CSC 7700: Scientific Computing

Module C: Advanced Programming Tools

Lectures 3/4: Eclipse IDE

Dr. Steven R. Brandt

Center for Computation and Technology
Louisiana State University, Baton Rouge, LA

September 28, 2012

Goals

Goals

- The module *Advanced Programming Tools* will teach:
  - Eclipse
  - Installing Eclipse and the JDK
  - Views, Editors, Codes
  - Tools for Static Analysis
  - Tools for Debugging
  - Linux Tools for Eclipse
  - HPC Toolkit Plugin
  - Mojave
- We will use Cactus as an example of an Application Framework.

Eclipse

Eclipse

- IDE: Integrated Development Environment
- Advanced editing capabilities
- Written in Java for multi-platform support
- open-source
- extensible
- Supports C, C++, Fortran, etc. through "plugins"

Why use an IDE?

- Easier source code navigation, aids you in understanding your code.
  - **Find a source file** - Source trees can be complex and many layered. You might not remember exactly where a file with a certain name lives.
  - **Finding the definition of a symbol** - A difficult task in C/C++, as it may involve tracing header files and unraveling macros.
  - **Finding all uses of a symbol** - Not as simple as "grep." You need to ignore comments, quotes, etc.
- Simplified editing, aids maintainability.
  - **Symbol completion** - The editor knows what valid symbols are in the current context. Encourages longer variable names.
  - **Refactoring** - Rename a variable, along with all its uses. Extract a function. How much help is language specific. Helps with code maintenance.
- Colors, annotations, and highlighting - saves time
  - **Mark Errors** - See and find problems before you compile. Find files with errors.
  - **Color Syntax Highlighting** - See code pieces

Installing the Oracle JDK

Installing the Oracle JDK

Java Platform (JDK) 7

- Goto http://java.oracle.com
- Click on "Java SE" under "Top Downloads"
- Click on the "Java Download" button
- Scroll down and get the Java SE 7 JDK (Java Development Kit)

Installing the Oracle JDK

After you run the installer, add the following to your .bashrc

- `export JAVA_HOME=/usr/java/jdk1.7.0_40`
- `export PATH=$JAVA_HOME/bin:$PATH`
- Now source your .bashrc

Installing Eclipse

Installing Eclipse

- Goto `http://www.eclipse.org/downloads/`
- Get Eclipse for Parallel Application Developers
- Command:
  `tar xvf ~/Download/eclipse-SDK-4.2-linux-gtk-x86_64.tar.gz`
- Command: `cd eclipse/`
- Command:

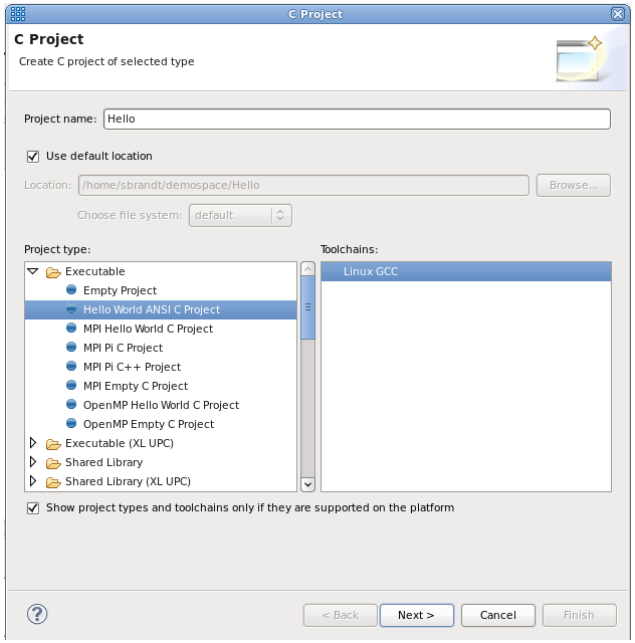
```
./eclipse -Xms1024m -Xmx2048m \
    -XX:PermSize=256m -XX:MaxPermSize=512m &
```

- You'll see a prompt for selecting a workspace. Check the box which says "Use this as the default and do not ask again"

Hello World in C

Hello World in C

- File > New > C Project (or select "Project Explorer" and type `New Menu`: Shift-Alt-N))
- Select "Hello World ANSI C Project"
- Fill in project name
- Click "Next"



Hello World in C

"Project > Build Project" builds a project

C/C++ - Hello/src/Hello.c - Eclipse SDK

File Edit Source Refactor Navigate Search Run Project Mojave Window Help

Proje

- CactusWave
- Hello
  - Binaries
  - Debug
  - Includes
  - src
- resourceManagers

Hello.c

```
1 /*
2  ==========================================================
3  Name        : Hello.c
4  Author      : Steven R. Brandt
5  Version     :
6  Copyright   : Your copyright notice
7  Description : Hello World in C, Ansi-style
8  ==========================================================
9  */
10
11 #include <stdio.h>
12 #include <stdlib.h>
13
14 int main(void) {
15     puts("!!!Hello World!!!"); /* prints !!!Hello World!!! *
```

Ou

- stdio.h
- stdlib.h
- main(void) : int

Problems | Tasks | Console | Properties | GEM Browser | GEM Analyzer | GEM Console

CDT Build Console [Hello]

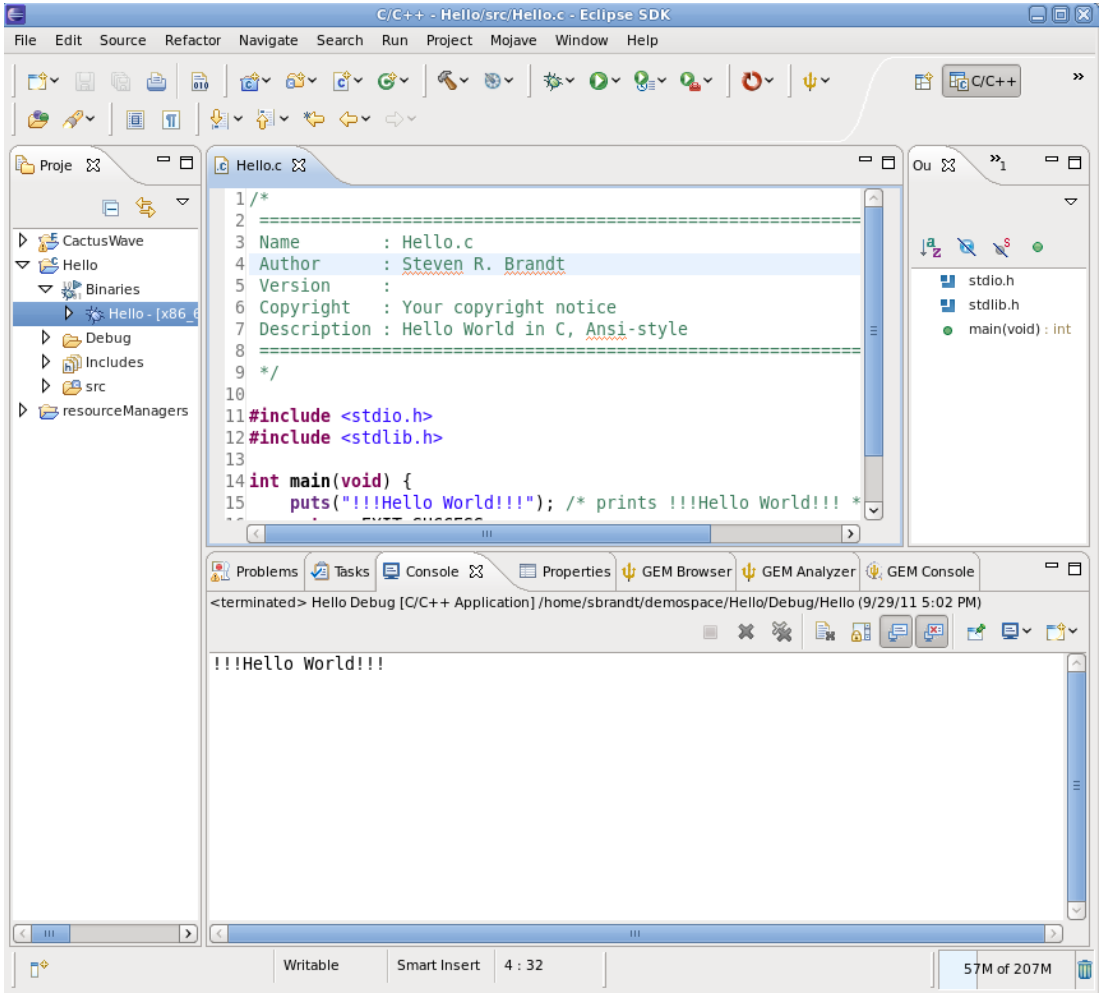
```
make all
Building file: ../src/Hello.c
Invoking: GCC C Compiler
gcc -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"src/Hello.d" -MT"src/Hello.d"
-o "src/Hello.o" "../src/Hello.c"
Finished building: ../src/Hello.c

Building target: Hello
Invoking: GCC C Linker
gcc  -o "Hello"  ./src/Hello.o
Finished building target: Hello
```

