**Eclipse环境调试Impala**

**注意：SVN提交代码时只提交自己修改过的文件，不要把编译生成的文件也提交，否则硬盘放不下。**

Impala分为前端和后端两个部分，前端使用Java编写，而后端则使用C++编写。通过配置Eclipse，可以同时调试前端和后端，为开发带来很大的便利。下面给出具体的配置过程。本文使用Eclipse版本为Luna-SR1，JDK版本需要1.7，若默认不符，需要在eclipse.ini起始处配置，如图1所示。

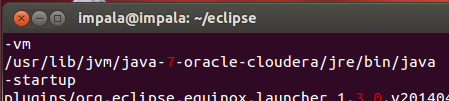


图1 配置eclipse.ini

在 htt调试前，需保证在命令行开启statestored和catalogd服务，详见《Impala 2.0安装手册》。

# Impala后端（BE）配置

## 在Eclipse中添加CDT

Help->Install New Software

在Work with中选择<http://download.eclipse.org/releases/luna>

将“Group items by category”选项取消，并在“type Filter text”中填入Eclipse IDE，安装Eclipse IDE for C/C++ Developers，如图2所示。

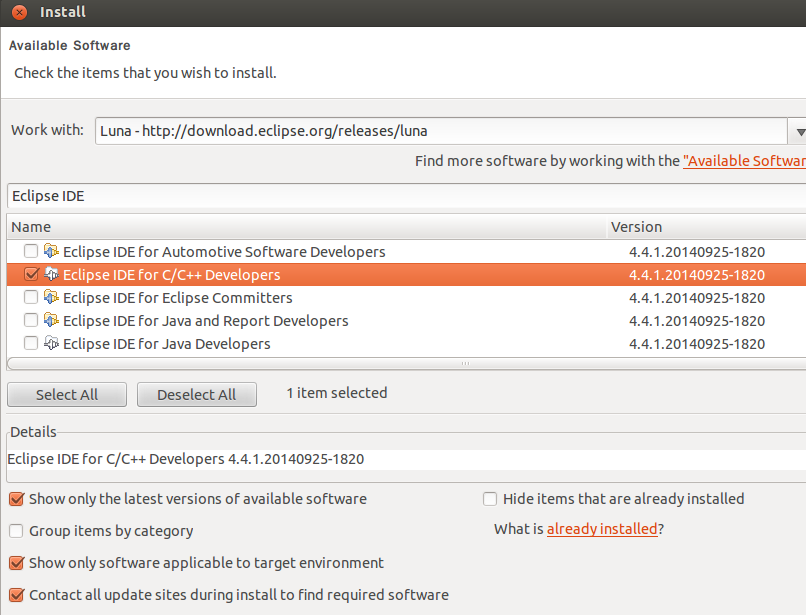


图2 安装CDT

## 在Eclipse中配置C/C++标准库

1. 在终端下:;jjkl查看C库文件路径

在终端输入如下命令，

$ echo "int main() { return 0; }" > a1.c

$ gcc -v a1.c

终端会显示：

……

#include <...> search starts here:

/usr/lib/gcc/x86\_64-linux-gnu/4.6/include

/usr/local/include

/usr/lib/gcc/x86\_64-linux-gnu/4.6/include-fixed

/usr/include/x86\_64-linux-gnu

/usr/include

End of search list.

……

1. 在终端下查看C++库文件路径

在终端输入如下命令，

$ g++ -v a1.c

终端会显示

……

#include <...> search starts here:

/usr/include/c++/4.6

/usr/include/c++/4.6/x86\_64-linux-gnu/.

/usr/include/c++/4.6/backward

/usr/lib/gcc/x86\_64-linux-gnu/4.6/include

/usr/local/include

/usr/lib/gcc/x86\_64-linux-gnu/4.6/include-fixed

/usr/include/x86\_64-linux-gnu

/usr/include

End of search list.

