
SCIRun Installation Guide for SGI Irix

(v. 1.24.2)

Table of Contents

1. Introduction	1
2. Installation from Source Code	2
2.1. Downloading and Unpacking Sources	2
2.2. Building Internal Third Party Software	4
2.3. Building External Thirdparty Software	7
2.4. Building SCIRun	7
3. Before Starting SCIRun	9
3.1. Dynamic Compilation	9
3.2. Accessing Data Sets	10
4. Starting SCIRun	10
5. Getting Help	11
A. External Thirdparty Installation Instructions	11
A.1. BLAS (or ATLAS), and LAPACK Installation	11
A.1.1. Source Code	11
A.1.2. ATLAS	12
A.2. PETSc Installation	13
A.2.1. Introduction	13
A.2.2. Download and Unpack	13
A.2.3. Build	14
A.2.4. SCIRun PETSc Configure Options	15
A.2.5. The SolveMatrix Module	15
A.3. HDF5 Installation	15
A.3.1. Download and Unpack	16
A.3.2. Building	16
A.3.3. SCIRun HDF5 Configure Options	17
A.4. MDSplus Installation	17
A.4.1. MDSplus Configure Options	17
B. Single Source, Multiple Binaries	18
C. Local SCIRun Source Code Control Using CVS	18
C.1. Introduction	18
C.2. Importing a Release	19
C.3. Creating a Working Scirun Directory Tree	21
C.4. Common CVS Commands	23

1. Introduction

This guide provides instructions for installing SCIRun, its sample data sets, and SCIRun documentation on the SGI Irix platform. SCIRun may be installed from built from source code.

SGI Irix platform guidelines are listed in Table 1, "Platform Guidelines for SGI Irix".

Disk space requirements are listed in Table 2, "Disk Space Requirements".

Table 1. Platform Guidelines for SGI Irix

Hardware	Software
<ul style="list-style-type: none"> • MIPS R10000 (or more recent) CPU • 512 Mb (or more) RAM 	<ul style="list-style-type: none"> • Irix 6.5 • MIPSpro compiler 7.3.1.1m (or later) • SCIRun's MatlabInterface package requires Matlab version 6 or version 7.

Table 2. Disk Space Requirements

Build with debugging support	1.3 GB
Optimized build	650 MB
SCIRun sample data (optional)	1 GB

Proceed to Section 2, “Installation from Source Code” to build SCIRun from source code.

2. Installation from Source Code

Follow these steps to install SCIRun :

1. Download and unpack SCIRun sources, packages, third party software, sample data sets, and documentation.
2. Build internal third party software. Internal third party software is distributed with SCIRun.
3. Install external third party software.

External third party software is not distributed with SCIRun and must be obtained separately. Some external third party software is optional (e.g. BLAS/LAPACK) and if installed will provide SCIRun with additional capabilities.

Other external third party software is required by some SCIRun packages. For example, ITK is required by the Insight package.

4. Build SCIRun.

Proceed to Section 2.1, “Downloading and Unpacking Sources”.

2.1. Downloading and Unpacking Sources

1. Any number of SCIRun *Problem Solving Environment* (PSE) Packages may be built from source

code. A PSE is built from a particular set of SCIRun packages, which can be downloaded from SCI's software archive¹ web site. For example, to build the BioPSE PSE, download the following:

```
SCIRun.1.24.2.tar.gz
BioPSE.PKG.1.24.2.tar.gz
MatlabInterface.PKG.1.24.2.tar.gz
Teem.PKG.1.24.2.tar.gz
```

To build the Fusion PSE, download the following:

```
SCIRun1.24.2.tar.gz
Fusion.PKG.1.24.2.tar.gz
Teem.PKG.1.24.2.tar.gz
DataIO.PKG.1.24.2.tar.gz
```

In all cases, the file `Thirdparty_install.1.24.2.tar.gz` must also be obtained.

SCIRun's sample data, `SCIRunData.1.24.2.tar.gz` should also be downloaded.

SCIRun documentation, `SCIRunDocs.1.24.2.tar.gz`, may also be downloaded.

2. Unpack SCIRun sources:

```
gunzip -c SCIRun.1.24.2.tar.gz | tar xf -
```

This creates a directory called `SCIRun`. Below, the term `SCIRun` refers to this directory.

3. Unpack the compressed package sources into `SCIRun/src/Packages`. For example, to unpack the BioPSE package type:

```
gunzip -c BioPSE.PKG.1.24.2.tar.gz | (cd SCIRun/src/Packages && tar xf -)
```

Repeat for other packages.

4. Unpack the sample dataset tarball(s). The following instructions install the datasets in directory `SCIRunData` inside of the download directory.

- If unpacking `SCIRunData.1.24.0b_to_1.24.1.tar.gz`, first rename the existing `SCIRunData/1.20.0b` to `SCIRunData/1.20.1`:

```
pushd SCIRunData
mv 1.20.0b 1.20.1
```

then:

```
popd
gunzip -c SCIRunData.1.24.0b_to_1.24.1.tar.gz | tar xf -
```

- If installing `SCIRunData.1.24.2.tar.gz` type:

```
gunzip -c SCIRunData.1.24.2.tar.gz | tar xf -
```

¹ http://software.sci.utah.edu/archive_entry.html

- If installing FusionData.1.24.2.tar.gz type:

```
gunzip -c FusionData.1.24.2.tar.gz | tar xf -
```

5. Unpack third party sources.

```
gunzip -c Thirdparty_install.1.24.2.tar.gz | tar xf -
```

This creates the directory Thirdparty_install.1.24.2.