Hello

56M of 204M

Hello World in C

"Run > Run" runs the program



Advanced Editing

Control Sequences

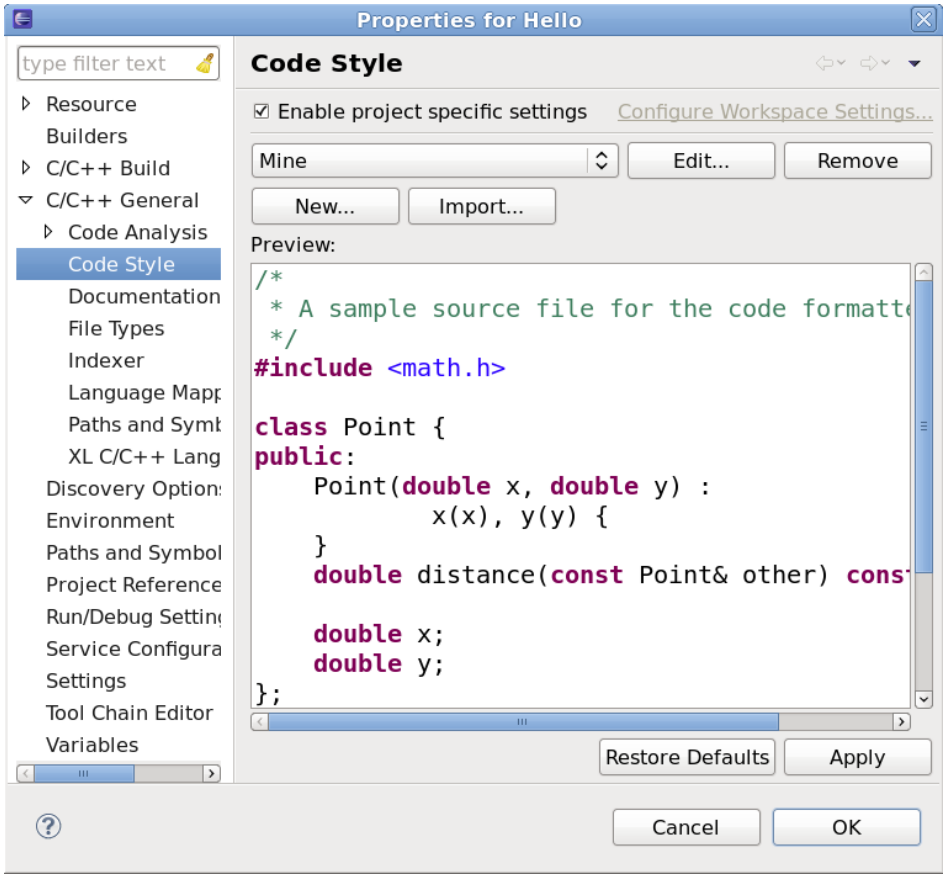
- Help > Key Assist... to find platform specific keys
- `Declaration in Workspace`: Ctrl-G
- `Open Declaration`: F3
- `Backward History`: Alt-Left (like
- `Forward History`: Alt-Right
- `References in Workspace`: Shift-Ctrl-G find symbol in workspace
- `Open Search Dialog`: Ctrl-H
- `Undo`: Ctrl-Z, `Redo`: Ctrl-Y

Control Sequences

- `Go to Matching Bracket`: Ctrl-Shift-P
- `Select Enclosing Element`: Alt-Shift-Up
- `Indent Line`: Ctrl-I
- `Toggle Comment`: Ctrl-/
- `Format`: Ctrl-Shift-F
- But what is the correct format?

Project Style

- Select a project
- Select: Project > Properties
- Open C/C++ General
- Select "Formatter"



Refactorings

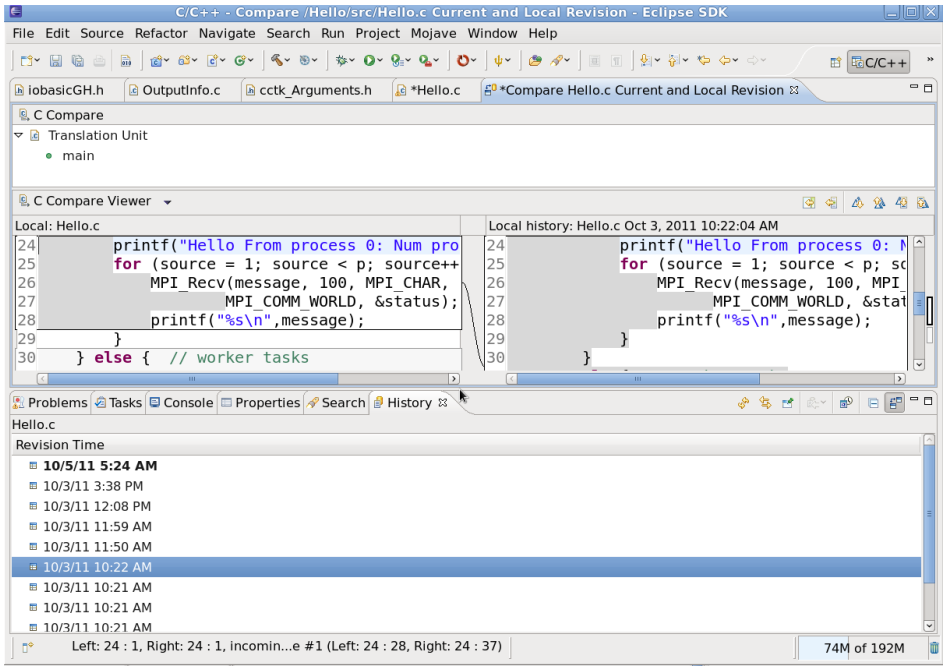
- `Rename`: Alt-Shift-R rename variable or function
- `Extract Local Variable`: Alt-Shift-L
- `Extract Method`: Alt-Shift-M
- `Surround With Quick Menu`: Alt-Shift-Z

Perspective / Editor / View

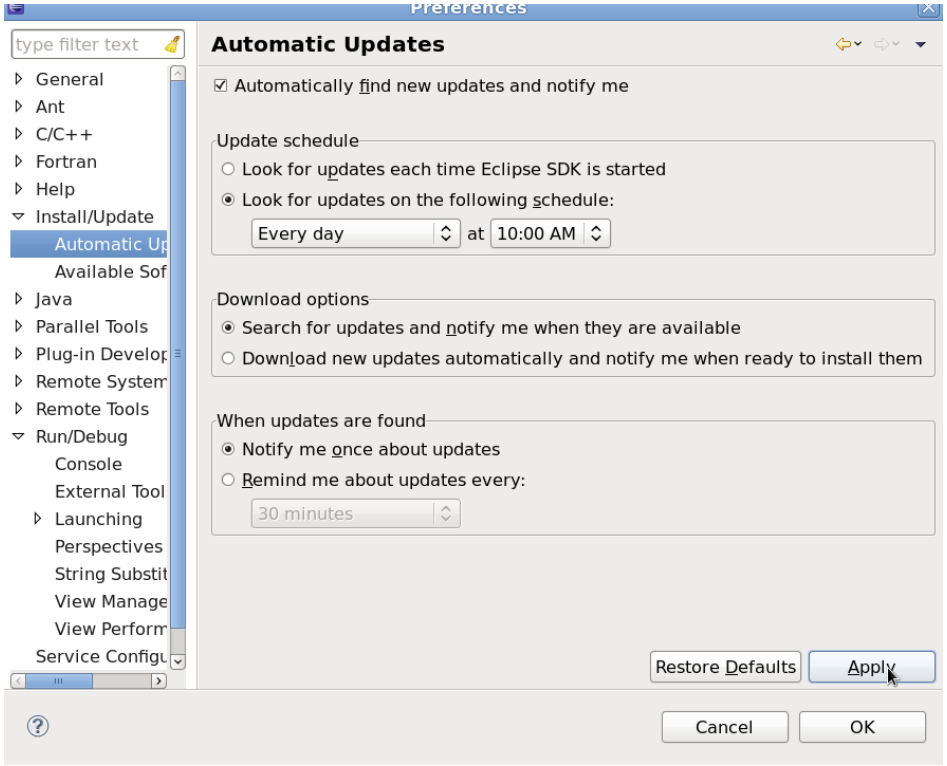
- A perspective is a collection of views. You can change them by going to `Window > Open Perspective...`
- On the right you see the outline view.
- Click the small x and it will go away.
- To bring it back, use `Window > Open View...`

History

- The "History" view tracks session edits
- It can recover old versions from within the session
- By default, you won't see edits from a previous session, but if you click the "Link with Editor and Selection" you can see them. 
- Old versions stored in `.metadata/.plugins/org.eclipse.core.resources/.history`



Automatic Updates

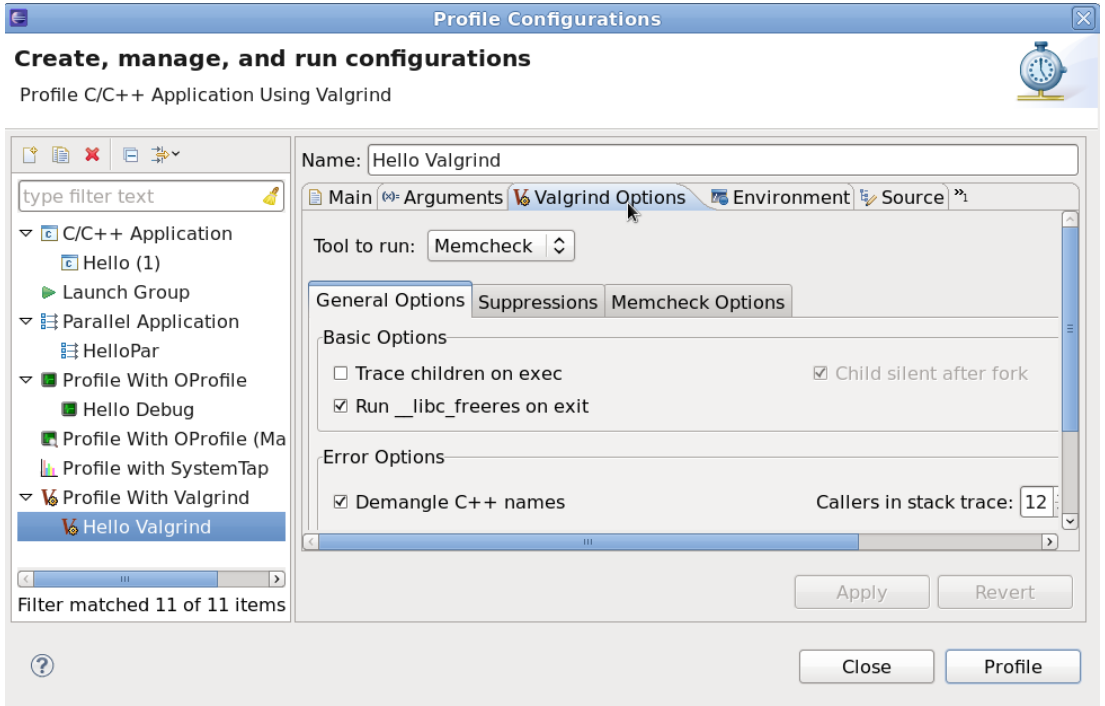


Linux Tools

LinuxTools: Valgrind

- Make sure "Valgrind Tools Integration" is installed.
- Select ProfilingDelegate Launcher.
- Under Run > Profile Configurations... > C/C++ Application > Application Name > Profiler, select "MemCheck" from the chooser.

