present, from **DIAG-1**.

c. Insert **DIAG-1** into diskette drive 0 (or A).

(2) Power on the system; it will boot from the diagnostic diskette.

Note: Do not operate or move any device, unless instructed to do so, while the diagnostics are running.

After about two minutes, the IBM logo screen will appear while the diagnostics are loading. Wait until the DIAGNOSTIC OPERATING INSTRUCTIONS screen appears.

This screen describes the function keys used when running diagnostics. The function keys and their uses are as follows:

Keys	**Usage**
Enter	Continues the procedure or performs an action.
End	Stops a test or an action.
F10	Returns to the previous menu and cancels any changes.
Esc	Resets the procedures and returns to the Diagnostic Operating Instructions.
Page Down	Displays the following page of information.
Page Up	Displays the preceding page of information.
Backspace	Moves the cursor back one position; used for correcting errors.

Update the DIAGNOSTICS TEST LIST as follows:

(1) After you have read the DIAGNOSTIC OPERATING INSTRUCTIONS screen, press **<Enter>**.

The FUNCTION SELECTION menu appears.

(2) Press **<Enter>** to select Item 1, *Diagnostic Routines.*

(3) Insert diagnostic diskette **DIAG-2** when prompted. Then press **<Enter>**.

Either the TEST OPTION menu or the DIAGNOSTICS TEST LIST menu appears.

Removing Options from Diagnostics

The TEST OPTION menu appears when an option that was previously recognized by the system is not found during Power-On Self Test (POST). The option not found is listed on the menu.

If you have replaced the fixed-disk adapter with the new Extended ESDI Magnetic Media Adapter, you will see the TEST OPTION menu with lines saying:

> The following option was detected previously by the diagnostics,
> but now undetected:
>
> IBM RT PC ESDI Magnetic Media Adapter

Other options or devices you may see are:

- Diskette Drive in position A or B
- R70 Fixed-Disk Drive in position C, D, or E
- IBM RT PC Floating-Point Accelerator Option in Slot D
- IBM RT PC 1MB Memory Expansion Option in Slot B or C
- IBM RT PC 2MB Memory Expansion Option in Slot B or C
- IBM RT PC 4MB Memory Expansion Option in Slot B or C

To remove options, use the following steps.

- If the option or adapter displayed has been removed:

  (a) Type **2** and press <**Enter**> to select "The option has been removed from the system or moved to another slot."

  (b) Repeat the above step if the menu appears again and the option or device has been removed.

- If the option or adapter displayed has not been removed:

  Note: When the device displayed is a diskette drive, use the above steps. Diagnostics will not work correctly unless ID 2 is selected. Do the same for the second diskette drive, if installed.

  (a) Type **1** and press <**Enter**> to select "The option has not been removed from the system or moved to another slot."

  (b) Repeat the above step if the menu appears again and the option or device has not been removed.

If the option of adapter is new, the DIAGNOSTIC TEST LIST menu appears. This menu must be updated to include the new adapter before diagnostics can be run.

If you have installed the new APC and Extended ESDI adapter cards, you will have the following new adapters:

- RT PC Advanced Processor and Memory Management Card
- RT PC Extended ESDI Magnetic Media Adapter

To update the list, use the information you recorded in "STEP 5. INSTALL THE NEW HARDWARE" above.

Check the displayed list against the information you recorded. Note that the displayed list should show the new adapters installed in your system in place of the ones you originally recorded.

Use **Page Down** and **Page Up** to see the complete list.

The following cards will not appear on the new list:

- RT PC Processor and Memory Management in Slot A
- RT PC Floating Point Accelerator in Slot B (if removed)
- Memory card installed in Slot C
- Memory card installed in Slot D (optional)
- RT PC ESDI Magnetic Media Adapter (Slot 1 or 2)

  Note: The DIAGNOSTICS TEST LIST may show two items installed in Slot A: a Floating Point Unit and the Advanced Processor card. The Floating Point Processor appears because of the Floating Point Accelerator installed on the APC. Displaying two items is correct.

  Also note that R70 disks are not listed and cannot be added until after you complete Step 7. After completing Step 7, you should plan to update the list before running the diagnostics again.

(1) If the list is correct, type **1** and press **< Enter >**.

(2) If the list is incorrect, type **2** and press **< Enter >**. Follow the instructions as they appear on the screen until the test list is correct.

(3) When the DIAGNOSTIC SELECTION menu appears, type **1** and press **< Enter >** to select *System Checkout.*

(4) When the TEST METHOD SELECTION menu appears, press **< Enter >** to select *Run Test One Time.* Follow the instructions as they appear on the screen to test the system hardware. Use diskette DIAG-3 when prompted.

(5) If no problems were found, TESTING COMPLETE appears.

a. Press **< Enter >** to return to the DIAGNOSTIC SELECTION menu.

b. Remove the diskette and return all three to the back of the *Problem Determination Guide.*

c. Power off your machine. Then go on to STEP 7 on the next page.

(6) If problems were found, the following message appears:

THERE IS A HARDWARE PROBLEM IN YOUR MACHINE.

If this occurs, do the following:

a. Turn the system power off.

b. Check that all the new adapters and cables are installed properly.

c. Run the diagnostics test again.

d. If the diagnostics fail again, see the *Problem Determination Guide* for complete procedures.

STEP 7. RUN THE IBM/4.3 STANDALONE FORMATTER

If your system *does not have* hd70r disks, skip this step and STEP 8. Go now to STEP 9.

This step converts the existing hd70r disks to hd70e. Follow the instructions below. For more information, see *format*(8R) in Volume I of *IBM Academic Operating System 4.3.*

(1) Insert in the diskette drive the diskette labeled SAUTIL (standalone utilities) that comes with IBM/4.3. If your system has two diskette drives, use the upper drive.

(2) Power on your IBM RT PC. After about 30 seconds, the LED display on the front panel of the RT shows "29," indicating that the sautil program is loading. When it has successfully loaded, it displays the following:

```
4.X BSD UNIX Standalone Maintenance Program $Revision:  X.X$

Choice    Description
1         boot - boot standalone program or kernel
2         format - format hard disk
3         dump - display disk or floppy (hex)
4         cat - display a file contents (ASCII)
5         ls - print directory of UNIX filesystem
6         copy - copy all/part of a disk or floppy
7         debugger - display memory etc.
8         iplsource - set boot order in nvram
9         minidisk - display/change minidisk directory
10        dosboot - boot standalone program or kernel from DOS diskette
11        convert - R70 ("hd70r") disk to an E70

Enter the menu choice desired then press Enter

Choice ?
```

(3) Type **11** and press **<Enter>** to convert the disk. The following information will appear. Sample responses are shown in **bold**. A drive is specified as **hd(unit,2)**, where **unit** is 0, 1, or 2.

Note: Conversion will take around 100 minutes per disk.

```
*** IBM 4.X BSD Standalone Format $Revision: X.X $  ***

Device to format?  hd(0,2)
Formatting drive hd0: verify (yes/no)?  yes
Warning: E70 conversion requires erasing the existing bad block table.
         So you'd be wise to select the "severe burnin" option.

Available test patterns are:

         1 - (f00f) RH750 worst case
         2 - (ec6d) media worst case
         3 - (a5a5) alternate 1's and 0's
         4 - (0) zero disk
         5 - (ffff) Severe burnin (takes several hours)

Select Pattern (one of the above, other to restart) [5]:  5
```

```
How many passes (from 1 to 32) [1]? 1
Device data: #cylinders=1, #tracks=2, #sectors=17

Initializing configuration record for a "hd70e" disk.
writing new configuration record
reopening hd(0,2)
New device data: #cylinders=583, #tracks=7, #sectors=35
Change defaults? [n] n
Attempt to preserve existing data (yes/no) [n] n
Start FORMAT (yes/no)? yes
```

(4) When the disk formatting is finished, press **<Enter>** to return to the menu shown in instruction 2 above.

(5) Repeat instructions 3 and 4 for each of the hd70r disks on your system.

When all of your hd70r disks have been reformatted, go to STEP 8.

STEP 8. RECREATE MINIDISKS

If your system *does not have* hd70e disks or all your charts from STEP 4 read "no minidisks," you can skip this step. Go now to STEP 9.

In this step, you will recreate the minidisks on the hd70e drives. You will need the information you saved in STEP 4.

If your system has non-standard minidisks, you may want to readjust disk space utilization before recreating the minidisks. (If you have standard minidisks, proceed to instruction (1) below.)

To best utilize the disk space, IBM/4.3 minidisks should be at cylinder boundaries and a multiple of cylinder size (245 blocks). Use the following formula to calculate the new minidisk size:

new size = 245 * round(size/245)

where "round" means "round to the nearest integer." Use this size for "newsize" when you recreate the minidisks.

(1) Insert the diskette labeled SAUTIL in the diskette drive. If your IBM RT PC has two diskette drives, use the upper drive.

(2) Press and hold **<Ctrl>-<Alt>-<Pause>**. After about 30 seconds, the LED display on the front panel of the RT shows "29," indicating that the sautil program is loading. When the sautil program has successfully loaded, it displays the following:

```
4.X BSD UNIX Standalone Maintenance Program $Revision: X.X$

Choice   Description
1        boot - boot standalone program or kernel
2        format - format hard disk
3        dump - display disk or floppy (hex)
4        cat - display a file contents (ASCII)
5        ls - print directory of UNIX filesystem
6        copy - copy all/part of a disk or floppy
7        debugger - display memory etc.
8        iplsource - set boot order in nvram
9        minidisk - display/change minidisk directory
10       dosboot - boot standalone program or kernel from DOS diskette
11       convert - R70 ("hd70r") disk to an E70

Enter the menu choice desired then press Enter

Choice ?
```

(3) Type **9** and press **<Enter>** to change the minidisk directory. Reply to the prompts as shown in the following example. (The prompts are shown in *italics*; your replies in **bold.**) Note that you will do this only for the R70 disks you converted.

```
          Typical disk names are hd(unit,2) where unit=0, 1 ,2
          disk      hd(2,2)
          hd(2,2):
          primary and secondary directories both zero - initializing
          number=1 level=1 unused=1 first=0 last=0 bad_block=141589 bad_size=1000

index   iodn   name   date   start   size      type
```

```
0       0      FREE           140     141450     00
```

(4) If you wish to create non-standard minidisks, skip this instruction and proceed with instruction (5) below. If you wish to create standard minidisks, type the following command:

```
> standard
```

Now skip to instruction (8) below.

(5) To align the IBM/4.3 minidisks on cylinder boundaries, type the following command:

```
> create boot 32746 105 01
```

(6) Using information from the charts you completed in STEP 4, type the following command to create the minidisk (skip "boot," if any; you created it in the preceding instruction). Note that you should use the new size you calculated at the beginning of this step, rather than the size you originally recorded.

```
> create name iodn newsize type
```

(7) Repeat the preceding instruction for each minidisk on the disk.

(8) Use the following command to review the minidisk directory:

```
> directory
```

(9) Use the following command to force the minidisk directory to disk:

```
> write
```

(10) Type **quit** and press **<Enter>** to leave the minidisk program and return to the main menu.

(11) Repeat instructions 3 through 10 for each hd70r disk.

See *minidisk*(8R) for more detailed information.

(12) When you have created your minidisks, turn off the machine in preparation for installation of IBM/4.3. Then go to STEP 9.

STEP 9. INSTALL ACADEMIC OPERATING SYSTEM 4.3

You have successfully installed the Upgrade Kit, and are now ready to install IBM/4.3. For step-by-step instructions for installing the operating system, see the article entitled "Installing and Operating IBM/4.3" in Volume II of the manual *IBM Academic Operating System 4.3*.

IBM/4.3-SMM:3-42

New Model Series Upgrade Instructions

December 1987

The IBM 3812 Pageprinter

ABSTRACT

This article provides information for using the IBM 3812 Pageprinter[1] attached to an IBM RT PC and for installing and converting fonts.

The article contains the following chapters and appendices:

1. **Introduction** provides general information about the 3812.
2. **Printer Installation** explains how to install the 3812 on an IBM RT PC running IBM/4.3.
3. **Printing Text Files** describes *pprint*(1), a command used to print text files on the 3812.
4. **Printing Ditroff Files** describes how to print ditroff files on the 3812. Ditroff (device-independent troff) support for the IBM 3812 Pageprinter[2] is a modified version of the Documenter's Workbench[3] ditroff.
5. **Fonts** describes how to install the separately-orderable fonts available from IBM.
6. **Using Code Page Tables** gives information about generating fonts for use with the 3812.

Appendix A. Fonts Available on the 3812 lists the IBM fonts that are available for printing with the 3812.

Appendix B. Code Page Tables lists the characters in each code page.

[1] Hereinafter referred to as "the 3812".

[2] Hereinafter referred to as "ditroff."

[3] Documenter's Workbench is a registered trademark of AT&T Technologies.

1. INTRODUCTION

This article provides information helpful in installing and using the 3812 on the IBM RT PC. The 3812 is a multifunction, nonimpact, tabletop page printer. It provides cut-sheet, letter-quality text and all-points-addressable graphics at a maximum rate of 12 pages per minute and a resolution of 240 points per inch. The 3812 attaches to an asynchronous port on an IBM RT PC. The printer can serve users on that machine or other machines on the network. The standard spooling system commands are used to send files to the printer, query their status, and cancel them (see *lpr*(1), *lprm*(1), *lpq*(1), and *lpc*(8)).

Several manual pages describe support for the 3812:

ap(4)	Defines the asynchronous line protocol that supports the 3812
asy(4)	Describes the multi-port asynchronous communications interface
cvt3812(8)	Converts IBM 3820 and 3800 fonts to 3812 format
font3812(5)	Defines the 3812 font structures
ibm3812pp(8)	Describes the 3812 print server daemon
pic(1)	Draws simple pictures on the 3812
pprint(1)	Prints text files on the 3812
ppt(8)	Describes the spooling system filter
printer3812(5)	Defines printer status information
psp(4)	Describes the planar serial port interface
ptroff(1)	Prints ditroff files on the 3812
width3812(8)	Builds table widths for 3812 fonts

2. PRINTER INSTALLATION

2.1. Hardware Requirements

To connect the 3812 to an IBM RT PC, you must have the following:

- A 3812 with an IBM 3812 PC diskette
- An IBM RT PC running IBM/4.3
- An RS232C 4-port asynchronous adapter if installing on an IBM 6151
- An IBM 3812 Printer Cable, part number 1348421 and an IBM RT PC Modem Cable, part number 6298240

  OR
- An IBM RT PC Printer cable, part number 6298525, option number 6294803

  (Note that the single IBM RT PC Printer Cable replaces both the IBM 3812 Printer Cable and the IBM RT PC Modem Cable.)

2.1.1. 3812 Setup

This section describes 3812 setup.

2.1.1.1. DIP Switches

On the inside of the back panel of the 3812 is the set of printer DIP switches. To run at 19200 baud, switches three (3) and seven (7) must be on; the rest must be off.

3 = asynchronous data communications mode

7 = 19200 baud

To run at other than 19200 baud, consult the *IBM 3812 Pageprinter Programming Reference* (S544-3268) for proper DIP switch settings.

2.1.1.2. 3812 Diskette

Inside the front cover of the IBM 3812 Pageprinter is a diskette drive. Load the diskette labeled "IBM 3812 PC"or "IBM 3812 RT" into the drive.

2.1.2. Connection between 3812 and IBM RT PC

If you are using two separate cables, follow these steps:

(1) Connect the 3812 Printer Cable to the plug labeled "RS232C" in the back of the 3812.

(2) Connect the other end of the 3812 Printer Cable to the Modem Cable.

(3) Connect the Modem Cable to one of the serial ports on the back of the IBM RT PC.

If you are using the single IBM RT PC printer cable, follow these steps:

(1) Connect the IBM RT PC printer cable to the plug labeled "RS232C" in the back of the 3812.

(2) Connect the other end of the cable to one of the serial ports on the back of the IBM RT PC.

Throughout this article, we will assume the cable is connected to the port corresponding to */dev/tty00*. Refer to *asy*(4) and *psp*(4) for other possibilities.

2.2. Software Requirements

This section describes the steps required to identify the 3812 to the system.

2.2.1. Verifying Device Files and Entries

Verify that the special file for the printer (*/dev/tty00*) is read/write and owned by daemon. To do so, type the following:

```
chown daemon /dev/tty00
ls -l /dev/tty00
```

The result should look like this:

```
crw------- 1 daemon    1,  0 Feb  3 15:05 /dev/tty00
```

Verify that the entry for *tty00* in the file */etc/ttys* (*ttys*(5)) is correct. To do so, use the following command:

```
grep tty00 /etc/ttys
```

The result should be of the form:

```
tty00  "/etc/getty std.19200"  printer       off
```

If the result you get is different, you must edit the /etc/ttys so that the entry for *tty00* looks like the entry above.

2.2.2. Establishing a Symbolic Link

Establish a symbolic link from */dev/pp* to */dev/tty00.* To do so, use the following command:

```
ln -s /dev/tty00 /dev/pp
```

2.2.3. Verifying Printcap Entries

To identify the 3812 to the spooling system, the proper entries must be in */etc/printcap.* As distributed, IBM/4.3 has the entries described here; you may need to modify them for local use.

Three sample printcap entries are shown below. The first defines a 3812 attachment on a remote IBM RT PC. The name of the remote host must be substituted for the words *remote 3812 print server name.* The second entry defines a 3812 attachment on a local IBM RT PC. The third entry selects one of the two previous entries. It currently points to the entry for a 3812 attached to a remote host. This must be changed to refer to the local entry (lpp) when installing the 3812 locally.

```
#
# Sample Printcap Entries for 3812
#
# Sample for Remote 3812 Pageprinter
#
rpp|remote 3812 Pageprinter:\
            :lp=:rm=remote 3812 print server name:rp=pp:sd=/usr/spool/rppd
#
# Sample for Locally Attached 3812 Pageprinter, connected at /dev/pp
#
lpp|local 3812 Pageprinter:\
            :lp=/dev/null:sd=/usr/spool/ppd:\
            :lf=/usr/adm/ppd-errs:\
            :af=/usr/adm/acct-pp:\
            :PP=/dev/pp:\
            :SS=/usr/spool/ppd/status3812:\
            :br#19200:\
            :sh:\
            :if=/usr/lib/p3812/txt3812:\
            :nf=/usr/lib/p3812/dit3812:\
            :vf=/usr/lib/p3812/pmp3812:\
            :pl#66:pw#80:px#2040:py#2640
#
# Change the rpp to lpp in the next line if the 3812 is attached locally
#
pp|3812 Pageprinter:\
            :tc=rpp:
```

Refer to *printcap*(5) for a complete description of the printcap file.

The spool directory (*sd*), accounting file (*af*), and log file (*lf*) must be created read/write by daemon. To do so, issue the following commands as root:

```
cd /etc
/usr/src/etc/printcap.install
```

Note that the filename at the end of the path in the accounting file entry (*af*) must end with a "-" followed by the printer name. In this example, the name is "pp".

Two new printcap capabilities, SS and PP, are defined for the 3812. The SS entry identifies the printer log file for the 3812. The current status message from the printer will be written to this log. It will contain the intervention-required messages such as "PAPER JAM," "OUT OF PAPER,"or "TONER LOW." A list of printer messages can be found in *printer3812*(5).

The PP entry identifies the special file where the 3812 is attached. It is usually a symbolic link to a specific tty port. This information is not provided by the line printer (*lp*) entry. The spooling system does not open or close the device where the printer is attached. Instead, the device is managed by the 3812 print server (*ibm3812pp*(8)), using a special asynchronous line protocol (*ap*(4)) that supports the 3812 printer. The *ap* line discipline is conditionally compiled and must be specified in the kernel configuration file if the 3812 is to be supported. The 3812 print server runs as a daemon and is started by *ppt*(8) when the first file is sent to the printer.

The *sh* entry indicates that spooling system header pages are suppressed. The individual filters (*if*, *nf*, *vf*) will print header pages if requested.

The following filters are specified in this printcap entry: the text filter (*if*) converts ASCII data to printer commands; the Ditroff filter (*nf*) converts device-independent troff output to Page Map Primitive (PMP) commands; the PMP filter (*vf*) sends PMP commands to the printer. PMP commands provide for vector graphics, font downloading, and all-points-addressable printing on the 3812. PMP is described in the manual entitled *IBM 3812 Pageprinter Programming Reference*, S544-3268.

2.3. Starting the 3812

Once you have completed the steps described thus far, you are ready to print your first document. Turn the printer on and check that paper is loaded in both paper cassettes. The 3812 uses the alternate paper cassette to print document separator pages; using colored paper here will make output separation easier. After a few minutes, a '701' will appear on the printer display, indicating that the printer is ready.

The print server (*ibm3812pp*(8)) starts when the first job is sent to be printed. To start the print server, for example, you can print a copy of */etc/printcap*:

```
pprint /etc/printcap
```

The '701' disappears from the printer display, the online light appears, and the document is printed.

2.4. Monitoring the 3812

The current status of the printer is recorded in the spool directory in the *status3812* file (SS in */etc/printcap*). The following command displays the current status:

```
cat /usr/spool/ppd/status3812
```

Error messages are recorded in the log file (*lf* in */etc/printcap*). The following command monitors the 3812 error log while documents are printing:

```
tail -f /usr/adm/ppd-errs
```

The error log (*lf*) and accounting file (*af*) will grow as documents are printed, and should be truncated periodically by your system administrator.

The print server (*ibm3812pp*(8)) runs as a daemon. It can be terminated by issuing the *kill*(1) command from root.

3. PRINTING TEXT FILES

Pprint(1) prints text files on the 3812. It filters input through *pr* and sends the resulting output to the spooling system. See the *pprint*(1) manual page for the various options. The file */etc/printcap* must contain an entry for the 3812 as described in Section 2.2.3, "Verifying Printcap Entries."

See Appendix A of this article for a list of fonts that can be specified with the **-f** option of *pprint*.

4. PRINTING DITROFF FILES

Ditroff includes modifications to the *troff*, *eqn*, *pic*, and *makedev* binaries in Documenter's Workbench. These binaries have been included in the IBM/4.3 package. A separate Documentor's Workbench license must be obtained form AT&T to order the source.

These programs have been modified to use the 3812 fonts (**-T3812**); otherwise, the documentation for *troff*, *pic*, and *eqn* has not changed. The output file generated by *troff* has not been redefined. Only the format of the width tables has been changed. The width tables in Documenter's Workbench contain widths for only one character size, size 12. The width of a character in other sizes is calculated by scaling the base width: if the A in size 12 is 32 pels, the A in size 6 is 16 pels, and the A in size 24 is 64 pels. The 3812 fonts are designed individually for each size, so the width tables have been augmented to contain the width of every character in every size.

A new command *ptroff*(1) processes *troff* source for printing on the 3812. See the *ptroff*(1) manual page for the various options. The file */etc/printcap* must contain an entry for the 3812 as described in Section 2.2.3, "Verifying Printcap Entries." See Appendix A of this article for a list of fonts that are available for use with *troff*, and Appendix B for a list of *troff* character names.

Because of the changes to the width tables, use */usr/ibm/troff*, */usr/ibm/pic*, and */usr/ibm/eqn* with **-T3812**. You can expect errors if you use */usr/bin/troff* or */usr/bin/eqn* with **-T3812**.

4.1. Setting Up Ditroff for Use with a Remote 3812 Print Server

The ditroff feature is set up so that each workstation can act as a print server. The following changes will allow users to run ditroff on their local RT PC and send files to be printed on the RT PC with the 3812 attached. Both the local and the print server IBM/4.3 system must be running IBM/4.3.

Modify your */etc/printcap* to indicate the name of a workstation with a 3812 attached. See Section 2.2.3, "Verifying Printcap Entries."

The font raster files in */usr/lib/font/dev3812/fonts* need only be kept on the RT PC with the printer attached because they are used only at the time of printing. To remove them, type

```
rm /usr/lib/font/dev3812/fonts/*
```

If the font raster files are changed (as they are when the fonts are rebuilt), the width tables (*/usr/lib/font/dev3812/*.out*) must be rebuilt and redistributed to remote workstations. **It is essential** that the remote width tables match the width tables on the 3812 print server.

The command *install.ditroff* provides an easy way to redistribute the ditroff binaries and width tables, if necessary. To update ditroff on *ws1*, *ws2*, and *ws3*, your account must have write access to directories */usr/ibm* and */usr/lib/font/dev3812* on the remote workstations. Enter the following commands on the print server workstation:

```
cd /usr/ibm
install.ditroff  ws1 ws2 ws3
```

This will copy the following files to the remote workstation(s):

```
/usr/ibm/ptroff
/usr/ibm/troff
/usr/ibm/eqn
/usr/ibm/pic
/usr/lib/font/dev3812/*.out
/usr/man/man1/pic.1
/usr/man/man1/ptroff.1
```

5. FONTS

The 3812 uses either the fonts provided on the 3812 diskette or fonts that are downloaded from the host. There are two sets of loadable fonts: a set of uniformly-spaced fonts is provided with IBM/4.3, and several typographical fonts can be ordered from IBM as separate products. This chapter describes both groups of fonts.

The Ditroff feature can be used with either the uniformly-spaced fonts or the typographical fonts. The uniformly-spaced fonts can be used to create documents which are near letter quality. These fonts have been installed for immediate use with the Ditroff binaries. To produce letter quality documents, it is necessary to order and install the typographical fonts.

5.1. Typographic Fonts

5.1.1. Ordering Typographic Fonts

The available typographic fonts include:

Sonoran Serif,[4] Program Number 5669-161, Feature 5124
Sonoran Sans Serif,[5] Program Number 5669-162, Feature 5125
Pi & Special,[6] Program Number 5669-163, Feature 5126

These fonts are shipped on diskettes. Samples of each font are shown in the *IBM 3800 Printing Subsystem Model III Font Catalog*, SH35-0053, and in Appendix A of this article.

5.1.2. Installing Typographic Fonts

(1) Login as root.

(2) Load the fonts from diskette.

There are three subdirectories in */usr/src/usr.lib/font/dev3812/typographic* that contain makefiles and tables for building fonts. Use *dosread*(1) to load the fonts from diskette to the IBM RT PC into the appropriate subdirectory.

(3) To load the Sonoran Serif font (Program Number 5669-161, Feature 5124), follow these steps:

```
cd /usr/src/usr.lib/font/dev3812/typographic/serif
```

Insert the first diskette of the Sonoran Serif font into the diskette reader, then type the following:

```
dosread
```

Repeat this step for all 12 diskettes in the Sonoran Serif font.

To load the Sonoran Sans Serif font (Program Number 5669-162, Feature 5125), follow these steps:

```
cd /usr/src/usr.lib/font/dev3812/typographic/sans
```

[4] Functional equivalent of Monotype Times New Roman, a trademark of The Monotype Corporation, Limited. Contains data derived under license from The Monotype Corporation, Limited.

[5] Functional equivalent of Monotype Ariel, a trademark of The Monotype Corporation, Limited. Contains data derived under license from The Monotype Corporation, Limited.

[6] Contains data derived under license from The Monotype Corporation, Limited.

Insert the first diskette of the Sonoran Sans Serif font into the diskette reader, then type the following:

```
dosread
```

Repeat this step for all 12 diskettes in the Sonoran Sans Serif font.

To load the Pi & Special font (Program Number 5669-163, Feature 5126), follow these steps:

```
cd /usr/src/usr.lib/font/dev3812/typographic/pispecial
```

Insert the first diskette of the Pi & Special font into the diskette reader, then type the following:

```
dosread
```

Repeat this step for both of the diskettes in the Pi & Special font.

Loading all the fonts from diskette takes approximately twenty-five minutes.

(4) Edit the *Makefile* in */usr/src/usr.lib/font/dev3812/typographic* to added the subdirectories *sans*, *serif* and *pispecial* to the *SUBDIRS*, so that they will get made automatically.

Change the line

```
SUBDIRS = Afont
```

to

```
SUBDIRS = Afont sans serif pispecial
```

The *Makefile* in */usr/src/usr.lib/font/dev3812/typographic* is now set up to descend into the typographical subdirectories and performs the makes on them.

(5) Convert the fonts into 3812 format.

Make automatically converts the fonts into the 3812 format (*cvt3812*(8)), and builds the width tables (*width3812*(8)). Type the following:

```
cd /usr/src/usr.lib/font/dev3812/typographic
make all
```

This step takes approximately thirty minutes.

(6) Install the fonts for use with the 3812.

Make with the install option does the following:

- Copies the width tables to */usr/lib/font/dev3812*.
- Copies the font raster pattern to */usr/lib/font/dev3812/fonts*.

To install, type the following:

```
cd /usr/src/usr.lib/font/dev3812/typographic
make install
```

At this point the typographic fonts have been installed in */usr/lib/font/dev3812/typographic*, and are available for use with *pprint*(1) for printing on the 3812.

(7) The typographical fonts are now ready for use with *pprint*.

To install the typographic fonts for use with Ditroff, do the following:

```
cd /usr/src/usr.lib/font/dev3812
make installwidths
```

(8) Once you have determined that the fonts have been properly installed, you will probably want to clean up the font directories to free up some space.

To clean the font directories, type the following:

```
cd /usr/src/usr.lib/font/dev3812
make clean
```

5.2. Uniformly-spaced Fonts

These fonts have been installed in */usr/lib/font/dev3812* and */usr/lib/font/dev3812/fonts* and are available for use with both *ptroff*(1) and *pprint*(1). They are shown in the *IBM 3812 Pageprinter Introduction and Planning Guide*, G544-3265, and the *IBM 3800 Printing Subsystem Model III Font Catalog*, SH35-0053. Their format is described in *font3812*(5).

As shipped, IBM/4.3 includes substitutes for the typographical fonts. These substitute fonts were built from the uniformly-spaced fonts. This allows the user to set up an IBM 3812 Pageprinter and immediately print *troff* documents. Output created when using these substitute font is near letter quality. To obtain true letter-quality output, the user must order the typographical fonts.

The IBM/4.3 package is set up to use these fonts with the Ditroff binaries.

10-pitch	12-pitch	Other fixed-pitch	Proportionally-spaced
APL	Courier *	13-pitch Letter Gothic	Barak
Courier *	Gothic Bold	15-pitch Gothic Text	Boldface *
Courier Italic	Gothic Italic	15-pitch Serif Text	Boldface Italic *
Gothic Bold	Gothic Text	15-pitch Shalom	Document *
Gothic Text	Letter Gothic *	20-pitch Shalom	Essay *
Katakana	Letter Gothic Bold *	20-pitch APL	Essay Bold *
Orator	Prestige Elite *	20-pitch APL	Essay Italic *
Orator Bold	Prestige Elite Bold *	20-pitch Gothic Text	Essay light *
Prestige Pica *	Prestige Elite Italic *	27-pitch Gothic Text	Gothic Tri-pitch
Roman Text	Script	15-pitch Format	
Serif Italic	Serif Bold		
Serif Text	Serif Italic		
Shalom	Shalom		
Math Symbol	Math Symbol		
OCR-A **	Format		
OCR-B **			
Format			

* These fonts contain international character sets of 221 printable characters.

