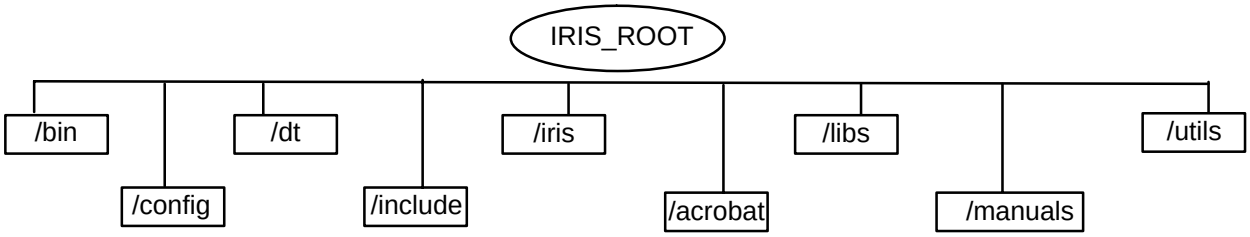
1. Introduction to IRIS Programming

1.1 Directory Organization

IRIS source files, object modules, and library routines are placed in separate branches of the IRIS directory tree, as shown in Figure 1–1.

Figure 1–1: IRIS Directory Tree



IRIS_ROOT refers to the root directory for the IRIS system. It is often **/usr/sigmet**; **IRIS_ROOT** is defined as an environment variable. When referring to a file within the IRIS directory tree, this manual assumes **IRIS_ROOT** to be the root directory. All pathnames are relative to **IRIS_ROOT**, unless otherwise noted.

Table 1–1 gives a short description of the library routine subdirectories, Table 1–2 lists the IRIS subdirectories, Table 1–3 describes the utility subdirectories, and Table 1–4 lists the Configuration subdirectories. All system wide header files are in the **${IRIS_ROOT}/include** directory.

Table 1–1: Contents of Library Subdirectories

Directory	Description
antenna	Antenna driver library routines that control the radar and antenna.
config	Configuration subroutines that save and load IRIS setup files.
dsp	DSP driver routines that control the actions of the signal processor.
lib	Files with the `.a` extension are stored here.
link	Link output file.
misc	Miscellaneous libraries.
nordrad	Nordrad supplied API.
printer	Printer driver library routines that send text and IRIS graphical products to the supported color printers.

Table 1–1: Contents of Library Subdirectories (cont.)

Directory	Description
share	Source code for shared memory routines that perform various operations, such as handling the ingest file directory, handling errors and other events, etc.
tvsubs	High level Graphics display library routines.
user	General-purpose routines.
uxsig	X-Windows support routines.
vtv	Virtual TV routines that read and write IRIS graphical products to buffers in memory.
xpm	X color pixmap library.
xsig	X functions.

Table 1–2: Contents of IRIS Subdirectories

Directory	Description
archive	Source code for the archive process.
examiners	Code for **rays** and **productx**.
ingest	Source code for the ingest process.
ingfio	Source code for the ingfio process.
network	Source code for the network process.
nordrad	Nordrad product receiver.
output	Source code for the output process.
product	Source code for the product process.
reingest	Source code for the reingest process.
server	Source code for the IRIS server.
siris	Source code for the `siris` program.
utils	Contains the source code for the **show_iris** and **show_machine_code** utilities.
watchdog	Source code for the watchdog process.
window	Source code for the X-window handler.
xuif	IRIS main menu program.

Table 1–3: Contents of Utilities Subdirectories

Directory	Description
custom	Custom utilities.
examples	Public source code, including UPI routines.
pipes_in	Source code for the input pipes.
pipes_out	Source code for the output pipes.

Table 1–4: Contents of Configuration Subdirectories

Directory	Description
init	Fonts and initializing scripts.
menu	Menu configuration files.
overlay	Overlay definition files.

1.2 Setting up the Development Environment

You may need to compile your own programs using IRIS headers, and which may link to IRIS libraries on occasion, for example:

- If you are writing your own code to read or write IRIS products using the IRIS data structures.
- If you are writing your own input or output pipe program to convert the data to another format.
- If you are writing your own code to interface to the RCP and RVP using our public APIs. This might be an offline diagnostic program, for example.
- When defining new product types using UPI.