LinuxTools: Valgrind



LinuxTools: Valgrind

C/C++ - Hello/src/Hello.c - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Mojave Window Help

Package Exp

- CactusWave
- foo
- Hello
  - Debug
    - src
      - Hello.d
      - Hello.o
      - subdir.mk
    - Hello
    - makefile
    - objects.mk
    - sources.mk
  - src
    - Hello.c
  - gmon.out
- resourceManagers

Hello.c

```
19    char message[100];
20    printf("%s\n",argv[1]);
21    MPI_Init(&argc, &argv);
22    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
23    MPI_Comm_size(MPI_COMM_WORLD, &p);
24
```

stdio.h
stdlib.h
mpi.h

Problems | Tasks | Console | Properties | OProfile | gcov | Valgrind

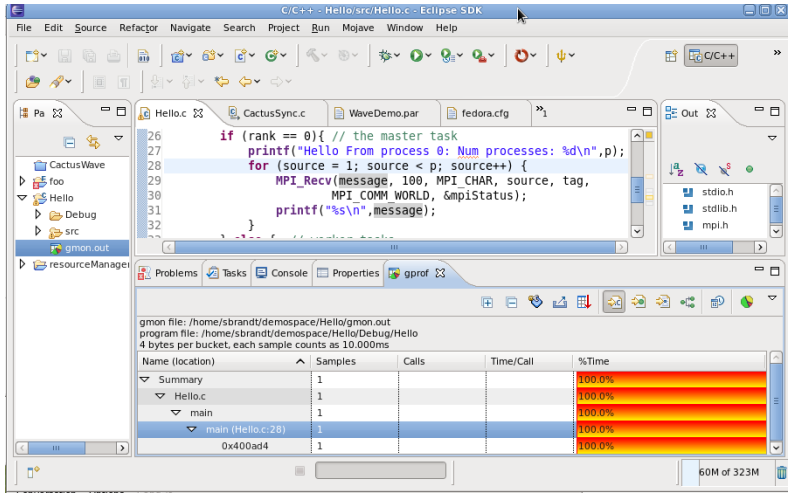
Hello Debug (1) [memcheck] /usr/bin/valgrind (10/5/11 8:17 PM)

- Invalid read of size 1 [PID: 5917]
  - at 0x4A06302: __GI_strlen (mc_replace_strmem.c:284)
  - by 0x376D667EAA: puts (in /lib64/libc-2.12.2.so)
  - by 0x400A2E: main (Hello.c:20)
  - Address 0x0 is not stack'd, malloc'd or (recently) free'd [PID: 5917]
- Process terminating with default action of signal 11 (SIGSEGV) [PID: 5917]

143M of 342M

LinuxTools: gprof

- Make sure "GProf Integration" is installed
- Project Properties > C/C++ Build > Settings > GCC Compiler > Debugging, then check `-pg`
- Compile and run
- You might have to hit `Refresh`: F5
- Linux Tools can process `gmon.out`

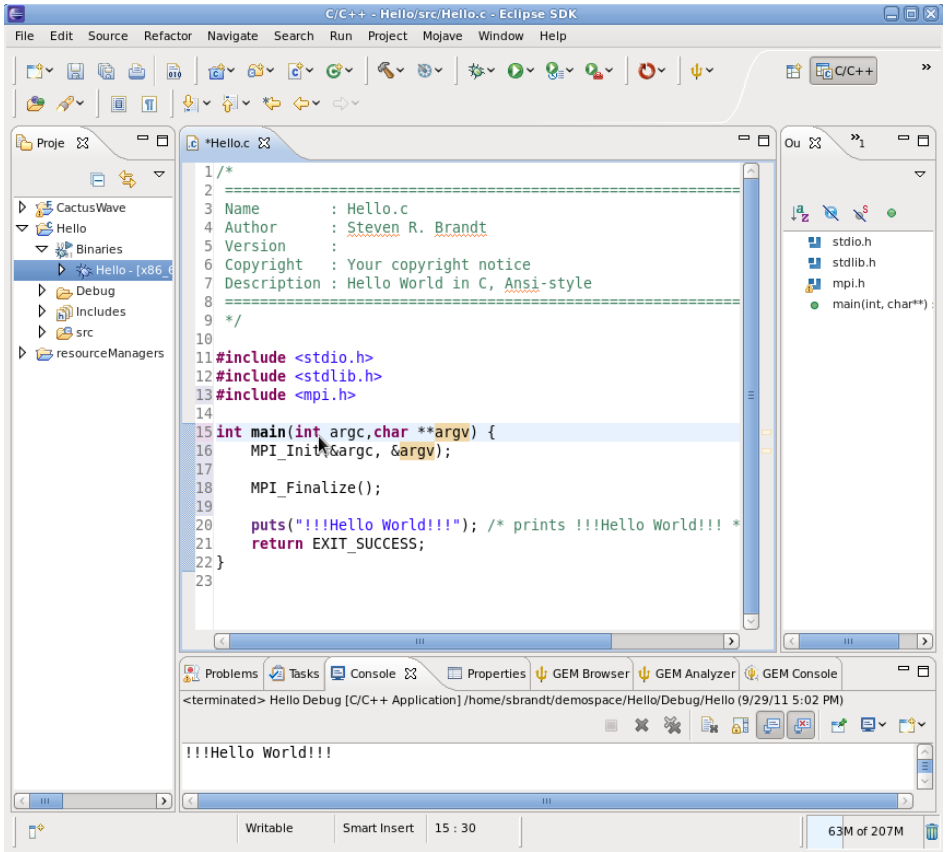


Hello World in C+MPI

Hello World in C+MPI

Upgrading the program to use MPI

- Add `#include <mpi.h>`
- Click inside the program type "mpi" then hit Ctrl-space. You'll see a code completion options. Choose "MPI Init and Finalize" you now have errors in your code. Use the editor to add argc and argv to main.



Hello World in C+MPI

Building will now produce errors.

C/C++ - Hello/src/Hello.c - Eclipse SDK

File Edit Source Refactor Navigate Search Run Project Mojave Window Help

Proje

- CactusWave
- Hello
  - Debug
  - Includes
  - src
- resourceManagers

Hello.c

```
1 /*
2 ==========================================================
3 Name        : Hello.c
4 Author      : Steven R. Brandt
5 Version     :
6 Copyright   : Your copyright notice
7 Description : Hello World in C, Ansi-style
8 ==========================================================
9 */
10
11 #include <stdio.h>
12 #include <stdlib.h>
13 #include <mpi.h>
14
15 int main(int argc,char **argv) {
16     MPI_Init(&argc, &argv);
17
18     MPI_Finalize();
```

Ou

- stdio.h
- stdlib.h
- mpi.h
- main(int, char**)

Problems | Tasks | Console | Properties | GEM Browser | GEM Analyzer | GEM Console

CDT Build Console [Hello]

```
make all
Building file: ../src/Hello.c
Invoking: GCC C Compiler
gcc -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"src/Hello.d" -MT"src/Hello.d"
-o "src/Hello.o" "../src/Hello.c"
../src/Hello.c:13:17: warning: mpi.h: No such file or directory
../src/Hello.c: In function 'main':
../src/Hello.c:16: warning: implicit declaration of function 'MPI_Init'
../src/Hello.c:18: warning: implicit declaration of function 'MPI_Finalize'
Finished building: ../src/Hello.c

Building target: Hello
```

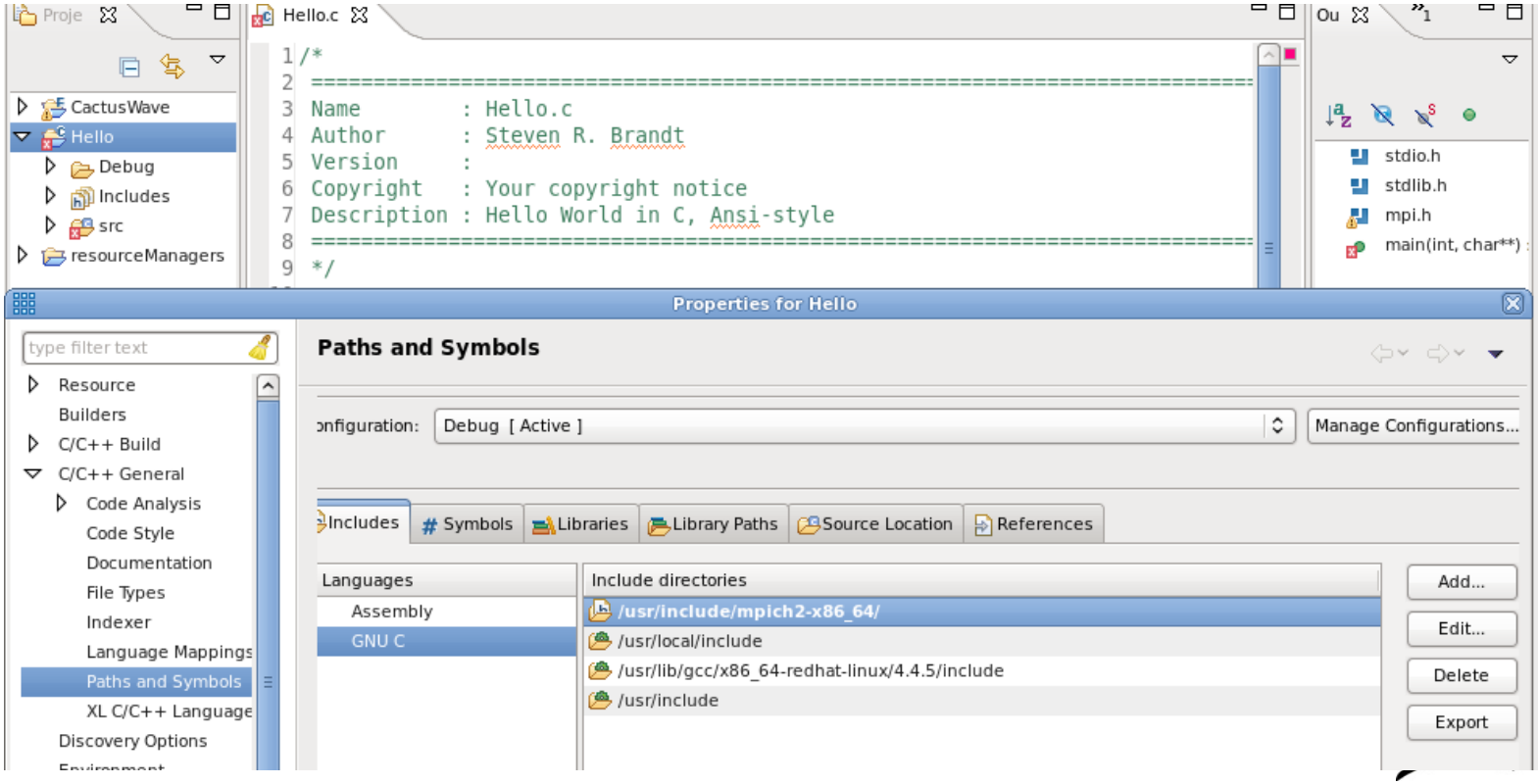
Hello

57M of 207M

Hello World in C+MPI

To resolve these errors...

- Right click on the "Hello" project in the project explorer view.
- Select "Properties"
- Open "C/C++ General" and click on "Paths and Symbols"
- Select the C language and click "Add" to add the include.



Hello World in C+MPI

How did I know the mpi include path?

