

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



1 Goals

2 Eclipse

- Installing the Oracle JDK
- Installing Eclipse

3 Hello World in C

- Advanced Editing
- Linux Tools
- Hello World in C+MPI
- Adding Message Passing

4 Mojave

- Installing Mojave
- The Wave Equation on your Local Machine
- The Wave Equation on Mike



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





- 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



- 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"

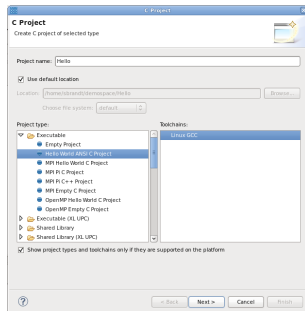


Hello World in C



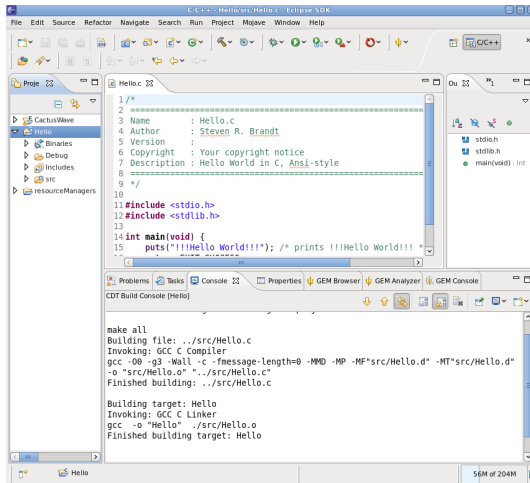
Hello World in C

- 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



```
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!!! */
16 }
```

CDT Build Console [Hello]

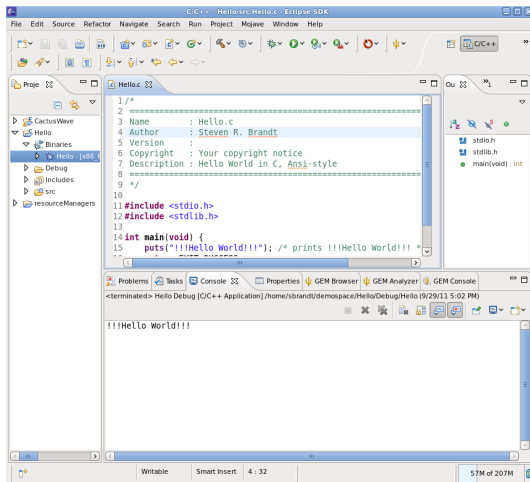
```
make all
Building file: ../src/Hello.c
Invoking: GCC C Compiler
gcc -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"src/Hello.d" -MT"src/Hello.o" -o "src/Hello.o" "../src/Hello.c"
Finished building: ../src/Hello.c

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



Hello World in C

“Run > Run” runs the program



The screenshot shows the Eclipse IDE interface. The main editor displays the source code for 'Hello.c'. The code includes standard headers and a main function that prints '!!!Hello World!!!'. The left sidebar shows a project tree with 'Hello.c' selected. The bottom console window shows the output of the program, which is '!!!Hello World!!!'.

```
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!!! */
16 }
```

Console Output:

```
<terminated> Hello Debug [C/C++ Application] /home/sbrandt/demospace/HelloDebug/Hello (9/29/11 5:02 PM)
!!!Hello World!!!
```



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



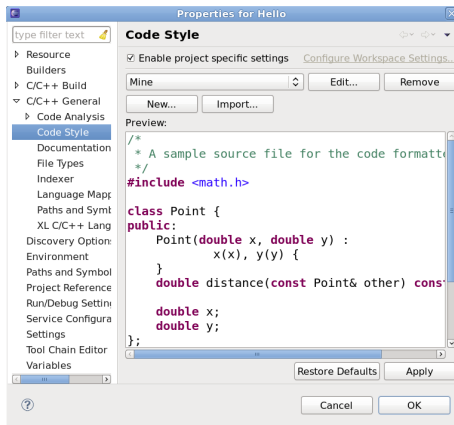
Control Sequences

- 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”




- 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

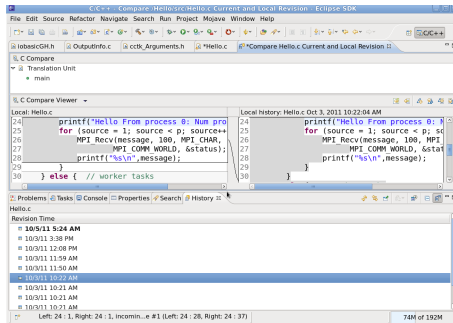


- A perspective is a collection of views. You can change them by going to `Window > Open Perspective...`
- On the right you see the outline view.
- Click the small x and it will go away.
- To bring it back, use `Window > Open View...`

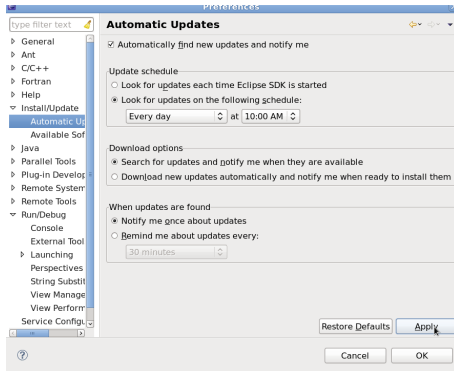


History

- 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



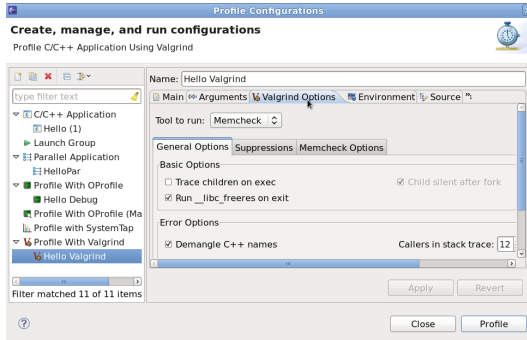
Linux Tools



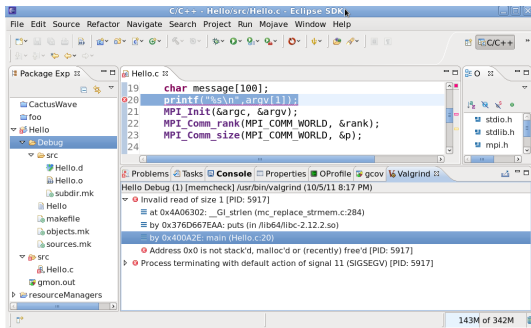
- Make sure “Valgrind Tools Integration” is installed.
- Select ProfilingDelegate Launcher.
- Under Run > Profile Configurations... > C/C++ Application > Application Name > Profiler, select “MemCheck” from the chooser.



LinuxTools: Valgrind



LinuxTools: Valgrind



The screenshot shows the Eclipse IDE with a C++ project named 'Hello'. The Package Explorer on the left shows the project structure. The main editor displays the file 'Hello.c' with the following code:

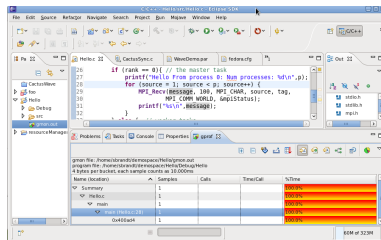
```
19 char message[100];
20 printf("%s\n", argv[1]);
21 MPI_Init(&argc, &argv);
22 MPI_Comm_rank(MPI_COMM_WORLD, &rank);
23 MPI_Comm_size(MPI_COMM_WORLD, &p);
24
```

The Console view at the bottom shows the output of the program, which is 'Hello Debug (1) [memcheck] /usr/bin/valgrind (10/5/11 8:17 PM)'. The output indicates an invalid read of size 1 at address 0x4A06302, which is a stack address. The process is terminating with a signal 11 (SIGSEGV).



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`



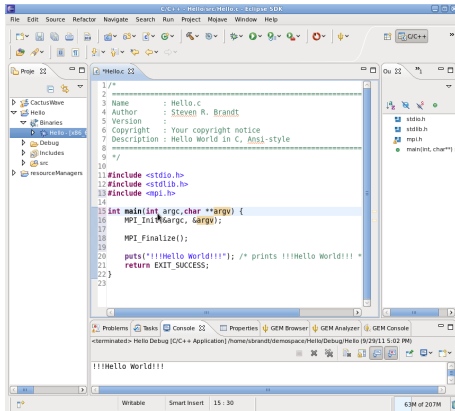
Hello World in C+MPI



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`.