……

1. 在eclipse中菜单Window --> Preferences --> C/C++ --> Build --> Environment添加环境变量（a和b中的）：

|  |  |
| --- | --- |
| C\_INCLUDE\_PATH | /usr/lib/gcc/x86\_64-linux-gnu/4.6/include:/usr/local/include:/usr/lib/gcc/x86\_64-linux-gnu/4.6/include-fixed:/usr/include/x86\_64-linux-gnu:/usr/include |
| CPLUS\_INCLUDE\_PATH | /usr/include/c++/4.6:/usr/include/c++/4.6/x86\_64-linux-gnu/.:/usr/include/c++/4.6/backward:/usr/lib/gcc/x86\_64-linux-gnu/4.6/include:/usr/local/include:/usr/lib/gcc/x86\_64-linux-gnu/4.6/include-fixed:/usr/include/x86\_64-linux-gnu: /usr/include |

设置以后如图3所示。

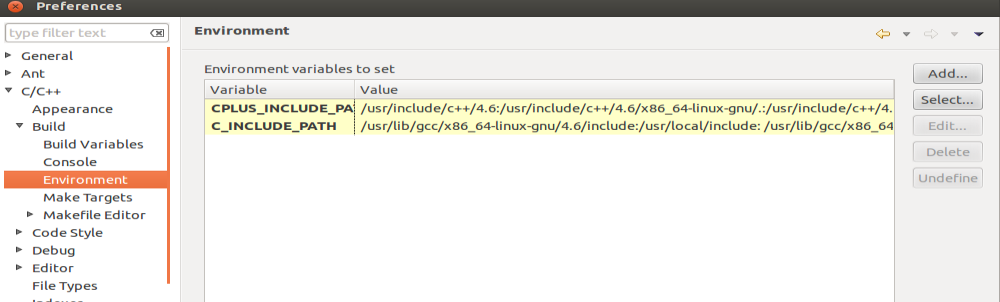


图3 配置C/C++标准库

## 导入已经编译好的Impala源码

该方法适用于第一次导入，若已放到github或svn等代码管理软件中，则只需check下来即可。

1. File->New->Project

选择C/C++中的MakeFile Project with Existing code，如图4所示。

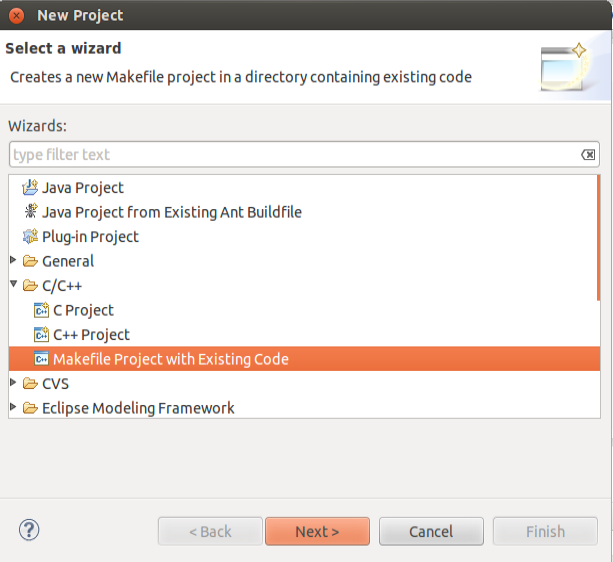


图4 导入Impala代码

1. 选择Next之后，进入如图5所示的界面。设置code location和project name。选择Finish,完成项目的导入。

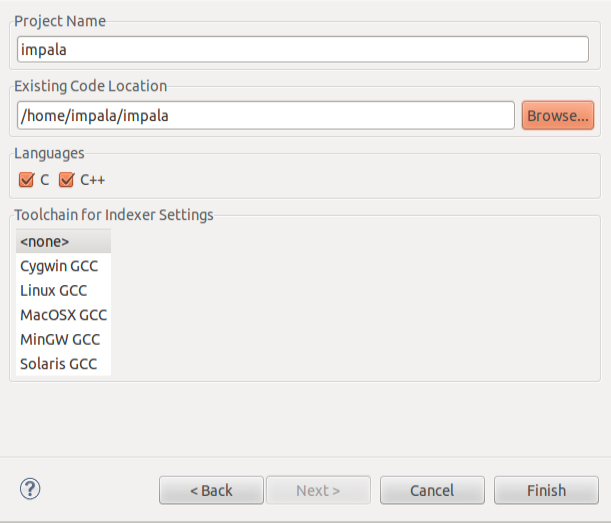
****

图5 导入Impala代码

## 调试配置

1. 右键点击工程，Properties->C/C++ Build->Settings，选择Binary Parsers，选择Elf Parser和GNU Elf Parser，之后点击Apply，如图6所示。

