Installing the Plugin project

Table of contents

Copyright © 2008 Hugh Leather All rights reserved.

1. How to install the Plugin project on your machine

This page takes you through the steps needed to get the plugin project downloaded and installed on your machine. It also describes how to build it from scratch.

Warning:

This project is still in alpha release. This, unfortunately, means that installing is not the simple, one-click process that it should be. Please bear with me.

2. Prerequisites

The plugin system works currently on Linux (though it will probably work on other Unixes and Cygwin). You will also need:

libiberty	Included in the GCC distribution
libffi	Included in the GCC distribution
libxml2	from xmlsoft or most package managers

3. Dowloading

The first step is to download the code. This can be done by getting the latest release from the SourceForge download site, here.

Note:

The current release is 0.1.0. It has only been tested on Linux and the current build has been created for x86 machines. If you have other requirements, please see section building on this page.

Once the package is downloaded, you will find the following structure in the file:

Directory/File	Description
libplugin	This is the main library which is used to build plugin applications. The directory contains both prebuilt shared and static libraries and header files for building applications and plugins
doc	Copy of this documentation
message	A plugin to make printing messages and logging easy. This plugin should work for any plugin application

Copyright © 2008 Hugh Leather All rights reserved.

gcc.patch	A patch to for GCC's trunk (as of version 4.4.0 20080422). This patch plugin enables GCC.
gcc.pass-manager	A plugin for GCC which allows you to respond to passes being executed and also to control what passes get executed
gcc.perfmon	A plugin for GCC. It provides simple instrumentation to get cycle counts.
gcc.print	A few printing functions, suitable for use with the message plugin
gcc.rtl-unroll-and-peel-loops	Provides control over RTL unrolling and peeling
gcc.toplev	A few extension points in GCC's toplev.c - giving control over the earliest stages of the compiler
gcc.print.passes.xml	A plugin which prints all the passes that are executed on your code
gcc.print.unrollable-loops.xml	A plugin which prints all the loops that can be unrolled
gcc.print.unrolled-loops.xml	A plugin which prints all the loops that *that are* unrolled

You should unpack these files somewhere.

If you aren't planning to use the GCC plugins then you are pretty much done. The prebuilt libraries should work for creating your own plugin applications. Have a look at the tutorials on this site for your next steps.

4. Installing for GCC

If you are intending to use the GCC plugins you will also need to download the source for GCC. You can find information about how to do that here. Check that you can build and install GCC by itself.

As quick crib, here's what I usually do to configure and install GCC:

```
svn -q checkout svn://gcc.gnu.org/svn/gcc/trunk my-gcc-dir
mkdir gcc.obj
mkdir gcc.bin
cd gcc.obj
my-gcc-dir/configure --prefix=path-to-gcc.bin/ --program-prefix=plugin-
--enable-languages=c
make && make install; beep
```


Copyright © 2008 Hugh Leather All rights reserved.

So, hopefully you now have your own GCC up and running. Now we have to apply the patch, *gcc.patch*, which is in the download distribution. This should be something like:

```
patch my-gcc-dir path-to-gcc.patch
```

Sadly, we aren't quite done yet. Embedded in the patch are a couple of nasty, hard-coded directories. Yuck. These will disappear once I read about auto tools and then the whole build/install process should become a one click process. Until then, well, "Sorry".

In your GCC source directory, go to *my-gcc-dir*/gcc/Makefile.in. On lines 278 and 279 there are two hard coded paths to */home/hleather/workspace/plugin/*. You need to replace these so that they point to where you unpacked the libplugin distribution.

With luck you should now be able to make and install GCC again:

```
cd gcc.obj
make && make install; beep
```

As a quick test, make a file, foo.c, somewhere, containing:

```
#include <stdio.h>

int main() {
        printf( "hello\n" );
        return 0;
}
```

And compile it with:

```
GCC_PLUGIN_LOGGING=true path-to-gcc.bin/bin/plugin-gcc foo.c
```

You should see something like:

```
/home/hleather/temp/gcc.bin/bin/../libexec/gcc/i686-pc-linux-gnu/4.4.0/cc1
- Log:Plugin system starting with path: (null)
/home/hleather/temp/gcc.bin/bin/../libexec/gcc/i686-pc-linux-gnu/4.4.0/cc1
- Log:Plugin system started
/home/hleather/temp/gcc.bin/bin/../libexec/gcc/i686-pc-linux-gnu/4.4.0/cc1
- Log:Plugin system stopping
/home/hleather/temp/gcc.bin/bin/../libexec/gcc/i686-pc-linux-gnu/4.4.0/cc1
- Log:Plugin system stopped
/home/hleather/temp/gcc.bin/bin/../libexec/gcc/i686-pc-linux-gnu/4.4.0/cc1
- Log:Plugin system cleaning up
/home/hleather/temp/gcc.bin/bin/../libexec/gcc/i686-pc-linux-gnu/4.4.0/cc1
- Log:Plugin system cleaned up
```


Copyright © 2008 Hugh Leather All rights reserved.

Which shows that the plugin system is working (no plugins were loaded because we didn't specify a plugin path).

Congratulations! I know that is all a bit painful. It will get better eventually, I promise. Your next steps should probably be to have a look at some of the tutorials on this site.

5. Building

Check out the source from source forge. You also need the GCC source built as above. You will need the Saxon 9 XSL processor and Java 1.6. Finally, to build the documentation you will need DOxygen and Apache Forrest.

Again, there are some nasty hard coded paths in the top level make file. They are only the ones at the top, though, the ones starting with '/home/hleather/'. Replace these with suitable paths on your system.

After that, apply the patch to GCC (as above).

Finally, you can (hopefully) build with *make*, or do the whole thing with *make dist*, which builds docs and a distributable zip as well.

Copyright © 2008 Hugh Leather All rights reserved.