SLIC From Scratch on Linux

Page is Deprecated

This page contains old instructions that are kept for reference purposes only. Please refer to the Simulation Software Distribution page for a much easier way of building slic and its dependencies.

The SLIC full simulator program requires the setup of 8 different software packages, not counting the required build tools.

This guide provides a step-by-step walkthrough covering package and tool installation.

Preliminaries

Tools

You need the following command line tools.

- cvs
- g++ (**gcc** 3.2 and 3.3 seem to work okay. I doubt that 3.4 works.)
- make
- gzip
- tar (that's the GNU version)

If you are missing one of these...are you sure that you're running Linux?

Any recent versions of these programs should do okay.

You also need a Java VM, for running ant.

- java (to run ant)

Also, wget is mighty useful for retrieving files via http.

Work Area

The SLIC package and its dependencies will be installed into a common work area.

1. From the shell, create a work directory where you have a lot of space and go into it.
```
cd ~
mkdir sim
cd sim
```
2. Create the file **setup.sh** with the following contents.
```
#!/bin/sh
export sim_work=~/sim
```
3. Source the script to setup the work dir.
```
source setup.sh
```

The **$sim_work/setup.sh** script will henceforth be referred to as **setup.sh**. At the end, it will have all of the environment variables required by SLIC and its dependencies. Throughout the guide, any time a line is added to **setup.sh**, you should also execute this line in your current **bash** shell. Probably the easiest way to do this is by adding to the script first and then (re)sourcing it.

Package Installations

You are now ready to install the simulation packages.

Possibly some of these software packages already exist on your system.
You still need to setup the environment accordingly. Scan over the package installation instructions to see what environment variables need to be defined. Usually, this is one variable per package (with the exception of Geant4, which has many).

The same shell window should be used throughout the installation process in order to preserve the environment variables.

Package Versions

Currently, the SLIC, LCDD, LCIO, GDML, and LCPhys CVS heads should all work fine together.

The following is a recent, tested configuration.

```
slic   v1r12p0
lcio   v1r5p0_slic
lcdd   v1r8p2
lcphys v1r0p1
gdml   2.3.0
geant4 7.1.0
clhep  1.9.2.1
xerces 2.6.0
```

The Geant4 version must be **7.1.x** or SLIC will not compile.

CLHEP

CLHEP has installation instructions (http://wwwasd.web.cern.ch/wwwasd/lhc\+\+/clhep/INSTALLATION/newCLHEP-install.html) for version 1.9 and up. But you should not need them to setup the package.

1. Create a working directory for CLHEP and go into it.
   ```
   mkdir clhep
   cd clhep
   ```
2. Download the CLHEP tarball.
   ```
   wget http://cern.ch/clhep/clhep-1.9.2.0.tgz
   ```
3. Unzip to your work directory.
   ```
   tar -zxvf clhep-1.9.2.0.tgz
   ```
4. Change to CLHEP directory.
   ```
   cd 1.9.2.0/CLHEP
   ```
5. Configure the build. (Now go get some coffee.)
   ```
   ./configure --prefix=`cd ../..; pwd` --disable-shared
   ```
6. Build the library and install it. (Get some more coffee.)
   ```
   make
   make install
   ```
7. Add the following to **setup.sh**

   ```
   export CLHEP_BASE_DIR=$sim_work/clhep
   ```

Now that the CLHEP dependency is satisfied, you should be able to install Geant4.

Geant4

Geant4 is probably the most difficult application to install of SLIC's dependencies, because there are a lot of options, it takes a long time, and it requires several different **make** commands. I will describe a minimal installation procedure with support for built-in UI and visualization drivers. You can always update the libraries later if you decide to change these settings.

Due to limitations of GDML, Geant4 must be compiled with granular libraries. This should be fixed soon.

These procedures are geared towards setting up a **minimal** Geant4 installation for SLIC. If you want a full Geant4 setup, I recommend running the **Configure** script. It will walk you through setting up a complete enviroment.