```
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();
19
20     puts("!!!Hello World!!!"); /* prints !!!Hello World!!! */
21     return EXIT_SUCCESS;
22 }
23
```

!!!Hello World!!!



Hello World in C+MPI

Building will now produce errors.

```
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();
19 }
```

Problems Console [Hello]

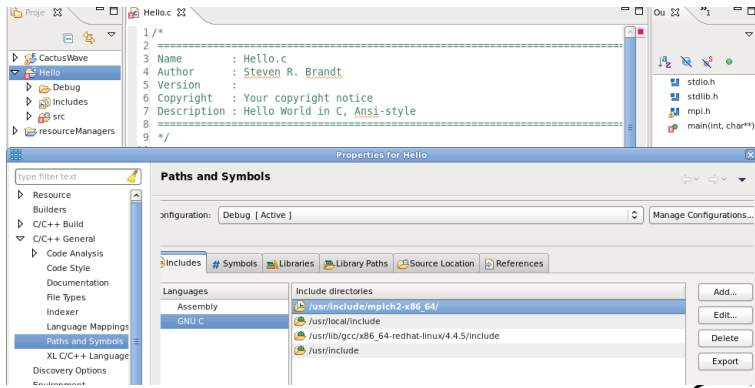
```
make all
Building file: ../src/Hello.c
Invoking: GCC C Compiler
gcc -D0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"src/Hello.d" -MT"src/Hello.o"
-o "src/Hello.o" "../src/Hello.c"
../src/Hello.c:13:17: warning: mpi.h: No such file or directory
../src/Hello.c: In function 'main':
../src/Hello.c:16: 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 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"

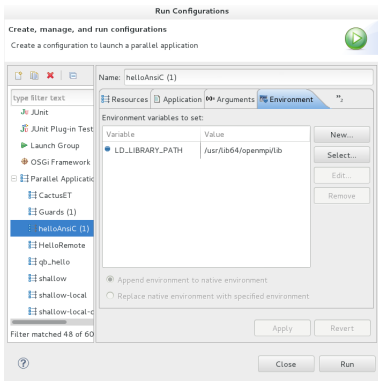
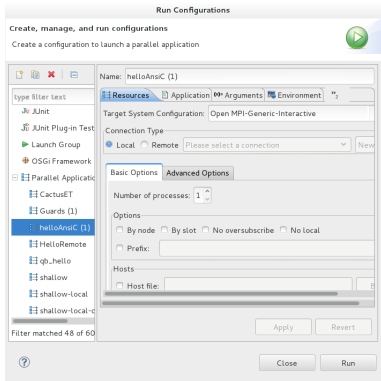


Hello World in C+MPI

- “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.