6. Unpack the documentation:

```
gunzip -c SCIRunDocs.1.24.2.tar.gz | tar xf -
```

This creates the directory SCIRunDocs.1.24.2. Point a browser at SCIRun-Docs.1.24.2/doc/index.html to access SCIRun's online documentation.

Note

Appendix C, *Local SCIRun Source Code Control Using CVS* discusses a method for putting the downloaded SCIRun sources under control of the CVS version control system. This is useful if the SCIRun distribution will be used for development. This is *not* a requirement for installing SCIRun.

Proceed to Section 2.2, "Building Internal Third Party Software"

2.2. Building Internal Third Party Software

Internal third party software is distributed with SCIRun and is required for SCIRun to run properly. It must be built before building SCIRun. This section provides instruction for building internal third party software.

Note

Python² (version 1.5.2 or later) and GNU make³ (version 3.79 or later) must be present to proceed with source installation.

Follow these steps to build the internal third-party distribution:

1. **cd** to the third-party software directory:

² <http://www.python.org>

³ <http://www.gnu.org/directory/make.html>

```
cd Thirdparty_install.x.x.x
```

2. *Optional:* Enable PNG support in the Teem third party library. To enable PNG support in the Teem third party library set the `TEEM_ZLIB` and `TEEM_PNG` environment variables:

For sh-type shells:

```
TEEM_ZLIB=1
TEEM_PNG=1
export TEEM_ZLIB TEEM_PNG
```

for csh-type shells:

```
setenv TEEM_ZLIB
setenv TEEM_PNG
```

If these variables are not set, PNG support will be omitted.

If PNG support is enabled and PNG libraries (and corresponding header files) are not installed in a standard location (`/usr/lib`) then environment variables `TEEM_PNG_IPATH` and `TEEM_PNG_LPATH` must be set. For example, if PNG libraries and headers are installed in `/usr/local` then `TEEM_PNG_IPATH` and `TEEM_PNG_LPATH` are set as follows:

For sh-type shells:

```
TEEM_PNG_LPATH=-I/usr/local/lib
TEEM_PNG_IPATH=-I/usr/local/include
export TEEM_PNG_LPATH TEEM_PNG_IPATH
```

for csh-type shells:

```
setenv TEEM_PNG_LPATH -I/usr/local/lib
setenv TEEM_PNG_IPATH -I/usr/local/include
```

If ZLIB libraries (and corresponding header files) are not installed in a standard location (`/usr/lib`) then environment variables `TEEM_ZLIB_IPATH` and `TEEM_ZLIB_LPATH` must be set. For example, if ZLIB libraries and headers are installed in `/usr/local` then `TEEM_ZLIB_IPATH` and `TEEM_ZLIB_LPATH` are set as follows:

For sh-type shells:

```
TEEM_ZLIB_LPATH=-I/usr/local/lib
TEEM_ZLIB_IPATH=-I/usr/local/include
export TEEM_ZLIB_LPATH TEEM_ZLIB_IPATH
```

for csh-type shells:

```
setenv TEEM_ZLIB_LPATH -I/usr/local/lib
setenv TEEM_ZLIB_IPATH -I/usr/local/include
```

3. Run the third party installation script.

The simplest script invocation requires a single argument:

```
python install dir
```

where *dir* is the location you wish to install the third-party installation.

The installation script takes additional optional arguments:

```
python install dir bits jobs
```

dir is the third-party installation directory. *bits* is either 32 or 64 and specifies one of 32 bit or 64 bit binaries. *jobs* is the number of processors available for **make**'s use when building the software. For example:

```
python install /usr/local/SCIRun/Thirdparty 64 2
```

or

```
python install /usr/local/SCIRun/Thirdparty 32 1
```

When finished, the installation script displays the complete path of the third-party software installation directory. Make note of this value, it is used in the configure step of the SCIRun build process described in Section 2.4, “Building SCIRun”.

The complete path name of the third-party installation directory reflects the parameters given to the install script. This allows the user to maintain multiple third-party installations for different operating systems and machine architectures. For example, on a SGI Irix machine, the installation command:

```
python install /usr/local/SCIRun/Thirdparty 64
```

creates the installation under:

```
/usr/local/SCIRun/Thirdparty/1.8/IRIX64/MIPSpro-7.3.1.3m-32bit
```

All, or part, of the third-party software may be reinstalled by repeating the install command line. The script asks for a **y** or **n** response to each tarball in the distribution. Answer **y** to reinstall.

Proceed to Section 2.3, “Building External Thirdparty Software”.

2.3. Building External Thirdparty Software

SCIRun for SGI Irix provides support for BLAS/LAPACK, PETSc⁴, MDSplus⁵, and HDF5⁶ software. External third party software provides SCIRun with additional capabilities:

- LAPACK routines are used by SCIRun modules to perform SVD, eigenvalue, and eigenvector computations. BLAS is used by SCIRun modules for matrix multiplication. SCIRun modules will use less efficient algorithms if BLAS and LAPACK are not installed.

BLAS/LAPACK must be installed if PETSc support is desired.

