ARIX
Corporation

ARIX OS90
Installation Guide and Release Notes

Release 3.3

ARIX
Computer Corporation

871 Fox Lane

San Jose, CA 95131

Part No. IP-07458-00 Rev. B

ARIX OS90 Release 3.3 Installation Guide
IP-07458-00 Rev. B
January 6, 1992

The information in this document is subject to change without notice. This document contains the best information available at the time of preparation, but may contain descriptions of functions not implemented at the time this manual was distributed. Please contact your ARIX representative for the latest information about levels of functional implementation and availability.

Table of Contents

Figures

Tables

1 Introduction

This guide describes how to install ARIX OS90 Release 3.3 on an ARIX System90/15, System90/25, System90/45, or System90/85 computer. It includes an appendix of Release Notes describing some new and enhanced features for this release of the software, and provides technical information about some operating system features and utilities.

This manual is organized into the following sections:

Chapter 1 (*Introduction*) explains the purpose of this document and how it is organized.

Chapter 2 (*Preparing for Installation*) lists the hardware required to run OS90 Release 3.3, describes how to power on your ARIX System90/x5 and how to determine device addressing information you need during installation.

Chapter 3 (*Installation Procedure*) describes how to install OS90 Release 3.3 software on a newly purchased ARIX System90/x5 computer.

Chapter 4 (*Completing the Installation*) describes administration tasks that should be performed after OS90 Release 3.3 installation, before making the system accessible to users.

Appendix A (*Release Notes*) provides a section of technical notes and tips.

Appendix B (*vi Notes*) supplies basic operating information for the vi editor to aid in making changes to configuration files.

The **Glossary** defines the acronyms used in this manual.

For a complete description of the procedures used in the administration of an ARIX computer running OS90 Release 3.3, see the *ARIX OS90 Release 3.3 System Administrator's Guide.*

The ARIX OS90 Release 3.3 Product Package

A standard ARIX shipment of OS90 Release 3.3 software generally includes:

- Bootimage, Root, Usr, and Man (root, */usr*, and */usr/man* file systems), on a 1/4-inch tape cartridge. The tape contains either the 68020 version or the 68040 version of OS90 Release 3.3, depending on the system purchased.
- The *ARIX OS90 Release 3.3 Installation Guide* (this document).

United States installations also receive:

- Domestic Overlay tape (utilities using encryption algorithms for U.S. sites), on the same medium as the Bootimage, Root, Usr, and Man tape.

Before you begin any of the installation procedures in this document, check the contents of your OS90 Release 3.3 product package against your packing slip. If your product package is not complete, or if you have any questions, contact ARIX Customer Support:

- 432-1200 from inside the 408 area code (ask for Customer Support).
- 800-521-5783 from inside California but outside the 408 area code.
- 800-237-2783 from outside California.
- 408-432-1200 international (ask for Customer Support).
- 408-435-1694 (FAX).

Additional Products

The Network File System (NFS) is offered as a product separate from ARIX OS90. For detailed information on the software package, see the *ARIX NFS Installation Guide.*

The Remote File Sharing (RFS) package is also offered as a product separate from ARIX OS90. For a description of software installation and build procedures, software notes, and information about documentation, see the *ARIX RFS Installation Guide.*

Network transport services required by ARIX RFS are provided as a separate product. Refer to the *ARIX Ethernet Installation Guide* for additional information.

Additional ARIX OS90 Documentation

Additional OS90 documentation can be ordered either in sets or individually.

ARIX OS90 Release 3.3 Network Programmer's Guide

ARIX OS90 Release 3.3 Programmer's Guide

ARIX OS90 Release 3.3 Programmer's Reference

ARIX OS90 Release 3.3 System Administrator's Guide

ARIX OS90 Release 3.3 System Administrator's Reference

ARIX OS90 Release 3.3 User's Guide

ARIX OS90 Release 3.3 User's Reference

ARIX OS90 Release 3.3 STREAMS Primer and Programmer's Guide

ARIX OS90 ASSIST User's and Development Tools Guide

For more information or to order these volumes, contact an ARIX representative.

Notation Conventions

This document uses the following typographical conventions to depict user interaction with the system.

Bold Bold indicates a command that is to be entered exactly as stated. For example:

Before shutting down the system, enter:

sync

When a command or program is referenced in text, the full form is shown in bold type. For example:

Enter the **sync** command to ensure that changes are flushed from main memory back to disk.

Because UNIX commands are case-sensitive, enter the command exactly as shown.

[ ] Optional parameters, arguments, or flags are enclosed in brackets.

ls [-l]

Italics Italics indicate either variable parameters for which the user substitutes a specific value, or names of files or directories. For example:

When an attempt is made to reopen the mirrored slice, the system checks the *mirrortab* file.

Italics are also used for comments within a screen display. For example:

The system console displays text similar to the following:

```
Testing SPM Integrity
Sram data ripple test.
```

(The lines that follow overwrite the line above.)

```
Sram address ripple test.
Sram content test.
```

`Courier` System responses to a user's command appear in Courier font. For example:

The system displays the message

```
You have mail.
```

«Key» The name of a key to press. For example:

To toggle between input and command modes, press «Esc».

To access the Set Up menu, press «F1».

When two keys are to be pressed simultaneously, each key name is shown within a set of double-angle brackets with no space in between commands. For example:

To scroll down one full screen, press «Ctrl»«c».

bold(n) A command followed by a number, such as **ed**(1), indicates a command, file, routine, or utility and the section of manual pages ('man pages') in which its syntax and description can be found.

... Horizontal ellipses indicate that the preceding command can be repeated one or more times.

Notes and Cautions

Portions of text are highlighted as 'Notes' or 'Cautions'.

NOTE:

A Note is supplementary or background information, often a hint or a reminder related to product use.

CAUTION:

A Caution presents information that is crucial to the operation of a system. Failure to heed a Caution can cause system failure and/or loss of data.

---

2 Preparing for Installation

This chapter describes how to power on your newly installed ARIX System90/x5 computer, and explains how to determine the device addressing information you need to complete the installation procedure, as described in Chapter 3.

Hardware Prerequisites

Table 2-1 lists the acceptable System90 board revision levels for OS90 Release 3.3.

NOTE:

The revision levels listed in Table 2-1 are provided as guidelines only and are subject to change. Contact ARIX Customer Support as described in Chapter 1 for the most recent information.

Board	Part No.	Current Revision	Acceptable Revisions
Processor Module 020	80-02194	D7	D6 – D7
Processor Module 040	80-07064	A1	A1
Memory Module	80-02196	D9	C2 – C3, D8 – D9
I/O Module II	80-06161	B1	B1
Service Processor	80-02222	D4	D3 – D4*
Real World Interface	80-02580	C4	C4
PSM Distribution	80-03319	C2	C1 – C2
Arbiter, 8-Slot	80-03363	A1	A1
System Console	80-03549	C1	B1, C1
Motherboard, 16-Slot	80-03776	B3	B1 – B3
Arbiter, 16-Slot	80-03818	A4	A4
AC Module	80-03465	A1	A1
Keyswitch 45/85	80-03973	C1	B1, C1
Display	80-03975	A1	A1
Temp Sense	80-03977	C1	B1, C1
LAN/WAN	80-04534	C1	B2, C1
IOPM 1MB	80-04640	C6	B4 – B6, C3 – C6
SCSI Dual	80-04688	B5	B5*
Motherboard, 8-Slot	80-05355	A	A
Power Bus	80-05434	A1	A1
Display, 25	80-05440	A2	A1 – A2
I/O SBA	80-05701	B1	B1
Async Comm DB RJ45	80-06037	A1	A1
Async Comm DB RJ12	80-04399	B1	B1
Async Comm Extend RJ45	80-06048	A1	A1
Async Comm Extend RJ12	80-04401	B1	B1
Async Comm RW RJ45	80-06042	A1	A1
Async Comm RW RJ12	80-05442	B1	B1
† SPM 80-02222 must be a minimum revision of D4 for use with Processor Module 68040.			
* If SPM 80-02222 is revision Cc or D4, DSDB 80-04688 must be at least Revision B5.			

Table 2-1 Hardware Prerequisites

Powering On the Computer

Once your System90/x5 computer is unpacked and its processor modules and peripherals are installed, follow this procedure the first time you power on the computer (refer to the *Hardware Installation Guide* for your System90 computer if you need additional detail).

1. Verify that the machine is plugged into a proper power source.

2. Turn the circuit breaker to the ON position. If your machine is a System90/25, the circuit breaker is located on the back of the unit, in the lower left corner; if your machine is a System90/45 or System90/85, it is on the back on the unit, in the lower right corner.

   The Service Processor Module powers on, and the system displays messages similar to the following:

   ```
   SPM ROM Version x.x
   Testing SPM Integrity
         Local CIO R/W Test
   SPM Power-up Diagnostics Passed

   Please turn key to 'ON' position when ready to continue, or
   ESC to enter local monitor.
   ```

3. Turn the key on the front panel clockwise, to the ON position, and enter the following:

   pwron «Return»

   This supplies power to the entire system, which displays:

   ```
   *CSS clock active (PROM Version x.xx).

         (diagnostic messages)

   Press any key to abort autoboot . . . .
   ```

4. If the autoboot process comes up, press any key on the keyboard within two seconds to abort it. If autoboot is already disabled, the system displays:

   ```
   Autoboot flag is disabled.
   ```

NOTE:

If the autoboot process is not stopped in time, the system tries to boot from the disk. When the disk is blank and the system cannot find the boot program, it lets you know by displaying error messages. If you fail to stop the autoboot process, turn the key on the front panel to the RESET position and start the installation procedure again.

When the autoboot process stops, the system displays a message in the format:

```
Returning to Monitor
SPM ROM Version x.xx.xx
PROM Level Menu:
```

You are now ready to determine the correct device addressing information for your system.

Determining Device Addressing Information

When installing OS90 Release 3.3, you must specify the physical address of the device onto which the root and */usr* file systems are to be loaded; therefore, you need to map the name of the device to its physical address.

UNIX addresses disk slices through special devices whose names reflect the logical controller and physical disk (for example, /dev/dsk/c0d1s3). The SPM PROMs and SPM Runtime Monitor address disk slices with names that reflect the physical slot that the controller resides in (for example, 7d1s3).

The following sections will help you identify the slot and the disk number you need to properly name the disk slice onto which you will load the system.

This section explains the mapping process and describes how to address devices on ARIX System90/x5 computers with OS90 Release 3.3. In a first-time installation, the procedures in this section are normally performed after the new system is powered on and the autoboot sequence is aborted.

NOTE:

The System90/85 computer can support several I/O buses in CSS/XA expansion cabinets connected to the main system bus. OS90 Release 3.3 provides an addressing format for the single-bus configurations and an alternate format for system configurations that include CSS/XA expansion cabinets. Examples of addressing devices controlled by IOPMs located in expansion cabinets are shown in this section only.

The System90/25 and System90/45 computers have only one I/O bus, and take the simpler addressing format shown in the examples throughout this guide.

Device-Mapping Your System

Enter the **system** command (short form: **sy**) at the `PROM Level Menu:` prompt to determine the slot numbers of the devices on your system:

```
sy
```

The system displays something similar to:

```
Power Supply 0 type is Switching Power
Power Supply 1 type is No P/S Detected.
Power Supply 2 type is No P/S Detected.
Power Supply 3 type is No P/S Detected.
Power Supply 4 type is No P/S Detected.
Runtime code default = 8d0s15spm
Configuration default = 7d0s15spm

   (additional messages)

Diagnostic AREA NVRAM checksum is:  7741

Any key to continue ...
```

Press any key to continue. The system displays messages similar to the following:

```
SPM             module in slot 0x0 (0)
PM              module in slot 0x1 (1)
MM              module in slot 0x4 (4)  (xx Meg)
IOPM            module in slot 0x8 (8)
IOPM            module in slot 0x9 (9)
IOPM            module in slot 0x0a (10)

Any key to continue ...
```

The system displays slot numbers for each device on your system. The slot numbers are given in both hexadecimal and decimal form. For example, the third IOPM module is in *hexadecimal* slot 0x0a, or *decimal* slot 10.

NOTE:

At this point in the installation, the system is unable to identify what kind of device board (DSDB, ACDB, or LAN/WAN) is attached to an IOPM in a particular slot — only that a module is present. Without opening the cabinet, the only way to determine what kind of device board is located in a particular slot is to check the device configuration information you received with your system.

If there is any discrepancy, contact ARIX Customer Support immediately, as described on page 1-2.

The name of a device (disk drive or tape drive), as known to the SPM PROMs or SPM Runtime Monitor, is formed by combining the device name (d or mt), SCSI ID number, and the system slot number for each device on the system.

The SCSI specification allows eight devices, numbered 0 – 7 (including the master), on a SCSI bus. The DSDB drives two SCSI buses and is SCSI ID 7 on each of the two buses.

Each SCSI device ID is determined by jumpers or dip switches on the device. Devices with SCSI IDs 0 – 6, when attached to SCSI 0 on the DSDB board, have DSDB SCSI ID numbers 0 – 6 (see Figure 2-1). Devices 0 – 6, when attached to SCSI 1 on the DSDB board, have DSDB SCSI ID numbers 8 – 14.

NOTE:

Record the physical device name and system slot number for each device in your system. You need this information during the OS90 Release 3.3 installation procedure.

The format for device addressing differs slightly for each physical device type. Read following sections, select the type of device you want to address, and refer to the examples to determine the format and address for the devices on your system.

Disk Addressing

Access the disk drive by using the following device addressing format:

*X*dZ

where:

X represents the number of the CSS slot in which the IOPM/DSDB being addressed resides. See the sections *System90/25*, *System90/45*, and *System90/85*.

Z represents the SCSI device ID number relative to the IOPM/DSDB of the disk being addressed. (*Z* is a value from 0 – 6 or 8 – 14, depending on the SCSI channel to which the drive is connected.) See Figure 2-1.

Each number is specified in decimal (base 10).

