WorldView™ Europe Reference Manual

Document Number 007-2151-003

CONTRIBUTORS

Revised by Gary Sloane
Production by Chris Everett
Engineering contributions by Regina Bustamante, Erik Fortune, Yukitomi Fujinaga, and Michael Kung

Silicon Graphics, Inc.
Mountain View, California

WorldView™ Europe Reference Manual
Document Number 007-2151-003

Contents

Chapter 1

Introduction

WorldView™ Europe, which is part of the WorldView family of national language products, helps you to use IRIX in your language of choice. WorldView allows you to use a variety of European data formats, and it provides full support for IRIX in French and German. (Future releases of WorldView Europe may provide full IRIX support for other European languages as well.)

Different languages and different places have different standard ways of formatting such data as times, dates, and numerical amounts. A database containing such local data formats for a specific language and location is called a *locale*. If an application program is written in a way that is independent of any locale—that is, if it is written to use any user-specified locale to provide input and output in the appropriate language and format—the program is said to be *internationalized*. If a database for a locale has been created to allow an internationalized program to use that locale, the program is said to be *localized* to that locale.

Since WorldView Europe provides localization of IRIX for French and German, all you have to do to use those languages is make the appropriate choice on a graphical control panel, or indicate the desired locale in a file in your home directory. For more details on how to do this, see “Choosing a Locale” in Chapter 2.

WorldView Europe also enables you to:

- Run applications localized for European languages
- Create, edit, and print documents using European languages
- Have your environment behave according to cultural expectations regarding date, time, currency, numbers, and other kinds of formatted data

IRIX 6.2 provides locale databases for the following languages:

- Czech
- Danish
- Dutch (Belgium, Netherlands)
- English (Australia, Canada, UK, and US)
- Finnish
- French (Belgium, Canada, France, Switzerland)
- German (Austria, Germany, Switzerland)
- Greek
- Icelandic
- Italian (Italy, Switzerland)
- Norwegian
- Polish
- Portuguese (Brazil, Portugal)
- Russian (Koi-8, ISO8859-5)
- Slovak
- Spanish (Argentina, Mexico, Spain)
- Swedish
- Turkish

Some of these languages have more than one associated locale because they are used in more than one country—a locale specifies both a language and a country.

WorldView also provides fonts for use with all of the above languages, with limited support for Czech, Greek, Polish, Russian, and Turkish.

What is in WorldView Europe?

The WorldView Europe 6.2 package contains:

- German and French translations for the messages produced by most desktop applications.
- Locale information needed to run X applications in Russian locale using the Koi-8 character set.

WorldView files are stored in various subdirectories of the */usr/lib* directory. For information about installing WorldView, see the pamphlet that came with your WorldView CD or consult the *WorldView Europe 6.2 Release Notes*.

What This Guide Contains

This guide explains how to set up a computer environment and printer using the locales provided in WorldView. Here are brief descriptions of the chapters of this guide:

- Chapter 1, “Introduction,” describes the contents of this manual and provides some hints on how to use it effectively.
- Chapter 2, “Getting Started,” tells you how to switch quickly from using English with IRIX to using French or German. This chapter also presents examples of what you can do with WorldView directly from your keyboard, including setting locales, locating message catalogs, and modifying appdefs.
- Chapter 3, “Working With Applications,” covers general concerns and specific behaviors of applications, system utilities, and shell-based applications.

Intended Audience

This guide assumes no prior knowledge about how to change languages in IRIX. It does assume that you know what shell you’re using and that you know how to do basic IRIX activities, such as logging in and out and using a text editor, such as *vi*, *emacs*, or *jot*. If you don’t know those things, see the book *IRIX Essentials*.

This guide is intended for end-users, but may also be used by system administrators or developers. A system administrator might want to change parts of message catalogs, or simply know how to assist WorldView users. A developer might use this guide to help set up an internationalized development environment.

Further Reading and Advanced Information

The following manuals provide supplementary information and are sometimes referenced in this manual, but most of them are fairly technical, and not necessary for a casual WorldView user to read.

- *IRIX System Programming Guide.* For programmers using the IRIX Development Option. See Chapter 10, “Internationalization” and Chapter 11, “Localization.” Silicon Graphics part number 008-1794-020.
- Nye, Adrian: *Xlib Programming Manual for Version 11 of the X Window System, Volume One*, third edition (covers Release 4 and Release 5). Sebastopol, California: O’Reilly & Associates, Inc. ISBN 1-56592-002-3. See Chapter 10, “Internationalization,” and Chapter 11, “Internationalized Text Input.”
- *OSF/Motif Programmer’s Reference*, Release 1.2. Englewood Cliffs, New Jersey: Prentice Hall, Inc. ISBN 0-13-6431151
- Quercia, Valerie and O’Reilly, Tim: *X Windows System User’s Guide for X11 Release 5*. Sebastopol, California: O’Reilly & Associates, Inc. ISBN 1-56592-014-7. For information on application defaults, see Chapter 10, “Setting Resources.”
- X/Open Company, Ltd.: *Internationalization Guide*, X/Open. Englewood Cliffs, New Jersey: Prentice Hall, Inc. ISBN 1-872630-20-0. Contains information on internationalization issues, locales, message catalogs and other topics.

