# *Atari System V*

## *Developer's Guide*

*1991*
*Atari Computer Corporation*
*Sunnyvale, CA 94089-1302*

# *Contents*

# *Contents, continued*

# *Contents, continued*

# List of Figures



# List of Tables

# *Overview*

This is the documentation for the prerelease Atari System V to selected developers and user sites. It is intended as an installation guide, a system administrator's guide, a users' guide, and a programmer's manual.

Whether you're a software application developer or a system administrator—or both—this guide introduces you to Atari System V and helps you set up, maintain, and use the software system.

As a developer or administrator, you probably have the necessary familiarity with UNIX, C programming language, and shared libraries. You'll find recommendations for supplemental reading at the end of each chapter.

All information specific to Atari System V is included in the four chapters of this guide. The appendixes contain references you may find helpful while getting acquainted with and using Atari System V.

## ATARI SYSTEM V RELEASE PACKAGE

You may have purchased the Atari System V for yourself as a standalone system or you may be responsible for installing it on a networked system. In any case, the package you receive includes the following:

- Atari System V Operating System
- X Window System Version 11, Release 4 for Atari System V
- OSF/Motif for the X Window System
- WISh2 (a graphical shell)
- Wx2 (a text editor)
- GNU C compiler, C++ compiler, and GNU debugger
- XFaceMaker 2 (an interactive graphical interface builder)
- Atari Applications Library

## HOW THIS GUIDE IS ARRANGED

*Atari System V is based on Motorola 68000 hardware, but no UNIX System V documentation specific to this hardware interface is available. Instead, we have referenced the AT&T 3B2 documentation throughout this guide.*

*However, some portions of the AT&T documentation do not apply, and those cases have been noted in the margins.*

This guide is designed for experienced users, system administrators, and developers, whether you simply want to know how to get started on Atari System V as an applications user, or whether you expect to use the advanced features to develop your own applications. The guide includes the following:

Chapter 1 "Installation" gives you unpacking tips, lists hardware requirements, and leads you through the steps for installing the software.

Chapter 2 "Interacting with the Atari System V Graphical Interface" explains how to log in, introduces the WISh2 shell, gives you instructions on setting up your password, and tells you how to customize your working environment.

Chapter 3 "Administrative Tasks" describes system processes and accounts, user accounts and applications, Atari tools, peripheral devices, file systems, and system reconfiguration.

Chapter 4 "Application Development" tells how to develop an application, write a device driver, or port a GEM/TOS application to the Atari System V.

The appendixes contain useful reference material:

Appendix A "Atari-Specific Manual Pages" is a list of the manual pages found on-line with Atari System V. The manual pages that were changed or omitted are also listed in order to indicate the differences between Atari System V and the AT&T System V, Release 4, 3B2 version.

Appendix B "Boot Text" provides some sample listings of the text that appears on the system console when you boot the system.

Appendix C "References" contains a list of the materials referenced in the guide.

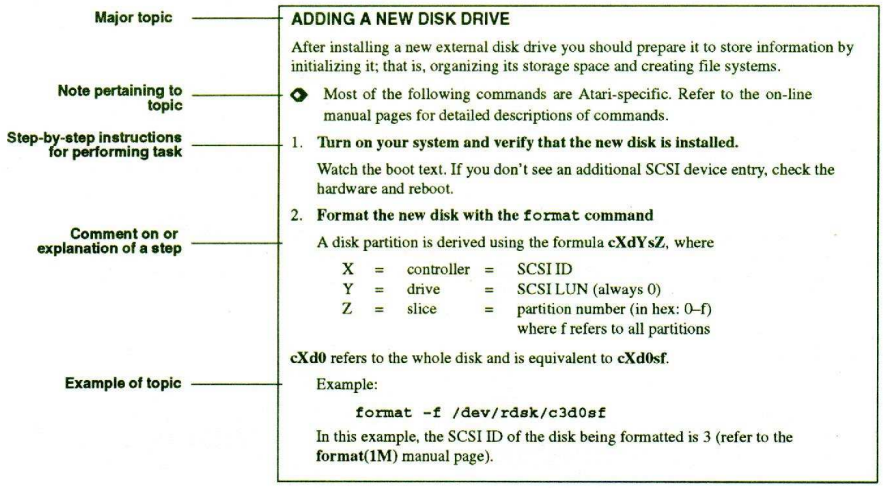
Appendix D "GEM—Xlib Equivalents" is a table to help you find Xlib functions that are equivalent to GEM functions when you port GEM/TOS applications to Atari System V.

Appendix E "Atari Enhancements to Internationalization Standards" contains a table of Atari System V functions and the corresponding XPG3 function.

Index

## USING THIS GUIDE

You needn't read the chapter contents in a particular order. Following is an illustration that shows a typical structure for describing a process.




## FONT CONVENTIONS AND SYMBOLS

The following font conventions and symbols are used throughout this guide.

| | |
|---|---|
| **bold** | The text used for the steps in a process appear in **bold**. |
| *italics* | Names of directories and files appear in *italics*. |
| `constant width` | System output, such as screen messages, appears in `constant width`. |
| | User input, such as commands, options, and arguments appear in **`bold type, constant width`**. |
| <Return> | Input that does not appear on the screen when typed, such as passwords, tabs, or a carriage return, appears between angle brackets. |

| | |
|---|---|
| **<file name>** | The words between angle brackets in examples of command lines explain what should be included when you use the command; i.e., your name, a file name, the date, etc. |
| **command(number)** | A command name followed by a number in parentheses refers to the part of a system reference manual in which that command is documented. |
| ◆ | The rotated box symbol calls your attention to a special note. |
| ◆ | The diamond symbol indicates a choice of tasks or procedures. |
| ▷ **Caution** | Read the message following this warning carefully before proceeding. |

CHAPTER 1

# *Installation*

## UNPACKING

The following items are shipped with Atari System V. Check the package you receive to be sure everything is included.

- License agreement
- Release notes
- Mail-in card for requesting GNU source code (there is a nominal fee for the distribution media)
- Atari System V Developer's Guide (this document)
- Atari Style Guide
- NSL WISh2 User's Manual
- NSL Wx2 User's Manual
- NSL XFaceMaker 2 User's Guide
- AT&T Product Overview and Master Index
- TOS diskette for
  - Resetting boot preference
  - Partitioning a drive
- Atari System V diskette(s) with software updates

## REQUIRED HARDWARE

Atari System V requires the following hardware:

- Atari TT030 with the following items (refer to the *Atari TT Computer Owner's Manual* describing the main unit and ports):
  - Atari TT hard disk cover
  - Brackets to support the hard disk
  - Atari TT bottom panel with ventilation louvers
  - Motherboard (see release notes for required revision level)
  - Atari TT bottom sheet metal piece with ventilation louvers
- 200MB disk drive
- RAM board combination to equal or exceed 8MB
- Atari 19-inch monochrome monitor
- Atari three-button mouse

*If you need specific hardware part numbers, refer to the release notes.*

An authorized dealer will verify this setup and be able to obtain any parts you may need.

## HARDWARE INSTALLATION

The dealer will install the Atari System V disk drive in your machine as the internal drive. This drive uses a SCSI (small computer system interface), which means that

- the internal drive must always be terminated, and
- the motherboard must always be terminated unless you have both an internal hard disk and SCSI devices connected to the external back port.

If you are upgrading an existing TT with hard-disk resident TOS data on an internal SCSI drive, you must back up the data before replacing the internal disk with the disk containing UNIX. The section "Adding a New Disk Drive" in Chapter 3 explains how you can partition the hard disk drive for both TOS and UNIX partitions.

The Atari System V supports external SCSI devices. Refer to the release notes included in the system package for a list of tested and supported SCSI devices that can be ordered through your dealer. These devices include an additional SCSI (not ACSI) hard disk and a tape drive.

The Atari VME Ethernet board may be installed if your computer will be used on a network.

## SYSTEM INSTALLATION

*Power on*

When the computer system is completely assembled, switch on all peripheral devices and then switch on the computer.

The diskette light goes on-off-on; when it stays on, the TT initialization is complete. At that same time, the hard disk light turns on. When the internal hard drive has completed its initialization, the light goes off.

When the TT and the hard disk are both initialized, use the following steps and information to boot Atari System V into your computer.

*Press space bar when TT and disk drive are ready*

1. **Press the space bar (or wait 90 seconds).**

   This message is displayed:

   ```
   Atari System V.4 <disk date>
   ```

   The system probes all SCSI devices and lists them. The Equipped Device Table is displayed and then you see the device specification and the name of the kernel that Atari System V will boot:

   ```
   hd(0,0)unix
   ```

*Press **<Return>** to continue boot process*

2. **Press <Return> to continue the boot with this kernel.**

   If you accidentally press some other key, the boot stops. To continue the boot, either

   a. Press the reset button on the side of the computer to start the boot from the beginning, or

   b. Type `unix -r` and then **<return>**

*If devices were added, the system reconfigures and reboots—go back to Step 1*

3. **If you added a tape drive, an Ethernet board, or some other supported device, the system reconfigures and automatically returns to Step 1.**

More system messages are displayed, including the Ethernet address (if added), memory capacities, and copyrights.

4. **Finally, this message is displayed**:

```
Press <Delete> to enter System Maintenance Mode.
```

*Appendix B contains boot text messages, some with and some without peripheral device changes.*

- To continue to come up in multiuser mode, do nothing and go to Step 5.
- To become superuser in system maintenance mode you have five seconds to press **<Delete>**and enter the superuser password at the prompt. (As shipped, no password is set for *root*, so set one as soon as possible.)
  - To exit system maintenance mode and to continue bringing up the system, type **exit** or press **<Control d>**.

As superuser in system maintenance mode you may perform any of the following tasks; just select the task and follow the instructions.

- Set the root password.
  a. Type **passwd.** <Return>
  b. Type the new password, <Return>, and at the prompt
  c. Type the new password a second time
- Set the time of day and the time zone.
  a. Type **setenv TZ <zzzXddd>**

*Set date and time of day*

| | | |
|---|---|---|
| **zzz** | = | local time zone |
| **X** | = | number of hours from Greenwich mean time |
| **ddd** | = | local daylight savings time zone |

For instance:

| | |
|---|---|
| Pacific Coast | = PST8PDT |
| East Coast | = EST5EDT |

  b. Type **date <mmddHHMMYY>**

| | | |
|---|---|---|
| **mm** | = | month |
| **dd** | = | day |
| **HH** | = | hour |
| **MM** | = | minute |
| **YY** | = | year |

*If the local language is not American English*

- Set the default language.
  a. Type **setlang**
  b. When the prompt appears, select a language from those available

Thereafter, all applications on your system (including the rest of this boot) default to this language. Individual accounts may elect to override this for their own sessions only. The **setlang** program can be run by superuser at any time, but affects only those sessions started after it has been run.

*Install software the first time you boot*

- Install software updates
  a. Type **pkgadd -d /dev/floppy**
  or
  **pkgadd -d /dev/tape**

b. When prompted, insert the diskettes or tapes

*Verify that the Atari VME Ethernet controller is installed*

◆ Set up the system for network access

a. Type **`setnetwork`**

b. You are asked for your machine's node name and IP address. Get these from your network administrator.

c. You are asked for the node name and IP address of another machine on the network. These are used to get the hosts file once the network is operational.

d. It's up to you to propagate your node name and IP address to all other nodes of the network. Ask your network administrator about this.

To connect your network to the public internet, contact SRI–NIC (SRI International, Network Information Center) to obtain a network number and domain name. Telephone (USA) 1–800–235–3155, or see AT&T, *UNIX System V, Release 4, Network User's and Administrator's Guide*, Chapter 2.

*To reboot, type **init 6***

In all system maintenance situations, if the program instructs you to reboot after making changes, type **`init 6`** and return to Step 1.

*To continue, type **exit***

If you are not instructed to reboot, type **`exit`** and the system continues to come up.

5. **Atari System V continues and when initialization is complete, this login screen is displayed:**


6. **Four login accounts are available:**

| | |
|---|---|
| **guest** | **croot** |
| **root** | **kroot** |

*Login*

Log in as **root** (or **croot** or **kroot**, if you're familiar with the C-shell or Korn shell) and create new user accounts. Do not edit */etc/passwd*; Atari System V maintains a shadow file containing the passwords and directly editing */etc/passwd* will confuse the password lookup mechanism. These files should be updated by system commands. See the "User Accounts" section in Chapter 3.

*It's important to learn how to back up the system.*

❑ Once your system is installed, your most important consideration is knowing how to back up that system. To learn the necessary backup procedures, see the section, "System Backups" in Chapter 3.

## REFERENCES

Atari, *Atari TT Computer Owner's Manual*, 1990, Atari Corporation

AT&T, *UNIX System V, Release 4*, Documentation, 1990, Prentice-Hall —as follows:

*Network User's and Administrator's Guide*
*System Administrator's Guide*
*Users Reference Manual*

CHAPTER 2

# *Interacting with the Atari System V Graphical Interface*

## LOGGING IN TO THE SYSTEM

This is what the screen looks like as soon as the system is up and running:


1. **Type in your login ID as set up by your system administrator. Press <Return>**
2. **Type your password (which will not be printed on the screen). Press <Return>.**

For generic access, the login ID **guest** will let you to use the WISh2 capabilities.

## WISh2 WINDOWING SHELL

When you log in to Atari System V, you are automatically brought to the WISh2 windowing shell running under the OSF/Motif Window Manager (MWM).

If this doesn't happen automatically, either have your system administrator set up your account to do so, or set it up yourself. See the "X Window System Session Management" section of Chapter 3.

Refer to the *OSF/Motif User's Guide* for information about manipulating windows, application menus, buttons, and dialogs. Figure 2-1 shows the WISh2 shell with its icons for the **guest** login.

### Mouse Buttons

Use the mouse buttons to choose objects on the screen, choose text to be edited, focus the keyboard on a particular window, or move objects on the screen. Table 2-1 describes the mouse buttons, their positions, names, actions, and functions.

**Figure 2-1**
**WISh2 Shell**

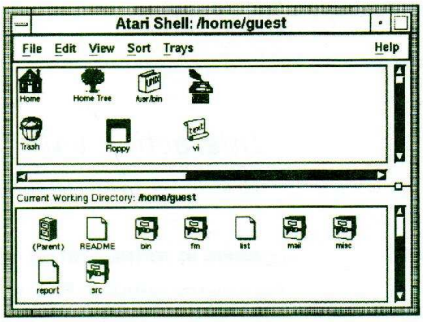



**Table 2-1**
**Mouse Buttons and Activities**

| Position | Name | Action | Function |
|---|---|---|---|
| Left | **Select** | **Click** = Move pointer to object, press and release. | **Selects** object. |
| | | **Double-click** = Move pointer to object, press and release twice. | Executes an object. |
| | | **Drag** = Move pointer to first object or text, press and hold down while moving mouse over selection, release. | Selects multiple objects or text. |
| Middle | **Drag** | **Drag** = Move pointer to first object or text, press and hold down while moving mouse over selection, release. | Alternate button for moving and copying objects. |
| Right | Custom or **Menu** | **Click** or **Drag** = Move pointer to menu, drag to access chosen menu. | Activate pop-up menus and perform application-specific tasks. |

### Getting Help

*__Select__ is used throughout this guide to tell you to position the mouse pointer on the item you want to select and then click the left mouse button.*

**Select** Help from the menu bar; the cursor changes to a small magnifying glass. Move the cursor to an item (field, label, text) and **Select** again. A help window will open and display information.

The **Exit** button is in the File menu (refer to the WISh2 documentation for further information on menus and icons).

The default MWM menus are shown in Table 2-2. Access these menus by the press-drag-release action of the appropriate mouse button on the background outside the windows.

## CHANGING YOUR PASSWORD

Your password prohibits others from using your account. Keeping your password a secret, along with using appropriate permissions on your files and directories, prevents others from altering or destroying your data. Be sure to change your password from time-to-time—your system administrator may even configure accounts so that you are required to periodically change your password.

**Table 2-2 MWM Mouse Buttons Menus**

| Mouse Button | Menu |
|---|---|
| Left | Desktop<br>WISh2<br>New window (use to open a command window)<br>XFaceMaker 2<br>Clock |
| Middle | User definable |
| Right | Window manager<br>Shuffle up<br>Shuffle down<br>Refresh<br>Restart<br>Exit |

To change your password,

1. **Select New Window from the MWM menu to open a command window.**

   A new window is placed on the desktop.

2. **Type the command `passwd` in this window.**

   A prompt asks you to type in your old password. Do this and you are prompted to type in your new password; when prompted, enter the password again to catch any typing errors.

   Note: To be sure others can't decipher it and thus compromise the security of your data, your password should

   - Have a least six characters
   - Contain at least two alphabetic characters and at least one numeric or special character
   - Differ from your login name and any reverse or circular shift of your login name
   - Differ from your old password by at least three characters

   For additional security, avoid obvious passwords, such as a part of your name, name of a family member, the make of your car, your license plate number, etc.

3. **Close the window**

   Continue with your tasks. The next time you log in to System V, you must enter your new password.

## LANGUAGE APPLICATIONS

When you first log in, your windows and applications are presented in the language set up by the system administrator for your particular computer system. However, if you prefer to work in another language, use the following steps to change:

1. **Select New Window from the MWM menu to open a command window.**

   Position the new window on the screen desktop.

2. **If the file *.environment* does not already exist, create it in your home directory and add the lines**

```
LANG=<language>_<territory>
export LANG
```

For example, if you speak German and live in Switzerland, you would enter

```
LANG=german_switzerland
export LANG
```

3. **Save the file. Exit MWM and then log in again.**

You should be using your new language.

## CHANGING CONSOLE SETTINGS

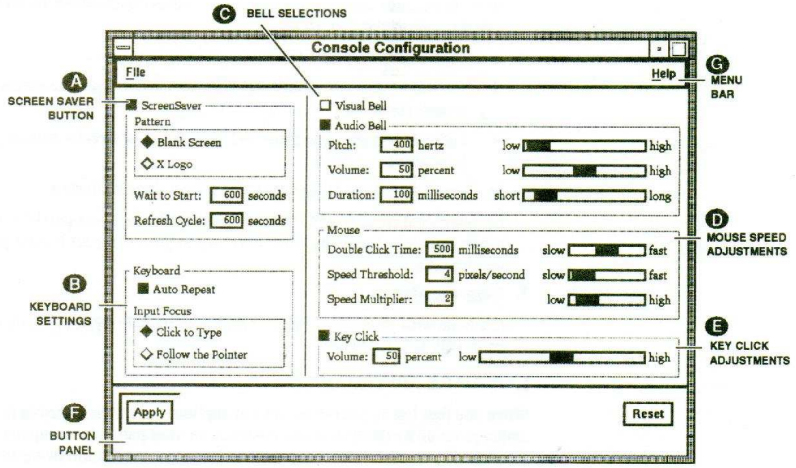
### Console Configuration Window

Use the Console Configuration Window to change console attributes such as screen saver characteristics, keyboard and mouse assignments, and bell and key click volume.