- The PETSC library provides equation solvers to SCIRun's SolveMatrix module. PETSC support is only available by building SCIRun from source code.
- SCIRun's Insight package encapsulates ITK toolkit segmentation, registration, and image processing filters in SCIRun modules. The ITK toolkit is required by SCIRun's Insight package.
- HDF5 libraries gives modules of SCIRun's DataIO package the ability to read HDF5 data.
- MDSplus libraries gives modules of SCIRun's DataIO package the ability to read MDSplus data from an MDSplus server.

If external thirdparty software is desired, it must be installed before building SCIRun.

Section A.1, “ BLAS (or ATLAS), and LAPACK Installation ” provides installation instructions for BLAS and LAPACK.

Section A.2, “PETSc Installation” provides installation instructions for PETSC.

Section A.3, “HDF5 Installation” provides installation instructions for the HDF5 libraries.

Section A.4, “MDSplus Installation” provides installation instructions for the MDSplus libraries.

Proceed to Section 2.4, “Building SCIRun” after building external third party software.

2.4. Building SCIRun

1. Run SCIRun's configure script. Do this from a sub-directory of directory SCIRun (but *not* SCIRun/src) as follows:

```
mkdir SCIRun/build-dir
cd SCIRun/build-dir

../src/configure --with-thirdparty=third-party-dir \
--enable-package="package-name1 package-name2 ..."
```

in which case SCIRun is built in the directory SCIRun/*build-dir*.

Build-dir is often bin or a name that reflects the machine architecture.

Above, replace *third-party-dir* by the path provided by the third-party installation script.

⁴ <http://www-unix.mcs.anl.gov/petsc/petsc-2/>

⁵ <http://www.mdsplus.org>

⁶ <http://hdf.ncsa.uiuc.edu/HDF5>

Replace "*package-name1 package-name2 ...*" by a list of package names, e.g. "BioPSE MatlabInterface". An example configure command is:

```
../src/configure \
--with-thirdparty=/usr/local/SCIRun/Thirdparty/1.8.0/Irix/CC-32bit \
--enable-package="BioPSE MatlabInterface"
```

Use option `--with-unipetsc` to add support for PETSc. Options `--with-lapack` and `--with-blas` must be used if the BLAS and LAPACK libraries are not in `/usr/lib` (Section A.2.4, "SCIRun PETSc Configure Options" describes these options). For example:

```
../src/configure \
--with-thirdparty=/usr/local/SCIRun/Thirdparty/1.8.0/Irix/CC-32bit \
--enable-package="BioPSE MatlabInterface" \
--with-unipetsc=/usr/local/petsc \
--with-blas=/usr/local/lib \
--with-lapack=/usr/local/lib
```

Use option `--with-hdf5` to add HDF5 support. Use option `--with-mdsplus` to add MDSplus support. Also enable packages Teem and DataIO. For example:

```
../src/configure \
--with-thirdparty=/usr/local/SCIRun/Thirdparty/1.8.0/Irix/CC-32bit \
--enable-package="Fusion Teem DataIO" \
--with-hdf5=/usr/local/hdf5 \
--with-mdsplus=/usr/local/mdsplus
```

Support for TrueType (scalable anti-aliased) fonts in PowerApp BioImage and module ViewSlices is enabled with option `--with-freetype=`*path-to-freetype-lib-directory*. *Path-to-freetype-lib-directory* is the path to a directory containing freetype2 libraries, for example `/usr/local/lib`. Option `--with-freetype` is not needed when freetype2 libraries exist in `/usr/lib`.

Not all of **configure**'s options have been mentioned. Running **configure --help** lists them all.

2. Run GNU make from *build-dir*.

```
cd build-dir
make
```

Be sure that **make** invokes GNU make. GNU make is sometimes named **gmake** or **gnumake**.

For a multiprocessor system GNU make's `-j` option can be used to reduce the build time:

```
make -j N
```

where *N* is the number of processors available for use by make. Using this option on a multiprocessor machine can significantly reduce the time to build SCIRun.

After a successful build, *build-dir* will contain the *scirun* and PowerApp executables.

Tip

It is possible to build SCIRun for multiple machine architectures and multiple sets of configure options within the same source code directory. See Appendix B, *Single Source, Multiple Binaries* for details.

Read Section 3, “Before Starting SCIRun” before starting SCIRun.

3. Before Starting SCIRun

Before starting SCIRun:

- Understand *dynamic compilation*— Section 3.1, “Dynamic Compilation”
- Initialize environment variables `SCIRUN_DATA`—Section 3.2, “Accessing Data Sets”

The following sections discuss each of these tasks.

3.1. Dynamic Compilation

Before starting SCIRun users should be aware of a feature called *Dynamic Compilation*.

Dynamic compilation is a technique used by SCIRun to discover and generate code for the data types and algorithms used by modules. This is done at runtime and is done once for each new data type and algorithm encountered. This technique provides a number of benefits (not discussed here).

By default, code generated by dynamic compilation is stored in each user's home directory in `~/SCIRun/on-the-fly-libs/os` where *os* is one of `Linux`, `IRIX`, etc. The location of dynamically generated code is changed by setting the value of the environment variable `SCIRUN_ON_THE_FLY_LIBS_DIR` to the desired directory. For example, in a shell terminal type:

```
mkdir ~/scratch/SCIRun_otfl
```

then, for sh-type shells type:

```
SCIRUN_ON_THE_FLY_LIBS_DIR=~/scratch/SCIRun_otfl  
export SCIRUN_ON_THE_FLY_LIBS_DIR
```

for csh-type shells type:

```
setenv SCIRUN_ON_THE_FLY_LIBS_DIR ~/scratch/SCIRun_otfl
```

The location of the “on-the-fly” directory can also be changed by setting the value of `SCIRUN_ON_THE_FLY_LIBS_DIR` in the `.scirunrc` file. This file is read by SCIRun at startup. `.scirunrc` could contain, for example:

```
SCIRUN_ON_THE_FLY_LIBS_DIR=$(HOME)/SCIRun_otf
```

See the *SCIRun Users Guide*⁷ for more information on the use of `.scirunrc`.