- Run `locate mpi.h|grep openmpi`
- Output will contain something like this: ...
  `/usr/include/openmpi-x86_64/mpi.h`
- Run
  `rpm -qilf /usr/include/openmpi-x86_64/mpi.h|grep /lib/`
- Output will contain something like this:

```
/usr/lib64/openmpi/lib/libmca_common_sm.so
/usr/lib64/openmpi/lib/libmpi.so
/usr/lib64/openmpi/lib/libmpi_cxx.so
/usr/lib64/openmpi/lib/libmpi_f77.so
/usr/lib64/openmpi/lib/libmpi_f90.so
/usr/lib64/openmpi/lib/libompitrace.so
...
```

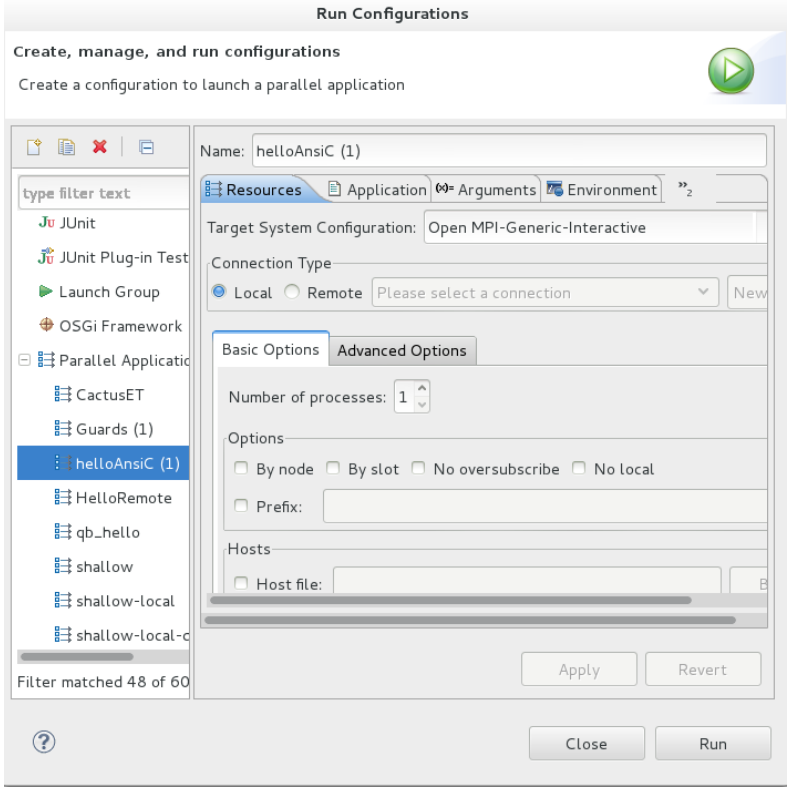
Hello World in C+MPI

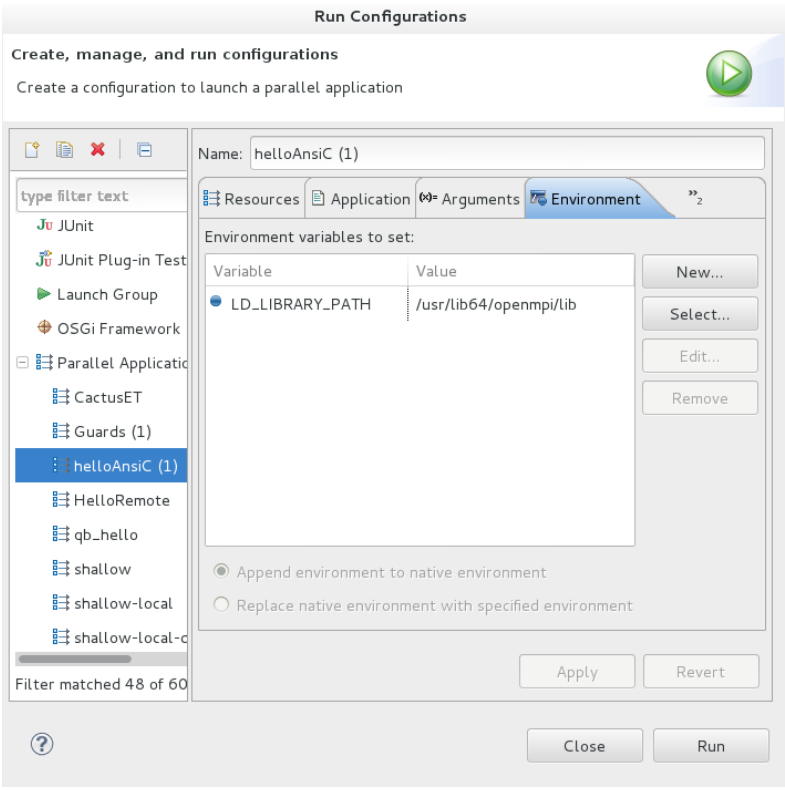
- Got to Project Properties > C/C++ General > Paths and Symbols. Select GNU C, then Add..., then add the path `/usr/include/openmpi-x86_64`.
- While you're still in "C/C++ General", click on the "Library Paths" tab and set that. (In my case, add /usr/lib64/openmpi/lib)
- Next, go to the "Libraries" tab and edit that to include "mpi"

Hello World in C+MPI

- "Project > Build" should now work again.
- Now we need to set up a parallel run. Click "Window > Open Perspective > Other... > Parallel Runtime"
- Right click inside the "Target System Configuration" tab.
- Choose OpenMP-Generic-Interactive
- Under Environment, set the `LD_LIBRARY_PATH` variable
- Under Application, select your Application Program
- Click Next > Finish.

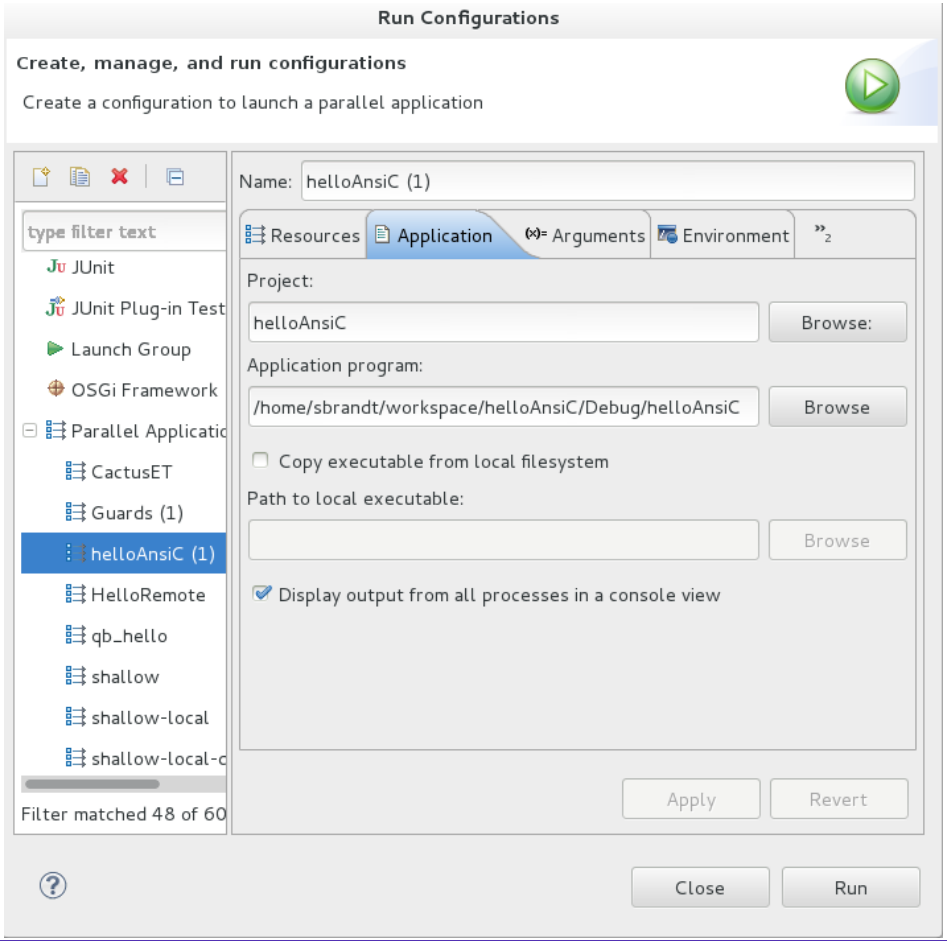
Hello World in C+MPI





Hello World in C+MPI

- Click `Run > Run Configurations...`
- Right Click on `Parallel Applications` and select `New`
- Select the new application, configure the "Application program."
- Click "Apply" and "Run"



Hello World in C+MPI

- Click `Run > Run Configurations...`
- Select your parallel application
- Select the "Resources" tab
- Adjust the "Number of processes" to 2
- Click "Apply" and "Run"

Adding Message Passing

Hello World in C+MPI

- Naviagate back to the C/C++ perspective (`Window > Open Perspective > C/C++`)
- Click in the code editor between `MPI_Init` and `MPI_Finalize`.
- Type "mpi" and hit Ctrl-space. Take the "mpisr" code completion. A complete skeleton for doing an MPI send and receive will appear. Fill in the missing variable declarations.

```
int rank,p,source,dest,tag=66;
MPI_Status status;
char message[100];
```

- Alt-P followed by B will build the current project.
- Click on the down arrow and select your run config.