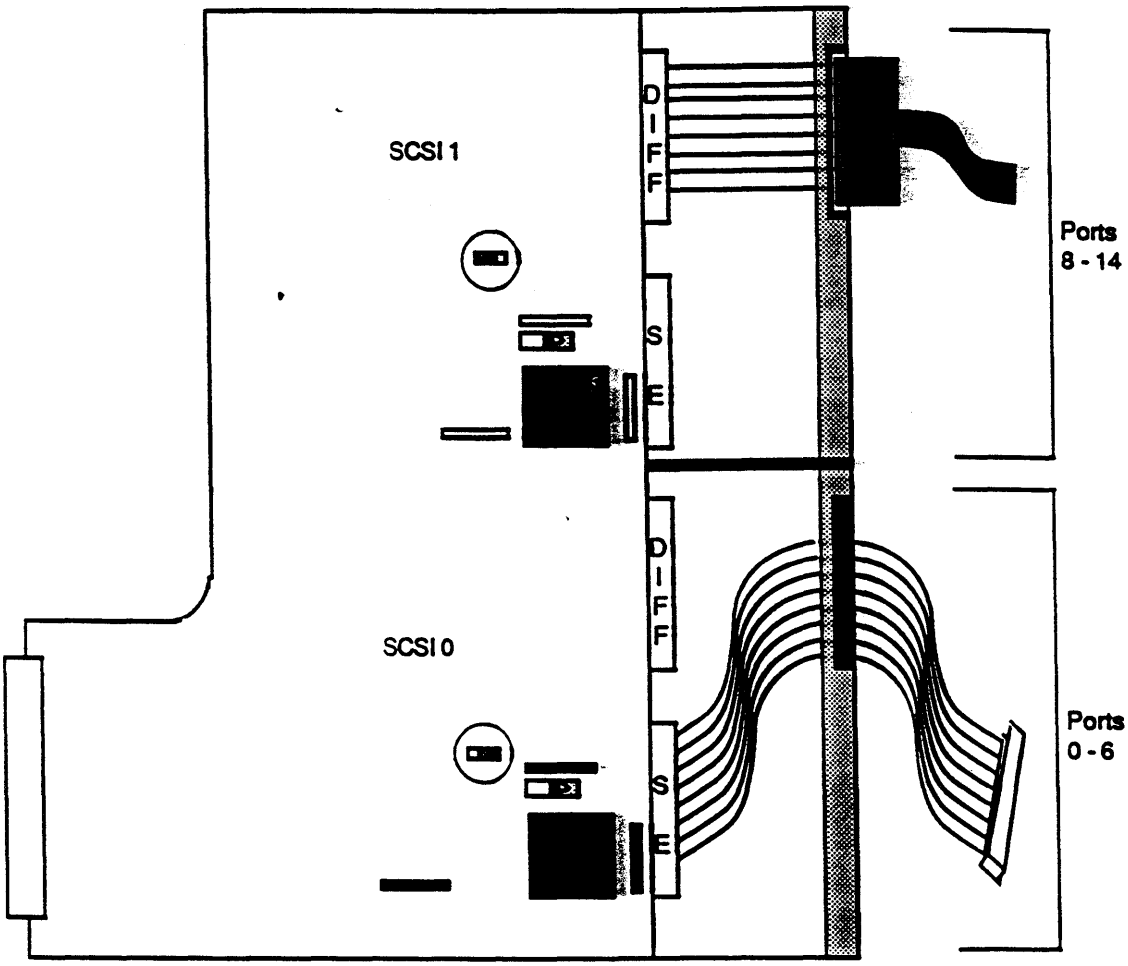

Figure 2-1 Dual SCSI Device Board

System90/25

The System90/25 can support two disk drive controllers. Disk drive controllers are located in slots 6 and 7 of the System90/25. If your system only has one controller, it is installed in slot 7.

At the UNIX level, the two controllers are identified as c0 and c1. The controller in the lowest-numbered slot (rightmost slot viewed from the front) is c0. The controller in the next higher-numbered slot is c1. That is, if your system configuration includes one controller, it is installed in slot 7, and it is identified as c0. If the system has two controllers, the first controller is installed in slot 6 and is identified as c0, and the second controller is installed in slot 7 and is identified as c1.

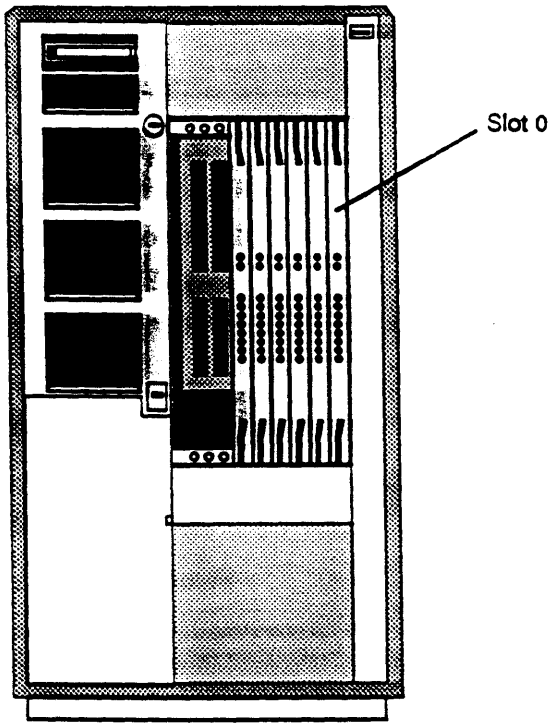

Figure 2-2 System90/25, Front View

System90/45, System90/85 Main Cabinet

The System90/45 can support up to six disk controllers, which are usually located in slots 9 through 15.

At the UNIX level, these controllers are identified as c0 through c5, corresponding to the slots in which the controllers are installed from right to left, viewing the system from the front. The first controller from the right is always c0.

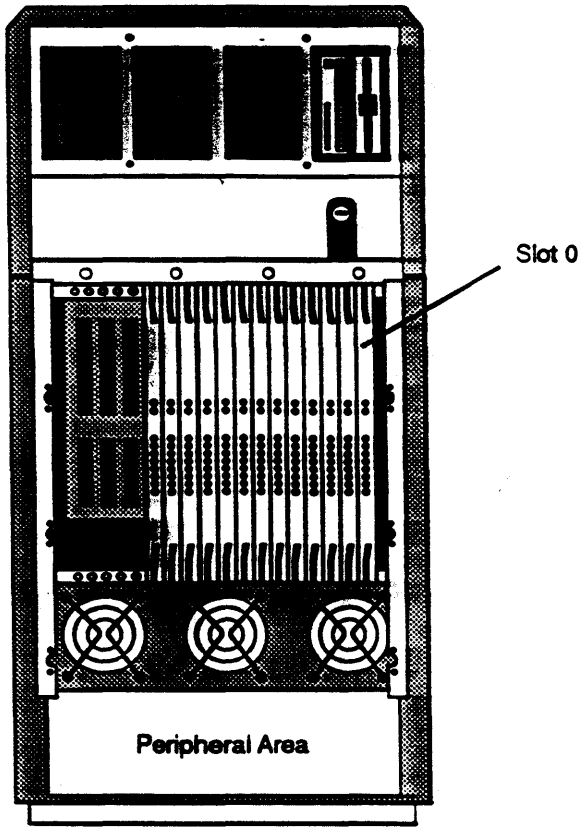

Figure 2-3 System90/45, Front View

System90/85 with CSS/XA Expansion Cabinet

The System90/85 can support up to six disk controllers, generally located in slots 9 through 15 and in any slot (0 – 15) in the expansion cabinet(s).

At the UNIX level, these controllers are identified as c0 through c5, corresponding to the slots in which the controllers are installed from right to left, viewing the system from the front. The first controller from the right is always c0.

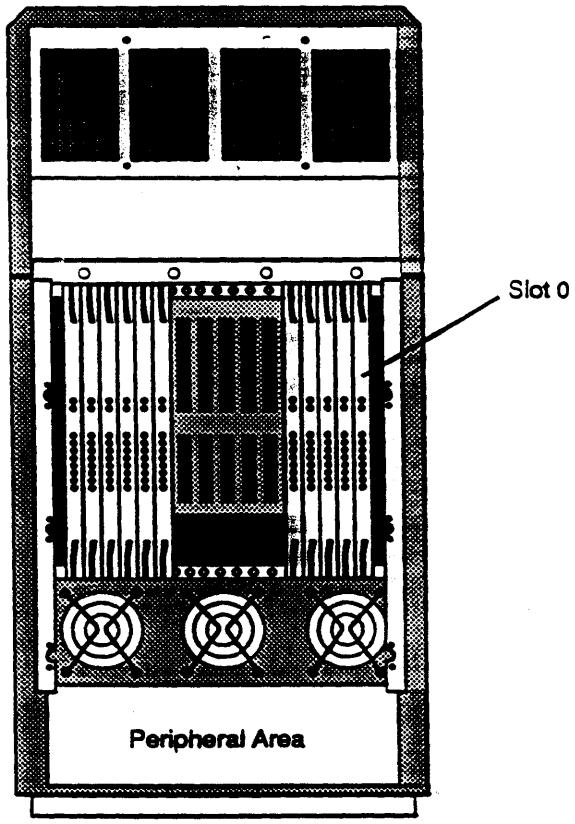

Figure 2-4 System90/85 CSS/XA Expansion Cabinet, Front View

Expansion cabinets are connected to the main CSS bus through IOM2 cards in the main cabinet. This means that IOPM/DSDBs in expansion cabinets are each associated with a slot in the main cabinet. In fact, for addressing purposes, the main cabinet slot with which a controller is associated is more significant than the expansion cabinet slot in which the controller resides. For example, if a system has IOPM/DSDBs in slots 9 and 11 in the main cabinet and one in slot 5 of an expansion cabinet controlled by an IOM2 in main cabinet slot 10, c0 would be the IOPM/DSDB in slot 9, c1 would be the IOPM/DSDB in the expansion cabinet, and c2 would be the IOPM/DSDB in slot 11. This example configuration is illustrated in Figure 2-5.

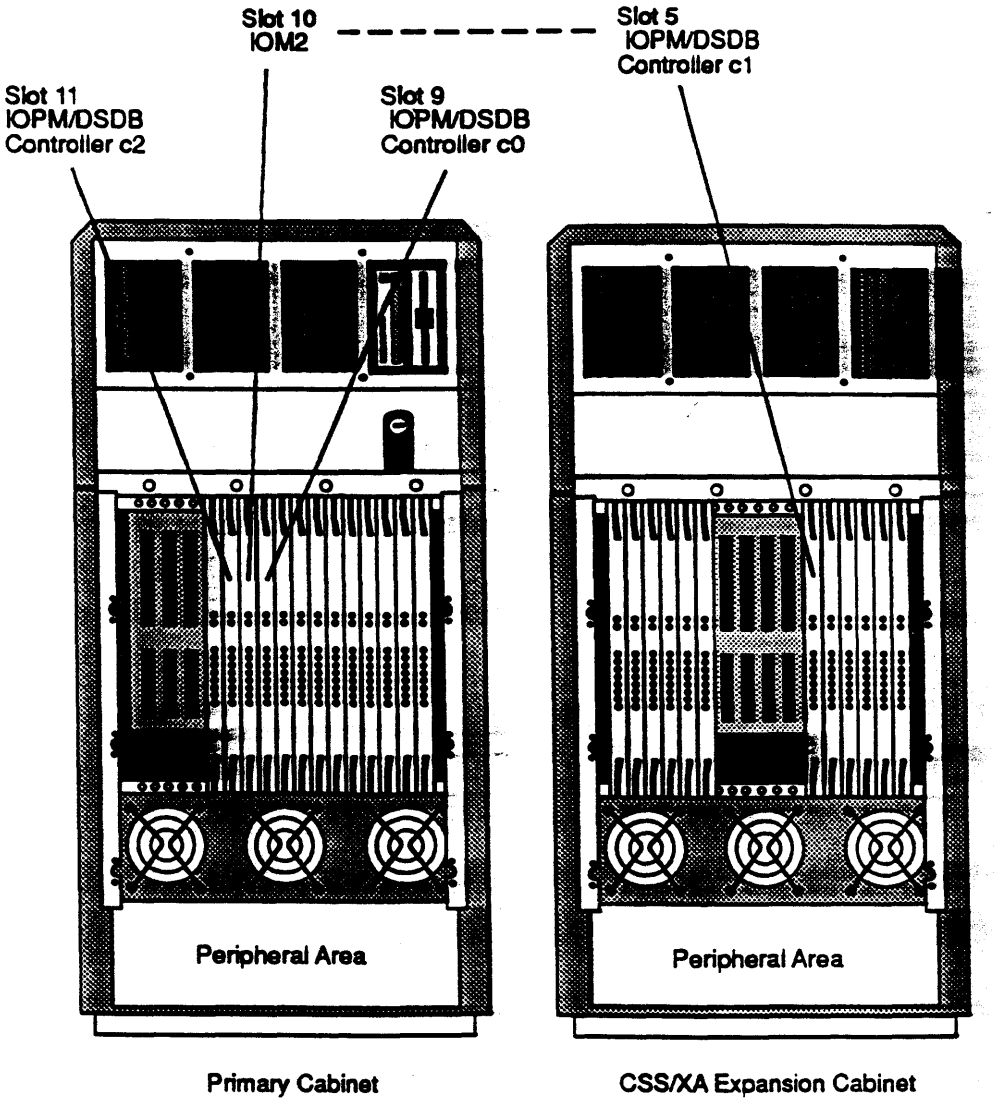

Figure 2-5 System90/85 Example IOPM/DSDB Configuration, Front View

Disk Slice Addressing

Access a particular slice (partition) on a disk by using the appropriate addressing format for your System90 computer.

System90/25 and System90/45

The disk slice addressing format is:

*X*dZs*N*

where:

X represents the number of the CSS slot in which the IOPM/DSDB being addressed resides. (See the sections *System90/25*, *System90/45*, and *System90/85*.)

Z represents the SCSI device ID number relative to the IOPM/DSDB of the disk being addressed. (*Z* is a value from 0 – 6 or 8 – 14, depending on the SCSI channel to which the drive is connected.) See Figure 2-1.

s*N* represents the logical (s)lice [partition] (*N*)umber relative to *Z* (see the section *Finding the Value for sN*.)

Each number is specified in decimal (base 10).

System90/85 Alternate Format

The disk slice addressing format is:

X/*Y*dZs*N*

where:

X represents the number of the CSS slot in which the IOM2 resides which connects to the expansion bus being addressed.

Y represents the number of the CSS/XA slot in which the IOPM/DSDB being addressed resides.

Z represents the SCSI device ID number relative to the IOPM/DSDB of the disk being addressed. (*Z* is a value from 0 – 6 or 8 – 14, depending on the SCSI channel to which the drive is connected.) See the Figure 2-1.

s*N* represents the logical (s)lice [partition] (*N*)umber relative to *Z*. See the section *Finding the Value for sN*.

Each number is specified in decimal (base 10).

NOTE:

The System90/85 has at least one CSS/XA cabinet; the System90/45 may have a peripheral expansion cabinet, but has no CSS/XA expansion cabinets.

Finding the Value for sN

sN represents the slice of the physical device (drive) that is being addressed. The available slices for any disk drive device can be determined by using the **iopmfmt** command from the Runtime Boot Level Menu. (**iopmfmt** also resides in */local/bin* in OS90.) **iopmfmt** prompts you for the controller slot number and the drive ID number, in the *XdZ* format discussed above. Once you have entered **iopmfmt** and selected the device, you can select **2 – Slice table management** followed by **4 – Display slice table** to display the list of slices for the disk drive selected. Refer to the section *Slicing a Disk Using iopmfmt* in Chapter 3.

Once OS90 is up and running, the slice table for any device can be displayed by using the **/etc/prtvtoc** utility in OS90. Refer to the *OS90 Release 3.3 System Administrator's Reference Manual* for the command and options.

NOTE:

If you are upgrading an existing system to Release 3.3 of OS90 and want to retain a customized disk configuration, be sure to note the slice table configuration before beginning the installation. The slice table configuration for the previous release is overwritten during the Release 3.3 installation, and you will need to re-enter your system's values.

Default Slices

The boot image is always loaded in slice 15 (s15) which is created by using **iopmfmt** when a drive is partitioned as a boot drive.

The root file system is usually on slice 0. It is loaded to that slice if you select the default slice during installation.

The */usr* file system is usually on slice 2, but this is left up to you during installation. There is no default slice for the */usr* file system.

The swap area is on slice 1 by default, when a boot drive is sliced using **iopmfmt**. There must be a swap area on the same physical drive as the root file system. Swap areas can be appended by using the **swap** utility in OS90.

Cartridge Tape Addressing

Access the cartridge tape drive by using the appropriate addressing format for your System90 computer.

System90/25, System90/45, and System90/85 Main Cabinet

The cartridge tape drive addressing format is:

*X*mt*W*

where:

X represents the number of the CSS slot in which the IOPM/DSDB being addressed resides.