3.2. Accessing Data Sets

The environment variables `SCIRUN_DATA` and `SCIRUN_DATASET` specify the complete path to a SCIRun *data set*. `SCIRUN_DATA` specifies a directory that contains a collection of data set directories. `SCIRUN_DATA` is used in conjunction with `SCIRUN_DATASET` to specify the full path of a data set. `SCIRUN_DATASET` specifies a specific data set directory.

When working with SCIRun's tutorial⁸, the value of `SCIRUN_DATA` must refer to the sample data sets directory, `SCIRunData`.

for sh-type shells type:

```
SCIRUN_DATA=path-to-SCIRunData
export SCIRUN_DATA
```

for csh-type shells type:

```
setenv SCIRUN_DATA path-to-SCIRunData
```

`SCIRUN_DATASET` must refer to one of the data set directories within `SCIRunData`, e.g. `utahtorso`:

```
SCIRUN_DATASET=utahtorso
export SCIRUN_DATASET
```

or

```
setenv SCIRUN_DATA utahtorso
```

4. Starting SCIRun

If built from source code, SCIRun's installation can be tested by changing the working directory (`cding`) to SCIRun's build directory and typing `./scirun`.

After starting SCIRun, its network editor window and shell prompt should appear.

If the window does not appear, note any displayed errors and see Section 5, "Getting Help".

See SCIRun's tutorial⁹ for an introduction to the use of SCIRun. The tutorial uses the previously downloaded sample data sets.

⁷ http://software.sci.utah.edu/doc/User/Guide/usersguide/srug4_5.html

⁸ http://software.sci.utah.edu/doc/User/Tutorials/SCIRun_Intro/index.html

⁹ http://software.sci.utah.edu/doc/User/Tutorials/SCIRun_Intro/index.html

See the SCIRun Users Guide¹⁰ for particulars on the use of SCIRun.

See tutorials for the BioFEM¹¹, BioTensor¹², BioImage¹³, and FusionViewer¹⁴ PowerApps.

5. Getting Help

Subscribe to the SCIRun users list. Use it to get answers to installation and use questions. To subscribe send email to: <majordomo@sci.utah.edu> with the following in the body of the message (no subject):

```
subscribe scirun-users
```

To ask a question send email to: <scirun-users@sci.utah.edu>

Include a verbatim (cut and paste) copy of errors displayed. Also include a short paragraph describing the context of the error (creating a module, running a network, etc.) and the version of SCIRun used.

See also the Installation¹⁵ and User¹⁶ FAQs.

A. External Thirdparty Installation Instructions

A.1. BLAS (or ATLAS), and LAPACK Installation

PETSc requires BLAS, or its equivalent ATLAS, and LAPACK. BLAS/ATLAS and LAPACK must be installed before installing PETSc. A vendor version of BLAS/ATLAS is recommended. Skip this section if BLAS/ATLAS and LAPACK are installed.

BLAS and LAPACK are installed from a combined BLAS and LAPACK source distribution.

ATLAS¹ is a portable BLAS implementation. ATLAS can be installed in place of BLAS. ATLAS is installed from its source code distribution.

A.1.1. Source Code

Follow these steps to install BLAS and LAPACK from the combined source distribution:

1. Download the source distribution:

¹⁰ <http://software.sci.utah.edu/doc/User/Guide/usersguide/index.html>

¹¹ <http://software.sci.utah.edu/doc/User/Tutorials/BioFEM/BioFEM.html>

¹² <http://software.sci.utah.edu/doc/User/Tutorials/BioTensor/BioTensor.html>

¹³ <http://software.sci.utah.edu/doc/User/Tutorials/BioImage/index.html>

¹⁴ <http://software.sci.utah.edu/doc/User/Tutorials/FusionViewer/FusionViewer.html>

¹⁵ <http://software.sci.utah.edu/doc/Installation/FAQ/faq.html>

¹⁶ <http://software.sci.utah.edu/doc/User/FAQ/faq.html>

¹ <http://math-atlas.sourceforge.net/>

```
ftp://ftp.mcs.anl.gov/pub/petsc/fblaslapack.tar.gz
```

2. Unpack the BLAS/LAPACK distribution:

```
gunzip -c fblaslapack.tar.gz | tar xvf -
```

This creates the `fblaslapack` directory.

3. Build BLAS/LAPACK:

```
cd fblaslapack  
make
```

This creates libraries `libfblas.a` and `liblapack.a` in directory `fblaslapack`.

A.1.2. ATLAS

Follow these steps to install ATLAS:

1. Download the source distribution:

```
https://sourceforge.net/project/showfiles.php?group\_id=23725
```

Choose from the list of files a stable, platform-independent distribution, e.g. `atlas3.6.0.tar.gz`.

2. Unpack the ATLAS distribution:

```
gunzip -c atlasx.x.x.tar.gz | tar xvf -
```

This creates a directory named `ATLAS`.