** The 3812 prints the OCR-A and OCR-B fonts with the same high quality as other type styles. IBM does not warrant and has not tested that these characters are readable by all OCR reading devices. Users of these fonts should test read compatibility before relying on the 3812 for OCR applications.

6. USING CODE PAGE TABLES

A code page table identifies the set of characters to be used from one or more fonts. A font contains the graphic pel pattern for each character along with its IBM character name. An entry in a code page table associates an IBM character name with a *troff* character name. The entry's position in the table gives the character's ASCII code.

Lowercase "a", for example, has the IBM character name LA010000 regardless of font or point size, and would be represented by a code page table entry containing "a LA010000". The ASCII code is 97, so the entry is the 97th unique table entry.

Similarly, the pound symbol (£), if it is in the table at all, is represented by an entry containing "£ SC020000". Since it has no ASCII code, the entry may appear in any unused position.

A code page table may contain up to 256 unique entries, plus synonym entries. Appendix A of this article gives a list of the fonts and their corresponding code page tables; Appendix B lists the information appearing in the code page tables.

Two examples are given here. The first example shows how to modify an existing code page table for use with ditroff. The second shows how to make a new code page table.

Example 1

If you want to change the troff name assigned to a character, or if you want to change the graphic associated with a troff name, you must modify the code page table accordingly. For example, to change the troff names of left double quote \(LQ and right double quote \(RQ to \(L" and \(R" respectively, you would change the troff names in code page table *stdcp*. Then you would need to rebuild the width tables for all fonts that use that code page table and copy the resulting width tables to */usr/lib/font/dev3812*. Copy the resulting code page index files to */usr/lib/font/dev3812/fonts*.

In this example, R, I, B, BI, H, HI, HB, HY, D, and SP use the *stdcp* code page table and need to be rebuilt. You would rebuild the width tables by typing the following:

```
cd /usr/src/usr.lib/font/dev3812/typographic/serif
make serif
cp R I B BI /usr/lib/font/dev3812
cp *.stdcp /usr/lib/font/dev3812/fonts

cd /usr/src/usr.lib/font/dev3812/typographic/sans
make sans
cp H HI HB HY /usr/lib/font/dev3812
cp *.stdcp /usr/lib/font/dev3812/fonts

cd /usr/src/usr.lib/font/dev3812/typographic/pi
make special
cp D SP /usr/lib/font/dev3812
cp *.stdcp /usr/lib/font/dev3812/fonts
```

Next, build the binary form of the widths. This step depends on having the ditroff feature installed. Go to */usr/lib/font/dev3812* and build the binary form of the troff width tables:

```
cd /usr/lib/font/dev3812
makedev DESC
```

Now the new character name is ready to be used by *troff*.

Example 2

The APL font is built using the *fcp* code page table, but there are IBM character names in the APL font that are not referenced by *.fcp.* To determine all the IBM character names in a font use the -N option on *cvt3812*(8). The following command generates all the IBM names for characters in the APL font:

```
cvt20to12 -N c0s0ae10 > /tmp/names
```

The IBM character names are documented in the *IBM 3800 Printing Subsystem Model III Font Catalog*, SH35-0053. Each element of */tmp/names* has the following format:

```
raster for x xx SL110000
```

SL11000 is the IBM character name. The "xx" in the list of character names is replaced with a *troff* character name. For example, the following will assign the name \(CS to the APL character Circle Star:

```
raster for x CS SL110000
```

Copy *fcp* to *aplcp* and add the new entries from */tmp/names* to *aplcp* into unused positions. Once the characters have been assigned a name, use *width3812*(8) to build the width table and the code page index files:

```
width3812 -s 5 10 0 -c aplcp -n AP APL
```

This will build the AP width tables for sizes 5 and 10, using the code page table *aplcp.* The width table (*AP*) must be copied to */usr/lib/font/dev3812* to be used by *troff.* The code page index files (*APL.10.aplcp* and *APL.5.aplcp*) must be copied to */usr/lib/font/dev3812/fonts.* This file can now be used with *pprint*(1). The following command will print a file using the APL font.

```
pprint -fAPL.10.aplcp filename
```

To use the font in a *troff* document, use *makedev* to generate the binary form of the troff width table:

```
cd /usr/lib/font/dev3812
makedev AP
```

Now use the *troff* commands for changing fonts (.ft AP) to print characters from the font.

APPENDIX A. FONTS AVAILABLE ON THE 3812

This appendix shows examples of the fonts available on the 3812. The typographic fonts are available for license from IBM; the uniformly-spaced fonts are provided with IBM/4.3.

Typographic Fonts

The following fonts are in the Sonoran type face, except for the Display and Petite fonts. Unless otherwise indicated, all fonts are available in point sizes 6, 7, 8, 9, 10, 11, 12, 14, 16, 18, 20, 24, 30, and 36. Samples of these fonts are found later in this section.

Sonoran Serif Typeface

ptroff Name	*pprint* Name	Point Size	Code Page Table	Notes	Font Name
R	s	All	stdcp	(1)	Serif Roman
I	s.I	All	stdcp	(1)	Serif Italic
B	s.B	All	stdcp	(1)	Serif Bold
BI	s.BI	All	stdcp	(1)	Serif Bold Italic
S	-	6-12	picp	(2)	Serif PI (symbols)
SB	-	6-12	picp	(2)	Serif PI Bold (symbols)
L	-	All	stdcp		Serif Roman Latin Characters
LI	-	All	stdcp		Serif Italic Latin Characters
LB	-	All	stdcp		Serif Bold Latin Characters
LY	-	All	stdcp		Serif Bold Italic Latin Characters
A	-	6-12	addcp	(2)	Serif Additional Characters

Sonoran Sans Serif Typeface

ptroff Name	*pprint* Name	Point Size	Code Page Table	Notes	Font Name
H	ss	All	stdcp	(1)	Sans Serif Roman
HI	ss.I	All	stdcp	(1)	Sans Serif Italic
HB	ss.B	All	stdcp	(1)	Sans Serif Bold
HY	ss.BI	All	stdcp	(1)	Sans Serif Bold Italic
HS	-	6-12	picp	(2)	Sans Serif PI (symbols)
HZ	-	6-12	picp	(2)	Sans Serif PI Bold (symbols)
K	-	All	stdcp		Sans Serif Roman Latin Characters
KI	-	All	stdcp		Sans Serif Italic Latin Characters
KB	-	All	stdcp		Sans Serif Bold Latin Characters
KY	-	All	stdcp		Sans Serif Bold Italic Latin Characters

Notes:

(1) When using *pprint*, these fonts lack the ~ and ^ characters.

(2) Font sizes 7, 9, and 11 are duplicates of 6, 8, and 10 respectively.

Special Fonts

ptroff Name	*pprint* Name	Point Size	Code Page-Table	Notes	Font Name
D	d	20, 36	stdcp	(1)	Display
SP	pe	4	stdcp	(2)	Petite

Notes:

(1) When using *pprint*, the Display font lacks the following characters:

+ < = > @ [ ] ^ _ { } ~

(2) When using *pprint*, the petite font lack the ~ and ^ characters.

Uniformly-Spaced Fonts

The following fonts are provided for use with *pprint*(1) and *ptroff*(1) and are built with the *fcp*, *symcp* or *acp* code page table. Not all fonts contain a complete set of ASCII characters. Samples of these fonts are found in the *IBM 3800 Printing Subsystem Model III Font Catalog*, SH35-0053.

ptroff Name	*pprint* Name	Point Size	Code Page Table	Notes	Font Name
AP	APL	5, 10	fcp		APL Characters
BR	BARAK.B	10	fcp		Barak Bold
Bb	BOOK.B	10	fcp		Book Bold (proportionally spaced)
Bi	BOOK.BI	10	fcp		Book Bold Italic (proportionally spaced)
CW	COURIER	9, 10	fcp		Courier 10
Ci	COURIER.I	10	fcp		Courier 10 Italic
Cw	Courier	4, 10	acp	(1)	Courier
Cb	Courier.B	10	acp	(1)	Courier Bold
Cc	Courier.C	10	acp	(1)	Courier Condensed
Cd	Courier.CB	10	acp	(1)	Courier Condensed Bold
Ce	Courier.E	10	acp	(1)	Courier Expanded
Cf	Courier.EB	10	acp	(1)	Courier Expanded Bold
Du	DOCUMENT	10	fcp		Document (proportionally spaced)
E	ESSAY	10	fcp		Essay (proportionally spaced)
EB	ESSAY.B	10	fcp		Essay Bold (proportionally spaced)
EI	ESSAY.I	10	fcp		Essay Italic (proportionally spaced)
EL	ESSAY.L	10	fcp		Essay Light (proportionally spaced)
F	FORMAT	8, 9, 10	fcp		Formatting characters
G	GOTHIC	4, 5, 8, 9, 10	fcp		Gothic
GB	GOTHIC.B	9, 10	fcp		Gothic Bold
GI	GOTHIC.I	9	fcp		Gothic Italic
GP	GOTHICP	9	fcp		Gothic Tri-pitch (proportionately spaced)
KT	KATAKANA	10	fcp		Katakana
Lr	LETTER	9	fcp		Letter Gothic
Lb	LETTER.B	9	fcp		Letter Gothic Bold
RA	OCRA	10	fcp		OCR-A
RB	OCRB	10	fcp		OCR-B
O	ORATOR	10	fcp		Orator
OB	ORATOR.B	10	fcp		Orator Bold
PP	PRESTIGE	9, 10	fcp		Prestige
PB	PRESTIGE.B	9	fcp		Prestige Bold
PI	PRESTIGE.I	9	fcp		Prestige Italic
RM	ROMAN	10	fcp		Roman Text
SC	SCRIPT	9	fcp		Script
Sf	SERIF	8, 9, 10	fcp		Serif
Sb	SERIF.B	9	fcp		Serif Bold
Si	SERIF.I	9, 10	fcp		Serif Italic
SH	SHALOM	8, 9, 10	fcp		Hebrew 10-pitch
TX	TEXT	10	fcp		Serif Text
SY	SYMBOLS	9, 10	symcp		Math Symbols

Notes:

(1) These fonts are IBM 5152 Printer Emulation fonts.

Examples of Typographic Fonts

The Sonoran fonts shown here are printed in 10-point sizes, with a vertical spacing of 12 points and with non-alphanumeric characters separated by 1/4-em space. These examples are representative selections of the available characters in each font; not every available character is shown. See Appendix B for the code page tables for each font.

Serif Fonts

Serif Roman (R)

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890
! $ % & ( ) ‘ ’ * + − . , / : ; = ? [ ] |
• – - — ¼ ½ ¾ ﬁ ﬂ ﬀ ﬃ ﬄ
° † ′ ¢ ” ′ \ _ ‘ / < > { }
@ + * § ¬ ‡ » «

Serif Bold (B)

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890
! $ % & ( ) ‘ ’ * + − . , / : ; = ? [ ] |
• – - — ¼ ½ ¾ ﬁ ﬂ ﬀ ﬃ ﬄ
° † ′ ¢ ” ′ \ _ ‘ / < > { }
@ + * § ¬ ‡ » «

Serif Italic (I)

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890
! $ % & ( ) ‘ ’ * + − . , / : ; = ? [ ] |
• – - — ¼ ½ ¾ ﬁ ﬂ ﬀ ﬃ ﬄ
° † ′ ¢ ” ′ \ _ ‘ / < > { }
@ + * § ¬ ‡ » «

Serif Bold Italic (BI)

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890
! $ % & ( ) ‘ ’ * + − . , / : ; = ? [ ] |
• – - — ¼ ½ ¾ ﬁ ﬂ ﬀ ﬃ ﬄ
° † ′ ¢ ” ′ \ _ ‘ / < > { }
@ + * § ¬ ‡ » «

Serif PI - symbols (S)

α β γ δ ε ζ η θ ι κ λ μ
ν ξ ο π ρ σ τ υ φ χ ψ ω
Γ Δ Θ Λ Ξ Π Σ Υ Φ Ψ Ω
√ ‾ ≥ ≤ ≡ ~ ≅ ≠ → ←
↑ ↓ × ÷ ± ∪ ∩ ⊂ ⊃ ∞
∂ ® © ^ ˜ — = ∫ ∝ ∅ ∈
☎ ○ □

Serif PI Bold - symbols (SB)

α β γ δ ε ζ η θ ι κ λ μ
ν ξ ο π ρ σ τ υ φ χ ψ ω
Γ Δ Θ Λ Ξ Π Σ Υ Φ Ψ Ω
√ ‾ ≥ ≤ ≡ ~ ≅ ≠ → ←
↑ ↓ × ÷ ± ∪ ∩ ⊂ ⊃ ∞
∂ ® © ^ ˜ — = ∫ ∝ ∅ ∈
☎ ○ □

Serif Additional Characters (A)

⎛ ⎝ ⎞ ⎠ ⎨ ⎬ ⎪ ⎣ ⎦ ⎡ ⎤ ς ∇ ⊆ ⊇

Sans Serif Fonts

Sans Serif Roman (H)

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890
! $ % & ( ) ' ' * + − . , / : ; = ? [ ] |
• — - — ¼ ½ ¾ fi fl ff ffi ffl
° † ′ ¢ " ′ \ _ ' / < > { }
@ + * § ¬ ‡ » «

Sans Serif Italic (HI)

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890
! $ % & ( ) ' ' * + − . , / : ; = ? [ ] |
• — - — ¼ ½ ¾ fi fl ff ffi ffl
° † ′ ¢ " ′ \ _ ' / < > { }
@ + * § ¬ ‡ » «

Sans Serif Pl - symbols (HS)

α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ σ τ
υ φ χ ψ ω
Γ Δ Θ Λ Ξ Π Σ Υ Φ Ψ Ω
√ ¯ ≥ ≤ ≡ ~ ≅ ≠ → ←
↑ ↓ × ÷ ± ∪ ∩ ⊂ ⊃ ∞
∂ ® © ^ ˜ — = ∫ ∝ ∅ ∈
☎ ○ □

Sans Serif Bold (HB)

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890
! $ % & ( ) ' ' * + − . , / : ; = ? [ ] |
• — - — ¼ ½ ¾ fi fl ff ffi ffl
° † ′ ¢ " ′ \ _ ' / < > { }
@ + * § ¬ ‡ » «

Sans Serif Bold Italic (HY)

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890
! $ % & ( ) ' ' * + − . , / : ; = ? [ ] |
• — - — ¼ ½ ¾ fi fl ff ffi ffl
° † ′ ¢ " ′ \ _ ' / < > { }
@ + * § ¬ ‡ » «

Sans Serif Pl Bold - symbols (HZ)

α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ σ
τ υ φ χ ψ ω
Γ Δ Θ Λ Ξ Π Σ Υ Φ Ψ Ω
√ ¯ ≥ ≤ ≡ ~ ≅ ≠ → ←
↑ ↓ × ÷ ± ∪ ∩ ⊂ ⊃ ∞
∂ ® © ^ ˜ — = ∫ ∝ ∅ ∈
☎ ○ □

Special Fonts

Two special fonts are available. Both are built with code page table *stdcp*.

The Display font is printed in 20 points, with a vertical spacing of 30 points and with non-alphanumeric characters separated by 1/4-em space. This font is available only in point sizes 20 and 36.

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890
! $ % & ( ) ‘ ’ * - “ ” . , / : ; ?

The Petite font is printed in 4 points, with a vertical spacing of 6 points and with non-alphanumeric characters separated by 1/4-em space. This font is available only in a 4-point size.

abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890
! $ % & ( ) ' ' * + - . , / : ; = ? [ ] |

Latin Characters

The Latin and special characters shown below are available in both the Sonoran Serif and Sans Serif font families in roman, italic, bold, and bold italic. They are built with code page table *stdcp*.

Serif Roman Latin Character (L)

á Á à À â Â ä Ä Ã å Å æ Æ ç Ç é É è È ê Ê ë
Ë ě Ě í Í ì Ì î Î ï Ï ĩ Ĩ ĳ ŀ Ŀ ń Ń ñ Ñ ň Ň
ó Ó ò Ò ô Ô ö Ö õ Õ œ Œ ø Ø ß ú Ú ù Ù û Û ü
Ü ũ Ũ ů Ů ÿ Ÿ ± < = > ÷ × ≤ ≥ ≠ ¤ ¥ ₧ § ¶
‰ ¡ ( ) , - · ¿ „ 🄽

Sans Serif Roman Latin Character (K)

á Á à À â Â ä Ä Ã å Å æ Æ ç Ç é É è È ê Ê ë
Ë ě Ě í Í ì Ì î Î ï Ï ĩ Ĩ ĳ ŀ Ŀ ń Ń ñ Ñ ň Ň
ó Ó ò Ò ô Ô ö Ö õ Õ œ Œ ø Ø ß ú Ú ù Ù û Û ü
Ü ũ Ũ ů Ů ÿ Ÿ ± < = > ÷ × ≤ ≥ ≠ ¤ ¥ ₧ § ¶
‰ ¡ ( ) , - · ¿ „ 🄽

APPENDIX B. CODE PAGE TABLES

This appendix lists the characters in each code page, with their *troff*, IBM, and descriptive names. Refer to files in */usr/src/usr.lib/font/dev3812/fonts/*cp* for precise code page contents.

STANDARD CODE PAGE (stdcp)

	Troff	IBM	
	Char	Char	
Char	Name	Name	Description
a	a	LA010000	a small
A	A	LA020000	A capital
á	\(a’	LA110000	a acute small
Á	\(A’	LA120000	A acute capital
à	\(a‘	LA130000	a grave small
À	\(A‘	LA140000	A grave capital
â	\(a^	LA150000	a circumflex small
Â	\(A^	LA160000	A circumflex capital
ä	\(a:	LA170000	a diaeresis/umlaut small
Ä	\(A:	LA180000	A diaeresis/umlaut capital
ã	\(a~	LA190000	a tilde small
Ã	\(A~	LA200000	A tilde capital
å	\(a.	LA270000	a overcircle small
Å	\(A.	LA280000	A overcircle capital
æ	\(ae	LA510000	ae diphthong small
Æ	\(AE	LA520000	AE diphthong capital
b	b	LB010000	b small
B	B	LB020000	B capital
c	c	LC010000	c small
C	C	LC020000	C capital
ç	\(c,	LC410000	c cedilla small
Ç	\(C,	LC420000	C cedilla capital
d	d	LD010000	d small
D	D	LD020000	D capital
e	e	LE010000	e small
E	E	LE020000	E capital
é	\(e’	LE110000	e acute small
É	\(E’	LE120000	E acute capital
è	\(e‘	LE130000	e grave small
È	\(E‘	LE140000	E grave capital
ê	\(e^	LE150000	e circumflex small
Ê	\(E^	LE160000	E circumflex capital
ë	\(e:	LE170000	e diaeresis/umlaut small
Ë	\(E:	LE180000	E diaeresis/umlaut capital
ě	\(e-	LE210000	C caron small
Ě	\(E-	LE220000	E caron capital
f	f	LF010000	f small
F	F	LF020000	F capital
ﬀ	\(ff	LF510000	ff ligature
ﬁ	\(fi	LF530000	fi ligature
ﬂ	\(fl	LF550000	fl ligature
ﬃ	\(Fi	LF570000	ffi ligature
ﬄ	\(Fl	LF590000	ffl ligature

g	g	LG010000	g small
G	G	LG020000	G capital
h	h	LH010000	h small
H	H	LH020000	H capital
i	i	LI010000	i small
I	I	LI020000	I capital
í	\(i’	LI110000	i acute small
Í	\(I’	LI120000	I acute capital
ì	\(i‘	LI130000	i grave small
Ì	\(I‘	LI140000	I grave capital
î	\(i^	LI150000	i circumflex small
Î	\(I^	LI160000	I circumflex capital
ï	\(i:	LI170000	i diaeresis/umlaut small
Ï	\(I:	LI180000	I diaeresis/umlaut capital
ĩ	\(i˜	LI190000	i tilde small
Ĩ	\(I˜	LI200000	I tilde capital
ĳ	\(ij	LI510000	ij ligature small
j	j	LJ010000	j small
J	J	LJ020000	J capital
k	k	LK010000	k small
K	K	LK020000	K capital
l	l	LL010000	l small
L	L	LL020000	L capital
ŀ	\(l.	LL630000	l middle dot small
Ŀ	\(L.	LL640000	L middle dot capital
m	m	LM010000	m small
M	M	LM020000	M capital
n	n	LN010000	n small
N	N	LN020000	N capital
ń	\(n’	LN110000	n acute small
Ń	\(N’	LN120000	N acute capital
ñ	\(n˜	LN190000	n tilde small
Ñ	\(N˜	LN200000	N tilde capital
ň	\(n-	LN210000	n caron small
Ň	\(N-	LN220000	N caron capital
o	o	LO010000	o small
O	O	LO020000	O capital
ó	\(o’	LO110000	o acute small
Ó	\(O’	LO120000	O acute capital
ò	\(o‘	LO130000	O grave small
Ò	\(O‘	LO140000	O grave capital
ô	\(o^	LO150000	o circumflex small
Ô	\(O^	LO160000	O circumflex capital
ö	\(o:	LO170000	o diaeresis/umlaut small
Ö	\(O:	LO180000	O diaeresis/umlaut capital
õ	\(o˜	LO190000	o tilde small
Õ	\(O˜	LO200000	O tilde capital
œ	\(oe	LO510000	oe diphthong small
Œ	\(OE	LO520000	OE diphthong capital
ø	\(o/	LO610000	o slash small
Ø	\(O/	LO620000	O slash capital
Ø	\(es	LO620000	Empty set
p	p	LP010000	p small

P	P	LP020000	P capital
q	q	LQ010000	q small
Q	Q	LQ020000	Q capital
r	r	LR010000	r small
R	R	LR020000	R capital
s	s	LS010000	s small
S	S	LS020000	S capital
ß	\(ss	LS610000	s sharp small
t	t	LT010000	t small
T	T	LT020000	T capital
u	u	LU010000	u small
U	U	LU020000	U capital
ú	\(u’	LU110000	u acute small
Ú	\(U’	LU120000	U acute capital
ù	\(u‘	LU130000	u grave small
Ù	\(U‘	LU140000	U grave capital
û	\(u^	LU150000	u circumflex small
Û	\(U^	LU160000	U circumflex capital
ü	\(u:	LU170000	u diaeresis/umlaut small
Ü	\(U:	LU180000	U diaeresis/umlaut capital
ũ	\(u~	LU190000	u tilde small
Ũ	\(U~	LU200000	U tilde capital
ů	\(u.	LU270000	u overcircle small
Ů	\(U.	LU280000	U overcircle capital
v	v	LV010000	v small
V	V	LV020000	V capital
w	w	LW010000	w small
W	W	LW020000	W capital
x	x	LX010000	x small
X	X	LX020000	X capital
y	y	LY010000	y small
Y	Y	LY020000	Y capital
ÿ	\(y:	LY170000	y diaeresis/umlaut small
Ÿ	\(Y:	LY180000	Y diaeresis/umlaut capital
z	z	LZ010000	z small
Z	Z	LZ020000	Z capital
1	1	ND010000	Numeric decimal one
^{1}	\(1S	ND011000	Numeric decimal one superscript
2	2	ND020000	Numeric decimal two
^{2}	\(2S	ND021000	Numeric decimal two superscript
3	3	ND030000	Numeric decimal three
^{3}	\(3S	ND031000	Numeric decimal three superscript
4	4	ND040000	Numeric decimal four
^{4}	\(4S	ND041000	Numeric decimal four superscript
5	5	ND050000	Numeric decimal five
^{5}	\(5S	ND051000	Numeric decimal five superscript
6	6	ND060000	Numeric decimal six
^{6}	\(6S	ND061000	Numeric decimal six superscript
7	7	ND070000	Numeric decimal seven
^{7}	\(7S	ND071000	Numeric decimal seven superscript
8	8	ND080000	Numeric decimal eight
^{8}	\(8S	ND081000	Numeric decimal eight superscript
9	9	ND090000	Numeric decimal nine

^{9}	\(9S	ND091000	Numeric decimal nine superscript
0	0	ND100000	Numeric decimal zero
^{0}	\(0S	ND101000	Numeric decimal zero superscript
½	\(12	NF010000	Numeric fraction one-half
¼	\(14	NF040000	Numeric fraction one-quarter
¾	\(34	NF050000	Numeric fraction three-quarters
⅛	\(18	NF180000	Numeric fraction one-eighth
⅜	\(38	NF190000	Numeric fraction three-eighths
⅝	\(58	NF200000	Numeric fraction five-eighths
⅞	\(78	NF210000	Numeric fraction seven-eighths
−	\-	SA000000	Current font minus
−	\(em	SA000000	Em-dash
−	\(mi	SA000000	Minus
+	+	SA010000	Plus
+	\(pl	SA010000	Plus
±	\(+-	SA020000	Plus-or-minus
$^{\pm}$	\(S-	SA021000	Plus-or-minus superscript
<	<	SA030000	Less-than
$^{<}$	\(S<	SA031000	Less-than superscript
=	=	SA040000	Equals
=	\(eq	SA040000	Equals
$^{=}$	\(S=	SA041000	Equals superscript
>	>	SA050000	Greater-than
$^{>}$	\(S>	SA051000	Greater-than superscript
÷	\(di	SA060000	Divide
$^{\div}$	\(S/	SA061000	Divide superscript
×	\(mu	SA070000	Multiply
$^{\times}$	\(Sx	SA071000	Multiply superscript
≤	\(<=	SA520000	Less-than-or-equal
$^{\leq}$	\(Sl	SA521000	Less-than-or-equal superscript
≥	\(>=	SA530000	Greater-than-or-equal
$^{\geq}$	\(Sg	SA531000	Greater-than-or-equal superscript
≠	\(!=	SA540000	Not-equal
$^{\neq}$	\(S!	SA541000	Not-equal superscript
¤	\(Ic	SC010000	International currency
£	\(Lb	SC020000	Pound
$	$	SC030000	Dollar
¢	\(ct	SC040000	Cent
¥	\(Y-	SC050000	Yen
₧	\(Ps	SC060000	Peseta
ƒ	\(fg	SC070000	Florin or guilder
#	#	SM010000	Number
%	%	SM020000	Percent
&	&	SM030000	Ampersand
*	*	SM040000	Asterisk
*	\(**	SM040000	Math star
@	@	SM050000	At
[	[	SM060000	Left bracket
\	\	SM070000	Reverse slash
]	]	SM080000	Right bracket
{	{	SM110000	Left brace
|	|	SM130000	Vertical bar, logical OR
|	\(br	SM130000	Vertical bar, logical OR

|	\(or	SM130000	Vertical bar, logical OR
}	}	SM140000	Right brace
µ	\(*m	SM170000	Mu
°	\(de	SM190000	Degree symbol
§	\(SS	SM240000	Section symbol (USA), paragraph symbol (Europe)
§	\(sc	SM240000	Section symbol (USA), paragraph symbol (Europe)
¶	\(PS	SM250000	Paragraph symbol (USA)
†	\(dg	SM340000	Dagger
‡	\(dd	SM350000	Double dagger
■	\(Ss	SM470000	Solid square, histogram
⌑	\(Lz	SM490000	Lozenge
‰	\(PM	SM560000	Per mill symbol
•	\(bu	SM570000	Bullet
¦	\(BV	SM650000	Broken vertical line
¬	\(no	SM660000	Logical NOT, end of line
¬	\(NO	SM660000	Logical NOT, end of line
—	\(ru	SM900000	Rule
	\(SP	SP010000	Interword space
!	!	SP020000	Exclamation point
¡	\(I!	SP030000	Exclamation point, inverted
″	\(QM	SP040000	Quotation marks
′	\(aa	SP050000	Acute
′	\(fm	SP050000	Foot mark
(	(	SP060000	Left parenthesis
⁽	\(So	SP061000	Left parenthesis superscript
)	)	SP070000	Right parenthesis
⁾	\(Sc	SP071000	Right parenthesis superscript
,	,	SP080000	Comma
ʼ	\(Sm	SP081000	Comma superscript
_	_	SP090000	Continuous underscore
_	\(ul	SP090000	Underline
-	-	SP100000	Hyphen
-	\(hy	SP100000	Hyphen
⁻	\(Sh	SP101000	Hyphen superscript
·	\(Sp	SP111000	Period, full stop superscript
/	/	SP120000	Slash
/	\(sl	SP120000	Slash
:	:	SP130000	Colon
;	;	SP140000	Semicolon
?	?	SP150000	Question mark
¿	\(I?	SP160000	Question mark, inverted
«	\(< <	SP170000	Left angle quotes
«	\(lh	SP170000	Left angle quotes
»	\(> >	SP180000	Right angle quotes
»	\(rh	SP180000	Right angle quotes
‘	‘	SP190000	Left single quote
“	\(LQ	SP210000	Left double quotes
”	”	SP220000	Right double quotes
”	\(RQ	SP220000	Right double quotes
„	\(L:	SP230000	Left lower double quotes (German)
	\(RS	SP300000	Required space
⍰	\(UN	SV320000	Replacement symbol (for undefined code points)

PI CODE PAGE (picp)

