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

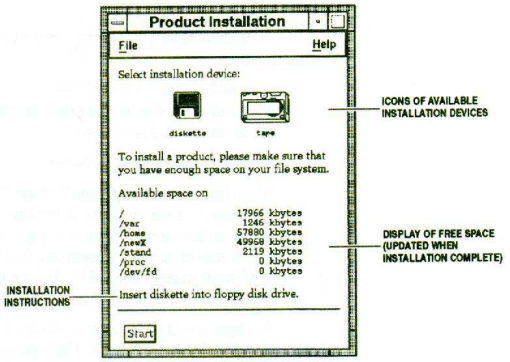
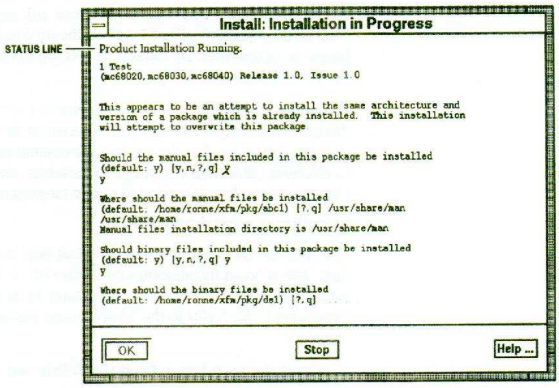


Figure 3-2
Progress Window



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

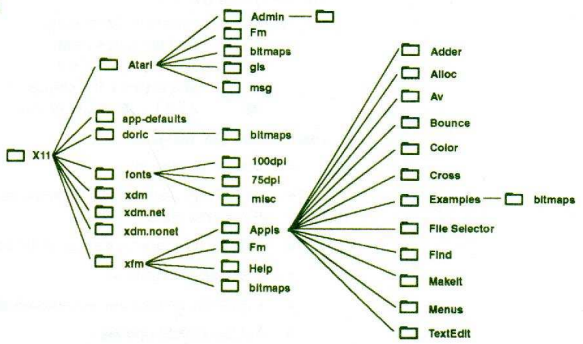
The main file system is called the root file system, and is usually depicted as a slash (/). Other file systems can only be accessed by users if they are mounted. That is, if the other file systems are logically connected to the root file system or a file system already mounted on the root file system.

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

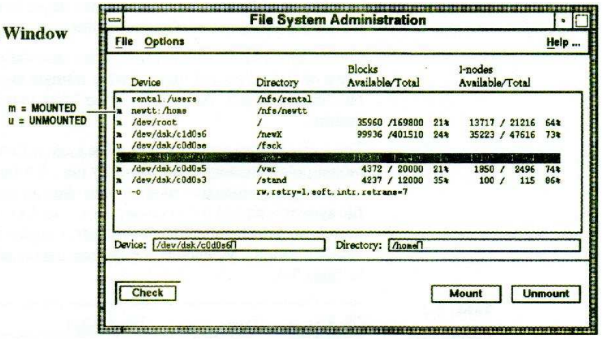
Use the Atari tool **filesys** to

- 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



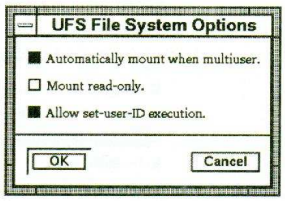
2. **Select a file system, or type the name of a file system in either the Device text box or the Directory text box, as appropriate.**

   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.

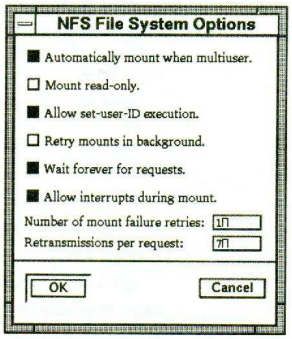
Figure 3-5
UFS File System Options



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).

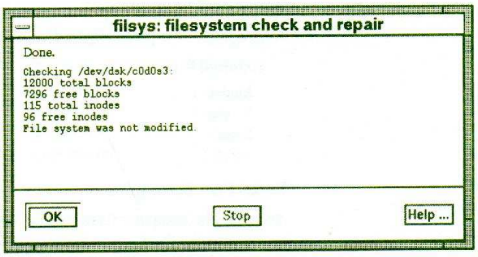
Type the **fsck** command, using the **-m** flag to determine whether a file system needs checking. Type

```
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 Three-button mouse
SCSI	Disk drives Tape drives
Floppy disk	720KB diskette
Parallel port (Centronics)	Printer
Two serial ports (Zilog SCC) Two serial ports (MFP)	Printers Dumb terminals 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`

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

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.

`Size (0)? 415435`

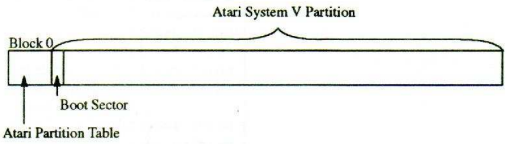
***Return** displays the partition table*

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

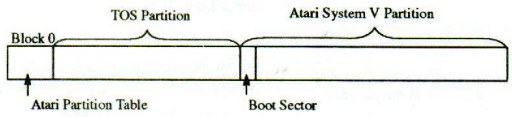
asks the same questions as the previous example:

```
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.**

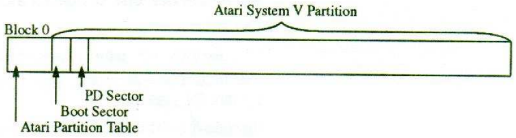
   The ***pdsector*** is a block that describes the physical layout of the disk.