3. Build ATLAS. Type:

```
make config CC=path-to-c-compiler
```

The above command will print an system architecture string. Use the architecture string in the following command:

```
make install arch=arch-string
```

The above command will build ATLAS libraries in directory `ATLAS/lib/arch-string` directory.

A.2. PETSc Installation

A.2.1. Introduction

The Portable, Extensible, Toolkit for Scientific Computation² (PETSc) is a library of data structures and functions for the parallel solution of partial differential equations.

When PETSc libraries are installed, and SCIRun is configured to use them, SCIRun's SolveMatrix module enables the use of PETSc solvers.

The installation of PETSc is optional. SolveMatrix provides built-in solvers if PETSc is not available.

PETSc is built and installed from its source code distribution. SCIRun supports PETSc versions 2.1.5 and 2.1.6. These are older versions of PETSc and are available via a web browser from the following site:

```
ftp://ftp.mcs.anl.gov/pub/petsc/software_old
```

Note

- SCIRun supports only the uniprocessor build of PETSc. SCIRun assumes a shared memory architecture, while multi-processor PETSc assumes a distributed memory architecture.
- BLAS or ATLAS and LAPACK libraries must be installed before installing PETSc. See Section A.1, “BLAS (or ATLAS), and LAPACK Installation” for BLAS³ and LAPACK⁴ installation instructions.
- PETSc must be installed before installing SCIRun. See Section A.2.3, “Build” for PETSc installation instructions.
- SCIRun must be configured to support PETSc. See Section A.2.4, “SCIRun PETSc Configure Options” for information on configuring SCIRun to use PETSc.

Start a terminal application before proceeding. Command line tools are used to install PETSc and BLAS/LAPACK.

A.2.2. Download and Unpack

Download and unpack PETSc's source distribution:

1. Browser download:

```
ftp://ftp.mcs.anl.gov/pub/petsc/software_old/v2.1.6.petsc.tar.gz5
```

2. Unpack:

² <http://www-unix.mcs.anl.gov/petsc/petsc-2/>

³ <http://www.netlib.org/blas/>

⁴ <http://www.netlib.org/lapack/>

⁵ <ftp://ftp.mcs.anl.gov/pub/petsc/petsc.tar.gz>

```
gunzip -c v2.1.6.petsc.tar.gz | tar xvf -
```

creates the directory `petsc-x.x.x` where `x.x.x` is the PETSc version.

A.2.3. Build

Follow the steps below to build PETSc. See the PETSc installation home page⁶ for complete installation instructions.

1. Change the working directory to the petsc directory:

```
cd path_to/petsc-x.x.x
```

2. Set `PETSC_DIR` environment variable. For an sh-type shell:

```
PETSC_DIR=`pwd`; export PETSC_DIR
```

for a csh-type shell:

```
setenv PETSC_DIR `pwd`
```

3. Choose system type and optimization level:

```
make PETSC_ARCH=os-type BOPT=O_c++
```

where *os-type* is one of `IRIX` (for a 32-bit Irix build) or `IRIX64` (for a 64 bit Irix build).

4. Configure PETSc:

```
./config/configure.py --with-mpi=0 --with-blas-lapack-dir=blas-lapack-lib-path
```

for example:

```
./config/configure.py --with-mpi=0 --with-blas-lapack-dir=/usr/local/lib
```

Option `--with-blas-lapack-dir` applies if ATLAS (rather than BLAS) is used.

⁶ <http://www-unix.mcs.anl.gov/petsc/petsc-2/documentation/installation.html>

5. Build PETSc:

```
make all
make test
```

A.2.4. SCIRun PETSc Configure Options

The configure options below add support for PETSc in SCIRun. Include these options in the SCIRun **configure** command (see Section 2.4, “Building SCIRun”).

<code>--with-unipetsc=</code> <i>path-to-petsc</i>	Specifies the path to the PETSc directory.
<code>--with-blas=</code> <i>path-to-blas-libs</i>	Specifies the path to the BLAS libraries. Not needed if using a vendor supplied library or if the BLAS RPM was installed.
<code>--with-atlas=</code> <i>path-to-atlas-libs</i>	Specifies the path to the ATLAS libraries. Not needed if using a vendor supplied library.
<code>--with-lapack=</code> <i>path-to-lapack-libs</i>	Specifies the path to the LAPACK libraries. Not needed if using a vendor supplied library or if the LAPACK RPM was installed.

Here is an example SCIRun **configure** command:

```
../src/configure \
--enable-package="BioPSE MatlabInterface Teem" \
--with-unipetsc=/usr/local/petsc \
--with-thirdparty=/usr/local/SCIRun/Thirdparty/1.22.0/Irix/CC-32bit \
--with-blas=/usr/local/lib \
--with-lapack=/usr/local/lib
```

A.2.5. The SolveMatrix Module

When SCIRun is built with PETSc support, the SolveMatrix module allows the user choose to from one of 12 PETSc solvers.

SolveMatrix passes a copy of matrix data on its input ports to a call of PETSc's `SLESSolve` function. SolveMatrix charts the progress of the solution while PETSc iterates.

A.3. HDF5 Installation

The Hierarchical Data Format (HDF)⁷ project provides file formats and software for scientific data management. HDF5 libraries provide modules of SCIRun's DataIO package the ability to read HDF5 data.

HDF5 libraries must be built from their source code. SCIRun supports HDF5 versions 1.6.x.