	Troff	IBM	
	Char	Char	
Char	Name	Name	Description
α	\(*a	GA010000	Alpha small
Α	\(*A	GA020000	Alpha capital
β	\(*b	GB010000	Beta small
Β	\(*B	GB020000	Beta capital
δ	\(*d	GD010000	Delta small
Δ	\(*D	GD020000	Delta capital
ε	\(*e	GE010000	Epsilon small
Ε	\(*E	GE020000	Epsilon capital
η	\(*y	GE310000	Eta small
Η	\(*Y	GE320000	Eta capital
φ	\(*f	GF010000	Phi small
Φ	\(*F	GF020000	Phi capital
γ	\(*g	GG010000	Gamma small
Γ	\(*G	GG020000	Gamma capital
χ	\(*x	GH010000	Chi small
Χ	\(*X	GH020000	Chi capital
ι	\(*i	GI010000	Iota small
Ι	\(*I	GI020000	Iota capital
κ	\(*k	GK010000	Kappa small
Κ	\(*K	GK020000	Kappa capital
λ	\(*l	GL010000	Lambda small
Λ	\(*L	GL020000	Lambda capital
μ	\(*m	GM010000	Mu small
Μ	\(*M	GM020000	Mu capital
ν	\(*n	GN010000	Nu small
Ν	\(*N	GN020000	Nu capital
ο	\(*o	GO010000	Omicron small
Ο	\(*O	GO020000	Omicron capital
ω	\(*w	GO310000	Omega small
Ω	\(*W	GO320000	Omega capital
π	\(*p	GP010000	Pi small
Π	\(*P	GP020000	Pi capital
ψ	\(*q	GP610000	Psi small
Ψ	\(*Q	GP620000	Psi capital
ρ	\(*r	GR010000	Rho small
Ρ	\(*R	GR020000	Rho capital
σ	\(*s	GS010000	Sigma small
Σ	\(*S	GS020000	Sigma capital
τ	\(*t	GT010000	Tau small
Τ	\(*T	GT020000	Tau capital
θ	\(*h	GT610000	Theta small
Θ	\(*H	GT620000	Theta small
υ	\(*u	GU010000	Upsilon small
Υ	\(*U	GU020000	Upsilon capital
ξ	\(*c	GX010000	Xi small
Ξ	\(*C	GX020000	Xi capital
ζ	\(*z	GZ010000	Zeta small

Z	\(*Z	GZ020000	Zeta capital
^{a}	\(aS	LA011000	a small superscript
ą	\(U1	LA430000	a ogonek small
Ą	\(U2	LA440000	A ogonek capital
^{b}	\(bS	LB011000	b small superscript
^{c}	\(cS	LC011000	c small superscript
^{d}	\(dS	LD011000	d small superscript
đ	\(D1	LD610000	d stroke small
Đ	\(D2	LD620000	D stroke capital and Eth Icelandic capital
ð	\(D3	LD630000	eth Icelandic small
^{e}	\(eS	LE011000	e small superscript
ę	\(E1	LE430000	e ogonek small
Ę	\(E2	LE440000	E ogonek capital
^{f}	\(fS	LF011000	f small superscript
^{g}	\(gS	LG011000	g small superscript
^{h}	\(hS	LH011000	h small superscript
^{i}	\(iS	LI011000	i small superscript
ı	\(I1	LI610000	i dotless small
^{j}	\(jS	LJ011000	j small superscript
^{k}	\(kS	LK011000	k small superscript
^{l}	\(lS	LL011000	l small superscript
ł	\(L1	LL610000	l stroke small
Ł	\(L2	LL620000	L stroke capital
^{m}	\(mS	LM011000	m small superscript
^{n}	\(nS	LN011000	n small superscript
^{o}	\(oS	LO011000	o small superscript
^{p}	\(pS	LP011000	p small superscript
^{q}	\(qS	LQ011000	q small superscript
^{r}	\(rS	LR011000	r small superscript
^{s}	\(sS	LS011000	s small superscript
^{t}	\(tS	LT011000	t small superscript
þ	\(T1	LT630000	Thorn Icelandic small
Þ	\(T2	LT640000	Thorn Icelandic capital
^{u}	\(uS	LU011000	u small superscript
^{v}	\(vS	LV011000	v small superscript
^{w}	\(wS	LW011000	w small superscript
^{x}	\(xS	LX011000	x small superscript
^{y}	\(yS	LY011000	y small superscript
ý	\(Y1	LY110000	y acute small
Ý	\(Y2	LY120000	Y acute capital
^{z}	\(zS	LZ011000	z small superscript
1	1	ND010000	Numeric decimal one
^{1}	\(1S	ND011000	Numeric decimal one superscript
2	2	ND020000	Numeric decimal two
^{2}	\(2S	ND021000	Numeric decimal two superscript
3	3	ND030000	Numeric decimal three
^{3}	\(3S	ND031000	Numeric decimal three superscript
4	4	ND040000	Numeric decimal four
^{4}	\(4S	ND041000	Numeric decimal four superscript
5	5	ND050000	Numeric decimal five
^{5}	\(5S	ND051000	Numeric decimal five superscript
6	6	ND060000	Numeric decimal six
^{6}	\(6S	ND061000	Numeric decimal six superscript

7	7	ND070000	Numeric decimal seven
^{7}	\(7S	ND071000	Numeric decimal seven superscript
8	8	ND080000	Numeric decimal eight
^{8}	\(8S	ND081000	Numeric decimal eight superscript
9	9	ND090000	Numeric decimal nine
^{9}	\(9S	ND091000	Numeric decimal nine superscript
0	0	ND100000	Numeric decimal zero
∅	\(0/	ND100008	Numeric decimal zero slash
Ø	\(es	ND100008	Empty Set
^{0}	\(0S	ND101000	Numeric decimal zero superscript
-	-	SA000000	Minus
+	+	SA010000	Plus
±	\(+-	SA020000	Plus-or-minus
=	=	SA040000	Equals
÷	\(di	SA060000	Divide
×	\(mu	SA070000	Multiply
~	\(ap	SA160000	Cycle symbol, equivalent to
‖	\(Pl	SA340000	Parallel symbol
∠	\(An	SA350000	Angle symbol
∈̄	\(Nm	SA360000	Is not an element of
∴	\(Tf	SA370000	Therefore symbol
∩	\(ca	SA380000	Intersection, logical product
∪	\(cu	SA390000	Union, logical sum
⊂	\(sb	SA400000	Included in, a subset of
⊃	\(sp	SA410000	Includes, a superset of
≅	\(˜=	SA430000	Congruent to
∞	\(if	SA450000	Infinity symbol
∝	\(pt	SA470000	Proportional to
≡	\(==	SA480000	Identity symbol
∂	\(pd	SA490000	Partial differential symbol
∫	\(is	SA510000	Integral symbol
≤	\(<=	SA520000	Less-than-or-equal
≥	\(>=	SA530000	Greater-than-or-equal
≠	\(!=	SA540000	Not-equal
⊕	\(Cs	SA550000	Closed sum
◇	\(Di	SA660000	Diamond
∈	\(mo	SA670000	"Is an element of"
≈	\(Ae	SA700000	Nearly equals
⊥	\(Pd	SA780000	Perpendicular to
·	\(Md	SA790000	Dot multiply, middle dot
´	\(aa	SD110000	Acute
‘	\(ga	SD130000	Grave
^	^	SD150000	Circumflex
¨	\(UM	SD170000	Diaeresis or umlaut
˜	˜	SD190000	Tilde
¸	\(CE	SD410000	Cedilla
≡	\(DU	SM100000	Double underscore
	\(rn	SM150000	Overline
ℓ	\(Lt	SM160000	Litre symbol
º	\(ML	SM200000	Ordinal indicator - masculine
ª	\(FE	SM210000	Ordinal indicator - feminine
√	\(sr	SM230000	Tape mark, radical
♂	\(Ma	SM280000	Male symbol

♀	\(Fe	SM290000	Female symbol
←	\(<-	SM300000	Left arrow
→	\(->	SM310000	Right arrow
↑	\(ua	SM320000	Up arrow
↓	\(da	SM330000	Down arrow
□	\(sq	SM450000	Open square
■	\(Ss	SM470000	Solid square, histogram
⌑	\(Lz	SM490000	Lozenge
′	\(MI	SM500000	Minutes symbol
″	\(SE	SM510000	Seconds symbol
©	\(co	SM520000	Copyright symbol
®	\(rg	SM530000	Registered trademark symbol
™	\(TM	SM540000	Trademark symbol
℞	\(Rx	SM550000	Prescription symbol
▶	\(AI	SM590000	Arrow indicator
▲	\(ST	SM600000	Solid triangle
◆	\(SD	SM610000	Solid diamond
|	|	SM650000	Broken vertical line
¦	\(BV	SM650000	Broken vertical line
¬	\(no	SM660000	Logical NOT, end of line
ƀ	\(b/	SM670000	Substitute blank
△	\(OT	SM730000	Open triangle, mode change
○	\(ci	SM750000	Open circle
↗	\(In	SM950000	Increase
↘	\(De	SM990000	Decrease
⑀	\(HO	SO000000	Hook
⑂	\(FO	SO010000	Fork
⑁	\(CH	SO020000	Chair
	\(SP	SP010000	Interword space
⁽	\(So	SP061000	Left parenthesis superscript
⁾	\(Sc	SP071000	Right parenthesis superscript
ʼ	\(Sm	SP081000	Comma superscript
˙	\(Sp	SP111000	Period, full stop superscript
∧	\(PO	SS390000	Pointer (for text insertion)
∑	\(SU	SS400000	Summation symbol
∶	\(RA	SS430000	Ratio symbol
ƒ	\(Fs	SS440000	Function symbol
ℊ	\(Rn	SS460000	"Real number" symbol
ℛ	\(Ri	SS470000	Riemann integral
ℒ	\(La	SS480000	LaPlace symbol
∵	\(TD	SS540000	"Because" symbol
★	\(CS	SS580000	Closed star upright
Ӿ	\(PT	SS610000	"Plaintiff" symbol
⚱	\(BS	SS630000	Bottle symbol
℅	\(CO	SS640000	"Care of" symbol
℁	\(AO	SS650000	"Account of" symbol
☏	\(TO	SS670000	Telephone symbol (open)
–	\(en	SS680000	En-dash
☎	\(bs	SS700000	Telephone symbol, closed
☎	\(TC	SS700000	Telephone symbol, closed
🄽	\(UN	SV320000	Replacement symbol (for undefined code points)
<	<	SA030000	Less-than
>	>	SA050000	Greater-than

¤	\(Ic	SC010000	International currency
£	\(Lb	SC020000	Pound
\	\	SM070000	Reverse slash
{	{	SM110000	Left brace
}	}	SM140000	Right brace
•	\(bu	SM570000	Bullet
-	-	SP100000	Hyphen

SYMBOLS CODE PAGE (symcp)

Char	Troff Char Name	IBM Char Name	Description
α	\(*a	GA010000	Alpha small
β	\(*b	GB010000	Beta small
δ	\(*d	GD010000	Delta small
Δ	\(*D	GD020000	Delta capital
ε	\(*e	GE010000	Epsilon small
η	\(*y	GE310000	Eta small
φ	\(*f	GF010000	Phi small
Φ	\(*F	GF020000	Phi capital
γ	\(*g	GG010000	Gamma small
Γ	\(*G	GG020000	Gamma capital
χ	\(*x	GH010000	Chi small
ι	\(*i	GI010000	Iota small
κ	\(*k	GK010000	Kappa small
λ	\(*l	GL010000	Lambda small
Λ	\(*L	GL020000	Lambda capital
μ	\(*m	GM010000	Mu small
ν	\(*n	GN010000	Nu small
∇	\(*o	GO010000	Omicron small
ω	\(*w	GO310000	Omega small
Ω	\(*W	GO320000	Omega capital
π	\(*p	GP010000	Pi small
Π	\(*P	GP020000	Pi capital
ψ	\(*q	GP610000	Psi small
Ψ	\(*Q	GP620000	Psi capital
ρ	\(*r	GR010000	Rho small
σ	\(*s	GS010000	Sigma small
Σ	\(*S	GS020000	Sigma capital
τ	\(*t	GT010000	Tau small
Τ	\(*T	GT020000	Tau capital
θ	\(*h	GT610000	Theta small
Θ	\(*H	GT620000	Theta small
υ	\(*u	GU010000	Upsilon small
ξ	\(*c	GX010000	Xi small
Ξ	\(*C	GX020000	Xi capital
ζ	\(*z	GZ010000	Zeta small
^{1}	\(1S	ND011000	Numeric decimal one superscript
^{2}	\(2S	ND021000	Numeric decimal two superscript
^{3}	\(3S	ND031000	Numeric decimal three superscript
^{4}	\(4S	ND041000	Numeric decimal four superscript
^{5}	\(5S	ND051000	Numeric decimal five superscript
^{6}	\(6S	ND061000	Numeric decimal six superscript
^{7}	\(7S	ND071000	Numeric decimal seven superscript
^{8}	\(8S	ND081000	Numeric decimal eight superscript
^{9}	\(9S	ND091000	Numeric decimal nine superscript
Ø	\(0/	ND100008	Numeric decimal zero slash
Ø	\(es	ND100008	Empty Set
^{0}	\(0S	ND101000	Numeric decimal zero superscript

<	<	SA030000	Less-than
>	>	SA050000	Greater-than
÷	\(di	SA060000	Divide
×	\(mu	SA070000	Multiply
‖	\(Pl	SA340000	Parallel symbol
∞	\(if	SA450000	Infinity symbol
∝	\(pt	SA470000	Proportional to
≡	\(= =	SA480000	Identity symbol
∂	\(pd	SA490000	Partial differential symbol
∫	\(is	SA510000	Integral symbol
≤	\(< =	SA520000	Less-than-or-equal
≥	\(> =	SA530000	Greater-than-or-equal
≠	\(!=	SA540000	Not-equal
¤	\(Ic	SC010000	International currency
£	\(Lb	SC020000	Pound
\	\	SM070000	Reverse slash
{	{	SM110000	Left brace
}	}	SM140000	Right brace
ℓ	\(Lt	SM160000	Litre symbol
←	\(< -	SM300000	Left arrow
→	\(- >	SM310000	Right arrow
↑	\(ua	SM320000	Up arrow
↓	\(da	SM330000	Down arrow
□	\(sq	SM450000	Open square
©	\(co	SM520000	Copyright symbol
®	\(rg	SM530000	Registered trademark symbol
™	\(TM	SM540000	Trademark symbol
℞	\(Rx	SM550000	Prescription symbol
•	\(bu	SM570000	Bullet
ƀ	\(b/	SM670000	Substitute blank
o	\(ci	SM750000	Open circle
	\(SP	SP010000	Interword space
–	-	SP100000	Hyphen
▧	\(UN	SV320000	Replacement symbol (for undefined code points)
ϑ	\(sd	GT630000	Script d
‹	<	SA181000	Less-than superscript
›	>	SA191000	Greater-than superscript
≃	\(˜=	SA440000	Approximates
√	\(sr	SA800000	Square root
Π	\(PR	SA810000	"Product of" symbol
~	˜	SA820000	Tilde
∇	\(gr	SL030000	Gradient
═	=	SM640000	Equals

ADDITIONAL CODE PAGE (addcp)

	Troff	**IBM**	
	Char	**Char**	
Char	**Name**	**Name**	**Description**
⎧	\(lt	ACIS0001	left top of big curly bracket
⎧	\(lt	ACIS0001	left top of big curly bracket
⎨	\(lk	ACIS0002	left center of big curly bracket
⎨	\(lk	ACIS0002	left center of big curly bracket
⎩	\(lb	ACIS0003	left bottom of big curly bracket
⎩	\(lb	ACIS0003	left bottom of big curly bracket
⎪	\(bv	ACIS0004	bold vertical
⎫	\(rt	ACIS0005	right top of big curly bracket
⎬	\(rk	ACIS0006	right center of big curly bracket
⎭	\(rb	ACIS0007	right bottom of big curly bracket
⊆	\(ib	ACIS0008	improper subset
⊇	\(ip	ACIS0009	improper superset
∇	\(gr	ACIS0010	gradient
⌊	\(lf	ACIS0011	left floor (left bottom of big square bracket)
⌈	\(lc	ACIS0012	left ceiling (top left)
⌋	\(rf	ACIS0013	right floor (right bottom)
⌉	\(rc	ACIS0014	right ceiling (right top)
ς	\(ts	ACIS0015	terminal sigma

FIXED CODE PAGE (fcp)

	Troff	IBM	
	Char	Char	
Char	Name	Name	Description
a	a	LA010000	a small
A	A	LA020000	A capital
á	\(a’	LA110000	a acute small
Á	\(A’	LA120000	A acute capital
à	\(a‘	LA130000	a grave small
À	\(A‘	LA140000	A grave capital
â	\(a^	LA150000	a circumflex small
Â	\(A^	LA160000	A circumflex capital
ä	\(a:	LA170000	a diaeresis/umlaut small
Ä	\(A:	LA180000	A diaeresis/umlaut capital
ã	\(a~	LA190000	a tilde small
Ã	\(A~	LA200000	A tilde capital
å	\(a.	LA270000	a overcircle small
Å	\(A.	LA280000	A overcircle capital
æ	\(ae	LA510000	ae diphthong small
Æ	\(AE	LA520000	AE diphthong capital
b	b	LB010000	b small
B	B	LB020000	B capital
c	c	LC010000	c small
C	C	LC020000	C capital
ç	\(c,	LC410000	c cedilla small
Ç	\(C,	LC420000	C cedilla capital
d	d	LD010000	d small
D	D	LD020000	D capital
Đ	\(D2	LD620000	D stroke capital and Eth Icelandic capital
ð	\(D3	LD630000	eth Icelandic small
e	e	LE010000	e small
E	E	LE020000	E capital
é	\(e’	LE110000	e acute small
É	\(E’	LE120000	E acute capital
è	\(e‘	LE130000	e grave small
È	\(E‘	LE140000	E grave capital
ê	\(e^	LE150000	e circumflex small
Ê	\(E^	LE160000	E circumflex capital
ë	\(e:	LE170000	e diaeresis/umlaut small
Ë	\(E:	LE180000	E diaeresis/umlaut capital
ě	\(e-	LE210000	C caron small
f	f	LF010000	f small
F	F	LF020000	F capital
g	g	LG010000	g small
G	G	LG020000	G capital
h	h	LH010000	h small
H	H	LH020000	H capital
i	i	LI010000	i small
I	I	LI020000	I capital
í	\(i’	LI110000	i acute small
Í	\(I’	LI120000	I acute capital

ì	\(i‘	LI130000	i grave small
Ì	\(I‘	LI140000	I grave capital
î	\(i^	LI150000	i circumflex small
Î	\(I^	LI160000	I circumflex capital
ï	\(i:	LI170000	i diaeresis/umlaut small
Ï	\(I:	LI180000	I diaeresis/umlaut capital
ĳ	\(ij	LI510000	ij ligature small
j	j	LJ010000	j small
J	J	LJ020000	J capital
k	k	LK010000	k small
K	K	LK020000	K capital
l	l	LL010000	l small
L	L	LL020000	L capital
ŀ	\(l.	LL630000	l middle dot small
Ŀ	\(L.	LL640000	L middle dot capital
m	m	LM010000	m small
M	M	LM020000	M capital
n	n	LN010000	n small
ⁿ	\(nS	LN011000	n small superscript
N	N	LN020000	N capital
ñ	\(n~	LN190000	n tilde small
Ñ	\(N~	LN200000	N tilde capital
ň	\(n-	LN210000	n caron small
o	o	LO010000	o small
O	O	LO020000	O capital
ó	\(o’	LO110000	o acute small
Ó	\(O’	LO120000	O acute capital
ò	\(o‘	LO130000	O grave small
Ò	\(O‘	LO140000	O grave capital
ô	\(o^	LO150000	o circumflex small
Ô	\(O^	LO160000	O circumflex capital
ö	\(o:	LO170000	o diaeresis/umlaut small
Ö	\(O:	LO180000	O diaeresis/umlaut capital
õ	\(o~	LO190000	o tilde small
Õ	\(O~	LO200000	O tilde capital
ọ	\(o.	LO450000	o small underdot
œ	\(oe	LO510000	oe diphthong small
Œ	\(OE	LO520000	OE diphthong capital
ø	\(o/	LO610000	o slash small
Ø	\(O/	LO620000	O slash capital
Ø	\(es	LO620000	Empty set
p	p	LP010000	p small
P	P	LP020000	P capital
q	q	LQ010000	q small
Q	Q	LQ020000	Q capital
r	r	LR010000	r small
R	R	LR020000	R capital
s	s	LS010000	s small
S	S	LS020000	S capital
ß	\(ss	LS610000	s sharp small
t	t	LT010000	t small
T	T	LT020000	T capital
þ	\(T1	LT630000	Thorn Icelandic small

Þ	\(T2	LT640000	Thorn Icelandic capital
u	u	LU010000	u small
U	U	LU020000	U capital
ú	\(u'	LU110000	u acute small
Ú	\(U'	LU120000	U acute capital
ù	\(u‘	LU130000	u grave small
Ù	\(U‘	LU140000	U grave capital
û	\(u^	LU150000	u circumflex small
Û	\(U^	LU160000	U circumflex capital
ü	\(u:	LU170000	u diaeresis/umlaut small
Ü	\(U:	LU180000	U diaeresis/umlaut capital
ů	\(u.	LU270000	u overcircle small
Ů	\(U.	LU280000	U overcircle capital
v	v	LV010000	v small
V	V	LV020000	V capital
w	w	LW010000	w small
W	W	LW020000	W capital
x	x	LX010000	x small
X	X	LX020000	X capital
y	y	LY010000	y small
Y	Y	LY020000	Y capital
ý	\(Y1	LY110000	y acute small
Ý	\(Y2	LY120000	Y acute capital
ÿ	\(y:	LY170000	y diaeresis/umlaut small
Ÿ	\(Y:	LY180000	Y diaeresis/umlaut capital
z	z	LZ010000	z small
Z	Z	LZ020000	Z capital
1	1	ND010000	Numeric decimal one
^{1}	\(1S	ND011000	Numeric decimal one superscript
2	2	ND020000	Numeric decimal two
^{2}	\(2S	ND021000	Numeric decimal two superscript
3	3	ND030000	Numeric decimal three
^{3}	\(3S	ND031000	Numeric decimal three superscript
4	4	ND040000	Numeric decimal four
^{4}	\(4S	ND041000	Numeric decimal four superscript
5	5	ND050000	Numeric decimal five
^{5}	\(5S	ND051000	Numeric decimal five superscript
6	6	ND060000	Numeric decimal six
^{6}	\(6S	ND061000	Numeric decimal six superscript
7	7	ND070000	Numeric decimal seven
^{7}	\(7S	ND071000	Numeric decimal seven superscript
8	8	ND080000	Numeric decimal eight
^{8}	\(8S	ND081000	Numeric decimal eight superscript
9	9	ND090000	Numeric decimal nine
^{9}	\(9S	ND091000	Numeric decimal nine superscript
0	0	ND100000	Numeric decimal zero
^{0}	\(0S	ND101000	Numeric decimal zero superscript
½	\(12	NF010000	Numeric fraction one-half
¼	\(14	NF040000	Numeric fraction one-quarter
¾	\(34	NF050000	Numeric fraction three-quarters
+	+	SA010000	Plus
±	\(+-	SA020000	Plus-or-minus
<	<	SA030000	Less-than

=	=	SA040000	Equals
⁼	\(S=	SA041000	Equals superscript
>	>	SA050000	Greater-than
÷	\(di	SA060000	Divide
×	\(mu	SA070000	Multiply
∩	\(ca	SA380000	Intersection, logical product
∞	\(if	SA450000	Infinity symbol
≡	\(==	SA480000	Identity symbol
≤	\(<=	SA520000	Less-than-or-equal
≥	\(>=	SA530000	Greater-than-or-equal
≠	\(!=	SA540000	Not-equal
≈	\(Ae	SA700000	Nearly equals
·	\(Md	SA790000	Dot multiply, middle dot
¤	\(Ic	SC010000	International currency
£	\(Lb	SC020000	Pound
$	$	SC030000	Dollar
¢	\(ct	SC040000	Cent
¥	\(Y-	SC050000	Yen
₧	\(Ps	SC060000	Peseta
ƒ	\(fg	SC070000	Florin or guilder
₢	\(Cx	SC090000	Cruzeiro
´	\(aa	SD110000	Acute
ˋ	`	SD130000	Grave
ˋ	\(ga	SD130000	Grave
ˆ	^	SD150000	Circumflex
¨	\(UM	SD170000	Diaeresis or umlaut
˜	~	SD190000	Tilde
¸	\(CE	SD410000	Cedilla
┌	\(Ul	SF010000	Upper left joint
┏	\(UZ	SF010002	Upper left joint, thick
└	\(Ll	SF020000	Lower left joint
┗	\(LZ	SF020002	Lower left joint, thick
┐	\(Ur	SF030000	Upper right joint
┓	\(UY	SF030002	Upper right joint, thick
┘	\(Lr	SF040000	Lower right joint
┛	\(LY	SF040002	Lower right joint, thick
┼	\(Mb	SF050000	Middle joint
╋	\(MZ	SF050002	Middle joint, thick
┬	\(Tm	SF060000	Top middle joint
┳	\(TZ	SF060002	Top middle joint, thick
┴	\(Bm	SF070000	Bottom middle joint
┻	\(BX	SF070002	Bottom middle joint, thick
├	\(Lm	SF080000	Left middle joint
┣	\(LX	SF080002	Left middle joint, thick
┤	\(Rm	SF090000	Right middle joint
┫	\(RX	SF090002	Right middle joint, thick
─	\(Hb	SF100000	Horizontal bar
━	\(HX	SF100002	Horizontal bar, thick
│	\(Vb	SF110000	Vertical bar
┃	\(VX	SF110002	Vertical bar, thick
#	#	SM010000	Number
%	%	SM020000	Percent
&	&	SM030000	Ampersand

*	*	SM040000	Asterisk
*	\(**	SM040000	Math star
@	@	SM050000	At
[	[	SM060000	Left bracket
\	\	SM070000	Reverse slash
]	]	SM080000	Right bracket
‗	\(DU	SM100000	Double underscore
{	{	SM110000	Left brace
¦	\(br	SM130000	Vertical bar, logical OR
¦	\(or	SM130000	Vertical bar, logical OR
¦	|	SM130000	Vertical bar, logical OR
}	}	SM140000	Right brace
‾	\(rn	SM150000	Overline
µ	\(MU	SM170000	Mu
°	\(de	SM190000	Degree symbol
º	\(ML	SM200000	Ordinal indicator - masculine
ª	\(FE	SM210000	Ordinal indicator - feminine
§	\(SS	SM240000	Section symbol (USA), paragraph symbol (Europe)
§	\(sc	SM240000	Section symbol (USA), paragraph symbol (Europe)
¶	\(PS	SM250000	Paragraph symbol (USA)
←	\(< -	SM300000	Left arrow
→	\(- >	SM310000	Right arrow
↑	\(ua	SM320000	Up arrow
↓	\(da	SM330000	Down arrow
†	\(dg	SM340000	Dagger
‡	\(dd	SM350000	Double dagger
■	\(Ss	SM470000	Solid square, histogram
¤	\(Lz	SM490000	Lozenge
©	\(co	SM520000	Copyright symbol
®	\(rg	SM530000	Registered trademark symbol
•	\(bu	SM570000	Bullet
¦	\(BV	SM650000	Broken vertical line
¬	\(no	SM660000	Logical NOT, end of line
	\(SP	SP010000	Interword space
!	!	SP020000	Exclamation point
¡	\(I!	SP030000	Exclamation point, inverted
"	”	SP040000	Quotation marks
"	\(LQ	SP040000	Left Quotation mark
"	\(QM	SP040000	Quotation marks
"	\(RQ	SP040000	Quotation mark
'	\(fm	SP050000	Foot mark
(	(	SP060000	Left parenthesis
(	\(So	SP061000	Left parenthesis superscript
)	)	SP070000	Right parenthesis
)	\(Sc	SP071000	Right parenthesis superscript
,	,	SP080000	Comma
_	_	SP090000	Continuous underscore
_	\(ul	SP090000	Underline
-	-	SP100000	Hyphen
-	\-	SP100000	Current font minus
-	\(em	SP100000	em-dash
-	\(en	SP100000	en-dash
-	\(hy	SP100000	Hyphen

−	\(mi	SP100000	Minus
−	\(ru	SP100000	Rule
/	/	SP120000	Slash
/	\(sl	SP120000	Slash
:	:	SP130000	Colon
;	;	SP140000	Semicolon
?	?	SP150000	Question mark
¿	\(I?	SP160000	Question mark, inverted
«	\(< <	SP170000	Left angle quotes
«	\(lh	SP170000	Left angle quotes
»	\(> >	SP180000	Right angle quotes
»	\(rh	SP180000	Right angle quotes
	\(RS	SP300000	Required space
⍰	\(UN	SV320000	Replacement symbol (for undefined code points)

ALTERNATE FIXED CODE PAGE (acp)

Char	Troff Char Name	IBM Char Name	Description
α	\(*a	GA010000	Alpha small
δ	\(*d	GD010000	Delta small
ε	\(*e	GE010000	Epsilon small
ϕ	\(*f	GF010000	Phi small
Φ	\(*F	GF020000	Phi capital
Γ	\(*G	GG020000	Gamma capital
μ	\(*m	GM010000	Mu small
Ω	\(*W	GO320000	Omega capital
π	\(*p	GP010000	Pi small
τ	\(*t	GT010000	Tau small
θ	\(*H	GT620000	Theta small
a	a	LA010000	a small
A	A	LA020000	A capital
á	\(a’	LA110000	a acute small
à	\(a‘	LA130000	a grave small
â	\(a^	LA150000	a circumflex small
ä	\(a:	LA170000	a diaeresis/umlaut small
Ä	\(A:	LA180000	diaeresis/umlaut capital
å	\(a.	LA270000	a overcircle small
Å	\(A.	LA280000	overcircle capital
æ	\(ae	LA510000	ae diphthong small
Æ	\(AE	LA520000	diphthong capital
b	b	LB010000	b small
B	B	LB020000	B capital
c	c	LC010000	c small
C	C	LC020000	C capital
ç	\(c,	LC410000	c cedilla small
Ç	\(C,	LC420000	cedilla capital
d	d	LD010000	d small
D	D	LD020000	D capital
e	e	LE010000	e small
E	E	LE020000	E capital
E	\(*E	LE020000	Epsilon capital
é	\(e’	LE110000	e acute small
É	\(E’	LE120000	acute capital
è	\(e‘	LE130000	e grave small
ê	\(e^	LE150000	e circumflex small
ë	\(e:	LE170000	e diaeresis/umlaut small
f	f	LF010000	f small
F	F	LF020000	F capital
g	g	LG010000	g small
G	G	LG020000	G capital
h	h	LH010000	h small
H	H	LH020000	H capital
H	\(*Y	LH020000	Eta capital
i	i	LI010000	i small
I	I	LI020000	I capital