1. **Select the Console Configuration icon.** 

The Console Configuration window pops up—ready for you to adjust the console settings to your own requirements.

**Figure 2-2 Console Settings Window**




2. **Refer to Figure 2-2 to identify each portion of your console while reading the description that matches this portion's attributes.**

*Screen Saver* (A)

### Screen Saver

1. **Select the Screen Saver button to turn it on or off.**

   The screen saver is a utility program that, after a specified number of seconds without keyboard activity, blanks the screen or displays a pattern. The pattern is redrawn (cycled) in the specified number of seconds. This prevents burn-in damage to the screen.

2. **Choose the display pattern by clicking on the appropriate button with the Select mouse button.**

   The screen can go blank or it can display the X Window System logo.

3. **Select the Wait to Start box and type in the number of seconds to wait before activating the screen saver.**

4. **Select the Refresh Cycle box and type the number of seconds to wait before the pattern is redrawn.**

*Keyboard adjustments* (B)

### Keyboard Adjustments

1. **Select the Auto Repeat toggle button to enable or disable automatic repetition of a depressed key.**

   In general, it's a good idea to keep auto-repeat enabled.

2. **Select a type of Input Focus.**

   - In Click-to-Type mode, you must **Select** into a window to enter text.
   - Follow-the-Pointer mode focuses the keyboard on whatever window the mouse cursor is pointing to. This allows you to enter text without clicking into the window.

### Bell

*Bell selections* (C)

1. **Select the Visual Bell button to turn the visual bell on or off.** The visual bell will flash-invert the entire screen.

***Flash-invert:*** *Momentary reversal of screen color; e.g., black to white.*

2. **Select the Audio Bell button to turn it on or off.**

   Using the mouse move the slider knob to change the following Audio Bell attributes:

   - Pitch—how deep or high the sound is
   - Volume—how loud the sound is
   - Duration—how long the sound lasts

*Mouse adjustments* (D)

### Mouse

**To change one of its listed attributes, use the mouse to move the slider knob or Select a box and type a number to change the setting.**

| | |
|---|---|
| Double click speed | Maximum time (milliseconds) allowed between double clicks. |
| Speed threshold | Maximum true speed. When mouse is moved at this speed or slower, the on-screen cursor moves at the same speed. This is useful for manipulating small objects. When the mouse is moved faster, the on-screen cursor moves faster; this is useful for moving across the screen quickly. |

| | |
|---|---|
| Speed multiplier | The number used to multiply the speed of the on-screen cursor, if mouse is moved above the speed threshold. |

*Key click adjustment* E

### Key Click

1. **Select the Key Click button to turn the key click sound On or Off.**

   Key click refers to the clicking noise that is made when you press a key.

2. **If you choose to turn on Key Click, use the mouse to move the slider knob or Select the box and type in a number to set the volume of the key click.**

*Button Panel* F

### Button Panel

1. **Select the Apply button to apply all attribute changes to your current session.**

2. **Select the Reset button to reset all attribute adjustments to their last applied state.**

### Menu Bar

*Menu Bar* G

1. **Select the File button and Select Save from the File menu to save all attribute changes.**

   The new settings are applied to your current session as well as all subsequent sessions.

2. **Select Apply System Defaults from the File menu to change all settings to the system defaults.**

### REFERENCES

Use the documents referenced here for more information about subjects discussed in this chapter.

AT&T, *UNIX System V, Release 4, User's Reference Manual,* Prentice Hall, Inc., 1990.

Manual page **passwd(1)**

AT&T, *UNIX System V, Release 4, User's Guide,*Chapter 3 "Using the File System," Prentice Hall, Inc., 1990.

Non Standard Logics, *WISh2 User's Manual*, Paris, France, 1991

Open Software Foundation, *OSF/Motif User's Guide, Revision 1.0,* Prentice Hall Inc., 1990.

Sections entitled:

"Using the Mouse"
"Exploring PullDown Menus"
"Recognizing Common Controls"
"Moving a Window"
"Entering Text Into an Input Field"

CHAPTER 3

# *Administrative Tasks*

## OVERVIEW

System administration refers to the tasks associated with maintaining Atari System V. Typical tasks include managing file systems, installing new applications, managing user accounts, performing regular backups, ensuring a secure environment, and modifying the system itself to accommodate new needs.

Usually these tasks must be performed using shell commands. In keeping with Atari's goal of providing a friendly system to both the applications user and the advanced user, Atari System V includes window-based tools run from the WISh2 desktop to perform administrative chores. This means you don't need to know the details of how the commands are being used—what's emphasized is the action to be done. A few of these tools are available with this release; more will be available later. Those tools provided with this release are fully described.

## ACCOUNTS AND GROUPS

Accounts and groups are used to establish ownership of and access to files, directories, and commands on the system.

Accounts are set with a name that identifies the user and an identification (ID) number that is automatically assigned by the system. Login accounts identify the user who logged in, created files and directories, and executed commands. Each file or directory is owned by the user who created it; that is, new files and directories have the same account ID as the user. System accounts may be used to execute administrative commands, or may simply identify system resources that pertain to a specific purpose. Each account should be protected with a password.

Files, directories, and commands each have a group ID that affords an additional level of protection. Only accounts that belong to the same group can access those resources. The extent of access (read, write, or execute) can be changed to ensure appropriate security.

### System Accounts

Table 3-1 is a list of accounts shipped on Atari System V. The **root, croot, kroot,** and **guest** accounts are login accounts described later in this chapter.

### System Groups

Table 3-2 is a list of groups shipped with Atari System V. You can add groups with the **groupadd** command. Check the */etc/group* file for the next available reasonable group ID (there may be gaps in the assigned group ID numbers).

Group numbers 0—99 are reserved for administrative use; users should be assigned to group numbers over 100.

**Table 3-1**
**System Accounts**

| User ID | System Account | Description |
|---|---|---|
| 0 | **root** | Superuser account (runs the Bourne shell) |
| 0 | **croot** | Superuser account (runs the C shell) |
| 0 | **kroot** | Superuser account (runs the Korn shell) |
| 3 | bin | Owner of most binary executables |
| 2 | sys | Owner of system binaries requiring special permissions |
| 3 | adm | Owner of accounting and associated data files |
| 4 | nobody | Nonprivileged user ID for NFS |
| 4 | uucp | Owner of uucp files |
| 20 | nuucp | Default login for remote uucp |
| 7 | listen | Network listener |
| 8 | vmsys | Owner of FACE executables |
| 10 | oasys | Owner of object architecture files |
| 12 | sp | Line printer daemon |
| 6001 | **guest** | Unprivileged login |

## Superuser Account

Administrative tasks are done from the privileged account ***root***, commonly referred to as the "superuser" account. No one but the designated superuser may run these processes or update system files.

The ***root*** account differs from other accounts in that it has

- user ID 0,
- group ID 0, and
- permission to access all files and nodes on the system, irrespective of the permissions associated with those files.

**Table 3-2**
**System Groups**

| Group ID | Group | Group Description |
|---|---|---|
| 0 | root | Privileged superuser |
| 1 | other | Default user |
| 2 | bin | Public commands system |
| 3 | sys | Special commands system |
| 4 | adm | Administrative system |
| 5 | uucp | uucp system |
| 6 | mail | Mail system |
| 7 | tty | Terminal files system |
| 8 | lp | Printing files system |
| 10 | nuucp | Incoming uucp system group |
| 12 | daemon | Background processes system |
| 6001 | nobody | Nonprivileged ID for NFS |

Following are some important considerations for using ***root***.

- Become superuser only when you must perform administrative functions. Never use the **root** account for ordinary word processing or program compilation.
- Use the interactive forms of the commands **rm, mv**, and **cp** (e.g., **rm –i**) to warn you before removing files that should be kept.

- Avoid using wildcard characters (such as *) in file name specifications. When using the C-shell or Korn-shell, set variable *noclobber* to prevent overwriting existing files with output redirection.
- Record all administrative tasks performed as superuser in a log book.
- Be cautious and consider the implications of any action; many mistakes made as superuser are irreversible.
- Make backups of all important system files before you change them.

There are three entry points to the superuser account: **root, croot,** and **kroot**. The difference is the type of shell executed when you log in. The **root** account uses the Bourne shell, **croot** uses the C shell, and **kroot** uses the Korn shell.

Sophisticated users may have a preference, but you should use the Bourne shell unless you are familiar with the other shells. Remember to place passwords on all of these accounts whether you use them or not.

## User Accounts

Atari System V manages users with login names, passwords, and groups. To gain access to the system, a user must enter a valid account name (login ID) and password. File permissions allow or prevent user access to resources in the system.

*It's often useful to develop a standard for login names such as: user's initials, first name, first name and last initial, etc.*

Before specifying a new user's account and working environment, have a clear idea of the user's tasks and system needs—information such as

- User's full name; you must know the owner of each login account.
- User's login name: it's best to use only lowercase unaccented alphabetic characters, and it must be unique to the system or network. It must be printable and may not contain a colon or a new line character.
- The full pathname of the user's preferred command shell: */usr/bin/sh*, */usr/bin/csh*, or */usr/bin/ksh*.

Basic copies of the login and environment files needed for Atari System V shells and applications are provided in the directory */etc/skel*.

The following section describes the files used to customize behavior of the windowing software.

### *Guest Account*

Atari System V is shipped with a **guest** account that provides generic nonprivileged access to the system until user accounts are set up. The guest account has no password. To protect your system, either assign a password to this account or remove it—after creating other accounts.

### *Adding a User Account*

To add users to a new system, you must become superuser.

1. **Log in as root.**
2. **Type**

```
useradd -c <full name> -g <groupname> -m -s \
<shell> -k /etc/skel <login name>
```

If there are spaces in the full name, it must appear in quotes.

Example:

```
useradd -c "Hans F. Anders" -g staff -m -s \
/usr/bin/csh -k /etc/skel handers
```

- User's new login name is handers
- The new login account is locked until a password is assigned.
- A new user entry is added to both the */etc/passwd* and */etc/shadow* files.
- The new user is added to the group of *staff*.
- A home directory is created as */home/handers* with read, write, and execute permissions of the default group.
- If no login shell is indicated (*-s* option), it defaults to */sbin/sh*.
- Default startup files are copied from */etc/skel* into the new user's home directory.

3. **Type**

```
passwd <login name>
```

Example:

```
passwd handers
```

Enter the initial password. When prompted, enter the password again. Inform the user of the new account and initial password. The user can change the password thereafter. Refer to "Changing Your Password," in Chapter 2.

### *Disabling a User Account*

Occasionally, if a user no longer needs access to the computer, you may want to disable an account,

1. **The first step in removing a user from a system is to deny that user access to it. Type**

```
usermod -e <date> <login name>
```

The date may be in any form but Julian.

Examples:

```
usermod -e 9/10/91 handers
```

or,

```
usermod -e "18 May 1991" handers
```

After the date you entered, no one will able to access this login.

*Deleting a login name account is not recommended*

The **usermod** command disables, but doesn't delete, user accounts. Deleting the login name from */etc/passwd* and */etc/shadow* is not recommended, as it affects all files owned by that user. The system deletes the owner and uses a number to identify the files. You may forget who the user was and assign that number to another user.

System logs, mail files, and news files may have a record of login names, so it may be useful to maintain expired login entries for historical records.

2. **Locate all files that belong to that user account, back them up, move them, or delete them.**

a. Locate the files:

```
find / -user <login> -print > <file>
```

Example:

```
find / -user handers -print > /tmp/somefile
```

b. Back up the files using **cpio**.

```
cat <file> | cpio -odv > <backup device>
```

Example:

```
cat /tmp/somefile | cpio -odv > /dev/floppy
```

If you have a tape drive, you may want to back up the files onto the device called **/dev/tape** instead of **/dev/floppy**.

Example:

```
cat /tmp/somefile | cpio -odv > /dev/tape
```

c. **Delete the files.**

Do this only if you truly want to permanently purge the system of any trace of these files. If anyone using the system may need these files, copy them, changing ownership.

```
find -user <login name> -exec rm -i {} ;
```

Example:

```
find / -user handers -print | xargs -t rm -i
```

3. **Delete the login name account from the system.**

Do this only if you truly want to permanently purge the system of any trace of the user.

```
userdel -r <login name>
```

Example:

```
userdel -r handers
```

The home directory and files will be deleted (**-r** option). The login name is removed from the */etc/passwd* and */etc/shadow* files.

As mentioned before, deleting a login name account is not recommended.

### Setting Account Passwords

Passwords are maintained in the */etc/shadow* file, separate from the */etc/passwd* file that contains administrative information pertaining to each login account. Direct editing of the **passwd** file will not update the shadow file. Update both these files by commands only—**useradd(1M)**, **usermod(1M)**, **userdel(1M)**, and **passwd(1)**.

The superuser may change any account password, using the command **passwd <login name>**. Users may change their own password using the command **passwd**. This command prompts for the new password to be typed twice, in order to catch typing errors. The new password takes effect on login. Remember: all accounts should have passwords, including **guest**, **root**, **croot**, **kroot**, and **nuucp**.

### Security

The purpose of any security measure is to protect your system from unauthorized access and to maintain its integrity. Refer to the "System Security" chapter of the AT&T *System V Release 4 System Administrator's Guide* for security items to consider, such as passwords, access permissions, dial-up ports, and the **su** command.

In keeping with the goal of not allowing access without accountability, we strongly suggest that each remote system that accesses your system by means of dial-up ports be given a unique login name, rather than using **nuucp**. In this way, access from various systems will be accounted for.

## SYSTEM BACKUPS

As system administrator, it's your responsibility to back up the system periodically. Backup copies can be used to recover files that were removed, corrupted, or otherwise lost.

Suggested backup procedures are discussed in the AT&T, *UNIX System V, Release 4 System Administrator's Guide*, Chapter 3.

The examples in that chapter are implemented using the **backup_service** commands, which you may want to use if your backup responsibilities are extensive. However, the following section describes simple commands for backing up your Atari System V data. Refer to the manual pages in the AT&T, *UNIX System V Users Reference Manual.*

It's a good idea to

- occasionally back up your entire disk, and
- frequently back up file systems and directories that change.

***Streaming tape drive**: A high-speed magnetic tape drive.*

You can back up the Atari System V to either magnetic tape or diskette. The streaming tape drive may be attached to your system or attached to another system across a network.

### Before Backup

Before you start the backup, you should know the following:

- The names of the file systems to be backed up.

For example,

  - */home* has the personal files of all users
  - */var* contains the system variable tables
  - */newstuff* might include research data

- The names of specific directories and files to be backed up.

  For example,

  - */home/mark* might be the name of the directory that contains all the files in Mark's home directory
  - */var/adm/spellhist* is a log of words flagged by the spell character
  - */newstuff/lib/src/*.c* might represent some source code files

- The device on which the file system is located.

  For example,

  – */dev/rdsk/c0d0* represents the internal boot disk

  – */dev/rdsk/c3d0* is an external storage disk

- The name of the device to which the file system data is to be copied.

  For example,

  – */dev/tape* is the name for the streaming tape drive

  – */dev/floppy* is the name of the diskette drive

### Backup Commands

You can choose one of several commands to back up your system. The same command is used to restore data from its backup media. Choose the one that best applies to the task.

*Choose the **dd** command to back up a disk quickly*

◆ To backup an entire disk to tape (image backup):

– Use the **dd** command to back up an entire disk or file system quickly, if you don't care about restoring individual files. This command copies all bits from the disk, so the tape must have a capacity equal to or greater than the disk being backed up. Refer to the **dd(1M)** manual page.

– To increase the data transfer rate, when backing up from disk to disk, use a larger block size than that used for backing up onto tape.

– All file systems to be backed up must be unmounted. Enter the command

```
dd if=<source device> of=<backup device> \
bs=<blocksize>
```

Example:

```
dd if=/dev/rdsk/c0d0 of=/dev/tape bs=32k
```

◆ To restore an entire bootable disk **dd** formatted archive

– Use the **dd** command to restore an entire disk from an image backup. Use a second disk drive that is nonbootable. Restore data to the second disk drive and use the **cp** command to copy files to the bootable disk.

```
dd if=<backup device> of=<system device> \
bs=<blocksize>
```

Example:

```
dd if=/dev/tape of=/dev/rdsk/c1d0 bs=32k
```

This restores the tape contents to the disk with a SCSI ID of 1. The restored files can be copied to the bootable system disk with a SCSI ID of 0.

▷ **Caution** Attempting to restore your bootable system disk directly from an image backup will be unpredictable

❑ When you restore a disk from a backup of your boot disk, run **fsck** on the disk to eliminate ambiguity about the system's state.

You can also use **dd to** back up your entire bootable disk by backing up each file system separately. To restore the entire disk from archive,

restore all file systems except **root**. Boot with the *fsck* file system and restore the root file system from there.

Example:

```
dd if=c0d0s6 of=/dev/tape bs=32K
```

backs up the */home* file system that resides in partition 6 of the bootable disk.

For more information on file systems and partition names see "File Systems," and "Adding a New Disk" in Chapter 3.

*Choose the **cpio** command to backup an entire file system or specific directories and files*

◆ To backup an entire file system to tape, or

◆ To backup specific directories and files to tape

- Use the **cpio** command to back up and restore individual files and groups of files that match a file specification. It's slower than **dd**, but more flexible. Become familiar with the **find** command in order to use **cpio** to its potential. Refer to the **cpio(1)** and **find(1)** manual pages.
- If you backup data with absolute path names, the files are restored to the same place from which they were saved. If you back up data with relative path names, the files can be restored to any location.
- The file system must be mounted. Enter the command:

```
<list of files> | cpio -ovc -C <buffersize> \ >
<backup device>
```

- The input to `cpio` must be a list of files, one per line. Use `ls` or `find` or `cat` to generate the list of files.