Typographic Conventions

IRIX manual pages are referred to by name and section number, in this format:

name(sect)

where *name* is the name of a command, system call, or library routine, and *sect* is the manual-page section number where the entry resides. For example:

rpc(3R)

refers to the *rpc* manual page in section 3 of the IRIX manual pages (which is divided up into subsections, such as 3N and 3R). To look at that manual page, enter the command

```
IRIS% man 3 rpc
```

In syntax descriptions and examples, you will see these type conventions:

Bold is used for chapter, section, and table headings.

Italic indicates IRIX filenames, command names, and arguments to be replaced with a value. Also used for options to commands and to indicate a new term used for the first time, as well as for book titles.

`Fixed-width`
indicates system output.

`Fixed-width Bold`
indicates user input.

Chapter 2

Getting Started

This chapter discusses setting up your environment using WorldView Europe. In this chapter, you'll learn about:

- Choosing and quickly switching to a different locale
- Creating composite characters using the Compose key
- Locale names and categories
- Message Catalogs
- Modifying defaults for X-based applications
- Latin-1 and other non-ASCII encodings
- Printing non-ASCII characters using PostScript®
- Supplementary Installation Instructions

Choosing a Locale

Many users of WorldView may never want to use it for anything other than switching from English to another language, such as German or French. If you just want system messages to display in German, and you don't want to bother with anything more complicated, the procedure is simple. There are two ways to set your locale:

- From the Toolchest, select Desktop—>Customize—>Language
- From the command line, set the environment variable LANG. Setting LANG identifies the desired language and territory; see the *Locale* column in Table 2-1 for LANG settings that work with WorldView.

These methods are covered in more detail in the two following sections.

Note: Changing locales translates data formats only. Although system messages, help screens, and some IRIS InSight™ books are provided for French and German as well as English, filenames, program names, and the contents of files are not translated into the language of the new locale. Man pages remain in English.

Using ipanel to Change Locale or Keyboard

The Silicon Graphics Desktop System provides a set of system administration tools with graphical user interfaces. One such tool is *ipanel*, the language customization tool, which you invoke by choosing Desktop—>Customize—>Language from the Toolchest. The *ipanel* window is shown in Figure 2-1.

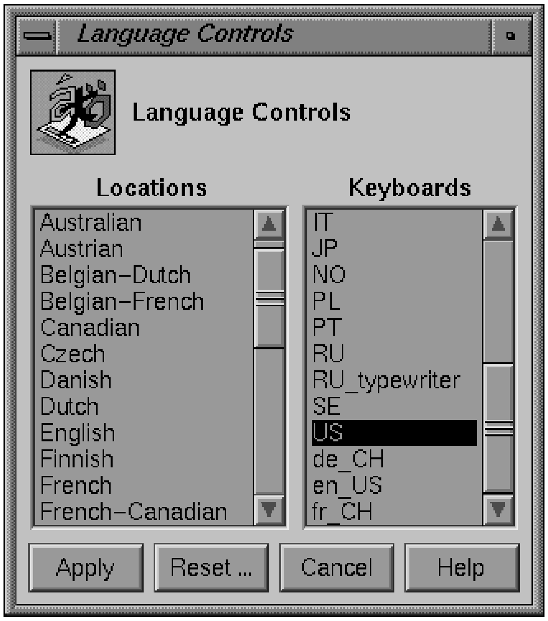

Figure 2-1 The *ipanel* window

When the *ipanel* window (labeled Language Controls) comes up, choose the locale you want from the Locations list on the left. If you do not have a US keyboard, choose the appropriate keyboard from the list on the right.

Standard Keyboard

The standard keyboard shipped with most SGI systems can enter only ASCII characters by default. Configured as shipped, it does not allow you to enter or use non-ASCII characters needed for many European languages.

Many locales make it possible to use non-ASCII characters with most system tools. The INTL keyboard layout is exactly like the default layout for the (101-key) US keyboard, except that the right Alt key (the key immediately to the right of the space bar) behaves as a Compose key, which makes it possible to enter many non-ASCII characters. See the *compose*(5) and *composetable*(5) man pages for more information on using the Compose key.

Special Keyboards

Many systems include a keyboard specially designed for a particular location, such as Switzerland, or a specific language, such as German. If you have a specialized, non-US keyboard, you should choose the appropriate keyboard from the Language Controls panel; this will make your keyboard generate the characters that appear on each key. For instance, the right Alt key on many international keyboards selects alternate characters for some of the keyboard keys. Pressing then releasing Shift and the right Alt key activates the **compose** function, as described in *compose*(5) man page. For multiple special characters on a single key, see “Inaccessible Alt Gr Characters” on page 27.

If you are not certain which keyboard you have, first look at the Enter key. If your Enter key is rectangular, you have a US keyboard and should use the INTL keyboard layout. If your Enter key is non-rectangular, you probably have a special keyboard for your language or location.