   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.

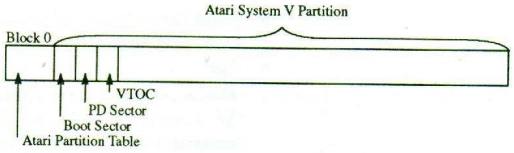
   prtvtoc /dev/rdsk/c3d0s0 > vtocdata prints the default VTOC into a file named *vtocdata* which you can then edit.

   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.

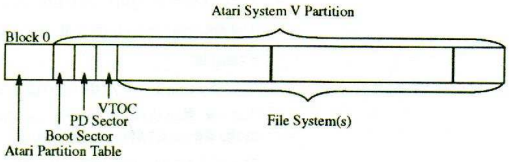


6. **Create one or more file systems using the mkfs command.**

   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:

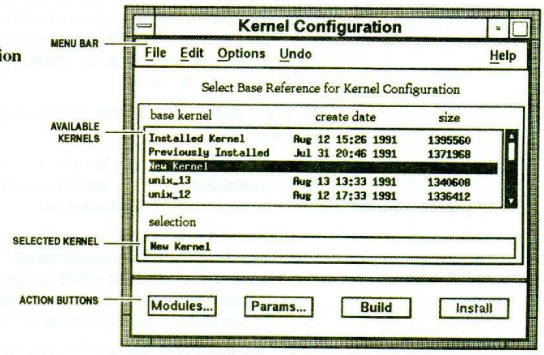
1. Select the base kernel to be changed.
2. Change the modules included in the kernel.
3. Change the kernel parameters.
4. Build the kernel.
5. Install the kernel.

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.

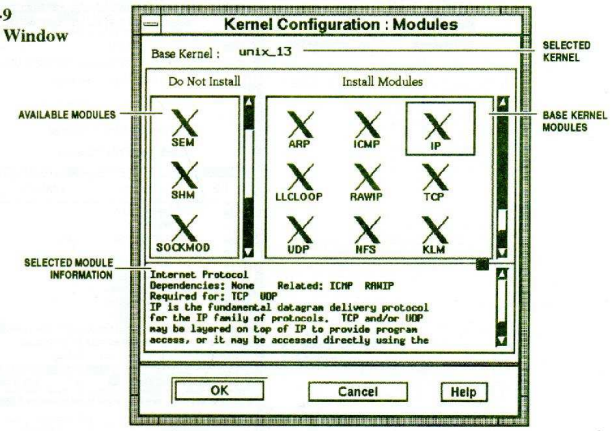
Other available modules are represented on the left side of the window. A scroll bar appears if there are too many icons to display.

- 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.

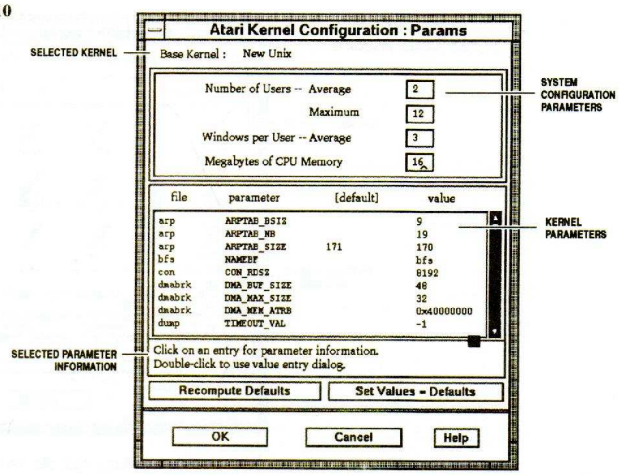
If changes are made, a new entry (**unix_[n+1]**) is added to the list of kernels in the main Kernel Configuration window and is automatically selected. Until it is successfully built, the name of this kernel appears in the list as `New Unix` and the **Install** button is disabled whenever this kernel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



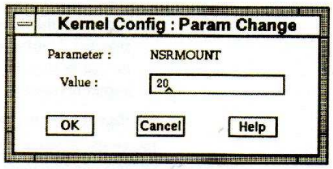
Defaults button to update the default column of kernel parameters. (Only the megabytes of the CPU memory entry have any effect on the default values in this release.)

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