Example:

```
cd /home
find . -depth -print | cpio -ovc -C 32768 > \
/dev/tape
```

backs up files and directories in */home* so that they have relative path names. The command to save the same data to have absolute path names is

```
find /home -depth -print | cpio -ovc -C \ 32768
> /dev/tape
```

▷ **Caution** Unless you have the expertise to write a shell script with a complicated **find** command, do not use `cpio` to backup your entire boot disk; it may hang when it reaches the */proc* directory and will hang on any open pipes.

*Browse the contents of a **cpio** formatted archive*

◆ To browse the contents of a `cpio` formatted archive

- Use the `cpio` command to list the contents of an archive.

```
cpio -itc -C<buffersize> < <backup device>
```

Example:

```
cpio -itc -C32768 < /dev/tape
```

The `-c` option assumes that the data was saved with a portable header. If the `-c` option doesn't work, try

```
cpio -it -32768  < /dev/tape
```

*Restore the files from a **cpio** formatted archive*

◆ To restore files from a `cpio` formatted archive

– Use the `cpio` command to restore information from `cpio`-formatted backups.

Enter the command:

```
cpio -idmuvc <buffersize> <target directory> \
<backup  device>
```

❑ The **-u** option overwrites files with the same name.

Example:

```
cpio -idmuvc "/home/mark/*" < /dev/tape
```

This option restores the files */home/mark/** to the exact location from which they were saved, assuming these files were saved with the absolute pathname prefix of */home*. If they were saved with the relative pathname prefix of *home*, the command

```
cd/
cpio -idmuvc -C32768 "home/mark/*" < /dev \
/tape
```

restores them to */home/mark*.

◆ To backup files and/or directories to diskette

*Choose the **tar** command to backup files and directories*

– Use the **tar** command to back up regular files—such as text files and binary files. **Tar** doesn't handle special files, such as device files. Refer to the **tar(1)** manual page.

– The file system where the files reside must be mounted.

– Enter the command:

```
tar -cvBf <backup device> <filenames>
```

Example:

```
cd /home
tar -cvBf /dev/floppy myfile \
/home/myhomedirectory
```

backs up onto the diskette the file *myfile* in the current working directory as well as all files and directories in */home/myhomedirectory*.

◆ To browse the contents of a **tar** formatted backup

*Browse the contents of a **tar** formatted backup*

– Use the **tar** command to list the contents of a **tar** formatted backup tape or diskette.

– Enter the command:

```
tar -tvBf <backup device>
```

Example:

```
tar -tvBf /dev/floppy
```

◆ To restore files from a **tar** formatted archive

*The **tar** command restores formatted backups*

– Use the **tar** command to restore information from **tar** formatted backups.

– Enter the command:

```
tar -xvBf <backup device> <file names>
```

Example:

```
tar -xvBf /dev/floppy
```

retrieves all files on the diskette and restores them to the directory from which you issued the command.

Example:

```
cd /tmp
tar -xvBf /dev/floppy \
/home/myhomedir/science.doc
```

retrieves one file */home/myhomedir/science.doc* from the diskette and restores it to the */tmp* directory.

*Back up files to a tape via the network*

◆ To back up files to a tape across the network

– The **cpio** or **tar** command must be executed on the system to which the tape drive is attached.

You can pipe the files into an **rsh** command that executes on another system:

Example 1:

```
tar -cvBf - /home | rsh somehost dd of= \
/dev/tape obs=32768
```

Example 2:

```
ls | cpio -ovc 1024 | rsh somehost \
dd=/dev/tape obs=32768
```

*Restore files via the network*

◆ To restore files across NFS from tape on another machine to directories on your machine:

Example 1:

```
rsh -n somehost dd if=/dev/tape bs=32k | tar \
-xvpBf
```

Example 2:

```
rsh -n somehost dd if=/dev/tape bs=32k | cpio \
-idmuvc
```

## SYSTEM ENVIRONMENT

In particular, the system environment consists of the variables maintained by the shell. These variables may be set from a system file or from a file in the user's home directory, or with a command.

In a more general sense, the system environment consists of all login scripts and preference files that affect behavior of the applications and shells of a login session. There are generic system files that are used automatically unless a customized file exists in the user's home directory.

In the most general sense, system environment refers to all system-wide settings and running processes that both support and affect a login session. Examples are date and time settings, language settings, WISh2 windowing shell configurations, login shell preferences, network support processes, printer support processes, and run level settings. This section describes the generic system environment and how it can be customized—both for the system in general, and individual users in particular.

### Setting the Date and Time

To set the date and time of the internal system clock, log in as superuser and use the **date(1)** command. If you want to set the time forward, using the **-a** option allows it to catch up slowly and not disturb the background process.

To set the time back, bring the system down to single-user mode before setting the date and time. If you change the date and time while in multiuser mode, the background **cron** process will try to catch up and may run unwanted processes.

### System Processes

A process is any computer program running on the system. Many processes run simultaneously in a multitasking system such as Atari System V. Processes execute independently yet may communicate with other processes, with the console and the keyboard, and with other system resources. Multiple instances of a program may be run simultaneously. For example, an independent WISh 2 process is running for each user using WISh 2. Each process is uniquely identified by a number called the process identification (PID).

Most processes running on Atari System V in multiuser mode are associated with users at a terminal. Administrative processes and daemon processes are referred to as system processes.

An administrative process may perform tasks that affect users; i.e., logging on, formatting disks, setting up new accounts, and managing file space.

A daemon process is not associated with a user, but performs system–wide tasks, such as scheduling the printer, communicating with the network, process scheduling, and managing internal memory.

The **ps(1)** command is used to obtain information about processes that are running at that moment. The names and functions of active processes expected in a normal running Atari System V system are shown in Table 3-3 (command **ps -e**). If your system is not attached to a network, the network support processes will not be running.

The **kill(1)** command sends a signal to one or more processes. The **kill -2** command (INT) is an interrupt signal; **kill -15** (TERM) is the software termination signal, and **kill -9** (KILL) sends the exit signal. Use **kill -2** and **kill -15** at least twice before resorting to **kill -9**.

Example:

```
kill -9 839
```

will cause the active PID 839 to exit.

### Run Levels

A run level designates a particular group of processes automatically started by the system. Atari System V boots to the default run level indicated in the */etc/inittab* file. It is set to come up to run level 4.

**Table 3-3 Active System Processes**

| ID | Command | Process |
|---|---|---|
| | | **System** |
| 0 | **sched** | Scheduler |
| 1 | **init** | Parent of all user processes |
| 2 | **pageout** | Page daemon |
| 3 | **fsflush** | File system daemon |
| 4 | **kmdaemon** | Kernel memory allocator daemon |
| | | **Network** |
| | **inetd** | Internet super-daemon. Starts other internet daemons on demand: **rlogind**, **ftpd**, **telnetd**, etc. |
| | **listen** | Network listener |
| | **lpNet** | Network copy of lpsched |
| | **rpcbind** | RPC (remote procedure call) |
| | **nfsd** | NFS daemon (usually four copies) |
| | **biod** | NFS block I/O daemon (usually four copies) |
| | **mountd** | NFS mount daemon |
| | **statd** | NFS status daemon |
| | **lockd** | NFS lock daemon |
| | | **X Window System** |
| | **wish** | Windowing shell |
| | **wishd** | WISh2 daemon |
| | **wx** | Wx2 program |
| | **wxd** | Wx2 daemon |
| | **xfm** | XFacemaker 2 |
| | **X** | X server |
| | | **Other** |
| | **sac** | Service access controller |
| | **lpsched** | Line printer scheduler |
| | **cron** | Clock daemon |
| | **ttymon** | Terminal (tty) port monitor |

Available run levels are

- S single user mode/system maintenance mode
- 0 shutdown
- 1 system administration mode
- 2 multiuser networking mode
- 3 multiuser NFS mode
- 4 multiuser, networking X Window System mode (plus NFS, if available)
- 6 shutdown and reboot

### *Changing the Run Level*

Once the system is started and running at the default run level, that level can be changed with the command **init <run-level>**. When this command is entered, **init** scans */etc/inittab* and executes all matching entries. Several messages may be displayed; there will be a final message when the change to the new run level is complete.

▷ **Caution** When you change to a new run level, the activities of other users currently logged in may be disrupted.

To change to a lower run level, use the **shutdown** command.

Type

```
shutdown -i <run level> -g <warning time>
```

Example:

```
shutdown -i1 -g300
```

warns all users that in 300 seconds (5 minutes) the system will be brought down to run level 1 and shut down.

#### *Setting the Default Run Level*

When booted, Atari System V starts the **init** process (*/sbin/init*)). This process reads the */etc/inittab* file to determine the run level to come up to, and then runs all the indicated initialization scripts and commands listed. Each line in */etc/inittab* is an entry containing four fields separated by colons. The line with *initdefault* in the third field is known as the *initdefault* entry; it specifies the default initial run level in its second field.

As superuser, you can edit */etc/inittab* to change the *initdefault* entry to a different default run level. The change to *inittab* takes effect on the next boot.

### Localization and Internationalization

All Atari System V application programs will support multilingual operation. That is, all communications between the program and the user can be in the language of that user. By default, this is the system-wide language that is set in the generic login file */etc/globals*.

To change languages, set the LANG environment variable in the login script of the home directory. For example, run a script that sets $LANG from the *.xdmsession* script. See the **environ(5)** manual page for detailed description of LANG and other shell environment variables. See "Adding a New Language" in Chapter 4 for a list of supported native languages.

### Installing a New Application

Use the Product Install window to install new software application onto the hard disk. The product installation tool needs only to know the device from which to install the product. The product must have been packaged using the Application Packaging tools. Refer to the "Application Packaging" section of Chapter 4.

1. **Select the Install icon from the WISh tool tray panel.** 

   The Product Installation window pops up (Figure 3-1).

2. **Check the display of available disk space to make sure it will hold the application.**
3. **Select the installation device icon.**
4. **Insert the product installation media into the appropriate device. (For example, insert diskettes in the diskette drive. )**
5. **Select the Start button to start product installation.**

   A progress window opens (Figure 3-2) to display product installation status. If required, it requests input for you to type into this window. When installation is complete, a message indicating success is displayed.

6. **When the product installation is complete, press <Return>.**

**Figure 3-1**
**Product Installation Window**

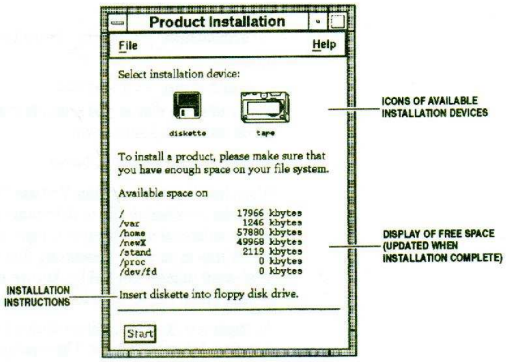
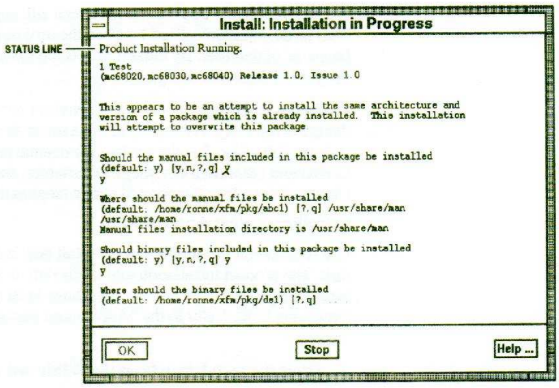



**Figure 3-2**
**Progress Window**




The display of free space is updated.

### Installing a New Icon

To install a new icon in the system-wide default WISh2 tool tray,

1. **Drag the new tool, file, or directory into your (superuser) tool tray with the Save File Positions option enabled.**

   This will save a new personal file for the superuser.

2. **Copy this file (*.wish_tools*) into system directory */usr/lib/X11/wish/tools*.**

   Thereafter, when users log in without a personalized configuration, the system-defined tool tray will come up.

In any case, users can drag the new tool from a directory into their individual WISh2 tool trays to make it more accessible.

### X Window System Session Management

Atari System V is set up to handle login as follows:

- **xdm** brings the system up for the X Window System. It then allows you to log in.
- **xdm** looks in your home directory *.xdmresources* for a list of resources that is read by **xrdb**. By default this is absent, which means the system default resources will be used.
- **xdm** looks in your home directory for the *.xdmsession* file and executes its contents using your login shell; if the file is absent, the system *.xdmsession* file is used.
- The default *.xdmsesssion* runs **mwm** and brings up the WISh2 Shell. When you exit the WISh shell, you are logged off automatically.

### Preference Files

To customize the behavior of the Motif window manager (MWM) or ot WISh2, the following files, if present, are read from the user's home directory.

- *.mwmrc*—Read instead of the system file when bringing up the MWM. Copy the system file in */usr/lib/X11/system.mwmrc* and make your own changes.
- *.Xdefaults*—Read by all X programs when started, except when resources are set with **xrdb** (such as those in the *.xdmresources* file). You can create *.Xdefaults* using X documentation for individual X programs.
- *.wish_options*—Applied instead of system file (*/usr/lib/X11/wish/options /.wish_options*). It cannot be created directly. WISh2 creates this file if the Save File Positions option is enabled. It saves the current behavior of file and window operations as defined by **File/Configuration/File Options...** and **File/Configuration/Window Options...** menu commands.
- *.wish_panels*—Applied instead of system file (*/usr/lib/X11/wish/panels /.wish_panels*). It is not directly creatable. WISh2 creates this file if the Save File Positions option is enabled. It saves the current window layout and will restart with this configuration whenever WISh 2 is started.
- *.wishrc*—Applied after the system file (*/usr/lib/X11/wish/classes/.wishrc*). It defines classes of icons and utilities. Your system manager will provide a standard system-wide definition of classes. See the Non Standard Logics, *WISh2 User's Manual,* "Configuring WISh2," for details.
- *.xfm.startup*—Preference file for XFaceMaker 2. Cannot be created directly, but by the XFaceMaker 2 Save Preference menu command.

### Command Shells

A shell is a command interpreter that allows communication with the operating system. The WISh 2 windowing shell provides a graphical interface to those commands.

In addition to the WISh 2 windowing shell, three command-line shells are available for communicating with Atari System V. Use them in command

windows, xterm windows, and in single-user mode where an operating system prompt is available.

The shell names and the names of the initialization files they look for are listed below. If the file is not present, the shell continues in a default manner defined by the system-wide initialization files.

- Bourne shell(sh)
  - executes system-wide profile */etc/profile* each time a login shell is started
  - executes the *.profile* file in the user's home directory each time a login shell is started
- C shell(csh)
  - executes */etc/login* each time a login shell is started
  - executes the *.login* file in the user's home directory each time a login shell is started
  - executes the *.cshrc* file in the user's home directory each time a C shell is started and each time a login shell is started
  - executes *.logout* file in the user's home directory each time a C shell is exited
- Korn shell(ksh)
  - executes system-wide profile */etc/profile* each time a login shell is started
  - executes the *.profile* file in the user's home directory each time a login shell is started
  - executes the file specified by the environment variable ENV each time a Korn shell is started and each time the login shell is started

## FILE SYSTEMS

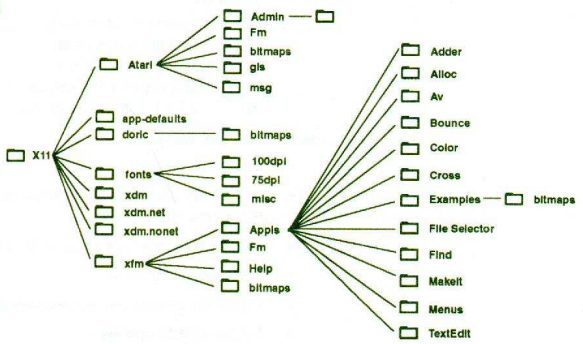
The main file system is called the root file system, and is usually depicted as a slash (/). Other file systems can only be accessed by users if they are mounted. That is, if the other file systems are logically connected to the root file system or a file system already mounted on the root file system.

All Atari System V information is stored within a directory tree that is a hierarchical structure of files. Some files are regular files in that they contain information, such as a letter, a report, an executable program, or a database data file. Others are directories that contain listings of files. Still others are special files used for various kinds of communication; most users rarely need to use special files. A directory tree for the X11 directory in */usr/lib* is depicted in Figure 3-3.

Atari System V files are located logically by type and function in the following standard directories:

- Configuration files are located in */etc*
- Log files are located in */var*
- Single-user administrative binaries are located in */sbin* or */usr/sbin*
- Other binaries are located in */usr/bin* and */usr/local/bin*.

**Figure 3-3**
**File Tree**
**Structure**




- In some cases the *AT&T System V, Release 4* documentation misrepresents the location of configuration and binary files that traditionally have resided in */etc*. In that documentation, configuration files that exist in */etc* are listed in */sbin*, */usr/sbin*, or a comparable directory containing binaries, and binaries that exist in */sbin* and */usr/sbin* are listed in */etc*.

  The Atari System V on-line manual pages do accurately reflect the location of the binary and configuration files.

Some directories directly below root are mountable file systems. A file system is a tree of directories and files stored as a single logically contiguous block of data on a storage device. For instance, a hard disk may be partitioned into several file systems.

The **mount** command associates a file system stored on a disk with a mount-point directory in the directory tree. A mount point is usually an empty (or near empty) directory used for providing an access point to a file system. A file system is named for its mount point directory. Refer to AT&T, *System V, Release 4, System Administration Guide*, Chapter 5, "File System Administration." Mountable file systems contained in Atari System V are shown in Table 3-4.

**Table 3-4**
**Mountable**
**File Systems**

| File System | Type | Description |
|---|---|---|
| / | ufs | The **root** file system contains system files, including executables and some configuration files |
| /stand | bfs | Bootable kernels and related files |
| /var | ufs | Variable system–wide information |
| /home | ufs | Personal files and application software |
| /fsck | ufs | Special mini–root for file system consistency checks |

Several types of file systems are supported by Atari System V. Each type has its own internal protocol, special features, and unique options. For more information, refer to the "Preface" of the AT&T, *System V, Release 4, Network User's and Administrator's Guide.*

The file system types are

| | |
|---|---|
| s5 | System V file system |
| ufs | Berkeley file system |
| bfs | boot file system V.4 |
| nfs | Sun network file system |
| rfs | AT&T remote file system |

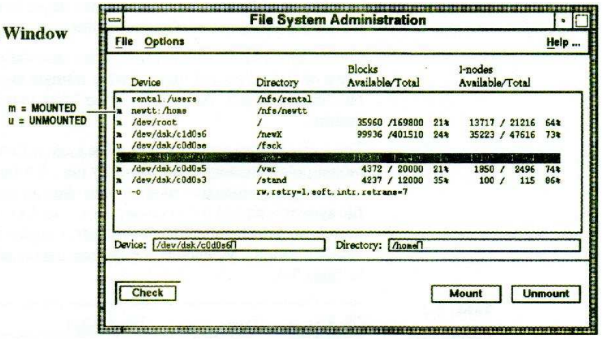
Use the Atari tool **filesys** to

- Mount file systems
- Display the status of file systems, including the relationship among fileservers and directories
- Display the amount of space available on each file system
- Unmount file systems
- Check file systems for internal consistency
- Set file system options

Only the person with superuser privileges can use this tool.