Mojave

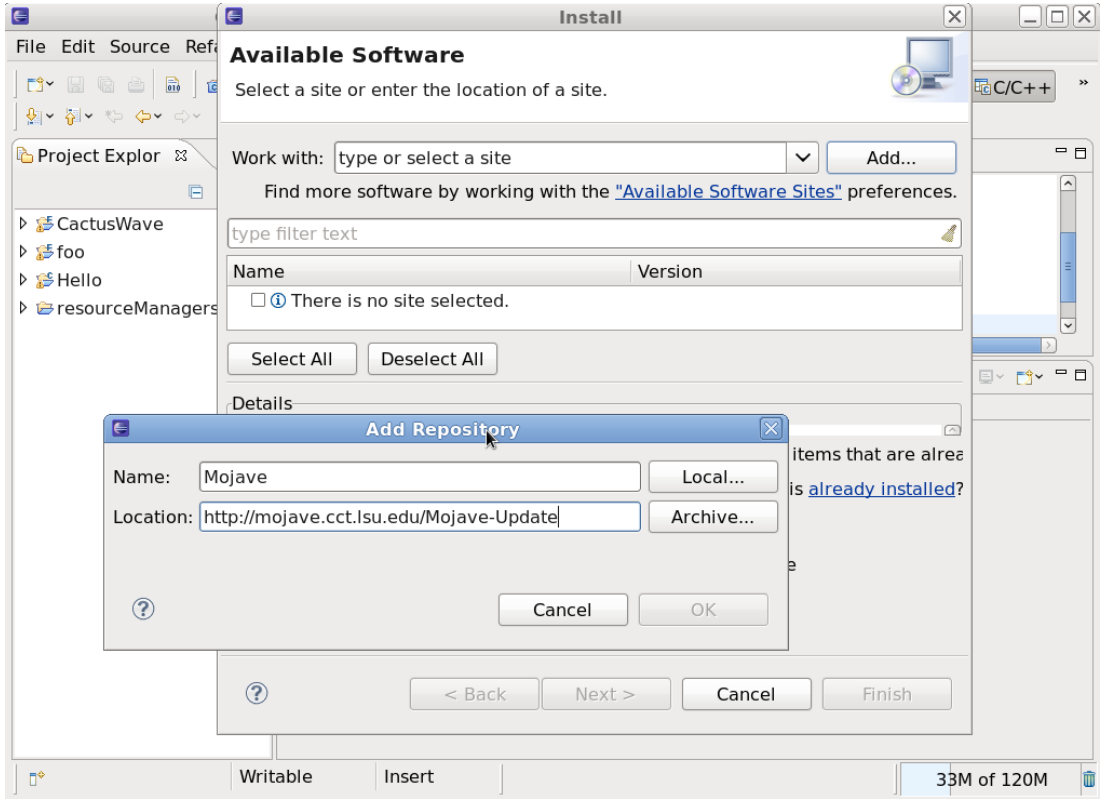
Mojave

- Mojave: A place for Cacti to live
- An interface to the Cactus Source Code
- An interface to SimFactory

Installing Mojave

Installing Mojave

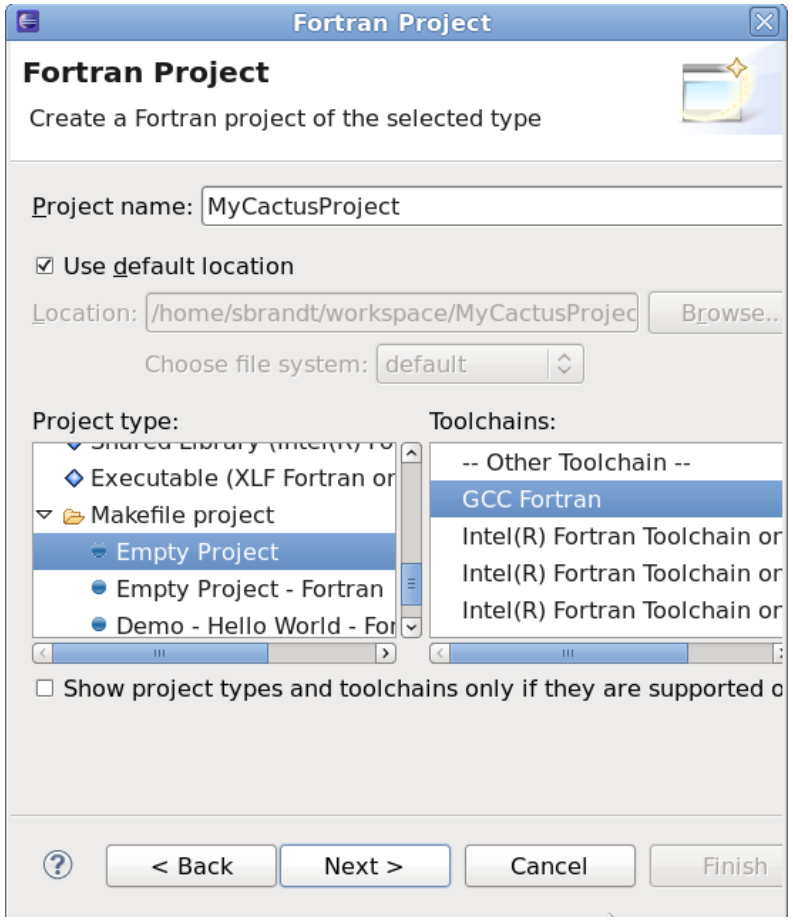
- Help > Install New Software > Add...
- A popup will appear. Fill in the Name and Location
- Hit "OK" then "Next" and accept, etc. on the following screens.



The Wave Equation on your Local Machine

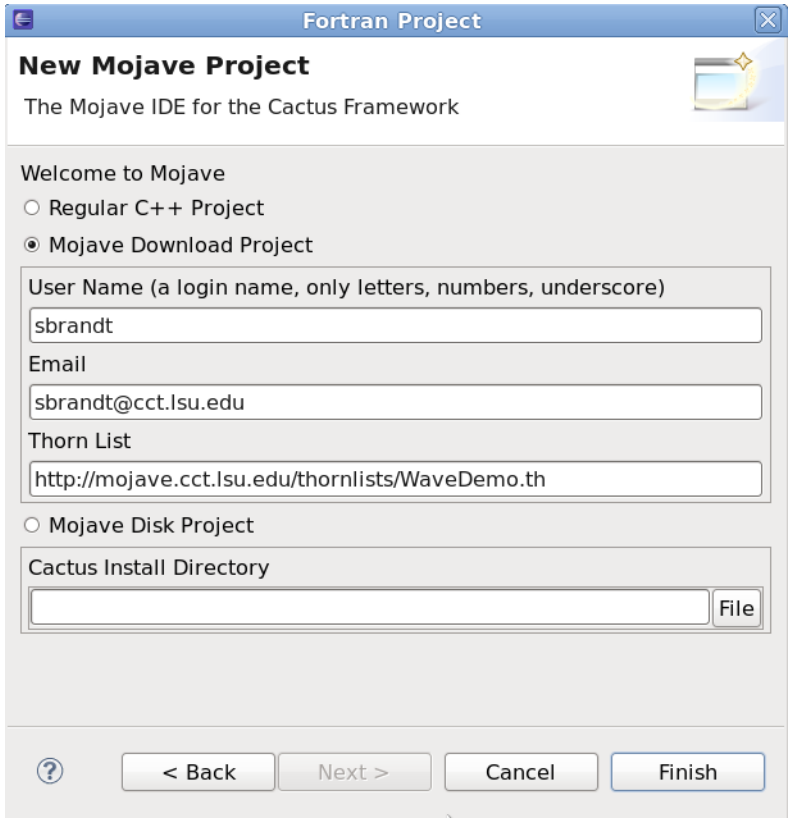
Mojave

- To create a Mojave Project, start by selecting
  `File > New Project > Other... > Fortan > Fortran Project`
- Select an empty makefile project



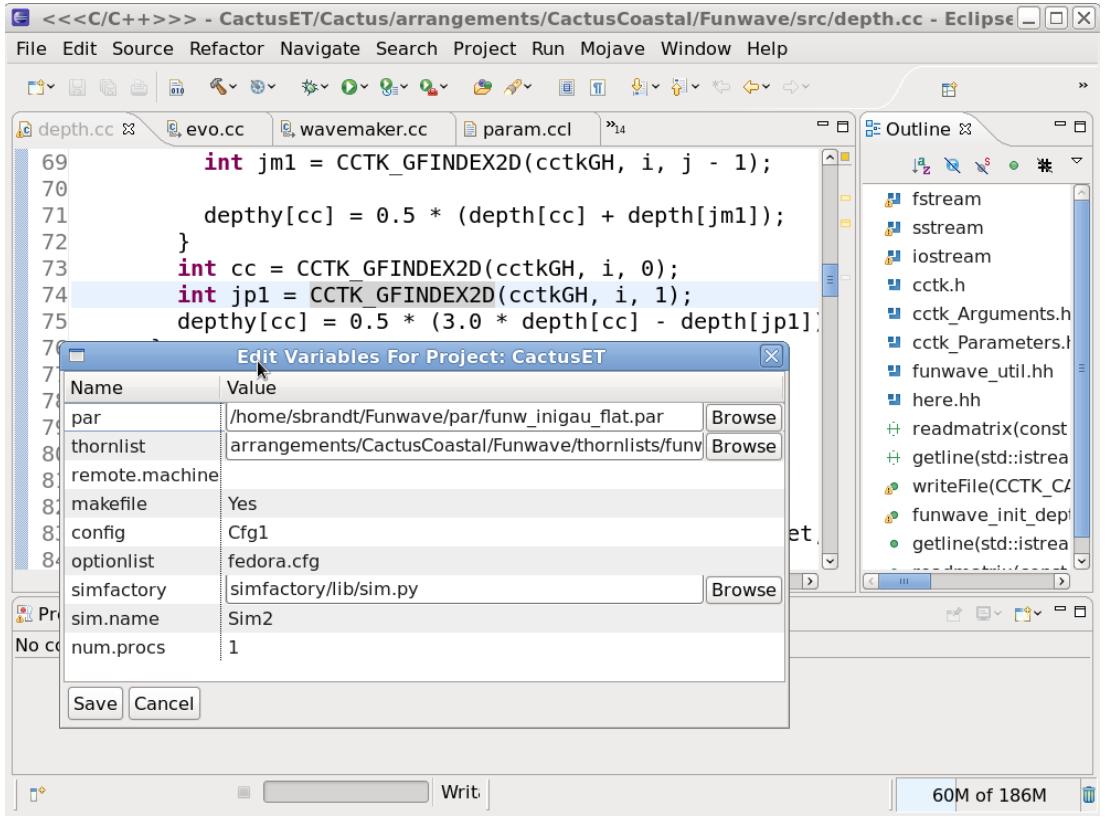
Mojave

- When you get to the last screen of the wizard, you will be prompted to create one of three types of projects.
- Select the "Mojave Download Project" and create a WaveDemo



