Replacement Pages for the Sun386i Developer's Guide

November 1988

Sun Microsystems, Inc. • Two Federal Street • Billerica, MA 01821 • (617) 667-0010

Part No: 814-5020-01
Revision A of November 1988

These replacement pages are for the *Sun386i Developer's Guide*, part number 814-1009-10, Revision A, May 1988. They are the same pages that were issued in June 1988, when the Sun386i workstation was first released.

The *Sun386i Developer's Guide* is included in the *Sun386i Developer's Toolkit Documentation Set*. Please remove existing pages in the *Sun386i Developer's Guide* and replace them with the pages in this package, according to the table shown below.

Replace Pages	*With Pages*
ix–xii	ix–xii
67–74	67–74
95–98	95–98C
111–116	111–116
137–146	137–146A
153–154	153–154
179–180	179–180
193–207	193–208
273–287	273–286

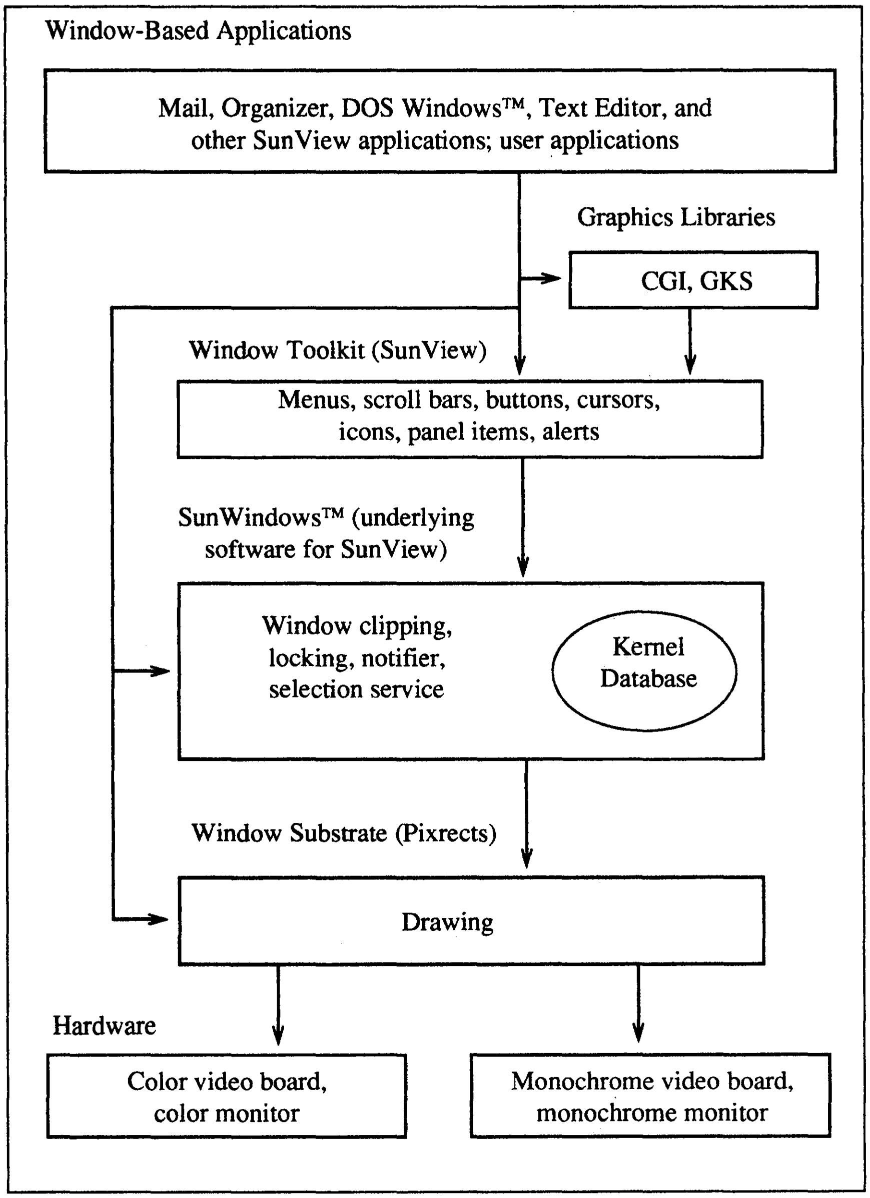
Window-Based Applications
Mail, Organizer, DOS Windows™, Text Editor, and other SunView applications; user applications
Graphics Libraries
CGI, GKS
Window Toolkit (SunView)
Menus, scroll bars, buttons, cursors, icons, panel items, alerts
SunWindows™ (underlying software for SunView)
Window clipping, locking, notifier, selection service
Kernel Database
Window Substrate (Pixrects)
Drawing
Hardware
Color video board, color monitor
Monochrome video board, monochrome monitor

Figure 6-1 *Window System and Graphics Software*

As shown above, the Sun386i system supports a version of the ANSI Computer Graphics Interface (CGI) standard graphics package for generation of two-dimensional images, called SunCGI. In addition, the Sun386i system supports SunGKS, the well-established Graphical Kernel System (GKS) standard for interactive two-dimensional graphics. Both SunCGI and SunGKS are unbundled products.

The next section discusses the Sun Organizer™, one of the window-based applications shown above, which you can use to create icons for your application's files.

The **organizer** Program

Organizer (organizer(1)) displays the contents of directories by using icons to denote file types. Pop-up menus, panel buttons, and property sheets provide SunOS file system commands such as mv(1), cp(1), rm(1), lpr(1), edit(1), open(2V), rename(2), chmod(1V), find(1), and mkdir(1).

Users can display directories with or without icons, and they can sort the contents of these directories by name, file type, size, or date. In addition, with the Show Map feature users can graphically view or browse the file system hierarchy in one window. You can create icons for your application's files and include them in the .orgrc file, described in the following section. When you do, organizer automatically displays your file-type icons in its Show List and Show Map windows.

For background information on the organizer interface, refer to the *Sun386i User's Guide*, *Sun386i Advanced Skills*, and the on-screen *Organizer Handbook.*

Displaying File Types for Your Applications

organizer(1) can display three-color icons representing your application's files if you:

- Create icons for your file types.
- Load your .orgrc file as share/data/.orgrc and your icon files as share/icons/*icon_names*.icon as part of the installation procedure. The administrator loading your application must create an *application_name* directory and go (cd(1)) to that directory before installing your files, as shown on page 96. Page 207 describes the suggested hierarchy for your application files.
- As part of your installation notes, instruct system administrators to create a volume for your application and to append your .orgrc file to ~users/defaults/.orgrc and to ~*groupname*/defaults/.orgrc for each group in use. Then users must issue the cat(1) command to append your .orgrc file to their individual version of the file in ~/.orgrc after your application is installed. Finally, users must quit from and re-enter Organizer to see your application's icons. (source(1) does not work on .orgrc.) See pages 96-98B for details.

After users append your .orgrc to their own copy of the file, they can open a file or run an application by double-clicking on the icons you supply. If an administrator appends your .orgrc file to ~users/defaults/.orgrc, the default version of the file, then user accounts created after your application is installed will automatically receive a copy of .orgrc that contains your icon information.

The steps to create and incorporate a file-type icon into organizer are:

1. Determine the name-matching expression to identify your file type. For example, the defaults that come with the system are *.c for C program files, *.h for header files, *.o for object files, and *.icon for icon files. It is the responsibility of the system administrator to rename duplicate file-type extensions that might occur.
2. Create one set of icons per file type that you are defining with the iconedit(1) utility. To enable three-color icons, you must create:
   - A background icon for the icon's background and
   - A foreground icon, which goes on top of the background icon

   To create two-color icons, you need only specify a background or foreground icon.

3. Quit from the organizer if it is running.
4. Add entries describing your icons to the version of .orgrc in your home directory (~/.orgrc). The format of .orgrc is shown in the following sections.
5. Re-enter organizer to view the icons created or changed.

After restarting organizer, you can use coloredit(1) to change the colors of icons. When you quit from organizer, the changes made are saved to the version of .orgrc in your home directory.

Once you determine the colors you like best, make a copy of your home .orgrc file for distribution.

.orgrc Parameters

.orgrc files contain two kinds of parameters:

- File-type parameters — You must create one set of file-type parameters for each icon accompanying your software. These parameters are described in the next section, with required parameters shown in bold.
- Color-palette parameters — These parameters define the default colors that the Sun386i system uses for directory, text, executable, and device files. You can use the colors supplied by these parameters for your icons, add to this file if your application uses files other than the four default types, or ignore this section of .orgrc and use your own RGB values in your file-type parameters. These parameters are described on the next page.

All .orgrc parameters, as well as all values for parameters, are case insensitive.

File-Type Parameters

Begin File Type Definition
These four words are required to denote the start of each file type within .orgrc.

Name
The expression used for name matching; you can use any valid SunOS wildcard. For instance, for a file type ending with .pbj, you could enter *.pbj as a value for this parameter. Another example is *.s for SCCS (Source Code Control System) files. Alternatively, you could enter the exact file name, as with the core file type provided with the Sun386i system.

Background Icon
The path name of the background icon that organizer will display.

Foreground Icon
The path name of the foreground icon (placed on top of the background icon) that organizer will display.

Name Color
The RGB values for the color of the text of the file name as it appears on the icon. Also used to color the rectangle that surrounds a file when it is selected.

`Icon Background Color`
The RGB values for the background color of the icon.

`Icon Foreground Color`
The RGB values for the foreground color of the icon.

`Highlight Name Color`
The RGB values for the color of the text of the file name when the file or directory is selected (highlighted).

`Execute Application`
The name of the application to call to open or execute this file type. Some file types that users should not open are object files, intermediate database format files, libraries, and binary data files.

If the application that you're opening accepts file name arguments, you can use the `$(FILE)` keyword to tell `organizer` to open that application's files when users double-click on a file name. Just use `$(FILE)` instead of the file name in the command line, as shown in the following example:

```
    Execute Application = textedit $(FILE)
```

`Edit Application`
The name of the application to call to edit this file type. If you do not supply a value for this parameter, users will be unable to edit any files of this file type from within `organizer`. As with `Execute Application`, you can specify the `$(FILE)` keyword so that users can edit the selected file.

`Print Application`
The name of the application to call to print this file type. If you do not supply a value for this parameter, users will be unable to print any files of this file type from within `organizer`. If the application can print files automatically, you can include the `$(FILE)` keyword to enable users to automatically print the application's files through `organizer`.

The `Name Color`, `Icon Foreground Color`, `Icon Background Color`, and `Highlight Name Color` parameters are optional because you can use `coloredit` to determine icon colors; when you quit from `organizer`, it automatically writes the colors chosen to `.orgrc` in your home directory. You can then include the RGB values from your home `.orgrc` in the version that you supply with your application.

`End File Type Definition`
These four words are required to denote the end of the definition.

Color Palette Parameters

Instead of providing numerical values for your icon colors in the preceding color-related parameters, you could use the names of the color palette parameters shown below for directory, text, executable, and device file icons. You might want to use these parameter names instead of RGB values in your file-type definitions because:

- It's easier to use them than to use trial and error to determine colors for your applications.
- The colors of your directory, text, executable, and device icons will match the default values for SunView versions of these file types; the graphic of your icons rather than their colors will differentiate them.
- If users change the color palette RGB values, your icons will match the colors they choose.
- You might not want to use any more of the colormap on your icons, particularly if your application uses a lot of color (the Sun386i system permits a maximum of 256 colors at one time).

The default color palette portion of `.orgrc` supplied with the Sun386i system follows.

```
Begin Color Palette
 Background Color = 255, 255, 255
 Directory Name Color = 0, 146, 236
 Directory Icon Foreground Color = 255, 227, 185
 Directory Icon Background Color = 114, 45, 0
 Directory Highlight Name Color = 255, 247, 9
 Text Name Color = 0, 166, 143
 Text Icon Foreground Color = 255, 255, 255
 Text Icon Background Color = 0, 0, 0
 Text Highlight Name Color = 255, 255, 0
 Executable Name Color = 255, 0, 104
 Executable Icon Foreground Color = 243, 255, 254
 Executable Icon Background Color = 157, 162, 187
 Executable Highlight Name Color = 255, 247, 9
 Device Name Color = 111, 111, 111
 Device Icon Foreground Color = 243, 255, 254
 Device Icon Background Color = 157, 162, 187
 Device Highlight Name Color = 255, 255, 0
 Scrollbar Color = 0, 87, 185
End Color Palette
```

Most of the color palette parameters correspond to file-type parameters. For instance, to use the default colors for a text file icon, include the lines below in your file-type definition:

```
Name Color = Text Name Color
Icon Foreground Color = Text Icon Foreground
  Color
Icon Background Color = Text Icon Background
  Color
Highlight Name Color = Text Highlight Name
  Color
```

The first and last color palette parameters, `Background Color` and `Scrollbar Color`, do not have file-type counterparts.