1. **Select the File System icon**  . The File System window shown in Figure 3-4 pops up.

**Figure 3-4**
**File System Window**



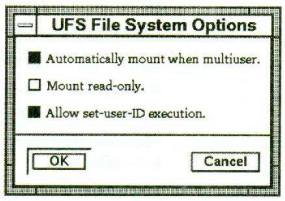

2. **Select a file system, or type the name of a file system in either the Device text box or the Directory text box, as appropriate.**

   When only one file system is selected and it is unmounted, you can

   - Change the mount directory by typing a full pathname in the Directory text box.
   - Change the mount options by means of the Options menu (see Figure 3-5 and Figure 3-6).

The UNIX File System (UFS) options window (Figure 3–5) pops up if the selected file system is on your hard disk.

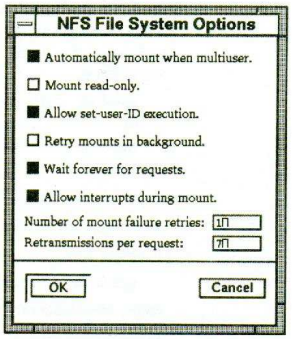
**Figure 3-5**
**UFS File System Options**




The Network File System (NFS) options window (Figure 3–6) pops up if the file system is remote.

Window options for the other file system types will become available as they are implemented.

**Figure 3-6**
**NFS File System Options**




When one or more file systems are selected, you can

- mount, using the **Mount** button,
- unmount, using the **Unmount** button, and
- check, using the **Check** button (must be unmounted).

### Maintaining File Systems

Without the File System tool you will need a summary of commands used to mount, unmount, monitor, and check file systems.

#### *Mounting*

To mount an existing file system to make it available for use, use the **mount(1M)** command. Most standard file systems are automatically mounted during system startup. See the manual pages for **mnttab(4)** and **vfstab(4)**.

To specify the type of file system, use the **–F** option. Type

```
mount -F <file_system_type> <device> <mount_point>
```

Example 1:

```
mount -F ufs /dev/dsk/c0d0s6 /home
```

where device */dev/dsk/c0d0s6* is mounted on */home* as a ufs file system type.

Example 2:

```
mount /home
```

The file system type and device are listed in */etc/vfstab* and therefore do not need to be listed here.

Example 3:

```
mount -F nfs -o retry=4,intr newtt:/home /mnt
```

mounts an nfs file system. The **-o** shows options. The remote system **newtt** has its home directory mounted on **/mnt**

#### *Unmounting*

To unmount a file system that is mounted on the system in order to remove it from use, use the **umount()** command. (See the manual page for **mount(1M)**). Most standard file systems are automatically unmounted during system shutdown procedures. The **mount** command with no arguments lists all mounted file systems.

```
umount <directory>
```

or

```
umount <file system>
```

Example 1:

```
umount /mnt
```

where */mnt* is the mount point directory for a file system.

Example 2:

```
umount /dev/dsk/c0d0s1
```

where */dev/dsk/c0d0s1* is the resident device for a file system.

#### *Monitoring Disk Usage*

To find out how disk space is being used, use the **df** command. This command requires the **-F** option to specify file system type, if the file system you want information about is unmounted, and if it is not found in */etc/vfstab*. See the manual pages for **df(1M)** for a list of generic options, along with options specific to each file system type.

#### *Checking*

To check, and possibly repair, unmounted file systems to ensure data integrity, use the **Check** button in the File System Administration window to invoke the **filsys** window (Figure 3-7).

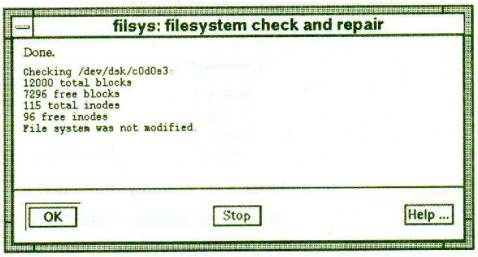
Type the **fsck** command, using the **-m** flag to determine whether a file system needs checking. Type

```
fsck -m -F <file system type> <file system>
```

Example:

```
fsck -m -F ufs /dev/dsk/c0d0s1
```

**Figure 3-7 File System Check Window**




This command requires the **–F** option to specify the file system type, if not found in */etc/vfstab*. See the manual pages for **fsck(1M)** for a list of generic options, along with specific options for each file system type.

▷ **Caution** Because the root file system cannot be unmounted, refer to it as */dev/rroot* (note the extra r) when checking it, and reboot immediately afterward. Otherwise, the in-memory copy of the superblock will later be written to the disk during a periodic update, undoing some of the changes just made and possibly leaving the file system in an inconsistent state.

## PERIPHERAL DEVICES

Device drivers are present and enabled for the peripheral devices shown in Table 3-5. In general, all that must be done is to plug the device in. (Refer to the owner's manual for your computer.)

**Table 3-5 Peripheral Devices**

| External Ports | Supported Devices |
|---|---|
| Console | 19-inch monochrome Atari monitor |
| Keyboard | Atari TT intelligent keyboard<br>Three-button mouse |
| SCSI | Disk drives<br>Tape drives |
| Floppy disk | 720KB diskette |
| Parallel port (Centronics) | Printer |
| Two serial ports (Zilog SCC)<br>Two serial ports (MFP) | Printers<br>Dumb terminals<br>Modem |
| VME | Ethernet Network |

Additional device drivers may be written and included in the system. See Chapter 4, "Application Development."

### Serial Port Configuration

Atari System V is preconfigured with the following port assignments:

| External Port | Device Path | Device | Service |
|---|---|---|---|
| Modem 1 | */dev/term/m1* | Terminal | login |
| Modem 2 | */dev/term/m2* | Modem | uucp |
| Serial 1 | */dev/term/s1* | Terminal | login |
| Serial 2 | */dev/term/s2* | Terminal | login |

### Serial Port Management

Access to the computer from remote devices, including terminals and modems, is supported by the Service Access Facility (SAF). The SAF replaces both the *getty* and the *listen* processes on older releases of UNIX systems. The SAF consists of several layers. The top layer is the Service Access Controller (SAC) process which oversees the service machine that manages several services, including terminal port monitoring and network access for UUCP and other nonlogin services.

The SAC administrative command is **sacadm** which

- adds or removes port monitors,
- starts or stops a port monitor,
- enables or disables a port monitor, and
- prints requested port monitor information.

The SAC lower level administrative command is **pmadm**, which facilitates the servicing of port monitors. The **pmadm** command

- adds or removes a port service,
- enables or a disables a port service, and
- prints requested port service information.

### Port Monitor

The SAC program, started from **inittab**, consults a table of enabled services and starts service-specific daemons to monitor ports in their specific domain. Typically, SAC starts **inetd**, the basic Ethernet daemon, and **ttymon**, the terminal port monitor daemon. These services in turn start specific handler processes, such as **login**, when activity is detected on a device under their management.

The **ttymon** process is responsible for monitoring serial ports and, when activity is detected, invoking a service configured for that port, usually **login**.

The **ttymon** process has three main functions:

- It initializes and monitors **tty** ports.
- It sets terminal modes and line speeds.
- It invokes the service associated with a port whenever it receives a connection request.

Port services determine the behavior of a connection and, in the case of serial ports, the service is usually terminal login, which prints a login banner on the connected terminal and controls the login process.

### TT Port Monitor Configuration

The TT is delivered with the following port monitor configuration:

- **ttymon1** supports modem port 1 and serial port 1, providing login service
- **ttymon2** supports modem port 2, providing login service and modem support
- **ttymon2** supports serial port 2, providing login service

The port monitors **ttymon1** and **ttymon2** are started and enabled when the TT enters multiuser state (initialization states 2, 3, and 4). Services supported by the port monitors (login) are now available to the ports upon an incoming connection request.

Under most operational conditions, the TT port monitor configuration does not require maintenance, except for disabling of a **ttymon** port service to accommodate connection of a serial printer.

For example, the command

```
pmamd -d -p ttymon2 -s s2
```

disables the serial port 2 login service in order to connect a serial printer.

The service access command

```
pmadm -e -p ttymon2 -s s2
```

enables serial port 2, the **ttymon2** login service.

Following are examples of adding a port monitor to the **ttymon** configuration to support a serial port (serial port 5 in this example).

```
sacadm -a -p ttymon5 -t ttymon -c \
      /usr/lib/saf/ttymon -v 'ttyadm -V'
```

Use the following service access command to add login service for service support of the serial port 5.

```
pmadm -a -p ttymon5 -s s5 -i root -fu -v \  'ttyadm
-V' \
-s /usr/bin/login -m ldterm -p \" login:\'"'"
```

Start and enable the new port monitor **ttymon5** and port monitor service for service support of serial port 5 as follows:

1. **Start port monitor, ttymon5**

   ```
   sacadm -s -p ttymon5
   ```

2. **Enable port monitor, ttymon5**

   ```
   sacadm -e -p ttymon5
   ```

3. **Enable port monitor service, ttymon5**

   ```
   pmadm -e -p ttymon5 -s s5
   ```

For more complete information on port management, refer to the AT&T, *UNIX System V Release 4, System Administrator's Guide*, Section 13, "Service Access."

### Serial Port Printer Configuration

One or more of the serial ports may be connected to a serial printer. The simplest method of connecting a serial printer is by making a direct connection to the selected serial port.

To configure the system software to allow operation of a serial printer, disable the port service associated with the selected serial port and enter the appropriate **lpadmin** commands.

For example, the steps necessary to configure serial port 2 for serial printer operations are

1. **Disable serial port 2 port service.**

   ```
   pmadm -d -p ttymon2 -s s2
   ```

2. **Make the physical printer connection to serial port 2.**

3. **Add printer name to lp service**

   ```
   lpadmin -p <printer name> -v <pathname>
   ```

   Example:

   ```
   lpadmin -p laserjet -v /dev/terms/m2
   ```

4. **Tell the LP print service to accept the printer.**

   ```
   accept laserjet
   ```

5. **Tell the LP print service to enable the printer.**

   ```
   enable laserjet
   ```

For more information on printer configurations and print services, refer to the AT&T, *UNIX System V Release 4, System Administrator's Guide*, Section 9, "Print Service."

### Serial Port Modem Connection and UUCP System Support

The TT is delivered with modem port 2 configured to support UUCP communications services, modem connection, and direct TT-to-TT connections, which require that a connection method to a remote system be specified and supported in its database files. The information required would include data specific to your connection method and that of the remote machine you wish to connect to. The UUCP database files are located in the directory */etc/uucp*. Table 3-6 lists these files and gives a brief description of each one.

### Specifying a Connection Method to a Remote Machine

The UUCP files and file contents necessary to establish a connection to a remote machine with a Hayes-compatible modem are as follows:

- *Devices*

```
ACU term/m2,M-2400 hayes
ACU term/m2,M-1200 hayes
ACU term/m2,M-300  hayes

Direct term/m2,M-300 direct
Direct term/m2,M-2400 direct
Direct term/m2,M-1200 direct
```

**Table 3-6 UCCP Database Files**

| UUCP File | Description |
|---|---|
| *Config* | Allows you to override some parameters used by the *uucp* protocols. The defaults are always sufficient. |
| *Devconfig* | Allows you to override some parameters used by devices other than modems. The defaults are almost always sufficient. |
| *Devices* | Lists the devices present on your system and specifies how to manipulate those devices. You edit this file when adding new *uucp tty* lines. |
| *Dialers* | Describes how to dial a phone number on types of modems connected to the system. You edit this file when adding a new type of modem. |
| *Dialcodes* | Gives symbolic names for phone number prefixes to be used within the *Systems* file. Thus numbers can be shortened and a common *Systems* file can be shared by multiple machines. |
| *Grades* | Permits jobs to be partitioned into multiple queues of different priorities. The defaults provide for three priority grades, which is usually sufficient. |
| *Limits* | Allows you to limit the maximum number of *uucico*, *uusched*, and *uuxqt* processes that may run simultaneously. |
| *Permissions* | Describes access rights for other systems that call your machine and that your machine calls. |
| *Poll* | Used by *uudemon.poll* to determine what times a system will be polled. This is useful for systems that cannot call your system. You edit this file when setting up polled sites. |
| *Systems* | Describes the systems known to *uucp* and how to connect to each one. You edit this file when you add a system to be contacted. |
| *Sysfiles* | The *Sysfiles*, *Devices*, and *Dialers* files can consist of multiple files. The *Sysfile* contains references to these files so that UUCP knows where to look for system information. |

- *Dialers*

```
hayes =,-, "" \dAT\r\c OK\r \EATDT\T\r\c \
CONNECT\m
```

- *Systems*

```
my_sys  Any ACU 1200 - "" \d ogin:--ogin:--ogin:
nuucp
rdd2    Any ACU 1200 - "" \d ogin:--ogin:--ogin:
nuucp
wooky   Any Direct 1200 - "" \d ogin:--ogin:--ogin:
nuucp
```

- *Sysfiles*

```
service=cu    systems=Systems \
              devices=Devices \
              dialers=Dialers

service=uucio systems=Systems.cico:Systems \
              devices=Devices.cico:Devices \
              dialers=Dialers.cico:Dialers
```

To summarize, communication services

1. Make connections by consulting *sysfiles* to determine the files to use for other operations.

2. Consult the appropriate *Systems* file to find out what kind of connection to use for the call.
3. Follow the device name (third field of *Systems*) into the *Devices* file, where they determine what hardware implements that connection method.
4. Follow the fifth field of *Devices* into the appropriate *Dialers* file, to determine how to talk to that specific device.
5. When the call is made using the *Devices* and *Dialers* information, *cu* returns control to the keyboard, or *uucico* goes back to the *Systems* entry and logs in to the remote machine.

When these steps are complete, the connection is made and data communication begins.

Use the following examples to make a remote connection based on the foregoing configuration information.

1. **Copy *myfile* from your local machine to the remote machine, *rdd2*.**

```
uucp /home/john/myfile \
rdd2!/var/spool/uucppublic/myfile
```

2. **Make a direct connection from your machine to the remote machine, *wooky*.**

   This connection can be made via dial-up telephone lines, through a hard-wired data link, or across a local area network.

```
cu wooky
```

For more information on UUCP file configurations and remote connections, refer to the AT&T, *UNIX System V Release 4, System Administrator's Guide,* Section 7, "Network Services."

## ADDING A NEW TERMINAL TYPE

Before adding a new terminal type, be sure that your language choice is properly displayed so that your responses are interpreted as you intend.

Atari System V supports internationalization on text-based terminals. The TERM and LANG environment variables enable the appropriate character-mapping file.

- Set the TERM environment variable to that of the terminal
- Set the LANG environment variable to that of the language.

Atari System V internal code set is ISO 8859-1. A terminal that uses a different code set or that uses only part of this code set requires a character conversion between the terminal and the system.

The GLS software mapping module translates between the terminal STREAMS device and the system. This module uses the binary character translation table indicated by the TERM and LANG variable settings.

Atari System V provides a binary character translation table for the TT console. This table is installed the *codeset* directory *(/usr/lib/codeset/$LANG)* for each supported language.

The **codesio** command creates binary character translation tables from ASCII code set mapping source tables. Code set source is found in the */usr/lib/codeset/$TERM_mappings* files, where *$TERM* is one of the following:

- TT
- VT100
- VT52
- ST52

To install the VT100, the VT52, or the ST52, use the **codesio** command to compile the ASCII source into object file format. Install the resulting object file (*$TERM*) in the code set directory *(/usr/lib/codeset/$LANG)* for each language.

Example:

```
codesio -f vt52_mappings -c GERMAN-MAIN -l \ german
-t vt52
```

The **-f** flag identifies *vt52_mappings* as the input source table file; **-c** selects the codeset *GERMAN–MAIN* within the file *vt52_mappings*; **-l** sets the language name within the object file to German; and **-t** names the output object file and sets the terminal name within the object file to be *vt52*.

If characters are mapped identically, you can link to an existing code set for the same terminal type in another language directory.

Example:

```
ln /usr/lib/codeset/english_usa/vt52 \
/usr/lib/codeset/english_UK/vt52
```

To attach a terminal type that is not listed above, you must create a **codeset** mapping source table for the terminal type. Copy an existing source table from the directory */usr/lib/codeset* and use your favorite editor to customize it for the new terminal type. See the **codesio(1M)** manual page for required content and format.

Generate a new object file using **codesio** as described above. Test it using the command **glscnv**, which will output details of how each character will be converted. When debugged, install the new object file in the *codeset* directory for each language as described above.

Thereafter, this object file is loaded when the combination of LANG and TERM is set.

❑ All X fonts used by Atari System V are ISO 8859-1 fonts.

### ADDING A NEW DISK DRIVE

After installing a new external disk drive you should prepare it to store information by initializing it; that is, organizing its storage space and creating file systems.

❑ Most of the following commands are Atari-specific. Refer to the on-line manual pages for detailed descriptions of commands.

1. **Turn on your system and verify that the new disk is installed.**

   Watch the boot text. If you don't see an additional SCSI device entry, check the hardware and reboot.

**2. Format the new disk with the `format` command**

A disk partition is derived using the formula **cXdYsZ**, where

| | | | | |
|---|---|---|---|---|
| X | = | controller | = | SCSI ID |
| Y | = | drive | = | SCSI LUN (always 0) |
| Z | = | slice | = | partition number (in hex: 0–f) where f refers to all partitions |

**cXd0** refers to the whole disk and is equivalent to **cXd0sf**.

Example:

```
format -f /dev/rdsk/c3d0sf
```

In this example, the SCSI ID of the disk being formatted is 3 (refer to the **format(1M)** manual page).