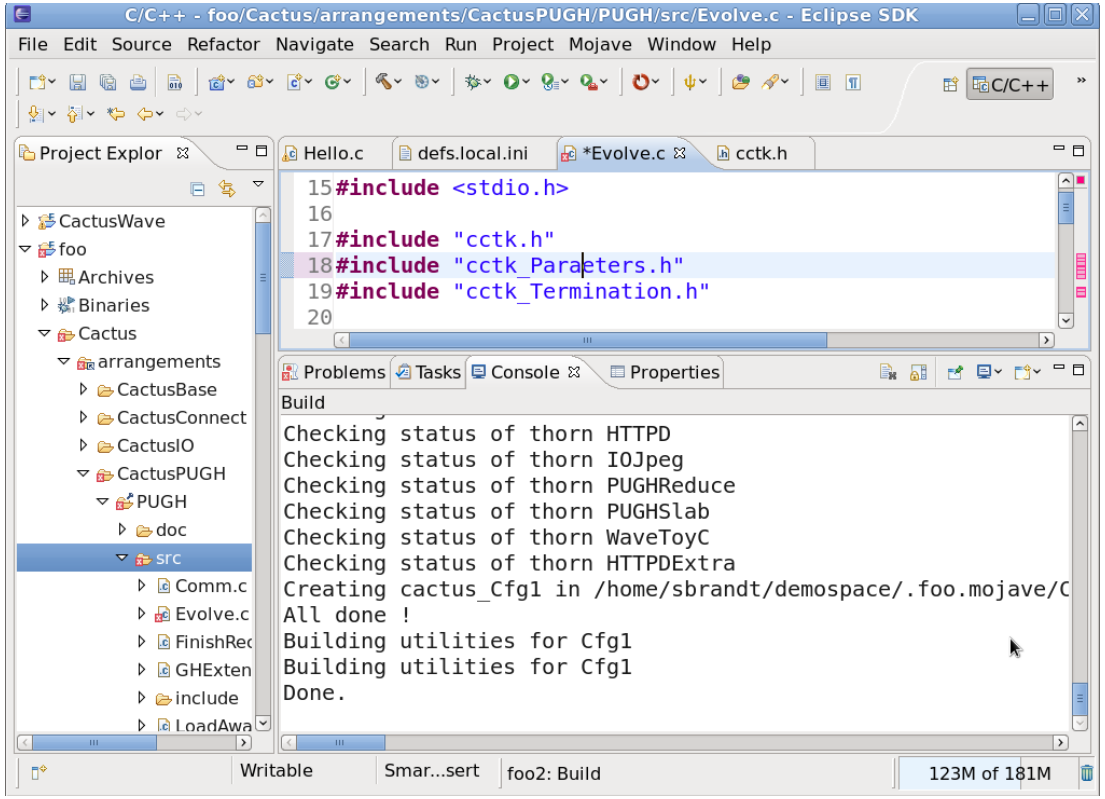
Mojave

- Now that you've downloaded a project, please configure using the mojave variable editor
- `Mojave > Edit Variables...`
- Select the basic information needed for a cactus build/run



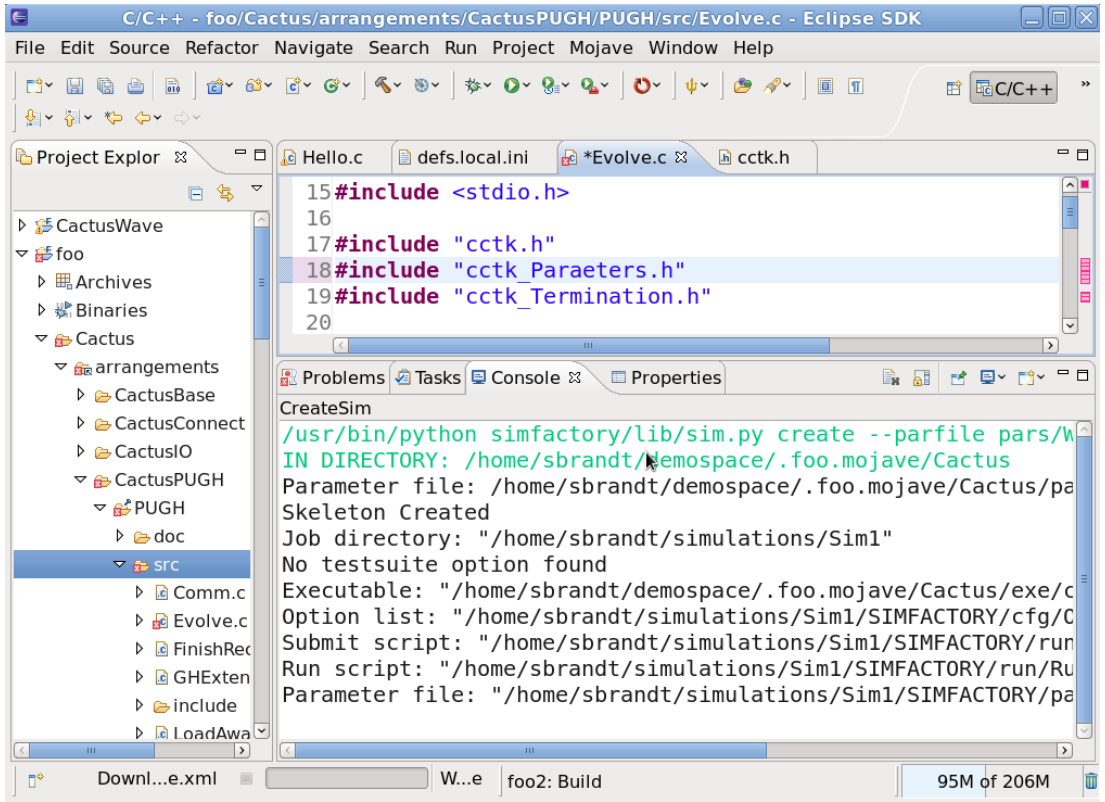
Mojave

- Mojave > Build
- Mojave > CreateSim
- Mojave > RunSim



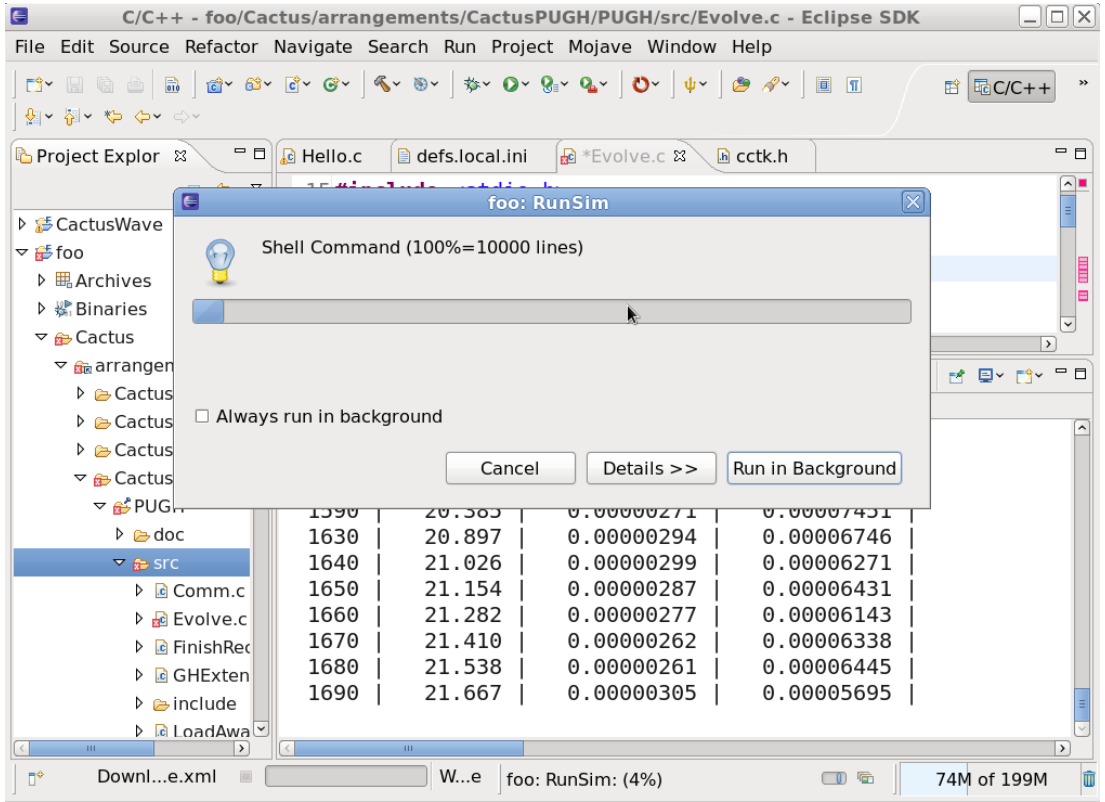
Mojave

- Mojave > Build
- Mojave > CreateSim
- Mojave > RunSim



Mojave

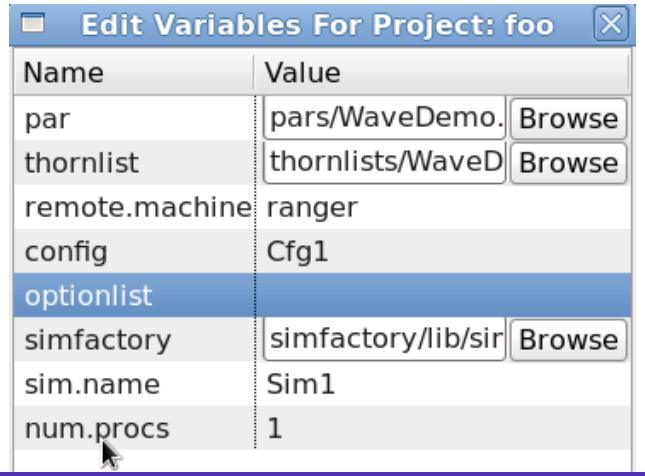
- Mojave > Build
- Mojave > CreateSim
- Mojave > RunSim



Mojave

- Now to use Supermike.
- Change variable definitions in Mojave.
- Make sure defs.local.ini is correct (you can edit it in Mojave as well, just use `Open Resource`: Shift-Ctrl-R and start typing to call up the file)
- My Supermike configuration looks like this:

```
[mike]
user = sbrandt
sourcebasedir = /work/@USER@
basedir = /work/@USER@/simulations
```