If your keyboard has a non-rectangular enter key but selecting the appropriate layout does not make it work properly, it is possible that you have a national keyboard for some country other than your own. You can try each of the keyboard layouts listed in the Language Controls panel until you find the one that matches your layout, or contact your sales and support office for assistance.

Unlike many graphical tools, *ipanel* implements changes as soon as you make them; that is, as soon as you choose a new locale, you are shifted into that locale without having to confirm your choice. If you make a mistake while choosing a locale, simply choose again before closing the *ipanel* window.

Once you've made your locale and keyboard selections, click the Cancel button to exit *ipanel*.

Choosing a new locale does not affect any currently-running programs, including any shell windows you may have open already. For instance, if you start in the default (*C*) locale, then choose the French locale, all open windows will still run in the *C* locale. Only new windows launched from the Toolchest will be in the French locale.

Note: To see your entire desktop environment and all your windows in the new locale, you must log out and log back in again.

Using the Command Line to Change Locale

You can also make changes from the IRIX command line. There are three steps to this process:

1. Put the name of the locale you want to use into the *.lang* file in your home directory. That file tells the window environment what locale to use. For example, to switch future console login sessions to German:

   ```
   IRIS% cd
   IRIS% rm .lang
   IRIS% cat >.lang
   de
   <ctrl-D>
   IRIS%
   ```

2. Set up the LANG environment variable so that it sets itself appropriately every time you log in. The idea is to leave the LANG variable alone if it's already been set, and to make sure the *.lang* file exists before using it to set LANG.

   1. In *csh*, you can do this by editing your *.login* file to include these lines:

      ```
      if (! $?LANG) then
          if ( -r $HOME/.lang ) then
              setenv LANG ‘cat $HOME/.lang‘
          endif
      endif
      ```

   2. If you use Bourne shell, edit your *.profile* file to include the following lines, which do exactly the same thing as the above C shell lines:

      ```
      if  [ x$LANG = x ]
      ```

```
then
    if [ -r $HOME/.lang ]
    then
        LANG=‘cat $HOME/.lang‘ ; export LANG
    fi
fi
```

In either case, the given line executes every time you log in, even if you log in from a remote terminal. It sets your locale to whatever the *.lang* file says your locale should be.

3. Log out and log back in again to switch your whole environment to the new locale.

Those two letters in your *.lang* file are all you really need to run IRIX in German. (Replace the *de* with *fr* to run IRIX in French.) However, if you want to learn more about locales, including how to change various aspects of a locale, read the rest of this document.

Switching Locales Temporarily

If you want to switch locales for only one session in only one shell window, rather than for all future sessions, you can do so using the command line, without editing any files at all. For example, in the C shell,

```
IRIS% setenv LANG fr
```

sets the locale of whatever window you enter it in to French, and that window remains in French until you explicitly change locales again. Any other windows you have, however, and all future sessions, remain in English (or whatever your locale was previously set to).

Character Composition

There are many characters used in non-English languages which are not part of the ASCII character set. Some of these characters can be typed directly from some keyboards, but if you’re using a US keyboard, you’ll have to type those characters by *composing* them.

Composing a character consists of typing several key strokes which, in combination, represent a composite character (such as *á* or *è*). To compose a character, you usually need to press a special key that indicates that the next few keystrokes should be interpreted as a composite character; then you enter the appropriate key sequence for the character you want to type. For instance, to produce the character *ñ*, you might press the Compose key, then the tilde (~), then *n*. Not every keyboard has a Compose key, however. If yours doesn't and you need one, you can use the *xmodmap*(1) command to remap a key you're not using to be the Compose key. For instance, to remap the Alt_R function key to be the Compose key, enter this command:

```
IRIS% xmodmap -e "keysym Alt_R = Multi_key"
```

You can, of course, replace the *Alt_R* with the name of any other key. For more detailed information, see the *xmodmap*(1) man page and "Inaccessible Alt Gr Characters" on page 27.

More Details on Choosing a Locale

Silicon Graphics WorldView lets you work on your computer in the language of your choice—given the supported locales and character sets—and have the time, date, money, numbers, and fonts appear as you would expect for that language. You select the language by setting an environment variable or choosing from a control panel, as shown in "Choosing a Locale" on page 7. To do anything more complicated than that, however, you need to know which locale or parts of a locale you want to use.

Overview of Locale Names

Locale naming is based on the POSIX.1 locale naming convention: *language[_territory.[.encoding]]*. The language name is a two-letter word taken from the *ISO-639 Language Code Standard*, and the territory name is a similar two-letter words from *ISO-3166 Country Code Standard*. Thus, the name of the locale has either one or two parts. The first part represents the language, such as *fr*, the name for the French language locale as used in France. However, French used in Canada is different, so the name of that locale is *fr_CA*, where the second part of the locale name represents Canada, the territory associated with the locale.

Table 2-1 shows the locale names and territories available in IRIX 6.2 for each European language, listed in alphabetical order by language name. Not listed is the default locale, called *C*, which uses American English but also assumes that all characters are ASCII; the *C* locale doesn't know what to do with non-ASCII characters. If you want to use non-ASCII characters with American English, use the *en_US* locale.

