SLIC from Scratch on Windows

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.

It presumes nothing except a working Windows machine (2000, XP) with internet access. Therefore, you may only need to use parts of this installation guide if you have some of the external dependencies already installed.

Standalone Windows Distribution

Cygwin needs to be installed to run SLIC.

Download the SLIC Windows binary from this location.

http://www.lcsim.org/dist/slic/slic-current-WIN32-g%2B%2B-bin.tar.gz

Open a Cygwin window and go to the directory where you saved the tarball.

Untar the file after it is saved to your computer.

```
tar -zxvf slic-current-WIN32-g++-bin.tar.gz
```

Now test the binary.

```
cd SimDist
./scripts/slic.sh
```

The slic usage screen should show.

Preliminary Setup for Installation

Cygwin

The Cygwin Linux emulation package is a prerequisite for building on Windows.

Unfortunately, the support for native WIN32 using project files is lacking.

These are brief instructions for installing the necessary Cygwin packages.

1. Download the Cygwin setup program.
2. Double-click on it and click **Next**.
3. Select **Install from Internet** and click **Next**.
4. Enter your preferred **Root Directory** and click **Next**.
5. Enter your preferred **Local Package Directory**, which can be the same as the **Root Directory**, and then click **Next**.
6. Select **Direct Connection** and click **Next**.
7. Select a site from the **Available Download Sites**. Servers inside your country will probably be fastest. I use ftp://ftp.sunsite.utk.edu.
8. Click **Next**.
9. In the **Cygwin Setup - Select Packages** window, you need to make sure that the following tools are selected by clicking in the corresponding box under the **New** column until you see a version number.
    a. Required packages.
        - Devel -> cvs
        - Devel -> gcc-core
        - Devel -> gcc-g++
        - Devel -> make
        - Base -> gzip
        - Base -> tar
    b. If OpenGL visualization is being used, these should also be installed.
        - X11 -> ALL
        - Graphics -> OpenGL
    c. Tool for downloading package tarballs and zip files.
        - Web -> wget
10. Click **Next** after you have selected the packages.
11. Cygwin will now automatically download and install all the selected packages. It might take awhile, so now is the time to go get some coffee.

12. If desired, select **Create icon on Desktop** or **Add icon to Start Menu**, and click **Finish**.

> **Cygwin Packages**
>
> It may be easier to simply install all Cygwin packages instead of selecting them individually.

If you need additional information on this installation process, Norman Graf has more detailed Cygwin installation instructions.

Testing the Cygwin Command Line Tools

1. Select **Start -> Programs -> Cygwin -> Cygwin Bash Shell**
2. Check that the following commands do not result in a *command not found* message.

```
cvs
gcc
g++
tar
wget
make
```

3. If a command was not found, rerun the Cygwin setup to select the missing package, making sure to select **Keep** on the **Cygwin Setup - Select Packages** screen so that all the packages are not reinstalled.

> Throughout this guide, I assume you are using **bash** or at least another **sh**-compatible shell. I take no responsibility if you decide to use **csh**, **tcsh**, et al.

Work Area

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

1. From the Cygwin shell, create a work directory and go into it.

```
cd /cygdrive/c
mkdir sim
cd sim
```

2. Create the file **setup.sh** with the following contents.

```
#!/bin/sh
export sim_work=/cygdrive/c/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.

Java

Java is required for LCIO installation.

1. Install an appropriate JDK from http://java.sun.com/ with a minimum version of 1.4.2.
2. In your **setup.sh**, set **JAVA_HOME** and **JDK_HOME** to the Java installation area.

```
export JAVA_HOME=/cygdrive/c/java/jdk1.4.2/
export JDK_HOME=${JAVA_HOME}
export PATH=$PATH:$JDK_HOME/bin
```

3. To test the Java installation, try to run the Java compiler.

```
javac
```

The above **JAVA_HOME** directory is an example only. You need to replace it with the correct path to your JDK.

Technically, installation of the LCIO C++ libraries only requires a Java JRE, but the JDK is useful for doing reconstruction and analysis where you may want to compile Java programs.

Proceed to Package Installation!

You are now ready to install the simulation packages.

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

Package Installations

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. (This took 15-20 minutes on my machine.)
```
./configure --prefix=`cd ../..; pwd` --disable-shared
```
6. Build the library and install it. (Also takes a long time!)
```
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.

> The default Geant4 library settings for WIN32 are granular and static. You may choose other settings, but this could require changes in installation settings "down the line" that I may not mention.