If your application uses files that don't fit one of the four default categories, you can add to the color palette section. However, do not change the defaults that are there, since these are the colors that Sun wants users to see; users can modify these colors if they so choose.

Alternatively, you might want to provide your own icon colors to make them stand out (but be aware that users can change these also). If so, provide RGB values instead of color palette parameter names in your file-type definitions.

Sample `.orgrc` Entries

This section provides sample entries for the `.orgrc` file for an application that will be available over a network. If your application will be available only on the system where it is installed, then use the path `/usr/local/`*application_name* instead.

```
Begin File Type Definition
 Name = *.c
 Background Icon = /vol/application_name/
   share/icons/bkgd.c.icon
 Foreground Icon = /vol/application_name/
    share/icons/frgd.c.icon
 Name Color = 236, 121, 85
 Icon Background Color = 205, 205, 254
 Icon Foreground Color = 51, 19, 92
 Highlight Name Color = 255, 247, 9
 Execute Application = cmdtool vi $(FILE)
 Edit Application = cmdtool vi $(FILE)
 Print Application = lpr $(FILE)
End File Type Definition

Begin File Type Definition
 Name = *.icon
 Background Icon = /vol/application_name/
   share/icons/bkgd.icon
 Foreground Icon = /vol/application_name/
   share/icons/frgd.icon
 Name Color = 124, 61, 140
 Icon Background Color = 243, 255, 254
 Icon Foreground Color = 157, 162, 187
 Highlight Name Color = 255, 254, 8
 Execute Application = iconedit $(FILE)
 Edit Application = iconedit $(FILE)
End File Type Definition

Begin File Type Definition
 Name = *.bits
```

```
    Background Icon = /vol/application_name/
      share/icons/bkgd.bits.icon
    Foreground Icon = /vol/application_name/
      share/icons/frgd.bits.icon
    Name Color = 124, 61, 140
    Icon Background Color = 243, 255, 254
    Icon Foreground Color = 157, 162, 187
    Highlight Name Color = 255, 254, 8
    Execute Application = fontedit $(FILE)
    Edit Application = fontedit $(FILE)
  End File Type Definition
```

6.2. On-Screen Help Facilities

In addition to man pages, the Sun386i system offers three areas of on-screen messages:

- Detailed instructions on how to create a new user account and how to log in
- Improved kernel error messages
- Cursor-sensitive help for SunView applications

The log-in screens are part of the new administration features, and are described more fully on page 99. This section describes kernel error messages and cursor-sensitive help for applications, including how to add to or change both types of these messages.

Kernel Error Messages

About 40 of the most frequently displayed kernel error messages have been reworded for the Sun386i system. The original messages are intercepted by a message daemon and used as keys into a message text file. The keyed messages, rewritten for clarity, then are delivered to a log file, to a SunView window, or to both.

Substituting Error Messages

There are several possible reasons for changing error message output:

- To translate the messages that Sun has reworded into another language (refer to the Native-Language Messages section on page 154 for more information)
- To intercept and reword the output of additional error messages generated by the kernel (those sent to /dev/klog by the syslogd(8) daemon)
- To intercept and reword the output of error messages generated by local programs, provided the program sends its messages to /dev/log via syslog(3)
- To intercept and reword the output of messages placed in the internet socket by programs on other systems

The steps below apply to the addition of substitute messages (the latter three cases).

1. Look through kernel source code (if you have the appropriate license), or the source code for the program containing messages that you want to replace. You can only substitute messages for programs that send their messages to /dev/log via syslog.

2. Add the text of the existing, unexpanded messages that you want to replace to /etc/In, preceding each message with a unique number. (You also can include suppression instructions for these messages, as described later in this section.)
3. Add the substitute text for these messages to /etc/Out, preceding each message with the number corresponding to the original message in /etc/In.
4. Use the `kill -1` command on the syslogd process to activate the changes made.

You don't have to use the default files /etc/In and /etc/Out for original and substitute text, respectively, but it's easier if you do. If you use other files, you must include the file names in /etc/syslog.conf. This file tells the system log daemon, syslogd(8), the information needed to perform the translation or suppression. syslogd intercepts error messages, checks /etc/In and the file you specify in /etc/syslog.conf (if any) for the text of those messages, and either suppresses, substitutes, or displays the original messages.

If you use files other than /etc/In and /etc/Out, you must add an entry to /etc/syslog.conf that contains the following five fields, separated by tabs:

```
translate source facility input_file output_file
```

translate — Denotes a translation entry.

source — Specifies the source(s) of an error message, separated by commas; recognized sources are:

- klog, indicating a kernel message sent to /dev/klog
- log, indicating a message generated by a local program and sent to /dev/log
- syslog, indicating a message placed in the internet socket by programs on other systems
- *, indicating all three of the above sources

facility — Specifies messages generated by other system facilities, separated by commas:

- user, indicating messages from user processes. This is the default for messages from programs or facilities not listed in syslog.conf.
- kern, indicating messages from the kernel
- mail, indicating messages from the mail system
- daemon, indicating messages from system daemons such as ftpd(8) and routed(8)
- auth, indicating messages from the authorization system (login(1), getty(8), and su(1))
- lpr, indicating messages from the line printer spooling system
- cron, indicating messages from the cron/at facility
- mark, indicating messages produced internally by syslogd
- *, indicating all facilities except mark

If you are using Frame, you can work on handbook files in the default help directory. If they have the appropriate links and the handbook contents page is in the Top Level (also in the default help directory), you can look at them at any time in the Help Viewer.

If you are using Interleaf, your working files will be in your `desktop` directory; you will have to transfer them to the default help directory before displaying them in the Help Viewer. However, first you must convert them to Printerleaf format by performing the steps below.

1. Pop up the Printer menu.
2. Pull right on **Printerleaf** and select **Document**. This saves the file in Printerleaf format and appends the `.pl` extension to its name.
3. When you move, copy, or rename the file in the default help directory, strip off the `.pl`.

Checking Topic Appearance and Function

As you create topics you will likely want to check their appearance and the functioning of hypertext links in the Help Viewer. To do this:

1. Copy your handbook files into the default help directory that you've specified with Defaults (if they are not already there).
2. Add your handbook to the copy of the `Top_Level` file in the default help directory so that you can link to it from the viewer's Top Level table of contents.

The following section discusses the second step.

Adding Handbooks to the Top Level

No one can access your handbook from the Help Viewer until you insert the handbook title in the Help Viewer's Top Level table of contents. `Top_Level` is a special text file (not a Frame or Interleaf file) in a subdirectory of the default help directory that you can edit for this purpose. `Top_Level` is initially in `/vol/help/language/USA-English`.

The file lists the standard set of Sun386i handbooks as a series of entries. Each entry consists of a text string in single quotes on the left and a path name and page number in square brackets on the right. The strings are displayed as underlined contents items; the path names are links to the files displayed after following the link.

You edit `Top_Level` as you would any other text file. The only things to note are the use of:

1. Backslash characters in front of apostrophes, or backslashes that are part of your handbook's title
2. Number signs in front of comments

The next section describes installing your help files, and includes suggestions for appending your handbook entry to `Top_Level` via a shell script.

Installing Your Help Files

This section contains the required steps that an installation script must perform to load help for your applications, as well as the steps that installation instructions must include to ensure that users can access your help.

Adding to the Top Level During Installation

The Top Level might already be customized to a certain extent. Therefore, it's suggested that you write a shell script that appends your handbook entry to the `Top_Level` file in `/vol/help.master` (a writable version of `/vol/help`) when a customer installs your application. Such a script would resemble the following:

```
      echo "'application_name Handbook' \
         [application/application_name_Handbook]" >>  \
         /vol/help.master/language/USA-English/Top_Level
```

Or, to check for the presence of your handbook before adding it:

```
      cd /vol/help.master/language/USA-English
      if grep 'application_name Handbook' Top_Level >   \
         /dev/null 2>&1
      then
        echo "'application_name Handbook' is already   \
         installed in 'Top_Level'

      else
        echo "'application_name Handbook'    \
         [application/application_name_Handbook]" >>   \
         /vol/help.master/language/USA-English/Top_Level
        echo "'application_name Handbook' installed in   \
         'Top_Level'"
      fi
```

Required Installation Steps for Applications with Help Files and Organizer Icons

As described on page 80, the default help directory is `/vol/help`. It is recommended that administrators retain this as the default directory for easier maintenance, and for disk space conservation and network-wide availability of Sun386i and third-party help files. For the Help system to access your files through `/vol/help`, you must add a few steps to your installation script, and must instruct system administrators to perform certain steps before and after installing your software. These steps are also mandatory if your application includes Organizer icons for your application's files.

Pre-Installation Instructions for System Administrators

Be sure that your installation instructions tell an administrator to do these steps *before* installing your application.

1. Log in as superuser on the system where you'll be installing the application.
2. Go (`cd(1)`) to the `/files<`*n*`>/vol` directory if the application will be available over the network, or to the `/usr/local` directory if it will be available only on this system. Choose the `/files` directory (`/files` or `/files1`, `/files2`, and so on) based on space and use.
3. Enter `mkdir` *application_name* to create an *application_name* subdirectory.
4. Enter `cd` *application_name.*
5. If the application is on diskette, enter:

   `bar xvf /dev/rfd0a` *installation_script*

   If the application is on high-density tape, enter:

   `tar xvf /dev/rst8` *installation_script*

If the application is on low-density tape, enter:

```
tar xvf /dev/rst0 installation_script
```

6. Enter *installation_script*.

The final step will invoke the installation script that you supply, which must perform the steps listed in the next section.

Installation Script Steps

As part of the installation procedure, your script must load the help files and the rest of the application as follows:

For diskettes, use the command:

```
bar xvf /dev/rfd0a directories
```

For high-density tapes, use the command:

```
tar xvf /dev/rst8 directories
```

For low-density tapes, use the command:

```
tar xvf /dev/rst0 directories
```

where *directories* is `language/`*language*`/help/`*application_name*`.info` for Spot Help files and `language/`*language*`/help/`*handbook*`/`*handbook_files* for handbook files. Pages 207-208 describe the suggested directory structure for applications. This structure is **required** for applications that include on-screen help files. Note that your script could use additional options with `bar` and `tar`, depending on the options you use when putting your files on tape or diskette.

You also must include either the start-up script supplied with the Sun386i system, `/usr/etc/start_applic`, or one that you provide. Pages 144-145 provide additional information about start-up and installation scripts.

Installation Instructions for System Administrators

If your application includes on-screen help files, or if you're adding Organizer icons for your application's files, whoever loads your application **must perform** the steps in the rest of this section to access your help or icons. Although these instructions also appear in *Sun386i Advanced Administration*, you should include them in the written installation instructions that you provide with your application.

This section assumes that the application will be available throughout a Sun386i network or for a domain specified. If for licensing reasons your application should only be available on the system where it is installed, the administrator should install the application in `/usr/local` (instead of `/files`*<n>*`/vol`) and should not export or create a volume for the application. Instead, administrators should only perform the steps starting with the Registering an Application section on page 98A.

If administrators are installing your application on a non-Sun386i network (an existing YP network with a master server that is not a Sun386i system), they must manually mount the application on all systems that will use it, and must devise their own registration procedure.

Exporting the Application

Exporting is controlled through the `/etc/exports` file (see the `exports`(5) `man` page for details). The following steps show how to create a link to an application that's being exported, export the application by editing the `/etc/exports` file, and issue the exportfs(8) command to activate modifications made to the /etc/exports file. If you want to include these steps in your installation script rather than have a system administrator perform them, see the Installation Script section on page 144.

Before performing these steps, an administrator must make sure that no one is adding a diskless client to the system and that no one is creating a user account on it through New User Accounts or SNAP. This is to avoid possible corruption of the /etc/exports file, used by both procedures and modified in the steps below.