Table 2-1 European Language Locales Supported by IRIX 6.2

Language	Territory	Locale
Czech	Czech Republic	cs
Danish	Denmark	da
Dutch	Netherlands	nl
Dutch	Belgium	nl_BE
English	Australia	en_AU
English	United Kingdom	en
English	Canada	en_CA
English	United States	en_US
Finnish	Finland	fi
French	France	fr
French	Belgium	fr_BE
French	Canada	fr_CA
French	Switzerland	fr_CH
German	Germany	de
German	Austria	de_AT
German	Switzerland	de_CH
Greek	Greece	el
Icelandic	Iceland	is
Italian	Italy	it
Italian	Switzerland	it_CH

Table 2-1 (continued) European Language Locales Supported by IRIX 6.2

Language	Territory	Locale
Norwegian	Norway	no
Polish	Poland	pl
Portuguese	Portugal	pt
Portuguese	Brazil	pt_BR
Russian	Russia	ru
Slovak	Slovakia	sk
Spanish	Spain	es
Spanish	Mexico	es_MX
Spanish	Argentina	es_AR
Swedish	Sweden	sv
Turkish	Turkey	tr

Some languages have multiple locales, corresponding to those languages' use in multiple territories. Conversely, some territories have more than one locale, corresponding to multiple languages.

Also, people who localize software, *localizers*, may create new locales at any time, so locales not on the above list may be available on your system.

Locale Categories

The various aspects of a locale, called *categories*, can be set independently of one another. For example, if you're using a French system and you have a list of Turkish companies that you want to alphabetize according to Turkish sorting rules, you need to set the French locale, then set the LC_COLLATE category to Turkish. This is called setting a category explicitly.

Each category corresponds to an environment variable with the same name. The available categories are described in Table 2-2.

Table 2-2 Locale Categories

Category	Description
LC_ALL	All categories below
LC_COLLATE	Collating sequence
LC_CTYPE	Character-type classification and transliteration
LC_MESSAGES	Yes / No response string and the text messages used by the applications
LC_MONETARY	Currency representation
LC_NUMERIC	Numbers and other nonmonetary numeric formatting information
LC_TIME	Dates and times

Here are slightly fuller descriptions of the categories:

- The collating sequence for a locale describes the correct way to order items when sorting them alphabetically.
- Character classification and conversion describes the rules to use in categorizing characters or converting them from one type to another. For instance, functions such as changing to uppercase or lowercase characters work differently from one language to another.
- Messages are the output of programs. For more information, see “Message Catalogs.”
- Currency formatting varies according to local conventions. For example, in Britain thirty-five and a half pounds would be written *£35.5*; in France thirty-five and a half francs would be written *35,50Fr.*
- Nonmonetary numbers are also formatted differently in different languages. For example, one thousand in English can be represented as *1,000.00*, whereas in French it can be represented as *1.000,00*.
- Dates and times can be written in many different ways. January 31, 1996, for instance, can be written *1/31/96* or *31/1/96* or several other ways.

Any of the categories of the locale that have been explicitly set previously remain unaffected when you set the LANG variable. The following section explains how to change specific categories.

Changing a Locale Category

The general locale-choosing mechanism provided by the LANG environment variable does not cover the case where, for example, you want to see an application's messages in one language while using the collation order of another language. Such a situation is supported by allowing you to set additional environment variables, one for each category that you want to differ from the LANG setting. You will rarely, if ever, need to change specific locale categories without changing the entire locale.

If any of the category environment variables are not defined in the current environment, those categories default to the value of LANG.

Note: If you've set a category explicitly but you now want its value to be controlled by the value of LANG again, use the *unsetenv*(1) command.

Always be sure to use compatible category settings. For example, setting LC_COLLATE to *ru* when LC_ALL is *en* leads to unpredictable results.

Note: Some programs that were developed before the X/Open and ANSI C library interfaces became available may require using the LANGUAGE environment variable instead of LANG. You can set LANGUAGE, if necessary, using exactly the same syntax you used to set LANG.

For information about the library interfaces that support localization, see the chapters "Internationalization" and "Localization" in the *IRIX System Programming Guide*.

An Example of Using Locales

The default locale is *C*. The locale is set when you log in and affects the appearance of certain utilities and applications on the system. The following example illustrates some effects that changing the locale has on the system.

Try the following steps (note that this will only work in the C shell):

1. Look at the date while in the default locale:

   ```
   IRIS% setenv LANG C
   IRIS% date
   Tue Aug 27 19:34:33 PDT 1996
   ```

2. Change to the French locale and check the date again:

   ```
   IRIS% setenv LANG fr
   IRIS% date
   mardi 27 août 1996, 19:35:22
   ```

Message Catalogs

A *message catalog* for a program is a file that contains the output strings used by that program. The people who localize an application create a new message catalog for each locale, translating the strings from the original message catalog into the appropriate language for the new locale. Unless you intend to modify a message catalog for use on your own system, you probably don't need to know any more than that about message catalogs.