Note

⁷ <http://hdf.ncsa.uiuc.edu/HDF5>

- HDF5 must be installed before building SCIRun
- HDF5 must be built thread safe—HDF5 binary distributions that are not thread safe cannot be used with SCIRun.
- SCIRun must be configured to support HDF5. See Section A.3.3, “SCIRun HDF5 Configure Options” for information on configuring SCIRun to use HDF5.

A.3.1. Download and Unpack

Download and unpack the HDF5 source code distribution:

- Web browser download:

```
ftp://ftp.ncsa.uiuc.edu/HDF/HDF5/hdf5-1.6.4/src/hdf5-1.6.4.tar.gz
```

- Unpack. In a terminal window type:

```
gunzip -c hdf5-1.6.4.tar.gz | tar xvf -
```

A.3.2. Building

Follow the steps below to build HDF5. See the HDF5 installation home page⁸ for complete installation instructions. To build HDF5:

- Change the working directory to the HDF5 directory:

```
cd path-to/hdf5-1.6.3
```

- Set compiler environment variables, for example,

For an sh-type shell:

```
CC=gcc  
CXX=g++  
export CC CXX
```

For an csh-type shell:

```
setenv CC gcc
```

⁸ ftp://ftp.ncsa.uiuc.edu/HDF/HDF5/current/src/unpacked/release_docs/INSTALL

```
setenv CXX g++
```

- Configure HDF5:

```
./configure --enable-threadsafe --with-pthread=/usr --prefix=/usr/local/hdf5
```

This will install HDF5 binaries, libraries, header files, and tools in /usr/local/hdf5.

- Build HDF5:

```
make all
```

Become root user and install HDF5:

```
su -  
make install
```

Leave the root account by typing **exit**.

A.3.3. SCIRun HDF5 Configure Options

The configure option below adds support for HDF5 in SCIRun. Include this option in the SCIRun **configure** command (Section 2.4, “Building SCIRun”):

```
--with-hdf5=/usr/local/hdf5
```

A.4. MDSplus Installation

MDSplus libraries gives modules of SCIRun's DataIO package the ability to read MDSplus data from an MDSplus server.

MDSplus software is installed from its source code.

Building MDSplus to work properly with SCIRun is somewhat involved. Contact Allen Sanderson (<allen@sci.utah.edu>) at the SCI institute for more information.

A.4.1. MDSplus Configure Options

For a source code build of SCIRun, include the following option to SCIRun's **configure** command to add support for MDSplus:

```
--with-mdsplus=/usr/local/mdsplus
```

B. Single Source, Multiple Binaries

It is possible to build SCIRun for multiple machine architectures and multiple sets of configure options using the same source code directory.

For instance, if the SCIRun source code directory resides on a shared volume then Linux, 32 bit and 64 bit Irix, and Mac OSX versions can be built as follows:

```
mkdir SCIRun/linux
mkdir SCIRun/irix32
mkdir SCIRun/irix64
mkdir SCIRun/osx
```

After logging onto the Linux machine:

```
cd SCIRun/linux
../src/configure options
make -j N
```

Then from the Irix machine(s):

```
cd SCIRun/irix32
../src/configure options
make -j N
cd SCIRun/irix64
../src/configure --enable-64bit options
make -j N
```

and the Mac OSX platform(s):

```
cd SCIRun/osx
../src/configure options
make -j N
```

It is possible to build SCIRun for different sets of configure options. For example, a debug version for Mac OSX can be built as follows:

```
mkdir SCIRun/osx-debug
cd SCIRun/osx-debug
../src/configure --enable-debug more-options
make -j N
```

C. Local SCIRun Source Code Control Using CVS

C.1. Introduction

The following is a brief description of Northeastern University's use of CVS to manage local SCIRun software development.

We are treating SCI Institute as a software vendor and using CVS's built-in capability to track third-party sources in parallel with the modifications we make to that source. As we download releases from SCI Institute, we use the **import** command to check the SCIRun/BioPSE source code and documentation onto a vendor branch in our local repository. CVS automatically establishes a main trunk, along which development versions of files get stored. Initially, files are "copied" from the vendor branch to the main trunk. As we check out files, make modifications, and/or add new files, and commit these changes, our modifications appear as new versions of these files on the main trunk. When a new release of SCIRun/BioPSE becomes available, we use import again to store the new code on the vendor branch. If a file has been both locally modified and changed in the new release, CVS gives a warning that the file from the new release must be merged into the main trunk. The **checkout -j** command is useful for doing this merge and is probably best issued on a file-by-file basis. If a file has been locally modified but not changed by the new release, no change is required; the locally modified file remains unchanged on the main trunk. If a file has not been locally modified, but changes from one release to the next, the new release revision is automatically "copied" to the main trunk. CVS reserves the tag "HEAD" to identify the latest revision of a file in the repository, which in our case will probably always be on the main trunk.

The CVS manual is quite useful and available at www.cvshome.org¹ along with the CVS downloads. The cvs manpage is also very informative.

C.2. Importing a Release

This section describes a procedure for downloading a release, extracting files, and importing files into the CVS repository

This procedure should be followed once for each release in order to import the released source code into the CVS repository. No compilation is done, therefore the result is not a working tree. See Section C.3, "Creating a Working Scirun Directory Tree" to create a working directory.