I	\(*I	LI020000	Iota capital
í	\(i’	LI110000	i acute small
ì	\(i‘	LI130000	i grave small
î	\(i^	LI150000	i circumflex small
ï	\(i:	LI170000	i diaeresis/umlaut small
j	j	LJ010000	j small
J	J	LJ020000	J capital
k	k	LK010000	k small
K	K	LK020000	K capital
K	\(*K	LK020000	Kappa capital
l	l	LL010000	l small
L	L	LL020000	L capital
m	m	LM010000	m small
M	M	LM020000	M capital
M	\(*M	LM020000	Mu capital
n	n	LN010000	n small
^{n}	\(nS	LN011000	n small superscript
N	N	LN020000	N capital
N	\(*N	LN020000	Nu capital
ñ	\(n~	LN190000	n tilde small
Ñ	\(N~	LN200000	tilde capital
o	o	LO010000	o small
O	O	LO020000	O capital
O	\(*O	LO020000	Omicron capital
ó	\(o’	LO110000	o acute small
ò	\(o‘	LO130000	O grave small
ô	\(o^	LO150000	o circumflex small
ö	\(o:	LO170000	o diaeresis/umlaut small
Ö	\(O:	LO180000	diaeresis/umlaut capital
p	p	LP010000	p small
P	P	LP020000	P capital
q	q	LQ010000	q small
Q	Q	LQ020000	Q capital
r	r	LR010000	r small
R	R	LR020000	R capital
P	\(*R	LR020000	Rho capital
s	s	LS010000	s small
S	S	LS020000	S capital
ß	\(ss	LS610000	s sharp small
t	t	LT010000	t small
T	T	LT020000	T capital
T	\(*T	LT020000	Tau capital
u	u	LU010000	u small
U	U	LU020000	U capital
Y	\(*U	LU020000	Upsilon capital
ú	\(u’	LU110000	u acute small
ù	\(u‘	LU130000	u grave small
û	\(u^	LU150000	u circumflex small
ü	\(u:	LU170000	u diaeresis/umlaut small
Ü	\(U:	LU180000	U diaeresis/umlaut capital
v	v	LV010000	v small
V	V	LV020000	V capital
w	w	LW010000	w small

W	W	LW020000	W capital
x	x	LX010000	x small
X	X	LX020000	X capital
X	\(*X	LX020000	Chi capital
y	y	LY010000	y small
Y	Y	LY020000	Y capital
ÿ	\(y:	LY170000	y diaeresis/umlaut small
z	z	LZ010000	z small
Z	Z	LZ020000	Z capital
Z	\(*Z	LZ020000	Zeta capital
1	1	ND010000	Numeric decimal one
2	2	ND020000	Numeric decimal two
²	\(2S	ND021000	Numeric decimal two superscript
3	3	ND030000	Numeric decimal three
4	4	ND040000	Numeric decimal four
5	5	ND050000	Numeric decimal five
6	6	ND060000	Numeric decimal six
7	7	ND070000	Numeric decimal seven
8	8	ND080000	Numeric decimal eight
9	9	ND090000	Numeric decimal nine
0	0	ND100000	Numeric decimal zero
½	\(12	NF010000	Numeric fraction one-half
¼	\(14	NF040000	Numeric fraction one-quarter
+	+	SA010000	Plus
±	\(+-	SA020000	Plus-or-minus
<	<	SA030000	Less-than
=	=	SA040000	Equals
>	>	SA050000	Greater-than
÷	\(di	SA060000	Divide
∩	\(ca	SA380000	Intersection, logical product
∞	\(if	SA450000	Infinity symbol
≡	\(==	SA480000	Identity symbol
≤	\(<=	SA520000	Less-than-or-equal
≥	\(>=	SA530000	Greater-than-or-equal
≈	\(Ae	SA700000	Nearly equals
•	\(Md	SA790000	Dot multiply, middle dot
£	\(Lb	SC020000	Pound
$	$	SC030000	Dollar
¢	\(ct	SC040000	Cent
¥	\(Y-	SC050000	Yen
Pt	\(Ps	SC060000	Peseta
ƒ	\(fg	SC070000	Florin or guilder
`	‘	SD130000	Grave
`	\(ga	SD130000	Grave
ˆ	^	SD150000	Circumflex
˜	~	SD190000	tilde
[	\(Ul	SF010000	Upper left joint
	\(Ll	SF020000	Lower left joint
]	\(Ur	SF030000	Upper right joint
	\(Lr	SF040000	Lower right joint
+	\(Mb	SF050000	Middle joint
I	\(Tm	SF060000	Top middle joint
	\(Bm	SF070000	Bottom middle joint

├	\(Lm	SF080000	Left middle joint
┤	\(Rm	SF090000	Right middle joint
─	\(Hb	SF100000	Horizontal bar
│	\(Vb	SF110000	Vertical bar
#	#	SM010000	Number
%	%	SM020000	Percent
&	&	SM030000	Ampersand
*	*	SM040000	Asterisk
*	\(**	SM040000	Math star
@	@	SM050000	At
[	[	SM060000	Left bracket
\	\	SM070000	Reverse slash
]	]	SM080000	Right bracket
{	{	SM110000	Left brace
¦	|	SM130000	Vertical bar, logical OR
¦	\(br	SM130000	Vertical bar, logical OR
¦	\(or	SM130000	Vertical bar, logical OR
}	}	SM140000	Right brace
°	\(de	SM190000	Degree symbol
º	\(ML	SM200000	Ordinal indicator - masculine
ª	\(FE	SM210000	Ordinal indicator - feminine
§	\(SS	SM240000	Section symbol (USA), paragraph symbol (Europe)
§	\(sc	SM240000	Section symbol (USA), paragraph symbol (Europe)
■	\(Ss	SM470000	Solid square, histogram
¬	\(no	SM660000	Logical NOT, end of line
	\(SP	SP010000	Interword space
!	!	SP020000	Exclamation point
¡	\(I!	SP030000	Exclamation point, inverted
"	”	SP040000	Quotation marks
"	\(QM	SP040000	Quotation marks
"	\(RQ	SP040000	Right Quotation marks
"	\(LQ	SP040000	Left Quotation marks
´	\(fm	SP050000	Foot mark
´	\(aa	SP050000	Acute
(	(	SP060000	Left parenthesis
)	)	SP070000	Right parenthesis
,	,	SP080000	Comma
_	_	SP090000	Continuous underscore
_	\(ul	SP090000	Underline
-	-	SP100000	Hyphen
-	\-	SP100000	Hyphen
-	\(hy	SP100000	Hyphen
-	\(mi	SP100000	Minus
-	\(em	SP100000	Em-Dash
-	\(en	SP100000	En-Dash
/	/	SP120000	Slash
:	:	SP130000	Colon
;	;	SP140000	Semicolon
?	?	SP150000	Question mark
¿	\(I?	SP160000	Question mark, inverted
»	\(> >	SP170000	Left angle quotes
«	\(lh	SP170000	Left angle quotes
»	\(> >	SP180000	Right angle quotes

»	\(rh	SP180000	Right angle quotes
	\(RS	SP300000	Required space
σ	\(*s	GS010000	Sigma small
Σ	\(*S	GS020000	Sigma capital

REFERENCES

You may want to obtain copies of the following documentation:

- *IBM 3812 Pageprinter Introduction and Planning Guide*, G544-3265
- *IBM 3812 Pageprinter Guide to Operations*, S544-3267
- *IBM 3812 Pageprinter Programming Reference*, S544-3268
- *IBM 3800 Printing Subsystem Model III Font Catalog*, SH35-0053

This page intentionally left blank.

IBM/4.3 Console Emulators

ABSTRACT

This paper explains the need for, and design of, console emulators in IBM/4.3. It contains the following sections:

1. **Overview** describes the need for console emulators.
2. **Emulator Package Functions** describes the functions provided.
3. **Output Emulator Interface** describes the interface between the emulator and the display-dependent routines.
4. **Input Emulator Interface** describes the interface between the devices and the input emulator.
5. **Window Manager Device-dependent Routines** describes the low-level routines available for controlling a graphics cursor.
6. **System Interface to the Emulator** lists all the routines necessary for a device driver to interface with an emulator.
7. **Console Driver's Relationship to IBM/4.3** shows the levels of flow between the console driver and the standard IBM/4.3 system.
8. **User Interface to the Emulator** describes the systems calls to the console driver and the mouse.
9. **Files Included with the Emulator** defines the files contains in the emulator package.

1. OVERVIEW

This emulator package was developed under IBM/4.3 to support the complex data streams characteristic of advanced workstations. Traditional line disciplines and console driver interfaces are not powerful enough to manage elaborate bit-mapped displays. Sophisticated keyboards, mouse devices, and multiple consoles add complexity to console management.

Emulators written using the IBM/4.3 emulator package can handle normal line-discipline I/O functions as though they were normal *tty* hardware drivers. The emulator package also supports window-manager device-dependent routines, allowing an emulator to act as an interface to a window-manager/graphics system and to control screen output.

1.1. Bit-Map Terminal Requirements

Bit-map terminals are bit-addressable; they deal not with characters, but with individual bits. To represent a single character on the screen, a bit-map terminal turns bits at different locations on or off. In scrolling, all bits on a bit-map screen move up a line at a time, while bits on the bottom line turn off. Some bit-map terminals handle some or all of this in hardware. However, to act as a normal *glass tty* console, each terminal must also be able to perform standard I/O operations. Supporting multiple display types requires code to emulate a glass tty on each display without reproducing the same code for each.

An emulator package solves the inability of bit-map displays to deal directly with characters. It provides a standard interface needed by the low-level, device-dependent drivers and called by the higher-level line disciplines. The device-dependent drivers contain functional procedures that determine where a character appears on the screen, while the device-independent line disciplines determine what the character looks like. When requested to display information on the screen, an emulator package works between the two to ensure that the correct character appears at the correct location.

1.2. Output Emulators

Emulator code intercepts characters and analyzes them according to the type of tty emulated, then calls the appropriate routines to operate on the display. Using this design, any emulator works on any display on the workstation without knowing anything about the display. This type of emulator is an *output emulator*.

1.3. Input Emulators

The same design approach applies to input emulation. Any keyboard or mouse on the system must be able to pass data to the user in a given format. An *input emulator* accepts and deciphers data appropriately.

2. EMULATOR PACKAGE FUNCTIONS

The emulator package performs the following functions:

- Initializes each display present on the workstation.
- Provides a default emulator, defined for any particular hardware. An application does not have to choose an emulator at start-up.
- Allows multiple displays to run on the system simultaneously. Each display can be associated with a different process. This allows a separate login to run on each display.

The emulator package also allows the user to:

- Decide which display should be the default on system boot.
- Reinitialize any hardware or emulator.
- Select from a set of existing emulators for a display.
- Switch between the displays currently open on the workstation. When you switch screens, the focus of the keyboard and mouse moves to that display. This allows you to run a different window system on each display and press a hot-key to switch between them.
- Find and change the hot-key.
- Lock out the user or the system from a display. User lockout is useful for window managers that want to keep other applications from taking over the screen. Kernel or system lockout is useful when you don't want the kernel to attempt to use a non-existent display; for example, when an adapter has no display attached to it.

3. OUTPUT EMULATOR INTERFACE

An emulator needs general information about the display it uses, such as the number and width of lines that fit on the screen in the current font.

The following device-dependent procedures support any basic output emulator:

(1) Determine whether the display is present.

(2) Initialize the display.

(3) Position the cursor anywhere on the display.

(4) Display a character at the cursor position by putting up a bitmap from an internal data font.

(5) Blank a given section of the display (by character).

(6) Move a group of lines on the display.

(7) Print screen contents on the standard printer.

(8) For the IBM 6152 Academic System, change the screen mode.

The following structure from <*machinecons/screen_conf.h*> describes the interface between the emulator and the display-dependent routines. From the top down to *flags* is the standard glass tty information; other entries are described later. This structure is initialized in */sys/machinecons/screen_conf.c.*

```
struct screen_sw {
        char    *name;                  /* Name of display */
        int     (*probe)();             /* Probe for screen */
        int     (*init)();              /* Initialize screen */
        int     (*s_putc)();            /* Put character on screen */
        int     (*pos_cur)();           /* Position cursor on screen */
        int     (*blank)();             /* Blank a section of screen */
        int     (*move)();              /* Move some lines on screen */
        int     (*printscreen)();       /* Routine to print screen */
        char    *rwaddr;                /* Read & writable addr on screen */
        short   lines;                  /* Number of lines on screen */
        short   width;                  /* Width of screen in characters */
        short   vbits;                  /* Vertical number of screen bits */
        short   hbits;                  /* Horizontal number of screen bits */
        int     flags;              /* Some flags about the screen */
        int     def_oute;           /* Default output emulator */
        int     (*pos_loc)();       /* Position locator on screen */
        int     (*load_loc)();      /* Load locator description */
        int     (*show_loc)();      /* Show locator on Screen*/
        int     (*hide_loc)();      /* Hide locator on Screen */
        int     (*apa_init)();      /* All points addressable screen init */
        int     (*color_table)();   /* Change color table */
        int     (color_entries;     /* Number entries in color table */
        int     (*put_status)();    /* Put status (smart devs only) */
        char    *addr;              /* Screen base address */
};
```

The synopsis below from *screen_conf.h* shows the interface to the above structure. The emulator need only use the following routines and attributes:

```
/* Character Attributes */
#define NORMAL_VIDEO            0x00
#define BLINK                   0x01
#define REVERSE_VIDEO           0x02
#define UNDERLINE_VIDEO         0x04
#define HI_INTENSITY            0x08
#define LITERAL_VIDEO           0xff

/* Color Table Values */
#define FOREGROUND_COLOR        0x01
#define BACKGROUND_COLOR        0x00
#define SCREEN_RED              0x04

/* Color Table Flags */
#define COLOR_SET               0x00    /* set the color table entry to color        */
#define COLOR_FG_INC            0x01    /* increment the fg color table entry        */
#define COLOR_FG_DEC            0x02    /* decrement the fg color table entry        */
#define COLOR_BG_INC            0x03    /* increment the bg color table entry        */
#define COLOR_BG_DEC            0x04    /* decrement the bg color table entry        */
#define REVERSE_COLOR           0x50    /* reverse the fg and bg color entries       */
#define CHANGE_DISPLAY_MODE     0x60    /* change display mode                       */
#define ENABLE_BLINK            0x70    /* enable possibility of blink               */
#define ENABLE_BG_INTENSITY     0x80    /* enable possibility of fg high intensity   */
#define ENABLE_FG_INTENSITY     0x90    /* enable possibility of bg high intensity   */

/* Monochrome displays are only interested in the high bit of a color */
#define SCREEN_HIGH_BIT  0x80000000

/* Tell the emulator the printf is from the kernel */
#define SCREEN_KERNEL    0x2

/* Defines for calling console screen-dependent switched routines */

#define screen_putc(c, screen_attr, fg, bg) (*screen_sw[WS].s_putc) (c, screen_attr, fg, bg)

/* Put the status out (smart devices only, i.e. aed) */
#define screen_put_status(pos,str) (*screen_sw[WS].put_status)(pos,str)
```

```
/* color table select */
#define screen_color_table(entry,red,green,blue,flags) (*screen_sw[WS].color_table)(entry,red,green,blue,flags)

/*Move cursor to x,y position */
#define pos_cursor(x, y) (*screen_sw[WS].pos_cur) (x, y)

/* blank with screen_attribute from start coordinates to end coordinates */
#define screen_blank(s_a, sy, sx, ey, ex, fg, bg) (*screen_sw[WS].blank) (s_a, sy, sx, ey, ex, fg, bg)

/* Macro for blanking a line */
#define blank_line(s_a, line, fg, bg)
                screen_blank(s_a, line, 0, line, SCREEN_WIDTH-1, fg, bg)

/* move line1 . . . line2 to dest */
#define screen_move(l1, l2, dest) (*screen_sw[WS].move) (l1, l2, dest)

/* Position screen locator on screen at x,y position with msbox restriction */
#define pos_locator(x, y, msbox) (*screen_sw[WS].pos_loc) (x, y, msbox)

/* Load a new screen locator description with msbox restriction */
#define load_locator(c, msbox) (*screen_sw[WS].load_loc) (c, msbox)

/* Show screen locator with msbox restriction */
#define show_locator(msbox) (*screen_sw[WS].show_loc) (msbox)

/* Hide screen locator */
#define hide_locator(bounds) (*screen_sw[WS].hide_loc) (bounds)

/* APA Screen init */
#define apa_initialize() (*screen_sw[WS].apa_init) ()

/* Real hardware addresses for the displays */
#define screen_addr(n) screen_sw[n].addr

#define lp_pos_cursor(col,line,dev)    (((line)*screen_sw[(dev)].width) + (col))
```

4. INPUT EMULATOR INTERFACE

The low-level interface to the input emulator is not defined as strictly as the one for the output emulator. Basically, a set of hardware routines in *keyboard.c*, *kls.c*, *speaker.c*, and *mouse.c* can be called by an input emulator to control the keyboard, speaker, and mouse. The input emulator receives a data interrupt from the keyboard or mouse. The emulator deciphers the data and tracks the state of the device; then passes its processed data to the user through a *line discipline* or some other emulator-specific method, such as shared memory.

Few procedures are needed to control a keyboard for setting the auto keyclick rate, bell tone, and key characteristics (repeat rate, make/break, etc.). Because few workstations support multiple keyboards simultaneously, there is no need to set up a switch table for these hardware routines. Workstation mouse devices also have few control operations (set sampling or resolution rate); input emulators do not yet deal with these operations directly, but instead pass *ioctl* system calls to the appropriate driver.

5. WINDOW MANAGER DEVICE-DEPENDENT ROUTINES

Listed below are the device-dependent routines available with the *screen_sw* low-level routines. These are normally used in an input emulator to control the graphics cursor. These are only used by x10 and are only available if pseudo-device xtenemul is defined. Not all display devices have graphics cursor support.

pos_loc()
: Position the locator at a given coordinate on the display.

load_loc()
: Load a locator bitmap for the display. This is the locator until the next load_loc.

show_loc()
: Make the locator visible and keep showing when positioned. Usually used after a hide_loc.

hide_loc()
: Make the locator invisible, but do not affect the tracking.

apa_init()
: Initialize the display for graphic operations needed by the locator. Useful for displays with hardware cursors/locators.

6. SYSTEM INTERFACE TO THE EMULATOR

A new emulator should be easy to add to a system and must be able to coexist with other emulators. An emulator has many functions and system entry points similar to those of a tty hardware device driver. The main difference is that emulators funnel through a single console driver and call a common set of hardware routines, while device drivers deal directly with the hardware.

The emulator switch table below lists all routines necessary for a device driver to interface with an emulator. To add an emulator to the system, add the following routines in the switch table structure shown below. This structure (modeled after line disciplines) is declared in *screen_conf.h*, and the table is initialized in *screen_conf.c*.

```
/*
 * Emulator line control switch.
 */
struct emulsw
{
        int     (*e_open)();
        int     (*e_close)();
        int     (*e_read)();
        int     (*e_write)();
        int     (*e_ioctl)();
        int     (*e_rint)();
        int     (*e_putc)();
        int     (*e_select)();
        int     (*e_putstatus)();  /* to put up status information */
        int     (*e_color_table)();
};
```

Each emulator, depending on its needs, has the following entry points in the kernel:

e_open()
: Open the emulator to do any necessary initialization. Perform initial operations such as clearing the screen, initializing the cursor, and positioning the cursor on the screen.

e_close()
: Close the emulator; do any cleanup necessary.

e_read()
: Read data from the emulator (used only by input emulators). For most emulators, this routine forwards the read request to the user-defined line discipline. The read routines in line disciplines currently perform the operations necessary for this routine. This consists of taking the already-received characters off a *clist queue* and passing them to the user program's read buffer.

e_write()
: Write data to the emulator (used only by output emulators). This procedure takes a character stream passed from the user-level program. Most emulators call the line discipline specified by the user to do any character preprocessing. Again, the line discipline routines already perform the necessary duties for this routine. This routine and the e_read() routines are in the emulator package for completeness and to allow flexibility. Some specialized emulators do use these routines for other than calling the associated line-discipline routines (see *buf_emul*(4)).

e_ioctl()
: I/O control to emulator for changing or setting characteristics of the emulator or performing operations that do not fit into the normal interface to the emulator. This routine should return a (− 1) if the command is not recognized.

e_rint()
: Receive interrupt to emulator (used only by input emulators). The emulator receives an interrupt from a driver's interrupt routine and processes the data depending on the type of interrupt received. This procedure passes the processed input data to the user-assigned line-discipline input routine. Some specialized window-manager emulators do not forward these data to a line discipline, but do their own queuing and interacting with a window manager.

e_putc()
: Put a character on the display (used only by output emulators). This emulator routine receives a character from a user's write or kernel printf. The emulator deciphers the data and interprets character strings before passing the appropriate characters to the hardware putc routine. This procedure makes use of the screen switch table (screen_sw) in calling the device-dependent routines to perform the emulation on any display.

e_select()
: Select call to emulator (used only by input emulators). This routine is used to perform the normal select duty of informing the user process when new data are ready.

e_putstatus()
: Put status call to emulator (used only by output emulators). The emulator takes the passed string and places it at an offset on the status line.

e_color_table()
: Changes the screen's color table (used only by output emulators).

7. CONSOLE DRIVER'S RELATIONSHIP TO IBM/4.3

The following diagram shows how the parts of the system described relate to the standard parts of a IBM/4.3 system:

<table>
<tr><th colspan="3">Console System Diagram</th></tr>
<tr><td colspan="2">User Application</td><td></td></tr>
<tr><td>System Call Interface</td><td>Event Queue in Shared Memory</td><td><b>User Level</b></td></tr>
<tr><td>Line Discipline Package</td><td>Emulator Package</td><td rowspan="2"><b>Kernel Level</b></td></tr>
<tr><td>Standard Device Drivers</td><td>Low Level Display Dependent Routines</td></tr>
<tr><td colspan="2">Displays/Keyboard/Speaker System or Serial Mouse</td><td><b>Hardware</b></td></tr>
</table>

The above is a conceptual view of the system. It does not show all parts and interfaces, but indicates the levels of flow. The console driver routes normal driver requests to the correct display and input/output emulator, depending on the minor device specified.

The following shows how the minor device number maps to an emulator and display:

Output Emulator Flag	**Bus**	**Display#**
bits 7 - 4	bit 3	bits 2 - 0
0 or 1	0 or 1	0 - 7

The following is a list of currently-used displays supported by IBM/4.3:

Console Displays

Display #	Symbolic Name	Description
0	CONS_GEN	Generic console (current display)
1	CONS_AED	ACIS experimental display (stream ordered)
2	CONS_APA16	IBM 6155 Extended Monochrome Graphics Display (bitmap)
3	CONS_APA8C	IBM 6154 Advanced Color Graphics Display (bitmap)
4	CONS_APA8	IBM 6153 Advanced Monochrome Graphics Display (bitmap)
5	CONS_EGA	IBM 5154 Enhanced Color Graphics Display (character)
6	CONS_MONO	IBM 5151 Monochrome Display (character)
7	CONS_MPEL	IBM 5081 Display Adapter (stream ordered)

Console Displays (continued)		
0	CONS_VGA	IBM Planar Video Graphic Array Adapter (character)
1	CONS_IBM8514	IBM 8514/A adapter (stream ordered) *

* There is no glass tty support for the 8514/A. It uses the vga emulator mode.

If the bus bit is set, opening the device grants access to the I/O bus. Without this bit, it is necessary to open */dev/bus* to gain access to I/O space. This bit is provided for compatibility; new applications should open */dev/bus* if they need bus access.

The emulator field in the minor device number tells the console driver that the default glass_tty input/output emulators will be used (0), or indicates that a non-standard output emulator will be used (nonzero). If a non-standard output emulator is used, the system restores the display to the standard state (*default emulators*) when the device is closed.

The following is a list of emulators currently available:

Emulators Available		
Emulator #	Symbolic Name	Description
0	E_KBDINPUT	Intelligent keyboard mapping input emulator (standard)
1	E_STDOUTPUT	Standard output emulator
2	E_IBMOUTPUT	IBM 3101 output emulator
3	E_ANSIOUTPUT	ANSI output emulator (not implemented)
4	E_XINPUT	X event queuing input emulator
5	E_BUFOUTPUT	Buffering output emulator
6	E_AED	Raw AED microcode interface emulator

The *default emulators* for each display are:

Default Input/Output Emulators for each Display		
Display	Input Emulator	Output Emulator
AED	E_KBDINPUT	E_STDOUTPUT
APA16	E_KBDINPUT	E_IBMOUTPUT
APA8C	E_KBDINPUT	E_IBMOUTPUT
APA8	E_KBDINPUT	E_IBMOUTPUT
MONO	E_KBDINPUT	E_IBMOUTPUT
MPEL	E_KBDINPUT	E_IBMOUTPUT
VGA	E_KBDINPUT	E_IBMOUTPUT

7.1. Input From Keyboard Scenario

(1) User types character on keyboard.

(2) Receive interrupt in keyboard driver, keyboard.c, interrupt routine *kbdint()*. This routine extracts key code from the hardware.

(3) Call emulator receive-interrupt routine from the switch table indexed by the current *input focus* after setting the emulator structure flag, indicating that this was a keyboard interrupt.

(4) Emulator checks whether this was a keyboard interrupt. If so, it either translates code into a character and calls normal line-discipline routine for this console with the translated character, or performs some emulator-specific function such as storing the raw key code in a shared-memory area (X-like) and setting a semaphore, also in shared memory, to inform the user process that a new event has arrived.

(5) If a line-discipline input routine is called, it performs its previously-described normal input (editing/mapping) and passes the result to the user through the read system call interface.

(6) If a shared-memory queue interface is used, the user process notes the queue update through the semaphore and proceeds to read the data from the shared memory without performing a read or any other system call.

(7) In either of the above two cases, a select would be satisfied if the user had previously done a select call.

7.2. Output To Display Scenario

This scenario applies only to glass-tty operations. The window-manager system goes directly to the display hardware through its own graphics routines. If a user tries to write through the system to the display while a window manager is controlling the display, a special *buffer emulator* is called instead of a glass-tty emulator, as in the scenario below.

(1) The user performs a *write* system call with a buffer of data for the display. These data consist of ASCII data or display order streams.

(2) The console write routine then calls the write routine of the output emulator selected by the minor device number.

(3) For most emulators, the output-emulator write routine then forwards these data to the line-discipline write routine specified by the user. Certain emulators, such as the buffer emulator, intercept these data and capture them for printing later.

(4) The line discipline interprets the data and calls the console start routine to print the ASCII characters.

(5) The console driver's start routine loops through dequeuing each character and calling the output emulator put-character routine, *e_putc,* for each character.

(6) The output emulator put-character routine then deciphers the data and calls the appropriate device-dependent routine to display the character or perform the display command.

8. USER INTERFACE TO THE EMULATORS

The user interface to the emulators consists of system calls to the console driver (see *cons*(4) and *mouse*(4)).

8.1. Interface to Keyboard Input and Display Output

8.1.1. Standard Interface

In the simplest case, a user program still performs the same operations as in the past, allowing previously-written programs to work without change. The following lists the normal scenario and what the emulator package does:

Standard Console Interface Device				
Permissions	owner	major	minor	device
crw-rw-rw-	root	0	0	/dev/console

(1) An application such as login opens */dev/console.* Since */dev/console* is the special CONS_GEN minor device, in "open" the output is mapped to the current console-focused display. The display at which a program is started is the display associated with the process. The system starts the default input and output emulators for that display. Input is received only if the input focus is set to CONS_GEN.

(2) The application reads or writes to the file descriptor returned from the open system call. The system maps writes to the CONS_GEN minor device to the display with the current input focus. This causes the appropriate display-indexed input/output emulators to be used. Input is focused to CONS_GEN if no console tty devices and no console graphic devices are open, except when using the default input emulator (E_KBDINPUT). If the input focus is not on CONS_GEN, the input focus follows the output focus.

(3) The application exits or closes the */dev/console* file descriptor. The system closes the input/output emulators and the stream for the display with the current input focus.

Mapping */dev/console* to the current display is important for most applications that do not need to know on which display they are running. This mapping is also important at system boot time where the single-user shell does not know on which display it is starting and is simply mapped to what the system chooses as the starting input focus. But some applications (for example, login, window manager) need to know on which display they should start. Therefore, the following devices are provided to support the displays available on IBM/4.3:

Standard TTY-like Display Devices

Permissions	Owner	Major	Minor	Device
crw-rw-rw-	root	0,	1	/dev/ttyaed
crw-rw-rw-	root	0,	2	/dev/ttyap16
crw-rw-rw-	root	0,	3	/dev/ttyap8c
crw-rw-rw-	root	0,	4	/dev/ttyapa8
crw-rw-rw-	root	0,	5	/dev/ttyega
crw-rw-rw-	root	0,	6	/dev/ttymono
crw-rw-rw-	root	0,	7	/dev/ttympel
crw-rw-rw-	root	0,	1	/dev/ttyvga
crw-rw-rw-	root	0,	2	/dev/tty8514 *

* /dev/tty8514 is currently a null device.

To start an application on a particular display, reassign its standard input and outputs to any of these devices or have the application specifically open one of them. The system routes output from the application to the appropriate display and its default emulators. Input to the application from the keyboard only occurs when the console focus is assigned to that display. To switch between open displays, press the specified hot-key for your system or use an application which performs an *ioctl* system call to set the input focus.

The different displays on the system are specified in the */etc/ttys* file that tells the system to start logins on each of the displays. A user can hot-key to the desired display and, after logging in, run any application needed. Any application started on that display stays associated with it, because it was started while the console focus was on that display. Because this is an application-transparent mapping to that display taking place in the kernel, a user can log on simultaneously to as many displays as needed.

8.1.2. Nonstandard Interface

For applications that call for a specific non-default emulator, the following devices are provided:

Nonstandard Display Devices				
Permissions	Owner	Major	Minor	Device
crw-rw-rw-	root	0,	65	/dev/aed
crw-rw-rw-	root	0,	66	/dev/apa16
crw-rw-rw-	root	0,	67	/dev/apa8c
crw-rw-rw-	root	0,	68	/dev/apa8
crw-rw-rw-	root	0,	69	/dev/ega
crw-rw-rw-	root	0,	70	/dev/mono
crw-rw-rw-	root	0,	61	/dev/mpel
crw-rw-rw-	root	0,	65	/dev/vga
crw-rw-rw-	root	0,	66	/dev/ibm8514

The emulator flag is nonzero for each of these devices. This indicates to the system that a nonstandard input and/or output emulator is going to be used on this display and that, on close, the system should return the display to its default emulators. The system opens the standard input emulator and a special buffering output emulator, because most applications that open the nonstandard device take over the screen and want output from other sources (such as kernel *printf*s) to be buffered and displayed when the display is closed. See *bufemul*(4) for more information. If bus access is required on open, add 8 to each minor device number.