However, there are two message catalog systems in widespread use: XPG and MNLS. IRIX uses MNLS; in that system, message catalogs used by the operating system are in the */usr/lib/locale/*localename*/LC_MESSAGES* directory.

The names of message catalogs and their locations in the filesystem can vary from one application to another. If you need to find an application's message catalogs, try looking for directories whose pathnames contain locale names or the name LC_MESSAGES.

XPG Message Catalogs

Some third-party applications may use XPG message catalogs instead of MNLS message catalogs. If you need to specify the location of such message catalogs, you can use the NLSPATH environment variable. See the *environ*(5) manual page for more information on NLSPATH if you ever need to set this variable. The documentation for any third-party application that uses XPG message catalogs should also describe how to use NLSPATH.

Note: Since IRIX doesn't use XPG message catalogs, you don't need to set NLSPATH for use with IRIX commands.

Programmers can find further information about message catalogs in the *IRIX System Programming Guide*.

Modifying X Resource Files

X-Window System-based programs use files called resource files which allow users to modify the program parameters without having to recompile the programs. These resource files sometimes contain culture-specific information. Localization of X resource files involves finding such information and saving it in files called application defaults files, or *appdefs*.

Users should rarely if ever need to modify *appdefs*. System administrators, however, might want to change some things; for instance, if an error message is confusing, a system administrator can modify the message in the *appdefs* file. Localizers may want to create new *appdefs*, based on the old ones, for new locales.

The default French and German *appdefs* for X-Window System-based applications are located in the */usr/lib/X11/fr/app-defaults* and */usr/lib/X11/de/app-defaults* directories, respectively. If you're having trouble finding other *appdefs*, try looking at the environment variables XAPPLRESDIR, XUSERFILESEARCHPATH, and XFILESEARCHPATH, which specify where to look for application-specific X resource files.

Almost all of the *appdefs* for a locale probably need to be customized if you're creating a new locale based on an older one. However, once a resource file has been localized for one language, other geographical regions using the same language can probably use that file with little or no modification.

The main items in *appdefs* that need to be changed in creating a new locale are character strings; they need to be translated into the new language. However, other items may need changing as well. Icons, for instance, may rely on idioms that don't translate very well; they may need to be redrawn. Also, translating a string may require resizing or repositioning the graphical elements (widgets) of the window the string appears in, since the translation may yield a significantly longer or shorter string.

ISO 8859 Encodings and Latin-1

ASCII is an *encoding*, a correspondence between a set of characters and a set of numbers. Most locales other than the default *C* locale are associated with other (non-ASCII) encodings. The International Standards Organization, ISO, has specified a series of character encodings numbered 8859-1 through 8859-9, each of which contains codes for all the characters necessary for several languages. The most common non-ASCII encoding, and the one used by most of the western European locales in WorldView is 8859-1, also known as Latin-1.

Printing Text Files

WorldView provides an internationalized way of printing text files on Silicon Graphics systems. Existing print solutions work for Latin-1 locales, such as France and Germany.

The solution described here is most appropriate for use with non-Latin1 locales. It contains a printer command, *ilptops*, and a new *lp* spooler interface called the Bitmap Postscript Interface. *ilptops* is used to generate PostScript files from text files in the locale of choice; the *lp* spooler interface allows addition of a printer in the current framework of *lp*.

Use the Add option of the Print Manager from the System menu in the Toolchest to install a printer. Choose the Bitmap PostScript for the Printer Type. While printing a text file using this printer, use the *-o* option with the *lp* command to send the locale to *ilptops*.

For instance, to print a Czech language file on a bitmap PostScript printer named PrLatin2, use the command:

```
IRIS% cat czechfile | lp -dPrLatin2 -o”-lcs”
```

For further information, please refer to the *ilptops* man page.

Printing Non-ASCII Characters in PostScript

The standard font encoding used by most PostScript printers includes many of the non-ASCII characters in the Latin-1 character set, but some such characters do not have the same encodings they have in Latin-1. Each time you print a file which contains non-ASCII symbols on a PostScript printer, the printer's fonts must be reencoded to use the appropriate character encodings.

In many cases, when you tell an application to print, it automatically takes care of this remapping in the PostScript file it generates. Also, the current version of IRIX makes it possible for you to print files containing non-ASCII text on a PostScript printer directly by using the usual *lp* command.

Note: In order to print Latin-1 files correctly over the network, the machine to which the printer is attached must be running IRIX version 4.0.5 or later, and the PostScript printer must have been installed using the *Generic PostScript* driver. Do not install *transcript* (also known as *Laser Printer Support*) if you wish to print files which contain Latin-1 text.

You are encouraged to use the optional IRIS Impressario printing support package, which replaces *transcript*. IRIS Impressario contains support for Latin-1 printing and international paper sizes as well as font support.

Supplementary Installation Instructions

This section provides supplemental information to the *IRIS Software Installation Guide*.

WorldView Europe Language Subsystem

The WorldView Europe language subsystem contains the following components:

- *WorldView_euro.books.european*
  This subsystem contains the IRIS InSight™ manual that describes WorldView and how it works.
- *WorldView_euro.books.de*
  These subsystems contain the German IRIS InSight™ manual. These provide complete online German help on the desktop software environment.