1. Return to the work dir, create a Geant4 subdir and go into it.

```
cd $sim_work
mkdir geant4
cd geant4
```

2. Download the Geant4 tarball.

```
wget http://geant4.cern.ch/geant4/source/source/geant4.7.1.tar.gz
```

3. Unzip it.

```
tar -zxvf geant4.7.0.p01.tar.gz
```

4. Set the following variables in **setup.sh**. (You should not need to run **Configure**.)

```
export G4INSTALL=${sim_work}/geant4/geant4.7.1
export G4SYSTEM=Linux-g++
export G4LIB_USE_GRANULAR=1
```

5. If you want to enable OpenGL-based visualization, set these variables, too.

```
export G4VIS_BUILD_OPENGLX_DRIVER=1
export G4VIS_USE_OPENGLX=1
export OGLHOME=/usr
```

6. Refer to the Application Guide Section on Enviroment Variables for additional enviroment variables that you might want to set.
7. Go into the Geant4 base dir.

```
cd geant4.7.1
```

8. LCPhys requires that a special flag is set in order to use the latest Kaon model. At the end of **config/architecture.gmk**, insert the following line **ex actly as it appears below**:

```
CPPFLAGS += -DG4BERTINI_KAON
```

   Hopefully, this hack will be remedied soon!
9. Build the libraries, which will be placed at **$G4INSTALL/lib/Linux-g++**. (This could take up to a few hours depending on your machine.)

```
cd source
make
```

10. Install the headers into the **$G4INSTALL/include** directory.

```
make includes
```

11. Build the physics list libraries. These will go into the default location at **$G4INSTALL/lib/plists/Linux-g++**.

```
cd ../physics_lists/hadronic
make
```

Hopefully, Geant4 has been installed successfully, and you don't have too many more gray hairs! (Did you remember to set **G4BERTINI_KAON**? LCPhys won't work without it.)

LCPhys

SLIC requires a special physics list written by Dennis Wright for Linear Collider physics.

1. Go back to the work dir.
   ```
   cd $sim_work
   ```
2. Checkout the physics list from CVS.
   ```
   cvs -d :pserver:anonymous@cvs.freehep.org:/cvs/lcd checkout LCPhys
   ```
3. Assuming that the environment from the Geant4 installation is still in place, you can build this like any other physics list, and the library should be installed into **$G4INSTALL/lib/plists/WIN32-g++**.
   ```
   cd LCPhys
   make
   ```
4. Set the LCPhys variable in **setup.sh**.
   ```
   LCPHYS_BASE=$sim_work/LCPhys
   ```

LCIO

LCIO provides binary output capabilities.

> Installation requires a working Java runtime for **ant** support.

LCIO has a very nice manual with a whole section on installation. Thanks, Frank!

I will still walk you through the basic procedure.

1. Go back to the work dir.
   ```
   cd $sim_work
   ```
2. Checkout LCIO from CVS.
   ```
   cvs -d :pserver:anonymous@cvs.freehep.org:/cvs/lcio checkout lcio
   ```
3. Add these lines to your **setup.sh**.
   ```
   export LCIO=${sim_work}/lcio
   export PATH=$LCIO/tools:$LCIO/bin:$PATH
   ```
4. Build the libraries using the bundled **aid** and **ant** tools.
   ```
   cd $sim_work/lcio
   ant aid.generate cpp
   ```

Xerces-C++

1. Go back to the work dir, create a subdir for Xerces-C++, and go into it.

```
cd $sim_work
mkdir xercesc
cd xercesc
```

2. Download the Xerces tarball.

```
wget http://www.apache.org/dist/xml/xerces-c/xerces-c-src_2_6_0.tar.gz
```

3. Unzip the tarball.

```
tar -zxvf xerces-c-src_2_6_0.tar.gz
```

4. Set **XERCESCROOT** for the build in your environment, only.

```
export XERCESCROOT=${sim_work}/xercesc/xerces-c-src_2_6_0
```

5. Go into the Xerces-C++ build area.

```
cd xerces-c-src_2_6_0/src/xercesc
```

6. Configure the build.

```
runConfigure -plinux -cgcc -xg++ -minmem -nsocket -tnative -rpthread -P `cd ../../..; pwd`
```

7. Build and install it.

```
make
make install
```

8. In **setup.sh**, set **XERCESCROOT** to the installation area and add the DLL location to the **PATH**.

```
export XERCESCROOT=${sim_work}/xercesc
export LD_LIBRARY_PATH=$XERCESCROOT/lib:$LD_LIBRARY_PATH
```

When rebuilding Xerces-C++, which you will probably not need to do once you get it working, **XERCESCROOT** needs to be set back to the Xerces-C++ source area rather than the installation base.

GDML

GDML's CVS is not directly accessible from the command line, but a tarball is available through a WWW interface.

