IRIS Explorer User's Guide
Release 3

Document Number NP2872
ISBN 1−85206−110−3

Appendix A

Configuring Your IRIS Explorer Environment

This appendix explains how to configure various aspects of your system in order to use the IRIS Explorer more effectively. It includes:

- modifying your *.explorerrc* file
- setting environment variable(s)
- configuration commands
- remote execution and passwords
- running the X server
- creating a larger kernel

A.1. The *.explorerrc* Configuration File

IRIS Explorer reads configuration information from files on start–up. This information tells IRIS Explorer, among other things:

- where to look for modules
- how to configure the Module Librarian
- where and how to run remote modules
- what size the memory arena should be

This information can come from two separate files, one global to the host machine and one private to the user. The global configuration file is in */usr/explorer/Explorer.config* and the private copy is in the user's home directory in the file called *.explorerrc*.

A.1.1. Remote Execution of Modules

The general idea of IRIS Explorer is that you initially run it on your own workstation, but then you can use it to run some modules on remote workstations.

Software Requirements

Each machine that you use to execute modules remotely must have most of IRIS Explorer installed on it. In general, the version installed remotely must match the version installed on the local machine.

This sense of remote execution is different from that offered by the X Window System, which lets you remotely log into a workstation on which IRIS Explorer is installed, reset the DISPLAY environment variable to reference your own screen, and then run IRIS Explorer. In this case, all of IRIS Explorer will initially be running on the remote machine, but the user interface will appear on the local machine.

Defining Hosts for Remote Modules

To execute IRIS Explorer modules remotely, you must first add any host names used to your *.explorerrc* file, in this format:

```
host hostname
```

For example, suppose your local machine, called *billiejoe*, is networked to another machine named *bob*. If you can remotely login into *bob*, and IRIS Explorer is installed on it, then you can very simply set up the two machines for remote execution.

Add this line to the *.explorerrc* file:

```
host bob
```

Do this for every host you want to run modules on. When you execute IRIS Explorer again, these host names will be listed on the Hosts menu in the Module Librarian window.

IRIS Explorer uses the *rsh* command to create a computation server on machines used for execution of remote modules. Hence, you must be able to access the remote hosts you use with the *rsh* command. IRIS Explorer also provides an alternative to *rsh*, which is *cxrexec*. It is capable of prompting for a password.

See **"Remote Execution and Passwords"** for more information.

A.1.2. If IRIS Explorer Is Not Installed in */usr/explorer*

If IRIS Explorer is not installed in the directory */usr/explorer*, then you must either:

- create a symbolic link from */usr/explorer* to the actual installation directory.

  If your system uses shared libraries, as do UNIX systems, you must install the symbolic link.
- define an environment variable, *EXPLORERHOME*, to contain the name of the directory where it is installed. See **"IRIS Explorer Environment Variables"**.

For example, if IRIS Explorer on the machine *billiejoe* is located at */usr/local/explorer*, then you must set *EXPLORERHOME* to be */usr/local/explorer* in the *.cshrc* file (see *csh(1)*).

Suppose that on the remote machine *bob*, IRIS Explorer is located at */usr/people/jim/explorer*. Then the *.cshrc* file in your home directory on host *bob* must contain this line:

```
setenv EXPLORERHOME /usr/people/jim/explorer
```

The easiest thing to do is to install IRIS Explorer in the same location on all machines, which by default is */usr/explorer*.

Using the Switch Statement in a Shared .cshrc

If you have a single *.cshrc* file on several workstations, each of which may have IRIS Explorer installed in a different location, you can use the *csh* "switch" statement to set *EXPLORERHOME* correctly according to the current machine.

For example:

```
switch (`hostname`)
case billiejoe:
       setenv EXPLORERHOME /usr/local/explorer
       breaksw
case bob:
       setenv EXPLORERHOME /usr/people/jim/explorer
       breaksw
endsw
```

Note: The character (`) surrounding `hostname` in the previous example is an accent grave, not an apostrophe.

A.1.3. The *.explorerrc* File

Both the *IRIS Explorer.config* and *.explorerrc* files can contain commands that configure IRIS Explorer. The syntax is line–oriented; a backslash at the end of a line indicates that the next line is a continuation.

Exclamation points (!) at the beginning of a line designate comments (but the # sign will also work). The line syntax is very similar to that of shell scripts, namely:

```
command [arguments] [modifiers]
```

The first word designates the command, and subsequent fields designate "arguments" or "modifiers" to the command.

Either single or double quotation marks may be used to surround arguments that contain embedded blank characters.