- *WorldView_euro.books.fr*
  These subsystems contain the French IRIS InSight™ manual. These provide complete online French help on the desktop software environment.
- *WorldView_euro.man.relnotes*
  This subsystem contains the release notes of the European Language Module. Use *grelnotes* or *relnotes* to view this online.
- *WorldView_euro.env.french*
  This subsystem contains French localization data in the form of translated system messages and X11 *app-default* files.
- *WorldView_euro.env.mmail_fr*
  This subsystem contains French localization data for MediaMail 3.2.3. This subsystem requires MediaMail 3.2.3 (in IRIX 6.2) for installation.
- *WorldView_euro.env.showcase_french*
  This subsystem contains French localization data for IRIS Showcase™ 3.4. This subsystem requires IRIS Showcase™ 3.4 (in IRIX 6.2) for installation.
- *WorldView_euro.env.german*
  This subsystem contains German localization data in the form of translated system messages and X11 *app-default* files.
- *WorldView_euro.env.mmail_german*
  This subsystem contains German localization data for MediaMail 3.2.3. This subsystem requires MediaMail 3.2.3 (in IRIX 6.2) for installation.
- *WorldView_euro.env.showcase_german*
  This subsystem contains German localization data for IRIS Showcase™ 3.4. This subsystem requires IRIS Showcase™ 3.4 (in IRIX 6.2) for installation.

Disk Space Requirements

This section lists the subsystems (and their sizes) of the WorldView software.

If you are installing this software for the first time, the subsystems marked *default* are those that are installed if you use the *go* menu item. To install a different set of subsystems, use the *install*, *remove*, *keep*, and *step* commands in *inst* to customize the list of subsystems to be installed, then select the *go* menu item.

Note: The listed subsystem sizes are approximate. Refer to the *IRIS Software Installation Guide* for information on finding exact sizes.

Table 2-3 WorldView Europe Language Subsystem Sizes

Subsystem Name	Approximate Subsystem Size (512-byte blocks)
WorldView_euro.books.european (default)	538
WorldView_euro.books.de_DesktopHelp (default)	944
WorldView_euro.books.de_ErrorHelp (default)	622
WorldView_euro.books.de_ErrorMessageHelp (default)	318
WorldView_euro.books.de_Help (default)	462
WorldView_euro.books.de_IRISEssentials (default)	7787
WorldView_euro.books.de_MediaMailHelp (default)	903
WorldView_euro.books.de_ConfTestErrors (default)	352
WorldView_euro.books.de_ConfTestHelp (default)	402
WorldView_euro.books.de_Doc_Catalog (default)	592
WorldView_euro.books.de_PerSysAdmin (default)	2025
WorldView_euro.books.de_ViewerHelp (default)	363
WorldView_euro.books.de_global (default)	479
WorldView_euro.books.de_help (default)	520
WorldView_euro.books.fr_DesktopHelp (default)	911
WorldView_euro.books.fr_ErrorHelp (default)	599
WorldView_euro.books.fr_ErrorMessageHelp (default)	318
WorldView_euro.books.fr_Help (default)	446
WorldView_euro.books.fr IRISEssentials (default)	6537
WorldView_euro.books.fr_MediaMailHelp (default)	862
WorldView_euro.books.fr_ConfTestErrors (default)	352
WorldView_euro.books.fr_ConfTestHelp (default)	399

Table 2-3 (continued) WorldView Europe Language Subsystem Sizes

Subsystem Name	Approximate Subsystem Size (512-byte blocks)
WorldView_euro.books.fr_Doc_Catalog (default)	594
WorldView_euro.books.fr_PerSysAdmin (default)	1999
WorldView_euro.books.fr_ViewerHelp (default)	360
WorldView_euro.books.fr_global (default)	900
WorldView_euro.books.fr_help (default)	516
WorldView_euro.man.relnotes (default)	13
WorldView_euro.env.Koi8 (default)	488
WorldView_euro.env.german (default)	1289
WorldView_euro.env.mmail_german	443
WorldView_euro.env.showcase_german	67
WorldView_euro.env.french (default)	1363
WorldView_euro.env.mmail_french	455
WorldView_euro.env.showcase_french	69

Installation Method

Refer to the *IRIS Software Installation Guide* for complete installation instructions. None of the subsystems require *miniroot* install.

Prerequisites

All *WorldView_euro.books* subsystems require *insight.sw.client*.

Chapter 3

Working With Applications

Many characters used in non-English languages, such as *ê*, *ã*, and *ö*, are not part of the ASCII codeset, and some applications do not properly recognize non-ASCII characters entered by a user. Applications fall roughly into three types, according to their abilities to handle such symbols:

- X-Window System-based and mixed-model applications, which generally recognize and correctly handle non-ASCII data entry
- GL-based applications that might not recognize non-ASCII symbols
- System utilities and terminal emulators that vary in their ability to handle non-ASCII symbols

This chapter discusses the behavior of each type of application as well as the behavior of a few specific applications.