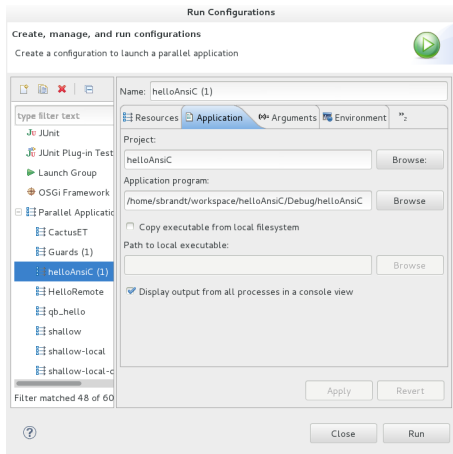


Hello World in C+MPI



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

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

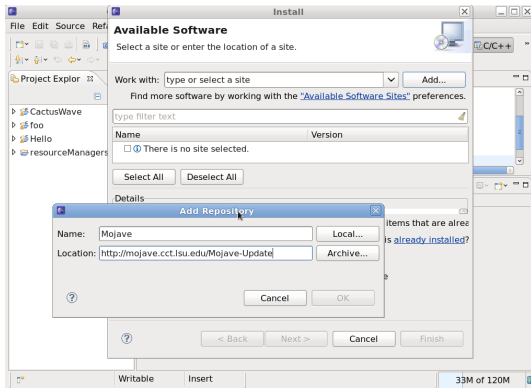


Installing Mojave



Installing Mojave

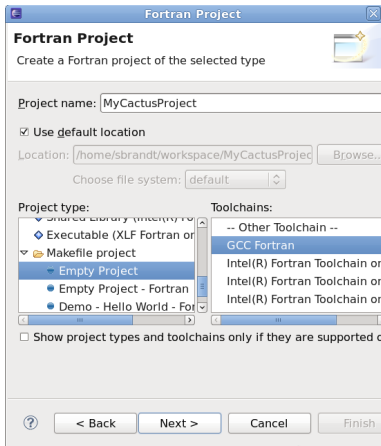
- Help > Install New Software > Add...
- A popup will appear. Fill in the Name and Location
- Hit “OK” then “Next” and accept, etc. on the following screens.



The Wave Equation on your Local Machine



- 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

Fortran Project

New Mojave Project

The Mojave IDE for the Cactus Framework

Welcome to Mojave

☐ Regular C++ Project

☒ Mojave Download Project

User Name (a login name, only letters, numbers, underscore)

sbrandt

Email

sbrandt@cct.lsu.edu

Thorn List

http://mojave.cct.lsu.edu/thornlists/WaveDemo.th

☐ Mojave Disk Project

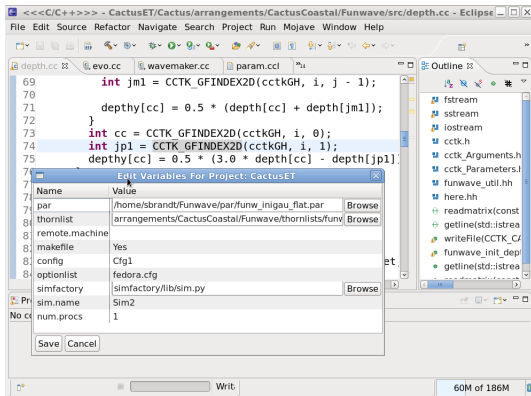
Cactus Install Directory

File

? < Back Next > Cancel Finish

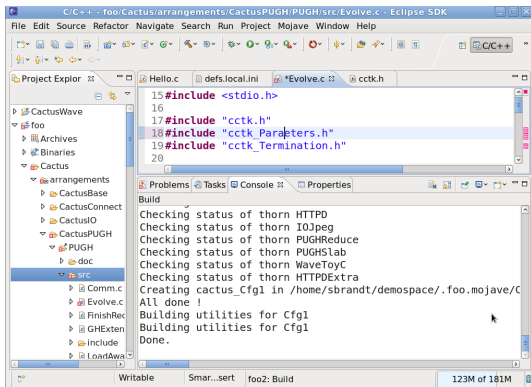


- Now that you've downloaded a project, please configure using the mojave variable editor
- Mojave > Edit Variables...
- Select the basic information needed for a cactus build/run



Mojave

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



```
C/C++ - foo/Cactus/arrangements/CactusPUGH/PUGH/src/Evolve.c - Eclipse SDK
File Edit Source Refactor Navigate Search Run Project Mojave Window Help

Project Explorer
CactusWave
└─ foo
   └─ Archives
      └─ Binaries
         └─ Cactus
            └─ arrangements
               └─ CactusBase
                  └─ CactusConnect
                     └─ CactusIO
                        └─ CactusPUGH
                           └─ PUGH
                              └─ doc
                                 └─ src
                                    └─ Comm.c
                                       └─ Evolve.c
                                          └─ FinishRec
                                             └─ GHExten
                                                └─ Include
                                                   └─ LoadAwa

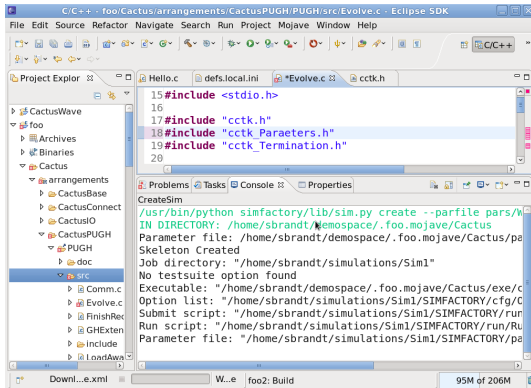
src/Evolve.c
15#include <stdio.h>
16
17#include "cctk.h"
18#include "cctk_Parameters.h"
19#include "cctk_Termination.h"
20

Problems Tasks Console Properties
Build
Checking status of thorn HTTPD
Checking status of thorn IOjpeg
Checking status of thorn PUGHReduce
Checking status of thorn PUGHSlab
Checking status of thorn WaveToyC
Checking status of thorn HTTPDExtra
Creating cactus_Cfg1 in /home/sbrandt/demospace/.foo.mojave/C
All done !
Building utilities for Cfg1
Building utilities for Cfg1
Done.
```