1. Log in as superuser to the system where the application files reside.
2. Ensure that /export/vol exists by entering the command:

```
# mkdir /export/vol > /dev/null 2>&1
```

3. Create a symbolic link using the command format:

```
# ln -s /files<n>/vol/application_name \
    /export/vol/application_name
```

4. Enter the following line to add an entry for the application in the /etc/exports file:

```
# echo "/export/vol/application_name \
    -ro,access=domain" >> /etc/exports
```

5. Enter the following command to export the application:

```
# exportfs /export/vol/application_name
```

The application is now available to users on any system within the domain netgroup (see the netgroup(5) man page) via the mount(1) command. The steps in the next section make the application accessible from an automatically mounted volume.

Creating a Volume

Creating a volume for an application's files makes administration and maintenance of that application easier. Volumes are attached to the Sun386i file system by the automounter (automount(8)), so there's no need to edit /etc/fstab files on client machines when a volume is moved. Administrators must export the application, as described in the previous section, before performing these steps.

1. Enter rlogin 'ypwhich -m auto.vol' to log in to the Yellow Pages master.
2. Become superuser by entering **su** and the superuser password.
3. Create a volume for the application by entering the following command, which adds an entry to the /etc/auto.vol file:

```
# echo "application_name  -ro \
  system:/export/vol/application_name" >> \
  /etc/auto.vol
```

   where *system* is the name of the system where the application files reside.
4. Rebuild the /etc/auto.vol map by issuing the commands:

```
# cd /var/yp; make
```

The next section describes how administrators will use the start-up script that you supply to *register* your application. If your application includes on-screen help files or Organizer icons, you must supply a start-up script and an administrator must register your application as shown in the next section. You are strongly encouraged to include a start-up script even if you are not providing help files or icons for your application. Page 145 provides more information about start-up scripts.

Registering an Application

Registering an application enables any user that can access the application to do so simply by entering *application_name*, which is the start-up script that invokes the application. Users need not know the location of the files nor the architecture of the machine where the files reside. Another benefit is that users do not have to modify their own environment (such as `.login` and `.cshrc` files) to use new applications. To register an application, an administrator must put a copy of the start-up script that you supply with your application in `/vol/local.master/bin`, which is part of every user's `$PATH`.

By default, the `/vol/local.master` directory and the master copy of the `/etc/auto.vol` file are on the same system, so administrators might be able to enter the following commands on the same machine where they created the volume in the previous section. If administrators installed the application in `/usr/local`, they should go directly to step 2.

1. Enter the command:

   ```
   # ypmatch local.master auto.vol
   ```

   The system displays:

   ```
   system:/export/vol/local
   ```

   If the system name displayed is the system where you created the volume, then you can perform these steps on the same system; if the name is different, then enter:

   ```
   # rlogin system
   # su
   ```

   where *system* is the name of the system displayed.
2. If you've exported and created a volume for the application, enter:

   ```
   # cd /vol/local.master/bin
   ```

   If instead you've installed the application in `/usr/local`, making it available only on an individual system, you must use the format:

   ```
   # cd /usr/local
   ```

3. Then issue the `cp(1)` command to copy the start-up script specified in the installation instructions to *application_name*. For instance, if the application uses the start-up script supplied with the Sun386i system, the command line would be in the format:

   ```
   # cp /usr/etc/start_applic application_name
   ```

The next two sections describe the final steps that an administrator must perform after installing an application with help files or Organizer icons, respectively.

Additional Steps for Applications with Help Files

After an administrator performs the steps in this section, your application and help files will be available to any user on the network, without users having to modify any of their environment variables. You might want to follow these steps yourself after your help files are completed, to be sure that everything is working correctly. If you do, be sure to move your files to `/vol/help/language/`*language* (page 208 lists values for *language*), and use Defaults on the SunView Menu to make sure that the Help directory is `/vol/help` if you've changed it while writing or checking your help text.

1. Log in as superuser to the system that has the master `/vol/help.master` file. To determine the correct system, use the command:

   ```
   ypmatch -m  help.master auto.vol
   ```

2. Create a link file for `.info` files in the directory `/vol/help.master`:

   `ln -s` *filename linkname*

   where *filename* is

   `/vol/`*application_name*`/language/`*language*`/help/`*application_name*`.info`

   if the application will be available on a network; if the application will be available only on systems where it is installed, *filename* must be

   `/usr/local/`*application_name*`/language/`*language*`/help/`*application_name*`.info`

   Regardless of where the application is installed, *linkname* has the format:
   `/vol/help.master/language/`*language*`/`*application_name*`.info`

3. Create a second link file in `/vol/help.master` for handbooks. The link has the same format as for `.info` files but *filename* is
   `/vol/`*application_name*`/language/`*language*`/help/`*application_name*`/`*handbook*

   for applications available on a network; for an application available only on systems where it is installed, *filename* is

   `/usr/local/`*application_name*`/language/`*language*`/help/`*application_name*`/`*handbook*

   For either case, *linkname* is

   `/vol/help.master/language/`*language*`/`*application_name*`/`*handbook*

Additional Steps for Applications with Icons

To access your application's icons, an administrator must append your `$`*application_name*`_ROOT/share/data/.orgrc` file to `~users/defaults/.orgrc` (the version shipped with the Sun386i system) and to `~`*groupname*`/defaults/.orgrc` for each group in use. Then an administrator must tell current users (via electronic mail) to:

1. Use the `cat(1)` command to append your `.orgrc` file to their personal copy of the file in `~/.orgrc`.
2. Quit from and re-enter Organizer to see your application's icons. (`source(1)` does not work on `.orgrc`.)

When an administrator appends your `.orgrc` file to `~users/defaults/.orgrc` and `~`*groupname*`/defaults/.orgrc`, users subsequently added to the system do not have to perform the above two steps.

6.3 Administration Facilities

The ease-of-use administration facilities described in this section are:

- `snap(1)`
- Automatic System Installation
- New User Accounts

All of these facilities are built on top of two new SunOS 4.0 features: secure RPCs (Remote Procedure Calls) and the Yellow Pages (YP) updater program. Secure RPCs use a public key encryption technology that provides user authentication in the network. The YP updater provides a way to update YP maps from programs. To use these features, your network must be running the Yellow Pages and a Sun386i system must be the YP master. *Sun386i SNAP Administration* describes RPCs and the Yellow Pages in detail.

The next three sections present an overview of the administration facilities on the Sun386i system. For details, see *Sun386i SNAP Administration.*

The snap Program

`snap(1)` provides a window interface for users with the required privileges to browse, add, delete, and modify user accounts, user groups, systems, modems, terminals, and printers. When a user confirms changes made, `snap` makes the appropriate changes to the Yellow Pages facility and directories. `snap` privileges are implemented through membership in special groups. As shipped, `snap` gives all users all `snap` privileges; most sites probably will want to restrict privileges to some degree.

`snap` also enables file backup and restore. The backup facility provides easy-to-use personal and system-wide backup functions. Both full and incremental backups are possible. Users can also restore files with `snap`.

Automatic System Installation

Using Automatic System Installation, a user can add a system to an existing Sun386i network in about 30 minutes after unpacking. The first time a user powers on the Sun386i system, one of the following four scenarios occurs, depending on the machine's configuration.

For a standalone, the system asks for confirmation that it is not going to be added to a network, and then continues with the boot procedure.

Memory

The Sun386i system allocates 640 Kbytes of its virtual memory for each application running in DOS Windows. In addition, up to 8 Mbytes of expanded memory is available for each application that uses the Lotus®-Intel-Microsoft (LIM) expanded memory specification. Since the Sun386i system allocates virtual memory, no added hardware is required for this support. If you want to take advantage of expanded memory, you must design applications to access LIM memory by switching parts of the program in and out of LIM address space, which starts at the standard location `D0000`. LIM memory is available to each DOS Windows application that can use it, even if several windows are running simultaneously.

Naming Your PC Applications

Because MS-DOS and SunOS systems have different file-naming conventions, the SunOS system provides a file name mapping scheme that enables you to specify programs from within either operating system. However, mapped file names are only temporary references; name mapping does not produce the same result each time. Therefore, do not build mapped names into your applications. The best course of action is to follow MS-DOS file naming conventions whenever possible. If you follow the suggestions below, file mapping will not be an issue for you or the users of your applications.

- Names can be up to eight characters (without an extension), or up to eleven characters (with a period and three-character extension). Only programs with `.EXE`, `.COM`, or `.BAT` extensions are executable from within MS-DOS.
- MS-DOS names are not case-sensitive, but almost all SunOS commands are lowercase; therefore use lowercase letters for all of your file names.
- Do not use these characters in file names: " . / [ ] : | < > + = ; ,
  You can use a period only as a separator between the file name and an extension.

Issuing SunOS Commands from DOS Windows

The system comes with a number of SunOS commands that you can issue from within DOS Windows, provided that `/etc/dos/unix` is part of the your MS-DOS `path`. These commands are MS-DOS `.COM` programs that point to the actual SunOS commands. They accept the ampersand (&), so you can run them in the background. The preinstalled commands are listed below.

Table 7-1 *Preinstalled SunOS Commands*

```
at
awk
calendar
cat
chgrp
chmod
chown
cmdtool
cmp
comm
cp
csh
cut
date
diff
echo
egrep
fgrep
file
find
grep
head
kill
ln
lpq
lpr
lprm
ls
mail
mailtool
make
man
mesg
mkdir
more
mount
mv
nice
passwd
pr
ps
pwd
rlogin
rm
rmdir
rsh
sed
size
sort
split
stty
tail
tar
tee
time
tr
umount
unix
vi
wc
whatis
whereis
who
write
yppasswd
```

To install additional SunOS commands for your applications, become superuser by entering the su command and then enter:

```
cd /etc/dos/unix
ln -s unix.com newcommand.com
```

at the SunOS prompt.

Text-Only Applications

dos(1) opens a window whenever you invoke most PC programs. However, this is not the case for text-only applications delivered with the Sun386i system. Text-only applications are those that do not attempt to address the cursor, clear the screen, or display graphics. DIR is a good example of such a program. vi is not a text-only application, since it controls the cursor position and makes assumptions about the screen geography.

Text-only applications do not require an 80x25 display. Therefore, if implicit execution is set with the DOSLOOKUP environment variable, dos executes text-only applications in a cmdtool or shelltool window, rather than automatically popping open a new DOS window. You can add to the list of text-only applications that dos recognizes by including the application's name to your setup.pc files, as a value for TEXT. (Refer to page 114 for more information about the setup.pc file.) The list of text-only applications that are shipped with the Sun386i system is shown in Table 7-2.

Table 7-2 *Preinstalled Text-Only Commands*

ATTRIB	DEBUG	LABEL	SUBST
ASSIGN	DIR	LINK	SYS
BACKUP	DISKCOMP	MODE	TIME
BREAK	DISKCOPY	RECOVER	TREE
CHKDSK	EXE2BIN	REPLACE	TYPE
COMMAND	FDISK	RESTORE	VER
COMP	FIND	SELECT	VERIFY
COPY	FORMAT	SHARE	XCOPY
DATE	JOIN	SORT	XDIR

Running make(1) on MS-DOS Targets

In addition, if you specify MS-DOS files as text-only files, you can run compilers, assemblers, and SunOS make(1) files on them in cmdtool or shelltool windows. For example,

```
file.exe: file.c
          dos -w -c cc file.c
```

where *file*.exe is the MS-DOS target file, and *file*.c is the file that the target file depends upon. The -w option to the dos command declares *file*.c as a text-only file, and the -c option indicates that the command, in this case cc, follows.

File Permission Differences

Generally, access to files is the same under SunOS and MS-DOS systems. The exceptions are:

- MS-DOS does not recognize execute restrictions. That is, any user with read permission to a file can execute that file. Without read permission, users cannot execute files.
- Drive C: does not support SunOS file permissions, since the SunOS system cannot directly access files on drive C:. However, because the SunOS system views drive C: as one large file, you can restrict access to all drive C: files to a specific owner or group.

7.3. Peripheral Issues

The MS-DOS drive designations are:

Drives A: and B: — Reserved for diskettes.

Drive C: — A "virtual" hard disk of up to 20 Mbytes, that the SunOS system cannot access; use this drive only to install copy-protected or install-protected PC software.

Drives D: through S: — Virtual hard disks tied to system SunOS directories that can expand as required; use these drives for data files and unprotected PC applications.