Files Containing Non-ASCII Text

Here are some miscellaneous notes about non-ASCII text and how it interacts with the system and applications:

- Many applications and system utilities allow you to create files that contain non-ASCII characters. You can use *cp* or *rcp* to copy such files.
- Unless you use MediaMail or a similar application, you cannot directly mail files that contain non-ASCII characters. First you must use the *uuencode*(1) command to convert the files to a form that can be safely sent through electronic mail. See “Using Mail With Non-ASCII Text” on page 30 for a discussion of how to use *uuencode*. Also note that user names and machine names—and everything else in the mail header—must be limited to ASCII characters in order to be recognized by mailers along the mail path.

- You can use *cat* or *more* in *xwsh* or *xterm* to view files that contain non-ASCII characters; just be sure to set an appropriate locale and use an appropriate font. (See the man pages for *xwsh*(1G) and *xterm*(1) to find out how to change fonts in those programs.)
- You can create files whose names contain non-ASCII characters using the Bourne shell (*sh*), the C shell (*csh*), and most X-based applications. Most graphical applications let you use a file dialog box to select such files, even though some GL-based applications may not allow you to enter non-ASCII symbols from the keyboard.

Problems With X-based and Mixed-Model Applications

X Window System-based applications query the X server to find out which symbols are engraved on each key of the keyboard you are using. The X protocol and library were designed to use non-ASCII symbols, so most X applications recognize keys with non-ASCII symbols. You may, however, encounter a few problems with some fonts and some applications under the X Window System. This section describes two such problems.

Characters Displayed As Spaces

In order to correctly display text that contains non-ASCII symbols in an X-based application, you must use fonts that support the codeset you need. If you try to display text that contains non-ASCII symbols using a font that doesn't include them, any non-ASCII symbol may drop out. For example, the string

```
Ich bin müde.
```

might (depending on the font) be displayed like this:

```
Ich bin m de.
```

or like this:

```
Ich bin mde.
```

Most X fonts support the entire Latin-1 character set, so most applications correctly display text with Latin-1 symbols. To display characters from the other ISO 8859 encodings (Latin-2, Latin-3, and so on), you need other fonts that use those encodings.

Inaccessible *Alt Gr* Characters

Some keys on some keyboards have more than two symbols associated with them, as shown in Figure 3-1. You can usually access the third or fourth symbol by holding down the *Alt Gr* (Alternate Group, also known as the mode switch) key while pressing the key for the desired character.

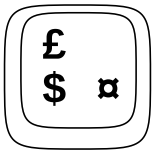

Figure 3-1 A key with an *Alt Gr*-accessible character on it

Most X-based applications handle keys with three or four symbols correctly. There are a few X-based applications, however, which do not correctly recognize more than two symbols on a given key, regardless of what those symbols might be. Pressing *Alt Gr* and a key in such an application does not generate any input.

You can use *xmodmap*(1) to rebind an unused key (such as a function key) to generate an otherwise inaccessible symbol in such an application. For example, the command:

```
IRIS% xmodmap -e "keysym F12 = ntilde Ntilde"
```

modifies the *F12* key symbol (keysym) to generate *ñ* and *Shift-F12* to generate *Ñ* in any X-based application. Most, though not all, applications correctly recognize characters generated by remapped keys.

If an application seems to have trouble finding the third or fourth character on a key, try using *xmodmap* to remove all other keysyms from the key that generates the *Mode_switch* keysym.

A Small Sampling of Silicon Graphics Applications

CASEVision tools are based on IRIS IM and provide basic internationalization support. The *jot* editor allows entry of non-ASCII symbols and displays them correctly if you use a Latin-1 font.

Note: IRIS IM is Silicon Graphics' port of OSF / Motif.

IRIS Showcase™

You can enter non-ASCII symbols into an IRIS Showcase™ (version 2.1.2 or later) document using the regular keyboard keys. All IRIS Showcase™ documents can contain non-ASCII symbols.

You can import text files that contain non-ASCII symbols if you have an appropriate font in which to display those symbols.

Other Applications

This section discusses the behavior of some specific applications.

FrameMaker

FrameMaker 3.1X and later correctly recognizes and uses non-ASCII symbols entered using keyboards which contain such symbols.

FrameMaker does not, however, recognize more than two symbols on a key. See "Inaccessible Alt Gr Characters" on page 27 for some tips on working around this problem.

FrameMaker also has its own mechanism for entering non-ASCII characters with an American keyboard. For instructions, refer to the FrameMaker documentation.

UniPress and GNU Emacs

Both of the widely-available types of *emacs* have problems with characters in the upper half of the ISO character sets (that is, 8-bit characters).

Both UniPress and GNU *emacs* display non-ASCII symbols as 3-digit octal numbers. For example, they both display the string *\351* instead of the character *é* when editing a file. Also, both kinds of *emacs* are erratic in interpreting non-ASCII keystrokes; some keys are simply ignored, while other keys are interpreted as *emacs* commands.

Terminal Emulators

The terminal emulators *xwsh* and *xterm* accept input of non-ASCII characters and correctly display text that contains non-ASCII symbols, provided that you use an appropriate font. For details on font selection, see the manual pages for these programs.