W represents the tape device's SCSI drive ID number. (*W* is a value from 0 – 6 or 8 – 14, depending on the SCSI channel to which the drive is connected.) See Figure 2-1.

System90/85 CSS/XA Cabinet

The cartridge tape drive addressing format is:

X/*Y*mt*W*

where:

X represents the number of the CSS slot number in which the IOM2 resides which is cabled to the I/O cabinet.

Y represents the number of the CSS/XA slot in which the IOPM/DSDB being addressed resides.

W is the tape device's SCSI drive ID number. (*W* is a value from 0 – 6 or 8 – 14, depending on the SCSI channel to which the drive is connected.) See Figure 2-1.

NOTE:

The device addressing examples in Chapter 3 show the ARIX System90/25 and System90/45 format only. If your computer is an ARIX System90/85, substitute the alternate format as needed.

You are now ready to load the Bootimage, Root, Usr, and Man tape. Continue with the instructions in Chapter 3, *Installation Procedure.*

3 Installation Procedure

This chapter describes how to install OS90 Release 3.3 software on your ARIX System90/x5 computer.

It is assumed that:

- The installer has superuser privileges, and is already familiar with procedures such as reconfiguring the OS90 kernel and booting the system;
- The installer uses standard OS90 naming conventions (such as the naming of the */usr* slice); and
- The installer uses the UNIX vi editor to change any necessary entries in the system files. If you are unfamiliar with vi, refer to Appendix B for a summary of basic vi commands and operations.

NOTE:

The device addressing examples in Chapter 3 show the ARIX System90/25 and System90/45 format only. If your computer is an ARIX System90/85, substitute the alternate format as needed (see Chapter 2 for information).

For further information on reconfiguring the kernel and booting the system, see the *ARIX OS90 Release 3.3 System Administrator's Guide.*

For full syntax and definitions of the system commands referenced in this installation procedure, see the *ARIX OS90 Release 3.3 System Administrator's Reference Manual.*

Loading the Bootimage Files

The bootimage files are a set of standalone programs, the diagnostic image and the SPM runtime program. These programs include the **iopmfmt** disk formatting and partitioning utility.

The bootimage programs are the first set of files on the tape. You must copy the bootimage files to the system before all other files.

Complete these steps to boot the standalone programs from tape, prepare the disks, and load the bootimage files onto your system.

Booting the SPM Runtime Code from Tape

1. Insert the tape labeled *Bootimage Root Usr Man* into the tape drive. Leave the tape in the drive throughout these procedures and do not remove it until instructed to do so.

NOTE:

Check the CPU version on the tape label before attempting to load the tape. If your system has a 68040 CPU, make sure you insert the 68040 release tape. If the system has a 68020 CPU, make sure you use the 68020 release tape. Although OS90 Release 3.3 functionality is the same for both CPUs, the tapes are CPU-specific and cannot be interchanged.

---

2. Boot the SPM Runtime Code from your tape. At the `PROM Level Menu:` prompt, enter the **boot** (or **b**) command in the following format:

   b *X*mt*Y*spm

   where *X* represents the slot where the tape drive is physically installed and *Y* is the SCSI ID of your system's cartridge tape drive.

   For example, if your tape drive's controller is in slot 8 and its SCSI ID is 3, you would enter:

   b 8mt3spm

The system displays information similar to the following:

```
Read directory block
Read binary image

SPM Runtime Code: Version 0x######## ID x.x.x

SPM                module in slot 0
PM20               module in slot 1
MM (xx Meg)        module in slot 4
IOPM/DSDB          module in slot 8
IOPM/LAN-WAN       module in slot 9
IOPM/ACDB          module in slot 10

initializing memory ... done

Type `?' for help
Runtime Boot Level Menu:
```

Running the iopmfmt Utility

The **iopmfmt** utility program is a tool for formatting a disk, partitioning a disk into slices, and displaying the layout of a disk. Before loading a copy of the bootimage files to a disk, use this utility program to verify the format of the disk, partition the disk, and display the layout of the disk. Disks supplied by ARIX are formatted at the factory and do not normally require you to format them again. Use these steps to check the formatting and ready the disk for the bootimage files.

1. Access the **iopmfmt** utility. At the `Runtime Boot Level Menu:` prompt, enter:

   iopmfmt *boot device*

   where *bootdevice* is the device name of the tape drive you used to boot OS90 3.3.

   The system displays a message similar to the following:

```
Loading from 8mt3spm
default base is decimal; for octal use 0NNNNN; for hex
use 0xNNNNN
Enter device name:
```

2. Enter the device address for your disk drive. Use the format:

   *X*dY «Return»

   where:

   X represents the CSS slot number of the IOPM/DSDB being accessed.

   dY represents the SCSI ID of the drive relative to *X*.

   (If necessary, refer to the section *Determining Device Addressing Information* in Chapter 2.)

3. The following is an example of what should be displayed next.

```
device is (XdY)
XdY CDC      model 94171-9

The device is formatted.

      MAIN MENU Ver:xxxxxx

1-Disk Format and initialization
2-Slice table management
3-Disktest
0-Exit

Select:
```

a. If you see the above messages and menu display, the disk is already formatted and is ready to be partitioned. Skip the next subsection (*Formatting a Disk Using iopmfmt*) and proceed to the subsection entitled *Slicing a Disk Using iopmfmt.*

b. If **iopmfmt** finds the disk drive at the disk address but the disk is not formatted (or the disk is not in the IOPM format), the program displays messages similar to the following:

```
device is (8d0)
8d0 CDC      model 94171-9

The device has an unknown format.

      MAIN MENU Ver:XXXXXXX

1-Disk Format and initialization
2-Slice table management
3-Disktest
0-Exit

Select:
```

In this case, you need to format the disk before it can be partitioned. Proceed to the next subsection, *Formatting a Disk Using iopmfmt.*

Formatting a Disk Using iopmfmt

If necessary, format (or reformat) the disk. Do the following:

1. Select Disk Format and initialization from the **iopmfmt** Main Menu by typing **1** and pressing «Return». **iopmfmt** displays:

```
        FORMAT MENU

1-Format and Configure disk
2-Format, discard reassigned blocks and Configure disk
3-Configure disk
0-Previous Menu

Select:
```

2. Select Format and Configure disk from the **iopmfmt** Format Menu by typing **1** and pressing «Return». **iopmfmt** displays:

```
WARNING - Formatting will destroy all of the data on
the disk!

Do you really wish to format?(y/n):
```

3. Enter **y** and press «Return» to format the disk.

   The formatting process can take from 5 to 35 minutes, depending on the type and size of the disk drive.

4. When formatting is complete, enter **0** and press «Return» to return to the **iopmfmt** Main Menu.

   iopmfmt redisplays the Main Menu:

```
        MAIN MENU Ver:XXXXXXX

1-Disk Format and initialization
2-Slice table management
3-Disktest
0-Exit

Select:
```

Slicing a Disk Using iopmfmt

The **iopmfmt** program is also used to organize the disk into partitions or logical slices. When you partition your system's disks, you can ensure that plenty of space is allocated for the root and */usr* file systems, as well as the system swap area by following these recommendations:

- It is recommended that you allocate at least 20 megabytes for the root slice, to allow room for the system files and for application software to be loaded later.
- The swap slice is automatically set to 16 megabytes by the procedure below. If necessary, you can add extra swap slices after the system is running, spreading the swap load across several devices. See the **swap**(1M) entry in the *ARIX OS90 Release 3.3 System Administrator's Reference Manual* for further information.
- If sufficient disk space is available, 60 megabytes are recommended for the */usr* slice. If both TCP/IP and MOTIF/X11 are to be installed, 70 megabytes are recommended for the */usr* slice.
- It is recommended that you allocate a separate slice for the */tmp* directory.
- You may also want to allocate separate partitions for any large databases or similar applications.

Refer to the *ARIX OS90 Release 3.3 System Administrator's Guide* for detailed information on planning the disk space allocation for your system.

Usually, the root, swap, and */usr* partitions are slices 0, 1, and 2, respectively.

To partition a disk into slices, do the following:

1. Select Slice table management from the **iopmfmt** Main Menu by typing **2** and pressing «Return». **iopmfmt** displays:

```
SLICE MENU

1-Auto slicing
2-Semi-auto slicing
3-Editing slice table
4-Display slice table
5-Save slice table
6-Show slice bar graph
0-Previous Menu

Select:
```

If you select **Auto slicing**, the system automatically sets slice 0 to 20MB, slice 1 to 16MB, and slice 2 to 60MB; if you want to change these sizes you have to edit the slice table. ARIX recommends using semi-auto slicing as the slicing method. Selecting **Semi-auto slicing** displays prompts that allow you to enter the size and type for each slice, eliminating the need to edit the slice table.

2. At the `Select:` prompt, type **2** and press «Return». The following is displayed:

```
Semi-auto slicing sets the disk up with a diagnostic
and defect slice.

Align slices on cylinder boundaries? (y/n)
```

3. At the prompt, enter **y** and press «Return». The following is displayed:

```
Is bootable system disk? (y/n)
```

4. Responding **y** to this question tells the program to set aside a small amount of space on slice 15 to store the standalone boot programs. It is a good practice to have these programs on all the disks in the system to enable booting the diagnostic levels in the event of failure of an operating system drive. Doing this defaults slice 1 to type swap.

   If you do not want a swap slice on slice 1, you can continue with steps 5 through 8 below, and when the slice menu is again displayed, type 3 and press «Return» to select **Editing slice table.** You would then need to enter the slice number, offset, and size of slice 1, and change the type to UNIX. Alternatively, you could or answer **n** to the `Is bootable system disk?` prompt, so that slice 1 does not default to type swap.

   After you finish responding to the `Is bootable system disk?` prompt, the following message should be displayed:

```
Kbytes remaining XXXXXX

Slice 0, enter slice size (Kbytes):
```

5. Enter the desired size for slice 0. For example, for a 20 megabyte slice, you would enter:

   20000

6. Next you are prompted to enter the file system type. At the prompt type **u** and press «Return» for a UNIX file system.

   Notice that the slice configuration table is displayed reflecting the size of slice 0 you just entered and that slice 1 is defaulted to type swap. After you enter each slice size the `Kbytes remaining XXXXXX` message is reduced accordingly.

7. Continue entering the desired sizes for the respective slices. When you reach the last slice to be set, you must enter the exact size displayed in the `Kbytes remaining XXXXXX` message to achieve the correct results.

8. Once the last slice has been set to the exact number of remaining Kbytes, the slice menu is again displayed.

9. View the configuration you have just created and verify that it is correct. At the prompt select Display slice table by typing **4** and pressing «Return».

10. When you are certain the table is correct, save it to the disk. At the prompt select **Save slice table** by typing **5** and pressing «Return». Type **y** and press «Return» when you are prompted to confirm that you really want to save the slice table.

11. Exit the slice menu by typing **0** and pressing «Return». This redisplays the **iopmfmt** main menu.

12. Exit the **iopmfmt main menu** by typing **0** and pressing «Return». The following prompt is displayed:

```
Would you like to run iopmfmt on another disk?  (y/n):
```

    Type **n** and press **«Return»** to redisplay the `Runtime Boot Level Menu:` prompt.

If your system has more than one disk drive, you must verify the formatting of each disk and slice each disk into partitions. For each disk, repeat the procedures in the sections *Running the iopmfmt Utility*, *Formatting a Disk Using iopmfmt*, and *Slicing a Disk Using iopmfmt*. Then continue with the section *Loading the Bootimage Files to Disk*, below.

Loading the Bootimage Files to Disk

Complete these steps to copy the bootimage to disk(s).

1. Load the standalone bootimage onto the disk. At the `Runtime Boot Level Menu:` prompt, enter:

   ldsa

   The system displays:

   ```
   Standalone ldsa ver x.xx
   Enter source device (X/YmtZ, XmtY or `?'):
   ```

NOTE:

If you want to exit the standalone boot level menu (ldsa) and return to the runtime boot level menu prompt, press «Esc».

---

2. Enter the address of the source tape device.

   If your computer is a System90/25 or System90/45, use the format:

   X*mt*Y

   X represents the CSS slot number of the IOPM/DSDB being accessed.

   mt*Y* represents the SCSI ID number of the tape drive relative to *X*. (If necessary, refer to the section *Determining Device Addressing Information* in Chapter 2.)

   The system displays:

   ```
   Enter target device (X/YdZrv, XdYsZ, or `?'):
   ```

3. Enter the device address of the your target disk drive.

   If your computer is a System90/25 or System90/45, use the format:

   X*d*Y*s*Z

   where *X*d*Y* is the device address of the disk for your system, and s*Z* is the boot slice (the default boot slice created by autoslicing is 15, so this last entry is likely to be **s15**). (If necessary, refer to the section *Determining Device Addressing Information* in Chapter 2.)

The system copies the bootimage files to the address you specify, and then displays a message in the following format:

```
Copying bootimage to XdYsZ.
xxxxx bytes transferred

Type `?` for help
Runtime Boot Level Menu:
```

The `xxxxx bytes transferred` message lets you know that the copy of the bootimage files is complete.

NOTE:

If your system has more than one bootable disk drive, it is recommended that you copy the bootimage files to all such drives. This enables the system to boot from any of the drives.

Copy the bootimage files to other drives by repeating the copy procedures. Remember to replace the default boot path with the appropriate boot path for each drive.

NOTE:

If instead of the above display you see a SCSI fatal error message, your disk needs to be reformatted before you can load the bootimage files. Follow the steps in the section *Running the iopmfmt Utility* to format the disk.

---

4. Verify that the bootimage files are properly installed on your system by accessing the bootimage program from disk. Accomplish this by exiting the Runtime Boot Level Menu; enter:

   exit

Type **y** and press «Return» when you are prompted to confirm that you really want to reset the system. The system resets and begins diagnostic testing (the diagnostic programs and SPM runtime code are loaded with the bootimage). When the testing is done, the system displays:

```
*CSS clock active (PROM Version x.x.x).

        (diagnostic messages)

Autoboot flag is disabled
Returning to Monitor

SPM ROM Version x.x.x
PROM Level Menu:
```

5. Boot the SPM Runtime code from disk. Enter the boot command in the format:

   b *XdYsZ***spm**

   where *XdY* is the device address of the disk for your system, and s*Z* is the boot slice (where the Runtime code is stored).

   When the SPM Runtime code boots, the system displays something similar to the following:

```
SPM Runtime Code: Version xxxxxx, ID xxx

SPM                 module in slot 0
PM20                module in slot 1
MM (xx Meg)         module in slot 4
IOPM/DSDB           module in slot 8
IOPM/ACDB           module in slot 9
IOPM/LAN-WAN        module in slot 10

initializing memory ... done

Type `?` for help
Runtime Boot Level Menu:
```

Continue to the next section, *Loading the Root File System.*

Loading the Root File System

This procedure describes how to make and load the root file system.

1. Make the root file system. At the SPM `Runtime Boot Level Menu:` prompt, enter:

   mkfs

   The system prompts:

   ```
   Standalone mkfs VER x.xx

   Load the file system tape.
   Enter the tape device number (X/YmtZ, XmtZ, or `?'):
   ```

2. Verify that the tape labeled *Bootimage Root Usr Man* is still in the tape drive, then enter the tape device number. Use the format:

   *X*mt*Y*

   where *X* is the controller's CSS slot number and Y is the tape drive's SCSI ID number. The system displays:

   ```
   Does this tape contain both a Bootimage and a Root
   System?
   ```

3. Enter y. The system displays:

   ```
   file system disk device (X/YdZsN, XdYsZ, or '?'):
   ```

4. Enter the device address (including the slice) of the disk on which you want to store the root file system. Use the format:

   *X*dZs*N*

   where:

   X represents the CSS slot number of the IOPM/DSDB being accessed.