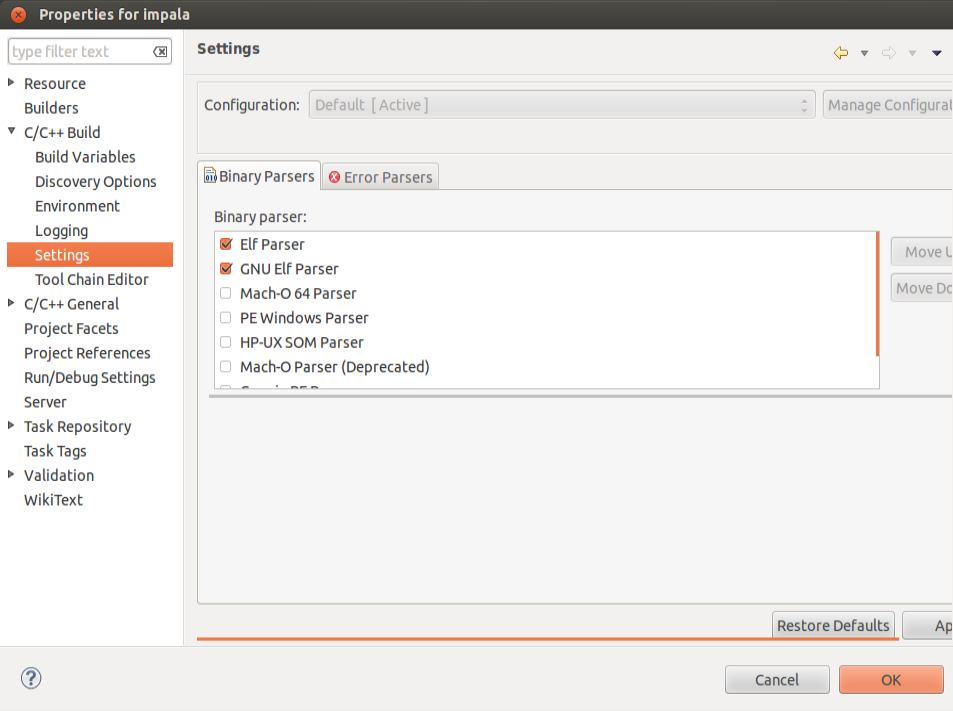


图6 C++编译器配置

1. Properties->C/C++ Build->Environment
2. 设置CLASSPTH，其值由${IMPALA\_HOME}/bin/set-classpath.sh得到（注：直接执行${IMPALA\_HOME}/bin/set-classpath.sh或者在~/.bashrc直接设置，在调试的时候，eclipse报错：Not Set Environment CLASSPATH）。具体过程如下：
3. 在命令行输入

. $IMPALA\_HOME/bin/set-classpath.sh

echo $CLASSPATH >tmp

1. 将tmp文件中的内容复制到CLASSPTH的值。
2. 设置IMPALA\_HOME，命令行下输入echo $IMPALA\_HOME，将其值添加到环境配置里，之后点击Apply。

最终结果如图7所示。

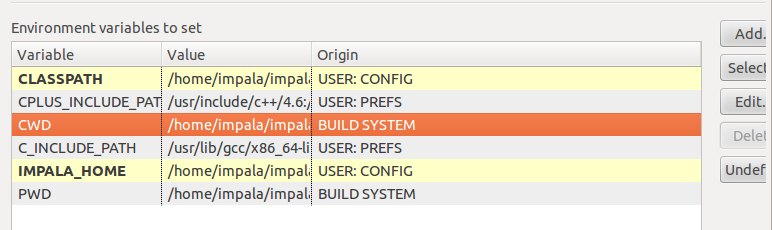


图7 配置环境变量

1. Properties->C/C++ General->Paths and Symbols。选择Includes，在GNU C语言中点击Add…，添加/usr/lib/jvm/j2sdk1.6-oracle/include，选择”Add to all configurations”和”Add to all languages”，如图8所示，点击Apply（可能会提示是否自动进行Indexing，选“是”）。