The following is a list of commands available through the *ioctl* system call to the console emulator package:

Ioctl Commands to Emulator Package

Command	Read	Write	Description
CON_SELECT_SCREEN	Yes	Yes	Output focus is set to display number (arg > 0) or to next display in list (arg < 0). Previous display number is returned.
CON_GET_SCREEN	Yes	No	Just returns the current output focus display number.
EIGETD	Yes	No	Get the number of the current input emulator for this display.
EOGETD	Yes	No	Get the number of the current output emulator for this display.
EISETD	Yes	Yes	Set the input emulator and return the previous for this display.
EOSETD	Yes	Yes	Set the output emulator and return the previous for this display.
CON_INIT_SCREEN	No	Yes	Initialize the specified display (arg > = 0) or this display (arg < 0).
CON_GET_FOCUS_CODE	Yes	No	Get the current keyboard code for setting the console focus (xemul only).
CON_SET_FOCUS_CODE	Yes	Yes	Set the current keyboard code for setting the console focus (xemul only), and return the previous code.
SCRIOCGETF	Yes	Yes	Get screen control flags for the given display number.
SCRIOCSETC	Yes	Yes	Set screen control flags for the given display number.

All of the above commands take integer arguments except the last two. SCRIOCGETF and SCRIOCSETC use the following structure:

```
struct screen_control  {
                                int       device;  /* which screen/display to control */
                                int       switches; /* Flags for this screen */
};
```

Flags for Each Display

Flag	Description
CONSDEV_PRESENT	Display is present on this system.
CONSDEV_KERNEL	Display is available to the kernel.
CONSDEV_USER	Display is available to the user.
CONSDEV_INIT	Display has been initialized for output.
CONSDEV_TTY	Tty display has been opened directly by minor device number.
CONSDEV_GRA	Graphics display has been opened directly by minor device number.

All of the above *ioctl* system calls are device-independent controls for dealing with the emulators.

Each emulator has its own set of *ioctl*s for its own emulation purposes. These other *ioctl*s are used in window-manager emulators for operations such as passing/positioning the mouse locator for/on the display. See the man page for any particular emulator for more information.

8.2. Mouse Input Interface

The interface to the system mouse is similar to that of the keyboard. If the generic mouse device */dev/mouse,* minor device 0, is opened, the mouse input is attached to the display which has the current input focus. Opening any other mouse device attaches the mouse input stream to the process only when the input focus is on the associated display.

The following mouse devices are provided:

Mouse Input Devices

Permissions	Owner	Major	Minor	Device
crw-rw-rw-	root	15,	0	/dev/mouse
crw-rw-rw-	root	15,	1	/dev/msaed
crw-rw-rw-	root	15,	2	/dev/msapa16
crw-rw-rw-	root	15,	3	/dev/msapa8c
crw-rw-rw-	root	15,	4	/dev/msapa8
crw-rw-rw-	root	15,	5	/dev/msega
crw-rw-rw-	root	15,	6	/dev/msmono
crw-rw-rw-	root	15,	7	/dev/msmpel
crw-rw-rw-	root	15,	1	/dev/msvga
crw-rw-rw-	root	15,	2	/dev/ibm85

The *system mouse* driver is essentially the same as those in other 4.3BSD-based systems. This driver hooks into the emulator package by selecting a special line discipline. The line discipline filters the mouse data and then passes a generic data packet to the user through normal read system calls or calls the user-specified emulator with the data packet. This line discipline is explained in the *tb*(4) manual page.

For compatibility, the default line discipline may be set using the upper four bits of the minor device number. To get the device interface specified in *mouse*(4), use the discipline MSLINEDISC from <*machineio/mouseio.h*>. Any new software that uses the mouse should set its desired discipline explicitly.

9. FILES INCLUDED WITH THE EMULATOR

The following tables briefly explain the files contained in the emulator package. The tables specify files according to function. Each table states where the files are located and describes what each file contains. Tables with a column marked *User* distinguish between purely kernel files and user/kernel-shared include files. These user include files are needed to access emulator functions.

Emulator Control Files

/sys/machinecons

File	User	Description
cons.c	no	Console driver routes requests to appropriate emulator and its input/output device or to the emulator controller. Console driver is also responsible for console message forwarding.
consdefs.h	no	This file contains hardware interface information about system input devices. Emulators as well as device dependent routines use this to interface with each other and the hardware.
consio.h	yes	Defines which displays and screen controls/flags are available. The ioctl commands and structure for screen_control are in this file. This file is indirectly included by screen_conf.h.
consvars.h	no	Emulator control variables are declared here.
bus.c	no	I/O bus control driver, which allows access to the I/O bus on a per-process basis. Window managers that need to get directly at the display from user space should open /dev/bus.
screen_conf.c	no	Where all the displays and emulators are configured for the system. This file also contains the emulator control routines discussed in "Emulator Package Functions", above.
screen_conf.h	yes	Where all the structures, defines, and macros for the emulator package live. This contains all the macros for an emulator to interface with the device-dependent routines as well as the ioctl information for the user to interface with the emulator package.

Emulators

/sys/machinecons

File	User	Description
aed.c aeddefs.h	no	AED raw microcode graphic emulator
buf_emul.c	no	Buffering emulator, which saves messages sent to display, then flushes them when the output emulator is changed
ibm_emul.c ibmemul.h	no	IBM3101 output emulator; takes a considerable subset of IBM3101 commands defined in tcap
kbd_emul.c kbd_emul.h kbde_codes.h	no	A keyboard emulator which allows mapping of key codes to a character stream
std_emul.c std_emul.h	no	Standard output emulator routines, which send raw characters to the display on output. This output emulator is used for displays that perform their own emulations.
x_emul.c	no	X window system input emulator, which queues up keyboard and mouse events into a memory area shared between kernel and user. This emulator also has a variety of ioctls for controlling the locator on a display, as well as performing other X-related functions (e.g. tracking the cursor, etc.).
xio.h	yes	X-dependent structures and defines for kernel and usr process
qevent.h	yes	Event Queue structures and defines used by the X emulator

New/Changed Line Discipline files		
/sys/sys and /sys/h		
File	User	Description
tty_conf.c	no	Line discipline configure file
tty.h	yes	Line discipline structures and defines
tty_tb.c	no	Normal tablet line discipline changed to support system/serial type mouse devices also; also changed for forwarding data packets to input emulator if specified
tbdefs.h	no	Tablet/mouse generic data packet structures and defines
tbioctl.h	yes	Ioctl commands and structures
pc_bios.h	yes	PC BIOS interrupt structures and defines
pc_afi.h	yes	PC Advance Function Interface structure and defines
pc_afidata.h	yes	PC Advance Function Interface structures and defines

Low Level Output Display Dependent Files	
/sys/machinecons	
File	Description
aedtty.h	Macros and defines for interfacing with the glass tty microcode for the AED display
aed_tty_mcode.h	Glass tty microcode for download to the AED display
aedloc.c	AED locator low-level device-dependent routines
aedtty.c	AED glass tty low-level device-dependent routines
apa16loc.c	APA16 locator low-level device-dependent routines
apa16tty.c	APA16 glass tty low-level device-dependent routines
apa16tty.h	APA16 device-dependent structures and define
apa16tty_font.h	APA16 font for glass tty emulation
apa8cloc.c	APA8 color locator low-level device-dependent routines
apa8ctty.c	APA8 color glass tty low-level device-dependent routines
apa8ctty.h	APA8 color device-dependent structures and define
apa8loc.c	APA8 locator low-level device-dependent routines
apa8tty.c	APA8 glass tty low-level device-dependent routines
apa8tty.h	APA8 device-dependent structures and define
apa8tty_font.h	APA8 and APA8 color font for glass tty emulation
apa_fontblt.c	Generic routines for font manipulation on APA displays

Low Level Output Display Dependent Files	
/sys/machinecons	
File	Description
apa_structs.h	Generic structures and defines for font manipulation on APA displays
egatty.c	EGA glass tty low level device dependent routines
egatty.h	EGA device-dependent structures and defines
ega_init.h	Initialization sequences for the EGA
ega_font.h	Fonts for the EGA
apaaed.h	Structures and defines for dealing with the AED as an APA display
lptty.c	APA print screen support and print
monotty.c	Monochrome glass tty low-level device-dependent routines
monocons.h	Monochrome device-dependent structures
monotty.h	Monochrome device-dependent defines
mpeltty.c	Megapel glasstty driver
mpeltty.h	Megapel structures and defines
mpeltty_font.n	Megapel font
mpeltty_mcode.c	Megapel microcode (standalone only; standca)
ibm8514.c	IBM 8514/A glass tty low-level device-dependent routines (ca atr)
ibm8514.h	Structure and defines for IBM8514/A (ca atr)
vga.c	IBM Planar Video Graphics Array glass tty low-level device-dependent routines (ca atr)
vgadefs.h	Structure and defines for Planar Video Graphics Array (ca atr)
display_debug.h	ATR displays debugging information

Low Level Keyboard Device Dependent Routines	
/sys/machinecons	
File	Description
keyboard.c	System keyboard hardware routines
keyboard.h	System keyboard hardware structures and defines
kls.c	Keyboard/mouse/speaker common routines
kls.h	Keyboard/mouse/speaker low level defines

System Mouse Device Driver		
/sys/machincio		
File	User	Description
mouse.c	no	Driver for system mouse
mouseio.h	yes	System mouse structures and defines; also includes ioctl controls/defines for user processes
mousereg.h	no	System mouse driver declarations
speaker.c	no	Speaker driver
speakerio.h	yes	Speaker structures and defines
speakervar.h	no	Internal speaker data structures

The Remote Virtual Disk System

This article is an updated version of several articles written by J. H. Saltzer, J. Van Sciver, L. W. Allen, P. Prindeville, and Michael Greenwald at MIT between 1983 and 1986. The original articles, based on MIT's Project Athena, have been rewritten and include additions and changes for the IBM RT PC, IBM 6152 Academic System, and IBM/4.3.

This article describes the Remote Virtual Disk (RVD) system for use with IBM/4.3 on the IBM RT PC and the IBM 6152 Academic System. It contains the following chapters:

1. **Overview** contains background information on RVD.
2. **RVD Structure** describes the structure of the RVD system.
3. **Installing RVD** describes RVD installation.
4. **RVD Protocol Specification** describes the RVD communications protocol (optional reading).
5. **RVD Control Protocol Specification** describes the RVD remote server maintenance protocol (optional reading).

1. OVERVIEW

The Remote Virtual Disk (RVD) system is a network service that provides a client computer with the appearance of removable-media disk drives and an unlimited number of removable disk packs. The removable disk packs are actually stored in private regions of large disks on an RVD server. When a remote disk pack is "spun up", it appears to most software to be just another disk drive. Although read and write requests are actually accomplished by sending messages across the network to the server, on a local area network the performance of a remote disk pack is only slightly less than that of a local fixed disk.

RVD is a very simple system. Its only addition to the usual list of functions of a hardware disk is remote access. Its design makes little use of operating system features, so it is fairly independent of the operating system. An RVD client may be implemented for any operating system that allows installation of device drivers, and an RVD server may be implemented under any operating system that permits access to either disk partitions or large files. A server that runs under one operating system may be used by a client that runs under another.

1.1. Remote Virtual Disk Packs

A remote virtual disk pack is a portion of a real disk, located on an RVD server. RVD packs are named and allocated by an administrator for the particular RVD server. The name (a character string) and the size (measured in sectors of 512 bytes) are negotiated between the administrator and the prospective user. Once allocated, the space is reserved on the physical disk for the lifetime of the RVD pack.

When a client computer uses ("spins up") an RVD pack, the client specifies one of two modes of access: read-only access or read/write exclusive access. These modes follow the usual rules for read/write compatibility: there may be several simultaneous readers, or exactly one exclusive-mode user of any one virtual disk.

Access to an RVD pack may be protected by passwords, with a separate one for each of the modes of access. Thus one might protect an RVD pack used as a group library by requiring one password (or no password at all) for read access, and a different password for exclusive access. A private RVD pack might use the same password for both modes. It is also possible, by arrangement with the server's administrator, to specify (by internet address) a preferred client that may spin up a password-protected RVD pack without providing the password.

When a new RVD pack is allocated, the first thing one normally does is create an initialized, empty file system on that pack.

1.2. Supporting Tools

Normally one treats a remote disk pack just like any removable storage medium; all standard commands and tools are applicable. In addition, there are a few specialized tools that are useful in managing the remote disk system.

Client management: The RVD client code is packaged as a driver. There are commands that display information and state of the client part of the RVD system.

Server management: The RVD server is designed to be operated from a distance via a network connection. Client commands are available to invoke any remote management operation of the server.

Remote pack management: A high-speed copy command provides a high-performance way of duplicating the contents of one remote pack onto a second one.

1.3. Hazards

RVD is an example of a distributed system, in which failures of the server and of the data communication network can occur independently of failures in the client. This failure-independence can lead to some situations that might not have been anticipated, or that are so rare when using a local disk that they are not handled well, in the writing of the software of the client operating system or applications.

The most common failure is that a packet is lost in the data communication network. The RVD client-server protocol includes a sophisticated request-retry procedure that will immediately and automatically recover from occasional lost packets. It will also automatically recover from short network outages (up to a minute or so,) although some application programs may have timers that get impatient with the delays involved in waiting for a network to recover.

Generally, longer network outages, or crash and restart of the server, are reported as errors back to the invoking file system; whether or not the user is able to recover depends on the application's response to these errors.

If the client crashes and requires rebooting or is powered down while it has RVD packs spun up, it loses its local record of spunup packs, but the server still has a record. There is a general cleanup function, named *rvdflush*(8), that sends a request to a server to spin down all RVD packs associated with this client. It is general good practice to spin down all packs at the end of a client session, and to run *rvdflush* at the beginning of every client session, in case the previous session ended with a crash.

Most operating systems have a file system integrity-checking program. It is usually necessary to use such a program to review, and if necessary, to repair, the contents of an RVD pack following a crash, just as with a local disk. It is also good practice to run such a program just before using any newly spunup pack, especially if that pack may be used by other clients.

As a general rule, all software works correctly with RVD unless it is written to be dependent on hardware parameters of specific physical disks. However, most operating systems have some programs that know too much; those programs must be avoided.

1.4. Network Protocols

The Remote Virtual Disk system uses two network protocols, named RVD and RVDCTL. The client driver and the server communicate with the RVD protocol, to perform spinup, disk read/write, and spindown. The RVD protocol is a transport protocol, using the Internet Protocol (IP) as its base. It is described in detail in Chapter 3 of this article.

The Remote Virtual Disk Control (RVDCTL) protocol supplies a Remote Virtual Disk server with both operating instructions and information about its configuration. An RVD server process comes into existence with no knowledge of the physical configuration of the system in which it is embedded or the logical configuration of the (possibly already existing) virtual disk packs it is to manage. By supplying this information via a network connection instead of from files on the server host, it becomes possible to administer all aspects of server operation remotely. RVDCTL is an application protocol, using the User Datagram Protocol (UDP) as its base. It is described in detail in Chapter 5 of this document.

2. RVD STRUCTURE

This section describes the general structure of the RVD server. The RVD server is organized as a user program that requires little of the underlying operating system apart from access to a disk device. It is started by invoking (as root, or in */etc/rc.local*) the command *rvdsrv*(8), normally run in the background. The server has no configuration description files. Instead, it opens a network listening port (the RVDCTL port) and expects that someone will send its configuration and all operating instructions to it on that port.

2.1. Authorization

The installer of RVD should create a file named */etc/rvd/rvdauthor*, owned by root and readable only by root. It contains an unencrypted ASCII authorization password that the server demands on most uses of the RVDCTL connection. If the contents of *rvdauthor* are changed, the change takes effect the next time the control request "require_authorization" is sent to the server, or the next time the server is started.

2.2. File Placement

The server program is installed in */usr/ibm/rvdsrv*. There is a set of server-management commands in */usr/ibm.*

For convenience, the directory */etc/rvd* contains the RVD initialization data file, the text of any user message, and RVD operation logs. These files are described in the next two sections.

2.3. Initialization

As mentioned, the RVD server, once started, takes its configuration initialization, as well as operating instructions and also instructions to change its configuration, as a series of operation requests sent to it over a UDP socket. Since all operation requests for the RVD server are ASCII text strings, RVD server initialization is conventionally accomplished by maintaining somewhere on the server an ASCII text file containing the sequence of initialization instructions. A program named *rvdsend*(8) can be used to send the contents of that file over the control connection.

If a user message is posted at the server, it is a good idea to keep a copy of its text in a standard place so that it can be reposted if the server needs to be reinitialized.

It is convenient to divide the initialization instructions into two files, one of which, *rvddb*(5), initializes only the configuration, while the other (*rvdenable*) contains instructions to start the server operating. A typical invocation of RVD at boot time then appears as the following sequence in the file */etc/rvd/rvdstart*:

```
/etc/rvd/rvdsrv -l 11 &                # start server with logging
/etc/rvd/rvdsend /etc/rvd/rvddb        # set up server configuration
/etc/rvd/rvdsend /etc/rvd/rvdenable    # tell server to start work
```

where the file */etc/rvd/rvdenable* contains:

```
operation=allow_spinups
mode=5
operation=require_authorization
```

One reason for dividing the initialization instructions into two files is that there is a configuration-management program, *vddb*(8), that provides a convenient user interface for creating and changing a file that contains the disk-pack configuration. A second is that while the system is in single-user mode, the file *rvdenable* may temporarily be replaced with an alternate file that starts the server in a different way, perhaps by forbidding any but operations use.

2.4. Logging

The RVD server uses the *syslogd*(8) facility. All RVD server logging is to the *syslog* identifier LOCAL7. The RVD server uses the conventional *syslog* levels as follows:

ALERT	(0)	serious server problems from which recovery is unlikely
ERROR	(1)	recoverable server errors such as bad disk blocks
INFO	(2)	spinups, spindowns, and name exchanges
	(8)	errors made by clients: bad passwords, attempt to read packs that aren't spun up, etc.
DEBUG	(4)	all read and write requests
	(16)	complete packet level trace of RVD operation

The numbers in the above list are logging classes. Items in class zero are always logged; RVD has a log control system that allows one to turn each of the other classes of logging on or off independently. The command invocation line for the server includes a parameter (the sum of the class numbers) to turn on the initial logging classes. The command *rvdchlog*(8) sends a control protocol request to change the classes of events that are logged.

The *syslog* configuration file (*/etc/syslog.conf*) can direct all logging output from LOCAL7 to an appropriate file, for example, */etc/rvd/rvdlog*. In addition, if there is a system log that is reviewed daily, RVD logging output of levels ALERT and ERROR might appropriately be directed there, too.

Logging all read/write requests or every packet produces a noticeable performance degradation of the server, so it is not recommended for normal operation. Logging spinups and client errors provides information about usage of the RVD service and also often records entries that suggest particular clients are misconfigured or are having some problem. If spinups and spindowns are logged, a busy server can fill 100 Kbytes of log in a day. Thus it is a good idea that a crontab (see *cron*(8)) entry invoke a nightly script to move the RVD log aside and start a new one.

2.5. Remote Management

The RVD server is designed to allow all management to be done remotely. The program that manages the initialization data, named *vddb*, can be run either locally on the server, or elsewhere in the network. Remote management of several servers can be accomplished by setting up a directory on the management host that contains a copy of the *rvddb* initialization file for each server to be managed, named with the network name of the server. When *vddb* is invoked with the name of the server, all configuration changes that the system administrator requests are made to the central *rvddb* initialization file for that server and they are also performed, via the control connection, on the server itself. If the server is set up to reboot and reinitialize itself automatically from a local copy of the *rvddb* initialization file, the system administrator should, when *vddb* has completed, copy the newly modified initialization file to the server.

2.6. Remote Partition Management

The command *savervd*(8) copies virtual disk packs to tape, and *zaprvd* (see *savervd*(8)) does the reverse. If the server does not have a tape drive attached, it is possible to do this operation remotely, by using RVD twice. The basic trick is to set up links in */dev* so that there are two names for every disk partition managed by the RVD server. The server uses one of these names (e.g. *vdsrv1*) for virtual pack assignment; it uses the other name (e.g.

rdvdsrv1) for a single virtual pack that overlays the entire partition. That overlaying virtual pack can then be spun up on a remote system that has a tape drive available. Once spun up, the raw RVD device that represents the spunup virtual pack can be treated just like a local disk partition on the remote system, and another copy of the RVD server can be operated on that system. A symbolic link to that raw RVD device, but with the same name as the link used for virtual pack assignment on the original system (e.g. *vdsrv1*) allows a copy of the *rvddb* from the first server to be reused at the remote site. This trick allows *savervd* and *zaprvd* to think they are operating on the original server.

3. INSTALLING RVD

This section describes how to install RVD on an IBM RT PC or an IBM 6152 Academic System running IBM/4.3. The first part gives a description of installation; the second part is a step-by-step description of installation.

3.1. Description of RVD Installation

As the RVD system follows a client-server model, there are two kinds of installation procedures. Two scripts in */etc/rvd* facilitate these procedures: *rvd.mkserver* for the server machine, and *rvd.mkclient* for the client machine. These scripts are user-friendly front ends for RVD installation procedures.

3.1.1. Installing an RVD Server Machine

Installing an RVD server machine involves the following tasks:

- Creating the required virtual disks
- Configuring the file systems that the RVD disk packs will hold
- Starting the RVD server program

3.1.1.1. Creating Virtual Disks

Creating the disks requires assigning local physical disk drives to hold the required file systems, and configuring a data base file *rvddb* for the RVD software (see *rvddb*(5) and *vddb*(8)). The *rvd.mkserver* script will prompt the user for existing devices, and will create links (named *vdsrv0* through *vdsrv9*) to these devices. These links serve as RVD's interface to the physical disks. During the creation of the *rvddb* data base, the *vddb* program prompts the user for information associated with the partitioning of the chosen disk drives and the allotment of these partitions to RVD disk packs. The program uses the information supplied by the user to create the *rvddb* file, which is in turn used by the RVD server program.

3.1.1.2. Configuring the File Systems

The administrator for the RVD server system will determine what file systems the RVD packs will hold. The decision is not particularly sensitive, except in the case where one mounts the */usr* file system via RVD. Because the */usr* file system is particularly large, it is desirable to mount it via RVD on client machines. However, the */usr* file system contains files and directories which must be local to the client machine. (Generally, any file or directory which must be writable must be local.) When "usr" is specified as a pack name, the *rvd.mkserver* script creates and makes symbolic links to a new file system, */site*, which is set up to contain the necessary local files and directories. Some of these files and directories, such as */usr/spool*, are put in */site* by default; others are put in the *usr* RVD pack by default. The user is prompted to determine the disposition of files and directories in */usr* which are not recognized by *rvd.mkserver*.

Because there will most likely be occasional changes made to the *usr* pack, it will be necessary to keep a backup *usr* pack on the server machine. Making changes to an RVD pack requires that that pack be spun up in the exclusive read/write mode, an operation which cannot be performed on a pack spun up by any other RVD user. In general, when configuring an RVD server machine, one should keep in mind the need for space for a backup copy of any RVD pack that will require changes while in service.

3.1.1.3. Starting the RVD Server Program

Finally, *rvd.mkserver* prompts the user to set the RVD command authorization password, if it has not already been set in the file */etc/rvd/rvdauthor*, and the RVD server program *rvdsrv* is started. Full RVD services are available at the completion of the *rvd.mkserver* script.

3.1.2. Installing an RVD Client Machine

Installing an RVD client machine involves the following tasks:

- Creating the *rvdtab* data base (see *rvdtab*(5)) and, perhaps, modifying */etc/fstab*
- Rebooting the client machine to reconfigure and remount its file systems if the */usr* file system is being mounted by RVD.
- Setting up the *rvdusr* file tree if the *rvdusr* pack is to be used

3.1.2.1. Creating the rvdtab Data Base

The first thing the *rvd.mkclient* script does is check for the existence of the *rvdtab* file (nominally at */etc/rvd/rvdtab*). If the *rvdtab* file already exists, *rvd.mkclient* echoes its contents to the user. Then *rvd.mkclient* prompts the user for the information required to build or extend the *rvdtab* file. When the file is completed, its contents are echoed to the user for inspection.

After dealing with the *rvdtab* file, *rvd.mkclient* creates the directories used as mount points for selected RVD packs, e.g. *rvdusr* and *src*. If the *usr* pack is going to be used, the */site* directory is created and */etc/fstab* is modified so that the local */usr* file system will be mounted at */site* upon reboot. The *usr* pack spun up from a server comes with symbolic links to */site* for all files and directories required to be kept in local (non-RVD) storage. These links are set up automatically when the RVD *usr* pack is created using *rvd.mkserver.*

3.1.2.2. Rebooting the Client Machine

Rvd.mkclient gives its user a 30-second warning before calling for a system shutdown in one minute. After shutdown, the system automatically reboots, coming back up with the appropriate RVD client configuration. At this time any packs designated as "default" in the creation of *rvdtab* are spun up (barring network and server problems). If the *usr* pack is being used, it is spun up at this juncture; any network or server problems preventing the spinup of the *usr* pack will cause the reboot process to hang. (This is one good reason to have the *usr* pack available on more than one server machine.)

3.1.2.3. Setting up the rvdusr File Tree

After the client machine is rebooted, the user needs to spin up any packs currently needed but not specified as default packs in the creation of *rvdtab.* If the *rvdusr* pack is being spun up to have parts of the */usr* file system accessed via RVD, the user will want to execute the *rvdusr.config* script. This script goes through all files and directories at the level directly below the root of the */rvdusr* file system (that is, the *rvdusr* pack) one at a time, and tells the user exactly how much local disk space can be saved by accessing that file or directory via RVD. This part of the process takes about 30 minutes. The user is then prompted for the choice between RVD and local storage, and the appropriate action is taken. If the choice is local storage, nothing is done (i.e. the file/directory stays on the local */usr* file system disk). If the choice is to put the file/directory on RVD, it is removed from the local disk and replaced by a symbolic link to the corresponding file or directory in */rvdusr*. The space previously occupied by that file or directory on the local disk is then free for other uses.

3.1.3. Notes on Installing RVD

When a client machine is mounting its */usr* file system via RVD, it is prudent to have the */usr* pack available on more than one server machine. A client machine cannot afford to be without a */usr* file system, since the executable code for many fundamental IBM/4.3 utilities resides in the */usr* directory. Thus the loss of access to the server for the *usr* pack, for any reason, is a serious handicap and should be avoided through having multiple servers available with this pack. *Rvd.mkclient* reminds the user of this when creating the *rvdtab* file.

The */usr/src* file system is also a good candidate for installation as an RVD pack. It must be mounted separately from the usr pack, as it is a distinct file system. Note that this file system has the classic characteristics of a good RVD pack candidate: it requires a large amount of disk space, will not be written to by the average user, and will be written to or changed (by anyone) only infrequently. Such file systems can be accessed in read-only mode by many users simultaneously, and rarely require any attention on the part of the system administrator.

Care should be taken if new workstations are to be installed at your site via network connections (see "RVD Installation Steps" below). The source machine for such installations should in general NOT be an RVD server; if it is, the target machine will end up being an identical server machine. (However, this might be useful for servers featuring the *usr* pack.) Also, target machines should be installed over the network before their configuration as RVD clients. This maintains full flexibility in that configuration process.

3.2. RVD Installation Steps

This section describes the steps involved in installing an RVD server machine and an RVD client machine, and the scripts (*/etc/rvd/rvd.mkserver* and */etc/rvd/rvd.mkclient*) used to do this.

3.2.1. Installing an RVD Server Machine

Installing an RVD server machine is done by using the *rvd.mkserver* script, which resides in */etc/rvd*. This script provides an easy and thorough procedure for the installation. This document describes the prompts provided by *rvd.mkserver* and the appropriate user responses.

3.2.1.1. Starting the rvd.mkserver Script

Before using *rvd.mkserver*, you must first *su* to root, because of the major changes being made. Failure to do so will result in the following error message:

you must su to root to run this script

3.2.2. Creating Virtual Disks

Upon successful invocation, *rvd.mkserver* responds with the following:

creating virtual disk drives:
we will create links to existing devices, which
should be drivers for existing local disk drives.

here is a list of currently mounted physical devices:

What follows is the output of the command *mount* which shows the available mounted physical disk drives. These drives are candidates for use in creation of the RVD virtual disk drives. Note that there may also be unmounted disk drives which could also be used.

There are 10 links, named *vdsrv0* through *vdsrv9*, that *rvd.mkserver* makes available. *Rvd.mkserver* automatically keeps track of the next available link and presents the following prompt, where X is between 0 and 9:

next available link is vdsrvX - make link? (y/n)

A response of **y** or **yes** (referred to as an "affirmative"response in this chapter) will continue the process of creating the link; all other responses will cause *rvd.mkserver* to move on to creating the *rvddb* data base file (see below).

An affirmative response to the above prompt generates this prompt:

which device to link to? (< list of devices > CR)

where *< list of devices >* is a list of (mounted) candidate devices.

Type your response in the following format:

/dev/disk_drive*X*

or

disk_drive*X*

or **< CR >**, where *disk_driveX* is an existing *block special* or *character special* file. A carriage return (**< CR >**) will go on to the next available device; an incorrect response will generate the error message:

device must be a disk driveX, (your reply) is not

Rvd.mkserver will loop until it receives a satisfactory reply. Once that reply is received, it responds as follows, where X is again between 0 and 9:

linking /dev/vdsrvX to (your reply)

3.2.2.1. Creating the rvddb Data Base File

Next *rvd.mkserver* will respond:

creating rvd data base /etc/rvd/rvddb...

If there is already a file */etc/rvd/rvddb* in existence, *rvd.mkserver* responds with:

/etc/rvd/rvddb already exists - it looks like this:

whereupon the contents of that file are echoed to the terminal. (Note that the file may be empty!)

The *rvddb* file is a data base of RVD virtual disk partitions (see *rvddb*(5)). *Rvd.mkserver* gives the user an opportunity to create or modify *rvddb* with the following prompt:

do you wish to modify /etc/rvd/rvddb? (y/n)

There is a standard RVD utility program *vddb* (see *vddb*(8)) used for creating and modifying *rvddb* which *rvd.mkserver* calls upon an affirmative response, with this message:

calling vddb (see vddb(8)).

As *vddb* is a fairly obscure program to the new user, *rvd.mkserver* offers the user a glimpse of what a typical session with *vddb* might look like, with the prompt:

would you like to see an example first? (y/n)

An affirmative response generates the following:

```
        a typical session might go like this:

Ready
> add physical
Physical disk file name: /dev/vdsrv0
Size in 512-byte blocks: 88536    (note: see diskpart(8) for more info on this)
Are you sure (y or n)? y

Ready
> add virtual
Virtual disk name: src
Description: /usr/src file system
Read-only password:
Exclusive password: src_password
Shared password:
Disk size in 512-byte blocks: 88536
Allowable modes: 5
Owning host (<CR> for none):
Physical disk name (<CR> for any): /dev/vdsrv0
Are you sure (y or n)? y

Ready
> quit

        hit return key to begin vddb session:
```

Pressing **<Enter>** causes *rvd.mkserver* to invoke *vddb*. Invoking *vddb* requires the RVD *command authorization password*, which *rvd.mkserver* will prompt you for. Note that this password will be the null string (equivalent to a carriage return) if it has not been previously set, thus the prompt for a password may be ignored. If it has been set it will be found in */etc/rvd/rvdauthor*, where it is readable by *root*.