   Z represents the SCSI ID number of the drive relative to *X*.

   N represents the slice.

5. The system prompts for the file system size:

   ```
   file system size [default=xxxx Kblocks]:
   ```

6. Press «Return» to accept the default file system size.

7. The system prompts for the file system name:

```
file system name:
```

8. Name the root file system. If you want to name it "root" (/), enter /; if you prefer another name, it must be six or fewer characters in length. (For more on file system names, refer to the **labelit**(1M) command in the *ARIX OS90 Release 3.3 System Administrator's Reference Manual.*)

   After naming the root file system, press «Return». The system displays:

```
volume name:
```

9. Name the root file system volume. If you want to name it os3.3, enter that name; if you prefer another name, it must be six or fewer characters in length.

   When you press «Return», the system displays messages similar to:

```
fsize=xxxxx
isize=xxxx
interlace x, sectors/cylinder xxx
```

   After a pause, the system displays:

```
Verbose mode?
```

   Verbose mode scrolls the names of the files on the root file system to the screen as they are loaded onto the disk. If you do not choose verbose mode, the screen remains as is until all files are loaded to the disk.

10. For verbose mode, enter y as the root file system loads; otherwise, enter **n**.

NOTE:

It takes approximately 5-15 minutes to load the files to the disk.

When you press «Return», the system displays:

```
Type `?` for help
Runtime Boot Level Menu:
```

The root file system is now loaded onto the computer.

11. Boot the UNIX kernel. At the `Runtime Boot Level Menu:` prompt, enter:

    boot *X***dZs***N***arix**

    where:

    X represents the CSS slot number of the IOPM/DSDB being accessed

    dZ represents the SCSI ID number of the drive relative to *X*.

    s*N* represents the logical slice the root file system was loaded on relative to **dZ**.

    arix is the default name of the UNIX kernel.

    The system displays something similar to the following:

```
.....
Loading from 8d0s0arix
download completed.
PM in slot # booted.
PM in slot # booted.
IOPM x present but not loaded.
IOPM x present but not loaded
IOPM x in slot # loaded.

PM 2: ARIX-OS/S90 V.3 Release POS3.3_x, Version xxxx
PM 2: (C) 1988 ARIX Corporation
PM 2: Node syst, System S90_V.3
IOPMx = 0x#### available memory
PM 2: Total real memory  = xxxxxxxx
PM 2: Available memory   = xxxxxxxx

INIT: SINGLE USER MODE
#
```

NOTE:

If you see any of the following messages, ignore them:

```
fsstat: root file system needs checking
the root file system is being checked automatically
```

(followed by output from the fsck program)

```
*** ROOT FILE SYSTEM WAS MODIFIED ***
*** ROOT FILE SYSTEM WAS REMOUNTED **
```

---

12. To access the slices on your disk drives, you must make device nodes for each slice. The **mkdsk** program is used to add new disk devices to an already-configured system, or to configure the disk drives in a new system for the first time. Disk drives are always attached to a Dual SCSI Device Board (DSDB) in your System90/x5 computer; all System90/x5 computers come with at least one DSDB.

    If necessary, use the **mkdsk** program to make nodes for any Dual SCSI Device Board(s) in the system. At the # prompt, enter:

    /local/bin/mkdsk

13. **mkdsk** displays:

    ```
    To which IOPM will this disk drive be attached (0-5)?
    ```

    Enter the number of the IOPM/DSDB for which you want to create nodes. IOPM/DSDBs are numbered as follows: first, IOPM/DSDBs in the main cabinet are numbered from right to left. Next, for System90/85 systems only, IOPM/DSDBs in the first expansion cabinet (that is, the expansion cabinet connected to the rightmost IOM in the main cabinet) are numbered from right to left. This method continues to the second expansion cabinet, and so forth.

14. **mkdsk** displays:

    ```
    Which SCSI id number on IOPM x?
    ```

    Enter the proper identifier for the disk attached to the DSDB.

15. **mkdsk** displays:

    ```
    How many logical slices should be created for this drive?
    ```

    You can create up to 16 logical slices. When you respond to this prompt, **mkdsk** makes the nodes for the disk attached to the first DSDB; when finished, **mkdsk** terminates and the system redisplays the # prompt.

16. Repeat Steps 12 through 14 for every disk in your system. When you have created nodes for every disk in your system, move on to Step 17 below.

17. Next you need to create devices for your system's terminals. The **mktty** program is used to add new tty devices to an already-configured system, or to find and correct any missing or incorrectly installed tty devices. tty devices are attached to the ACRW (System90/25 only) or ACDB (System90/25, System90/45, and System90/85).

    If you are installing ARIX OS90 on a System90/25 computer, use the **mktty** program to make nodes for your system's ACRW board. At the # prompt, enter:

    /local/bin/mktty

18. **mktty** displays:

```
What kind of board do you want to create nodes for:
SPM/ACRW (S), or IOPM/ACDB (I) ?
```

Enter S to make nodes for an ACRW board.

19. **mktty** displays:

```
What starting tty number would you like to use
(default 0) ?
```

Under most conditions, it is recommended that you enter **0** at this prompt.

20. **mktty** displays:

```
How many tty devices would you like to create
(default 16) ?
```

Look in back of the system cabinet to determine the number of ports you want to create; an ACRW board can support up to 56 ttys.

mktty makes the nodes for the ACRW; when finished **mktty** terminates and the system redisplays the # prompt.

NOTE:

By default, eight tty devices already exist in the system. When you run **mktty**, the program asks if you want to delete these devices and create new ones. Enter **y** for yes.

---

21. If necessary, use the **mktty** program to make nodes for each ACDB board(s) in the system. At the # prompt, enter:

/local/bin/mktty

22. **mktty** displays:

```
What kind of board do you want to create nodes for:
SPM/ACRW (S), or IOPM/ACDB (I) ?
```

Enter I to make nodes for an ACDB board.

23. **mktty** displays:

```
Which IOPM/ACDB are these devices for (0-3) ?
```

Enter numeral **0** for the first ACDB in the system; if your system has more than one ACDB, you can create the rest of the nodes later.

24. **mktty** displays:

```
What starting tty number would you like to use
(default 0) ?
```

The number you start with depends on how many ttys you configured on ACRW boards in Steps 16 through 19. For example, if you created eight devices for ACRW boards, they are numbered from 0 to 7; start numbering ACDB ports at tty 8.

25. **mktty** displays:

```
How many tty devices would you like to create
(default 16) ?
```

Look in back of the system cabinet to determine how many tty ports you have on this ACDB board; each ACDB can support up to 64 ttys.

mktty makes the nodes for the first ACDB; when finished **mktty** terminates and the system redisplays the # prompt.

NOTE:

By default, eight tty devices already exist in the system. When you run mktty, the program asks if you want to delete these devices and create new ones. Enter y for yes.

---

26. Repeat Steps 17 through 26 for every ACDB in your system. When you have created nodes for every ACDB in your system, move on to Step 27 below.

27. If necessary, use the **mkrmt** program to create the correct device nodes for your system's tape drive.

    If your tape drive is configured to use a SCSI ID number other than the default of 3, you must run **/local/bin/mkrmt** to create the correct device nodes for the tape drive.

28. Initialize the */etc/mnttab* file: at the # prompt, enter:

    devnm / | setmnt

29. The **fsck**(1M) program checks the file systems for any discrepancies. See the *File System Administration* chapter in the *ARIX OS90 Release 3.3 System Administrator's Guide* for more information on fsck.

Perform a file system check on the root slice of the disk drive by entering the **fsck** command and target disk drive device address at the # prompt:

```
sync
fsck  /dev/dsk/cXdYsZ
```

where *X* represents the relative disk controller, *Y* represents the SCSI ID number of the disk, and *Z* represents the number of the disk slice where the copy of the kernel is stored (this is the root slice).

The system displays messages similar to the following:

```
/dev/dsk/cXdYsZ
/dev/dsk/cXdYsZ  File System: xxxx  Volume: xxxx

/dev/dsk/cXdYsZ  ** Phase 1 - Check Blocks and Sizes
/dev/dsk/cXdYsZ  ** Phase 2 - Check Pathnames
/dev/dsk/cXdYsZ  ** Phase 3 - Check Connectivity
/dev/dsk/cXdYsZ  ** Phase 4 - Check Reference Counts
/dev/dsk/cXdYsZ  ** Phase 5 - Check Free List
/dev/dsk/cXdYsZ  xxx files xxxx blocks XXXXX free
#
```

If there are no error messages from fsck, proceed to the next section in the installation procedure, *Loading the /usr File System.*

If there are error messages from **fsck**, the system displays a message for each error as it is encountered and asks you to respond. Enter the appropriate response, referring to the *File System Administration* chapter in the *ARIX OS90 Release 3.3 System Administrator's Guide* if necessary. The final message the system displays is usually:

```
*** ROOT FILE SYSTEM WAS MODIFIED ***
```

Do the following:

1. Return to the `Runtime Boot Level Menu:` prompt; enter:

   «Control-t»

CAUTION:

The «Control-t» character sequence has special significance to the System90. When you enter this sequence from the console, the SPM Runtime Monitor can be accessed directly from both single-user and multi-user modes.

Many actions can be performed at the `Runtime Boot Level Menu:` prompt that could disrupt system activity. For example, the entire system could be reset. Use extreme caution when accessing the Runtime Boot Level Menu, especially if you are in multi-user mode, so that other users are not affected. You can use the **con** command to return to UNIX mode.

---

2. Reset the system. At the `Runtime Boot Level Menu:` prompt, enter:

   exit

   Before the system resets, reboots, and returns to single-user mode, it displays:

   ```
   Do you really want to exit?
   Type 'y' or 'n'
   ```

3. Reload the bootimage and root file system. Carefully repeat all of the procedures in this section to reload the bootimage and the root file system. If errors persist, call your ARIX representative.

When there are no errors, the root file system is properly loaded, and you can load the */usr* file system.

Proceed to the next section, *Loading the /usr File System.*

Loading the /usr File System

This procedure describes making and loading the */usr* file system on your computer.

1. Create the filesystem for */usr*: the recommended location for the */usr* file system is slice 2 on the same disk as the root file system.

   a. If you want to place the */usr* file system on the disk slice c0d0s2 (as recommended), enter the following command at the # prompt:

      mkfs /dev/rdsk/c0d0s2

      The system prompts with messages similar to this:

```
mkfs: /dev/rdsk/cXdYsZ
creating ARIX 1k filesystem
bytes per logical block = xxxx
total logical blocks = xxxx
total inodes = xxxx
gap (physical blocks) = x
cylinder size (physical blocks) = xxx
(Press DEL if wrong)
Making file system /dev/rdsk/cXdYsZ
```

      After a pause, the system displays the # prompt.

      After you invoke the **mkfs** command, the program echoes the device name you entered and pauses. If you entered an incorrect device name, press «Delete» within 10 seconds. This interrupts the program, enabling you to re-enter the device name.

   b. If you do not intend to put the */usr* file system on slice c0d0s2 (or if you did not put the root file system on c0d0s0), you must modify the */etc/fstab* default file. Enter the following commands:

      echo /dev/dsk/c*X*d*Y*s*Z* /usr > /etc/fstab

      echo /dev/rdsk/c*X*d*Y*s*Z* > /etc/checklist (*usr slice*)

      echo /dev/dsk/c*X*d*Y*s*Z* >> /etc/checklist (*root slice*)

      where *X*, *Y*, and *Z* are the controller, disk, and slice, respectively, on which you are putting the root and */usr* file system.

      These commands are appropriate only for the minimum configuration of slices; see the *ARIX OS90 Release 3.3 System Administrator's Guide* for full descriptions of the */etc/fstab* and */etc/checklist* files.

2. Name the */usr* file system by entering the **labelit** command in the format:

   labelit /dev/rdsk/cXdYs2 /usr *file_system_name*

   where **cXdY** is the device address of the disk for your system, slice 2 is the slice where the */usr* file system is stored, and *file_system_name* is a name for the */usr* file system, such as **os3.3**.

   (If you want to enter a different name, use six characters or less.)

   The system displays a message showing the new file system name and new volume name; if you change your mind about the names you just entered, press «Delete» now and **labelit** erases them.

3. Mount */usr* by entering:

   mount /usr

4. Verify that the tape labeled *Bootimage Root Usr Man* is still in the tape drive, then move to the */usr* directory by entering:

   cd /usr

5. Position the tape in front of the */usr* file system by entering:

   < /dev/rmt1 (*rewind the tape*)
   < /dev/rmt0 (*seek past the bootimage*)
   < /dev/rmt0 (*seek past the root file system*)

   The tape should advance to the proper position within two minutes. The system redisplays the # prompt.

6. Load the */usr* file system by entering:

   cpio -idvBmu < /dev/rmt0

   The system displays the names of the files on the */usr* file system as they are loaded to the disk. This process takes about 15 minutes.

   The last four lines to be displayed during this process are similar to:

```
sys/vuifile
tmp
xxxxx blocks
#
```

   The names of the last two files on the tape may differ from the above. The tape is now in position for the next procedure, *Loading the Manual Pages*; do not remove the tape from the drive.

7. Set the permissions on the */usr* file system by entering the command:

   chmod 755 /usr

8. Initialize the *lost+found* directories by entering the commands:

```
/local/bin/lost+found   /usr
/local/bin/lost+found   /
```

9. Verify that the */usr* file system loaded properly by entering the following commands:

```
sync
cd /
umount /usr
fsck /dev/rdsk/cXdYsZ
```

where **cXdYsZ** is the device address of the disk for your system.

The system displays messages similar to:

```
/dev/dsk/cXdYs2
/dev/dsk/cXdYsZ  File System: xxx  Volume: xxxx

/dev/dsk/cXdYsZ  ** Phase 1 - Check Blocks and Sizes
/dev/dsk/cXdYsZ  ** Phase 2 - Check Pathnames
/dev/dsk/cXdYsZ  ** Phase 3 - Check Connectivity
/dev/dsk/cXdYsZ  ** Phase 4 - Check Reference Counts
/dev/dsk/cXdYsZ  ** Phase 5 - Check Free List
/dev/dsk/cXdYsZ  xxx files xxxx blocks XXXXX free
#
```

When there are no errors, the */usr* file system is installed, and the system is ready for you to remount the */usr* file system.

Leave the tape labeled *Bootimage Root Usr Man* in the tape drive, and proceed to the next section, *Loading the Manual Pages*.

Loading the Manual Pages

The on-line manual pages are stored in the */usr/catman* directory. Follow the steps below to load the */usr/catman* directory:

1. Ensure that the tape labeled *Bootimage Root Usr Man* is still in the tape drive.

2. Remount the */usr* file system. Enter:

   mount /usr

3. Make sure you are in the */usr* directory. Enter:

   cd /usr

4. Load the manual pages into the */usr/catman* directory.

NOTE:

If for any reason you removed the tape after loading the */usr* file system, or if you are loading the manual pages at a different time from the rest of the OS90 distribution, use the following commands to ensure that the tape is properly positioned before proceeding with step 4:

< /dev/rmt1 (*rewind the tape*)
< /dev/rmt0 (*seek past the bootimage*)
< /dev/rmt0 (*seek past the root file system*)
< /dev/rmt0 (*seek past the /usr file system*)

These commands should position the tape within six to seven minutes.