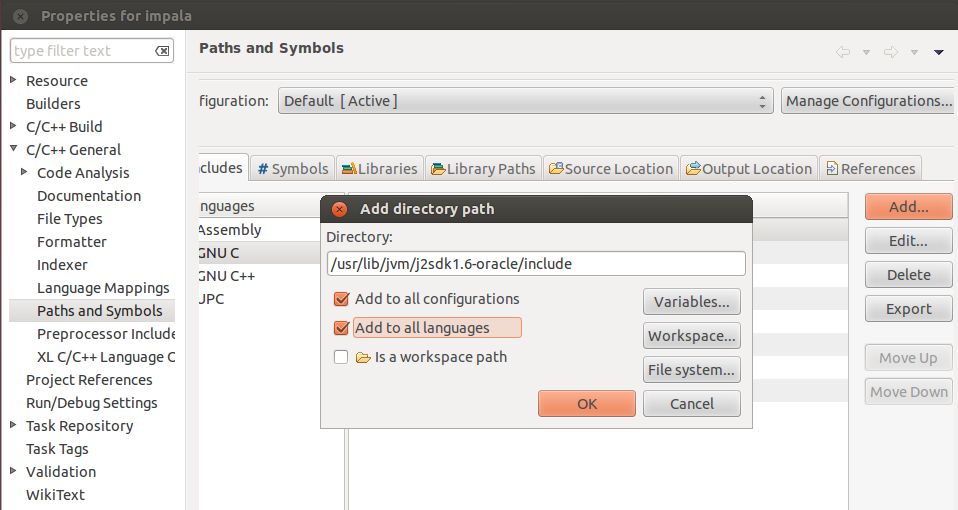


图8 配置符号和路径

1. 在工程中新建.gdbinit文件，其内容为

|  |
| --- |
| handle SIGSEGV nostop noprint pass  handle SIGABRT nostop noprint pass |