3.2.2.2. Configuration of a usr RVD Pack

After finishing the session with *vddb*, *rvd.mkserver* will search the */etc/rvd/rvddb* data base file to see if a pack named *usr* is being created. If it finds mention of such a pack in */etc/rvd/rvddb*, a complex series of actions is initiated. If your installation does not include a *usr* pack, you may skip this section.

Rvd.mkserver signals that it has found that a *usr* pack is being created with this message:

```
/usr will be an rvd disk pack, must make arrangements...
```

The *usr* RVD pack is used to hold the */usr* file system. The */usr* file system is complex, and contains several directories which generally must be local (i.e., non-RVD) because they require write permission. Thus *rvd.mkserver* proceeds to create a directory */site* in the *root* (or /) file system to hold these local directories. First *rvd.mkserver* checks to see if there is enough free space on the *root* file system for the directories which it knows that */site* must hold. If *rvd.mkserver* finds that there is not enough space it will display the following:

```
there is not enough space on / for /site
we need X kilobytes for /site
there are only Y kilobytes available on /
exiting
```

If this happens, *rvd.mkserver* exits and the system administrator must reconfigure the local disks to provide the necessary space if he or she wishes to have the *usr* RVD pack.[1]

If *rvd.mkserver* finds that there is sufficient space for */site*, it will respond with the following:

> *creation of /site will require X kilobytes in / file system*
> *there are Y kilobytes free in / file system*
> *do you wish to proceed with creating /site? (y/n)*

An affirmative response generates:

> *proceeding...*

Any other response causes *rvd.mkserver* to exit with:

> *exiting*

The above choice is given because the figures given by *rvd.mkserver* may indicate that the creation of */site* will leave an unacceptably small amount of free space on the *root* file system. Thus the system administrator may wish to reconfigure the local disks to allow more space on the *root* file system.

If the user has chosen to proceed with the creation of */site, rvd.mkserver* displays the following:

> *creating /site on / file system to store local /usr files*

At this juncture *rvd.mkserver* creates and/or set the correct access modes on */site*, then proceeds to move those directories it knows in advance must be local from */usr* to */site* with this message:

> *moving (adm guest msgs preserve spool tmp) from /usr to /site:*

Before actually moving a directory, *rvd.mkserver* will check to see if that directory has already been linked to */site*. If it is, this message will appear:

> */usr/(directory) is already linked to /site/(directory)*

where (directory) is the directory in question. Also, *rvd.mkserver* will check to see if a directory of that name already exits in */site*. If it finds one, this appears:

> */site/(directory) already exists - best check it. we will proceed.*

and no action is taken. If neither of the above errors occurs the move and link are made and the directory name is echoed. When finished with all directories, *rvd.mkserver* outputs:

> *...done.*

Next *rvd.mkserver* moves and links */usr/lib/crontab* to */site* with the message:

> *moving and linking /usr/lib/crontab to /site/lib/crontab*

If */usr/lib/crontab* is already linked to */site*, this message will appear:

> */usr/lib/crontab already linked to /site/lib/crontab*

Rvd.mkserver has a list of directories it expects to find in all */usr* file systems, and automatically disposes as RVD or local. It also checks for directories not in that list and prompts the user to determine their disposal. That process begins with this

[1] Note that there may be even more space required for */site* than is indicated by *rvd.mkserver* at this step. The estimate given is based only on the space required by the directories that *rvd.mkserver* knows *a priori* must be local. There may be other directories peculiar to your site that must also be local, and thus accommodated on */site*.

message:

```
looking at other files & directories in /usr to make local or remote
default is local (not on rvd pack)
```

One by one *rvd.mkserver* gets the size of each directory, reviews the space left on the *root* file system, and responds with:

```
looking at /usr/(directory)...
there are X kilobytes left on /site
/usr/(directory) requires Y kilobytes
put /usr/(directory) on usr rvd pack? (default is move to /site) (y/n)
```

An affirmative response means that the directory will stay on the */usr* file system rather than being moved to */site.* Such directories will then be available to RVD clients on the *usr* RVD pack. Each client that uses the *usr* pack will save the "Y" kilobytes of local disk space required by that directory, but that client will generally not be able to write to that directory. Thus *write permission* and *space savings* are the two factors that should be weighed in deciding this response. An affirmative response generates the reply:

```
/usr/(directory) will be on usr rvd pack
```

A non-affirmative response means that the directory should be local. Thus *rvd.mkserver* attempts to move the directory to */site* and make a link from */usr* to */site.* If the directory is already a link to */site rvd.mkserver* displays:

```
/usr/(directory) already linked to /site/(directory)
```

If a directory of the same name already exists on */site*, this appears:

```
/site/(directory) already exists - best check it. we will proceed.
```

and *rvd.mkserver* proceeds without taking any action.

Otherwise, if all goes well, this appears:

```
moving and linking /usr/(directory) to /site/(directory)
```

3.2.2.3. Setting the RVD Command Password

The RVD system requires a *command authorization password* to accompany all network commands for remote maintenance of the RVD server. If this password is not already set (in the file */etc/rvd/rvdauthor*) *rvd.mkserver* prompts the user to set that password:

```
setting rvd command authorization password
enter new password:
```

After the user has entered the password, *rvd.mkserver* responds:

```
installing new password in /etc/rvd/rvdauthor... done.
```

3.2.2.4. Setting RVD Operation Modes

Another requirement for RVD operation is a file */etc/rvd/rvdenable* which contains information on the operational modes of the RVD server. If it doesn't already exist, *rvd.mkserver* creates that file and displays the following:

```
creating /etc/rvd/rvdenable file... done.
```

Note that this step requires no input from the user.

3.2.2.5. Starting the RVD Server Daemon

Finally, *rvd.mkserver* starts the RVD server daemon after displaying this message:

starting rvd server program

and prompting for the *command authorization password* (which again will be null if not previously set in */etc/rvd/rvdauthor*). If the RVD server daemon was already running at this point *rvd.mkserver* displays:

rvd server program is already running - no further actions will be taken

3.2.2.6. Setting the RVD System Message

An RVD server has an RVD system message option. This message is used to communicate with RVD clients about new packs, changes to existing packs, etc. *Rvd.mkserver* calls *rvdsetm*(8) to allow the user to set this RVD system message. *Rvdsetm*(8) prompts for the message, which you type in and terminate with a <CTRL-D>. Then *rvdsetm*(8) will prompt for the RVD *command authorization password*, which may again be the null string.

The installation of the RVD server is now complete. *Rvd.mkserver* exits with a status of 0, and all RVD services are now available on this server machine.

3.2.3. Installing an RVD Client Machine

Installing an RVD client machine is done using the *rvd.mkclient* script, which resides in */etc/rvd*. This script provides an easy and thorough procedure for the installation. This section describes the prompts provided by *rvd.mkclient* and the appropriate user responses.

3.2.3.1. Starting the rvd.mkclient Script

Before starting you must *su* to root, because of the major changes being made. Failure to do so will result in the following error message:

you must su to root to run this script

Upon successful invocation, *rvd.mkclient* will first check to see if the RVD devices exist in */dev*. If not, *rvd.mkclient* will create them with a call to *MAKEDEV* after presenting this message:

making rvd devices...

If *rvd.mkclient* makes the RVD devices, it will next show the user what devices it created:

```
here is a list of currently available rvd devices:

crw-rw-rw-  1 root     17,  0 Aug  7 11:21 /dev/rvd0a
crw-rw-rw-  1 root     17,  1 Aug  7 11:21 /dev/rvd1a
crw-rw-rw-  1 root     17,  2 Aug  7 11:21 /dev/rvd2a
crw-rw-rw-  1 root     17,  3 Aug  7 11:21 /dev/rvd3a
crw-rw-rw-  1 root     17,  4 Aug  7 11:21 /dev/rvd4a
crw-rw-rw-  1 root     17,  5 Aug  7 11:21 /dev/rvd5a
crw-rw-rw-  1 root     17,  6 Aug  7 11:21 /dev/rvd6a
crw-rw-rw-  1 root     17,  7 Aug  7 11:21 /dev/rvd7a
crw-rw-rw-  1 root     17,  8 Aug  7 11:21 /dev/rvd8a
crw-rw-rw-  1 root     17,  9 Aug  7 11:21 /dev/rvd9a
brw-rw-rw-  1 root      6,  0 Aug  7 11:21 /dev/vd0a
brw-rw-rw-  1 root      6,  1 Aug  7 11:21 /dev/vd1a
brw-rw-rw-  1 root      6,  2 Aug  7 11:21 /dev/vd2a
brw-rw-rw-  1 root      6,  3 Aug  7 11:21 /dev/vd3a
brw-rw-rw-  1 root      6,  4 Aug  7 11:21 /dev/vd4a
brw-rw-rw-  1 root      6,  5 Aug  7 11:21 /dev/vd5a
```

```
brw-rw-rw-  1 root      6,   6 Aug  7 11:21 /dev/vd6a
brw-rw-rw-  1 root      6,   7 Aug  7 11:21 /dev/vd7a
brw-rw-rw-  1 root      6,   8 Aug  7 11:21 /dev/vd8a
brw-rw-rw-  1 root      6,   9 Aug  7 11:21 /dev/vd9a
```

Note that this process requires no input from the user.

3.2.3.2. Creating rvdtab

The next step *rvd.mkclient* takes is to give the user a chance to create or modify the */etc/rvd/rvdtab* file (see *rvdtab*(5)). *Rvdtab* stores information concerning the RVD packs that this client uses; *rvd.mkclient* will allow the user to easily build or extend this file. *Rvd.mkclient* announces this phase of the installation procedure with this message:

```
creating/modifying /etc/rvd/rvdtab...
```

First *rvd.mkclient* checks for the existence of */etc/rvd/rvdtab*. If the file is found, *rvd.mkclient* gives this message:

```
/etc/rvd/rvdtab already exists - it looks like this:
```

After this the contents of */etc/rvd/rvdtab* are echoed to the user. (Note that the file may be empty!)

Then *rvd.mkclient* gives the user the chance to add new RVD packs to the *rvdtab* file with this prompt:

```
enter a new pack into database? (y/n)
```

An affirmative response to this query causes *rvd.mkclient* to prompt the user for all the fields required for an *rvdtab* entry. This process will be repeated until *rvd.mkclient* receives a non-affirmative response from the user. If *rvd.mkclient* receives such a non-affirmative response when there was previously no *rvdtab* file and no packs have been added to *rvdtab*, this message appears:

```
you must create an rvdtab file to have an rvd client machine
exiting
```

and the script exits.

3.2.3.2.1. Pack Name

First *rvd.mkclient* prompts for the RVD pack name. This name must be the same as the pack name on the server machine, typically something such as *usr* or *src*. (Note that the pack *name* is not necessarily the same as the pack's *pathname*; for example, the pack named *usr* is mounted as */usr*.) *Rvd.mkclient* asks for the pack name:

```
what will the pack's name be?
```

Type the pack name after this prompt.

3.2.3.2.2. RVD Pack Mounting Status

Next *rvd.mkclient* asks the user what the mounting status of this pack is to be. The pack may be specified as an essential pack which must be mounted for client operation (as with the *usr* pack), in which case there will be several attempts to spin it up at boot time; as a "default" pack for which one attempt to spin up should be made at boot time; or as a pack which is not to be mounted automatically at boot time. *Rvd.mkclient* asks for this information with this message:

```
is this pack to be mounted by default (d) or is it
absolutely-must-be-mounted (a)? (default is no mount)
enter a, d, or <Enter>:
```

If *rvd.mkclient* does not recognize the user's response, it goes back to the beginning of the add-new-pack loop with this error message:

flag (response) is not known; starting over

where (response) is the user's input.

If the name of this pack is *usr*, *rvd.mkclient* does not prompt the user for input, but rather responds with:

usr pack is being made a must-be-mounted pack

This done because the *usr* pack is assumed to contain the */usr* file system, without which the functionality of a UNIX operating system machine is severely compromised.

3.2.3.2.3. RVD Pack Spinup Modes

Now *rvd.mkclient* needs to know what spinup mode(s) to allow for this pack. There are three options: read-only, exclusive read/write, or both. *Rvd.mkclient* prompts the user with:

what spinup mode, read-only (r), exclusive read/write (x),
or both (rx), do you want? (default is read-only)

If *rvd.mkclient* gets either a null string or an unrecognized response it responds with:

using default (read-only) mode

and will make this pack a read-only pack in *rvdtab.*

3.2.3.2.4. Server Machine for RVD Pack

Next *rvd.mkclient* needs to know upon which server this pack resides. Since having the *usr* pack spun up is essential to the normal operation of an RVD client which uses that pack, *rvd.mkclient* reminds its user of the importance of having the *usr* pack available on more than one server for reliability's sake with this message (given only if the pack name is *usr*):

NOTE: we suggest that usr pack be available on more than one server

(Note that only ONE server machine may be specified per entry in *rvdtab.*)

Rvd.mkclient then prompts with:

on what server(s) does this pack reside?

If the response received is null (i.e., < Enter >) *rvd.mkclient* gives this message:

this is not an optional field...

and prompts for the server machine again.

3.2.3.2.5. RVD Drive Number

There are ten drives available for RVD packs. A particular drive number must be specified:

what drive number (0-9)? (no default)

If a null or unrecognized response is received, *rvd.mkclient* responds with:

sorry, must have drive number

and prompts for the drive number again.

3.2.3.2.6. Mount Point

Next comes the issue of where this pack will be mounted. This is an optional field, with defaults for some pack names:

where do you want to mount this file system? (default is /usr for usr pack, /rvdusr for rvdusr, and /usr/src for src)

Here you should enter the *pathname* of the directory where this RVD pack is to be mounted when spun up.

3.2.3.2.7. RVD Pack Password

There may be passwords required for access to an RVD pack. If these passwords exist and are known, they may be entered at this prompt:

enter the password for this pack, if any:

Null responses to the above prompt are typical.

3.2.3.2.8. Comments in rvdtab

There may be (single line) comments associated with each pack in the *rvdtab* file. This comment is optional, and can be typed in response to this prompt:

any comment to associate with this pack in the data base entry?

3.2.3.2.9. Finishing the Loop

At this point *rvd.mkclient* will ask again if the user wishes to enter another pack in *rvdtab*. The user may enter as many or as few packs as desired. When a non-affirmative response to the offer to enter a new pack causes *rvd.mkclient* to exit that loop and the user has altered *rvdtab* by adding one or more packs, this message appears:

/etc/rvd/rvdtab now looks like this:

and the contents of the *rvdtab* file are echoed to the user.

3.2.3.3. Creating Mount Points

It is the user's responsibility to provide the mount directories for an RVD pack (see *mkdir*(8)). In the case of the *usr* pack, *rvd.mkclient* assumes that this mount point already exists. In the case that RVD packs *rvdusr* and/or *src* have been added to *rvdtab*, and no other mount point was specified by the user when prompted, *rvd.mkclient* will respond:

making /rvdusr as remote mount point

or

making /usr/src as remote mount point

or both. *Rvd.mkclient* creates the directories mentioned, if these messages appear.

3.2.3.4. Rebooting the Client Machine for usr Pack

If the *usr* RVD pack is being used, the file systems on the client machine must be reshuffled to move the */usr* file system to */site*, thus clearing the way for the *usr* pack to be mounted as the */usr* file system. *Rvd.mkclient* will do this automatically, after giving the user this warning:

starting major changes in 30 seconds; interrupt now
if you don't want your /usr mounted via rvd

At this juncture, *rvd.mkclient* will edit */etc/fstab* to have the */usr* file system remounted as */site*, if not interrupted by the user with <CTRL-C>. Then *rvd.mkclient* gives a second warning:

You have 30 seconds to cancel an automatic system reboot

If not interrupted in this time, *rvd.mkclient* will call for a system shutdown in one minute:

shutting down in 1 minute...

When the machine has been successfully rebooted it will have the *usr* RVD pack in place and spun up. Note that the */usr* file system has been moved to */site* at this point and may have redundant files (to be found on the *usr* RVD pack) which may be culled to save disk space.

3.2.4. Configuring an rvdusr Pack on a Client Machine

The *rvdusr* RVD pack is used when a client does not want to rely upon a server machine for its entire */usr* file system, but would like to access selected directories in */usr* via RVD in order to save space on local disks. When a client machine decides to use the *rvdusr* pack, the system administrator needs to decide which files and directories in */usr* to delete from local storage and link the *rvdusr* pack (which is mounted as */rvdusr* by default). The *rvdusr.config* script found in */etc/rvd* is designed to facilitate this process.

3.2.4.1.1. Invoking rvdusr.config

The user of *rvdusr* may need to have special permissions in order to remove the necessary directories from */usr*. Check the ownership and permissions in */usr* before running this script, or simply *su* to root before using the script.

3.2.4.1.2. Running rvdusr.config

Rvdusr checks to see what directories are available in */rvdusr* and, one by one, offers you the choice of removing the corresponding directory from the local */usr* file system and replacing it with a link to *rvdusr*. After this is done, all accesses to that directory are made via RVD and the space taken up by that directory on the local */usr* file system is freed for other uses. Note that RVD access to files and directories generally *read-only*; this should be taken into account when deciding about whether to keep a directory locally or access it via RVD.

For each directory in */rvdusr rvdusr.config* will first display this message:

looking at /usr/(directory)...

where (directory) is the next directory, alphabetically. At this point *rvdusr.config* gets the amount of space that that directory occupies on the local disk, and comes back with this:

you can save X blocks by putting (directory) on rvd
put (directory) on rvd? (keep local is default) (y/n)

An affirmative response causes *rvdusr.config* to remove (directory) from the */usr* file system and to replace it with a symbolic link to */rvdusr* with this message:

linking /usr/(directory) to /rvdusr/(directory)...

and after a brief pause during which the changes are made:

done

If the response is not affirmative, *rvdusr.config* uses the default action, which is to leave things as they are:

keeping /usr/(directory) on local disk drive

After all directories in */rvdusr* are covered, *rvdusr.config* exits with a status of 0.

3.2.4.1.3. Recovering RVD Directories to Local Storage

Should the user(s) of the RVD client machine ever wish to return any directory of */rvdusr* to local storage in */usr*, the symbolic link to */rvdusr* can be removed and the directory copied from */rvdusr* back to */usr*.

4. RVD PROTOCOL SPECIFICATION

This section is optional reading.

4.1. Introduction

4.1.1. Motivation

The Remote Virtual Disk (RVD) Protocol provides the ability to dynamically attach arbitrary disks of different sizes to arbitrary computers. It is especially useful when the computers are physically remote or when user intervention is impractical (e.g. when local disks are non-removable, removable packs are expensive, or a wide variety of disk sizes is desired). The RVD Protocol allows either exclusive or shared use of remote devices. The latter mode is useful as a low overhead means of sharing read-only data among physically remote machines.

4.1.2. Scope

The Remote Virtual Disk Protocol simply allows network access to additional disk drives. The protocol does not provide any services beyond those provided by existing disk drives. Specifically, the RVD Protocol:

- does not implement sharing, or any form of object storage or naming mechanism, other than that found on any disk drive.
- does not guarantee reliable writes to the disk.
- does not attempt to resolve architectural byte ordering differences among machines.

This design of this protocol has been concerned primarily with simplicity of implementation, ease of use, and performance.

4.1.3. Use

The RVD protocol is layered on top of the DoD IP protocol and an IP protocol number of 66 decimal (102 octal) has been assigned. This protocol corresponds to level three of the ISO networking standard.

The RVD protocol allows a client machine to communicate with a remote server machine's disk drives as if they were local drives. On the client side, the protocol is used by the I/O subsystem software, typically a device driver, to communicate with the remote drive. All client software would then use the client device driver to treat the remote drive as if it were just another local device. On the server side, either an application process or the addition of operating system support could be used to make the connection between protocol requests and local disk requests. The level at which the server is implemented is not part of this specification.

4.2. Specification

The RVD Protocol is used in a client/server scenario. The client makes requests and the server responds. The protocol defines four request/response pairs and an error response. The pairs are (request/response):

SPIN-UP/SPIN-ACK:
The client requests use of one of the server's drives, and the server acknowledges the client's spin-up request.

SPIN-DOWN/SPIN-DOWN-ACK:
The client disconnects from one of the server's drives, and the server acknowledges the disconnection.

READ/BLOCK:
The client requests one or more blocks from a spun-up device, and the server responds with the requested data blocks.

WRITE/WRITE-ACK:
The client writes blocks to a spun-up device, and the server acknowledges one or more write requests.

ERROR:
The server reports a protocol error while processing a client request; e.g., an error response would be used if the client had omitted a required field in an RVD packet. This response is not used for operation errors; e.g., if a failure occurs when writing to a physical disk then this failure is reported in the status word of the WRITE-ACK response.

The request/response dialog is conducted by exchanging packets between the client and server. All RVD packets consist of a standard header followed by data or parameters specific to the packet type. Descriptions of the RVD packet header and of each packet type follow.

4.2.1. RVD Packet Format

Generalized Format of RVD Packet

<Byte 0>	<Byte 1>	<Byte 2>	<Byte 3>
Packet Type	Padding		RVD Version
Drive			
Nonce			
Index			
Checksum			
Reserved			
Specific Parameters			

- Packet Type

  This byte specifies one of nine RVD packet types:

  SPIN-UP
  SPIN-ACK
  SPIN-DOWN
  SPIN-DOWN-ACK
  READ
  BLOCK
  WRITE
  WRITE-ACK
  ERROR

- Padding

  These two bytes are unused by most of the packet types. SPIN-UP and ERROR do allocate the first byte of this field for their own purposes.

- RVD Version

  The client and server set this field to the version number of the RVD Protocol being used. If either party discovers a mismatch between version numbers then an error occurs. If the server makes the discovery it returns a protocol ERROR packet to the client. This packet will describe the mismatch error.

- Drive

  An index that represents the drive number on the client machine. It is used by the client to specify a virtual disk drive on the server. This index is an integer between zero and (2**32)-1 and is encoded as a 32 bit binary integer. Drive numbers are unique on a given client but there is no correlation between drive numbers on different clients. The server uniquely identifies a virtual disk by the client-drive pair. The server returns the given drive number in its response to the client.

- Nonce

  A 32-bit unique identifier. The nonce is an unsigned integer between zero and (2**32)-1, encoded as a 32-bit binary integer. Since the nonce is a fixed range number it will be unique only over a fixed period of time. It is assumed to be unique for an interval of time that is several times the lifetime of a single packet. The nonce is used to identify a request/response dialog between the client and server. As such, the client inserts a nonce value into its request packet and the server will insert the same nonce into the appropriate response packet.

- Index

  The index is a hint to the server on how to find connection specific information. (A connection is a virtual drive/client drive pairing.) The index has no predefined meaning and the server may use it as any manner of hint desired. The only client request which does not specify an index is SPIN-UP. The responding SPIN-ACK packet will contain, among other things, the server Index, representing the connection that was created by the SPIN-UP request. It is up to the client to return this index with every packet that goes out to this virtual drive.

  If the user ever submits an incorrect Index the server will still find the connection information. It will then send an ERROR packet specifying the correct index. The server would still process the request normally.

- Checksum

  A 32-bit *checksum* of the packet. The *checksum* is the only assurance of reliable data transfer. It is assumed that if the *checksum* is correct then the data is the same as on the disk.

  The *checksum* is computed by adding together all the 32-bit words in the packet. The *checksum* field is considered to be zero for this computation. If the packet does not end on a 32 bit boundary, then the *checksum* computation assumes that the packet is padded out by zeros. The low order 32 bits are then used as the checksum. (The 32 bit sum is taken modulo 2**32.) *Checksum* is computed in Vax byte ordering.

- Specific Parameters

  See the descriptions of the packet types for additional parameters.

4.2.2. Packet Format: SPIN-UP

Format of SPIN-UP Packet

<table>
<tr><th><Byte 0></th><th><Byte 1></th><th><Byte 2></th><th><Byte 3></th></tr>
<tr><td>Packet Type</td><td>Mode</td><td>Padding</td><td>RVD Version</td></tr>
<tr><td colspan="4">Drive</td></tr>
<tr><td colspan="4">Nonce</td></tr>
<tr><td colspan="4">Index</td></tr>
<tr><td colspan="4">Checksum</td></tr>
<tr><td colspan="4">Reserved</td></tr>
<tr><td colspan="4">Pack Name</td></tr>
<tr><td colspan="4">Capability</td></tr>
<tr><td colspan="3">Padding</td><td>Blocking Factor</td></tr>
</table>

(See the section on RVD Packet Format for a complete description of the fields in the standard RVD header: Packet Type, RVD Version, Drive, Nonce, Index, and Checksum.)

4.2.2.1. Field Definitions for Packet Type: SPIN-UP

- Mode

  This byte describes the access mode of the virtual drive. Once a virtual drive is spun-up in a particular mode any other client who wants to use that same drive (even from another machine) must open it in the same mode. The mode is one of three values:

  - READ-ONLY gives the client read-only access to the drive. Any other client can read the drive as long as they have also spun it up in READ-ONLY mode.
  - SHARED allows read/write access to the drive by more than one user.
  - EXCLUSIVE gives the client read and write access to the disk, but locks the disk so that no other client can access the disk while it is spun up.

- Index

  this is the only packet type that does not specify an index. The server will return an appropriate index in the SPIN-ACK response packet. This value must then be included in all future client request packets.

- Pack Name

  An ASCII string representing the name of the virtual disk pack that a client wishes to associate with the specified local disk drive. The pack name field is a fixed length string of 32 characters. Each of these characters is represented as an 8-bit byte. The string is null terminated unless the name length is greater than or equal to 32 characters. In that case, the string is truncated to the 32-byte pack name field size.

- Capability

  This field is, like the pack name, a maximum 32-character, null terminated, ASCII string. There are separate capabilities for each drive in each of the spin-up modes. If the mode were, for example, READ-ONLY, then the client would fill in the capability field with the READ-ONLY capability string for this drive.

- Blocking Factor

  This is the read blocking factor, the maximum number of blocks the client can read at one time in a single packet. More exactly, it is the maximum number of 512 byte data blocks that the client will accept in a BLOCK packet type. If the blocking factor is greater than the maximum number of blocks the server can send it will be modified in the SPIN-ACK response packet.

4.2.2.2. Operation

Spinning up a disk establishes a connection between the server's virtual drive and the client's local drive. For example, if the client MYMACHINE wishes to spin up the remote virtual disk "Foo" as his local drive 3, then he sends a SPIN-UP packet to the server. He fills in Packet Type with SPIN-UP and Mode with a valid mode. He fills in Drive with the integer 3. He also supplies the capability for that mode and places the string "Foo" in the Pack Name field.

Upon receipt of the SPIN-UP packet, the server would attempt to fulfill the request. If everything is correct (the drive exists, the capability is correct and so on), the server associates virtual disk "Foo" with drive 3 from client MYMACHINE. From now on, any reference from client MYMACHINE to drive 3 will refer to virtual disk "Foo." The server responds with a SPIN-ACK packet back to the client.

If the server detects an error, it will reply with an ERROR packet. This ERROR packet can be caused by many different classes of errors. First, "real" disk errors; for example, the physical disk containing "Foo" is trashed, or any error that might occur when accessing a physical disk. This is different than the typical case of a computer connected to a physical disk. When accessing a local drive, the error would not be detected until an operation was performed on the drive.

Another type of error is invalid argument values such as a non-existent virtual drive, a bad password, or a bad *checksum*. There can also be inconsistency errors: a disk is already spun up as drive 3, or another client has disk "Foo" spun-up in a conflicting mode. All of these errors will be reported via the ERROR reply.

4.2.3. Packet Format: SPIN-ACK

Format of SPIN-ACK Packet

<Byte 0>	<Byte 1>	<Byte 2>	<Byte 3>
Packet Type	Padding		RVD Version
Drive			
Nonce			
Index			
Checksum			
Reserved			
Number of Blocks on Drive			
Burst		Queue Length	
Padding			Blocking Factor

(See the section on RVD Packet Format for a complete description of the fields in the standard RVD header: Packet Type, RVD Version, Drive, Nonce, Index, and Checksum.)

4.2.3.1. Field Definitions for Packet Type: SPIN-ACK

- Index

  The client must save the server's index for use in all future transmissions.

- Number of Blocks

  The server returns the size of the drive in 512 byte blocks. The client should not send any read or write requests for blocks outside of the drive boundaries. If the client does attempt an out of bounds request the server will inform him using the status word in the BLOCK or WRITE-ACK reply packet. The ERROR packet is primarily used for protocol errors.

- Burst

  This is a 2-byte integer that represents the maximum number of packets the server will handle in a single transmission. This value and the blocking factor value are then used by the client when partitioning read and write requests.

- Queue Length

  This is the maximum number of outstanding requests the server will handle for a virtual drive at any one time. This value is different than burst size. The client can send multiple transmissions of burst size packets until the number of packets equals queue length. The client has then saturated the server for this drive and must wait for the server's response.

- Blocking Factor

  This is the read blocking factor, the maximum number of blocks the client can read at one time in a single packet. The value of this field is not necessarily the same value transmitted by the client in the SPIN_UP packet. If the transmitted blocking factor is greater than the maximum number of blocks the server can send, it will be modified in the SPIN-ACK response packet.

4.2.3.2. Operation

The server sends a SPIN-ACK packet in response to a valid SPIN-UP packet. This indicates to the client that the connection request for the server's virtual drive to the client's local drive was successful.

In addition to returning the size of the drive and other connection details, the server provides an index value. Although it is recommended that the client use this index value in all future requests, the server can operate with incorrect Index values. If the server receives a bad index, it will send an ERROR packet of BAD-INDEX type to the user, but the operation will still occur correctly. The most likely cause of a bad index would be the client crashing then attempting to reuse the connection. In all probability the index would be lost, but the BAD-INDEX packet would correct that.

An example helps in explaining the difference between burst size, queue length, and blocking factor. Suppose the client must read forty blocks and that the SPIN-ACK response has reported a blocking factor of two, a burst size of five, and a queue length of ten. The blocking factor limits each read request to two blocks. Thus the client must transmit two sets of five read request packets for the first twenty blocks, wait for the server to respond with the data, then transmit the next two sets of five requests.