Enter:

cpio -idvBmu < /dev/rmt1

The system displays the names of the files in the */usr/catman* directory as they are loaded to disk. This process takes five to fifteen minutes.

The last four lines to be displayed are similar to:

```
catman/p_man/man3/sputl.z
catman/p_man/man3/sgetl.z
xxxxxx blocks
#
```

The names of the last two files on the tape may differ from the above.

For U.S. installations, proceed to the next section, *Loading the Domestic Overlay*. For other installations, the installation is now complete. Remove the *Bootimage Root Usr Man* tape from the tape drive and turn to Chapter 4.

Loading the Domestic Overlay (U.S. Installations Only)

The U.S. version of OS90 Release 3.3 requires you to load the Domestic Overlay to complete your Operating System installation.

Follow these steps:

1. Shut the system down to single-user mode, if it is not currently in this mode.

2. Remove the *Bootimage Root Usr Man* tape from the tape drive and insert the tape labeled *Domestic Overlay.*

3. Extract the information from the tape onto the root and */usr* file systems: at the # prompt, enter the following command:

   # sysadm

   The system responds with:

```
1 diskmgmt       disk management menu
2 filemgmt       file management menu
3 machinemgmt    machine management menu
4 packagemgmt    package management menu
5 softwaremgmt   software management menu
7 syssetup       system setup menu
8 tapemgmt       tape management menu
9 usermgmt       user management menu

Enter a number, a name, the initial part of the name, or
? or ,number>? for HELP, q to QUIT:
```

4. Choose the software management menu by typing:

   5

   The system responds with:

```
1 installpkg  install new software package onto built-in disk
2 listpkg     list packages already installed
3 removepkg   remove previously installed package from built-in disk

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP, ^ to GO BACK, q to QUIT:
```

5. Choose installpkg by typing:

   1

   The system responds with:

   ```
   Are you installing a new package, or do you want to re-install
   a package previously removed by removepkg [i, r, q]:
   ```

6. Choose install new package by typing:

   i

   The system copies the files on the *Domestic Overlay* tape onto the disk; the names of the files scroll down the screen as they are copied.

7. Type the following to ~~exist~~from the system administration menus:

   q exit

8. Unmount the */usr* file system by entering the following commands:

   sync
   cd /
   umount /usr

9. Use the **fsck** command to check the root and */usr* file systems. Use the format:

   fsck /dev/dsk/c*X*d*Y*s*Z* /dev/rdsk/c*X*d*Y*s*Z*
   (*root filesystem slice*) (*/usr filesystem slice*)

   For example, if you have installed the operating system onto the default disk slices of */dev/dsk/c0d0s0* for the root filesystem and */dev/rdsk/c0d0s2* for the */usr* filesystem, enter the **fsck** command at the # prompt as follows:

   fsck /dev/dsk/c0d0s0 /dev/rdsk/c0d0s2

In this case, the system displays messages similar to:

```
/dev/dsk/c0d0s0
/dev/dsk/c0d0s0  File System: xxx  Volume: xxx

/dev/dsk/c0d0s0  ** Phase 1 - Check Blocks and Sizes
/dev/dsk/c0d0s0  ** Phase 2 - Check Pathnames
/dev/dsk/c0d0s0  ** Phase 3 - Check Connectivity
/dev/dsk/c0d0s0  ** Phase 4 - Check Reference Counts
/dev/dsk/c0d0s0  ** Phase 5 - Check Free List
/dev/dsk/c0d0s0  xxx files xxxx blocks XXXXX free
/dev/dsk/c0d0s2
/dev/dsk/c0d0s2  /dev/dsk/c0d0s2

/dev/dsk/c0d0s2  File System:  Volume:
/dev/dsk/c0d0s2  ** Phase 1 - Check Blocks and Sizes
/dev/dsk/c0d0s2  ** Phase 2 - Check Pathnames
/dev/dsk/c0d0s2  ** Phase 3 - Check Connectivity
/dev/dsk/c0d0s2  ** Phase 4 - Check Reference Counts
/dev/dsk/c0d0s2  ** Phase 5 - Check Free List
/dev/dsk/c0d0s2  xxx files xxxx blocks XXXXX free
#
```

When there are no errors, the Domestic Overlay files are installed, and the system is ready for you to remount the */usr* file system.

The installation is now complete. Remove the *Domestic Overlay* tape from the tape drive and turn to Chapter 4.

4 After the Installation

This chapter covers system administrative procedures you can use immediately following the installation or upgrading of OS90 Release 3.3. These procedures are included in this guide to help you get your new operating system ready for end users as quickly as possible.

Creating lost+found Directories

If you add file systems to your system in addition to root and */usr* (which are required), you must create a *lost+found* directory for each mountable file system you add.

NOTE:

The *lost+found* directories for root and */usr* are created during the regular installation process.

The **fsck** utility checks for discrepancies in the file system. When fsck finds a lost file, it places the file in the *lost+found* directory. An appropriately sized *lost+found* directory must be created and placed in the top directory of each mounted file system, including the root and */usr* file systems.

Create *lost+found* directories and place them in each of your mounted file systems:

1. If you have not already done so, make the new file systems.

   Refer to **mkfs**(1M) in the *ARIX OS90 Release 3.3 System Administrator's Reference Manual.*

2. Mount the new file systems.

3. Run the lost+found script for each newly-created file system. For each file system, enter the lost+found command in the format:

   /local/bin/lost+found *directoryname*

   where *directoryname* is the mount point for that particular file system.

   For more on running the script, refer to **lost+found**(1M) in the *ARIX OS90 Release 3.3 System Administrator's Reference Manual.*

Bringing Up the System in Multi-User Mode

Once you have created *lost+found* directories for all your mounted file systems and created support for all ACDB modules in the system, bring the System90 from single-user to multi-user state.

Do the following:

1. At the # prompt, enter:

   init 2

   The system displays messages similar to:

   ```
   INIT: New run level: 2
   Is the date xxx xxx x xx:xx:xx xxx 19xx correct? (y or n)
   ```

2. Enter **y** to accept the date; **n** to correct the date.

   If you enter **n**, the system displays:

   ```
   Enter the correct date:
   ```

   Enter the correct date in the format:

   MMDDHHMMYY

   where the starting *MM* represents the number of the month, the *DD* is the number of the day, the *HH* is the hour (in 24-hour time), the second *MM* is the minutes, and the *YY* is the last two digits of the year. All fields must be two-digit numbers.

   (For example, 6:30 a.m. on February 8, 1991 would be expressed as **0208063091.**)

   The system displays:

   ```
   Is the date xxx xxx x xx:xx:xx xxx 19xx correct? (y or n)
   ```

   Again, you can either correct the date by entering **n** or accept the date by entering **y**.

   After you enter **y**, the system displays:

   ```
   This machine has not been used in multi-user mode yet.
   If all the file systems have not been checked with fsck,
   they should be.  Do you want to check the file systems
   ("y" or "n") ?
   ```

3. It is important to check the file systems before bringing the system on-line if it has been taken down for any reason. Check the file systems by entering **y**; otherwise, enter **n**.

   If you enter **y**, the system runs fsck and displays messages similar to the following:

```
/dev/rdsk/c0d0s2
/dev/rdsk/c0d0s2    File System: xxx Volume: xxx

/dev/rdsk/c0d0s2    ** Phase 1 - Check Blocks and Sizes
/dev/rdsk/c0d0s2    ** Phase 2 - Check Pathnames
/dev/rdsk/c0d0s2    ** Phase 3 - Check Connectivity
/dev/rdsk/c0d0s2    ** Phase 4 - Check Reference Counts
/dev/rdsk/c0d0s2    ** Phase 5 - Check Free List
/dev/rdsk/c0d0s2    xxx files xxxx blocks XXXXX free

/dev/dsk/c0d0s0
/dev/dsk/c0d0s0    File System: xxx Volume:xxx

/dev/dsk/c0d0s0    ** Phase 1 - Check Blocks and Sizes
/dev/dsk/c0d0s0    ** Phase 2 - Check Pathnames
/dev/dsk/c0d0s0    ** Phase 3 - Check Connectivity
/dev/dsk/c0d0s0    ** Phase 4 - Check Reference Counts
/dev/dsk/c0d0s0    ** Phase 5 - Check Free List
/dev/dsk/c0d0s0    xxx files xxxx blocks XXXXX free
.
.
.
This machine has to be set up by you.  When you get to
the login prompt, log in as "setup".  This will start a
procedure that leads you through those things that should
be done the "first time" the machine is used in multi-
user mode.

mount -f SEC1k /dev/dsk/cXdYsZ /usr
/etc/rc2.d/S75cron: /etc/cron started
/etc/rc2.d/S80error: errdaemon started
Starting LP Scheduler
Print Services Started
The system is ready.

syst 9600 console login:
```

4. Log into the system console as **setup**.

   Logging in as setup starts the **setup** program. This program is generally only run the first time a system is moved into multi-user mode; it lets you:

   - Set the current time and time zone.
   - Set the current date and time.
   - Set up logins for individual end-users on the system.
   - Set passwords for administrative logins such as setup, sysadm, checkfsys, makefsys, mountfsys, and umountfsys.

- Set passwords for system logins such as root, daemon, bin, sys, adm, uucp, nuucp, lp, and listen.
- Change the name of the machine.

Follow the setup program instructions, filling in the requested information at each prompt. Before setup displays the next prompt, it re-displays the information you just entered; to confirm this information, type **y**, to change the information, type **n**; to quit the program without saving any changes, type **q**.

NOTE:

Although password protection for administrative and system logins is strictly optional, it is strongly recommended that you specify password protection for all of the above-mentioned sensitive logins.

When you have finished specifying all the necessary setup information, setup records the new system parameters and identifications and displays:

```
System configuration complete.
```

Now the system is completely initialized, loaded, and ready to operate in a multi-user state.

To load other communications or applications software, see the Installation Guides for those software products.

Release Notes A

This chapter provides a section of technical tips and notes for installing and using OS90 Release 3.3 on your new ARIX System90/x5 computer. These notes include information on new and enhanced features in OS90 Release 3.3.

The information in this chapter is intended for use by the System Administrator or System Operator only; in the majority of cases, the tips and workarounds discussed here do not affect the average end user.

New Features for This Release

- (68040 version only) Support for the MC68040 CPU has been added in Release 3.3. Release 3.3 continues to run all non-OS dependent System90 binaries.

NOTE:

Series 800 binaries are not supported by the 68040 version of OS90 Release 3.3.

NOTE:

ARIX has made every effort to maintain compatibility between OS90 Release 3.2 and OS90 Release 3.3. However, this was not possible in a few areas as discussed in these release notes. To ensure maximum functionality of your new operating system, ARIX recommends recompiling applications from source. See the section *Recompiling Applications for the 68040 Processor*, this chapter, for details.

- Release 3.3 processes can have as many open files as necessary. Two new tunable parameters allow you to set the default open file limits, and two new system calls allow processes to check and set the limits.
- An automatic STREAMS push of the **ldterm** module onto the tty driver has been added.

- Support for creating and using symbolic links has been added. The utilities that operate on files and file systems have been modified to handle symbolic links. New system calls are also provided to work with symbolic links.

- The Release 3.3 **iopmfmt**(1M) command has been enhanced to include a simpler user interface to the disk slicing functions and a disktest capability to facilitate identification and reassignment of bad blocks.

- Support for bi-directional modems has been added to the kernel.

- Quad-density nine-track tape drives are now fully supported.

- OS90 Release 3.3 can be booted from expansion cabinets, as well as from the main system cabinet.

- The **apropos**(1) command has been added, allowing users to search for a word or phrase in the online manual page database. Users can search on any keyword in the NAME line of the manual page. All manual pages containing that keyword are listed on the screen.

- The **mfsck**(1M) utility has been added to allow checking of multiple filesystems in parallel.

- Machine-specific code can be compiled for the 68020, 68030 or 68040 CPU regardless of the CPU in the compiling system.

- Filesystems using 64-byte inodes can now be mounted read/write.

- The */etc/checklist* file now accepts comments.

- In Release 3.3, the **cpio** utility has been enhanced to include support for Access Control Lists (ACLs). The new ACL option **-Z** can be used with both the **-o** option (when creating cpio archives) and the **-i** option (when restoring cpio archives). See **cpio**(1) in the *ARIX OS90 Release 3.3 User's Reference Manual* and the Trusted OS90 documentation for further information.

- A new **-E** option has been added to **getty**(1M) to allow environment variables to be passed to users at login.

See the *Technical Notes* section below for more detail about new features and enhancements in Release 3.3.

NOTE:

Information concerning the LP system forms capability will be provided in the production release version of this documentation.

Technical Tips

Recompiling Applications for the 68040 Processor

Under certain circumstances applications must be recompiled for the 68040 version of OS90 Release 3.3:

- The **trap2()** system call is not available on the 68040 version of OS90. Binaries that use the **trap2()** system call for test and set get an illegal instruction error on the 68040 version.

NOTE:

The **trap 2** emulation will not be available for the 68020 in future OS90 releases.

- Binaries that depend on the **tas()** system call, or use the instruction **cas** or other atomic 680X0 instructions to set locks, must also use atomic instructions to clear these locks. Any applications that do not clear locks of this type with atomic instructions may experience failures due to race conditions during an unlock operation. Any such applications must be recompiled to use atomic instructions to perform unlocks as well as locks.

  This recompilation is necessitated by differences in the 68040 CPU's implementation of the **cas** instructions, as opposed to the 68020 chip.

- Binaries that use certain */usr/include/sys* files (such as *user.h*, *proc.h*, *own.h*, *lio.h*, and *kmem.h*) must be recompiled. Because there are differences in various system structures and addresses, software that is dependent on kernel-specific structures, such as debuggers and kernel drivers, must be recompiled.

NOTE:

ARIX has made every effort to maintain compatibility between OS90 Release 3.2 and OS90 Release 3.3. However, this was not possible in a few areas as discussed in these release notes. To ensure maximum functionality of your new operating system, ARIX recommends recompiling applications from source. See the section *Recompiling Applications for the 68040 Processor* for details.

Transitioning Between SVID and POSIX Environments

A user's environment (System V or POSIX-compliant System V) is determined by the gcos field of the */etc/passwd* file entry.

If the gcos field setting is ...	The user's environment is ...
SVID (basic System V calls)	SVID, no job control
NOJOBS	POSIX, no job control
empty (neither of the above)	POSIX with job control

For example:

```
myron:vQ2LD5gnPPKCk:100:1:Mr Draw SVID:/:/bin/ksh
bertrand:vQ2LD5gnPPKCk:101:1:Mr Tennis NOJOBS:/:/bin/ksh
sanjoy:vQ2LD5gnPPKCk:101:1:Mr X.25:/:/bin/ksh
```

In the above segment of this sample */etc/passwd* file, myron's environment is SVID with no job control, bertrand's environment is POSIX with no job control, and sanjoy's environment is POSIX with job control.

To determine which environment the user is in, echo the OS environment variable. The system displays either `POSIX` or `SVID`. For example:

```
$ echo $OS «Return»
  POSIX
$
```

Note that changing the OS variable does not change your POSIX/SVID status.

Environment Versus Binary Type

The following table describes the interaction between the SVID and POSIX environments and binary files created by either standard SVID compilers or POSIX compilers.