All MS-DOS drivers listed in `CONFIG.SYS`, the MS-DOS configuration file, must actually be on drive C:, where `CONFIG.SYS` resides. This is because MS-DOS loads drivers before it begins to communicate with the SunOS system (toward the end of the `AUTOEXEC.BAT` file), and drive C: is the only drive that MS-DOS can access until SunOS communications are activated. The message `Bad or missing` *xxx*`.sys` appears if you try to access a device that has a driver that is not on drive C:.

You can add MS-DOS peripherals either by:

- Adding an AT card that uses an MS-DOS driver provided by the card manufacturer
- Adding an AT card that uses a device-specific driver that you write

Regardless of the method used, you must add information about new drivers to three files—`CONFIG.SYS` (an MS-DOS file), and `setup.pc` and `boards.pc` (two SunOS files described in the following sections). Then you must invoke DOS Windows from the Desktop menu before using the new driver, or enter **dos -s** to save the new driver in `.quickpc`, a quick-start file containing a snap-shot image of MS-DOS. (Refer to *Sun386i Advanced Skills* for more information.)

The `setup.pc` file contains configuration information on all devices attached to a system that users might want to access via MS-DOS, including the SunOS files associated with those devices. The `boards.pc` file contains a list of the boards that only MS-DOS, not the SunOS system, can access on the system. MS-DOS cannot access a peripheral listed in the `setup.pc` file unless it is also in the `boards.pc` file. The `boards.pc` file is in the `/etc/dos/defaults` directory, and `setup.pc` is in the user's home directory, `~/pc/setup.pc`.

setup.pc File

The first time you open DOS Windows, `dos` creates a `pc` directory under the home directory, and places a copy of `setup.pc` there. You can edit this file, but generally should not delete anything in it. A description of the default `setup.pc` file follows, with number signs (#) indicating comment lines. (Descriptions shown here are not part of the default file.) For more general information about the `setup.pc` file, refer to *Sun386i Advanced Skills.*

```
# MS-DOS Device         SunOS Device Path Name

A                       /dev/rfd0c
# Diskette device name.

C                       ~/pc/C:
# Drive C: file name.

COM1                    /dev/ttya
# Specifies the serial device attached to the serial port.

LPT1                    lpr
# Specifies how to process MS-DOS LPT1 text; the default is the default printer.

LPT2                    cat >>~/lpt-2
# Specifies how to process MS-DOS LPT2 text; the default is to append to the
# ~/lpt-2 file in the user's home directory.

LPT3                    psfx80 | lpr
# Specifies how to process MS-DOS LPT3 text; the default is Epson™ FX-80
# emulation on the default printer.

SAVE                    ~/pc/.quickpc
# Specifies .quickpc, a quick-start file created with the dos -s command
# that contains a snapshot image of MS-DOS after it has read CONFIG.SYS and
# most of AUTOEXEC.BAT (up to the RUNDOS line). This file is used when any
# dos command other than dos -b (the default command issued when started
# from the Desktop menu) or dos -s is issued. Starting MS-DOS is much
# quicker with the .quickpc file.

# TEXT
# List of user-specified text-only applications, in addition to the standard ones
# shipped with the Sun386i system. Running these applications from the SunOS
# system will send output to the current window instead of opening a new DOS
# window.

# BOARDS
# List of boards from /etc/dos/defaults/boards.pc that you want
# to attempt to access upon opening DOS Windows. If a board is already in use,
# it will appear as detached in the Devices submenu on the Sun386i system. In
# this case, you can release the device from the window that owns it, and then
# attach the device from the current window.
```

boards.pc File

As with setup.pc, you also can edit the boards.pc file; however, unlike setup.pc, each system should have only one copy of boards.pc, which affects all users on the system. If you add a device to run under MS-DOS, you must include its board name and block I/O information in the file. You must also include interrupt-level information for boards that use interrupt levels, as well as indicate whether or not the device can be shared. The boards.pc file included with the Sun386i system contains a list of commonly used boards included as comment lines; you can remove the comment symbols for those boards that you have and want PC applications to use.

The following table shows the AT bus I/O address spaces that dos emulates. If you add a card in one of these address spaces, MS-DOS ignores it. If you specify an emulated address in the boards.pc file, the next time you open a window the system displays a message stating that the address range is already in use. You can turn off emulation for all but the hard disk by placing a comment character (#) at the start of the pertinent line in setup.pc.

Table 7-3 *I/O Address Space Emulation*

Address	*MS-DOS Use*
1F8 – 1FF	Hard disk emulation
230 – 237	Bus mouse emulation
278 – 27F	Parallel port 2
378 – 37F	Parallel port 1
3B0 – 3BF	Monochrome display adapter
3D0 – 3DF	Color display adapter
3F0 – 3F7	Diskette controller

No two boards in the same system can have the same interrupt level. Because many boards have a factory-set interrupt level of 3, occasionally you might have to rejumper the board to set a new interrupt level, as on regular PCs. You must then also change the interrupt-level information in the boards.pc file before accessing the attached device. Table 7-4 shows the availability of interrupt levels for the Sun386i system. For more details about adding a board to the boards.pc file, refer to *Sun386i Advanced Skills*.

Table 7-4 *Interrupt Level Availability*

Interrupt Level	*Availability*
0	Unavailable; used for timer emulation
1	Unavailable; used for keyboard emulation
2	Unavailable; used for interrupt controller 2 cascade
3	Available for board (specified in `setup.pc`)
4	Available for board, unless COM1 emulation in use (specified in `setup.pc`)
5	Available for board, unless LPT2 emulation in use (specified in `setup.pc`)
6	Unavailable; used for diskette emulation
7	Unavailable; used by built-in parallel port
8	Unavailable; used for real-time clock emulation
9	Unavailable; used by built-in serial port
10	Available for board
11	Available for board
12	Available for board
13	Unavailable; used for 8087 numeric coprocessor emulation
14	Unavailable; used for hard disk emulation
15	Available for board

7.4. Capabilities and Limitations

This section describes some MS-DOS features and limitations that you should know about. It contains sections on:

- Conversion programs for converting text files from MS-DOS to the SunOS system and vice versa
- Differences between the MS-DOS and SunOS command interpreter
- Determining the DOS Windows number to create unique file names and help avoid network collisions
- 80386 instructions supported
- Limitations such as those relating to screen height, remote port use, certain types of applications, running simultaneous versions of MS-DOS applications, interrupt rates, and space issues

Converting Between MS-DOS and SunOS Text Files

MS-DOS and the SunOS system have slightly different conventions regarding text file delimiters. Consequently, the Sun386i system includes special utilities to convert files from one set of conventions to the other. The program to convert MS-DOS files to SunOS files is called `dos2unix(1)`; the program to convert SunOS conventions to MS-DOS conventions is called `unix2dos(1)`. The Sun386i system contains two versions of each program, a SunOS version and an MS-DOS version, to make it easier to run both utilities from either system.

Conversion does not happen automatically. You must invoke these programs as necessary, and can do so from either the MS-DOS prompt or the SunOS prompt. Include a source file name and a destination file name on the command line.

9

Applications Delivery

9

Applications Delivery

This chapter describes software delivery — both how Sun delivers its software for the Sun386i system and the preferred method for you to deliver your applications for this system. The first part of the chapter discusses the division and distribution of system software in two major parts, and the groups of files, called *clusters*, constituting those parts. The last section describes the steps you should follow to enable users to easily install your applications.

9.1. System Software Overview

Sun386i system software is divided into two major sections: Application SunOS and SunOS Developer's Toolkit. Figure 9-1 below shows the two major divisions of system software and their subsets.

Application SunOS (unbundled)

Hardware Diagnostics (separate diskette)

Core System (shipped on disk)

Optional Clusters (on diskettes or tape)

Recovery Software (on diskettes or tape)

SunOS Developer's Toolkit (unbundled;
loadable clusters on diskettes or tape)

Figure 9-1 *System Software Divisions*

Each site receives the core system on the Sun386i system disk, as well as the *Sun386i Owner's Set* documentation. All other system software and documentation is unbundled; that is, users must purchase both Application SunOS and Developer's Toolkit to have full SunOS 4.0 functionality and accompanying documentation. Sun strongly recommends that each user site purchase at least one copy of Application SunOS. By doing so, a site receives all four pieces of Application SunOS shown in Figure 9-1, plus the *Owner's Supplement Documentation Set.* The next section describes Application SunOS more fully. In addition, users can order the Developer's Toolkit, discussed in Section 9.3 on page 143.

9.2. Application SunOS

Application SunOS includes:

- Hardware Diagnostics — a set of standalone diagnostics available on diskette
- Core system — the base system, providing the ability to run most commercially available Sun and third-party applications; shipped on the Sun386i disk
- Optional clusters — additional software for capabilities such as extended mail and extended networking; available on diskettes or tape
- Recovery software — a backup version of the core system, available on diskettes or tape

Users can purchase Application SunOS on diskettes (approximately 26) or quarter-inch tape (two). In addition to the software listed above, users who purchase Application SunOS also receive the *Sun386i Owner's Supplement Documentation Set.* The following subsections provide details of each Application SunOS subset.

Hardware Diagnostics

The first part of Application SunOS is a set of standalone Hardware Diagnostics. These programs do not require the SunOS operating system. You should run hardware diagnostics when:

- You cannot start your system
- The system displays numerous messages indicating a hardware problem
- You have upgraded your system with a new frame buffer, memory board, or hard disk (to make sure that the new part works properly)
- Your system crashes repeatedly

For information about how to run Hardware Diagnostics and the individual tests they perform, refer to *Sun386i System Setup and Maintenance.*

Core System

The core system provides the minimum subset of the SunOS operating system required by every user. It is sufficient to allow users to operate most commercially available Sun and third-party applications.

The core system includes software that users always need. It uses about 19 Mbytes of disk space and is preloaded by Sun manufacturing on the formatted hard disk that comes with each system (either 91 or 327 Mbytes). There is no automated method to remove any part of the core system, since users should leave all of it on the disk.

The core system includes the groups of files listed below. The file `/usr/lib/load/filesizes` contains the names and sizes of files in each group.

base_root — the root directory (`/`), which includes the kernel; system databases and start-up files; single-user mode requirements; commands such as `automount`(8), `chown`(8), `fastboot`(8), `fasthalt`(8), and `reboot`(8); the `adb`(1) and `kadb`(8S) debuggers; and the file `/usr/lib/load/filesizes`

sunview — SunView tools, icons, commands, and demos, as well as all Sun screen fonts

boot_server — the boot server for booting diskless Sun386i systems from Sun386i system servers, as well as the boot server enabling the Sun386i system to be a server for Sun-2, Sun-3, and Sun-4 systems

encryption — file encryption commands such as `des`(1)

calendar — `calendar`(1) program and required files

basic_commands — most commonly used user commands such as `date`(1V), `grep`(1V), `arch`(1), `csh`(1), `passwd`(1), `crontab`(1), and `kill`(1), as well as `ed`(1), `ex`(1), and `vi`(1) editors

mail — basic mail directories, files, and commands such as `mail`(1), `biff`(1), `sendmail`(8), and `newaliases`(8)

at_commands — commands such as `at`(1), `atq`(1), and `atrm`(1), for executing commands or scripts at a later time

print_spooler — printing commands such as `lpc`(8), `lpd`(8), `lpq`(1), `lpr`(1), and `lprm`(1)

non_readonly — configuration files, spool directories, and other nonread-only files required by optional software such as the Network File System (NFS), the print spooler, extended mail, the audit trail maintenance package, and `uucp`(1C) and `tip`(1C)

sun_specific_commands — commands such as `click`(1) and `screenblank`(1)

online_help — Spot Help and Help Viewer files, plus new kernel error messages

key — encryption keys for secure networking; includes `chkey`(1), `keylogin`(1), `keyserv`(8C), and `keyenvoy`(8C)

basic_networking, **rpc_base**, and **nfs** — contain networking software, with the exception of the boot server; includes network configuration files, daemons, and administrative and user commands such as `ping`(8C), `rmt`(8C), `rcp`(1C), and `rlogin`(1C)

ease_of_use — `snap`(1) and `organizer`(1) programs

yellow_pages — the Yellow Pages database

dos — MS-DOS 3.3, required to run DOS Windows

load — the commands required to load and unload clusters, including `load`(1), `unload`(1), `loadc`(1), `unloadc`(1), and `cluster`(1)