4.2.4. Packet Format: SPIN-DOWN

Format of SPIN-DOWN Packet

<Byte 0>	<Byte 1>	<Byte 2>	<Byte 3>
Packet Type	Padding		RVD Version
Drive			
Nonce			
Index			
Checksum			
Reserved			
Capability			

(See the section on RVD Packet Format for a complete description of the fields in the standard RVD header: Packet Type, RVD Version, Drive, Nonce, Index, and Checksum.)

4.2.4.1. Field Definitions for Packet Type: SPIN-DOWN

- Index

  Should be the index returned by the server for this drive in the SPIN-ACK packet.

4.2.4.2. Operation

If a user wishes to spin down the virtual disk in his local drive 3, then he simply sends a SPIN-DOWN packet to the server. He fills in Packet Type with SPIN-DOWN, and he fills in drive with the 32 bit integer 3. Upon receipt of the SPIN-DOWN packet, the server would attempt to terminate the connection between the client, local drive 3, and the virtual drive. If this worked correctly, the server sends a SPIN-DOWN-ACK back to the client. If the drive was not spun up, or there were some other error, then the server replies with an ERROR packet. It is not polite for a user to consider his disk spun down until he receives the SPIN-DOWN-ACK from the server.

4.2.5. Packet Format: SPIN-DOWN-ACK

Format of SPIN-DOWN-ACK Packet

<Byte 0>	<Byte 1>	<Byte 2>	<Byte 3>
Packet Type	Padding		RVD Version
Drive			
Nonce			
Index			
Checksum			
Reserved			

(See the section on RVD Packet Format for a complete description of the fields in the standard RVD header: Packet Type, RVD Version, Drive, Nonce, Index, and Checksum.)

4.2.5.1. Field Definitions Packet Type: SPIN-DOWN-ACK

4.2.5.2. Operation

This is the success acknowledgment to a client's spin-down request. Drive, nonce, and index have the same values as those specified by the client.

4.2.6. Packet Format: READ

Format of READ Packet

< Byte 0 >	< Byte 1 >	< Byte 2 >	< Byte 3 >
Packet Type	Padding		RVD Version
Drive			
Nonce			
Index			
Checksum			
Reserved			
Starting Block Number			
Block Count			

(See the section on RVD Packet Format for a complete description of the fields in the standard RVD header: Packet Type, RVD Version, Drive, Nonce, Index, and Checksum.)

4.2.6.1. Field Definitions Packet Type: READ

- Starting Block

  The number of the first block that the client wishes to read. This is an absolute offset from the beginning of the virtual drive. This number is a 32 bit integer with a range of zero to (2**32)-1.

- Block Count

  The number of blocks that the client wishes to read.

4.2.6.2. Operation

To read data from a local drive, the client sends a READ packet with the appropriate values. (This drive must have been associated with a virtual disk through a previous spin-up request.) The Drive field is filled with the drive number; the nonce, index, and *checksum* fields are initialized appropriately; and the desired block offset and number of blocks are written into the packet. Upon receiving the packet, the server looks up the connection between the client's local drive and the server's virtual drive.

If the connection exists, the server then tries to send the requested data to the client. If the read request is within the bounds of the disk and the physical read is successful, the server responds with a BLOCK packet. A BLOCK packet contains a block of data, a block number, and a 32 bit status word.

If the server detects an error, it still sends a BLOCK packet, although it has a nonzero status word. Errors can be the result of physical errors on the disk or any of the assorted things that can go wrong on a disk. If the data in the packet is valid, then the invalid-data field in the status word must be zero. The count field in the status word is set to the number of times the server attempted to read the block *before* it was successful. (I.e., if the first try succeeded, the count is zero, if the second try succeeded, count is one, and so on.) If the count exceeds the size of the field, then count is set to the maximum in the field. If the Start Block was invalid, then the bad block address field in the status word will be set. If a multiple block read extends beyond the drive boundaries, then only the in-bounds disk blocks will be returned and the bad-block address field in the status word will be set.

The only time the server sends an ERROR packet in response to a READ is in the case of a malformed READ packet. Protocol errors cause ERROR packets and disk errors cause non-zero status word. Typically ERROR packets will be sent for invalid *checksums*, bad index, or a zero in the Block Count field.

READs can timeout. This protocol does not guarantee delivery of packets. It is assumed that most packets will reach their intended destination, but there are no guarantees. It is up to clients to handle READ timeouts the same way they would treat physical disk timeouts.

4.2.7. Packet Format: BLOCK

Format of BLOCK Packet

< Byte 0 >	< Byte 1 >	< Byte 2 >	< Byte 3 >
Packet Type	Padding		RVD Version
Drive			
Nonce			
Index			
Checksum			
Reserved			
Block Number			
Drive Status			
Data			

(See the section on RVD Packet Format for a complete description of the fields in the standard RVD header: Packet Type, RVD Version, Drive, Nonce, Index, and Checksum.)

4.2.7.1. Field Definitions Packet Type: BLOCK

- Block Number

  This identifies this block in the virtual disk.

- Drive Status

  Drive Status is a 32-bit status word describing the status of the virtual disk. Disk errors are reported through this word. Protocol errors are reported with an ERROR packet.

- Data

  This is data read from the virtual disk. There are (512 x blocking_factor) bytes of data in this field. The *checksum* guarantees reliable data transmission. (The protocol cannot guarantee the accuracy of the disk to protocol data transmission.)

4.2.7.2. Operation

The BLOCK packet is the response to the READ request. It includes the data requested and enough information to allow the client to determine which request this is in response to. (Note that there can be many outstanding READ requests, even in the case where a drive's access is restricted to a single outstanding request. A client can always have requests out to more than one drive.)

In case of an error, the server fills in the appropriate bits in the status. The client must check the data-valid field in the status as the data may be valid even in the case of a non-zero status.

4.2.8. Packet Format: WRITE

Format of WRITE Packet

< Byte 0 >	< Byte 1 >	< Byte 2 >	< Byte 3 >
Packet Type	Padding		RVD Version
Drive			
Nonce			
Index			
Checksum			
Reserved			
Block Number			
Total Blocks in Request			
Index of this Block in Request			
Data			

(See the section on RVD Packet Format for a complete description of the fields in the standard RVD header: Packet Type, RVD Version, Drive, Nonce, Index, and Checksum.)

4.2.8.1. Field Definitions for Packet Type: WRITE

- Block Number

  This is the starting block number for the write. It is an integer that represents the absolute block offset from the beginning of the virtual drive for this data block.

- Total Blocks

  This is the total number of blocks in a sequence of write requests.

- Block Index

  This is the block number of this request.

- Data

  This is the data that is to be written to the virtual drive starting at the specified block number. The data bytes are ordered in the packet exactly the way they are ordered on disk. The bits in a byte are ordered in accordance with the IP specification. The size of the data field is determined by subtracting the size of the header fields from the total size of the packet. The packet size is given by the IP protocol.

4.2.8.2. Operation

The WRITE packet type has been designed to be sent as a burst of packets. If a client wishes to write data to the virtual disk, it creates a sequence of packets, each with the same Total Blocks value but incrementing the Block Number and Block Index. The data is then copied to the data fields of these sequential packets in 1024/512 byte chunks.

Upon receipt of this burst of WRITE packets the server orders the packets, copies the data to a single contiguous buffer, and does the write as one operation. The server sends a WRITEACK if the write was a success, but only for the first packet in the burst.

The WRITEACK includes a 32-bit status word. In the case of a disk error, a WRITEACK with a non-zero status word will be returned. Generally these will be the same type of errors as on a READ request. Additionally, if you try to write to a READ-ONLY disk, then the no-write-permission field is set in the status word. Protocol errors will cause ERROR packets to be sent.

Writes can timeout. Again, clients are expected to deal with a WRITE timeout in the same way in which they would deal with a disk timeout.

4.2.9. Packet Format: WRITEACK

Format of WRITEACK Packet

< Byte 0 >	< Byte 1 >	< Byte 2 >	< Byte 3 >
Packet Type	Padding		RVD Version
Drive			
Nonce			
Index			
Checksum			
Reserved			
Block Number			
Drive Status			
Number of Blocks for this ACK			

(See the section on RVD Packet Format for a complete description of the fields in the standard RVD header: Packet Type, RVD Version, Drive, Nonce, Index, and Checksum.)

4.2.9.1. Field Definitions for Packet Type: WRITEACK

- Block Number

  The number of the block that the server is acknowledging having written.

- Status

  The 32-bit Status word that represents the state of the disk and reports errors in the write procedure.

- Number of Blocks

  The number of blocks that have been successfully written.

4.2.9.2. Operation

WRITEACK is the response to the WRITE request. It signals the completion of the WRITE request. The WRITEACK is not sent until after the burst of write requests has been written to the physical disk. Only one WRITEACK is sent per burst.

Disk errors are reported through the status word.

4.2.10. Packet Format: ERROR

Format of ERROR Packet

< Byte 0 >	< Byte 1 >	< Byte 2 >	< Byte 3 >
Packet Type	Error Type	Padding	RVD Version
Drive			
Nonce			
Index			
Checksum			
Reserved			
Up to RVDDSIZE (512) Bytes of Error Dependent Data			

(See the section on RVD Packet Format for a complete description of the fields in the standard RVD header: Packet Type, RVD Version, Drive, Nonce, Index, and Checksum.)

4.2.10.1. Field Definitions for Packet Type: ERROR

- Error Type

  A byte representing the type of error that the ERROR packet is reporting.

4.2.10.2. Operation

ERROR packets are sent out when the server wants to tell the user that some error has occurred. They are usually sent when the error was in the protocol, or some other high level thing wrong with the virtual disk system. Errors that are roughly equivalent to those that a physical disk would give are typically returned in the 32 bit status word that is part of BLOCK and WRITEACK.

4.2.11. RVD Protocol Constants

```
RVDVERSION      4       /* Current protocol version */


/*
 * IP protocol number
 */
RVDPROTO        66

/* Packet types */
RVDSPIN         1       /* SPIN-UP packet */
RVDSDOWN        2       /* SPIN-DOWN packet */
RVDREAD         3       /* READ packet */
RVDWRITE        4       /* WRITE packet */
RVDSPACK        17      /* Ack for SPIN-UP */
RVDERROR        18      /* ERROR packet */
RVDACK          19      /* Ack for SPIN-DOWN */
RVDBLOCK        20      /* Block of data */
RVDWACK         21      /* Ack for WRITE */


/*
 * Status word masks
 */
RVDSTVAL        0001    /* If 0 then valid data */
RVDSTCNT        0036    /* Count of tries on foreign end - 1 */
RVDSTADR        0040    /* Bad block address */
RVDSTWRL        0100    /* Write attempted on read-only disk */


/*
 * Opening modes
 */
RVDMRO          0001    /* Read-only mode */
RVDMSHR         0002    /* Shared mode */
RVDMEXC         0004    /* Exclusive mode */


/*
 * Error types
 */
RVDENOER        0000    /* No error */
RVDEND          0001    /* Non-existent drive */
RVDEBPWD        0002    /* Bad password for mode */
RVDEOMD         0003    /* Already open in a different mode */
RVDECKSM        0004    /* Invalid Checksum */
RVDEIDX         0005    /* Index correction */
RVDEPACK        0006    /* Non-existent disk-pack */
RVDESPN         0007    /* Drive already spun up */
RVDEBMD         0010    /* Bad mode */
RVDEPKT         0011    /* Unknown packet type */
RVDENAH         0012    /* Non Active Host */
```

```
RVDEXMD        0013   /* Pack was spun up in EXCLUSIVE mode */
RVDEZBL        0014   /* Zero blocks requested */
RVDETBL        0015   /* Too many blocks requested */
RVDEPNM        0016   /* Pack not physically mounted */
RVDETCG        0017   /* Too many connections for this server */
RVDETGH        0020   /* Too many connections for this host */
RVDESNA        0021   /* Server not currently active */
RVDEIDA        0022   /* Identical pack already spun up in this
                         drive, in the requested mode */
RVDERQU        0023   /* Requested mode unavailable. */
RVDETIM        0064   /* Timeout */
RVDEBVER       0065   /* Invalid version */
```

5. RVD CONTROL PROTOCOL SPECIFICATION

This section is optional reading.

5.1. Overview

The Remote Virtual Disk Control (RVDCTL) protocol supplies a Remote Virtual Disk server with both operating instructions and information about its configuration. An RVD server process comes into existence with no knowledge of the physical configuration of the system in which it is embedded or the logical configuration of the virtual disk packs it is to manage. These virtual disk packs may already exist. Supplying this information via a network connection instead of from files on the server host makes it possible to administer all aspects of server operation remotely.

This description assumes that the reader is already familiar with the basic concepts and terminology of the Remote Virtual Disk system, as described in Chapter 1 of this article.

* Operations and operands marked with an asterisk to the left have not yet been implemented.

There are four scenarios in which the RVDCTL protocol is used:

(1) Initialization

The first step after creating an RVD server process is to send it, using the RVDCTL protocol, a description of the physical and virtual disk configuration it is to manage. Because RVDCTL is a network protocol, the permanent data base that contains this state description may be managed on a machine different from that of the server.

(2) Permanent changes

When permanent changes to the physical and virtual disk pack configuration are desired, a management program both updates the permanent data base and sends to the server the same updates, again using the RVDCTL protocol.

(3) Temporary changes, for maintenance.

A client that can supply an operations password can invoke certain maintenance functions of the RVDCTL protocol, such as changing the logging level, shutting down the server, posting a message, or forcing off certain clients. For operations purposes, it is possible to invoke any of the update functions normally associated with permanent changes. Although a running RVD server would operate on the updated basis, if that server process were killed and recreated, such temporary changes would be forgotten, because the new server would receive its initialization from the permanent data base.

(4) Client use

RVD clients use the RVDCTL protocol for certain client-server interactions, such as flushing out old spinups, and inquiring about current operations.

There are several programs that invoke the RVDCTL protocol. Corresponding to the second scenario, above, is a data base management program (*vddb*(8)), which operates as follows:

- The person in charge of maintaining the data base runs the data base management program, which prompts for and validates input.
- The data base management program updates the disk files containing the permanent data base.
- The data base management program opens a control connection to the currently-running server and sends the server the updated information.

- The server modifies its state tables according to the requests.
- The server acknowledges the modification.

For simplicity, the data base management program stores its permanent data base in the form of a sequence of already-formatted protocol messages, so the initialization scenario is accomplished simply by sending a copy of the permanent data base to the server on the RVDCTL network connection. The *rvdsend*(8) program does this job.

The programs that perform the third and fourth scenarios are commands that can be run on any client (perhaps on behalf of another client) to invoke one or more specific RVDCTL functions at the server directly, without involving the database program.

Because most control operations for the RVD server require transferring only small amounts of data, and one wants to be able to implement servers on machines that do not provide a full TCP, the control protocol is UDP-based. It is a simple, lock-step, idempotent, message-response protocol. For all the control functions of interest, the control data fit into a single packet, which further simplifies the protocol. Idempotent means that if a client receives no response to a request (and is therefore unsure of whether or not the server acted on it), it is always safe to resend the same request, because by design successive repetitions of all RVDCTL operations have no ill effects.

The RVDCTL protocol carries very little traffic in comparison with the RVD protocol, so ease of construction and debugging therefore has a higher priority than performance. So that the control connection can handle operations of varying parameter requirements (to avoid the need to design a new packet format every time a new control function is added), and so that a single source implementation can apply to machines of different byte order, all data is transmitted as ASCII strings. For simplicity in parsing, arguments are transmitted in the format "keyword=value." This approach also makes RVDCTL packets self-explanatory when encountered during monitoring or auditing.

5.2. Syntax of the RVDCTL Protocol

```
<message>    :==   <opcode>  <operands>
<opcode>     :==   operation= <value> |
                   success= <value> |
                   failure= <value> error= <value>
<operands>   :==   <operand> | <operands> <operand>
<operand>    :==   <keyword> = <value>
<keyword>    :==   <string>
<value>      :==   <string>
```

<*string*> is a string of network ASCII characters, terminated by a separator character. The separator characters are space, tab, newline, carriage return, and formfeed. One or more separators must appear between operands. Separators may be included in strings by quoting them. The quote character is the backslash (\). A backslash may be included in an string by doubling it (\). Also, to include an equals sign (=) in a string, it must be quoted.

The particular keywords used depends on the operation being invoked. The keywords "operation," "password," "nonce," "success," "failure," and "error" are universal.

(1) The "operation" keyword must be the first keyword in each request packet. Its value is the name of the requested operation.

(2) The "password" keyword supplies a password operand if the operation requires one. There are three kinds of passwords. The operations password authorizes shutdown,

logging, and physical configuration management. The administrative password authorizes allocation and deallocation of virtual disks. Individual pack passwords authorize usage of those packs.

(3) The "nonce" keyword appears in every request, with a value chosen by the requester to be different from any other request for which a late response might still arrive. Every response contains a copy of the nonce of the request to which it responds.

(4) The "success" keyword must be the first keyword in a response to a successful request. Its value is the name of the operation performed.

(5) The "failure" keyword must be the first keyword in a response to an unsuccessful request. Its value is the name of the operation that failed. It may be accompanied by an "error=" operand describing the error which occurred. The value of the "error" keyword is a human-readable string describing the error which occurred.

Except for "operation," "success,"and "failure,"each be the first keyword in a message, the order of operands in a message is unimportant.

Where a number is called for, it is represented in the operand value string as an ASCII decimal integer. Where an Internet Protocol (IP) Address is called for, it is represented in the operand value string as .q "A.B.C.D" in network standard ASCII decimal form. Where a mode is called for, it is represented in the operand value string as an ASCII decimal number coded in the following way, in any sum desired:

1 = read-only spinups allowed
2 = shared spinups allowed (not currently implemented)
4 = exclusive spinups allowed
0 = no spinups allowed

Port: The RVDCTL protocol operates on UDP port 531.

5.3. Operations

(1) Add a physical device partition to the set of partitions managed by the RVD server.

operation = add_physical

Required operands:

password	The operations password for the RVD server.
filename	Path name of the device to be managed as a physical partition.
blocks	The number of 512-byte sectors in this physical partition.

The device need not be a real physical disk; any device (e.g., a file) that behaves like a raw disk partition will work equally well.

* If the server finds it is unable to open the physical device it marks the physical device as "disused" and returns an error. (See *disuse_physical.*)

Note that *add_physical* is normally invoked as part of updating the permanent data base that describes the server configuration. If *add_physical* is invoked without a data base update, the next time the server is shut down the change made by the add_physical operation will be forgotten.

(2) Delete a physical device partition from the set of partitions managed by the RVD server.

operation = delete_physical

Required operands:

password	The operations password for the RVD server.
filename	Path name of the device to be managed as a physical partition.

If there are any virtual disk packs allocated on this physical device, delete_physical returns an error response, and does not delete the device.

Note that the delete_physical operation is normally invoked as part of updating the permanent data base that describes the server configuration. If delete_physical is invoked without a data base update, the next time the server is shut down the change made by the delete_physical operation will be forgotten.

(3) Stop using a physical disk partition.

operation = disuse_physical

Required operands:

password	The operations password for this RVD server.
physical	The pathname of the device partition to be disused.

The *disuse_physical* operation allows an operator to take a partition out of use, for example because the disk is getting hard errors. (The server may, on its own, place a partition that is getting errors in disused mode.) *Add_virtual* and *delete_virtual* operations may be executed on a disused partition. Attempts to spinup packs that are located on a disused partition receive the error response "pack temporarily unavailable." The server continues to maintain records of existing connections and to allow spindowns, but attempts to read or write a previously spunup pack receive an error packet containing the error code "pack temporarily unavailable."

(4) Try to use a physical disk partition.

operation = use_physical

Required operands:

password	The operations password for this RVD server.
physical	The pathname of the device partition to be used.

If a physical partition is currently disused, this operation puts the partition back into service. If the physical partition does not exist or is already in use, *use_physical* returns an error.

(5) Allocate a virtual disk pack.

operation = add_virtual

Required operands:

password	The administrative password for this RVD server.
physical	The pathname of the device partition this virtual disk pack is to be on.
name	The name of this virtual disk pack (n.b., upper and lower case are distinguished.)
packid	The unique id of this pack on this server.
owner	The name of this virtual disk pack's owner.
rocap	The read-only mode password (may be null).
excap	The exclusive mode password (may be null).
shcap	The shared mode password (may be null).
modes	The allowable modes this virtual disk pack may be spun up in.
offset	The offset, in blocks, of this virtual disk pack from the start of the physical partition.
blocks	The number of 512-byte blocks in this virtual disk.

Optional operands:

ownhost	Internet address of the owning host of this virtual disk pack. If none is supplied, the disk is assumed to not have an owning host.

Add_virtual is normally invoked as part of updating the permanent data base that describes the server configuration. If *add_virtual* is invoked without a data base update, the next time the server is shut down the addition made by *add_virtual* operation will be forgotten.

(6) Deallocate a virtual disk pack

operation = delete_virtual

Required operands:

password	Administrative password.

Optional operands:

packid The unique identifier of the virtual disk pack to be deallocated.

name The name of the virtual disk pack to be deallocated.

One of the operands {packid, name} must be present. If both are present, they must refer to the same pack.

Delete_virtual is normally invoked as part of updating the permanent data base that describes the server configuration. If *delete_virtual* is invoked without a data base update, the next time the server is shut down the deletion made by *delete_virtual* operation will be forgotten.

(7) Modify the definition of a virtual disk pack.

operation = modify_virtual

Required operands:

password Administrative password.

name The name of the virtual disk pack whose description is to be modified.

Optional operands (any operand present supersedes the value previously supplied by *add_virtual* or *modify_virtual* of the corresponding parameter for this virtual disk pack):

packid The unique identifier of this pack. If provided, this operand is used to identify the pack to be modified, and the name operand is taken to be a new pack name.

owner The name of this virtual disk pack's owner.

rocap The read-only mode password.

excap The exclusive mode password.

shcap The shared mode password.

modes The allowable modes this virtual disk pack may be spun up in, as an ASCII decimal number.

blocks The number of 512-byte blocks in this virtual disk. Must be less than or equal to the current number of blocks on this disk. In general, changing a disk's size is a bad idea, especially if it is currently in use.

ownhost Internet address of the owning host of this disk.

Modify_virtual is normally invoked as part of updating the permanent data base that describes the server configuration. If *modify_virtual* is invoked without a data base update, the next time the server is shut down the changes made by the *modify_virtual* operation will be forgotten.

(8) Exchange the names of two virtual disk packs.

operation = exchange_names

Required operands:

name1	Desired name for the first virtual disk pack
packid1	Unique identifier of first pack
password1	The exclusive mode password of the first virtual disk pack
name2	Desired name for the second virtual disk pack
packid2	Unique identifier of second pack
password2	The exclusive mode password of the second virtual disk pack

The operands *name1* and *name2* must be the names presently associated with the two packs. Success for this operation means that those two names are now associated with the packs in the order requested, whether or not they were before the operation.

This operation is used as part of an update procedure, in which two copies of a library virtual disk pack are maintained. One copy is normally spun up by clients in read-only mode; the other is the "maintenance" copy, to which the owner makes changes. Once a consistent set of changes are ready for release, the owner exchanges the names of the packs. Other users can then spin the pack down and back up again by name to get the new copy. If the server shuts down and restarts, clients that have temporarily cached the packid can respin up the old pack by packid, to complete their session without being forced prematurely to switch to the new library.

Exchange_names is normally invoked as part of updating the permanent data base that describes the server configuration. If *exchange_names* is invoked without a data base update, the next time the server is shut down *exchange_names* will be forgotten.

(9) Force a virtual disk pack to be spun down.

operation = spindown_virtual

Required operands:

name	The name of the virtual disk pack to be forced down.
password	The exclusive mode password of the virtual disk pack to be forced down.

This operation is normally used by the owner of a virtual disk pack that was spun up on a machine that crashed. It forces the specified virtual disk pack to be spun down from all the machines that have it spun up.

(10) Force all virtual disk packs of a given client at this server to be spun down.

operation = spindown_host

Required operands:

name — The Internet address of the client whose disk packs are to be forced down.

Optional operands:

password — If the spindown_host request was not sent from the client whose disks are to be spun down, the operations password must be supplied.

This operation has two uses:

a) It should appear in a IBM/4.3 client's /etc/rc file, or a DOS client's autoexec.bat file, so that when a host recovers from a crash, all its previously spunup virtual disk packs are spun down. This spindown insures that the server state agrees with the client state.

b) An operator can use this operation to force down the virtual disks of a client which has crashed and that may be down for some time.

(11) Display all spinups involving a virtual disk pack, or a client.

operation = display_virtual

Required operands (exactly one of the following must be present):

name — The name of a virtual disk pack. If present, *display_virtual* returns a list of all the spinups of this disk pack. These are the spinups that would be forced down if a *spindown_virtual* operation naming this pack were performed.

host — The IP address of a client. If present, *display_virtual* returns a list of all the spinups of this client. These are the spinups that would be forced down if a *spindown_host* operation naming this client were performed.

Optional operand:

start = <value>   An ASCII decimal integer giving the offset of the first spinup description wanted. This operand is normally supplied if a previous invocation of display_virtual contained the response "more = true."

password	If the display_virtual operation requests information about a client different from the one making the request, the operations password must be supplied.

This operation returns a success packet containing an ASCII text string describing the spinups (host/drive number pairs) of this virtual disk. The response packet contains:

```
success = display_virtual
number = <value1>
connections = <value2>
more = true                 (optional response)
```

<value1>
The number of currently active spinups for this virtual disk pack or client.

<value2>
A canonicalized string, with one line per spinup, containing as many spinup descriptions as will fit in one UDP packet. Each line is a collection of space-separated tokens, as follows:

pack = library host = 18.72.0.5 drive = 9 mode = 4

Since the string is canonicalized, all spaces and CRLF sequences are quoted.

If there were more spinup descriptions than would fit in a single packet, the response operand "more = true" will appear.

(1) Log statistics of external interactions.

operation = log_external_statistics

Required operand:

password	The operations password

Dump into the log file all statistics kept by the RVD server concerning interactions with clients--number of packets exchanged, disk operations, etc.)

(2) Log all statistics

operation = log_all_statistics

Required operand:

password	The operations password

Dump into the log file all statistics kept by the rvd server.

(3) Shut down server

operation = shutdown

Required operands:

password	The operations password

Log all statistics, then perform a clean shutdown of the server.

(4) Change log level

operation = log_level

Required operands:

password	The operations password
level	New log level as a hex number (N.B., not decimal.)

Change which events are logged; see specification of the RVD protocol for definition of log levels.

(5) Truncate log

operation = log_truncate

Required operands:

password	The operations password

Truncate the log file to keep it from growing too large. [In the BSD 4.3 UNIX implementation of RVD, logging is done with the UNIX logging system (syslogd), so this operation has no effect.]

(6) Allow spinups

operation = allow_spinups

Required operands:

password	The operations password, for a physical device, or the exclusive mode pack password, for a single virtual pack.
mode	The mode of allowed spinups.

Optional operands:

physical	Path name of the device partition to which this mode setting applies. (If absent, the mode applies to all partitions managed by this server.)
name	The name of a virtual disk pack to which this mode setting applies.

Response operand:

oldmode The spinup mode that was formerly allowed for this partition or virtual pack.

This operation is used to prevent or allow further spinups of a single virtual pack, or all the virtual packs on a given device partition of this RVD server; it has no effect on spinups already in force. When a server first comes up it allows no spinups (mode = 0), so an invocation of *allow_spinups* is required as part of starting a server.

A separate allowed spinup mode value is maintained for each pack and for each partition; the actual modes permitted for a pack are given by the logical AND of the mode value for the pack and the mode value for the partition on which it is located.

The server rejects spinups that would be allowed by the static pack description but that are prevented by the current setting of *allow_spinups* with a distinct error code indicating temporary unavailability.

Usage scenarios: If a server is to be dumped, one might allow only read spinups during the dump; if a server is to be taken down one might sometime earlier allow no new spinups. The maintainer of a library disk pack that needs to be updated might first allow no spinups, then after a period of time adequate for most clients to finish their sessions, do a *spindown_virtual* to get rid of any remaining spinups.

(7) Post an operations message.

operation = set_message

Required operands

password The operations password

message = < string > The (canonicalized) message < string > replaces any previous operations message. If < string > is null, any previous message is cleared. The content of the message is limited to 400 bytes, and is network ASCII (lines terminated with canonicalized CRLF's).

This operation, together with the next one, allows an operator to post a message (e.g., "server going down at 5:00 p.m. for preventive maintenance") for clients of an RVD server.

(8) Get the operations message.

operation = get_message (no required or optional operands)

Response operands:

success = get_message

message = < string > < string > is a canonicalized string of network ASCII to be displayed as an operations message. If there is no current operations message, < string > is null. (Note that in either case < string > is terminated by an operand separator.)

This operation would normally be invoked by a client as part of bringing up a system that uses RVD and also whenever spinning up a virtual disk pack.

*

(9) Change a user password.

operation = change_password

Required operands

packname — The name of the virtual disk pack whose password is to be changed.

mode — The spinup modes for which a new password is being supplied. If more than one mode is specified, the operation will be rejected unless the old passwords for the several modes are all the same as the as the old_password operand.

old_password — The current password for this pack and mode; a null string if there is no current password.

new_password — The new password; a null string if there is to be np password.

Note that this function is not intended for direct use by a client, but rather for use by the database update system; if used by a client without also updating the database, the password will be restored to its old value the next time the RVD server is restarted.

(10) Return a list of active virtual packs

operation = display_active

Optional operands:

filename — Path name of device partition for which a list of active virtual packs is wanted. If omitted, a list of all active virtual packs is returned.

start = < value > — A number giving the offset of the first information line wanted. This operand is normally supplied if the previous invocation of display_active included the response operand "more = true."

Response operand:

number = < value1 >

activity = < value2 >
more = true (optional response)

< value1 >

The number of currently active packs on this partition or, if no partition was specified, on this server.

< value2 >

A single canonicalized netascii string containing one line of information for each active virtual pack. A typical line looks like:

partition = /dev/ra0g pack = library mode = 1 connections = 5 idle = 1721

If there were more activity descriptions than would fit in a single packet, the response operand "more = true" will appear.

Idle time is measured in seconds since most recent access. Note that the idle time is purely an activity hint, to determine whether or not a pack that appears to be spun up is actively in use. It is maintained by the server only to a rough approximation.

*

(11) Obtain server load statistics

operation = get_load

Required operands:

password	The operations password for the RVD server.

Response:

load = < string >	< string > is a canonicalized netascii string containing load statistics ready for display.