Mojave

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



The screenshot shows the Eclipse IDE interface. The Project Explorer on the left displays the project structure: CactusWave, foo, Archives, Binaries, Cactus, arrangements, CactusBase, CactusConnect, CactusIO, CactusPUGH, PUGH, doc, src, Comm.c, Evolve.c, FinishRec, GHExten, Include, and LoadAw. The main editor window shows the file CactusPUGH/src/Evolve.c, which contains the following code:

```
15#include <stdio.h>
16
17#include "cctk.h"
18#include "cctk_Parameters.h"
19#include "cctk_Termination.h"
20
```

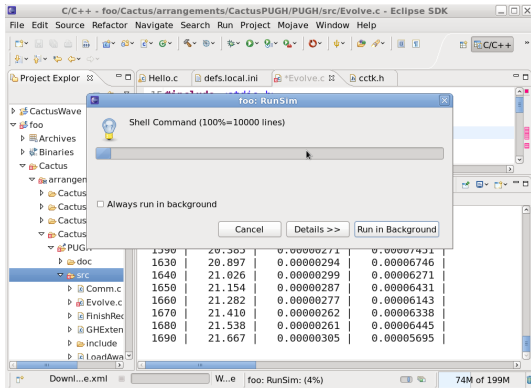
The Console window at the bottom shows the output of the 'CreateSim' command:

```
CreateSim
/usr/bin/python simfactory/lib/sim.py create --parfile pars/W
IN DIRECTORY: /home/sbrandt/demospace/.foo.mojave/Cactus
Parameter file: /home/sbrandt/demospace/.foo.mojave/Cactus/pa
Skeleton Created
Job directory: "/home/sbrandt/simulations/Sim1"
No testsuite option found
Executable: "/home/sbrandt/demospace/.foo.mojave/Cactus/exe/c
Option list: "/home/sbrandt/simulations/Sim1/SIMFACTORY/cfg/C
Submit script: "/home/sbrandt/simulations/Sim1/SIMFACTORY/run
Run script: "/home/sbrandt/simulations/Sim1/SIMFACTORY/run/Ru
Parameter file: "/home/sbrandt/simulations/Sim1/SIMFACTORY/pa
```



Mojave

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



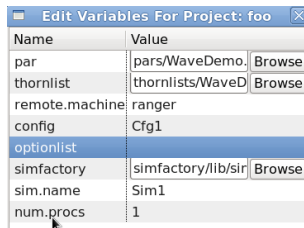
- Now to use Supermike.
- Change variable definitions in Mojave.
- Make sure defs.local.ini is correct (you can edit it in Mojave as well, just use Open Resource: Shift-Ctrl-R and start typing to call up the file)
- My Supermike configuration looks like this:

```
[mike]
```

```
user = sbrandt
```

```
sourcebasedir = /work/@USER@
```

```
basedir = /work/@USER@/simulations
```



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 GetComponent entries  
sync-sources    = <<EOT  
CONTRIBUTORS  
COPYRIGHT  
Makefile  
arrangements  
bin  
lib  
manifest  
repos  
simfactory  
pars  
thornlists  
...
```

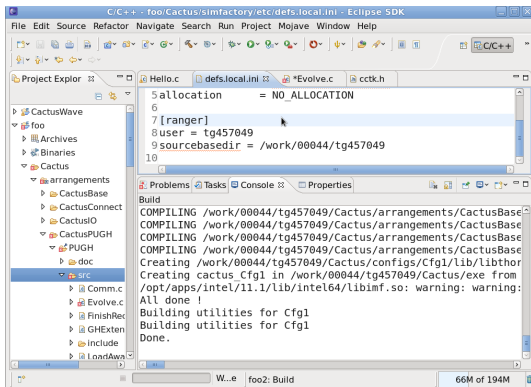


The Wave Equation on Mike



Mojave

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



The screenshot shows the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Run, Project, Mojave, Window, and Help. The Project Explorer on the left shows a project named 'CactusWave' with sub-projects 'Archives', 'Binaries', and 'Cactus'. The 'Cactus' project is expanded, showing 'arrangements', 'CactusBase', 'CactusConnect', 'CactusIO', 'CactusPUGH', and 'PUGH'. The 'src' directory is selected, showing files like 'Comm.c', 'Evolve.c', 'FinishRect', 'GHExten', 'Include', and 'LoadAwn'. The main editor window displays the file 'defs.local.ini' with the following content:

```
5 allocation = NO_ALLOCATION
6
7 [ranger]
8 user = tg457049
9 sourcebasedir = /work/00044/tg457049
10
```

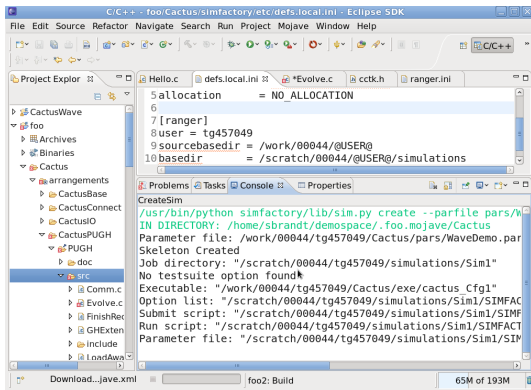
The Problems view at the bottom shows the build output:

```
Build
COMPILING /work/00044/tg457049/Cactus/arrangements/CactusBase
COMPILING /work/00044/tg457049/Cactus/arrangements/CactusBase
COMPILING /work/00044/tg457049/Cactus/arrangements/CactusBase
COMPILING /work/00044/tg457049/Cactus/arrangements/CactusBase
Creating /work/00044/tg457049/Cactus/configs/Cfg1/lib/libthor
Creating cactus.Cfg1 in /work/00044/tg457049/Cactus/exe from
/opt/apps/intel/11.1/lib/intel64/libimf.so: warning: warning:
All done !
Building utilities for Cfg1
Building utilities for Cfg1
Done.
```



Mojave

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



```
C/C++ - foo/Cactus/simfactory/etc/defs.local.ini - Eclipse SDK
File Edit Source Refactor Navigate Search Run Project Mojave Window Help

Project Explorer
CactusWave
  foo
    Archives
    Binaries
    Cactus
      arrangements
      CactusBase
      CactusConnect
      CactusIO
      CactusPUGH
        PUGH
          doc
          src
            Comm.c
            Evolve.c
            FinishRec
            GHExten
            Include
            LoadAwa

defs.local.ini
*Evolve.c
cctk.h
ranger.ini

5 allocation = NO_ALLOCATION
6
7 [ranger]
8 user = tg457049
9 sourcebasedir = /work/00044/@USER@
10 basedir = /scratch/00044/@USER@/simulations

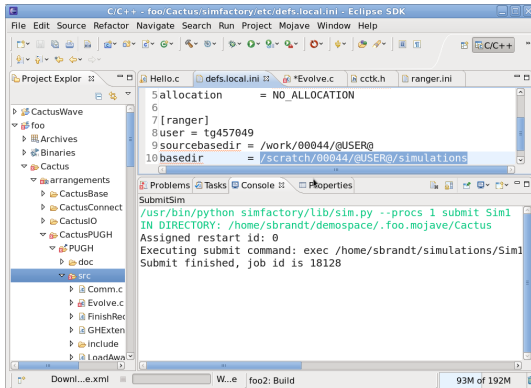
Problems Tasks Console Properties
CreateSim
/usr/bin/python simfactory/lib/sim.py create --parfile pars/W
IN DIRECTORY: /home/sbrandt/demospace/.foo.mojave/Cactus
Parameter file: /work/00044/tg457049/Cactus/pars/WaveDemo.par
Skeleton Created
Job directory: "/scratch/00044/tg457049/simulations/Sim1"
No test suite option found
Executable: "/work/00044/tg457049/Cactus/exe/cactus.Cfg1"
Option list: "/scratch/00044/tg457049/simulations/Sim1/SIMFACT"
Submit script: "/scratch/00044/tg457049/simulations/Sim1/SIMFACT"
Run script: "/scratch/00044/tg457049/simulations/Sim1/SIMFACT"
Parameter file: "/scratch/00044/tg457049/simulations/Sim1/SIMFACT"