Optional Clusters

The optional clusters included with Application SunOS are comprised of sets of related programs and files that users might need on the hard disk, in addition to the core system. Users can add individual clusters after installation by using the load(1) or loadc(1) commands (or the snap(1) administration tool), and can subsequently remove them by using the unload(1) and unloadc(1) commands. (Pages 12-13 describe all four commands.) When all optional clusters are loaded, they take about 14 Mbytes of disk space. A file that's part of the core system, /usr/lib/load/filesizes, lists the sizes of these and all other system software files. *Sun386i System Setup and Maintenance* provides more details about the contents of these clusters, listed below.

mail_plus — extended mail commands such as from(1), vacation(1), uuencode(1C), uudecode(1C), and mailstats(8)

spellcheck — spell(1) program and related commands

accounting — basic accounting programs and commands such as ac(8), accton(8), pac(8), last(1), and lastcomm(1)

sysV_commands — basic System V commands such as uname(1V), echo(1V), expr(1V), cat(1V), grep(1V), sdiff(1), and chmod(1V)

advanced_admin — advanced system administration commands such as chroot(8), dump(8), and restore(8)

extended_commands — additional commands such as pagesize(1), trace(1), logger(1), and script(1)

networking_plus — extended networking utilities and commands such as in.fingerd, in.ftpd, and in.rwhod daemons and finger(1), ftp(1C), rwho(1C), gettable(8C), and rpcinfo(8C) commands

audit — audit trail maintenance package, including the auditd and rpc.pwdauthd daemons and audit(8), audit_warn(8), praudit(8), and C2conv(8)

comm — uucp(1C), tip(1C), and related commands

doc_prep — text processing tools such as nroff(1), troff(1), neqn(1), and tbl(1), and directories required to run them

disk_quotas — quota commands such as quot(8), edquota(8), and quotacheck(8)

name_server — in.named, in.tnamed, and sendmail.mx daemons

man_pages — on-line man pages and man commands

plot — plotting commands such as plot(1G) and spline(1G)

old_commands — backward-compatible commands such as make(1), perfmon(1), clocktool(1), setkeys(1), and syslog(1)

games — on-line games, including backgammon, Boggle, and cribbage

Recovery Software

Application SunOS includes recovery software for reloading the core system, if necessary. Recovery software is available on tape and diskettes. *Sun386i System Setup and Maintenance* describes how to load this software, should you need it.

9.3. SunOS Developer's Toolkit

SunOS Developer's Toolkit is a complement to Application SunOS, not a superset. It provides everything missing in Application SunOS needed to achieve full SunOS 4.0 functionality. In addition, the *Sun386i Developer's Toolkit Documentation Set* accompanies each copy of the Developer's Toolkit purchased. The Developer's Toolkit is available on diskettes or quarter-inch tape. As with Application SunOS, you can add and remove Developer's Toolkit clusters individually with the `load`(1) and `unload`(1) commands described briefly in the next section. The file `/usr/lib/load/filesizes`, part of the core system, lists the sizes of these and all other system software files.

SunOS Developer's Toolkit includes the software listed below.

base_devel — software development commands and utilities such as the C compiler, assembler, link editor, `dbx`(1); you must load this cluster to be able to use any of the Developer's Toolkit with the exception of the `help_guide` cluster, which does not require `base_devel`

config — System V files necessary to reconfigure the kernel such as `config`(8) and the `/usr/sys` directory

sunview_devel — SunView development libraries required for writing window-based applications

help_guide — *Help Writer's Handbook* for writing on-screen help for applications (the sections Spot Help Interface and Help Viewer Interface on pages 80–94 contain much of the same information)

plot_devel — libraries such as `libplot.a` and `libplotbg.a` for development plotting functions

proflibs — profiled libraries (denoted by the suffix `_p.a`) such as `libc_p.a`, `libm_p.a`, and `libcurses_p.a`

sccs — commands required by SCCS, the Source Code Control System

sysV_devel — libraries required to port System V applications, including utilities in `/usr/5bin` and `/usr/5lib` directories

9.4. Loading and Unloading Clusters

You can issue the `load`(1), `loadc`(1), `unload`(1), `unloadc`(1), and `cluster`(1) commands to:

- Add one or more Application SunOS or Developer's Toolkit clusters to the disk after installation
- Remove one or more clusters to make space for additional ones
- Display the name of a cluster containing a specified file
- Display a summary of all Application SunOS and Developer's Toolkit clusters, including a cluster's size and whether or not it is loaded

Pages 12-13 provide more information about these commands.

9.5. Releasing Your Software

The preferred method of releasing software on tape or diskette consists of three parts:

1. Creating an installation script.
2. Supplying a start-up script for your application, or using the `/usr/etc/start_applic` script supplied with the Sun386i system.
3. Using the `bar`(1) command (available on the Sun386i system only) to place the two files mentioned above and your application onto a formatted (`fdformat`(8)) diskette, or using the `tar`(1) command to place them onto tape.

You also should provide a copyright file and installation instructions for system administrators. If you want your application to be available over the network, include the steps starting on page 97 that tell administrators to create a volume for your application and export and register it. Then users can invoke your application from any machine on the network, regardless of architecture, without having to alter their `.login` or `.cshrc` files. If you are releasing on-screen help for your application or file-type icons that users can view via the Organizer, administrators **must** follow the steps starting on page 97 to see your help files or your application's icons.

Be sure that your installation instructions state how much space your application requires. Your instructions should also minimally tell administrators to do the following steps *before* installing your application.

1. Log in as superuser on the system where the application files will reside.
2. Go (`cd`(1)) to the `/files<`*n*`>/vol` directory if the application will be available over the network, or to the `/usr/local` directory if it will be available only on this system.
3. Enter `mkdir` *application_name* to create an *application_name* subdirectory.
4. Enter `cd` *application_name*.
5. If the application is on diskette, enter:

   ```
   bar xvf /dev/rfd0a installation_script
   ```

   If the application is on high-density tape, enter:

   ```
   tar xvf /dev/rst8 installation_script
   ```

   If the application is on low-density tape, enter:

   ```
   tar xvf /dev/rst0 installation_script
   ```

   where *installation_script* is the name of the script you supply. (You could use additional `bar` and `tar` options, depending on the options you used to put your script on tape or diskette.)
6. Enter *installation_script*.

The final step will invoke the installation script that you supply, which must perform the steps listed in the next section.

Installation Script

Installation scripts will vary between applications in content and complexity. The basic things an installation script should do are shown below. For additional information on what installation scripts and system administrators **must** do if your application includes on-screen help files or Organizer icons for your files, see page 97.

Pages 207-208 include a description of the directory hierarchy suggested for applications.

Instruct administrators not to perform these steps when anyone is adding a diskless system to the network or is creating a user account on it through New User Accounts or SNAP. This is to avoid possible corruption of the /etc/exports file, used by both procedures and modified in the steps below.

1. Load the application files into the current directory. For diskettes, use the command:

   ```
   bar xvf /dev/rfd0a directories
   ```

   For high-density tapes, use the command:

   ```
   tar xvf /dev/rst8 directories
   ```

   For low-density tapes, use the command:

   ```
   tar xvf /dev/rst0 directories
   ```

   The tar or bar options could be different, depending on the ones you used to put your files on tape or diskette.
2. If licensing issues permit, make your application available over the network by exporting it, as shown by the following steps. (Alternatively, you could have a system administrator perform these steps; see pages 97-98.)

   a. Make sure that the /export/vol directory exists:

   ```
   mkdir /export/vol > /dev/null 2>&1
   ```

   b. Create a symbolic link using the command format:

   ```
   ln -s '/bin/pwd' /export/vol/application_name
   ```

   c. Enter the following line to add an entry for the application in the /etc/exports file:

   ```
   echo "/export/vol/application_name -ro" >>
       /etc/exports
   ```

   d. Enter the following command to export the application:

   ```
   exportfs /export/vol/application_name
   ```

   If your application can only run on systems where it is installed, then your installation instructions should tell administrators to load your application into /usr/local on a system that is not a server. Your instructions should also tell administrators not to export your application.

You might also want to include an interactive portion in the script that queries an administrator about the configuration and about the architecture (if the release includes binaries for several different architectures).

NOTE *Be sure to use only relative path names in your script and application. Regardless of where your application is initially installed, administrators are likely to move it elsewhere to suit their needs. Where a complete path name is necessary, use the $application_name_ROOT environment variable described on page 207.*

Start-Up Script

The primary benefit of using a start-up script to invoke your application is that system administrators can then *register* your application, enabling anyone on a network or within a specified netgroup (see the netgroup(5) man page) to invoke your application without modifying `/etc/fstab` for each system on the network, or `.login` or `.cshrc` files. Users also need not know the path to your application, nor the architecture of the machine where your files reside.

Smaller applications consisting of one or two files do not need a start-up script, provided that the files are accessed from either `/vol/local/bin.`*arch* or `/usr/local/bin`. (*arch* can be either `sun386`, `sun2`, `sun3`, or `sun4`.) The `/vol/local` and `/usr/local` directories are all included in user's default `$PATH` values.

If you do include a start-up script, you must:

- Use the `$`*application_name*`_ROOT` environment variable in your application binaries if you refer to your files. If you program in C, use the `getenv`(3) command in the format `getenv ("`*application_name*`_ROOT")`. Page 207 provides more information about the `$`*application_name*`_ROOT` variable.
- Adhere to the application directory hierarchy described on pages 207-208.

In addition, to make your application available on a network, your installation instructions must tell system administrators to:

1. Export your application, if your script does not do this for them.
2. Create a volume for your application.
3. Copy your start-up script or `/usr/etc/start_applic` to `/vol/local.master/bin`, giving the script the name of your application. (`/vol/local.master/bin` is included in every user's `$PATH` by default.)

Chapter 6 delineates the steps a system administrator must perform, whether your application will be available on a network or only on one system, starting on page 96. The next section describes the `/usr/etc/start_applic` script that is supplied with the Sun386i system.

/usr/etc/start_applic

The `/usr/etc/start_applic` script is a short, generic shell script that an administrator can copy or link into either `/vol/local.master/bin/`*application_name* (for applications available on a network) or `/usr/local/bin/`*application_name* (for applications available only on the systems where they are installed). The script ensures that the `$PATH`, `$`*application_name*`_ROOT`, and `$LANG` variables are correctly set when the application runs.

In most cases, the `/usr/etc/start_applic` script that is provided with the Sun386i system should meet your start-up script needs. Your application must include an executable binary containing the application name in the location `$`*application_name*`_ROOT/bin.`*arch*`/`*application_name* if your application uses the `/usr/etc/start_applic` start-up script.

If your application does not have an *application_name* binary in the `bin.`*arch* subdirectory, you can create a customized start-up script. If you create a script, it should be a superset of the contents of `/usr/etc/start_applic`.

`/usr/etc/start_applic` locates and invokes the application, if possible. If it cannot start the application, it displays one or more error messages to try to explain the difficulty. The `start_applic`(8) `man` page contains additional information.

Making the Distribution

For distributions on diskette, be sure to format each diskette with `fdformat` for high-density diskettes or `fdformat -L` for low-density diskettes before copying your files to them. (The `fdformat`(8) `man` page contains more information.) Then use the `bar`(1) command to put your files on the diskette. You should also use the `bar` command if any of your application files exceeds the size of one diskette or tape, since `bar` can copy information across multiple volumes. See the `bar`(1) `man` page for details.

The `bar` command is available for loading tapes and diskettes only on Sun386i systems. Therefore, unless one of your files is too large for one cartridge tape, use the `tar`(1) command for tape distributions. The `tar`(1) `man` page provides specifics.