Mojave

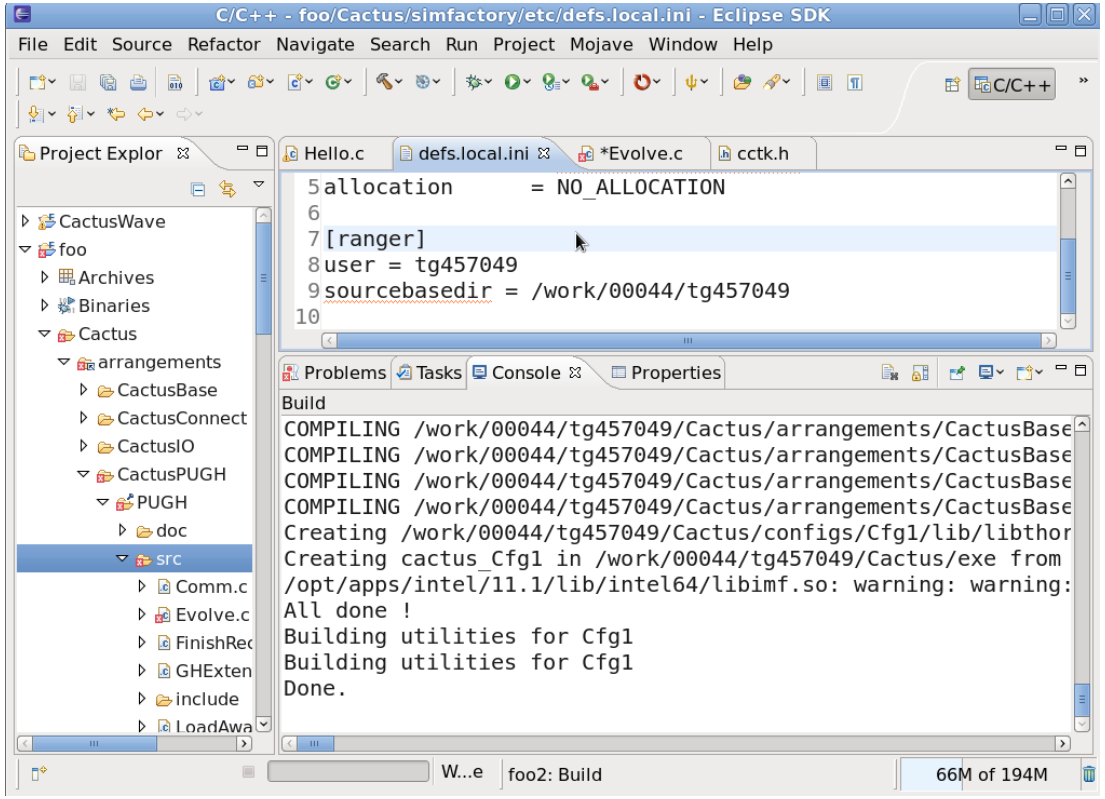
Next, change defs.ini. It should have the directories named "thornlists" and "pars" under sync-sources:

```
# -*-conf-*-
# any <<EOT...EOT entries are converted internally to lists, since all
# these options do actually represent lists of files
# Official Cactus, SimFactory, and GetComponents entries
sync-sources   = <<EOT
CONTRIBUTORS
COPYRIGHT
Makefile
arrangements
bin
lib
manifest
repos
simfactory
pars
thornlists
...
```