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 geant4.7.1.tar.gz
```

4. Set the following variables in **setup.sh**. (You should not need to run Geant4's **Configure** script.)

```
export G4INSTALL=${sim_work}/geant4/geant4.7.1
export G4SYSTEM=WIN32-g++
```

   If you would like to enable OpenGL visualization, add the following variable definitions.
   This assumes you installed the X11 libraries when you set up cygwin.

```
export OGLHOME=/usr/X11R6
export G4VIS_BUILD_OPENGLX_DRIVER=1
export G4VIS_USE_OPENGLX=1
```

5. Go into the Geant4 base dir.

```
cd geant4.7.1
```

6. 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!
7. Now for another lovely Geant4 hack. In order to successfully compile the HepRep driver, the following line in the file **config/sys/WIN32-g++.gmk**

```
CXXFLAGS  := -Wall -ansi -pedantic -pipe
```

   should be replaced with

```
CXXFLAGS  := -W -Wall -ansi -pedantic -Wno-non-virtual-dtor -Wno-long-long
CXXFLAGS  += -Wwrite-strings -Wpointer-arith -Woverloaded-virtual -pipe
```

8. Build the libraries, which will be placed at **$G4INSTALL/lib/WIN32-g++**. (This could take up to a few hours depending on your machine.)

```
cd source
make
```

9. Install the headers into **$G4INSTALL/include**.

```
make includes
```

10. Build the physics list libraries. These will go into **$G4INSTALL/lib/plists/WIN32-g++**.

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

Hopefully, Geant4 has been installed successfully, and you don't have too many more gray hairs.

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**.
```
export 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 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 -pcygwin -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=${sim_work}/xercesc
export PATH=$XERCESCROOT/bin:$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 using this link in your browser: http://simu.cvs.cern.ch/cgi-bin/simu.cgi/simu/GDML2/GDML2.tar.gz?tarball=1.
   - Save the tarball to **sim_wrk**, which should be **C:\sim**.
2. Unzip the tarball.
```
tar -zxvf GDML2.tar.gz
```
3. Change into the CPPGDML directory.

```
cd GDML2/CPPGDML
```

4. Set **GDML_BASE** 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 Windows 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 (note that we currently do not support visualization).

```
export G4VIS_NONE=1
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. Select **Start -> Programs -> Cygwin -> Cygwin Bash Shell**.
2. Add Xerces-C++ bin to the path.

```
export PATH=/cygdrive/c/sim/xerces/bin:$PATH
```

3. Go to the SLIC directory.

```
cd /cygdrive/c/sim/slic
```

4. Run the executable.

```
bin/WIN32-g++/slic [options]
```

If you receive an error message about a missing DLL **cygxerces-c26.dll**, then make sure that the **PATH** is setup correctly and Xerces-C++ was properly installed.

If you encounter difficulties running macros, it may be due to the different DOS and Unix end-of-line definitions. You can convert files using:

```
dos2unix mymacro.mac
```

In order to run with OpenGL visualization, an X-server needs to be running on your machine.

This command will start the Cygwin X-server.

```
startxwin.sh
```

Then you need to run SLIC from the xterm that pops-up. At least on my setup, it does not work to run the binary from the plain Cygwin window.

Final Setup Script

The final version of **setup.sh** (without OpenGL visualization) should be similar to this.

```
#!/bin/sh

# 1. work area
export sim_work=/cygdrive/c/work/nsim

# 2. java
export JAVA_HOME=/cygdrive/c/Java/jdk1.4.2
export JDK_HOME=${JAVA_HOME}
export PATH=${JAVA_HOME}/bin:${PATH}

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

# 4. geant4
export G4INSTALL=${sim_work}/geant4/geant4.7.0.p01
export G4SYSTEM=WIN32-g++

# 5. LCPhys
export LCPHYS_BASE=${sim_work}/LCPhys

# 6. LCIO
export LCIO=${sim_work}/lcio
export PATH=$LCIO/tools:$LCIO/bin:$PATH

# 7. Xerces-C++ installation area
export XERCESCROOT=${sim_work}/xercesc
export PATH=$XERCESCROOT/bin:$PATH

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

# 9. LCDD
export LCDD_BASE=${sim_work}/lcdd

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

The above should be sufficient to "bootstrap" any future (re)builds.

Done!

That's it.

Happy simulating!

See links in the next section for more information.

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

Links

- Geant4 on Windows with CL and Cygwin