Feature	Environment/Binary			
	POSIX/ POSIX	POSIX/ SVID	SVID/ POSIX	SVID/ SVID
Job Control	yes	yes	no	no
Supplementary Group Check	yes	yes	no	no
POSIX errno	yes	no	yes	no
POSIX gid inherit	yes	yes	no	no
POSIX pipe I/O	yes	no	yes	no

NOTE:

Although POSIX-created binaries can handle straight SVID environments, SVID-created binaries do not function correctly in POSIX environments.

GID Inheritance

The following table describes gid inheritance on a created file.

Directory File	POSIX Environment	SVID Environment
no sgid bit set	dir gid	proc gid
sgid bit set	proc gid	proc uid

Printer Support Notes

Printer Configuration: General

To configure a direct-connect printer, either serial or parallel, you must first "push" the **lpmod** STREAMS module. One way to accomplish this is as follows:

/local/bin/openlp /dev/lp 9600

In the command above, **/dev/lp** is a link to a printer device on an IOPM/ACDB or an ACRW. In the case of a parallel printer, the **9600** option is ignored.

Printer Configuration: System90/25 Only

In order to use **setlp** with a printer connected to the SPM/ACRW in a System90/25, the **lpmod** STREAMS module must first be configured into the kernel.

Create a master file in */etc/master.d* called **lpmod**. It should look like this:

```
lpmod -- streams printer module

%STRMODULE
Prefix:  lpmod              # prefix added to functions
%PARAM
NUM LPMOD 16                # number of lpmods
```

Next, add this entry to */etc/system* in the STREAMS modules section:

```
lpmod 1
```

Now rebuild the kernel and follow the instructions for using **openit** and **stty** with the correct device for the SPM/ACRW printer.

Running Shell Scripts in the Background from sysinit

Do not run shell scripts in the background from **init**'s sysinit state, or allow any sub-shell within such scripts to run in the background. A shell script or sub-shell that is run in the background from sysinit executes the specified I/O redirection, but it does not run any commands. When run in the sysinit state, a parent process is killed after spawning a child process in the background, if this is its only task; as a result, the child process is also killed before it can execute.

Whenever the system is booted, **init**(1M) reads */etc/inittab* for lines with the **action** sysinit. (These commands may be binary programs or shell scripts.) It executes the commands in these lines in the order that they appear in */etc/inittab*, before doing anything else—including displaying the `INIT: SINGLE USER MODE:` prompt.

When the shell runs a command "in the background," the shell does not wait for the command to finish before continuing on to the next task. The command and the shell run in parallel until the command finishes.

If you cannot run the shell scripts at a later time in the initialization process, you may be able to work around this situation by having the parent process perform some additional steps or sleep long enough to allow the child process to complete its background processing.

To run a shell script in the background, use the following syntax:

command [*optional_arguments*] [*optional_I/O_redirection*] &

To run a sub-shell script in the background, use the following syntax:

(*command* ... [*optional_further_commands*]) &

Using /local/bin/iopmfmt

When used under OS90, the **iopmfmt** command (*/local/bin/iopmfmt*) requires you to specify the SCSI pass-through device in the device specification. For example:

/local/bin/iopmfmt /dev/scsi_pt/c0d0

This command can be used only if there are no mounted file systems on the device you are trying to access. If the device has a mounted file system, the following error message is displayed:

```
/dev/scsi_pt/c0d0: device open error!
Invalid argument
```

Technical Notes

This section offers some additional information about ARIX OS90 Release 3.3. Notes about commands, system calls, files, and other system features are listed alphabetically. If you encounter any further problems, contact ARIX Customer Support as described in Chapter 1 of this guide and refer to the reference number following the note.

acdbsetup script (reference # 879)

A new script, **/etc/init.d/acdbsetup**, has been added to more fully automate the process of bringing up the system for the first time. This script is run the first time the system enters multiuser state to create the IOPM load scripts for the ACDBs and also create the nodes for the ttys.

ACRW I/O (reference # 258)

Lines 255 characters in length can now be read from an ACRW tty port in canonical mode. In earlier releases, the longest line that could be read was 254 characters.

apropos(1) command (reference # 436, 653, 660)

The **apropos** manual page lookup capability has been added. This command allows you to search the online manual page database for a specified word or phrase. See the **apropos**(1) manual page for details.

as(1) assembler (reference # 480)

The **–r** option to the assembler has been removed. To create read-only data use the **const** qualifier in data declaration, or use the more general **–Xnostrwrite** flag to the C compiler. Read-only data allows programs to run with more information shared between processes, thus reducing per-process overhead. For more information refer to the **cc**(1) manual page.

as(1) assembler (reference # 505)

For the 4D50D release of the 68040 chip, an NOP must be placed prior to a single or sequence of MOV16 instructions.

bad block devices (reference # 563)

A bad block special device (a major number for which no IOPM driver exists) no longer causes an ASSERT error message to be displayed. In Release 3.3, the IOPM instead returns the ENODEV error.

baud rate (reference # 845)

The kernel no longer allows the baud rate to be set to values that are not supported by the ACRW board (specifically, 200 baud and 38,400 baud).

bi-directional modems (reference # 205, 207)

Support has been added in the kernel for bi-directional modems. In earlier releases the **cu**(1) command would time out when **uugetty** had an open pending on a line connected via a bidirectional modem. A new ioctl call set has been added to allow multiple pending opens in the stream head. See the **termio**(7) manual page for more information.

boot (reference # 510, 672, 777)

It is now possible to boot from an IOPM/DSDB in an expansion cabinet. As a result it is no longer necessary to have an IOPM/DSDB in the main cabinet, making room for more main processor or memory modules.

bootimage (reference # 846)

The kernel default boot path is no longer reset every time the SPM path is changed. The kernel boot path is no longer initialized to non-IOPM/DSDB slots.

bootimage (reference # 1045)

The diagnostic image now boots on 68040 systems. However, its functionality is restricted and its use is not recommended.

bootimage (reference # 1197)

The command **ls -l** now works correctly as a standalone command in SPM runtime mode.

bootimage (reference # 1230)

Standalone **mkfs** how works correctly on very large file systems (larger than 60,000 blocks) and does not produce the error message `can't find indirect block`. In addition, standalone **mkfs** now updates the screen display periodically while processing.

booting from the /usr slice (reference # 1004, 1042)

An empty file */usr/etc/bcheckrc* has been added to eliminate init error messages when booting from */usr*. The file */usr/bin/tty* has also been added to eliminate error messages when booting from */usr*.

NOTE:

A system administrator does not normally need to boot from */usr*. This capability is provided in case the root file system becomes unreadable.

---

cc(1) compiler (reference # 32)

Optimization while using the **-b** option now properly handles fcmp to ftest conversion.

cc(1) compiler (reference # 292)

String literals are now writable by default. The **-Xnostrwrite** option to cc can be used to put string literals in the text segment to save space in runtime images.

cc(1) compiler (reference # 315)

Bitfields are now aligned on 32-bit boundaries. Previously, in the case where the contents of a pointer were being assigned to a bitfield which crossed a 32-bit boundary, the data in the second word was lost.

cc(1) compiler (reference # 320)

An error occurs in handling of the use of exclusive OR in a function call. The line:

```
func( str().a ^ str().b )
```

could fail with the return of the first call to *str* being overwritten by the return from the second call to *str*.

cc(1) compiler (reference # 407)

Optimization of second and subsequent assignments to a conv node of a constant has been removed. Code of the following form failed to initialize variable j, but is now optimized correctly:

```
short i, j;
int k:

i = 0;

k = j = i;
```

cc(1) compiler (reference # 417)

Information for uninitialized global data is now generated correctly and works correctly with the **sdb**(1) debugger. The compiler in earlier releases was not generating the correct information.

cc(1) compiler (reference # 496)

A new **–T** option has been added to allow building of machine-specific code on any CPU type. Arguments to **–T** are as follows:

- **–T2** – use the 68020 version of **as**(1), **cpp**(1), and the optimizer.
- **–T3** – use the 68030 version of **as**(1), **cpp**(1), and the optimizer.
- **–T4** – use the 68040 version of **as**(1), **cpp**(1), and the optimizer.

The default is to use **/bin/as**, **/lib/cpp**, and **/lib/optim**. See the cc(1) manual page for more information.

cc(1) compiler (reference # 499)

The *libPW* function **alloca()** depends on the use of **link** and **unlk** in the calling routine. The current optimizer removes these instructions in addition to the use of the frame pointer. With this in mind, modules that contain a call to **alloca()** must not be optimized. See the **–b** option in the cc(1) manual page for more information.

cc(1) compiler (reference # 617)

Exception processing for floating point exceptions under the ANSI or POSIX mode of the C compiler has been changed to conform to IEEE as it does under the **–M0/–SYSV** mode of the C compiler. Returns from **math** calls such as **log()** should always be checked for validity using a test of **errno**.

cc(1) compiler (reference # 786, 792, 836)

Typecasting an object with its own type no longer produces an error.

cc(1) compiler (reference # 787)

Mixing System V and ANSI prototyping no longer produces wrong type clashes.

cc(1) compiler (reference # 788)

Initialization and brackets are now handled correctly. The **const** qualifier is now propogated correctly throughout structure initialization as specified by the ANSI standard.

cc(1) compiler (reference # 828)

Profiling (the **–p** option to **cc**) is now supported correctly in both ANSI and non-ANSI modes.

cc(1) compiler (reference # 1065)

A warning message is now displayed if no input files are supplied on the command line.

cc(1) compiler (reference # 1115)

Identifiers in function prototypes no longer cause the compiler to display a `Bad bigsize` error message.

cc(1) compiler (reference # 1175)

The **–f** flag is now provided to support floating-point emulation. This flag instructs **cc** to use the emulated versions of the math libraries.

chgrp(1) command (reference # 240)

The new option **–h** has been added so that symbolic links can be followed. See the **chown**(1) manual page for more information about **chgrp**.

chmod(1) command (reference # 302)

The **chmod** command changes permission on the target of a symbolic link only. See the **chmod**(1) manual page for more information.

chown(1) command (reference # 302)

The new option **–h** has been added to support changing owner of a symbolic link. See the **chown**(1) manual page for details.

cp(1), mv, and ln commands (reference # 302)

These utilities can now handle symbolic links. The options **–i** (interactive) and **–p** (preserve modification time) have been added to the **cp** command. The **–s** option has been added to **ln** to support symbolic links of files across filesystems. See the **cp**(1) manual page for more information.

cpio(1) command (reference # 238)

The new option **–L** has been added so that symbolic link files can be followed or read as link files. See the **cpio**(1) manual page for more information.

cpio(1) command (reference # 316)

The **–A** and **–F** options to cpio are no longer supported.

- The **–c** option is only supported in the SVID environment. The **–g** option is only supported in the POSIX environment . These two options select binary or ASCII headers. (The command **echo $OS** displays the current environment.)
- The **–G** option has been added to allow OS90 users to read **cpio** tapes written with previous versions of **cpio.**
- **cpio**'s default buffer size is 512 bytes, and the number of blocks displayed at the end of an archive read/write is given in 512-byte blocks. Use the **–B** or **–C** option to modify the buffer size used by **cpio.**

See the **cpio**(1) manual page for more information.

cpio(1) command (reference # 366)

To read in files that have filenames greater than 14 characters, a user must be in the SVID environment. In the SVID environment, the filename is truncated to 14 characters; in the POSIX environment a warning message is displayed and the file is not loaded. (The command **echo $OS** displays the current environment.) See the **cpio**(1) manual page for more information.

cpp(1) C language preprocessor (reference # 791)

cpp now parses the backslash continuation correctly.

css(1M) command (reference # 868)

The **css** command now identifies the device board (ACDB, DSDB, LANWAN) for each IOPM in the system. See the **css**(1M) manual page for more information.

css(1M) command (reference # 949)

The **css** command can now be run by all users. In previous releases only root could run **css**. See the **css**(1M) manual page for further information.

cu(1C) command (reference # 936)

The **cu** program now provides quicker responses. Characters are echoed to the screen as they arrive. See the **cu**(1C) manual page for more information.

curses(3X) subroutines (reference # 187, 120)

The **halfdelay**() and **wtimeout**() routines now work as documented. In earlier releases timeout occurred immediately.

curses(3X) subroutines (reference # 199)

Better error checking has been added to numerous **curses** routines.

curses(3X) subroutines (reference # 349, 232)

Function keys are now recognized properly by **curses** and utilities based on curses (such as the **vi** editor). Previously, a timing problem while using the alarm sequence caused the arrow key and mapped sequences to be misinterpreted in some cases.

dd command

Occasionally, use of the **dd** command with a nine-track tape drive can cause the drive to display the message `rc(75) error` and hang the system.

debug.h header file (reference # 110)

The *debug.h* header file has been modified to comply with the ANSI C standard. Programs which include <sys/debug.h> no longer get warning messages from the ANSI C preprocessor.

du(1M) command (reference # 302)

The **du** command does not traverse a symbolic link when trying to calculate disk usage. See the **du**(1M) manual page for more information.

fcntl(2) system call (reference # 528)

A check has been added to **fcntl()** for out-of-range requests, and ERANGE is returned (instead of EMFILE, as in previous releases). See the **fcntl**(2) manual page for more information.

file(1) command (reference # 302)

The new option **-h** has been added to provide information about the target of a symbolic link. See the **file**(1) manual page for details.

file systems (reference # 111)

File systems that were created with earlier releases that used smaller (64-byte) i-nodes can now be mounted read/write.

filesave(1M) command (reference # 446)

The **filesave** utility now requires three arguments to correctly copy the bootimage. See the **filesave**(1M) manual page for details.

find(1) command (reference # 302)

The **find** utility has been enhanced to cross symbolic links and detect a symbolic link loop. The new option **-follow** has been added, and the **-type** option now has an **l** suboption when searching for symbolic link files. See the **find**(1) manual page for details.

floating point emulation (reference # 921)

Programs intensively using floating point operations no longer prevent other processes from running or monopolize the system.

fsck(1M) command (reference # 302)

The **fsck** utility recognizes symbolic link files and does not remove them. See the **fsck**(1M) manual page for more information.

fsck(1M) command (reference # 564)

The */etc/checklist* file, used to specify filesystems for **fsck**, can now include comments. Lines beginning with a pound sign (#) are ignored. Lines can also end with comments after a space, tab, or #.

fsdb(1M) command (reference # 302)

The **fsdb** utility recognizes a symbolic link file and shows its type. See the **fsdb**(1M) manual page for more information.

getitimer(2) and setitimer system calls (reference # 229)

The **getitimer** system call now returns 1 μsec for the timer value if the timer has just expired and is about to be reloaded from the interval value. The **setitimer** system call now saves the new timer value before using the new value pointer. See the **getitimer**(2) manual page for more information about these system calls.

getitimer(2) and setitimer system calls (reference # 471)

The virtual interval timer and the profiling interval timer aspects of the **getitimer** and **setitimer** system calls are now available for use. In earlier releases these features were disabled. See the **getitimer**(2) manual page for more information.

getrlimit(2) and setrlimit system calls (reference # 298)

A process can examine its resource limits using **getrlimit**. The soft limit (SFNOLIM tunable parameter) can be modified by a user process through the **setrlimit** system call, provided the process does not attempt to set the soft limit beyond the hard limit (HFNOLIM tunable parameter). Only a process with an effective user ID of root can raise the hard limit. See the **getrlimit**(2) manual page for more information.

getty(1M) command (reference # 735)

The job control switch character (^Z) is now set by **getty**.

getty(1M) command (reference #1154)

The **getty** command now accepts the **–E** command line option to allow passing of environment variables into the user's environment. See the **getty**(1M) manual page for more information.

ioctl(2) system call

Tape marks are now written correctly on nine-track tapes. See the **ioctl(2)** manual page for more information.