1. Set environment variable:

```
setenv CVSROOT /proj/cdsp/biomed2/cvsSCIRun
```

2. Create directory to store tar files. It is suggested to move away from roland and begin storing these files on the ECE file system, eg., /proj/cdsp/biomed2/SCIRunDownloads/v1.6.0_tars.
3. Download zipped tar files from http://software.sci.utah.edu/archive_entry.html.² A username, which is an email address, and a password may be needed to access this page.
4. Use gunzip to inflate all zip files.
5. Extract source code files for import into the CVS repository. For example:
 - a. Create a directory on an appropriate file system: SCIRun.1.6.0 and **cd** into it
 - b.

```
tar xvf /proj/cdsp/biomed2/SCIRunDownloads/v1.6.0_tars/SCIRun.1.6.0
```

¹ <http://www.cvshome.org/docs/manual/>

² http://software.sci.utah.edu/archive_entry.html

c.

```
tar xvf /proj/cdsp/biomed2/SCIRunDownloads/v1.6.0_tars/docs.1.6.0.tar
```

d.

```
cd ./SCIRun/src/Packages
```

e.

```
tar xvf \  
/proj/cdsp/biomed2/SCIRunDownloads/v1.6.0_tars/BioPSE.PKG.1.6.0.tar
```

f.

```
tar xvf \  
/proj/cdsp/biomed2/SCIRunDownloads/v1.6.0_tars/MatlabInterface.PKG.1.6.0.tar
```

6. Import source files into CVS repository. Run the following command in the ./SCIRun directory.

```
cvs import -m"Import of release 1.6.0" /proj/cdsp/biomed2/cvsSCIRun  
SCI_DIST V1_6_0 > cvsimport.out
```

Redirect output to a file in order to reliably identify any conflicts.

Note

Files that have been “retired” in the new release, need to be removed and committed individually (or en masse). Use the following commands to remove retired files:

```
cvs rm -f `cat remove.txt`  
cvs commit -m"sync removed files" `cat remove.txt`
```

remove.txt is a space-delimited list of retired files. The cvs **rm** command marks files for removal; the **commit** command modifies the database, implementing the removal. The files remain in the archive and can still be retrieved at a later time, but they will not be automatically retrieved as part of a global checkout.

Tip

The **add** and **rm** commands are similar in that they schedule an action to take place in the repository, but do not actually modify the repository themselves. The **commit** command implements the action scheduled by **add** or **rm**. This is in contrast to other cvs commands, such as **checkout** and **checkin**, that take their own action.

7. Resolve any conflicts that occur between the imported files and locally modified files:

```
cvs checkout -jSCI_DIST:yesterday -jSCI_DIST filename
```

(see Section 13 of the CVS manual).

C.3. Creating a Working Scirun Directory Tree

This procedure assumes the latest SCIRun/BioPSE tar files have already been downloaded, and the source code has been imported, and merged if necessary, into the repository. The following steps build a working directory by extracting the Thirdparty software, DataSets, and documentation from tar files and obtaining the SCIRun/BioPSE source code from the CVS repository. This procedure also assumes, for the present, that the BioPSE and MatlabInterface packages are desired in the working directory, and no other packages have been imported into the repository. Under this model, it is expected that each developer will be writing and compiling code in his/her own working directory, independently of other developers. Periodically, all developers will check their work in (commit) to the common repository.

1.

```
setenv CVSROOT /proj/cdsp/biomed2/cvsSCIRun
```

or to retrieve source code via ssh:

```
setenv CVSROOT gateway.ece.neu.edu:/proj/cdsp/biomed2/cvsSCIRun
setenv CVS_RSH ssh
```

2. Create a top-level SCIRun directory. This can be anything because it is a personal working directory, eg., /usr/people/scirun/SCIRun.1.6.0.lk or /export/scratch/SCIRun.1.6.0.test.

3. **cd** into this directory

4. Assuming unzipped tar files reside in /proj/cdsp/biomed2/SCIRunDownloads/vx.x.x_tars, issue the following commands (version numbers given as an example):

a.

```
tar xvf \
/proj/cdsp/biomed2/SCIRunDownloads /v1.6.0_tars/Thirdparty_install.1.6.0
```

b.

```
tar xvf /proj/cdsp/biomed2/SCIRunDownloads /v1.6.0_tars/DataSets.1.6.0.t
```

c.

```
tar xvf /proj/cdsp/biomed2/SCIRunDownloads /v1.6.0_tars/docs.1.6.0.tar
```

5. **cd** into the Thirdparty directory created in the previous step, eg.,

```
cd /export/scratch/SCIRun.1.6.0.test/Thirdparty_install.1.4.3.
```

Note

The Thirdparty software distributed with v1.6.0 is unchanged from v1.4.2. This is why the numbers 1.4.3 and 1.4.2 appear.

Run the install script:

```
python install INSTALLDIR [irix, Linux] [sgi, gcc] [32, 64]
```

where *INSTALLDIR* is the top-level working directory, eg., `/export/scratch/SCIRun.1.6.0.test`, and brackets indicate lists of options for each argument.

This creates a 1.4.2 directory (because the thirdparty software has not changed with v1.6.0) under *INSTALLDIR* containing all the necessary thirdparty software libraries.

The previous 5 steps take care of the SCIRun installation that is independent of CVS. The remaining steps retrieve the SCIRun/BioPSE source code that has been imported to the repository, and build the SCIRun program.

6. **cd** back to *INSTALLDIR*
7. Create a SCIRun directory below *INSTALLDIR*
8. **cd** into SCIRun
9. Enter the following command to checkout the HEAD revision of the SCIRun/BioPSE source code files. Note that the period at the end of the line below is an essential part of the command.

```
cvs checkout .
```

This command creates and populates the src and doc directories below `./SCIRun`.