System Utilities and Shell-Based Applications

This section describes the non-ASCII-related capabilities and known bugs of the IRIX shells, a few commands *(file, ar,* and *tar*), the *vi* editor, and mail.

Using Shells

The C shell (*csh*) and Korn shell (*ksh*) correctly recognize and handle non-ASCII symbols. The Bourne shell (*sh*) handles non-ASCII characters in filenames correctly most of the time. A few bugs occur when you use wildcards on filenames with non-ASCII symbols in the Bourne shell, but these are minor and easily worked around. In most cases, wildcard characters correctly match non-ASCII as well as ASCII characters.

The *file* Command

The *file* command tries to determine the type of a specified file based on its contents. *file* can't tell the difference between text files containing Latin-1 and text files containing Latin-2 (or any of the other ISO 8859 encodings); as far as *file* is concerned, they're all 8-bit text. *file* can tell the difference, however, between ASCII files and 8-bit text files.

Creating Archives with *ar* and *tar*

You can create and read *ar* or *tar* archives that contain files with names that include non-ASCII symbols, but you may not be able to retrieve such archives on other manufacturers' systems or under earlier releases of IRIX. In other words, you can create such archives for backup purposes, but be careful about distributing them for extraction on other vendors' machines if their *ar* or *tar* commands support only ASCII characters.

The *vi* Editor

vi uses the LANG environment variable to determine character class. As a result, *vi* may not display non-ASCII characters in the *C* locale. To edit a file that contains non-ASCII characters, you may have to set the LANG environment variable to a non-ASCII locale (such as *en_US* or *fr*).

Using Mail With Non-ASCII Text

Most UNIX systems, including IRIX, use the Simple Mail Transfer Protocol (*SMTP*) to send electronic mail from system to system. The protocol specifies that data is sent in 7-bit bytes. The eighth bit of any transmitted byte is stripped off and ignored.

When you compose a message, most mail programs allow you to enter non-ASCII characters. But since all non-ASCII codesets use 8-bit bytes, any non-ASCII characters in your message are converted to essentially random ASCII characters by the time the message is received.

The fact that nearly every version of *sendmail* strips the eighth bits from mail makes it difficult to come up with a Silicon Graphics-specific solution. Even if Silicon Graphics violated SMTP protocols by modifying its system to pass 8-bit characters, any non-SGI (or older SGI) systems that the mail passed through would strip the eighth bit. The UNIX community is well aware of this problem, and a working group is designing a solution.

SGI's MediaMail software automatically encodes 8-bit characters into a 7-bit format when you send a message containing such characters, and it automatically decodes those characters at the other end. However, if you use MediaMail to send 8-bit characters to someone who isn't using MediaMail, the recipient must decode the message by hand.

Note: To make MediaMail handle 8-bit characters properly, you have to set the *textpart_charset* variable using the Options menu. See the MediaMail documentation for more information.

If you are sending a brief message that contains only a few non-ASCII symbols, you can use some of the common pure ASCII substitutions instead. For example, it is common to use "e:" to denote the 'ë' character in Internet mail.

Using *uuencode* to Encode Non-ASCII Symbols in Mail

If you are transmitting a larger document that contains many non-ASCII symbols, you can encode the document with *uuencode* before you send it. The *uuencode* program converts any binary data into a 7-bits-per-byte form that can be safely sent through electronic mail.

On the *uuencode* command line, you must specify the filename to be used when the file is decoded on the remote system. For example, the command:

```
IRIS% uuencode message.new < message > message.uu
```

encodes the contents of *message* into *message.uu*. When the *uudecode* program unpacks the message on a remote system, it stores the decoded message in a file named *message.new*.

Once you have encoded a file using *uuencode*, you can use the standard mechanism provided by your mail program to include it in a mail message. When you receive an encoded message, you can run it through *uudecode* to get the original file.

Refer to the *uuencode*(1) and *uudecode*(1) manual pages for a complete description of these commands. Refer to the documentation for your mail program to learn how to incorporate a file into a mail message.

Index

Tell Us About This Manual

As a user of Silicon Graphics documentation, your comments are important to us. They help us to better understand your needs and to improve the quality of our documentation.

Any information that you provide will be useful. Here is a list of suggested topics to comment on:

- General impression of the document
- Omission of material that you expected to find
- Technical errors
- Relevance of the material to the job you had to do
- Quality of the printing and binding

Important Note

Please include the title and part number of the document you are commenting on. The part number for this document is 007-2151-003.

Thank you!

Three Ways to Reach Us

The **postcard** opposite this page has space for your comments. Write your comments on the postage-paid card for your country, then detach and mail it. If your country is not listed, either use the international card and apply the necessary postage or use electronic mail or FAX for your reply.

If **electronic mail** is available to you, write your comments in an e-mail message and mail it to either of these addresses:

- If you are on the Internet, use this address: techpubs@sgi.com
- For UUCP mail, use this address through any backbone site: *[your_site]*!sgi!techpubs

You can forward your comments (or annotated copies of pages from the manual) to Technical Publications at this **FAX** number:
415 965-0964