IOPM buffer headers (reference # 26)

The number of IOPM buffer headers available has been increased to 400 in Release 3.3. In earlier releases, insufficient buffer headers could cause a slight performance degradation under heavy disk usage conditions.

IOPM code (reference # 861)

Utilities that use up most of the kernel's STREAMS message blocks of a particular size and use multiply-referenced data blocks no longer cause an IOPM ASSERT error message to be displayed.

IOPM code (reference # 782)

The read/write timeout value for cartridge tape drives has been increased, eliminating intermittent timeout problems when using **cpio**(1) with tape drives.

IOPM code (reference # 954)

An IOPM assertion failure no longer occurs if an attempt is made to open a non-disk drive as a disk drive device. Instead, an appropriate error code is returned.

IOPM code (reference # 1016)

The message `IOPMX: WARNING: bufmgr: no d_print routine for dev XXXX, msg = 'no space'` is no longer displayed. The new message is `IOPMX: NOTICE: no space on SCSI disk drive (idX), logical disk X.`

IOPM code (reference # 1053)

The message `request is out of partition space` is no longer displayed when an attempt is made to access a disk past the end of a slice. Instead, the request fails with error code ENXIO.

IOPM code

The IOPM code (*/iopm/iopm*, */iopm/iopm.dsdb*, and */iopm/did/**) in Release 3.3 is "asserted" code. This means that the code prints diagnostic messages if an error occurs. It also means that the green RDY LED on the IOPM blinks.

If it is desirable to run the non-asserted version of the IOPM code, it is available in */usr/sys/debug/iopm*, */usr/sys/debug/dsdb*, and */usr/sys/debug/acdb*. To run the non-asserted code, perform the following copy commands, then reboot the system following the procedures in the *ARIX OS90 Release 3.3 System Administrator's Manual.*

```
cp   /usr/sys/debug/iopm/iopm.NDBG     /iopm/iopm
cp   /usr/sys/debug/dsdb/iopm.dsdb.NDBG   /iopm/iopm.dsdb
cp   /usr/sys/debug/acdb/ldterm.NDBG   /iopm/did/ldterm
cp   /usr/sys/debug/acdb/lpdvr.NDBG   /iopm/did/lpdvr
cp   /usr/sys/debug/acdb/lpmod.NDBG   /iopm/did/lpmod
cp   /usr/sys/debug/acdb/ttydvr.NDBG   /iopm/did/ttydvr
```

IOPM configuration (reference # 610)

IOPM download code must be compiled for the type of CPU, 68020 or 68040, in the system. If the system administrator tries to load an IOPM card with code compiled to work with one CPU type on a system running another CPU type, the IOPM displays a message similar to the following and the system panics:

```
IOPM compiled to work with 68020 kernel.  Kernel is 68040
kernel.
```

IOPM configuration files (reference # 715)

The utility loading the IOPM now receives an error and exits gracefully when an IOPM is loaded using a configuration file in */iopm/cf* that specifies too many streams messages or queues. In previous releases, an ASSERT fail or bus error would result under these conditions.

IOPM errors (reference # 674)

IOPM errors that cause an IOPM stack backtrace to be printed no longer fill the console with backtrace information. In earlier releases, the IOPM could get into a nearly infinite loop of backtraces.

iopmfmt(1M) command (reference # 266)

The user interface for standalone iopmfmt has been enhanced:

- The Previous Menu and Exit Menu entries are now always entry 0.
- The Write Slice Table option now prompts for confirmation and responds with result status.
- Cylinder alignment is now handled correctly.
- The Auto Slice option now sets the recommended default sizes for the root, swap, and usr slices.
- The slice editing function now provides default values for slice size and starting block number, displays allowable value ranges, and displays the slice table after every change.
- The slice table display now includes the last block number in each slice, as well as the starting block and slice size.
- A new slice table bar chart shows where slices appear on the disk.
- The spare bad block function now prompts accurately and accepts both decimal and hexadecimal input.
- UNIX mode command line processing has been improved.

See the **iopmfmt**(1M) manual page for more information.

iopmfmt(1M) command (reference # 447)

iopmfmt no longer creates a defect slice when formatting.

An option has been added to the format menu to allow discarding of reassigned blocks when formatting.

A disktest capability has been added to allow testing to be done on a disk in order to help detect bad blocks. Selecting option 3, Disktest, from the **iopmfmt** main menu displays the following menu:

```
DISKTEST MENU

1-Select test type (read, write, compare)
2-Select one block
3-Select block range
4-Select number of blocks per read/write operation
5-Display read buffer
6-Select write buffer
7-Single test pass
8-Loop on test
9-Reassign Blocks
0-Previous Menu
```

Option 1, Select Test Type, displays the following menu:

```
SELECT TEST TYPE

1-read
2-write
3-write, read, compare
0-Previous Menu
```

The most useful tests are options 1 (read) and 3 (write, read, and compare). Read-only tests are useful because blocks which lose data over time can be detected without making changes to the data on the disk. The write tests destroy any data in the blocks being tested.

Option 2 on the disktest menu, Select One Block, allows a single block to be chosen for testing. This is useful when a particular block is suspected of being bad. By default this option is set at block 2 (see option 3 below)

Option 3, Select Block Range, allows a range of blocks to be specified for testing. For this option and for option 2, a warning is printed if blocks 0 or 1 are selected, since these blocks are reserved to store slice and format information. If write tests are performed on these blocks, all data on the disk option of the Disk Format and Initialization menu, and the disk will then need to be resliced.

Option 4, Select Number of Blocks Per Read/Write Operation, specifies the number of 1K blocks to be transferred in each read and write operation. The default is one block, and the maximum number of blocks that can be transferred is 256.

Option 5, Display Read Buffer, displays the data returned by the last test read operation.

Option 6, Select Write Buffer, displays the following menu:

```
SELECT WRITE BUFFER

1-incrementing
2-constant
3-shifted ones
0-Previous Menu
```

The contents of the write buffer can be set to an incrementing value, a constant (the system prompts for the value), or to shifted ones (hexadecimal 01, 02, 04, ..., 80, fe, fd, ..., 7f).

Option 7 on the disktest menu, Single Test Pass, causes the test to be performed once over the selected range.

Option 8, Loop On Test, causes the test to repeat a specified number of times.

Option 9, Reassign Blocks, allows the user to allocate a new physical block for the selected logical block. Typically, this block is one that has caused read or write errors to be reported during disktest or during normal system operation. This option is the same as the Bad Block Assignment option available in earlier releases of **iopmfmt**.

kernel (reference # 850)

File accounting is now handled correctly when a fork fails due to insufficient memory.

kernel (reference # 1204)

The kernel file */arix* is now linked to the non-debugging version of the kernel (*/arix.NDBG*). To build kernels using the debug libraries, make the following changes to */usr/sys/Makefile*.

Change the following two lines:

```
#LIB        =    debug/pm
LIB         =    lib
```

To the following:

```
LIB         =    debug/pm
#LIB        =    lib
```

kernel rebuild (reference # 535)

(68040 only) The **chmagic** program (**/usr/sys/cf/chmagic**) is used during kernel rebuilds to set the correct magic number (0523) for the **kernel** on 68040 systems. The syntax for **chmagic** is:

chmagic *file_name new_magic*

ksh shell

When a user's login shell is **ksh**, the cat /etc/motd statement in */etc/profile* is not executed. For the *motd* file to be displayed at login, the user's login shell must be **sh**.

ksh shell

If a user interrupts execution of the *.kshrc* initialization during login, **ksh** is put in a state where it occasionally core-dumps when executing shell scripts. Allow the *.kshrc* initialization to complete before attempting to run shell scripts.

ksh shell

When **ksh** reads *.kshrc* when doing a shell escape from vi, only exported variables and aliases are recognized. See **ksh**(1) for more information.

ksh(1) shell (reference # 402)

The shell **ksh** defaults to using physical path names upon crossing a symbolic link. The **–V** option can be set to use virtual pathnames. See the **ksh**(1) manual page for more information.

ksh(1) shell (reference # 642)

Supplying very large numbers as arguments to **ksh** no longer causes a bus error.

ksh(1) shell (reference # 839)

ksh now supports symbolic links.

ld(1) link editor (reference # 542)

The **–z** option, which formerly did not function as documented, now works correctly. The effect is that a reference to a null pointer now causes a segmentation violation. A new option, **–0** (zero), has been added to disable this effect and is now the default. See the **ld**(1) manual page for more information.

ld(1) link editor (reference # 778)

The link editor will change in the next release (3.4) to create binaries in which references to address zero are illegal. See the **cc**(1) manual page for the **–z** option, which will become the default in Release 3.4, and the **–0** option, which is the Release 3.3 default.

ldsa command (reference # 266)

The handling of device name input errors by standalone **ldsa** (part of the SPM runtime code) has been improved.

ldterm STREAMS module (reference # 373)

Data that satisfies vmin/vtime requirements is now immediately passed to the stream head to be read. In the past raw data was kept in the ldterm module until a read or poll request was received, and as a result programs that instead expected the data to be at the stream head did not receive the data as requested.

ldterm STREAMS module (reference # 716, 723)

Characters received with parity errors and with ISTRIP set are now correctly marked by the sequence 0xff,0x00,0x*nn* where 0x*nn* is the character with the parity error. The ldterm stream module now correctly strips all 0xff characters that are not part of the parity error sequence if ISTRIP is set.

lint(1) utility (reference # 99)

The **lint** utility now processes the void data type in the ANSI mode of the C compiler.

login(1) command (reference # 571)

The file **/bin/login** is now an suid program owned by root.

login(1) command (reference # 461)

In previous releases, **/bin/login** left the file */etc/login.parms* open when it executed a shell. It now closes the */etc/login.parms* file.

login(1) command (reference # 1155)

The **login** command now has a **–E** option to apply environment variables to all user logins.

login.parms configuration file (reference # 691)

The value of the **sleeptime** parameter in **/etc/login.parms** has been changed from 20 to 5.

lpmod STREAMS module (reference # 1084)

The command **stty push lpmod < /dev/tty00** now works correctly, no longer displaying the error message `Not a directory`.

ls(1) command (reference # 131)

Using **ls**(1) on a file mounted remotely no longer results in an RFS warning message about an unknown system call.

ls(1) command (reference # 302)

The **–L** option has been added to display information about symbolic links. The **ls –l** and **ls –F** commands also provide information about symbolic links. See the **ls**(1) manual page for more information.

make(1) command (reference # 584)

In previous releases **make** occasionally printed out an internal debug message such as:

```
bfk=2 (123) ...
```

In Release 3.3 these messages no longer appear.

mcs(1) command (reference # 1225)

The command mcs –a no longer creates object files that are not linkable.

memory management (reference # 512)

System crashes can no longer be caused by reading */dev/kmem*. */dev/kmem* and */dev/mem* read and write as much as they can near the end of memory and return the number of bytes actually transferred. A bad user address causes an EFAULT error, rather than ENXIO as in previous releases.

memory management (reference # 562)

/dev/kmem is more restrictive than in previous releases: it allows reads and writes only to valid pages in the kernel's virtual memory. The system is no longer crashed by attempts to read an invalid page. However, it is possible to read the kernel's own segment, which includes the per-processor u area and the own structure. See the **mem**(7) manual page for more information.

mfsck(1M) command (reference # 467)

A new **mfsck** utility has been added in Release 3.3. This utility runs the **fsck** command in parallel on file systems that require checking. See the **mfsck**(1M) manual page for more information.

Mirroring

The mirror initialization script, */etc/mrinit*, run at sysinit time may try to run scripts in the background. (See note entitled *Running Shell Scripts in the Background* for a workaround.)

mkdir(1) command (reference # 1157, 1158)

The **–m** option to **mkdir** to set directory permissions with symbolic arguments now functions properly. The **–p** option also sets all new directories in a list to the permissions specified by **–m**. See the **mkdir**(1) manual page for more information.

mkfs(1M) command (reference # 266)

Standalone **mkfs** now accepts a standard binary cpio format tape without the restrictions of previous versions. Note that the root directory is owned by root, group root, unless the tape contains an entry for '.'.

mkfs(1M) command (reference # 419)

The standalone **mkfs** utility can now read cpio tapes with character headers. This allows tapes created using POSIX cpio, which uses character headers, to be restored from the standalone environment.

mknod.acdb utility (reference # 882)

The utility **/iopm/util/mknod.acdb** has been deleted. Use the utility **/local/bin/mktty** to perform the same functions.

mknod.dsdb utility (reference # 1101)

The utility **/iopm/util/mknod.dsdb** has been deleted from the system. The commands **/local/bin/mkdsk** and **/local/bin/mkrmt** perform the same functions. See the **mkdev**(1M) manual page for more information.

mktty command (reference # 438)

The **mktty** command no longer supports Series 800 tty ports. See the **mkdev**(1M) manual page for more information on **mktty**.

mktty command (reference # 876)

mktty now makes nodes for a fifth ACDB correctly, without overwriting the nodes made for the first ACDB. It is no longer possible to make nodes for a sixth ACDB. The arguments to **mktty** can now optionally be entered on the command line. See the **mkdev**(1M) manual page for more information.

multiuser state (reference # 1138)

The **fsck**(1M) utility is no longer invoked twice for the root slice on entering the multiuser state.

NBLKxx tunable parameters (reference # 450)

The NBLKxx streams parameters are now set as documented in the *ARIX OS90 Release 3.3 System Administrator's Guide.*

open files (reference # 107)

It is now possible to have open as many files as the kernel will allow. The define _NFILE in *stdio.h* is no longer the limit.

open files (reference # 1066)

The structures and methods used to maintain standard I/O (stdio) file descriptors have been updated to allow a dynamically configurable number of file descriptors. This new method is now the default. This change affects link level (*.o* file) compatibility between Release 3.3 and previous operating system releases.

To compile and link for and with libraries and object modules from operating systems prior to this change, use the **–i** flag to cc(1). See the **cc**(1) manual page for additional documentation. For a more detailed understanding of the changes, see the sections of the include file ifdefed with DYNAMIC and _NODYNAMIC in the file */usr/include/stdio.h.*

openlp(1M) command (reference # 1123)

The **openlp** command now works correctly. **openlp** now works with direct-connect printers. See the **openlp**(1M) manual page for more information.

parallel printer driver (reference # 23)

The OS90 driver does not send a form feed character on any open or close. The user application is responsible for all formatting of data going to the port. Use the **setlp**(1M) command to configure the parallel printer port to suit the needs of the application.

parallel printer driver

The SPM/ACRW parallel printer driver now operates correctly in systems with more than one memory module.

passwd(1) command (reference # 596)

The **–r**, **–f**, **–l**, **–u**, and **–x** options to the **passwd** command now work as documented. See the **passwd**(1) manual page for more information.

profiler(1M) (reference # 1129)

The number of kernel symbols that the profiler can handle has been increased to 4096. As a result, the profiler now works properly when the kernel is configured with a large number of drivers and options. See the **profiler**(1M) manual page for further information.

prtvtoc(1M) command (reference # 1200)

The **prtvtoc** command now prints correct data for controllers other than zero.

ps(1) command (reference # 865)

The **–n** option to **ps** is no longer needed and is therefore no longer supported. See the **ps**(1) manual page for information on supported **ps** options.

ptrace(2) system call (reference # 670)

The **ptrace** system call now supports the 68020 and 68040 transparently. Debuggers should be modified to use u_ar0 as an offset into the user page rather than an absolute address. For details see the **ptrace**(2) manual page.