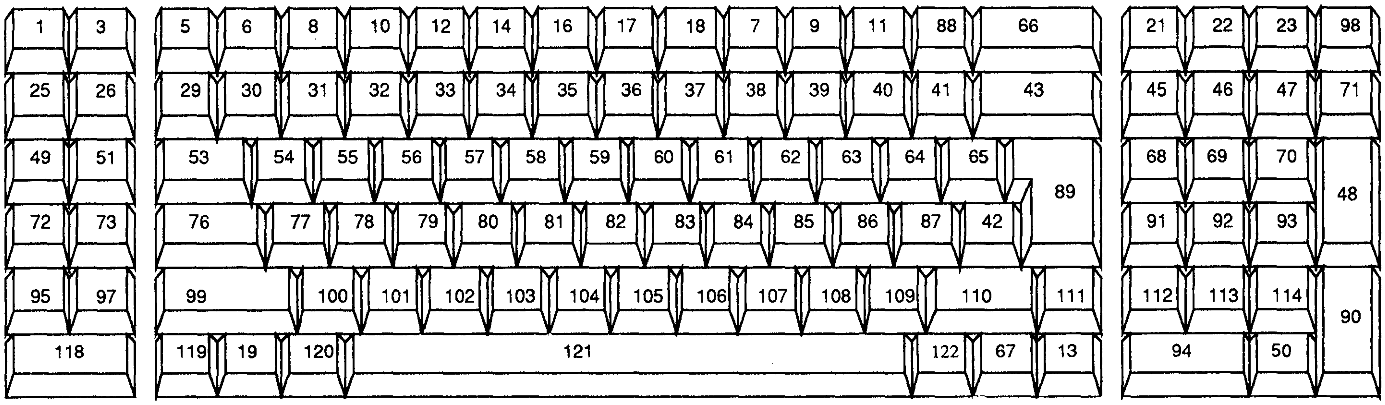

Figure 10-1 *U.S. Keystation Map*

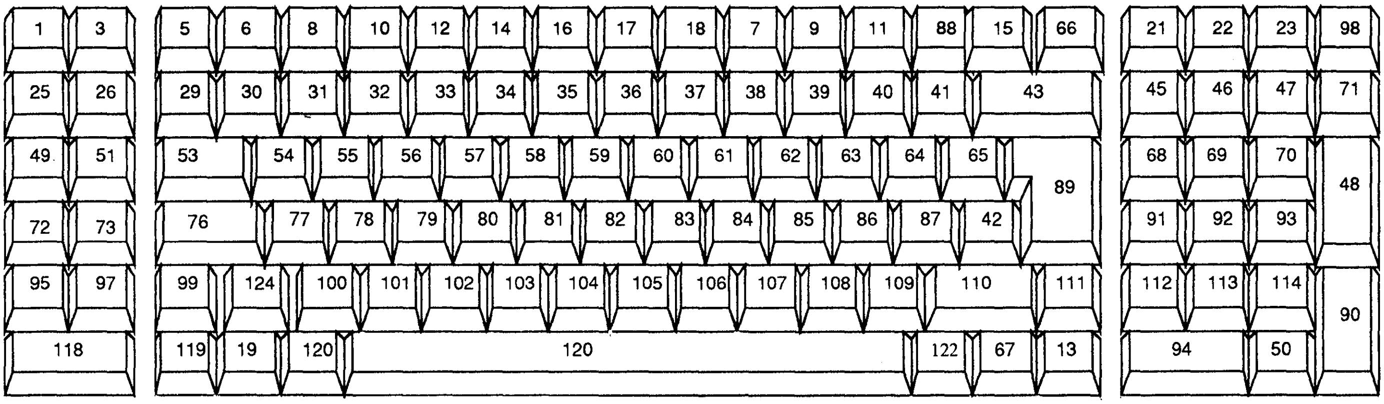

Figure 10-2 *International Keystation Map*

Table 10-2 *ISO Character Set*

	Upper Nibble															
Lower Nibble	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00			SP	0	@	P	‘	p			NBSP	°	À	Đ	à	đ
01			!	1	A	Q	a	q			¡	±	Á	Ñ	á	ñ
02			"	2	B	R	b	r			¢	2	Â	Ò	â	ò
03			#	3	C	S	c	s			£	3	Ã	Ó	ã	ó
04			$	4	D	T	d	t			¤	´	Ä	Ô	ä	ô
05			%	5	E	U	e	u			¥	µ	Å	Õ	å	õ
06			&	6	F	V	f	v			¦	¶	Æ	Ö	æ	ö
07			’	7	G	W	g	w			§	·	Ç	x	ç	÷
08			(	8	H	X	h	x			¨	¸	È	Ø	è	ø
09			)	9	I	Y	i	y			©	1	É	Ù	é	ù
0A			*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú
0B			+	;	K	[	k	{			«	»	Ë	Û	ë	û
0C			,	<	L	\	l	|			¬	1/4	Ì	Ü	ì	ü
0D			-	=	M	]	m	}			SHY	1/2	Í	Ý	í	ý
0E			.	>	N	^	n	~			®	3/4	Î	Þ	î	þ
0F			/	?	O	_	o				-	¿	Ï	ß	ï	ÿ

Floating Accent Key

You can create accent characters with the floating accent key available on international keyboards. The floating accent key works similarly to the Compose key, in that you press the floating accent key and then the character that you want to accent. You can accent both lower- and uppercase characters; to accent the latter, press the floating accent key, the Shift key, and then the character that you are accenting.

Native-Language Messages

To aid in the translation process, the Sun386i system manages external message libraries and routines for selecting and accessing system messages and on-screen help text, as described below.

System Messages

System messages are those originating in the SunOS kernel and system daemons. The `syslogd`(8) daemon intercepts error messages and uses them as keys into a message

8-Bit (byte) General Registers

`%al`	Low byte of `%ax` register
`%ah`	High byte of `%ax` register
`%cl`	Low byte of `%cx` register
`%ch`	High byte of `%cx` register
`%dl`	Low byte of `%dx` register
`%dh`	High byte of `%dx` register
`%bl`	Low byte of `%bx` register
`%bh`	High byte of `%bx` register

16-Bit (word) General Registers

`%ax`	Low 16-bits of `%eax` register
`%cx`	Low 16-bits of `%ecx` register
`%dx`	Low 16-bits of `%edx` register
`%bx`	Low 16-bits of `%ebx` register
`%sp`	Low 16-bits of the stack pointer
`%bp`	Low 16-bits of the frame pointer
`%si`	Low 16-bits of the source index register
`%di`	Low 16-bits of the destination index register

32-Bit (long) General Registers

`%eax`	32-bit general register
`%ecx`	32-bit general register
`%edx`	32-bit general register
`%ebx`	32-bit general register
`%esp`	32-bit stack pointer
`%ebp`	32-bit frame pointer
`%esi`	32-bit source index register
`%edi`	32-bit destination index register

Segment Registers

`%cs`	Code segment register; all references to the instruction space use this register
`%ds`	Data segment register, the default segment register for most references to memory operands
`%ss`	Stack segment register, the default segment register for memory operands in the stack (i.e., default segment register for `%bp`, `%sp`, `%esp`, and `%ebp`)
`%es`	General-purpose segment register; some string instructions use this extra segment as their default segment
`%fs`	General-purpose segment register
`%gs`	General-purpose segment register

Introduction to Instruction Descriptions

This section describes the SunOS 386 instruction syntax. Refer to page 178 for the differences between the SunOS 386 and the Intel 386 assemblers.

Because the assembler assumes it is generating code for a 32-bit segment, it also assumes a 32-bit address and automatically precedes word operations with a 16-bit data prefix byte.

Notational Conventions

This section uses the following notational conventions:

- The mnemonics are expressed in a regular expression-type syntax. Alternatives separated by a vertical bar (|) and enclosed within square brackets ([ ]) denote that you must choose one of them. Alternatives enclosed within curly braces ({ }) denote that you can use one or none of them. The vertical bar separates different suffixes for operators or operands. For example, `imm[8|16|32]` indicates that an 8-, 16-, or 32-bit immediate value is permitted in an instruction.
- `imm[8|16|32|48]` — an immediate value. You define immediate values using the regular expression syntax previously described (see also Expressions and Immediate Values on page 182). If there is a choice between operand sizes, the assembler will choose the smallest representation.
- `reg[8|16|32]` — a general-purpose register, where each number indicates one of the following:
  - 32: `(%eax)`, `(%ecx)`, `(%edx)`, `(%ebx)`, `(%esi)`, `(%edi)`, `(%ebp)`, `(%esp)`
  - 16: `(%ax)`, `(%cx)`, `(%dx)`, `(%bx)`, `(%si)`, `(%di)`, `(%bp)`, `(%sp)`
  - 8: `(%al)`, `(%ah)`, `(%cl)`, `(%ch)`, `(%dl)`, `(%dh)`, `(%bl)`, `(%bh)`
- `mem[8|16|32|48]` — a memory operand; the 8, 16, 32, and 48 suffixes represent byte, word, doubleword, and inter-segment memory address quantities, respectively.
- `r/m[8|16|32]` — a general purpose register or memory operand; the operand type is determined from the suffix. They are: 8 = byte, 16 = word, and 32 = doubleword. The registers for each operand size are the same as `reg[8|16|32]` above.
- `creg` — a control register; the control registers are: `%cr0`, `%cr2`, or `%cr3`.
- `dreg` — a debug register; the debug registers are: `%db0`, `%db1`, `%db2`, `%db3`, `%db6`, `%db7`.
- `sreg` — a segment register; the segment registers are: `%cs`, `%ds`, `%ss`, `%es`, `%fs`, and `%gs`.
- `treg` — a test register; the test registers are: `%tr6` and `%tr7`.
- `cc` — condition codes; the 30 condition codes are:

a	above
ae	above or equal
b	below
be	below or equal
c	carry
e	equal

C

File System Layout

C

File System Layout

This appendix describes the file system layout for the Sun386i system. Much of this structure is similar to that of the SunOS 4.0 system on other architectures. The 4.0 layout is intended to make it easier for a single server to support clients of different architectures. The Sun386i layout also provides a standard place to mount additional disks and makes it possible for users to access network files and applications without knowing the location of files or the architecture of the workstation being used, and without changing `.login` or `.cshrc` files. In addition, the Sun386i layout consolidates all of the free disk space onto one partition.

Some file system differences also exist between Sun386i systems and other Sun systems to accommodate the division of Sun386i system software. The Sun386i system groups much of system software into related files and programs called clusters, which users optionally can add to their systems at any time (Chapter 9 contains details). For compatibility between systems, and because previously existing programs expect to find files in their traditional directories, some directories now contain symbolic links to directories that contain the actual files.

If you'll be distributing software for the Sun386i system, be sure to review the last section, C.8, which describes the preferred directory structure for releasing software for this workstation.

C.1. Terms

The Sun Network File System (NFS) allows any computer with a local disk to act as a file server by exporting its file systems to clients on a network. The client computers may themselves be file servers of other file systems.

The Yellow Pages (YP) is a distributed network database. Key information about the systems and users on the network is stored in the YP database on the master server and slave servers. The master server keeps the master copy of the database, using it to update identical copies on slave servers. The YP is stored on the master server and all the slave servers to ensure the availability of the database in case a server goes down. However, the master server must be running for the updates to YP to occur. *Sun386i Advanced Administration* discusses NFS and YP in more detail.

The automounter (`automount`(8)) is a daemon that automatically and transparently mounts an NFS file system at a temporary mount point whenever a file or directory within that system is opened. The mounted file system is made available using a symbolic link to the mount point. *Sun386i Advanced Administration* and the `automount`(8) `man` page contain more information.

Several Sun386i 4.0 directories on diskful systems are *loopback mounted.* Files that appear to be in a loopback-mounted directory on diskful systems actually reside on another locally mounted directory. For example, on a diskful system files in `/tmp` really reside in `/files/tmp/localhost` because `/tmp` is a loopback mount to `/files/tmp/localhost` on diskful systems. On diskless systems, `/tmp` is in the client's root directory, which is mounted from its server. See the `lofs`(4S) `man` page for additional information about loopback mounts.

C.2. Layout Overview

The SunOS 4.0 file system layout:

- Provides easier maintenance of servers and clients
- Enables easier mixing of remote and local file systems
- Provides cleaner support of multiple architectures
- Provides a hierarchy that accounts for growth, enabling users to mount additional disks without affecting file names
- Minimizes disruption to existing programs when files are moved
- Minimizes symbolic link confusion
- Minimizes user confusion when they log in to different systems

Sun386i software is divided among three primary file systems: / (root), `/usr`, and `/files`. In addition, the `/vol` and `/home` directories help to ensure that the file system looks the same to all network users. These file systems and directories are described briefly in the next section.

System Disk

The disk that the system boots from is called the *system disk.* The standard system disk layout consists of the following special device files and partitions:

- `/dev/roota` — contains the root (/) file system
- `/dev/rootb` — contains the system's swap area
- `/dev/rootg` — contains the `/usr` file system
- `/dev/rooth` — contains the `/files` file system

The default `/etc/fstab` file contains entries to mount the corresponding file systems using the partitions listed above. This enables users to boot systems from any disk without modifying `/etc/fstab`.