**3. Install an Atari partition table and boot sector.**

An Atari partition table may have a maximum of four entries, only one of which may be an Atari System V entry. A volume table of contents (VTOC) is used by Atari System V to keep track of its internal partitions (see step 5).

The Atari System V partition table:

```
Partition ID No. 0,1,2,3

Start     First block of the partition

Size      Size of the partition, in blocks

ID        Tag identifying partition type:
             UNX = Atari System V
             GEM = Atari TOS

Flag      Flag identifying the partition:
          St-boot = TOS boot partition
          Unix-boot = Atari System V boot partition
```

Block 0 contains the Atari partition table that has entries describing the layout of the disk. Each entry describes one disk partition.

The first example and explanation below describes how to create a disk that contains only Atari System V; this is followed by an example of a disk that is divided between Atari System V and TOS.

Example 1–Atari-only partition:

```
partinit -Ibim -o /dev/rdsk/c3d0sf
```

*`-I` option clears block zero*

**`-I`** clears block zero, destroying existing Atari partition tables

*`-b` option identifies bootable disk*

**`-b`** identifies the disk as bootable; set this option in case you ever want to boot from the disk.

*`-i` option installs bootstrap files*

**`-i`** installs bootstrap files; set **-i** in case you ever want to boot from the disk and to have a backup copy of the bootstrap files.

*`-m` option requests interactive prompting sequence*

**`-m`** requests the interactive prompting sequence. **Partinit** displays the size of the disk in sectors (512-byte units).

*`-o /dev/rdsk/c3d0sf` specifies device*

**`-o /dev/rdsk/c3d0sf`** specifies the device on which to install the modifications; this must reference the whole physical disk.

This command asks the questions displayed in the box below; you supply the answers.

```
Current Physical size 415436
Nonbootable disk

Which Partition? 0
Flags, one of: St-Boot, Delete, Unix-boot, Non-Boot? u
Id message ()? UNX
Start sector (0)? 1
Size (0)? 415435
Which Partition? <Return>
Partition  Start   Size       Id
0          1       415435     UNX     unix-boot
Which Partition? q
Writing sector 0
```

Explanation:

*`partinit` prompts for a partition number*

```
Which Partition? 0
```

| | |
|---|---|
| **0–3** | lets you modify one disk partition entry |
| **<Return>** | prints out the current partition table |
| **q** | writes changes and quits |
| **<Control D>** | abandons any changes and quits |

You can create up to three partitions, numbered 0 to 3. In this example you will modify partition 0, the first partition.

*`partinit` prompts for a flag*

```
Flags, one of: St-Boot, Delete, Unix-boot,
Non-boot? u
```

| | |
|---|---|
| `Delete` | removes this entry from the partition table |
| `Non-Boot` | unsets current bootable status of the selected partition |
| `St-Boot` or `Unix-boot` | sets the partition to be bootable for the specified operating system |

This will be a bootable Atari System 5 partition; therefore select **u** for `Unix-Boot`. For any other choice, type the first letter using either upper or lowercase.

*`partinit` prompts for an ID*

```
Id message ()? UNX
```

Type **UNX** for the partition that contains Atari System V or **GEM** for partitions that contain TOS.

*`partinit` prompts for the first sector in the partition*

```
Start sector (0)? 1
```

The **`partinit`** information is stored in sector 0. If there will be at least one TOS partition, start the first one in Sector 2; otherwise, start it in Sector 1.

*`partinit` prompts for the number of sectors in the partition*

```
Size (0)? 415435
```

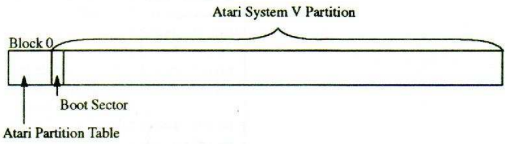
***Return** displays the partition table*

Since one sector is reserved for **partinit** data, `Size` is the current physical size minus 1.

```
Which Partition? <Return>
Partition    Start   Size     ID
0            1       415435   UNX    unix-boot
```

Pressing <Return> displays a printout of your entries. Type **q** to save the data to disk. If you made a mistake along the way, type **<Control D>**.

The responses in this example allotted the entire disk to one bootable Atari System V partition. The resulting disk layout looks like this:




Example 2–Atari/TOS-shared disk:

The command

**partinit -iIbm -o /dev/rdsk/c3d0sf**

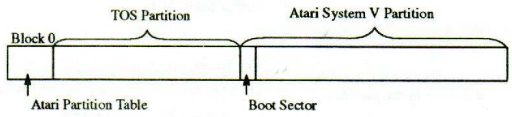
asks the same questions as the previous example:

```
Current Physical size 415436
Nonbootable disk
Which Partition? 0
Flags, one of: St-Boot, Delete, Unix-boot, Non-Boot? n
Id message ()? GEM
Start sector (0)? 2
Size (0)? 21674
Which Partition? 1
Flags, one of ST-Boot, Delete, Unix-Boot, Non-Boot? u
Id message ()? UNX
Start sector (0)? 21676
Size (0) 393760
Which partition? q
Writing sector 0
```

Explanation:

In this example sectors 2 through 21675 are reserved for TOS, and the rest of the disk is reserved for Atari System V. The Atari partition is bootable.

If a TOS partition was created, as in Example 2 above, it would be located immediately after block 0 as follows:



Refer to the on-line **partition(1M)** manual page for instructions for specifying all parameters on the command line or in a data file, rather than interactively.

4. **Add a *pdsector* (physical disk sector) to the disk.**

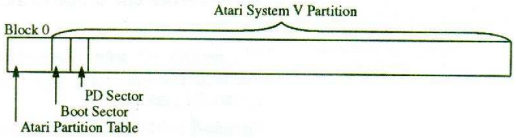
   The ***pdsector*** is a block that describes the physical layout of the disk.

   Example:

   ```
   format -w /dev/rdsk/c3d0s0
   ```

   The **-w** flag does not format the disk; it writes the ***pdsector*** to the disk.Refer to the **format(1M)** manual page.

   The disk layout now looks like this:




5. **Create the VTOC for the Atari System V partition.**

   The VTOC divides Atari System V into separate partitions, some of which will be file systems and others which will be swap space or used for other types of disk functions.

   **setvtoc –i /dev/rdsk/c3d0s0** sets an initial VTOC on the disk; one mountable partition the size of the allocatable disk with a generic name.

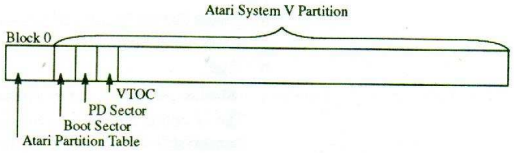
   **prtvtoc /dev/rdsk/c3d0s0 > vtocdata** prints the default VTOC into a file named *vtocdata* which you can then edit.

   **vi vtocdata** edits the file to set up partitions as you want them.

   **setvtoc –i /dev/rdsk/c3d0s0 –s vtocdata** command installs information from *vtocdata* back into the VTOC on the disk.

   Refer to the **setvtoc(1M)** and **prtvtoc(1M)** manual pages. The **setvtoc** manual page describes the content and format of the VTOC information that you edited.

   The VTOC partitioning information heads the Atari System V partition as shown.

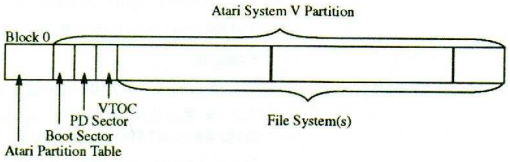



6. **Create one or more file systems using the mkfs command.**

   Example:

   ```
   mkfs -F ufs /dev/rdsk/c3d0s6 <options>
   ```

Refer to the **mkfs(1M)** generic manual page and manual pages specific to each type of file system for options.




7. **Mount the new disk to make it available on your system.**

   Example:

   ```
   mount -F ufs /dev/dsk/c3d0s6 /newstuff
   ```

   You must use **–F** to specify type if you did not specify it in the */etc/vfstab* file for this file system.

   **/newstuff** is the mount point directory to which you decided to attach this Atari System V partition. Refer to the **mount(1M)** and **mnttab(4)** manual pages.

8. **Edit */etc/vfstab* so these new file systems will be automatically mounted and unmounted on system start up and shutdown.**

   Following is a sample */etc/vfstab* with a new entry for /newstuff:

| #special | fsckdev | mountp | fstype | fsckpass | automnt | mntflags |
|---|---|---|---|---|---|---|
| /proc | – | /proc | proc | – | no | – |
| /dev/fd | – | /dev/fd | fd | – | no | – |
| /dev/root | – | /dev/rroot | ufs | – | no | – |
| /dev/dsk/c0d0s3 | /dev/rdsk/c0d0s3 | /stand | bfs | 1 | yes | – |
| /dev/dsk/c0d0s5 | /dev/rdsk/c0d0s5 | /var | ufs | 1 | yes | – |
| /dev/dsk/c0d0s6 | /dev/rdsk/c0d0s6 | /home | ufs | 1 | yes | – |
| /dev/dsk/c0d0se | /dev/rdsk/c0d0se | /fsck | ufs | 1 | no | – |
| /dev/dsk/c3d0s6 | /dev/rdsk/c3d0s6 | /newstuff | ufs | 1 | yes | – |

   Refer to the **vfstab(4)** manual page.

9. **Make new file systems available for mounting by remote systems.**

   Example: Make the */newstuff* file system accessible to remote systems via NFS:

   a. Ensure that the file */etc/dfs/fstypes* contains the word **nfs** as the first word on one of its lines.

   b. Type

      ```
      share -F nfs -orw /newstuff
      ```

      The **–F** option specifies the file system type by which this file system can be accessed. The *ufs* and *s5* file systems can be accessed by **nfs**.

      The –o rw option specifies that this file system is read–write via NFS.

      This example makes the entire */newstuff* file system available via NFS. Specifying */newstuff/src* would make a portion of the file system available to remote systems.

c. Add an entry to */etc/dfs/dfstab* so this file system will be automatically available to remote systems whenever **nfs** is started. That is, you will not have to type in the **share** command each time your system is booted, or **nfs** is started. Each line in */etc/dfs/dfstab* is a share command like the one issued in Step 9b.

**10. To access the */newstuff* file system via NFS, each remote system must now:**

a. Create a mount point directory to which the */newstuff* file system will be mounted. Choose a name that is meaningful on the remote system. For example: **mkdir /nfs/somehost**

b. Mount the */newstuff* file system. For example, **mount somehost:/newstuff /nfs/somehost**. In this example **somehost** is the name of the system on which the */newstuff* file system resides. The contents of */newstuff* are now available to the remote system by referring to files and directories in the */nfs/somehost* directory.

c. Add an entry to */etc/vfstab* so that the */newstuff* file system is automatically available when this remote system starts up.

Refer to the **share(1M)**, **dfstab(4)**, **sharetab(4)**, and **mount(1M)** manual pages.

## SYSTEM RECONFIGURATION

Atari System V should be reconfigured whenever one of the following occurs:

- The system parameters in the kernel, such as those for tuning, are changed.
- A software module is added to the system, with no change in the system hardware configuration.
- A software module is added to the system because of a change in the system hardware configuration.

For all of these circumstances, use the Atari tool Kernel Configuration from the System Administration Tools icon on the WISh desktop.

As superuser, **Select** the Kernel Configuration icon from the WISh2 tool tray . The Kernel Configuration window pops up (Figure 3-8).

Use this five-step sequence to reconfigure Atari System V:

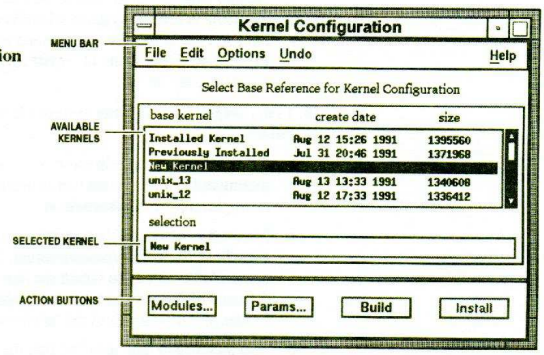
1. Select the base kernel to be changed.
2. Change the modules included in the kernel.
3. Change the kernel parameters.
4. Build the kernel.
5. Install the kernel.

Following is a detailed description of each of these steps.

**1. Select the base kernel to be changed.**

Available kernels are displayed on the main Kernel Configuration window. The chosen kernel is displayed in the selection box. **Select** one of the actions

**Figure 3-8 Kernel Configuration Window**




at the bottom of the window to use this kernel as the base reference kernel or to view its configuration.

There will be entries for all kernels for which *build* archive directories exist in the */usr/local/lib/kernels*.

At a minimum, the following specific kernel entries should be present:

- currently running */stand/unix*
- previously installed */stand/OLDunix*

The selected kernel may be **Installed** immediately. **Build** is disabled for a kernel that is already built. Use the **Select** button to double-click on a kernel entry to bring up both the Modules and Params windows for that kernel.

***Double click**: Use left (**Select**) mouse button to point to object and execute two clicks in quick succession.*

**2. Change the modules included in the kernel**

◆ Select the **Modules...** button from the Kernel Configuration window. The Modules window pops up (Figure 3-9).

Every module included in the selected base kernel is represented by an icon on the right side of the window. A scroll bar appears if there are too many icons to display.

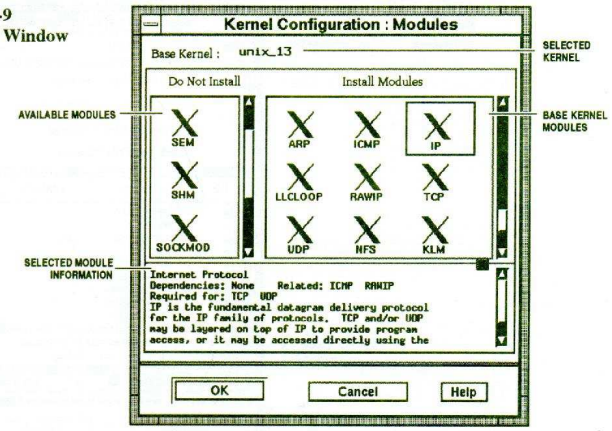
Other available modules are represented on the left side of the window. A scroll bar appears if there are too many icons to display.

- Double click, using the **Select** (left) mouse button, on a module's icon to move it to the other side.

If a module to be included has dependencies on modules not included, you will be warned before the module is moved to the included list. The dependencies are not moved automatically. Module dependencies are specified in the */etc/master.d* file on the first line of the *DEPENDENCIES/VARIABLES* column.

The Kernel Configuration tool knows only about the */stand/system* file: it does not read the */stand/edt_data* file, which has a list of all possible files.

**Figure 3-9**
**Modules Window**




Basically, */stand/system* along with the */etc/master.d* files, describes the software portion of the system configuration—the portion the Kernel Configuration tool deals with. The */stand/edt_data* file is a list of supported hardware devices; the system's built-in autoconfiguration support deals with checking for these devices and including support for them, if present.

- **Select** a module icon to view information about that module such as
  - Expanded name of module
  - Module description
  - Dependencies on other modules or parameters

  Change an icon by placing a bitmap file with the same name (in uppercase letters) as the module, plus the *.bm* extension in the directory */modules.d*; otherwise a default icon is displayed.
- **Select OK** to confirm and save changes
- **Select Cancel** to return without saving changes.

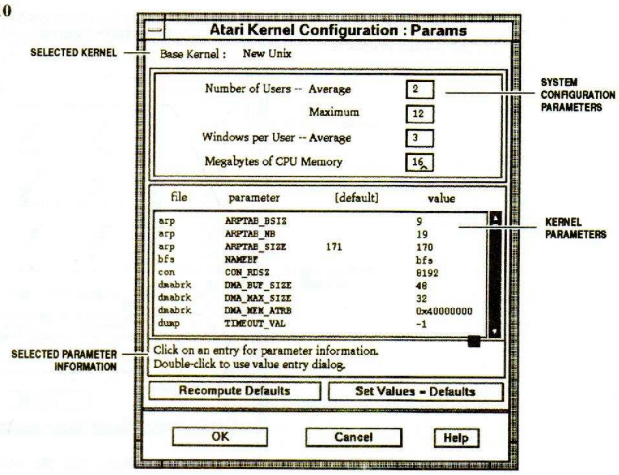
If changes are made, a new entry (**unix_[n+1]**) is added to the list of kernels in the main Kernel Configuration window and is automatically selected. Until it is successfully built, the name of this kernel appears in the list as `New Unix` and the **Install** button is disabled whenever this kernel is selected.

3. **Change the kernel parameters**

**Select** the **Params...** button from the Kernel Configuration window. The Params window pops up, as shown in Figure 3-10.

Values are displayed for the selected base kernel; these are reference values. If you change a system configuration parameter, **Select** the **Recompute**

**Figure 3-10 Params Window**



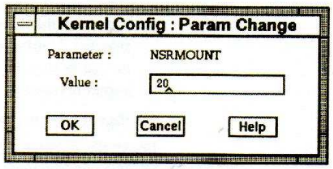

**Defaults** button to update the default column of kernel parameters. (Only the megabytes of the CPU memory entry have any effect on the default values in this release.)

The system configuration parameters are

- average number of users
- maximum number of users
- average number of windows per user
- megabytes of CPU memory

Double-click an entry in the list of kernel parameters to change its value. A pop-up dialog box appears showing the name of the selected parameter (Figure 3-11). The current value will be used to initialize the value entry field, or you may type in a new value. **Select OK** to keep the value or save a change. **Select Cancel** to close the pop-up dialog box without making any change.

**Figure 3-11 Parameter Change Dialog Box**




The `[default]` value is for reference only. If you are satisfied with all of the `[default]` values, click the **Set Values = Defaults** button to update

the **value** column. **Select** an entry in the list to get a helpful description of that parameter.

When all parameters have been set as you want them (all system configuration parameters and the `value` column of kernel parameters), **Select OK** to save the changes and exit the Params window.

Select **Cancel** to discard all changes and exit the Params window.

If changes were made, a `New Unix (unix_[n+1])` entry is added to the list of kernels in the main Kernel Configuration window and automatically selected. The Install button is disabled until this kernel is successfully built.

4. **Build the kernel**

   a. **Select** the **Build** button from the main Kernel Configuration window (refer to Figure 3-8).

      Module and Params windows must be closed before a build is allowed.

      **Build** creates a new version of the kernel, using the specified modules and parameter values. Messages are sent to an ***xterm execute*** window.

   b. To turn verbose mode off or on, **Select Options** from the menu bar and **Select** the **Enable Verbose Mode** check button, as preferred.