Most arguments to commands may encode UNIX environment variables using the shell dollar−sign notation. For example, *$EXPLORERHOME* will expand to the value of the environment variable *EXPLORERHOME*, as will *${EXPLORERHOME}*. The C−shell "tilde" (~) notation for designating home directories is recognized as well. However, strings surrounded by single quotation marks (apostrophes) are not expanded in this manner.

If the file permissions are set to be executable, the file will be executed and the output read as the configuration file. If IRIS Explorer recognizes the #!CPP notation, the file is run through the C preprocessor and all environment variables are defined to the C preprocessor as C preprocessor symbols. This means you can use conditional statements for values such as hostname, which makes the file more flexible.

Note: If you use the #!CPP notation in your .explorerrc file, then you must use ! rather than # at the beginning of a comment line.

This is an example of a typical *.explorerrc* file. Comments start with an exclamation point (!) in column 1.

```
!
! Define some remote hosts
host willard
host louis

! Also look for modules in my local directory
modulepath −append ~/modules

! Set the directory where the arena and pipes go
set tempdir /usr/tmp/explorer

! Add some frequently used modules into the shelf
category shelf Render DisplayImg IsosurfaceLat ReadImg\ ReadLat Contour\

! Put the Geometric modules into their own category
category Geometry BoundBox Contour IsosurfaceLat \
   IsosurfacePyr LatToGeom PyrToGeom \
   PickLat ReadGeom VolumeToGeom

! Increase the maximum arena size for larger data sets
set arenasize 24mb
```

A.1.4. Configuration Commands

The commands that are accepted in an IRIS Explorer configuration file are described below. Optional portions are enclosed in square brackets. Ellipses denote that the previous argument may be repeated any number of times. Commands (reserved words) are shown in bold.

Setting the Module Path

The module path is a list of directories in which IRIS Explorer looks for modules and maps. The default list contains only *$EXPLORERHOME/modules*, but the *Explorer.config* file shipped with IRIS Explorer extends it. You can reset module paths from scratch, or you can extend them using the "−append" or "−prepend" modifiers.

The modulepath command states where to look for modules.

```
modulepath [−prepend] path ...
modulepath [−append] path ...
```

This command defines a list of directories to be used. By default, the existing module path is extended by the list for directories (this is equivalent to using the "−append" modifier). The default path value is *$EXPLORERHOME/modules.* Used alone, the command clears the module path to the default path value. If the "−prepend" modifier is used, the *path* is added at the head of the current directory list.

set

You can use the *set* command to set the temporary directory and the size of the arena.

set tempdir

IRIS Explorer uses a number of temporary files during its execution. While almost all of these are small, the shared memory arena file can get quite large.

```
set tempdir path
```

The *set tempdir* command states where to place any temporary files that may be needed during the course of execution, such as named pipes and shared memory arena files, if applicable. Typically, such files are placed in a uniquely named subdirectory of the directory. The default value for the base directory is */usr/tmp.*

Because shared memory arenas can become large, it is often advantageous to place the *tempdir* somewhere other than below */usr.*

set arenasize

On some workstations, modules use shared memory for data communications. The shared memory is mapped into a file that resides on disk as determined by the *set tempdir* command above.

```
set arenasize string
```

For a shared memory system, make the maximum size of the shared memory arena the designated size.

The "string" should be a number optionally followed by "mb" or "kb" for megabytes and kilobytes, respectively. This option may also be set from the command line. The default size is 16Mb.

category

Defines a new Module Librarian category with the name "name" and containing the list of modules given.

```
category name [-append] [-filter] [arg] module ...
```

The "−append" modifier appends a list of modules to a specified category. The list of modules can contain module names, such as *GenLat*, or directory names. In the latter case, the directory is added to the module path, and all modules and maps in the directory are added to the category. For example,

```
category myModules ~/explorer/modules
```

adds all modules in directory *~/explorer/modules* to the category myModules. You can use the "−filter" modifier with an **arg** to filter the list of modules against the argument pattern. The **arg** accepts shell wildcards such as *, ? and [. For example,

```
category Readers -filter Read* Modules
```

puts all the *Read** modules in directory *Modules* in a category called *Readers.*

There is a designated category in the Module Librarian called "shelf." Modules in the shelf category are displayed in a small panel at the bottom of the Librarian’s window.

host

Modules are organized by categories and by hosts in the Module Librarian. When you add a host, modules can be executed on that machine.

```
host name [-command string]
```

The modifier "−command" may be used to define an alternate way of starting a local communications server (LC) on remote hosts. The single argument after "−command" (which must be enclosed in quotation marks if it contains blanks) specifies the alternate command.