The **sdb** debugger is supported in this release.

rm(1) command (reference # 302)

The **rm** command can now handle symbolic link files. See the **rm**(1) manual page for details.

sadc command (reference # 867)

On transition to multiuser state (**init 2**), a message indicating that the file */tmp/sa.adrfl* cannot be accessed is no longer displayed. The **/usr/lib/sa/sadc** command now stores this file in */usr/adm/sa/sa.adrfl* and accesses it correctly.

sar command (reference # 1099)

The **sar** command now reports free memory values and average.

Series 800 support (reference # 255, 266, 379)

Release 3.3 of OS90 contains no support for Series 800 I/O. Utilities to maintain the Series 800 I/O subsystem (defectlist, disktest, dsetup, icb, ldicb, imac, mac) are no longer included in OS90 as of this release.

setpgrp(2) system call (reference # 1064, 1078)

The system call **setpgrp**(2) has been removed from the POSIX *libc.a* library. This call is only defined under SVID System V C. To link programs which use the **setpgrp** system call, use the **–SYSV** flag to the **cc** compiler. The POSIX functional equivalent to **setpgrp** is the **setpgid**(2) system call; **setsid**(2) provides a similar routine. See the **setpgrp**(2), **setsid**(2), and **setpgid**(2) manual pages for more information.

settape(1M) command (reference # 655)

Support has been added for quad density nine-track tape drives. Support for 800 and 3200 bpi densities have also been added.

The format of the command is:

settape *tape_device* [*command* ...]

where *command* is one or more of the following:

high_speed	use high speed mode
low_speed	use low speed mode
6250_bpi	use 6250 bpi (high) density mode
3200_bpi	use 3200 bpi density mode
1600_bpi	use 1600 bpi (low) density mode
800_bpi	use 800 bpi (low) density mode
has_cache	tape drive has cache
no_cache	tape drive does not have cache

See the **settape**(1M) manual page for details.

settape(1M) command (reference # 1201)

The **settape** command now prints out the correct value for the tape density of the nine-track tape drive.

SFNOLIM and HFNOLIM tunable parameters (reference # 298)

The NOFILES parameter in */etc/system* (number of open files per process) no longer exists in Release 3.3. It has been replaced with the tunable parameters SFNOLIM (default soft limit for number of open files per process) and HFNOLIM(default hard limit for number of open files per process) These parameters are both initially set to 64. SFNOLIM is analogous to the old NOFILES parameter: it is the number of open files available to normal processes which do nothing to modify their limits. SFNOLIM cannot be set larger than 64. See the *ARIX OS90 Release 3.3 System Administrator's Guide* for more information on these parameters.

sh(1) shell (reference # 645)

The Internal Field Separator (IFS) now only works for the **read** built-in command. Thus, the two lines:

```
$ IFS=/; export IFS
$ /bin/ls
```

now generates output from the **ls** command. Previously it would try to run a command called **bin**. IFS still works for the **read** command, so the following lines still parse the *passwd* file correctly:

```
$ IFS=:; export  IFS
$ while read unam pw uid rest
do
echo $rest
done < /etc/passwd
```

sh(1) shell (reference # 1125)

Shell scripts that include commands to be run in the background now execute correctly.

shutdown command

Issuing the **shutdown** command from a remote system does not actually shut down the system, but instead closes the remote connection itself, with the system still running at init 2. However, the remote terminal displays normal shutdown messages. System shutdown should be run from the system console.

shutdown command

When you try to use **shutdown** while there are active processes utilizing network STREAMS services, the network-oriented user processes are not terminated as they should be. Instead they remain in CLOSE mode for an interval, and eventually terminate due to time-out. In this situation, allow sufficient timeout for all processes to terminate and shutdown to complete; this can require up to ten minutes. Then use the **umountall** command to ensure that all filesystems are unmounted before powering down or rebooting the system.

shutdown(1M) command (reference # 1109)

The **/etc/shutdown** script now supports the **–i** option. The init states 0, 1, 5, and 6 selected by the **–i** option are now correctly supported. See the **shutdown**(1M) manual page for more information.

signal handling

A problem with the handling of POSIX job control stop signals (SIGSTOP, SIGTTIN, SIGTTOU) has been corrected. Formerly if one of these signals was taken during certain system calls, and the process was orphaned, the system would crash.

spacewatch utility (reference # 31)

The utility **/usr/lbin/spacewatch** now functions correctly, sending an urgent broadcast to users when the remaining free space becomes less than the pre-set limit. This utility is part of the **sysadm**(1M) menu system.

SPM Runtime code (reference # 581)

The SPM now identifies old IOMs as "obsolete IOM" and flags the fact that there is an unsupported IOM in the system.

SPM runtime image (reference # 266)

The following changes have been added in Release 3.3:

- If a system contains both 68040 and 68020 processor modules, the 68020 PMs automatically deconfigure themselves.
- IOM/IOA expansion cabinets are automatically deconfigured.
- IOPM/DSDB code is downloaded before the kernel is loaded; if there are any problems with the DSDB load, the boot stops. In previous versions, the kernel continued to boot and the IOPM/DSDBs would panic.
- Memory initialization performance has been improved.
- When the SPM runtime image is booted from a tape drive, it now searches for the standalone utilities on the same tape drive, not on a disk drive as in previous versions.

stdio (reference # 1194)

POSIX-linked programs now include low-level **stdio** routines only when the high-level **stdio** routines are called. The change is in the ordering of routines in */lib/libca.a* and */lib/libc.a*. As a result, data is no longer lost at the end of ANSI-linked programs if the output is fully buffered. Specifically, when an ANSI-linked program that writes to **stdout** has its output redirected to a file, the file now contains the full output of the program.

STREAMS (reference # 51)

The **ldterm** module is automatically pushed on the stream stack of the tty driver the first time the tty driver is opened for that stream. As a result, STREAMS tty devices now have the same capabilities, upon being opened, as a non-STREAMS tty device. See the **termio**(7) manual page for more information.

STREAMS (reference # 428)

In communication protocols, unrequested messages that come in from another node before the communication stack is fully assembled are now discarded. Since the remote machine does not receive an acknowledgement to its probe message, it sends another message. When the communication stack is assembled on the local system, it acknowledges the message. When the IOPM or PM discards a message because it was in the process of building the communication stack (linking streams), the console displays the message:

```
WARNING:  messages discarded from xxxx
```

where *xxxx* describes the queue where the message was when it was discarded.

stripe driver (reference # 234)

The stripe driver now keeps all of its components open and read-only from the time when a stripe is created until it is destroyed. In earlier releases, the stripe driver closed components when a stripe was not in use, making the components vulnerable to being overwritten.

swap(1M) command (reference # 678)

The **–l** option now shows the path name of the swap device. See the **swap**(1M) manual page for more information.

swap daemon (reference # 519)

The name of process 0, 'sched' in previous releases, is now 'swapper'. The intent of this change is to eliminate confusion over the function of this process, which swaps large processes in and out of memory as system conditions require, a function separate from process scheduling.

symbolic links (reference # 237, 285)

Support for symbolic links has been added. Four new system calls have been added: **lstat**, **symlink**, **readlink**, and **lchown**. See the **stat**(2), **chown**(2), **readlink**(2), and **symlink**(2) manual pages for details.

sysadm command

The **sysadm** menus contain an option to configure SXT devices for the **shl** shell layer manager, the addsxt choice on the **sysadm** usermgmt menu. Implementation of **shl** is planned for a future release but has not been accomplished in ARIX OS90 Release 3.3. As a result, selecting addsxt performs no function and produces an error message; do not select this option.

sysadm restore utility (reference # 33)

The listing option of the restore submenu of **sysadm**(1M) now functions correctly.

sysdef utility (reference # 779)

A new magic number has been established for 68040 binaries. **/etc/sysdef** now recognizes this number.

System90/15 support (reference # 322)

The SPM runtime code has been modified to support the System90/15.

tar(1) command (reference # 302)

The **–L** option has been added to **tar** to allow it to access the target of a symbolic link file. See the **tar**(1) manual page for details.

tar(1) command (reference # 895)

The **tar** command does not support the **–e** option. See the tar(1) manual page for more information about supported **tar** options.

tape drive support (reference # 1017, 1193)

Users can now write and read tape after the EOM mark on nine-track tapes. After receiving an error (ENOSPC) from a **write**(1) to a nine-track tape, additional writes can be performed. If the application continues to write beyond the physical end of tape mark, the write will again fail with ENOSPC, and no further writes of any kind (including tape marks) can be performed.

trap 2 instruction (68040 only)

The test-and-set emulation that was available in previous releases using **trap 2** is no longer available on the 68040 product. This feature was previously provided for Series 800 compatibility. This was necessary because of differences in the 68040's atomic instruction operation from earlier members of the 680x0 CPU family. In particular, the **cas**, **cas2**, and **tas** instructions all perform a write operation whether they have succeeded or not.

NOTE:

The **trap 2** emulation will not be available for the 68020 in future OS90 releases.

---

tty driver (reference # 359)

In earlier releases, modems that continuously send CR characters could stay ahead of the shell or getty process trying to read the characters. This could cause more STREAMS messages to accumulate than the kernel could handle, causing all STREAMS activity to stop. In Release 3.3, the amount of data that can accumulate on one tty stream has been limited to prevent this occurrence.

tty driver (reference # 374)

In earlier releases, utilities that set the terminal in raw mode and used vmin or vtime would result in the shell receiving an EOF character when the utility exited, causing the shell to exit. In Release 3.3 this no longer occurs.

tty driver (reference # 632)

Many operations, such as write, that succeeded (incorrectly) after a stream had been hung up now fail.

tty driver (reference # 641)

After a NODELAY open, a second open of the port that is not NODELAY waits for carrier. In earlier releases, the second open did not wait for carrier.

tty driver (reference # 520)

Several open error returns have been changed. An invalid minor number now returns ENXIO instead of ENOSPC as in earlier releases. Port not available (no ACE card) now also returns ENXIO, instead of ENODEV as in earlier releases.

tty driver (reference # 631)

Writes to a stream that has been hung up on now return EIO. Previously these writes returned ENXIO.

tty driver (reference # 851)

Certain combinations of jobs invoked with **nohup** and terminals being turned off no longer cause an ASSERT error message to be displayed.

tty driver (reference # 1130)

A break sent out a port immediately before the port is closed no longer leaves the port indefinitely in a state of sending out a break. The break now stops in at most one second after all data is flushed from the driver during a close. A normal break of 1/4 second duration stops in 1/4 second. A user requested long break (longer than one second) may be truncated.

uadmin(1M) command (reference # 730)

The command **/etc/uadmin** has been duplicated on the **/usr** slice as **/usr/etc/uadmin**. This command should be used when exiting UNIX.

ulim utility (reference # 853)

The utility **/local/bin/ulim** now accepts four arguments in */etc/ulimrc*. A new argument sets the number of files the user can open. This value **must** be less than or equal to the hard limit set by root. The old format for **ulim** can still be used, and the new argument can be replaced by a dash (–) in which case the current value is not changed. See the **ulim**(1M) manual page for further information.

ulimit(2) system call (reference # 420)

The **ulimit()** system call now works properly from an rlogin shell.

uname(1) command (reference # 533)

The system release field returned by **uname –r**, and found in the **utsname** structure, has a new format:

CPUtype:Product-KernelID

For example:

```
2:3.3-04      68020 kernel, ARIX OS90 3.3, kernel ID 04
4:3.3-12      68040 kernel, ARIX OS90 3.3, kernel ID 12
```

See the **uname**(1) manual page for more information.

uucp(1C) command (reference # 1080)

The **uucp** command and its related utilities now work correctly in the POSIX environment.

version numbers (reference # 1240)

Standalone programs, such as **iopmfmt**, now print correct version numbers in response to the **file** command.

vi editor

When using the **vi** editor supplied with the Domestic Overlay tape, you cannot operate the **vi –x** command in both the regular shell and a subshell simultaneously. For example, if you enter the following commands:

vi –x file1 «Return»
:!vi –x file2 «Return» (*subshell*)
:q (*quit subshell*)

vi should display:

```
HIT RETURN TO CONTINUE
```

However, nothing happens and the terminal does not accept input. If you do not attempt to run the command in a subshell, **vi –x** works correctly.

vi editor

Function keys now operate correctly within the **vi** editor family. Previously, a timing problem while using the **curses** alarm system caused the arrow key and mapped sequences to be misinterpreted in some cases.

whodo(1M) command (reference # 1069)

The **whodo** command now displays the proper information. See the **whodo**(1M) manual page for further information.

xterm (reference # 762)

In previous releases, **xterm** would occasionally open a window without a controlling tty, resulting in inability to pipe to **pg** or use /dev/tty. This error has been corrected.

xterm terminfo entry (reference # 1108)

The recognized cursor sequences for the **xterm** terminfo entry have been changed. The sequences now recognized are:

Up:	ESC [A
Down:	ESC [B
Left:	ESC [C
Right:	ESC [D

The sequences recognized in previous releases are:

Up:	OA
Down:	OB
Left:	OC
Right:	OD

vi Notes B

The vi full-screen editor is a standard UNIX operating system utility that can be used to modify ASCII system configuration files.

The chart below summarizes the most basic vi commands and operations. For a complete vi summary, see the vi(1) manual page in the *ARIX OS90 Release 3.3 User's Reference Manual*. For a detailed vi tutorial, see the *ARIX OS90 Release 3.3 User's Guide*.

The vi editor has two basic operating modes: command mode, in which the program interprets keystrokes as the instructions listed below; and insert mode, in which keystrokes are entered into the file. To exit from insert mode and return to command mode, press «Esc».

Command	Description	Command	Description
vi *filename*	Edit the file *filename*	o	Add text below the current line
ZZ	Save and exit	O	Add text above the current line
:q!	Exit without saving	x	Delete character
:w	Save, continue editing	dw	Delete word
«Control»«d»	Scroll down	dd	Delete line
«Control»«u»	Scroll up	r	Replace character
:n	Go to line number n	cw	Change word
/*string*	Search forward	cc	Change line
n	Repeat last search	u	Undo previous command
?*string*	Search backward	U	Undo previous commands for current line only
a	Add text to right of cursor	:e!	Undo all commands since last :w
i	Insert text at cursor	«Esc»	Exit insert mode, return to command mode

Item	Description
ACDB	Asynchronous Communications Device Board.
ACE	Asynchronous Communications Extender.
ACRW	Asynchronous Communications Real Word (System90/15 and System90/25 only).
CSS	Computational Sub-System (main system bus).
CSS/XA	Computational Sub-System Expansion (expansion CSS bus).
DPM40	Dual Processor Module with 68040 processors.
DSDB	Dual SCSI Device Board.
IOM	Input/Output Module.
IOPM	Input/Output Processor Module.
IOSBA	Input/Output System Bus Adapter.
LAN/WAN	Local Area Network/Wide Area Network.
PM	Processor Module.
RWI	Real World Interface (System90/45 and System90/85 only).
SCSI	Small Computer Systems Interface, a standard I/O interface popular for storage devices.
SPM	Service Processor Module.