      After a successful build, the entry for the new kernel `(unix_[n+1])` is changed to `unix_[n+1]` and is selected.

   c. To remove the last kernel that was built in this session and is not yet installed, select **Undo** from menu bar, then select **Remove Last Built**.

5. **Install the kernel**

   a. **Select** the **Install** from the main Kernel Configuration window to install the selected kernel as the bootable file.

   b. To swap the currently running kernel with the previously installed *OLDunix*, **Select Undo** from menu bar, then **Select Reinstall Previous**. If there is no */stand/OLDunix*, this request will be rejected.

   c. To exit from Kernel Configuration, **Select File** from menu bar, then select **Exit**.

      If Module or Params changes were made, but no kernel was built, the changes are discarded.

### Reconfiguration Details

This section is a description of what occurs when the system automatically reconfigures itself.

*AT&T 3B2 Computer documentation does not apply to information in the "Reconfiguration Details" section of the Atari V System Developer's Guide.*

The **boot** code loads files from the */stand* partition only. The **boot** command scans the VTOC (*sys/vtoc.h*) for the *stand* partition then loads and executes the desired bootable kernel. Leave a working copy of a kernel in */stand*, perhaps named */stand/unix.old*, because errors in a driver may make the new kernel unbootable.

The boot program recognizes two types of boot: an auto boot and a demand boot. In an auto boot, the Equipped Device Table (EDT) probe programs are run and the date of the */stand/system* file is checked. If there is a mismatch between devices supported by the kernel and those found by probing, or if the system file

has been changed, the file */stand/mUNIX* is loaded and the system will run **cunix** to rebuild */stand/unix* when it comes up. In a demand boot, no such checking is performed.

If the initial kernel name in *bootname* is *unix*, an auto boot is performed; otherwise, a demand boot is performed. If the system maintenance mode is entered, a demand boot is assumed, unless otherwise requested.

The utility **cunix** builds kernels from a set of boot modules.

For each device driver or configurable part of the kernel there is a binary file called a boot module and a corresponding **master(4)** file. The core kernel is in the module named **kernel**. Each module has the same name as its master file, but in uppercase letters.

Boot modules are themselves produced by the utility **mkboot(1M)**, which combines a master file with either a driver object file or an existing boot module.

The **cunix** utility combines the boot modules together according to the specifications in a **system(4)** file.

A file **conf.s** containing the device switch tables, interrupt vectors, etc., is generated automatically by **cunix**, and assembled to produce **conf.o**. Finally, **cunix** invokes **ld(1)** to link the driver and software modules with **conf.o** to create a single, absolute unix.

Boot time probing is used to determine hardware configuration, using a set of probe programs installed in */stand*, one per device. The information is passed to **cunix** and used to determine the hardware modules to be included. Configuration details for each possible hardware device are listed in the EDT data file **edt_data(4)**.

Each configurable board or controller has an associated software probe routine (**EDTP_device**). The EDTP determines whether the device is present and functioning and, if it is, records information about the device in the EDT. The boot code loads all *EDTP_device* files from */stand* and executes them one-by-one.

When reconfiguration is necessary, the autoboot sequence looks like this:

1. EDT probe routines, executed by the boot code, generate an in-core EDT. The required devices **r** flag in */stand/edt_data* is added to the in-core EDT.
2. The boot code compares the in-core EDT with the EDT in */stand/unix*. If they are different (i.e., hardware driver added) go to step 4.
3. The code compares the modification date of */stand/unix* with that of */stand/system*. If */stand/unix* is older (if a software module was added or system parameters were modified), go to step 4; otherwise go to step 5.
4. Reconfiguration is necessary. The boot places */stand/system* in the *bootname* field of the in-core *bootargs* structure and boots */stand/mUNIX*. */stand/mUNIX* is a mini-kernel that contains the minimal set of modules necessary to reconfigure the kernel.

   Keep a copy of **mUNIX** in */stand* at all times. When the boot code invokes **mUNIX**, it passes along a pointer to the in-core *bootargs* structure. **mUNIX** creates a copy of this structure, making **bootname** available to other

programs, using the **sysm68k()** system call. A non-null **bootname** indicates reconfiguration is needed.

5. */etc/init* begins processing */etc/inittab* entries
6. */etc/ckmunix*, executed as specified in */etc/inittab*, gets **bootname** using **sysm68k()**. If **bootname** is non-null and */stand/noautoconfig* is not present, then rebuild the system; go to step 9.
7. (*/stand/noautoconfig* was present or **bootname** was null) */etc/init* brings the system to its default initialization state.
8. Boot continues. No reconfiguration is done.
9. Rebuild the system. */etc/ckmkunix* copies **bootname** to */etc/.sysfile* and executes */etc/buildsys -s*.
10. */etc/buildsys* runs */usr/bin/cunix* using the system file name from */etc/.sysfile*. The new kernel is placed in */unix*.
11. */etc/buildsys* reboots the system (*uadmin 2 1*) if *cunix* returns successfully.

## CHANGING THE BOOT PREFERENCE

Atari System V maintains an operating system boot preference in nonvolatile memory (NVRAM). The preference can be set to one of three values:

- Boot Atari System V
- Boot TOS
- No Preference

If No Preference is the choice, the first bootable operating system is selected. The search sequence is partitions 0—3 in order from:

1. SCSI devices 0—7, in ascending numerical order
2. ACSI devices 0—7, in ascending numerical order
3. The diskette

The diskette does not have a partition table. At this time, a bootable Atari System V partition can only be found on a SCSI device; TOS can be found on any of the three.

If the preference is set to boot a specific operating system, then the first bootable partition which has a flag value matching the preference is booted. If that operating system doesn't exist on your hard disk, the boot program will hang, because it continues the search for a boot partition with that particular operating system.

If this should happen, insert the TOS **setboot** disk and press any key. The TOS system will come up and you can execute **SETBOOT.PRG** and set the boot preference to an operating system that exists on your hard disk. Setting it to No Preference always boots the system.

The system is shipped with No Preference. To change the boot preference from TOS to Atari System V, use the TOS program **SETBOOT.PRG** provided on a special diskette. To change the boot preference from Atari System V to TOS, use the **setboot** command in System V.

### TOS Boot Preference

Under TOS, the **SETBOOT.PRG** command brings up the **setboot** main screen, displaying a menu bar with the following titles:

- Desk, with the entry About Setboot... and the desk accessories
- File, containing the menu entry Quit
- Boot, with the entries
  - Set Boot Preference
  - Display Boot Preference
  - Initialize NVRAM
- Help

**Select** About Setboot... from the Desk menu to pop up the copyright box. **Select OK** to close the box.

To quit the program, **Select** Quit from the File menu. You will be prompted to confirm this action.

To set the current value of the boot preference in the NVRAM, **Select** Set Boot Preference from the Boot menu. A dialog box with the title Select Preferred Operating System pops up showing the following radio buttons:

- TOS
- Atari System V
- No Preference

The highlighted radio button represents the current boot preference in the NVRAM. No Preference selects the first bootable operating system. To change the boot preference, **Select** the radio button of the operating system to be booted and **Select OK**. A dialog box pops up asking you to reboot the system. This is necessary to boot the newly selected operating system.

To cancel the dialog box, **Select** the **Cancel** button.

Selecting Display Boot Preference from the Boot menu opens a dialog box showing the current boot preference in the NVRAM. Clicking on **OK** closes the dialog box.

If it should ever be necessary to initialize NVRAM because invalid data values have affected the checksum, **Select** Initialize NVRAM from the Boot menu. A dialog box pops up to confirm this action. Initialization clears NVRAM and sets the checksum.

The Help menu offers help boxes for the menu entries in the Boot menu as well as a general description of the purpose of the program.

Following is an example of how to set the boot preference to Atari System V using **SETBOOT.PRG**:

1. **Execute the SETBOOT.PRG command from the GEM desktop.**
2. **Select Set Boot Preferences from the Boot menu; the dialog box with radio buttons pops up.**
3. **Change the boot preference to UNIX by selecting the Atari System V radio button and click on the OK button.**

You will be asked to reboot the system; when done, Atari System V is booted.

### Atari System V Boot Preference

The Atari System V version of the boot preference utility is similar to the TOS version. The program is called **setboot**. Upon invocation from the WISh tool tray it opens a main window with the following components:

- A menu bar with the titles File and Options
- Three radio buttons:
  - TOS
  - Atari System V
  - No Preference
- An Apply button

The highlighted radio button displays the value of the boot preference in the NVRAM. To change the boot preference, select a different radio button and **Select** the **Apply** button. This changes the value in the NVRAM.

To re-read the NVRAM, **Select** Read NVRAM from the Options menu. This causes the program to read the NVRAM and set the radio buttons according to the boot preference value in the NVRAM.

If an error message appears while setting the boot preference, you should initialize the NVRAM by selecting Initialize NVRAM in the Options menu. This action must be confirmed.

To exit the program, **Select** Exit from the File menu.

When you set the boot preference to TOS, you must shut down the system to boot TOS. At the next reboot the system will then boot TOS.

### REFERENCES

AT&T *UNIX System V Release 4 Documentation*, Prentice-Hall, 1990:

*System Administrator's Reference Manual,*

**passwd(4)** **useradd(1M)**
**shadow(4)** **usermod(1M)**
**group(4)** **userdel(1M)**
**inittab(4)** **init(1M)**

*User's Reference Manual,*

**passwd(1)**

*System Services and Application Packaging Tools*, Chapter 8, "Packaging Application Software"

*System Administrators Guide,*
Chapter 5, "Administering a File System"
Chapter 12, "Security"
Chapter 13, "Service Access"
Chapter 17, "User and Group Management"

*Product Overview and Master Index*

Bach, M.J., *The Design of the UNIX Operating System*, Prentice-Hall, 1986

Non Standard Logics, *WISh2 User's Manual*, Paris, France, 1991

CHAPTER 4

# *Application Development*

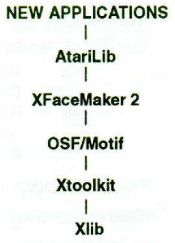
## OVERVIEW

Use this chapter for designing and creating programs in the Atari System V environment. Some of the information you'll find here includes

- Brief descriptions of software development tools and libraries.
- Guidelines for creating window-based applications, and internationalized applications in particular.
- A summary of how to package an application.
- An overview of how to write a device driver program and the steps used to include it in the system.
- A section on rewriting existing TOS-GEM programs for the Atari System V windowing environment.

## APPLICATION DEVELOPMENT LIBRARIES

User applications are developed on the Atari System V by writing programs in C programming language that use the functions and routines that are provided in several software libraries. Each library can be thought of as a layer.




Each layer is built upon the layers below it; that is, AtariLib makes direct calls to OSF/Motif routines, Xtoolkit routines, and Xlib routines, as well as XFaceMaker 2 routines.

Each layer, from the bottom up, contributes to the construction of a new application in the following ways:

- Xlib is library of basic windowing routines, such as mouse event, move window, size window.
- Xtoolkit is a library of windowing-associated widgets, including a scroll bar, a pop-up menu, and a toggle button.

- OSF/Motif is a library of windowing and widget routines that define the OSF/Motif style.
- XFacemaker 2 is an interactive application that creates the graphical interface of a new windowing application. It is also a library of interface routines.
- AtariLib is a library of graphical user interface routines, such as internationalization, alert boxes, and context-sensitive help.

Libraries in the Atari System V distribution include those shown in Table 4-1.

**Table 4-1 Atari System V Libraries**

| Name | Description | Source |
|---|---|---|
| Atari | Atari library | Atari |
| Fm | XFaceMaker2 | NSL |
| Xm | Motif | OSF |
| Xt | X Toolkit | MIT |
| X11 | X Library | MIT |
| socket | Socket library | V.4 |
| nsl | Socket hostname library | V.4 |
| malloc | Network services library | V.4 |
| gen | General purpose routiens | V.4 |
| m | Math library | V.4 |

To link these libraries into your own program, enter the following lines near the top of your *makefile*.

```
LIBS=-lAtari -lFm -lXm -lXt -lX11 -lsocket \
-lsockhost -lnsl -lmalloc -lgen -lm
```

## TOOLS

Included in the Atari System V set of software tools for applications development are the following GNU tools from the Free Software Foundation:

| | |
|---|---|
| gcc | C compiler |
| g++ | C++ compiler |
| gdb | C debugger |
| bison | compiler-generator |
| RCS | Revision Control System |

## PROGRAMMING NOTES

When programming, note the following:

- The object file format used in Atari System V is the Executable and Linking format (ELF), not the Common Object File Format (COFF) used in earlier releases of System V.
- When compiling X code, use **imake** to generate makefiles. The **imake** program automatically inserts the compiler flag **–DSYSV** to accommodate X11 header files in a System V environment.
- The debugger support format is called DWARF; it is supported by **gdb**.
- The gcc default compiler may issue calls to *gnulib*. Therefore, when porting an application to a platform that does not have it, include *gnulib* in the application package.

## INTERNATIONALIZED APPLICATION DEVELOPMENT

Internationalization involves generalizing programs or systems so they can handle a variety of languages, character sets, and national customs.

All text visible to the user must be internationalized. That is, it must be displayed in the language of the environment variable LANG. This includes titles, label strings, icon names, product names, and format strings used to compose other strings. Even a format string as simple as %s:%s must be internationalized, since punctuation differs from language to language.

File names and log file output should not be internationalized. If you are developing an OSF/Motif-based application, routines in the Atari library simplify the internationalization process.

Within the source code, do the following:

1. **Initialize the program environment to be the language of the user's locale.**

   See the **environ(5)** manual page for a description of the environment variables that define a locale.

   The **setlocale** routine initializes the program environment for a particular language for one or more of the following variables:

   - LC_TYPE affects the behavior of character handling and multibyte character functions
   - LC_COLLATE affects string collation and transformation
   - LC_MESSAGES affects message catalog functions
   - LC_MONETARY affects monetary formats
   - LC_NUMERIC affects numeric formatting functions
   - LC_TIME affects date and time string conversions

   In addition, LC_ALL affects all of the above.

   Example:

   ```
   setlocale(LC_ALL,"")
   ```

   A value of "" for locale specifies that the locale should be taken from environment variables. Refer to the **setlocale(3C)** manual page for details.

2. **Use the following Atari System V macros and routines that automatically handle internationalization requirements.**

   See *Appendix E* to read how these routines correspond to the standard specified in the *X/Open Portability Guide*, Issue 3.

   Character routines that handle characters and strings according to the locale setting are

   | | |
   |---|---|
   | **ctype(3C)** | character handling |
   | **conv(3C)** | character translation |
   | **mbchar(3C)** | multibyte character handling |
   | **mbstring(3C)** | multibyte string functions |
   | **strcoll(3C)** | string collation |
   | **string(3C)** | string operations |
   | **strxfrm(3C)** | string transformation |

The following local convention routines handle language-dependent representation of numbers, dates, and times. These functions are affected by the current locale setting.

| | |
|---|---|
| **nl_langinfo(3C)** | retrieve local convention information from environment table |
| **localeconv(3C)** | get numeric formatting information |
| **ctime(3C)** | convert date and time to string |
| **perror(3C)** | print system error messages |
| **printf(3C)** | print formatted output |
| **regexp(5)** | regular expression matching routines |
| **strftime(3C)** | convert date and time to string |
| **strtod(3C)** | convert string to number |
| **scanf(3C)** | convert formatted input |
| **vprintf(3C)** | print formatted output |

3. **Create and install a message catalog specific to the application.**

   **gencat(1)** produce a message catalog from a text file

   The output of the **gencat** command is placed in the directory

   */usr/lib/locale/$LANG/LC_ MESSAGES*

   replacing *$LANG* with the appropriate language. For example, if the language is French from France, then *LANG=french_france* and the catalog would be placed in

   */usr/lib/locale/french_france/LC_MESSAGES*

   In addition, the *gencat* file upon which all other translations are based should be placed in

   /usr/lib/local/C/LC_MESSAGES

   Translators may then **ungencat** the *message* file, translate it, **gencat** the translation, and move the file to the appropriate place for that language.

4. **Use the message catalog indicated by the locale setting.**

| | |
|---|---|
| **catopen(3C)** | open message catalog specified by NLSPATH |
| **catgets(3C)** | retrieve messages from message catalog |
| **catclose(3C)** | close the message catalog |

If possible, use the Atari Library routine **FmCatGetS** rather than these standard C library message catalog routines.

## ADDING A LANGUAGE TO THE SYSTEM ENVIRONMENT

Table 4-2 shows locales supported by Atari System V environment files.

Use the following steps to add another locale:

1. **Decide on a language name.**

   `lang_territory.codeset`

   Since it is almost always ISO 8859-1, you can drop **.codeset**. **Territory** may also be optional. For instance, if the new language were Romansch, it need not be **romansch_swiss.8859-1**, but simply **romansch**.

2. **Create and install both a character translation table and a numeric representation table.**

**Table 4-2 Environment File Locales**

| Locale | Description |
|---|---|
| english_usa* | American English |
| french_canada | Canadian French |
| danish | Danish |
| dutch | Dutch |
| english_uk | English |
| finnish | Finnish |
| french_france* | French |
| french_switzerland | Swiss French |
| german_germany* | German |
| italian_italy* | Italian |
| italian_switzerland | Swiss Italian |
| norwegian | Norwegian |
| portuguese | Portuguese |
| spanish | Spanish |
| swedish | Swedish |
| icelandic | Icelandic |
| C | Locale used when LANG not specified. Also used for translating to other locales, generally english. |

*Default locale for this language

a. Construct an input file using the file supplied in */usr/lib/locale/C/chrtbl_C* as the starting point. (See manual page **chrtbl(1M).**)

b. Generate tables using the **chrtbl** command.

c. Install the two output tables in their respective directories, */usr/lib/locale/$LANG/LC_NUMERIC* and /usr/lib/locale/$LANG/LC_TYPE.

**3. Create and install a character collation table.**

a. Construct an input file that describes the collating sequence for the new language. You may use */usr/lib/locale/C/colltbl_C* as a starting point, but it may be more useful to use */usr/lib/locale/english_usa/colltbl_C*. (See manual page **colltbl(1M)** for format and content of this file.)

b. Use the **colltbl** command to generate a collation table.

c. Install the collation table in */usr/lib/locale/$LANG/LC_COLLATE*.