10. Create an sgi32 or linux (or appropriate other) directory below the `./SCIRun` directory, and **cd** into this directory.
11. Run configure as follows (eg.):

```
../src/configure
--with-thirdparty=/export/scratch/SCIRun.1.6.0.test/1.4.2/Linux/gcc-32bit
--enable-package='BioPSE MatlabInterface'--enable-debug
```

If configure does not run successfully under sgi, try setting the following environment variables and repeat the configure command.

```
setenv CC cc
setenv CXX CC
setenv F77 f77
```

12. Compile SCIRun/BioPSE:

```
gmake -j JOBNUMBER
```

where 5 is a typical value for *JOBNUMBER*

SCIRun may appear to compile successfully, but run-time errors may occur if environment variables starting with `LD_LIBRARY` were set prior to compilation. Therefore, it is recommended to unset these environment variables when compiling and running SCIRun, e.g.,

```
unsetenv LD_LIBRARY_PATH
```

C.4. Common CVS Commands

CVS commands are issued by typing `cv`s, followed by a list of global options (if any), followed by the desired cvs command name, followed by a list of options specific to that command (if any). Below are some commonly used commands and examples of how they are invoked. A CVS command can apply to a single file, a directory or the whole repository.

Tip

If entering `cv`s alone results in a "command not found" error, type `/usr/bin/cvs`.

checkout	Obtain an editable version of a file(s) from the repository. Can be abbreviated as co .
cv s co .	Checkout the entire tree; invoke where the <code>src</code> and <code>doc</code> directories are to be created
cv s co src/ Core/Datatypes/TriSurfMesh.cc	Checkout the HEAD (the tip of the tree) revision of a specific file; invoke at the top of the working tree, where <code>src</code> and <code>doc</code> subdirectories reside or where they are to be created. Invoking this command in the subdirectory where the file resides, or without giving the path specification as indicated, will not work.
cv s co -rV1_4_2 src/ Core/Datatypes/TriSurfMesh.cc	Checkout a tagged revision of a specific file; invoke at the top of the working tree, where <code>src</code> and <code>doc</code> subdirectories reside, or where they are to be created.
update	Bring the working directory up to date with the repository. This command replaces files in the working directory if a more recent version of the file exists in the repository and the

local copy is unchanged. If the repository contains a revision more recent than the ancestor of the local copy, and the local copy has been modified, this command attempts to merge the latest revision in the repository with the local copy. It is a good idea to first run this command with the `-n` global option. Running update with the `-n` option causes cvs to tell you what it would do without modifying any files. It is best to issue this command by piping its output to a file so that all files in which a conflict occurred can be reliably identified. If conflicts arise, cvs indicates this with a 'C' in the output. Running update prior to checking in files is strongly recommended to ensure that someone else's changes are not over-written.

cvs -n update -A -P -d > cvsupdate.out

Find out what cvs would do if the update command were issued; invoke in the top level directory (where src and doc reside), and redirect output to a file. Option `-A` resets any sticky tags, `-P` prunes empty directories, and `-d` creates any directories that exist in the repository if they are missing from the working directory.

cvs update -A -P -d

Do the update for real. Options same as above. The output of this command is:

```
RCS file: /proj/cdsp/biomed2/cvsSCIRun/src/Core/Da
retrieving revision 1.2
retrieving revision 1.3
Merging differences between 1.2 and 1.3 into TriSu
C Core/Datatypes/TriSurfMesh.cc
RCS file: /proj/cdsp/biomed2/cvsSCIRun/src/Core/Da
retrieving revision 1.2
retrieving revision 1.4
Merging differences between 1.2 and 1.4 into TriSu
M Core/Datatypes/TriSurfMesh.h
U Core/Datatypes/TypeName.h
? Core/Datatypes/TriSurfMesh.cc.1.2
```

The character at the beginning of certain lines indicates the result of the update. The "C" indicates that cvs tried to merge two revisions and encountered conflicts, which are highlighted by the standard diff demarcations. Manual resolution is required. "M" means the files were merged successfully, "U" means the file is unchanged, and "?" means there is no record of the file in the repository.

add

To schedule newly created file(s) to be added to the repository, there must be a working copy of the new file. The add command is used to tell CVS about the new file (see the manual for instructions on adding new directories). The add command must be invoked from the directory in which the new file exists. Then run the commit command to check the file into the repository. For example:

```
cvs add filename
```

commit

Incorporate changes from the working directory into the repository. **commit** is abbreviated as **ci**.

Tip

When checking in files, cvs will prompt for comments using the vi editor, unless otherwise specified. To avoid vi, set the EDITOR environment variable to point to an editor. For example:

```
setenv EDITOR emacs
```

The command:

```
cvs ci TriSurfMesh.cc
```

commits the working version of a file, making it the HEAD (tip of the tree) revision; typically invoked in the directory where the file resides without any path specification (not sure if it can be run any other way). CVS will automatically assign the next internal revision number to the file upon checkin.

The -m option can be used to avoid being put into an editor:

```
cvs ci -m"this is my checkin comment" TriSurfMesh.cc
```

rdiff Create a patch file for use by SCI to incorporate changes made locally into the next official release. See manual for details.

log View log information about a file, such as revision numbers and check in comments. Most conveniently invoked in the directory in which the file of interest resides, e.g.:

```
cvs log TriSurfMesh.cc
```