若无此文件，在调试的时候Thread会出现Segmentation fault，调试无法进行下去；或者需要手动干预才能继续执行。

1. 右键点击工程，Debug As->Local C/C++ Application，选择impalad，如图9所示。

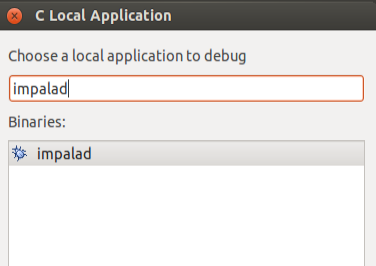


图9 选择impalad二进制进行调试

选择gdb/mi，OK。如图10所示。

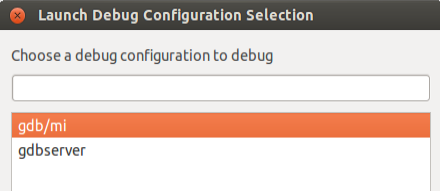


图10 选择调试配置

之后进入调试界面，如图11所示。

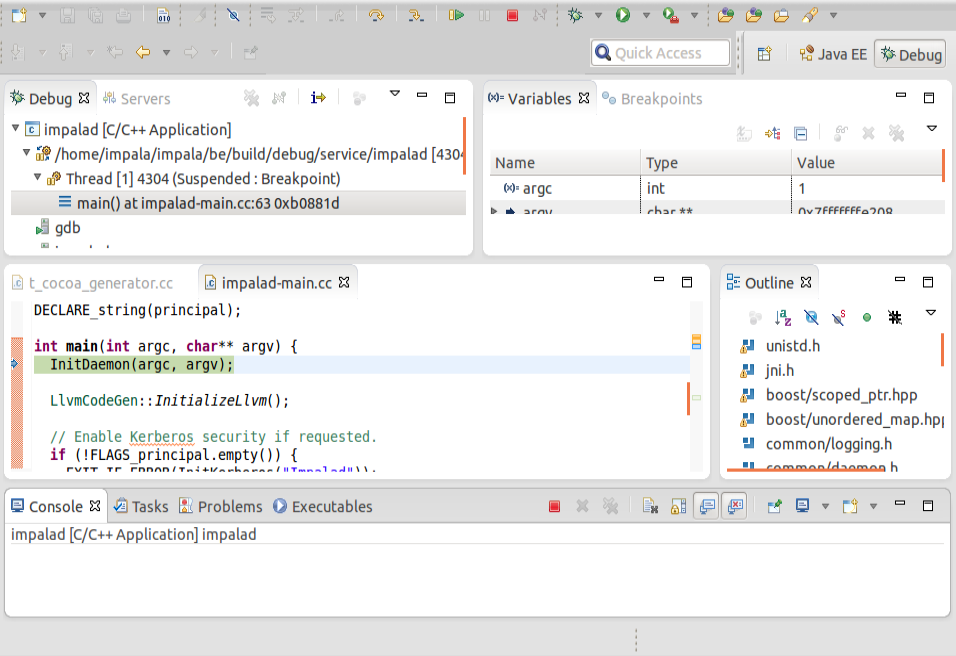


图11 Impalad调试节面

# Impala前端（FE）配置

## 在Eclipse中配置启动参数

右键点击工程，Properties->C/C++ Build->Environment，添加如下参数

|  |  |
| --- | --- |
| JAVA\_TOOL\_OPTIONS | -agentlib:jdwp=transport=dt\_socket,address=localhost:9009,server=y,suspend=y |

效果如图12所示。

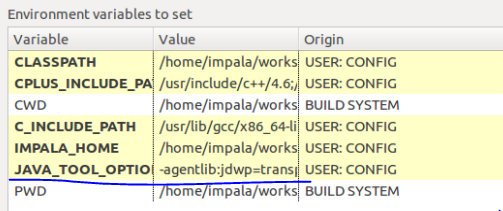


图12 配置启动参数

## 导入impala前端代码

1. File->Import...，选择Maven->Existing Maven Projects，如图13所示。

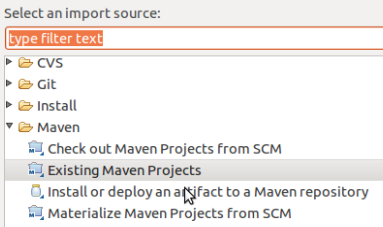


图13 导入已有maven项目

1. 选择pom.xml导入，如图14所示。



图14 通过Maven导入项目

1. 导入后右键点击工程->Properties，选择Java Build Path->Source，对源代码文件夹进行配置修改，结果如图15所示。

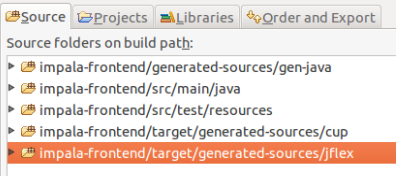


图15 配置Java源代码目录

## 调试配置

菜单栏Run->Debug configurations，找到Remote Java Application，点击New图标，创建一个实例，配置如图16所示。

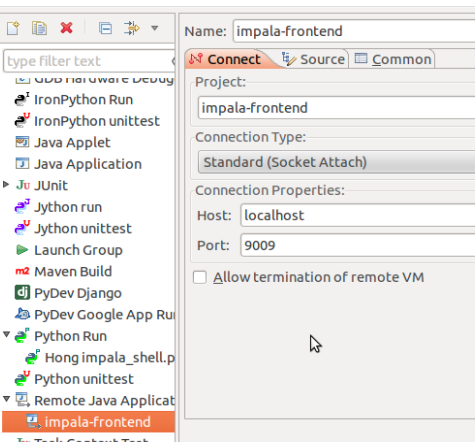


图16 Impala前端调试配置

## 调试举例

在Planner.java某行设置一个断点，如图17所示。

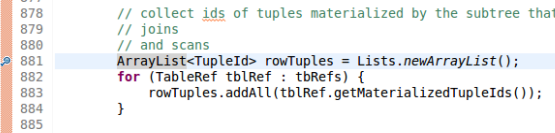


图17 设置fe部分断点

首先以debug模式运行impalad，运行后如图18所示。

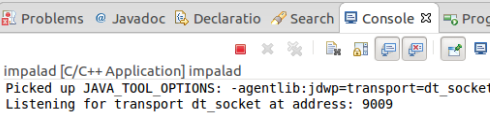


图18 调试模式的impalad

使用debug模式启动刚才配置的前端调试，运行后如图19所示。

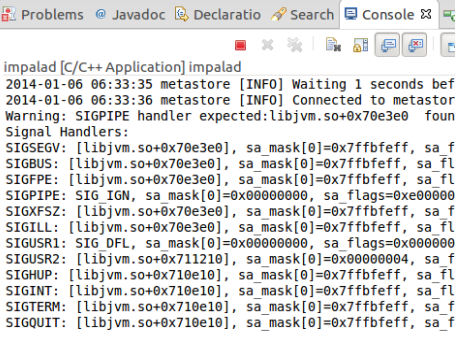


图19 启动前端调试后的impalad

接下来即可使用命令行进行查询等操作。