**4. Create and install a table of monetary representations.**

a. Construct an input file to describe formatting conventions for monetary quantities for the new language (see the **montbl(1M)** manual page for specifications). You can use the file */usr/lib/locale/C/montbl_C* as a starting point, but it may be more useful to use */usr/lib/locale/english_usa/colltbl_C*.

b. Create a monetary database table using the **montbl** command.

c. Install the monetary database table in the directory */usr/lib/locale/$LANG/LC_MONETARY*.

5. **Create and install a file of time representation.**
   a. See the format specified in the **strftime(4)** manual page. Use the file */usr/lib/locale/C/time_C* as a starting point.
   b. Install in directory */usr/lib/locale/$LANG/LC_TIME*
6. **Create and install message catalogs for applications that will be used in the new language.**
   a. Use the **cp** command to copy the original language version of the message file from */usr/lib/local/C/LC_MESSAGES;* using a text editor, change the text portions to the new language.
   b. Generate a formatted message catalog from the text source file.
   c. Install the formatted message catalog in the directory */usr/lib/locale/$LANG/LC_MESSAGES*

## APPLICATION IMPLEMENTATION GUIDELINES

This section outlines recommendations for implementing XFM applications. It deals specifically with the interaction between the application and the */usr/lib/X11/Atari* library. All applications must conform to the Atari Style, as described in the *Atari Style Guide*.

### Atari Library

The Atari library routines make it easier to conform to Atari Style. These routines are listed at the end of this section. On-line manual pages are listed in Appendix A.

Atari library routines facilitate the use of XFaceMaker 2 functions that

- integrate internationalization into an XFaceMaker application,
- implement both context-sensitive and general help, and
- implement pop-up alert boxes.

The Atari library provides functions that implement input callback for most application needs. For example, X library routines call back with an arbitrary amount of input data; use an Atari library function that will pack this data up in the amounts you want before calling your application back.

#### *Window Construction*

If there is only one primary window, it must be an XmApplicationShell. You must use XmTopLevelShell for the second and any subsequent primary windows. All primary windows must be fully decorated, must have XmMainWindow as their immediate child, and must have a menu bar.

You must use XmTransientShell for secondary windows, which may not have menu bars. They must have a row of buttons at the bottom and a separator above them. Secondary windows that are modal should always appear near the widget that caused them to appear. This is done automatically for Help and Alert boxes. Others may use **PositionNear()**. Modeless secondary windows may appear wherever the application designer thinks appropriate, but should defer to the window manager, if possible.

Secondary windows, even modal ones, must have a title bar so that they can be moved by the user, and they must not restrict the cursor to the window interior, which would prevent the user from consulting other applications.

All windows should be unmapped in their *.fm* file. After all *.fm* files are loaded, all active values attached, and all library initialization routines called, the application should use **FmShowWidget** to make the first primary window appear. The pointer to the shell widget may be obtained through an active value.

### *Internationalization*

Within the source code, use the routine **FmCatGetS** to obtain an internationalized message. Mnemonics for the set number and message number should be declared with **#define** directives, instead of using plain integers. The message file (suffix *.msg)* should contain the mnemonic before the set or message declaration to aid in decoding.

For example, if you define

```
#define ERRORMSGS_SET 123
#define CANNOT_OPEN 32
```

the *.msg* file should contain something like

*The dollar sign ($) is a comment sign for messages*

```
$ ERRORMSGS_SET
set 123
$ CANNOT_OPEN
32 Cannot open that file.
```

- **FmCatGetS** uses **catgets(3I)** to obtain messages. These messages are read into a static space, so the string returned by FmCatGetS should be copied if it is to be used after the next call to FmCatGetS (or **catgets**).

### *Alert Popup Dialogs*

Alert dialogs are used for all application errors. The only exception is when the **AlertInit()** function fails. In this case, the application writes a message to the log file and exits with a nonzero status. The alert displayed to the user will be of a nontechnical nature, with all technical and system call error information directed to the log file.

Fatal alert messages should be presented with the **InternalError()** function, which exits after displaying the message.

Nonfatal alert messages must be presented whenever a system call or library routine fails in a way that adversely affects the application, especially if the user provided the information, such as a file name, for the routine that failed. Some typical routines that may incur such errors are **fopen()** and **unlink()**.

### *Help Popup Dialogs*

All applications must provide Help On Context and Help On Version. If an application uses a mnemonic not found in the OSF/Motif Style Guide, it must also provide Help On Keys. If it binds any custom mouse actions, it must also provide Help On Mouse. Help On Keys and Help On Mouse may be provided even though not required by the above conditions. Help On Help, Help On Window, Index help, and Tutorial help may be available, but are not required.

### *Running Subprocesses*

The **system()** library routine should not be used to run processes, since it waits for the child to exit before returning and causes problems for the X11 Window

System's asynchronous event processing. Another drawback of the **system()** library routine is that it uses a shell process to execute the command, which should not be necessary. For this reason, the **popen()** routine should also be avoided. The **RunProcess()** function provides a way to run subprocesses with input/output redirection, including pipes.

Note that the asynchronous nature of X11 also requires that the application not block the waiting period for a subprocess to terminate. Thus, the SIGCLD signal must be caught by the application, and the wait for the terminated process must be done in the signal handler. To minimize the intrusion to X processing, the signal handler should limit its activity to obtaining the termination status of the process by means of the **wait(2)** system call and then registering a work process to do the actual signal handling. See the documentation on the X Toolkit routine, **XAppAddWorkProc**, for details on adding a work process.

#### *Input/Output Handling*

Any read or write operations that can potentially block may also interfere with the operation of the X11 Window System. When an application is programmed with all code in-line, effectively blocking whatever is waiting for file input or keyboard input—the window cannot be resized, iconified, moved, or otherwise manipulated during this wait time. In contrast, the window of an application programmed with input callbacks can be resized, iconified, moved, etc., regardless of where the data is.

For this reason, the X Toolkit routine **XtAppAddInput** allows input/output routines to be called only when the request may be done without blocking. Although the routine is called **XtAppAddInput**, it may in fact be used for detecting read, write, and exception conditions. Pipes, pseudo-TTYs, and network files are all especially susceptible to being blocked, on both reading and writing. Even ordinary files may be network files because of NFS and other transparent file systems, so it's a good idea to always use **XtAppAddInput** to handle input/output processing.

This input-callback programming technique is described in

- Nye, A., and O'Reilly, T.,.*X Toolkit Intrinsics Programming Manual, Volume Four*, Section 8, "Input Techniques" Subsection 8.3, "File,Pipe, and Socket Input"
- Nye, A. and O'Reilly, T., *X Toolkit Intrinsics Reference Manual, Volume Five,* "Xt Functions and Macros Subsection Event Handling, XtAddInput"
- Young, D. A., *The X Window System—Programming and Applications with Xt; OSF/Motif Edition*, Section 5.8.1, "Using Input Callbacks"

#### *Log Files*

Applications should log their progress using the logging routines provided in the Atari library. Every major state change, such as opening or closing a window, should be written to the log. More detailed information should use the **LogDebug** macro so it will not be compiled into the code for production.

Any abnormal condition, especially one resulting in a fatal alert message, should be noted in the log file.

The application must call **LogClose** upon normal termination in order to remove the log file.

### Atari Library Routines

***Help Subsystem:***

| | |
|---|---|
| **HelpInit** | Initialize help subsystem |
| **PopupHelpAndWait** | Present help box and wait for response |
| **HelpOnContext** | Provides help on context for application |

***Alert Subsystem:***

| | |
|---|---|
| **AlertInit** | Initialize alert box subsystem |
| **InternalError** | Present an internal error alert box, then exit |
| **AlertSetButtons** | Set buttons in an alert box |
| **AlertHelp** | Present help box for an alert box |
| **PopupAlertAndWait** | Present alert box and wait for a response |

Convenience routines present a standard dialog box and wait for a response:

**PopupErrorAndWait**
**PopupInformationAndWait**
**PopupMessageAndWait**
**PopupQuestionAndWait**
**PopupWarningAndWait**
**PopupWorkingAndWait**

***Interactive Command Execution Subsystem:***

| | |
|---|---|
| **ExecuteList** | Execute a command and a list of arguments |
| **ExecuteListV** | Execute a command and a vector of arguments |
| **ExecuteString** | Use the shell to execute a command in a string |

***Noninteractive Command Execution Subsystem:***

| | |
|---|---|
| **RunProcess** | Run a process with I/O redirection |
| **RunProcessV** | Run a process with command arguments as a vector |
| **AddChildHandler** | Add a child handler to a running process |
| **GetCommandStatus** | Run a process and call a function with exit status |
| **GetCommandStatusV** | Same, with command arguments as a vector |

***High-level Asynchronous Input/Output Routines:***

| | |
|---|---|
| **ReadFileData** | Read raw data from a file |
| **ReadFileLines** | Read lines from a file |
| **ReadFileStrings** | Read lines from a file and store in array of strings |
| **ReadPipeData** | Read raw data from a pipe |
| **ReadPipeDataV** | Like ReadPipeData, with command argument vector |
| **ReadPipeLines** | Read lines from a pipe |
| **ReadPipeLinesV** | Like ReadPipeLines, with command argument vector |
| **ReadPipeStrings** | Read lines from pipe and store in an array of strings |
| **ReadPipeStringsV** | Like ReadPipeStrings, with command argument vector |

***Low-level Asynchronous Input/Output Routines:***

| | |
|---|---|
| **AddIoProc** | Register an asynchronous I/O procedure. |
| **RemoveIoProc** | Remove an asynchronous I/O procedure |
| **OpenReadPipe** | Open a pipe for reading |
| **OpenReadPipeV** | Open a read pipe, with command arguments as a vector |

| | |
|---|---|
| **OpenWritePipe** | Open a pipe for writing |
| **OpenWritePipeV** | Open a write pipe, with command arguments as vector |
| **OpenFilter** | Run a filter process with pipes both in and out |
| **OpenFilterV** | Run a filter, with command arguments as a vector |
| **GetPipePid** | Get process ID of a pipe from its file descriptor |
| **ClosePipe** | Close pipe file descriptor |

***Application Logging Routines:***

| | |
|---|---|
| **LogOpen** | Open a log file |
| **LogWrite** | Write to a log file |
| **LogDebug** | Put debug message in log file |
| **LogAssert** | Check an assertion and report failure in log file |
| **LogSystemError** | Report a system error to the log file |
| **LogClose** | Close and remove a log file |
| **LogReopen** | Change a log file |

***Miscellaneous routines:***

| | |
|---|---|
| **AddPath** | Add an element to a path-like environment variable |
| **AddPathV** | Add a vector of elements to a path variable |
| **AdjustWmPadding** | Adjust padding values depending on resource values |
| **FindWmPadding** | Find out size of window manager decorations |
| **FmCatGetSAlloc** | Call FmCatGetS and copy result in a dynamic buffer |
| **FmCatGetSRealloc** | Call FmCatGetS and re-use given buffer |
| **GetSimpleCharSet** | Get default character set for OSF/Motif strings |
| **PositionBeneath** | Position a widget beneath another widget |
| **PositionCenter** | Position a widget centered over another widget |
| **PositionNear** | Position a widget near another widget |
| **PositionOver** | Position a widget over another widget |
| **PositionRootCenter** | Position a widget in the center of the root window |

## APPLICATION PACKAGING

You should bundle your application into an installable product so that it can be installed automatically using the system administration tool Product Installation, or the command **pkgadd**. There are no choices with regard to the directory in which the product will be installed, or to the parts of the package to install, unless incorporated into the installation portions of the package.

This section is an overview of how an application should be bundled into an installable product. For more detailed information, refer to the instructions, sample files, and scripts in the AT&T, *Unix System V Release 4 Programmer's Guide: System Services and Application Packaging Tools*, Chapter 8 "Packaging Application Software," Appendix B "Manual Pages," and Appendix C "Package Installation Case Studies."

1. **Create a file called *pkginfo*.**

   This ASCII file describes the application package name, release, and version numbers.

2. **Create a file called *prototype*.**

   This ASCII file has one entry per file that is a part of the application package, including the *pkginfo* and *request* files.

3. **Create an (optional) installation script.**

   This file may be a Bourne shell (sh) script, or may be an executable program. It can be a request, a class action, or a procedure script. Customize this script for the application package. Typical things an installation script can do are

   - Set up for selective installation (ask which parts of the package should be installed and where they should be placed).
   - Install a device driver (ask how many device nodes to create, run a postinstall script, and reboot the system upon installation).
   - Define extra disk space requirements required for this package (create a file called *space*).
   - Display a copyright message (create a file called *copyright*)
   - Define any software dependencies associated with this package (create a file called *depend*).
   - Modify a system file during installation.

   At the end of the request script, relevant parameters are made available to the installation environment for **pkgadd**.

4. **Run the pkgmk command, which will gather all components of a package, copy then onto the installation medium, and place them into a structure that pkgadd will recognize.**

## DEVICE DRIVERS

Atari System V treats devices as special files that data is either read from or written to. These files are called device drivers. Files that provide interfaces to other system resources are called modules or "software drivers" (i.e., there is no physically removeable hardware device per se). Some examples of these device drivers and software modules are

- The block device driver that controls the hardware disk unit.
- The streams driver or module that controls the hardware terminal or the software terminal line-discipline module.
- The line discipline module.

Certain drivers are required to run the system—the keyboard driver, for example. Those shown in Table 4-3 and a few additional drivers are present in Atari System V.

Additonal drivers can be added to your system; write them as needed, or find someone who already has one that suits your needs.

A device driver must concern itself with three specific interfaces:

- the hardware device,
- the kernel, and
- the boot

The interface with the kernel is by means of data structures. See the *AT&T Device Driver Interface/Driver-Kernel Interface(DDI/DKI) Reference Manual*

**Table 4-3 Atari System V Device Drivers**

| Driver | Function |
|---|---|
| IKDB | Intelligent keyboard |
| SCSI | Generic SCSI manager |
| HD | SCSI hard disk driver(interfaces with SCSI driver) |
| TP | SCSI tape (interfaces with SCSI driver) archive viper tape streamer |
| FFD | 720KB floppy disk |
| CEN | Centronics parallel port |
| SCCIO | Two serial ports on Zilog 6530 SCC chip |
| USART | Two serial ports on MFP chips |
| LA | VME Ethernet board driver |
| VIDEO | Video subsystem |
| PSG | Programmable sound generator |
| CLOCK | Real-time clock |
| MFP | Multifunction peripheral/interrupt controller |

for the System V.4 protocol approved by AT&T. There may be some name changes for writing device drivers in a Motorola 68000 environment, but the functionality should be the same.

If you are writing a device driver for a SCSI device, refer to the on-line "Guide to Writing Device Drivers for the Generic SCSI" located in the file */usr/local/src/samples/scsidriver/scsigen.doc*.

The source for a sample SCSI printer driver is available on-line in the file */usr/local/src/samples/scsidriver/pr.c*. Compile it as follows:

```
gcc -Xt -D_KERNEL -O -c pr.c
```

### Adding a Device Driver

1. **Create an object file (in ELF format) with gcc, which will be added to the kernel. If a device uses several *a.out* files, link them together.**
2. **Create a master(4) file in */etc/master.d* directory.**
3. **If it is a software-only driver, add an entry to the */stand/system* file.**
4. **If the driver is for a hardware device, add entry to */stand/edt_data* file.**

   You cannot use **edittbl(1M)**, because the format of *edt_data* has been changed in the Atari System V release.

5. **Install the driver with drvinstall(1M). This creates a boot module from the master file and the driver object file and places it in /boot.**
6. **If the driver is for a hardware device, install a boot probe program in */stand*. When the system boots, it looks for these.**

   In */usr/src/uts/boot/probe* there is a probe program called **naiveprobe.c** that checks for a bus error when performing a byte read at the device base address. For straightforward cases this may be compiled unmodified for each device, with the define DEVNAME variable set to the appropriate device name by means of a **–DDEVNAME** flag in the makefile **probe.mk**. For more complex cases, dedicated probe programs may be written for each device. The *xedt* library contains the necessary structures.

7. **If the driver is a hardware device, make special files with *mknod(1M)*.**
8. **Reconfigure the kernel.** Refer to Chapter 3, "System Reconfiguration."
9. **Reboot and use the device.**

## PORTING TOS/GEM APPLICATIONS

There is no simple rule for converting GEM programs to OSF/Motif. Both are graphical user interfaces with a windowing system and have several features in common.

### Porting By Means of XFaceMaker 2

Atari System V uses the OSF/Motif widget set and the XFaceMaker 2 interface builder, which allow the programmer to concentrate on the functionality of the application. XFaceMaker also hides the complexity of the X Window System.

However, the XFaceMaker 2 library is more advanced, including such features as a built-in C-like language, callbacks, active values, and the availabiltity of powerful widgets.

As a GEM programmer, you are familiar with writing graphical user interface software. The software you want to port usually consists of two parts: the application and the user interface.

#### *Application*

If the application was written in C, it is easy to transfer it to Atari System V and recompile it. Calls to the TOS operating system must be converted to the equivalant Atari System V operating system calls. The most compatible way is to use the C standard library functions—**fopen**, **fwrite**, etc. (Refer to the section "Input/Output Handling" in this chapter.

#### *User Interface*

The user interface must be redesigned. OSF/Motif has a different look and feel than GEM and has new graphical objects you may want to use. Also, the application now has to run in a multi-tasking environment.

To start, redesign the user interface using XFaceMaker 2. Create a main window that contains the menu bar. The contents of the window should be the contents of your main window (form) in the GEM application.

### Forms and Windows

Under the X Window system, windows are used, rather than forms; these windows may overlap. The windows controlled by the window manager have a title bar and are similar to GEM windows.

Thus, a form dialog box in GEM must be implemented as a window under Atari System V. XFaceMaker 2 helps the developer create windows that behave in the same way as dialog forms.

XFaceMaker 2 allows you to create a window and place interactive widgets and text edit fields inside it. You can specify callback functions that are the names of functions in the program you are writing. These functions will be called by the XFaceMaker 2 library when this object is selected.

After creating the window, save it and write the program. Usually the XFaceMaker 2 library opens the window automatically and handles all events. If an object such as a button is selected, the XFaceMaker 2 library calls the callback fucntion specified for the object and acts on the user's input. When using the XFaceMaker 2 library, the application developer has less to do than when using GEM.

To open a window without using XFaceMaker 2, use the Xt Toolkit. To draw circles, rectangles, etc., inside a window use the Xlib. It's similar to drawing with the VDI inside a GEM window. If you use Xlib drawings, redrawing the window is more complicated. It's possible to draw with Xlib functions inside a window created by XFaceMaker 2. See Appendix D for a table of GEM/Xlib equivalents.