The Wave Equation on Mike

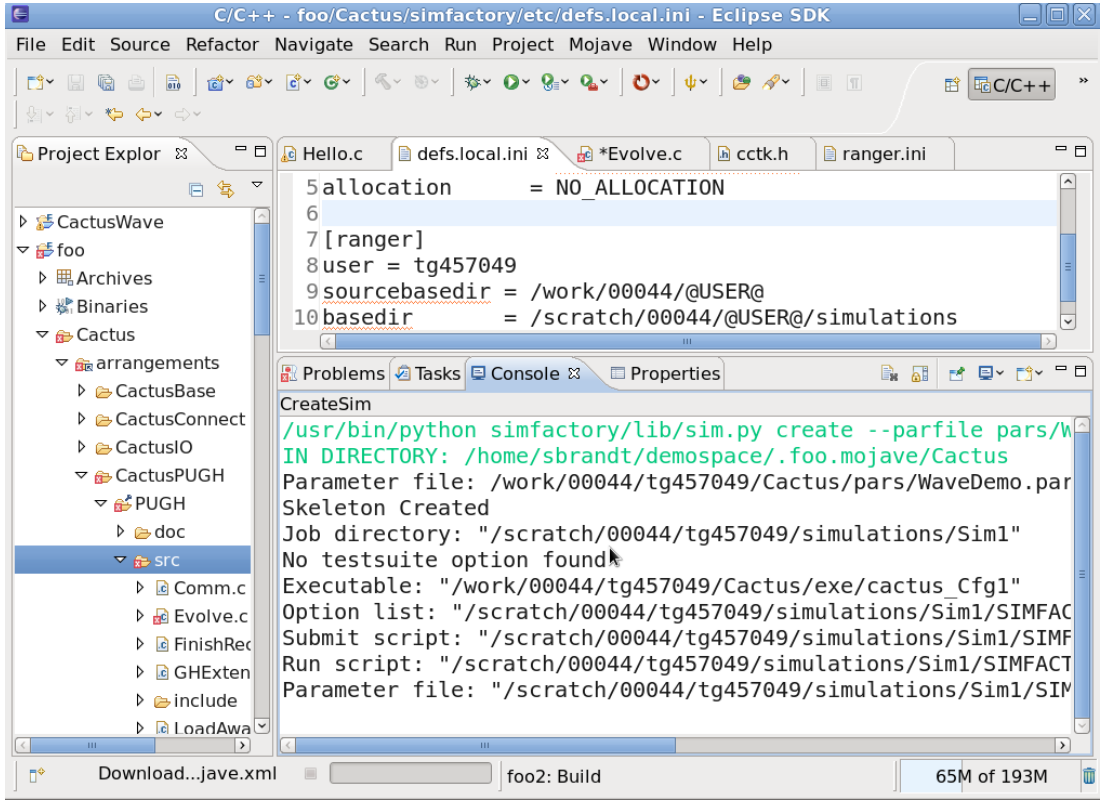
Mojave

- Mojave > Build
- Mojave > CreateSim
- Mojave > SubmitSim



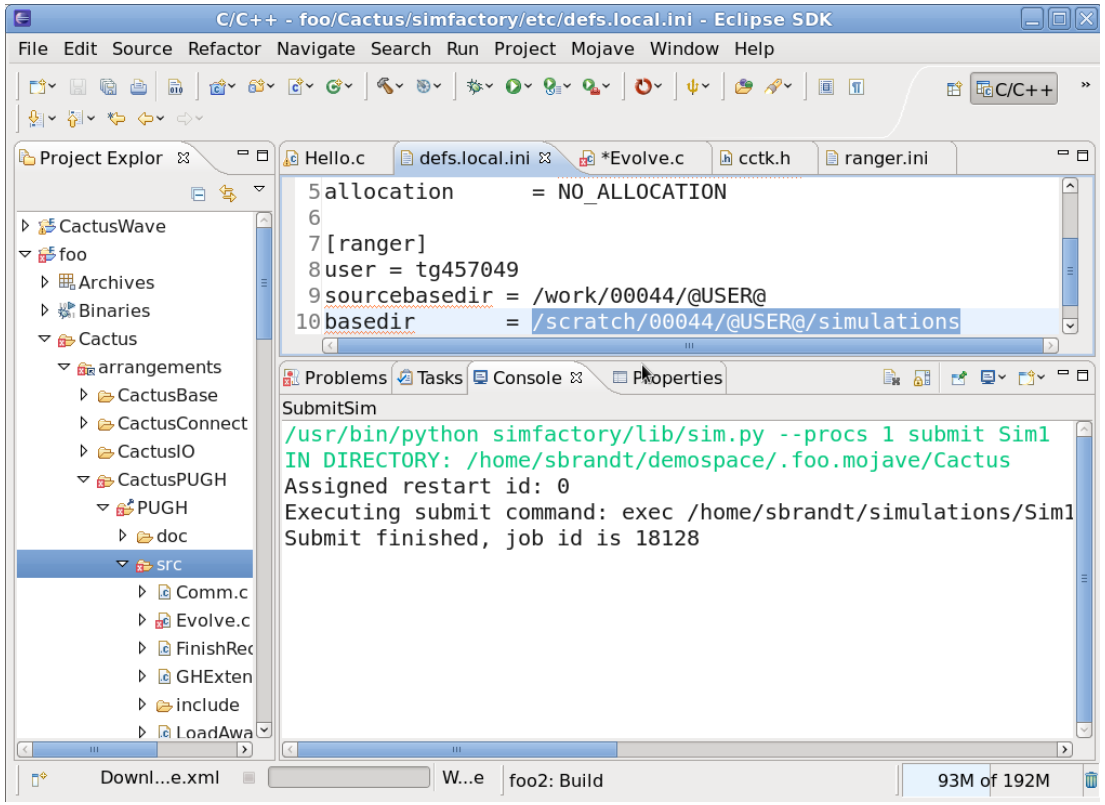
Mojave

- Mojave > Build
- Mojave > CreateSim
- Mojave > SubmitSim



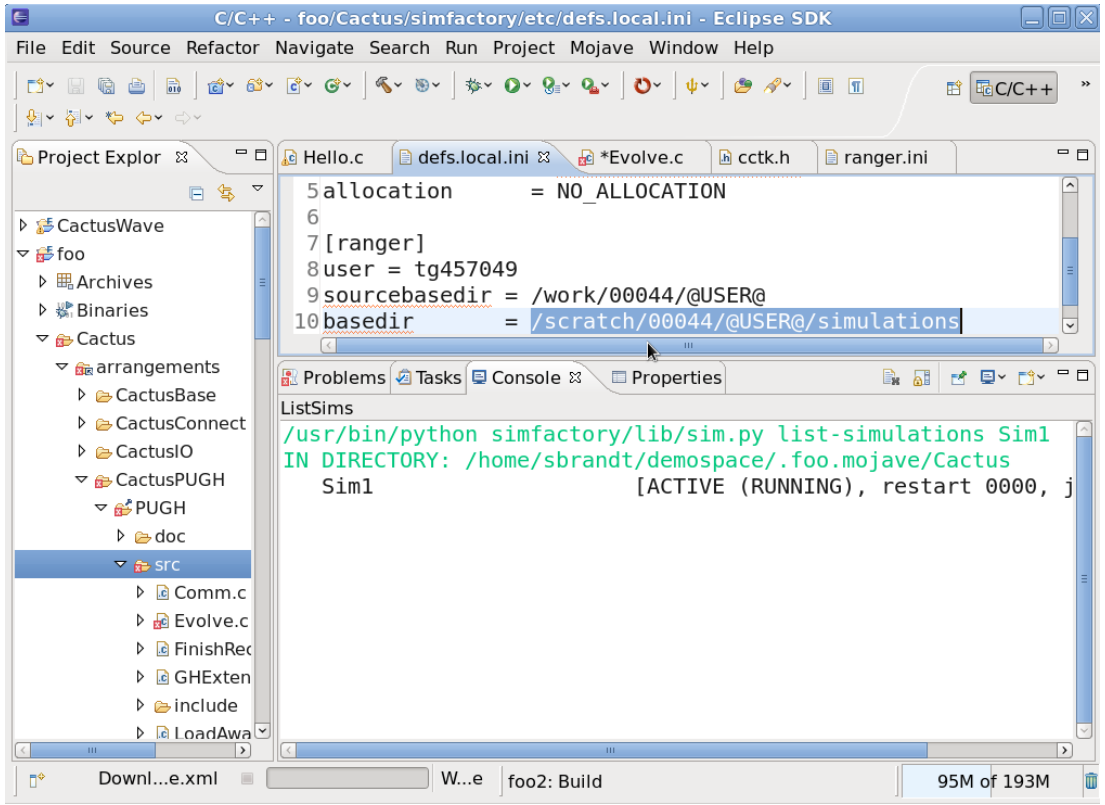
Mojave

- Mojave > Build
- Mojave > CreateSim
- Mojave > SubmitSim



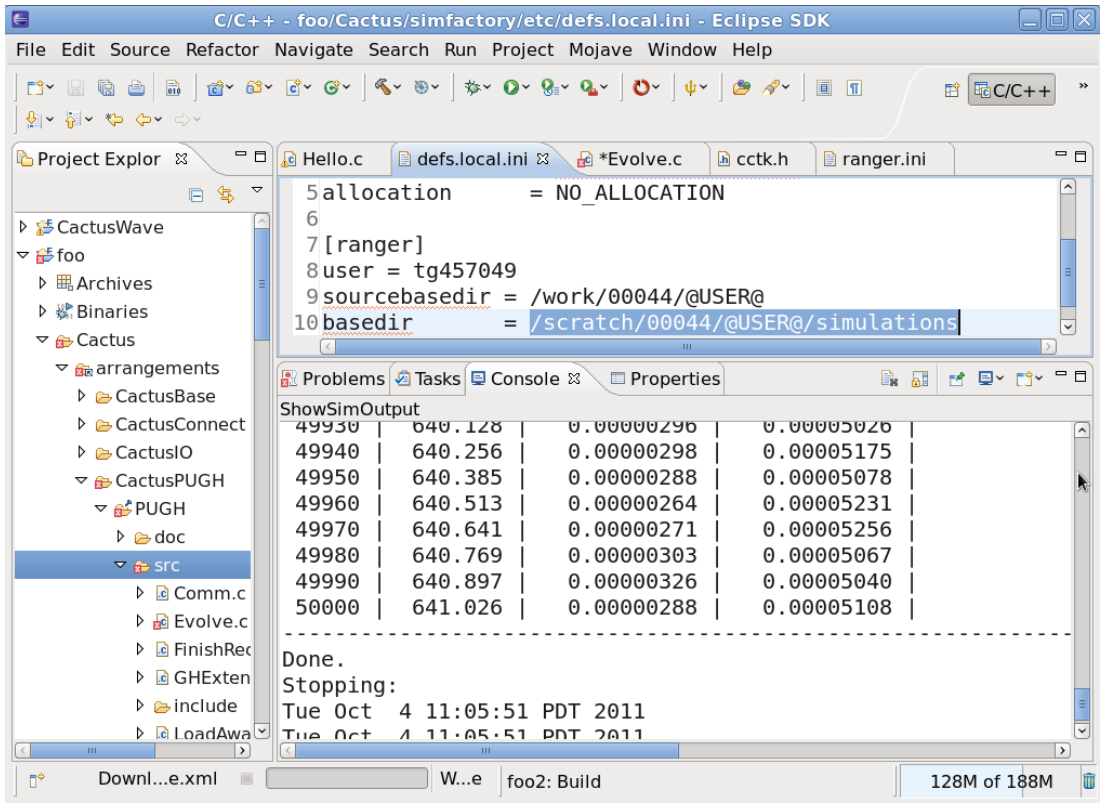
Mojave

- Mojave > Build
- Mojave > CreateSim
- Mojave > SubmitSim
- Mojave > ListSims



Mojave

- Mojave > Build
- Mojave > CreateSim
- Mojave > SubmitSim
- Mojave > ListSims
- Mojave > ShowSimOutput



Mojave

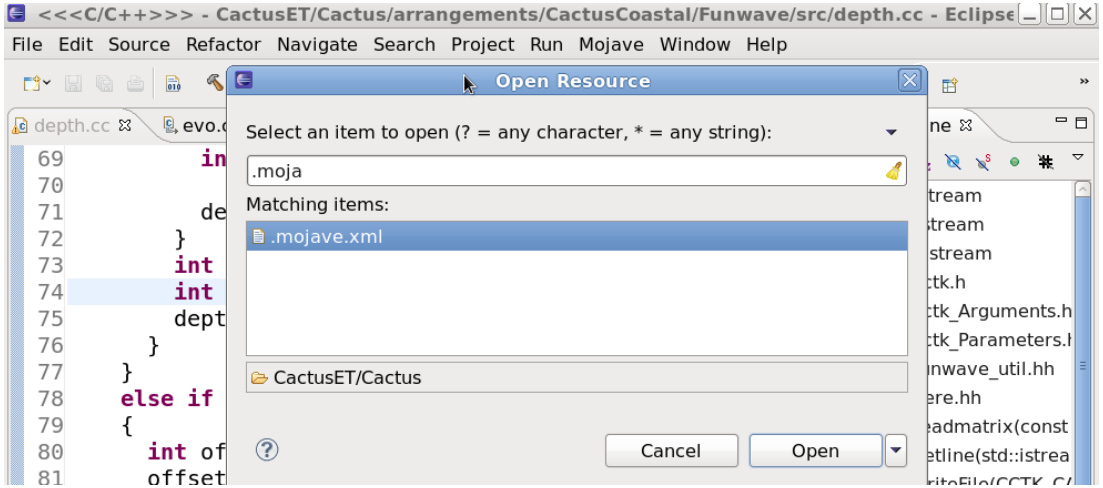
Using the Mojave Menu

- The Mojave menu comes pre-configured with all the basic commands you might want to run.
- "New Thorn" allows you to add to the existing set of thorns.
- "Update Repo" will invoke GetComponents to update the existing installation.
- Build provides an alternative to the `Project > Build` menu.
- CreateSim will create a simfactory run configuration
- CleanupSim will call simfactory cleanup on a run configuration
- PurgeSim will erase the data associated with a simfactory run

Mojave

Advanced Configuration of the Mojave Menu

- Type `Open Resource`: Shift-Ctrl-R, then .mojave.xml.
- Before you finish typing, Eclipse will find the file and complete the name for you. Pull it up in the editor.
- The file name will show in matching items.



Mojave

- .mojave.xml contains variables and actions.
- actions are made up of a sequence of commands
- commands can have a name, or a name value pair
- if a value is not defined, neither the name nor value will be placed into the generated shell command
- editing .mojave.xml directly can be useful for adding or changing variables, or adding or changing menu items.

Mojave

<<<C/C++>>> - CactusET/Cactus/.mojave.xml - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Mojave Window Help

depth.cc | schedule.ccl | evo.cc | wavemaker.cc | param.ccl | .mojave.xml

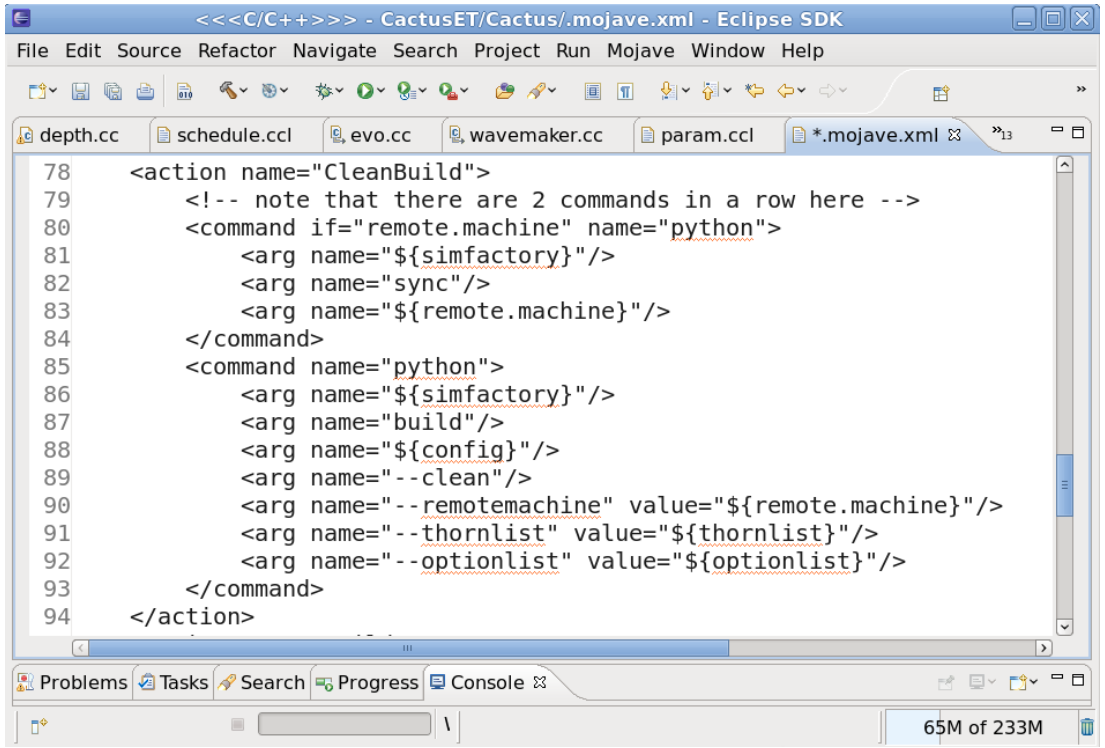
```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?><mojave>
    <!-- Want to have a var editing menu
         Some values should be per-project.
         Those will have the naming convention
         project.xxx. The variable project.dir
         is built-in.
         -->
    <var name="config" type="string" value="Cfg1"/>
    <var name="thornlist" type="file" value="arrangements/CactusCoastal
    <var name="par" type="file" value="/home/sbrandt/Funwave/par/funw_i
    <var name="simfactory" type="file" value="simfactory/lib/sim.py"/>
    <var name="num.procs" type="string" value="1"/>
    <var name="remote.machine" type="string" value=""/>
    <var name="optionlist" type="string" value="fedora.cfg"/>
    <var name="sim.name" type="string" value="Sim2"/>
    <var name="makefile" type="string" value="Yes"/>
    <action name="Update Repo">
        <command name="perl">
            <arg name="../GetComponents"/>
            <arg name="--update"/>
            <arg name="${thornlist}"/>
        </command>
    </action>
    <action name="Create">
        <command name="python">
            <arg name="${simfactory}"/>
            <arg name="create"/>
            <arg name="--parfile" value="${par}"/>
            <arg name="--configuration" value="${config}"/>
            <arg name="--remotemachine" value="${remote.machine}"/>
            <arg name="${sim.name}"/>
        </command>
    </action>
```

Problems | Tasks | Search | Progress | Console

52M of 242M

Mojave

- Commands can be conditional on a variable definition
- Multiple commands can be part of a single action with combined console output.



Mojave

- To update symbols after a cactus build, right click on the project then select `Index > Rebuild`
- Eclipse will then update all its symbol information with the contents of the files generated during the build.
- By default, Mojave builds cactus files inside the `~/.mojaveconfig` directory.