(12) Change authorization for operations and administrative operations.

operation = require_authorization (no required or optional operands)

When an RVD server begins operation, it accepts RVD control protocol requests only from the same host on which it is operating, and it does not require operations or administrative passwords. (Starting without passwords allows automating initialization without the need to store those passwords in clear form.) The *require_authorization* operation causes the server to read operations and administrative passwords from a file in the file system of the server's host. After *require_authorization* is executed all operations listed above as requiring either an administrative or operations password do actually require them. Whenever *require_authorization* is invoked, the RVD server reinitializes its copy of the operations and administrative passwords from */etc/rvdauthor.*

There are two scenarios of use of require_authorization. The first is at system initialization time:

- start server
- send initializing control sequences, if any
- send require_authorization
- await success of require_authorization
- declare initialization successful.

The second scenario is to change the operations or administrative passwords.

- modify file containing operations and maintenance passwords.
- send require_authorization

DMA Reference Manual

ABSTRACT

This paper describes the set of kernel utility routines provided with IBM/4.3 to facilitate the use of the RT's eight Direct Memory Access (DMA) channels by device driver writers. The article contains the following sections:

1. **Introduction** describes the purpose of the DMA utility routines.
2. **The Hardware** provides a brief overview of how the DMA hardware works.
3. **The Software** describes the structures, flags, and callable routines provided.
4. **Using DMA in a Device Driver** describes the normal flow of control between device drivers and the DMA code.

July 1987

1. Introduction

IBM/4.3 includes a set of utility routines designed to facilitate how device drivers use the IBM RT PC's eight DMA channels. This article describes the interface the writer of a device driver uses, and the way the DMA hardware works. For a more complete description of the DMA hardware, see the *IBM RT PC Hardware Technical Reference Volume I*, Number 75X0232.

2. The Hardware

The DMA hardware on the RT PC is influenced by the DMA implementations on the PC/AT, PC/XT, and PC. Transfers are mapped from the PC address that the various adapters use to an RT memory address (either physical or virtual) by a set of registers called Translation Control Words (TCWs). The number, location, and format of the TCWs usable by a given device are dependent upon which of several modes the DMA will operate in.

2.1. System DMA vs. Alternate DMA

DMA transfers can be assisted by hardware on the RT PC System Board, or controlled by a DMA controller on an adapter card. The former is known as system DMA (as well as third party DMA, or non-cascade mode). The latter is known as alternate DMA (as well as first party DMA, or cascade mode). The mode a driver can use is constrained by the hardware for which the driver was written. Adapters without DMA controllers on the card must use system DMA. Adapters with DMA controllers must use alternate DMA. The DMA utility routines will set up system DMA as well as the required TCWs. The device driver writer must set up alternate DMA.

2.2. 16 bit vs. 8 bit

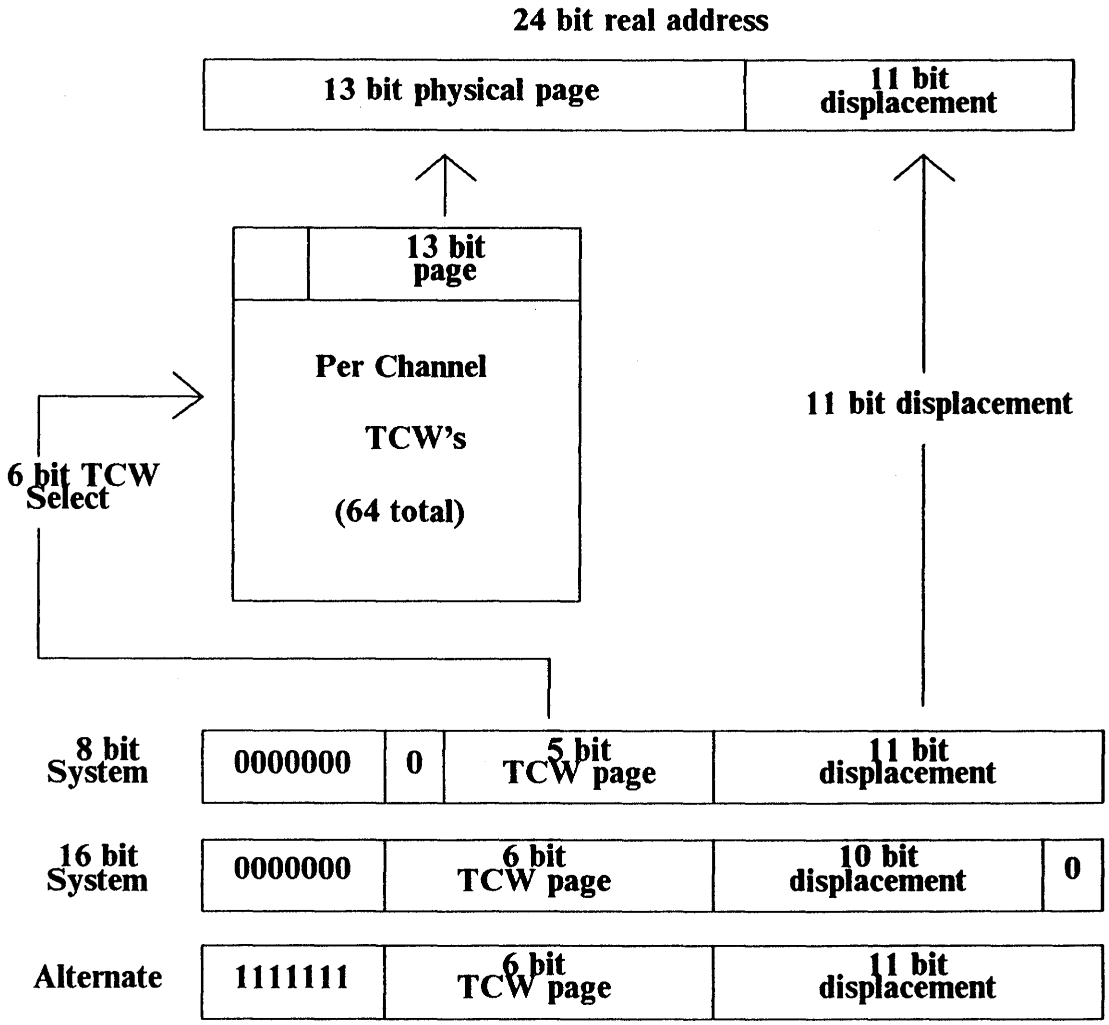
When using system DMA, DMA channels 0 to 3 are connected to an eight bit DMA controller, while channels 5 to 7 are connected to a 16 bit DMA controller. The eight bit controller transfers data one byte at a time. Addresses can be aligned on any arbitrary byte boundary, and counts can be either even or odd. The maximum transfer length is 64K, due to addressing limitations of the DMA controller.

The 16 bit controller transfers data two bytes at a time. Addresses must be half-word aligned, and counts must be even. The DMA code will truncate odd addresses and counts. The maximum transfer length is 128K.

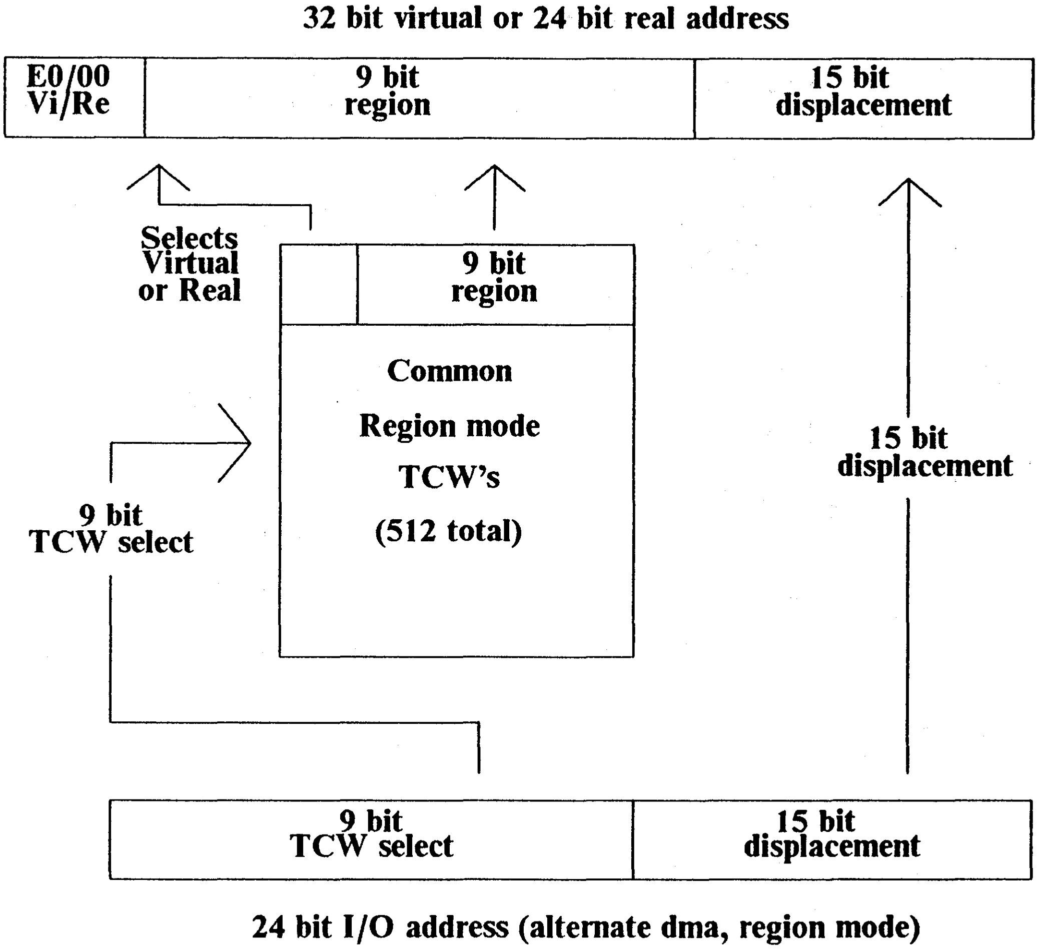
Note that this distinction is only applicable when using system DMA. Alternate DMA treats all these channels the same.

2.3. Page Mode vs. Region Mode

The TCWs used and the meaning of each TCW are controlled by the DMA mode register. This register can select either page mode or region mode. In page mode, the I/O address specified by the device doing the DMA (or in the case of system DMA, by the DMA controller) is separated into two pieces: a TCW page select (six bits) and a page displacement (11 bits). The TCW page select is used to address one of 64 TCWs belonging to the DMA channel which generated the request. The TCW contains a 13 bit physical page address which is added to the 11 bit displacement to produce a 24 bit real address. This corresponds to the 2K virtual page size used by IBM/4.3. (This means one TCW is needed for each page of virtual memory to which the driver transfers.) The DMA mapping routines automatically set up these TCWs from valid paged-in virtual buffers. In page mode, the maximum DMA transfer is 128K. Alternate controllers can generate 24 bit I/O addresses. For alternate DMA using page mode, the upper seven bits must be 1's.



Region mode also separates addresses into two pieces: a TCW region select (nine bits) and a region displacement (15 bits). The TCW region select points to one of 512 TCWs shared by all the DMA channels. These TCWs contain a nine bit region which is concatenated to the 15 bit displacement. In addition to the nine bit region, region mode TCWs also contain status information specifying if the transfer is to real or virtual memory. If the transfer is real, the nine bit region select plus 15 bit displacement becomes the 24 bit real address. If the transfer is virtual, the hex value E0 is prepended to the 24 bit address and sent to the MMU to be translated. Note that region mode DMA can potentially transfer 16 Meg of data in one transfer. However, the granularity of the translation is 32K contiguous. Since under normal circumstances, it is unlikely transfers will be to 32K contiguous real memory, most devices which use region mode would use region mode virtual.



System DMA devices must use page mode DMA only. Alternate controllers can use either; however, the DMA utility code only supports page mode DMA.

2.4. Special and Restricted Channels

As mentioned above, channels 0 to 3 are eight bit controller channels, and channels 5 to 7 are 16 bit controller channels. These distinctions apply only when using system DMA. Channel 8 is a special DMA channel designed for use with the 286 co-processor card. Since there is no system controller for channel 8, no system DMA can be used by it. Nor are there any page mode TCWs for channel 8; only region mode DMA can be used. Channel 4 is a reserved channel not available for use by DMA devices.

2.5. Selecting the Proper DMA Channels and Modes

The adapter hardware determines most of the modes that a DMA device can use. The channels that a device can use are usually fixed (though some adapters can be programmed to use some or all of the DMA channels). Whether the adapter can use system DMA or alternate DMA is also an attribute of the adapter. Writers of device drivers must determine what type of DMA the adapter can use.

3. The Software

3.1. Structures Used When Calling DMA Utility Routines

There are several fields reserved in various I/O structures for use by DMA. The fields listed in this section are those used by a device driver to send information to and receive it from the DMA code. The format of the list is as follows:

structure:include file:description

for the header, and

field|descriptions| [*valid parameters*]

for the fields.

struct iocc_ctlr: */sys/machineio/ioccvar.h*: per controller structure to *dma_setup()*

short ic_dmachannel | DMA channel number | DM_CHAN[0-3,5-8]
int ic_dmaflags | DMA transfer flags | see DMA flags below
struct buf *ic_dmabuf | buffer describing the DMA transfer |

struct dma_callback:*/sys/machineio/ioccvar.h*: callback structure used by the driver when calling *dma_wait()*

caddr_t d_info | value to be passed to the callback routine |
void (*d_wakeup)() | callback routine, called when the channel is no longer exclusive |

struct buf:*/sys/h/buf.h*:general buffer structure

long b_flags | flags describing the transfer. DMA is only interested in the B_PHYS flag (which must be set if the buffer points to an address in user space) and B_READ (which is set if the transfer is a read). |
caddr_t b_un.b_addr | address of the transfer |
long b_bcount | size of the transfer in bytes |

3.2. DMA flags passed with ic_dmaflags

The following flags can be passed using ic_dmaflags to tell the DMA code how to set up the hardware. Defines for these flags can be found in */sys/machineio/dmavar.h.*

Flag(s): DMA_DEMAND, DMA_SINGLE, DMA_BLOCK, DMA_CASCADE
Default: DMA_SINGLE (system DMA), DMA_CASCADE (alternate DMA)
Hardware: DMA controller mode register (see *IBM RT PC Hardware Technical Reference Volume I*)
Effect: Controls transfer mode characteristics. DEMAND is for asynchronous input such as from keyboards and serial lines; SINGLE is for cycle stealing mode; BLOCK is for burst mode; CASCADE is for alternate DMA (first party) mode.
Restrictions/Side Effects: DMA_CASCADE must be used for alternate DMA, and cannot be used for system DMA.

Flag(s): DMA_PAGE, DMA_REGION
Default: DMA_PAGE
Hardware: DMA mode register (see *IBM RT PC Hardware Technical Reference Volume I*; note that this is NOT the same register as above).
Effect: Controls whether page mode or region mode transfers are used (see "Page Mode vs. Region Mode" above).
Restrictions/Side Effects: System DMA can use only page mode. Region mode DMA is not supported.

Flag(s): DMA_PHYSICAL, DMA_VIRTUAL
Default: DMA_PHYSICAL
Hardware: TCW (see *IBM RT PC Hardware Technical Reference Volume I*).
Effect: Controls whether the translated address is real or virtual.
Restrictions/Side Effects: DMA_VIRTUAL is not valid for page mode transfers.

Flag(s): DMA_CANTINT
Default: off
Hardware: software only
Effect: Tells the DMA code that the device can't interrupt after the DMA has completed. The DMA code will call the device driver's interrupt routine whenever the DMA gets an interrupt. It is up to the device driver to determine whether or not the transfer has really completed. If the transfer has not completed, the device driver should return INT_NOT_MINE.
Restrictions/Side Effects: The DMA controller only interrupts on completion of system DMA transfers.

Flag(s): DMA_EXCLUSIVE
Default: off
Hardware: software only
Effect: This flag tells the DMA code that the device driver intends to use this channel indefinitely. Once an exclusive device has control of the channel, any further attempts to queue a request will result in the DMA code requesting the exclusive device driver to release the channel. If the exclusive device driver refuses, the request will be rejected with a DMA_EXCLUSIVE_RET. See Using DMA_EXCLUSIVE below for more details.
Restrictions/Side Effects: Only one DMA_EXCLUSIVE request can be queued at one time. If a DMA_EXCLUSIVE request is not yet running, other requests on that channel have priority.

Flag(s): DMA_CANTWAIT
Default: off
Hardware: software only
Effect: This flag tells the DMA code that the device driver can't wait for the channel to become not busy.
Restrictions/Side Effects: See "*dma_setup()*" below.

3.3. Driver Callable DMA Routines

There are several routines in the DMA code which are designed to be called by a device driver when it wishes to use DMA services.

int
dma_setup(ic)
struct iocc_ctlr *ic;

Dma_setup() is called whenever a device first wants to initiate a transfer. It is passed a pointer to a controller structure which has the channel number (ic_dmachannel), flags (ic_dmaflags), and a buffer describing the DMA transfer (ic_dmabuf). *Ic_dmabuf* can be set to 0 to get the channel only. If the request can be queued, *dma_setup()* will return DMA_OK_RET. If the request cannot be queued because there is an exclusive device, dma_setup will return DMA_EXCLUSIVE_RET. If the channel is busy, and the DMA_CANTWAIT flag is on, *dma_setup()* will return DMA_BUSY_RET.

short
dma_select_chan(channel_array)
short channel_array[];

Dma_select_chan() returns the "least busy" channel out of all the channels in channel_array[]. Channel_array is an array of valid channels that the calling device driver can use terminated with the value DMA_END_CHAN. *Dma_select_chan()* first rotors through the array looking for the first non-busy channel. If none are found, *dma_select_chan()* rotors through the array again looking for the first channel which does not have a DMA_EXCLUSIVE request queued on it. If it finds none, *dma_select_chan()* rotors through the array looking for the first channel the does not have a DMA_EXCLUSIVE request running. If no free channel is found, dma_select_chan will return the channel on the top of the channel_array.

unsigned
dma_map(chan,bp)
short chan;
struct buf *bp;

Dma_map() is used to map TCWs for DMA transfers. *Dma_setup()* automatically calls *dma_map()* for the buffer pointer passed in ic_dmabuf. Chan is the channel for the transfer. Bp is a pointer to a buffer structure which describes the transfer. *Dma_map()* returns the ioaddr for the transfer. If there aren't enough contiguous free TCWs to support the transfer, *dma_map()* returns DMA_INV_IOADDR.

void
dma_free_map(chan,ioaddr,len)
short chan;
unsigned ioaddr;
int len;

Dma_free_map() frees TCWs allocated by the *dma_map()* call. Chan is the channel, ioaddr is the I/O address returned by *dma_map()*, and len is the length of the transfer in bytes.

void
dma_go(channel)
short channel;

Dma_go() enables the DMA channel for a transfer.

void
dma_done(channel)
short channel;

Dma_done() frees the channel for other devices to use. All TCWs are automatically freed. If the request was a DMA_EXCLUSIVE request, any device drivers on the callback queue are called. The channel is disabled. The request is dequeued, and the next request on the channel is started.

```
void
dma_wait(chan,callback)
short chan;
struct dma_callback *callback;
```

Dma_wait() queues the callback structure on the DMA channel *chan*. Whenever the channel is free (no DMA_EXCLUSIVE requests are running), the routine specified in the callback structure is called. This allows drivers to wait on DMA channels that are tied up by exclusive requests that refuse to release the bus. If there are no exclusive requests on the channel, the routine specified in the callback structure is called immediately.

```
void
(*dma_get_minphys(ic))()
struct iocc_ctlr *ic;
```

Dma_get_minphys() returns a pointer to the proper minphys routine to be called. The typical usage is in a physio() call when doing raw I/O.

3.4. DMA Callable Routines in the Device Driver

The device driver routines that the DMA code calls are in the device driver's iocc_driver structure (where the probe, slave, attach, and other routines are defined).

```
int
xx_chanrelse (channel)
short channel;
```

Xx_chanrelse() (the routine pointed to by xxdriver->idr_chanrelse) is necessary only for device drivers which use the DMA_EXCLUSIVE flag. This routine is called from dma_setup whenever the exclusive device has the channel, and another device queues a request to use the channel. *Xx_chanrelse()* must either: 1) take steps to release the channel now, or in the near future, then return 0, or 2) decide it cannot release the channel now and return non-zero. If zero is returned, the DMA code assumes that the exclusive device has released the channel (or will do so as soon as the current transfer is completed) and allows the new DMA request to be queued.

```
void
xx_dgo(ic,len,ioaddr,bp)
struct iocc_ctlr *ic;
int len;
unsigned ioaddr;
struct buf *bp;
```

When the channel become available, the DMA code will call *xx_dgo()* so that the device driver can set up the device specific portions of the DMA transfer. *Ic* is a pointer to the controller structure passed by *dma_setup()*. *Len* is the length of the transfer. If no tcw's have been allocated, len is 0. *Xx_dgo()* is still called to let the driver know that an error has occurred. *Ioaddr* is the I/O address for the transfer. If there were not enough TCWs to map the transfer, this value is set to DMA_INV_IOADDR. *Bp* is the buffer passed to *dma_start* in the *ic_dmabuf* field. **int**

```
xx_int(ctlr,irq)
int ctlr;
int irq;
```

Xx_int() is called by DMA code only if the DMA_CANTINT flag is set. Usually the device will generate a normal interrupt when a DMA transfer is completed. This interrupt will be routed to *xx_int()* through the standard interrupt handlers.

4. Using DMA in a Device Driver

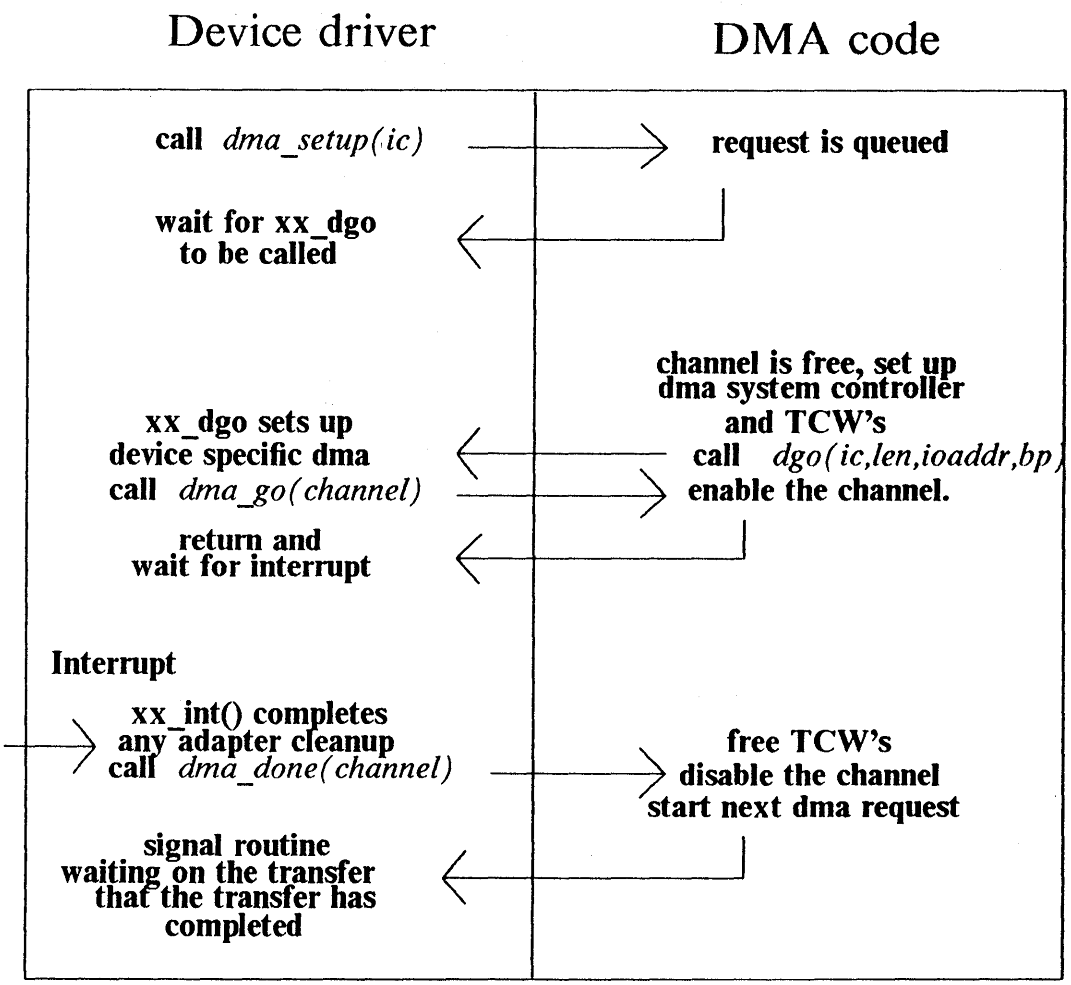
The method of calling DMA varies depending on the way the device driver uses the DMA code. This section describes the normal flow of control between device drivers and the DMA code for several general situations.

4.1. Standard Device Drivers

A standard device driver is a device driver that calls the DMA code each time it wishes to start a transfer, and notifies the DMA code when the transfer has been completed. To start a transfer, the standard device driver loads the ic structure with the channel, flags, and buffer to transfer. The device driver then calls *dma_setup()*, passing it a pointer to the ic structure. *Dma_setup()* will return "DMA_OK_RET" if it successfully queues the DMA request.

When the channel becomes not busy, the DMA code will set up the TCWs and system controller for the transfer and call the device driver's *xx_dgo()*. *Xx_dgo()* is responsible for setting up everything the device needs to start a transfer. When *xx_dgo()* finally needs to enable the channel, it calls *dma_go()*, passing the channel number on which it wishes to transfer. Note that if the channel is not busy when *dma_setup()* is called, all these calls happen before setup returns. *Xx_dgo()* is not guaranteed to be called at the proper interrupt level or in the context of the user process whose transfer it is supposed to service.

When the transfer has completed, the device will generate an interrupt. (If the device doesn't generate an interrupt, see "Using DMA_CANTINT" below). Once the device's interrupt handler determines that the interrupt indicates an end to a DMA transfer, the interrupt handler does any necessary post-transfer cleanup of the adapter and device driver data structures, and calls *dma_done()* with the channel number to indicate that the device driver is done using the channel.



4.2. Using DMA_EXCLUSIVE

Device drivers which plan to hold a channel for an indefinite period of time must use the DMA_EXCLUSIVE flag. This flag tells the DMA code that the device driver must be notified if another driver tries to queue a request on the channel. The exclusive device's *xx_chanrelse()* routine will be called when this happens. *Xx_chanrelse()* must signal the device driver to release the channel. If the device driver cannot release the channel, *xx_chanrelse()* must return non-zero. If the channel can be released, *xx_chanrelse()* must return 0. Releasing the channel means disabling the exclusive device from starting DMA and calling dma_done.

4.3. Using DMA_CANTINT

Some devices don't interrupt when a DMA transfer has been completed. If the device uses system DMA, the device driver can set the DMA_CANTINT flag. When the system DMA controllers complete a transfer, they will generate an interrupt. Any devices which have set DMA_CANTINT will be called. It is up to the individual device driver interrupt routines to determine if transfers have been completed. If a transfer has not been completed, the device driver interrupt routine must return INT_NOT_MINE (defined in

/sys/h/ioccvar.h).

4.4. Using Dma_wait()

It is possible that dma_setup() will reject a DMA request because an exclusive device refuses to give up the channel. If this happens, the device driver can set up a callback structure and call dma_wait(). When the exclusive device finally finishes with the DMA channel (if ever), dma_wait will call the callback routine specified by the device driver in the callback structure. The callback routine could then restart the DMA request.

4.5. Devices Which Can Handle Multiple DMA Transfers

Dma_map() and *dma_free_map()* are provided for devices which can maintain multiple transfers on the same channel. For the first transfer the device driver calls *dma_setup()* normally. If another transfer request comes to the device driver before the first transfer is complete, the device driver can call *dma_map()* to set up the needed TCWs, then hand call *xx_dgo()*. When the first transfer completes, the interrupt routine calls *dma_free_map()* instead of *dma_done()*. *Dma_done()* is called only after all the outstanding transfers have been completed.

A device driver which does this must provide some mechanism to allow *dma_done()* to be called periodically so other device drivers can use the channel. This can be done by keeping a maximum count of consecutive transfers since the last *dma_done()* call. When the number of transfers equals this maximum count, no more transfers are started until all outstanding transfers have been completed and *dma_done()* has been called. An alternative is for the driver to set the DMA_EXCLUSIVE flag. The device driver merrily transfers until all requests have been completed, or until the device driver's *xx_chanrelse()* is called. When the *xx_chanrelse()* is called, it sets a flag to tell the device driver not to start any new requests, then returns 0. When all the outstanding DMA requests have been serviced, *dma_done()* is called and the channel is freed for use by another device driver. **Note:** *Dma_done()* will always free all the TCWs allocated by a channel with the *dma_map()* call.

4.6. Devices Which Can Use Multiple DMA Channels

Some adapters can software select which DMA channels they are going to use. The *dma_select_chan()* routine facilitates the best use of channels. Before calling *dma_setup()*, a device driver calls *dma_select_chan()* with an array of channels which it can use. *Dma_select_chan()* returns the least busy channel in the array. In the following example, the adapter can software select between channels 0, 5, and 7.

```
#include <../machineio/dmavar.h>
.
.
.

/*
 * set up channel array
*/

short dma_select_chan()
short xx_chan_array[] = {
DMA_CHAN0,
DMA_CHAN5,
DMA_CHAN7,
DMA_END_CHAN
};
.
.
.

/* get channel for transfer */
ic->ic_channel = dma_select_chan(xx_chan_array);
/* tell adapter which channel to use (can also be done in xx_dgo) */
xx_set_channel(ic->ic_channel);
/* start dma */
dma_setup(ic);
.
.
.
```

4.7. Consequences of Not Using the DMA Interface

Device drivers which do not use the DMA interface run the risk of being interfered with by other DMA devices. The kernel also uses information provided by the DMA code when it wishes to do things which are unsafe to do while DMA is running. If a device driver writer absolutely needs to grab a DMA channel permanently, the device can call *dma_setup()* with DMA_CANT_WAIT and DMA_EXCLUSIVE. (Of course, if another device has already grabbed the channel, this device driver is out of luck. Using DMA_EXCLUSIVE implies that the device doesn't plan on releasing the channel ever!) The device driver's *xx_chanrelse()* would then always return non-zero.