### Main Loop

The main loop in a GEM program is usually the **event_multi()** loop which waits for events to occur. The application then determines the type of event that occurred and calls the appropriate function.

With XFaceMaker 2, this is not necessary. Once the main event loop **(FmLoop)** has been called, your functions that react on events are called automatically.

During the design of the user interface of the application using XFaceMaker 2, you specify callback functions for interactive objects. The XFaceMaker 2 libary will call your function when an action takes place on that object.

For instance, you might create a window with one button. When the button is pressed, the window closes and the application exits. Design the window with XFaceMaker 2 and place the button inside it. Specify the name of the callback function for the button via the XFaceMaker 2 resource window using the activate callback resource. For example, use **button_pressed()**.

After saving the user interface you write your program. The core program in general has to contain only two functions:

- The **main()** function, calling **FmInitialize()**, **FmAttachFunction()** to connect the C-function with the user interface and the **FmLoop()**.
- Your callback function for the button in this example, **button_pressed()**. This function calls **exit(0)**, which causes the window to close and the application to exit.

The **FmLoop()** is like calling the **event_multi()** function in GEM and parsing the occured events, except that the parsing is done automatically by X and the XFaceMaker 2 library. Because the library knows the names and addresses of your functions, it is able to call them.

#### *Porting Example*

There is a small GEM address application available which was ported to OSF/Motif. There you are able to see how the GEM look and feel was converted to Motif/X Windows following the Atari Style Guide. This GEM application is on-line in */usr/local/src/samples/gem2motif*. The GEM example is in *subdirgem* and the Motif example in *subdirmotif*

## REFERENCES

Atari Computer Corp., *Atari Style Guide*, Atari, 1991

AT&T, *UNIX System V Release 4* Documentation, Prentice Hall, 1990:

*Device Driver Interface/Driver Kernel Interface(DDI/DKI) Reference Manual*
*Programmer's Guide: STREAMS*
*Programmer's Guide: Networking Interfaces*
*Programmer's Guide: BSD/XENIX Compatibility Guide*
*Programmer's Reference Manual*
*System Administrator's Guide*
*System Administrator's Reference Manual*

**environ(5)**
**langinfo(5)**
**nl_types(5)**
**chrtbl(1M)**
**colltbl(1M)**
**montbl(1M)**

*System Services and Application Packaging Tools*,
Chapter 8, "Packaging Application Software"
Appendix B,
Appendix C,

Non Standard Logics, *XFaceMaker 2 Programmer's Guide*, Paris, France, 1991

Non Standard Logics, *XFaceMaker 2 User's Guide*, Paris, France, 1991

Nye, A. and O'Reilly, T., O'Reilly & Associates, 1990:

*Volume One: Xlib Programming Manual*
*Volume Two: Xlib Reference Manual*
*Volume Four: X Toolkit Intrinsics Programming Manual*
*Volume Five: X Toolkit Intrinsics Reference Manual*

Open Software Foundation, *OSF/Motif Programmer's Guide*, Prentice-Hall, 1990

Open Software Foundation, *OSF/Motif Style Guide, Revision 1.1*, Prentice-Hall Inc., 1991

Open Software Foundation, *OSF/Motif Programmer's Reference*, Prentice-Hall, 1991

Young, Douglas, *The X Window System—Programming and Applications with Xt—OSF/Motif Edition*, Prentice-Hall Inc., 1990

X/Open Company, Ltd., *X/Open Portability Guide, XSI Supplementary Definitions*, Prentice Hall, Inc., 1989

# APPENDIX A

## *Atari-Specific Manual Pages*

Following is a list of changes, additions, and omissions that characterize the difference between the Atari System V on-line manual pages and the AT&T 3B2 manual page documentation. Some differences have to do with the nature of the Atari port, others are Atari enhancements.

Manual pages for commands and files that have been changed or added are available on-line by using the **man(1)** command. For example, for information about gcc, type **`man gcc.`**

Use the permuted index in the *AT&T Product Overview and Master Index* to locate specific manual pages in the manuals

### Changes

These manual pages were changed to accommodate the Atari System V port.

cunix(1M)
format(1M)
prtvtoc(1M)
newboot(1M)
edt_data(4)
master(4)

### Additions

The following manual pages contain Atari System V enhancements.

bison(1)
ci(1)
co(1)
codesio(1)
findsrc(1)
gcc(1)
g++(1)
gdb(1)
glscnv(1)
glsstty(1)
gnudiff(1)
hex(1)
ident(1)
indent(1)
kermit(1)
less(1)
lesskey(1)
makekit(1)
merge(1)
mush(1)
patch(1)
psc(1)
rcs(1)
rcsdiff(1)
rcsintro(1)
rcsmerge(1)
rlog(1)
sc(1)
shar(1)
unshar(1)

bootaddhdr(1M)
bpatch(1M)
consmode(1M)
e(1M)
nvram(1M)imagpartini
t(1M)
setlang(1M)
setnetwork(1M)
setvtoc(1M)

sysm68k(2)

addchildhandler(3A)
addioproc(3A)
addpath(3A)
alertinit(3A)
atarilib(3A)
closepipe(3A)
executelist(3A)
findwmpadding(3A)
fmcatgets(3)
fmcatopen(3A)
getcommandstatus(3A)
getpipepid(3A)
getsimplecharset(3A)
helpinit(3A)
logopen(3A)
logwrite(3A)
openfilter(3A)
openreadpipe(3A)
openwritepipe(3A)
position(3A)
readfiledata(3A)
readfilelines(3A)
readfilestrings(3A)
readpipedata(3A)
readpipelines(3A)
readpipestrings(3A)
runprocess(3A)
writefiledata(3A)
writefilelines(3A)
writepipedata(3A)
writepipelines(3A)

getedt(3C)
getxedt(3C)

rcsfile(5)

gls(7)

boot(8)
edtp(8)

## Omissions

Not a part of Atari System V.

| | |
|---|---|
| crash | ledre |
| disks | login |
| editsa | pump |
| edittbl | sadp |
| cmpress | wtinit |
| fltboot | xts |
| fmtflop | xtt |
| fmthard | zdump |
| ismpx | XWIN X11/NeWS server |
| jterm | X11/NeWS, tNt and Shapes |
| jwin | API toolkits |
| layers | EMD |
| ldsysdump | 3B2 Floppy Disk Driver ... |
| | AT&T Windowing Utilities |

# APPENDIX B

# *Boot Text*

## BOOT TEXT OUTPUT TO THE CONSOLE

The screen starts blank. After a 90-second time out (or when any key is depressed), the following message appears:

```
Atari System V.4 Fri Jun 7 14:02:04 EDT 1991
```

This message appears only on the console, not on a terminal output of the USART port. The date and message may change, depending on the specific generation of the System V boot.

The next section of information appears on both console and the USART port.

*This information differs, depending on the specific configurations of SCSI devices connected to the machine. Every SCSI device should appear on this list.*

```
SCSI:

TARGET LUN

0  0     MAXTOR   LXT-213S    4.20 Direct access
```

*Now there is a time-out of about 20 seconds. As administrator you may press <Return> to skip the pause.*

```
UNIX System V Release 4.0 Boot Loader

Boot: hd(0,0)unix

Building Equipped Device Table
CEN:         controller 0 at 0xFFFF8800
CLOCK:       controller 0 at 0xFFFF8960
FFD:         controller 0 at 0xFFFF8606
HD:          controller 0 at 0x0
IKBD:        controller 0 at 0xFFFFFC00
LA:          no devices found
MFP:         controller 0 at 0xFFFFFA01
ND:          no devices found
PR:          no devices found
PSG:         controller 0 at 0xFFFF8800
SCCIO:       controller 0 at 0xFFFF8C80
SCSI:        controller 0 at 0xFFFF8780
TP:          no devices found
VIDEO:       controller 0 at 0x0
USART:       controller 0 at 0xFFFFFA01
```

*The* `no devices found` *message means that those specific devices are not physically connected to the system.*

```
Loading unix: 820 + 0[193192] + 722896 + 93432 +
864

UniSoft UNIX (R) System V Release 4.0
            Version UE10 for the mc68030

Total real memory = 8388608
Available memory  = 6664192
```

*These numbers vary, depending on the amount of memory installed on the system and the configuration of the kernel.*

*The screen clears and is replaced by the gray background of root X Window; then the login screen is displayed.*

*This line varies, depending on the node name of the system.*

*Memory and device configuration will vary.*

```
**********************************************************
Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T-All Rights
Reserved
THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF AT&T INC.
The copyright notice above does not evidence any actual or
intended publication of such source code.
Portions provided by UniSoft Corp. under a development agree-
ment with AT&T and Motorola.
**********************************************************

*Node: noname

MOTOROLA M68K SYSTEM CONFIGURATION

Memory size: 8 Megabytes

System Peripherals:

   Device Name

   CEN
   CLOCK
   FFD
   HD

     MAXTOR    LXT-213S   4.20   0    203 Mbytes

   IKBD
   MFP
   PSG
   SCCIO
   SCSI
   USART
   VIDEO

SYSTEM V STREAMS TCP Release 1.0
(c)     1983,1984,1985,1986,1987,1988,1989 AT&T.
(c)     1986,1987,1988,1989 Sun Microsystems.
(c)     1987,1988,1989 Lachman Associates, Inc. (LAI).
                All Rights Reserved.

The system is ready.
```

# APPENDIX C

## *References*


Atari Computer Corporation, *ATARI TT Computer Owners Manual*, Atari Corporation, 1990

Atari Computer Corporation, Atari Style, Atari Corporation, 1991

AT&T, *UNIX System V Release 4*, Prentice-Hall, Inc., 1990—the following:

*Device Driver Interface/Driver Kernel Interface*
*Network User's and Administrator's Guide*
*Product Overview and Masters Index*
*Programmer's Guide: BSD/XENIX Compatibility Guide*
*Programmer's Guide: Networking Interfaces*
*Programmer's Guide: STREAMS*
*Programmer's Guide: System Services and Application Packaging Tools*
*Programmer's Reference Manual*
*System Administrator's Guide*
*System Administrator's Reference Manual*
*System Services and Application Packaging Tools*
*User's Guide*
*User's Reference Manual*

Bach, M.J., *The Design of the UNIX Operating System*, Prentice-Hall, 1986

Bourne, S., *The Unix System V Environment*,

McGilton, and Morgan, *Introducing Unix System V, McGraw Hill, 1983*

Non Standard Logics, *WISh2 User's Manual*, Paris, France, 1991

Non Standard Logics, *WX2 User's Guide*, Paris, France, 1991

Non Standard Logics, *XFaceMaker 2 User's Guide for Use With Version 1.0*

Nye, A. and O'Reilly, T., published by O'Reilly & Associates, 1990—the following:

*Volume One, Xlib Programming Manual for Version 11*
*Volume Two, Xlib Reference Manual for Version 11*
*Volume Four, : X Toolkit Intrinsics Programming Manual*
*Volume Five, X Toolkit Intrinsics Reference Manual for X Version 11*

Open Software Foundation, Prentice Hall Inc., 1990—the following:

*OSF/Motif Programmer's Guide, Revision 1.0*
*OSF/Motif Programmer's Reference, Revision 1.1*
*OSF/Motif Style Guide, Revision 1..1*
*OSF/Motif User's Guide, Revision 1.0*

Sobell, M., *A Practical Guide to Unix System V*—Second Edition, Release 4, Benjamin Cumming Publishing Company, 1991

Young, D. A., *The X Window System—Programming and Applications with Xt; OSF/Motif Edition*, Prentice-Hall, Inc., 1990

X/Open Company, Ltd, *X/Open Portability Guide, XSI Supplementary Definitions*, Prentice–Hall, Inc.

# APPENDIX D

## *GEM / Xlib Equivalents*

The following table of GEM/Xlib equivalents will help you find the Xlib function equivalent to a GEM function. The parameter **display** is always the returned display pointer from XOpenDisplay.

X provides several routines for drawing. The graphic context (GC) resource specifies variables such as color, line width, and fill pattern. A GC is specified as an argument to the drawing routine and modifies its appearance. For more details about the GC and how to write programs using Xlib, see the *Xlib Programming Manual*.

The table contains the most frequently used GEM functions, their Xlib equivalents, and the page number for the function description in the *Xlib Reference Manual*. Instead of attribute functions for each graphic primitive such as GEM has, Xlib uses only one function to set an attribute for all graphic primitives. In general, setting a GC attribute affects the drawing of every graphic primitive call.

**VDI—Xlib Equivalents**

| Frequently Used GEM Functions | Xlib Equivalents | Page No. |
|---|---|---|
| v_arc(handle, x, y, radius, begin_angle, end_angle | XDrawArc(display, drawable, gc, x, y, w, h, angle1, angle2)<br>XFillArc(display, drawable, gc, x, y, w, h, angle1, angle2<br>XSetArcMode(display, gc, arc_mode) | 134<br>172<br>397 |
| v_bar(handle, xyarray) | XFillRectangle(display, drawable, gc, x, y, w, h) | 178 |
| v_circle(handle, x, y, radius) | XDrawArc(display, drawable, gc, x, y, w, h, angle1, angle2)<br>XFillArc(display, drawable, gc, x, y, w, h, angle1, angle2)<br>XSetArcMode(display, gc, arc_mode) | 134<br>172<br>397 |
| v_clsvwk | XCloseDisplay(display) | 81 |
| v_ellarc | XDrawArc(display, drawable, gc, x, y, w, h, angle1, angle2)<br>XFillArc(display, drawable, gc, x, y, w, h, angle1, angle2)<br>XSetArcMode(display, gc, arc_mode) | 134<br>172<br>397 |
| v_ellipse | XDrawArc(display, drawable, gc, x, y, w, h, angle1, angle2)<br>XFillArc(display, drawable, gc, x, y, w, h, angle1, angle2)<br>XSetArcMode(display, gc, arc_mode) | 134<br>172<br>397 |
| v_enter_cur<br><br>v_exit_cur | no real equivalent for this under X; for terminal emulation open an xterm window<br>no real equivalent for this under X; for terminal emulation open an xterm window | |
| v_get_pixel | XGetPixel(ximage, x, y) gets a pixel from an IMAGE, not the whole screen area | 223 |

**VDI—Xlib Equivalents, continued**

| Frequently Used GEM Functions | Xlib Equivalents | Page No. |
|---|---|---|
| v_gtext(handle, x, y, string) | XDrawString(display, drawable, gc, x, y, string, length_of_string) | 157 |
| | XDrawString16(display, drawable, gc, x, y, string, length_of_string) | 159 |
| | XDrawImageString(display, drawable, gc, x, y, string, length_of_string) | 141 |
| | XDrawImageString16(display, drawable, gc, x, y, string, length_of_string) | 143 |

The most important functions that specify a drawing in a GC are

XSetArcMode()
XSetClipMask()
XSetClipOrigin()
XSetClipRectangles()
XSetRegion()
XSetDashes()
XSetLineAttributes()
XSetFillRule()
XSetFillStyle()
XSetTile()
XSetTSOrigin()
XSetForeground()
XSetGraphicExposures()
XSetSubwindowMode()
XSetBackground()
XSetFunction()
XSetPlaneMask()
XSetState()

Graphic primitives (drawing functions) are

XDraw()
XDrawArc()
XDrawArcs()
XDrwaFilled()
XDrawLine()
XDrawLines()
XDrawPoint()
XDrawPoints()
XDrawRectangle()
XDrawRectangles()
XDrawSegments()
XFillArc()
XFillArcs()
XFillPolygon()
XFillRectangle()
XFillRectangles()
XClearArea()
XClearWindow()

See the *Xlib Reference Manual* for details about the functions.

# APPENDIX E

## *Atari Enhancements to Internationalization Standards*

Atari System V fully supports the X/Open Portability Guide, Issue 3 (xpg3) Standard for internationalization of programming and character sets. Included here is a table of Atari System V functions and the corresponding xpg3 function. Most functions are part of the standard C library, the others are Atari enhancements.

| ATARI SYSTEM V | xpg3 FUNCTION | DESCRIPTION |
|---|---|---|
| catgets (3C) | catgets | Read message catalog |
| catopen(3C) | catopen, catclose | Open/close message catalog |
| ecvt(3C) | gcvt | Convert floating point number to string |
| ctime(3C) | | Convert date and time to string |
| ctype(3C) | isalnum<br>isalpha<br>iscntrl<br>isgraph<br>islower<br>isprint<br>ispunct<br>isspace<br>isupper | Character handling |
| conv(3C) | tolower, toupper | Character translation |
| codesio(1) [2] | | Character code set compiler |
| getdate(3C) | | Convert date and time format |
| gls(7) [2] | | Character mapping STREAMS module |
| glscnv(1) [2] | | Code set mapping test program |
| glstty(1) [2] | | Enables character mapping module |
| localeconv(3C) [1] | | Get numeric formatting formats |
| mbchar(3C) [1] | | Multibyte character handling |
| mbstring(3C) [1] | | Multibyte string handling |
| nl_langinfo(3C) | nl_langinfo | Get local formats |
| perror(3C) | perror | Print system error messages |
| printf(3C) | printf<br>fprintf,<br>sprintf | Print formatted output |
| regexp(5) | regexp | Regular expression matching |
| setlocale(3C) | setlocale | Get environment variables |
| scanf(3C) | scanf<br>fscanf<br>sscanf | Scan formatted input |

[1] Conforms to ANSI x3.159 programming language C standard
[2] Atari enhancement

| ATARI SYSTEM V | xpg3 FUNCTION | DESCRIPTION |
|---|---|---|
| strcoll(3C) | strcoll | String collation |
| strerror(3C) | strerror | Get error message string |
| strftime(3C) | strftime | Convert date and time to string |
| string(3C) | | String operations |
| strtod(3C) | strtod<br>atof | Convert string to number |
| strxfrm(3C) | strxfrm | string transformations |
| vprintf(3C) | vprintf<br>vfprintf<br>vsprintf | Print formatted output |

The standard headers are used to provide definitions and declarations for use with nationalized language functions:

| | |
|---|---|
| ctype.h | See **ctype(3C)** manual page |
| langinfo.h | Language constants; see **langinfo(5)** manual page |
| limits.h | Operating system limits; see **limits(4)** manual page |
| locale.h | Catagory names; see **setlocale(3C)** manual page |
| nl_types.h | Language data types; see **nl_types(5)** manual page |

# Index

## *T*



## *U*



## *V*



## *W*



## *X*