First, be sure to install the source files, object modules, libraries, and emacs on your system. This is done by pushing these 4 buttons in the **install** program. See the instructions for your operating system in the *IRIS Installation Manual*.

Second, you must also define several environment variables that are not required for normal operation of IRIS. This is done in your ~/.profile file, by adding a line similar to the following:

```
. /usr/sigmet/config_template/init/develop.profile
```

If you are compiling on Linux, then you need to add the following symbolic links:

```
# ln –s /usr/X11R6/include/Xm /usr/include/Xm
# ln –s /usr/X11R6/include/Mrm /usr/include/Mrm

# cd /usr/lib
# ln –s libtk8.3.so libtk.so
```

To compile the HDF5 related code you will need szlib. You can download a tar file with this code, or get it from Sigmet. It is not available in a rpm format and not available from RedHat.

Setting up a developer source tree:

Before you start writing new code, you need to set up directories to work in. Do this with the following commands in the operator account:

```
$ cd
$ mkdir sigmet
$ cd sigmet
$ mk_iris_dir –iris –create
```

All the main IRIS source code and libraries reside in the release directory tree starting at ${IRIS_ROOT}. It is all in 3 major subdirectories: libs, iris, and utils. When making a change, do not change the source in the release tree. Instead copy the files for the directory you are changing to the same subdirectory of ~/sigmet. For example, to make a change to the RainbowToIris input pipe, copy the files from

${IRIS_ROOT}/utils/rainbow to ~/sigmet/utils/rainbow. You can then make your changes and compile there. If you are modifying one of IRIS's standard pipes, SIGMET strongly recommends that you rename it to a custom name. That way it will not get replaced by the next upgrade.

Links for SIGMET Makefiles

Our Makefiles contain some private versions of shell commands. If you are developing in the operator account, you can bypass this by making the following links:

```
# cd /usr/local/bin
# ln –s /bin/chown ./rootchown
# ln –s /bin/chgrp ./rootchgrp
# ln –s /bin/chmod ./rootchmod
# ln –s /bin/cp    ./rootcp
```

To compile and link an IRIS library or application:

Each directory installed on your system contains a makefile to build the application or library contained there. To run the makefile, switch to the directory containing the source code and type "**`make depend`**" to build the dependencies, then "**`make`**".

To configure emacs:

For HP 10.20 and SGI platforms, emacs is installed from the IRIS release cdrom.

Linux platforms only: Emacs is installed as part of the Linux operating system. The "emacs" button in the IRIS **install** program just installs some configuration files. To configure emacs to use SIGMETs standard configuration, you need to make a symbolic link as follows:

```
# cd /usr/share/emacs/20.7
# ln –s /usr/sigmet/config_template/emacs ./site–lisp
```

Emacs is also controlled by your ~/.emacs file. Please edit this file. For the SIGMET fontlocks to work, you need to comment out the line:

```
;;(global–font–lock–mode t)
```

If you wish to remove the graphic toolbar, add:

```
(tool–bar–mode –1)
```

HP–UX 11.11 platforms only: Emacs is installed as part of the operating system. Do not press the "emacs" button in the IRIS **install** program. To configure emacs, you need to make a symbolic link as follows:

```
# cd /opt/OpenSource/share/emacs/20.7
# mv site–lisp site–lisp–org
# ln –s /usr/sigmet/config_template/emacs ./site–lisp
```

The emacs directory will vary depending on your emacs version.

To compile IRIS/Web code:

You need to install the tomcat5 and Java rpms. Do this by following the insturctions in the chapter on *Installing IRIS/Web Server* in the *Software Installation Manual*. Do the sections up through *Configuring Java*.

To compile IRIS HDF5 and python code:

You need to install the szlib tar file. Please contact Sigmet to get this.

To get uuencode and uudecode:

Linux Platforms Only: You need these utilities, get them by installing the individual rpm from the RHEL4 cdrom #4:

```
# cd /mnt/cdrom/RedHat/RPMS
# rpm –Uvh sharutils–4.2.1–22.i386.rpm
```