The standard Sun386i file systems and directories are briefly described below. Subsequent sections detail the contents of each one.

/ (root) File System

/ (root) is the major SunOS file system, located at the top of the hierarchical file system tree. It contains machine-specific files and directories crucial for system operation, such as the kernel, a device directory listing the equipment for the configuration, and programs used for booting the multiuser version of the operating system. The contents of / is described on the following page.

`/usr` File System

`/usr` contains executable commands, system programs, and library routines, as well as some executables that were formerly under / (such as system administration programs). This is a read-only file system to ensure that its contents are identical throughout the network and to enable network-wide mounting and sharing. Because /usr is read-only, users see the same files regardless of where they log in. /usr is intentionally very full. The contents of /usr is shown on page 200.

/files File System

/files contains free space remaining after allocation of the root, swap, and usr partitions. /files contains home directories, unbundled and third-party applications, optionally loaded Application SunOS and Developer's Toolkit clusters, and root, swap, and dump directories for diskless clients. The contents of /files is delineated on page 203. Disks added to the system are mounted on /files*n*, as described in the next section, Additional Disks.

/home Directory

/home is used in conjunction with the automounter (automount(8)) to provide transparent access to a user's home directory, regardless of where the directory resides on the network. /home is described further as part of the / file system on the next page.

/export Directory

/export contains symbolic links to the local files and directories that diskful systems export to other machines on the network. Only the links, not the directories themselves, are stored in /export. Other systems on the network cannot see the actual location of exported directories. Pages 204–206 provide more information on /export.

/vol Directory

/vol is an automount(8) directory for *volumes*. A volume is a collection of related files dedicated to the same function, such as data files or all files required to run a Sun unbundled or third-party application. /vol is described more fully on page 206.

Additional Disks

Additional disks added to the Sun386i system have one partition, sd*n*c, where *n* is the SCSI unit number. If the disk in the expansion unit is the system disk, *n* is 0. If the system unit houses the system disk, *n* is 2. sd*n*c provides access to the entire disk. Each additional disk is mounted on /dev/sd*n*c as /files*x*, where *x* indicates the order of the disk added (/files1, /files2, and so on). *x* and *n* are not related.

C.3. / File System

/ contains the following files and directories:

bin
boot
dev
etc
export
files<*n*>
home
kadb
lib
lost+found
mnt
net
sbin
sys
tftpboot
tmp
tmp_mnt
usr
var
VERSION
vmunix
vol

/bin

Starting with Release 4.0, /bin is a symbolic link to usr/bin/.

/boot

boot is the program used to load the SunOS operating system. Never alter or remove this program from a disk.

/dev

/dev is the device directory, which contains all device files (also called device nodes) such as rst0 (quarter-inch tape drive), /dev/ttya (serial port), or /dev/pp0 (parallel port), for a particular configuration.

/etc

/etc contains system-specific data files and subdirectories primarily used by system administrators; includes the files created during installation and some machine-specific MS-DOS files in /etc/dos/.

/export

/export is described in Section C.6, starting on page 204.

/files<*n*>

/files<*n*> is the mount point for the /files<*n*> file system, described in Section C.5 starting on page 203.

/home

/home is an automount(8) directory that provides automatic access to home directories for all users on the network. By default, users' home directories are stored in /files<*n*>/home/*groupname*/*username* on various systems, but passwd(5) Yellow Pages (YP) entries for each user specify the home directory path as /home/*username*. The automounter (automount(8)) takes references to /home/*username* and uses the auto.home YP map to return symbolic links to the home or ~*username* directory. /home is shipped empty; /home/*username* does not exist as part of the file system on disk, but rather is created only after an automount reference is made to it.

If the auto.home entry indicates that the home directory is on a remote system, the automounter creates a temporary mount point under /tmp_mnt and uses this point to mount the remote directory onto the local system via NFS. The automounter returns a symbolic link to the mount point.

If the home directory is on the local system, the automounter returns a symbolic link to the directory. For more information on the automounter, see *Sun386i Advanced Administration*, and the auto.home(5) and automount(8) man pages.

/kadb

kadb(8) is the kernel debugger program.

/lib

/lib is a symbolic link to usr/lib.

/lost+found

/lost+found is usually empty. However, if / becomes damaged, the file system check program (fsck(8)) places links to any files that it cannot link elsewhere in this file system in /lost+found.

/mnt

/mnt is a mount point for temporarily mounting systems with the mount(8) command.

/net

/net is an automount(8) point available on networked systems that is used by the Organizer and the SNAP backup facility. Experienced users can also use /net to access directories on remote NFS diskful systems, although /home and /vol are preferred. Before using /net, make sure the system is an NFS file server (diskful system) and the path name is exported (/net/*hostname*/export/*pathname*).

/sbin

/sbin contains executable files necessary to check and mount the /usr file system and to bring up a multiuser system at boot time:

- /sbin/fsck — checks and repairs the file system
- /sbin/init — performs process control initialization
- /sbin/mount — mounts file systems
- /sbin/netconfig — configures the network
- /sbin/reboot — reboots the system
- /sbin/sh — standard command interpreter

/sys

/sys is a symbolic link to ./usr/share/sys/.

/tftpboot

/tftpboot contains the files necessary to boot diskless clients on the network.

/tmp

/tmp holds files temporarily; utilities such as cc(1) and ar(1) create temporary data files in /tmp. All files in /tmp, with the exception of subdirectories, are deleted each time the system is rebooted. On Sun386i diskful systems, this is loopback mounted onto /files/tmp/localhost. For diskless systems, /tmp is in the client's root (mounted from the server).

/tmp_mnt

/tmp_mnt is the directory that the automounter (automount(8)) uses to make mount points for temporary file systems. Do not add files to or remove files from this directory.

/usr

/usr is the mount point for the /usr file system, described on the following page.

/var

/var contains the following directories and symbolic links. On Sun386i diskful systems, /var is loopback mounted onto /files/var/localhost. On diskless systems, /var is in the client's root (mounted from the server).

- /var/adm — contains system accounting and log files
- /var/crash — is reserved for kernel core dumps of servers and standalone systems if they crash; shipped empty
- /var/dos — reserved for MS-DOS files for root (the same files in ~*username*/pc for other users); shipped empty
- /var/log — directory for log files; shipped empty
- /var/preserve — holds files saved by the vi and ed editors if the system crashes
- /var/recover — directory for crash recovery scripts (on the Sun386i system only); shipped empty
- /var/spool — contains files used for printing and other spooling functions
- /var/sysex — directory where the System Exerciser writes its temporary and log files. The System Exerciser runs under the SunOS system and verifies operation of the total system, including operating system software (on Sun386i systems only).
- /var/tmp — contains temporary files placed here by programs; unlike /tmp, files in this directory are not deleted when you reboot the system
- /var/yp — directory containing Yellow Pages databases

/VERSION

VERSION is a text file specifying the version of the root file system.

/vmunix

/vmunix is the SunOS system kernel.

/vol

/vol is an automount(8) directory described in Section C.7.

C.4. /usr File System

/usr is mounted read-only and shared. It contains architecture-specific executables and libraries, including the files and directories:

5bin
5include
5lib
adm
bin
boot
cluster
dict
dos
etc
games
hosts
include
lib
local
lost+found
man
mdec
old
pub
sccs
share
spool
stand
sys
sysex
tmp
ucb
VERSION

/usr/5bin

/usr/5bin contains symbolic links to UNIX System V binary files, stored in cluster/devel/sysV_devel/5bin/ if the sysV_devel cluster is loaded.

/usr/5include

/usr/5include is a symbolic link to cluster/devel/sysV_devel/5include/, which contains UNIX System V include files if the sysV_devel cluster is loaded.

/usr/5lib

/usr/5lib contains symbolic links to UNIX System V libraries, stored in cluster/devel/sysV_devel/5lib/ if the sysV_devel cluster is loaded.

/usr/adm

/usr/adm is a symbolic link to ../var/adm.

/usr/bin

/usr/bin contains basic SunOS operating system commands, including those formerly located in /bin, such as ls(1V), cat(1V), and mkdir(1).

/usr/boot

/usr/boot is a directory that contains symbolic links to files in ../../sbin.

/usr/cluster

/usr/cluster is loopback mounted onto /files/cluster on Sun386i diskful systems. (On diskless Sun386i systems, /usr/cluster is a mount point.) /usr/cluster contains all of the optional clusters added to Sun386i systems, in the directories:

- /usr/cluster/appl — mount or load points for Application SunOS clusters added to the system
- /usr/cluster/devel — mount or load points for SunOS Developer's Toolkit clusters added to the system

NOTE *Do not use* /usr/cluster/<appl devel> *path names in programs, because the location of clusters could change in subsequent releases. Instead, use the link names that resolve to* /usr/cluster/<appl devel>, *such as* /usr/dict *(on the next page) and others described later in this appendix.*

/usr/dict

/usr/dict is a link to cluster/appl/spellcheck/dict/, which is a database that contains English language spelling lists used by the spell(1) spelling checker, if the optional cluster spellcheck is loaded; shipped empty.

/usr/dos

/usr/dos contains MS-DOS commands and files.

/usr/etc

/usr/etc contains the commands and files used for system administration and maintenance.

/usr/games

/usr/games is a symbolic link to cluster/appl/games/games/, which exists only if the optional games cluster is loaded.

/usr/hosts

/usr/hosts is a symbolic link to old/hosts.

/usr/include

/usr/include contains links to all of the standard include (header) files used in C programs; these files, traditionally named with a .h extension, contain definitions of useful constants and macros. Include files are either in the sys subdirectory or in ../cluster/devel/base_devel/include if the base_devel cluster is loaded. /usr/include contains the subdirectories shown below:

- /usr/include/images — contains SunView icons
- /usr/include/make — contains a default make file

/usr/lib

/usr/lib and its subdirectories contain more than 100 files and links to files used by SunOS utilities and files formerly located in /lib (which is now a symbolic link to /usr/lib).

/usr/lib includes the subdirectory /usr/lib/load, which contains the cluster size and file database used by the load(1), loadc(1), unload(1), unloadc(1), and cluster(1) commands; also contains the filesizes file which lists the names and sizes of files within each cluster (on the Sun386i system only).

/usr/local

/usr/local is loopback mounted to /export/local/*arch* on diskful systems, where *arch* can be either sun2, sun3, sun4, or sun386. On diskless systems, /usr/local is an NFS mount to /export/local/*arch* on the boot server. /usr/local contains binaries that only the local system can use and sharable data files in the subdirectories:

- /usr/local/bin — contains binaries and shell scripts for the local architecture; included in every user's $PATH by default
- /usr/local/lib — contains libraries for the local architecture
- /usr/local/*application_name* — reserved for installation of third-party software for access only by the local system

/usr/lost+found

/usr/lost+found is usually empty. However, if /usr becomes damaged, the file system check program (fsck(8)) places links to any files that it cannot link elsewhere in this file system in /usr/lost+found. If this happens, you'll probably have to reload the core system, as described in *Sun386i System Setup and Maintenance*, as well as any optional clusters that you had added.

/usr/man

/usr/man is a symbolic link to share/man/, described later in this section.

/usr/mdec

/usr/mdec is a symbolic link to cluster/appl/advanced_admin/mdec/, which exists only if the optional advanced_admin cluster (containing boot blocks and the install boot program for the Sun386i system) is loaded.

/usr/old

/usr/old is a symbolic link to cluster/appl/old_commands/old, which exists only if the optional old cluster is loaded. The old cluster contains commands that have been phased out but retained in this release for compatibility.

/usr/pub

/usr/pub is a symbolic link to cluster/appl/doc_prep/pub/, which contains data files used in formatting and printing if the optional doc_prep cluster is loaded.

/usr/sccs

/usr/sccs is a symbolic link to cluster/devel/sccs/sccs, which exists only if the optional sccs cluster is loaded.

/usr/share

/usr/share contains architecture-independent sharable files, shown below:

- /usr/share/lib — contains tab set, termcap, time zone, and New User Accounts login screens; also contains the terminfo file, which is a link to terminal information in ../../cluster/appl/sysV_commands/share/lib/terminfo/ if the optional sysV_commands cluster is loaded
- /usr/share/man — symbolic link to ../cluster/appl/man_pages/share/man/, which exists only if the optional man_pages cluster is loaded
- /usr/share/messages — contains messages files for utilities shipped on the Sun386i disk
- /usr/share/src/sun/suntool — symbolic link to source code for some SunView programs in ../cluster/devel/sunview_devel/share/src/sun/suntool, which exists only if the optional sunview_devel cluster is loaded
- /usr/share/sys — symbolic link to ../cluster/devel/config/share/sys/, which contains the files needed to build custom kernels, and exists only if the optional config cluster is loaded

/usr/spool

/usr/spool is a symbolic link to ../var/spool/.