1. Download a snapshot of the current CVS head to **sim_wrk** using this link in your browser: http://simu.cvs.cern.ch/cgi-bin/simu.cgi/simu/GDML2/GDML2.tar.gz?tarball=1.
2. Unzip the tarball.

```
tar -zxvf GDML2.tar.gz
```

3. Change into the CPPGDML directory.

```
cd GDML2/CPPGDML
```

4. Set **GDML_BASE** and **PLATFORM** in **setup.sh**.

```
export GDML_BASE=${sim_work}/GDML2/CPPGDML
```

5. Configure the build.

```
./configure --enable-shared-libs=no
```

6. Build it.

```
make
```

LCDD

1. Go to the work dir and checkout LCDD.

```
cd ${sim_work}
cvs -d :pserver:anonymous@cvs.freehep.org:/cvs/lcd checkout lcdd
```

2. Go into the LCDD dir.

```
cd lcdd
```

3. Configure the build.

```
./configure
```

4. Build the library.

```
make
```

5. Set the **LCDD_BASE** variable in **setup.sh**.

```
export LCDD_BASE=${sim_work}/lcdd
```

SLIC

Finally, you are ready to install the simulation "hub" package. After this, you will have a fully-featured Geant4 simulator on your Linux machine.

1. Go to the work dir and checkout SLIC.

```
cd ${sim_work}
cvs -d :pserver:anonymous@cvs.freehep.org:/cvs/lcd checkout slic
```

2. Go into the SLIC dir.

```
cd slic
```

3. Set the **SLIC_BASE** variable in **setup.sh**.

```
export SLIC_BASE=${sim_work}/slic
```

4. Configure the build.

```
./configure
```

5. Build the binary.

```
make all
```

If the build completes successfully, you should see SLIC's usage statement from the test run.

Running SLIC after Installation

When you want to run later in a Cygwin shell, **$XERCESCROOT/bin** should be in the **PATH**, so that Windows can find the DLL at runtime. Since the other applications were linked-in statically, this should be the only setup requirement.

This is the procedure for running SLIC from the Cygwin commandline.

1. Open a bash shell.
2. Add Xerces-C++ bin to the load path.
   ```
   export LD_LIBRARY_PATH=$XERCESCROOT/lib
   ```
3. Go to the SLIC directory.
   ```
   cd ~/sim/slic
   ```
4. Run the executable.
   ```
   bin/Linux-g++/slic [options]
   ```

If you receive an error message about a missing Xerces library, then make sure that the **PATH** is setup correctly and Xerces-C++ was properly installed.

Final Setup Script

The final version of **setup.sh** should be similar to this.

```
#!/bin/sh

# work area
export sim_work=~/sim

# clhep installation area
export CLHEP_BASE_DIR=${sim_work}/clhep

# geant4
export G4INSTALL=${sim_work}/geant4/geant4.7.1
export G4SYSTEM=Linux-g++

# LCPhys
export LCPHYS_BASE=${sim_work}/LCPhys

# LCIO
export LCIO=${sim_work}/lcio

# if rebuilding LCIO
export PATH=$LCIO/tools:$LCIO/bin:$PATH

# Xerces-C++ installation area
export XERCESCROOT=${sim_work}/xercesc
export LD_LIBRARY_PATH=$XERCESCROOT/lib:$LD_LIBRARY_PATH

# GDML
export GDML_BASE=${sim_work}/GDML2/CPPGDML

# LCDD
export LCDD_BASE=${sim_work}/lcdd

# SLIC
export SLIC_BASE=${sim_work}/slic
```

The above should be sufficient to "bootstrap" the environment for any future (re)builds of SLIC dependencies.

Running SLIC

To run SLIC once it is built, simply add the Xerces library location to your load path.

Then you can execute the binary from the **SLIC_BASE** directory.

```
export LD_LIBRARY_PATH=$XERCESCROOT/lib:$LD_LIBRARY_PATH
bin/Linux-g++/slic [args]
```

SLIC Run Script

You should create a run script for your site, so that simply typing "slic" from the command line executes the simulator. A sample run script is found in **$SLIC_BASE/scripts/run.sh**.

Done.

That's it.

If you think this guide could be improved in any way, then please contact the author

Someday, I will get around to packaging all of this as RPMs to save everyone the hassle.

Happy simulating...

Additional Resources

SLIC Homepage
Running SLIC at SLAC