The UNIX command IRIS Explorer uses to start the LC on a remote host is the given command (*rsh* by default) followed by the hostname, followed by a shell command that will be invoked remotely to start the LC. For example, suppose this appears in your *.explorerrc* file:

```
host bob −command "myrsh −x"
```

IRIS Explorer will invoke

```
myrsh −x bob <shell command to start the LC>
```

If the word HOST appears in the alternate command, the hostname is substituted in its place when executing the command. Therefore, you can use the command:

```
host bob −command "rsh HOST −l joe"
```

to start a remote LC under the account "joe" on *bob*.

A.2. IRIS Explorer Environment Variables

You can set this environment variable:

EXPLORERHOME

By default, the IRIS Explorer system is installed into the directory */usr/explorer*. If you choose to move the directory elsewhere, for example, because you need the disk space, then you will need to define a UNIX environment variable named *EXPLORERHOME*. Normally, this is not necessary. A more reliable way to keep IRIS Explorer elsewhere is to place a symbolic link from */usr/explorer* to the correct location.

There are also several environment variables that pertain to the building and installation of new modules. They are described in Chapter 2, "Understanding the Module Builder" in the *IRIS Explorer Module Writer's Guide.*

A.3. Remote Execution and Passwords

In this release, IRIS Explorer uses the *rsh(1)* command to start local communications servers (LC) on remote machines. This takes place only the first time you create a librarian or launch a module on a remote machine. In order for IRIS Explorer to be able to start LCs, *rsh* to those machines must work. For example, if you want to launch modules on a host named "labserver," verify that you have access via *rsh* to that machine by typing the following command:

```
/usr/bsd/rsh labserver pwd
```

If it responds with the name of your remote home directory, all is well. If it responds with "Permission denied" you need to enable *rsh.*

You can enable access to that machine by creating a file named *.rhost* in your remote home directory. See the *rsh(1)* manual page for details.

The "−command" modifier on the host line allows the redefinition of the command to use to try to get a module communications server running on a remote machine. By default, the *rsh* command is used. This may not be appropriate for all installations, because *rsh* does not allow for password prompting and the associated protections.

IRIS Explorer provides an alternate command similar to *rsh* that can prompt for a password. This command, *cxrexec*, uses the *rexec* library call and prompts for a password in a small pop−up window. For example, if access to a host named "securehost" requires a password, the following line in *.explorerrc* will result in the password prompt being issued:

```
host securehost -command cxrexec
```

The command *cxrexec* has the same command line options as *rsh*. Either command can be modified in *.explorerrc* to add options. However, when additional *rsh* or *cxrexec* options are listed, the entire command must be enclosed in quotation marks:

```
host superhost -command "rsh HOST -l guest"
host superhost -command "cxrexec HOST -l guest"
```

These examples demonstrate another feature of the alternate command option: wherever the word "HOST" appears, the name of the remote host will replace it.

If the word "HOST" does not appear in the command, it is appended to the end. Hence, the following two lines are functionally equivalent:

```
host superhost -command "rsh HOST"
host superhost -command "rsh"
```

Other UNIX programs may be used instead of *rsh* and *rexec*, depending on local conventions.

A.4. Running the X Server

The type of visual you run on your X server determines the number of colors available for modification, and this affects how many *GenerateColormap* modules you can run at one time.

A.4.1. Selecting a Visual

The X server supports several classes of color visuals. Each visual class uses a different method to display color. The *GenerateColormap* module is designed to function properly with the *PseudoColor* and *TrueColor* visual classes. The PseudoColor visual uses the hardware colormap to display a small number of colors from a much larger palette of possible colors. The TrueColor visual class writes colors directly to the graphics frame buffer.

The visual classes have a *depth* that specifies the number of bit planes it uses for display. A PseudoColor visual with a depth of 8 displays 256 colors at one time; one with a depth of 12 displays 4096 colors. The TrueColor visual supports depths of 24 and 12 bits. The 24−bit TrueColor visual can display 8 bits for red, green and blue, or over 16 million possible colors at one time. The 12−bit TrueColor visual displays 4096 colors at one time, with 4 bits for red, green and blue.

Choosing the best visual depends on your workstations hardware and the other applications you run. Your workstation must have at least as many bit planes as the depth of the visual that you choose. You generally want to run the 24−bit TrueColor visual if you can; otherwise, try the 12−bit TrueColor or PseudoColor visuals.

High−quality applications should run properly with any of these visuals, but other applications may not. Some applications, such as those that do color table animation, may not run with a TrueColor visual. Some applications may make assumptions about the depth of a PseudoColor visual, for example, that the color index will fit into 8 bits. You can start with the highest quality visual supported by your workstation, try your other applications, and then adjust the visual to fit your system's capabilities.

Note: See your local documentation on how to reconfigure the X server.