/usr/stand

/usr/stand is a symbolic link to ../stand/.

/usr/sys

/usr/sys is a symbolic link to share/sys/.

/usr/sysex

/usr/sysex contains System Exerciser executable files.

/usr/tmp

/usr/tmp is a symbolic link to ../var/tmp/.

/usr/ucb

/usr/ucb contains commands that originated with the Berkeley UNIX system (ucb is an acronym for University of California at Berkeley).

/usr/VERSION

/usr/VERSION is a text file specifying the version of this /usr file system.

C.5. /files<*n*> File System

/files<*n*> could contain the directories shown in this section (not all of these will exist on disks added to the expansion unit). /files is the name of this file system on the system disk. The name of this file system on the first additional disk added to the system is /files1, for the second disk added /files2, and so on.

cluster
dump
exec
help
home
local
lost+found
root
swap
tmp
var
vol
vol.local

/files/cluster

/files/cluster/<*arch*>.<*OS release*> contains optional clusters added to the Sun386i disk, in the directories shown below. <*arch*> can be either sun2, sun3, sun4, or sun386 and <*OS release*> has the format SunOS4.0.0, SunOS4.0.1, SunOS4.1.0, and so on.

- /files/cluster/<*arch*>.<*OS release*>/appl/*cluster_name* — contains Application SunOS clusters added to the system
- /files/cluster/<*arch*>.<*OS release*>/devel/*cluster_name* — contains SunOS Developer's Toolkit clusters added to the system

/files<*n*>/dump

/files<*n*>/dump is reserved for kernel core dumps of diskless clients if they crash; shipped empty and not used by default.

/files<*n*>/exec

/files<*n*>/exec contains the native executables (typically symbolic links to /usr file systems) of each Sun workstation architecture and each SunOS release on the server's disk.

/files/help

/files/help contains links to the on-screen help files supplied with the Sun386i system in /usr/lib/help. (These files are accessed through /vol/help.) If you delete anything in this directory, you will not be able to access any of the Sun386i on-screen help files. The only file in this directory that is not a link is /files/help/language/USA-English/Top_Level, which you can edit to add help handbooks that are not supplied with the Sun386i system.

/files<*n*>/home

/files<*n*>/home is reserved for the home directories of each user on a server (/files<*n*>/home/*groupname*/*username*).

/files<*n*>/local

/files<*n*>/local contains the subdirectories:

- /files/local/*arch* — contains the local system's /usr/local (a loopback mount from /export/local/*arch*) and is shared with any diskless client of the same architecture
- /files/local/*other_arch* — contains the /usr/local used by diskless clients that have an architecture different from the server

/files<*n*>/lost+found

/files<*n*>/lost+found is usually empty. However, if /files<*n*> becomes damaged, the file system check program (fsck(8)) places links to any files that it cannot link elsewhere in this file system in /files<*n*>/lost+found.

/files<*n*>/root

/files<*n*>/root is reserved for root directories for all diskless clients of a server (/files<*n*>/root/*hostname*).

/files<*n*>/swap

/files<*n*>/swap is reserved for individual swap areas for all diskless clients of a server (/files<*n*>/swap/*hostname*).

/files<*n*>/tmp

/files<*n*>/tmp contains the localhost directory. /tmp is loopback mounted to /files<*n*>/tmp/localhost on diskful systems; refer to the description for /tmp earlier in this appendix. To move the tmp directory to another disk (from /files to /files1, for example), edit the /etc/fstab file. The fstab(5) man page provides details.

/files<*n*>/var

/files<*n*>/var contains the localhost subdirectory. /var is loopback mounted to /files<*n*>/var/localhost on diskful systems; refer to the description for /var earlier in this appendix. To move the var directory to another disk (from /files to /files1, for example), edit the /etc/fstab file. The fstab(5) man page provides details.

/files<*n*>/vol

/files<*n*>/vol is reserved for third-party applications that will be available over the network.

/files<*n*>/vol.local

/files<*n*>/vol.local is on the server that contains /vol/local (the Yellow Pages master by default). It is reserved for shell scripts and architecture-dependent executable files.

- bin — designed to contain application start-up scripts, or links that point to /usr/etc/start_applic for each application that has its own volume
- bin.*arch* — contains architecture-dependent executable files, where *arch* can be either sun2, sun3, sun4, or sun386

C.6. /export Directory

/export contains symbolic links to local directories that diskful systems export to other machines on the network. Many of these links already exist on the system, and others are created when performing certain administrative functions via SNAP or New User Accounts. In both cases, the system edits the /etc/exports file to include information on exported directories.

Page 97 and the exports(5) and exportfs(8) man pages contain more information about exporting directories.

The leaf nodes (final components of path names) that the system typically exports are shown below.

/export/cluster

/export/cluster contains an <*arch*>.<*OS release*> directory that contains a link to ../../files/cluster/<*arch*>.<*OS release*>/, which contains clusters added to Sun386i systems. <*arch*> can be either sun2, sun3, sun4, or sun386 and <*OS release*> has the format SunOS4.0.0, SunOS4.0.1, SunOS4.1.0, and so on.

/export/dump

/export/dump on a server is reserved for manually setting up dump directories for diskless clients. By default, diskless clients dump to the swap partition.

/export/exec

/export/exec contains a symbolic link for each software architecture of the SunOS system loaded on this workstation. Each link points to the particular release's location:

> /export/exec/*arch* is a symbolic link to /export/exec/*<arch>.<OS release>*, which is a symbolic link to /usr/ if the server is running the same *<arch>.<OS release>* as the /exec being exported; if this is not the case, then /export/exec/*<arch>.<OS release>* is a symbolic link to /files*<n>*/exec/*<arch>.<OS release>*

where *<arch>* can be either sun2, sun3, sun4, or sun386 and *<OS release>* has the format SunOS4.0.0, SunOS4.0.1, SunOS4.1.0, and so on.

Diskless clients mount /usr from their boot server, using the entry that Automatic System Installation places in the bootparams Yellow Pages map:

> usr = *server*: /export/exec/*<arch>.<OS release>*

/export/help

/export/help is a symbolic link to ../files/help/.

/export/home

If this system has a disk with users' home directories, /export/home contains symbolic links to each user's home directory:

> /export/home/*groupname*/*username* is a symbolic link to /files*<n>*/home/*groupname*/*username*

/export/local

/export/local/*arch* points to /usr/local for systems of the architecture supported. *arch* can be either sun2, sun3, sun4, or sun386.

/export/root

If this system is a boot server for diskless clients, /export/root contains symbolic links to each client system's root directory:

> /export/root/*client_systemname* is a symbolic link to /files*<n>*/root/*client_systemname*

Diskless clients mount / from their boot server, using the entry that Automatic System Installation places in the bootparams Yellow Pages map:

> root = *server*: /export/root/*client_systemname*

/export/swap

If this system is a boot server for diskless clients, /export/swap contains symbolic links to each client system's swap file:

> /export/swap/*client_systemname* is a symbolic link to /files*<n>*/swap/*client_systemname*

/export/tmp

/export/tmp contains the localhost file, which is a symbolic link to ../../files/tmp/localhost.

/export/var

/export/var contains the localhost file, which is a symbolic link to ../../files/var/localhost.

/export/vol

/export/vol is a standard place from which to export volumes. By default /export/vol on the master server contains two subdirectories:

- /export/vol/help — a symbolic link to /files/help
- /export/vol/local — a symbolic link to /files/vol.local

On other systems /export/vol is designed to contain exported volumes.

C.7. /vol Directory

A volume is a collection of related files dedicated to the same function, such as all files required to run a third-party or unbundled application, or all data associated with a particular project. Volumes are attached to the Sun386i file system by the automounter (automount(8)). After installation on a Sun386i network, the Sun386i system minimally has these four volumes:

- /vol/help — the default help directory (read only) for on-screen help files
- /vol/help.master — a writable copy of /vol/help
- /vol/local — a network-wide, multiple-architecture directory (read only) for accessing programs; contains a /bin subdirectory for shell scripts and a bin.*arch* subdirectory for architecture-specific binaries (*arch* can be either sun2, sun3, sun4, or sun386)
- /vol/local.master — a writable copy of /vol/local

To create a volume for an application, an administrator must export the application as /export/vol/*application_name* and include an entry in the /etc/auto.vol file on the YP master with the format:

application_name [*-mount options*] *system:*/export/vol/*application_name*

where *-mount options* might include either -ro or -ro, secure, since volumes that you create are mounted with read-write access by default. (/vol/help and /vol/local are read-only.) Then an administrator must rebuild the auto.vol YP map by issuing the commands:

```
cd /var/yp; make
```

The automounter (automount(8)) takes references to /vol/*application_name* and uses the entry corresponding to *application_name* in the auto.vol Yellow Pages map to mount the volume on a temporary mount point under /tmp_mnt. /vol subdirectories do not exist as part of the file system on disk, but rather are created only after automount(8) references to them are made. (*Sun386i Advanced Administration* and the automount(8) man page describe the automounter in more detail.)

If the auto.vol entry indicates that the volume is on a remote system, the automounter creates a temporary mount point under /tmp_mnt and uses this point to mount the remote volume onto the local system via NFS. The automounter returns a symbolic link to the mount point.

If the volume is on the local system, the automounter returns a symbolic link to the volume.

If the application has been exported, is accessible from a volume, and has its start-up script in `/vol/local.master/bin` (which *registers* it), any user on the network can execute the application from a workstation of any supported architecture simply by entering *application_name*; users' `.login` and `.cshrc` files need no modification, and the application's full path name is not needed. Section 9.5 and *Sun386i Advanced Administration* provide details.

C.8. Application Directory Structure

This section describes the preferred subdirectories that you should use to release your applications. You should use this structure because many administrators will want to make your application available on a network (provided that licensing issues permit such access), which could consist of systems with different architectures.

In addition, system administrators will install and potentially move your application to wherever is best for them. That is why your application files should use the environment variable `$`*application_name*`_ROOT` in path names instead of using absolute paths. The `$`*application_name*`_ROOT` variable provides one place for all of your application files, and ensures that your application will work, regardless of its location. If your application uses the `$`*application_name*`_ROOT` variable as suggested, then you must also include a start-up script with your application to correctly set the environment variable. This can be either the one supplied with the Sun386i system, `/usr/etc/start_applic`, or one that you provide. Chapter 9 contains more information on start-up scripts.

To make applications available on a network, administrators should install your application in `/files<`*n*`>/vol/`*application_name*, and then export and create a volume for it, as described starting on pages 96-98. Using this scheme enables your application to work even when administrators (or users with workstations) move it to a different location, and eliminates the need for users to alter their `.login` and `.cshrc` files to use your application. Administrators should install your application under the `/usr/local/`*application_name* directory for applications that can only be accessed from one system (generally for licensing reasons).

You should include the following relevant subdirectories of `$`*application_name*`_ROOT` for your application. Note that all subdirectories not explicitly tagged with a processor architecture are shared among all processor architectures.

bin.*<arch>*

This directory is for architecture-dependent binary files, where *arch* can be either `sun2`, `sun3`, `sun4`, or `sun386`. Your application must include an executable binary containing the application name in the format `$`*application_name*`_ROOT/bin.`*arch*`/`*application_name* if your application uses the `/usr/etc/start_applic` start-up script.

bin

This directory is for shell scripts or other sharable, interpretted programs for the application.

share

Place architecture-independent files in share, using the subdirectories:

- data — for miscellaneous data files
- fonts — for font files
- icons — for SunView .icon files
- images — for SunView .image files

language

Place a subdirectory in language for each language supported:

- USA-English
- English
- French
- French_Swiss
- German
- German_Swiss
- Italian
- Swedish
- Spanish

Four standard subdirectories are available for each language directory:

- doc — containing on-line documentation for the application in the particular language
- help — containing Spot Help and handbook files for the application in the particular language
- man — containing man pages for the application in the particular language
- messages — containing message files for the application in the particular language

Index