Download...java.xml foo2: Build 65M of 193M
```



Mojave

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



The screenshot shows the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Run, Project, Mojave, Window, and Help. The Project Explorer on the left shows a project structure with folders like CactusWave, foo, Archives, Binaries, and Cactus. The main editor window displays the file defs.local.ini with the following content:

```
5 allocation = NO_ALLOCATION
6
7 [ranger]
8 user = tg457049
9 sourcebasedir = /work/00044/@USER@
10 basedir = /scratch/00044/@USER@/simulations
```

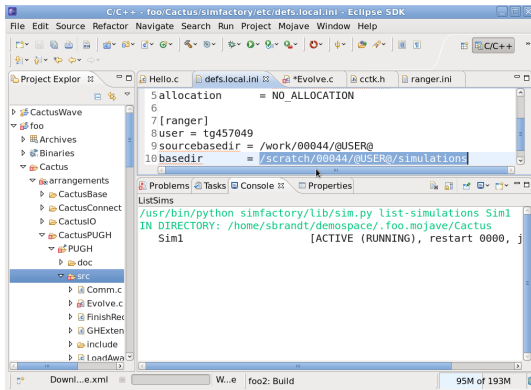
The Properties tab at the bottom shows the output of the SubmitSim command:

```
SubmitSim
/usr/bin/python simfactory/lib/sim.py --procs 1 submit Sim1
IN DIRECTORY: /home/sbrandt/demospace/.foo.mojave/Cactus
Assigned restart id: 0
Executing submit command: exec /home/sbrandt/simulations/Sim1
Submit finished, job id is 18128
```



Mojave

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



The screenshot shows the Eclipse IDE interface. The Project Explorer on the left displays the project structure for 'CactusWave', with 'foo' expanded to show 'src'. The main editor window shows the file 'defs.local.ini' with the following content:

```
5 allocation = NO_ALLOCATION
6
7 [ranger]
8 user = tg457049
9 sourcebasedir = /work/00044/@USER@
10 basedir = /scratch/00044/@USER@/simulations
```

The Console window at the bottom shows the output of the 'ListSims' command:

```
ListSims
/usr/bin/python simfactory/lib/sim.py list-simulations Sim1
IN DIRECTORY: /home/sbrandt/demospace/.foo.mojave/Cactus
Sim1 [ACTIVE (RUNNING), restart 0000, j]
```



Mojave

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

```
C/C++ - foo/Cactus/simfactory/etc/defs.local.ini - Eclipse SDK
File Edit Source Refactor Navigate Search Run Project Mojave Window Help

Project Explorer
  CactusWave
  foo
    Archives
    Binaries
    Cactus
      arrangements
      CactusBase
      CactusConnect
      CactusIO
      CactusPUGH
        doc
        src
          Comm.c
          Evolve.c
          FinishRec
          GHExten
          Include
          LoadAwa

defs.local.ini
  5 allocation = NO_ALLOCATION
  6
  7 [ranger]
  8 user = tg457049
  9 sourcebasedir = /work/00044/@USER@
  10 basedir = /scratch/00044/@USER@/simulations

Problems Tasks Console Properties
ShowSimOutput
49930 640.128 0.0000296 0.00005026
49940 640.256 0.0000298 0.00005175
49950 640.385 0.0000288 0.00005078
49960 640.513 0.0000264 0.00005231
49970 640.641 0.0000271 0.00005256
49980 640.769 0.0000303 0.00005067
49990 640.897 0.0000326 0.00005040
50000 641.026 0.0000288 0.00005108
-----
Done.
Stopping:
Tue Oct 4 11:05:51 PDT 2011
Tue Oct 4 11:05:51 PDT 2011

Download.xml W...e foo2: Build 128M of 188M
```



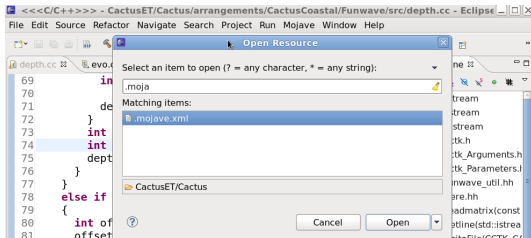
Using the Mojave Menu

- 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



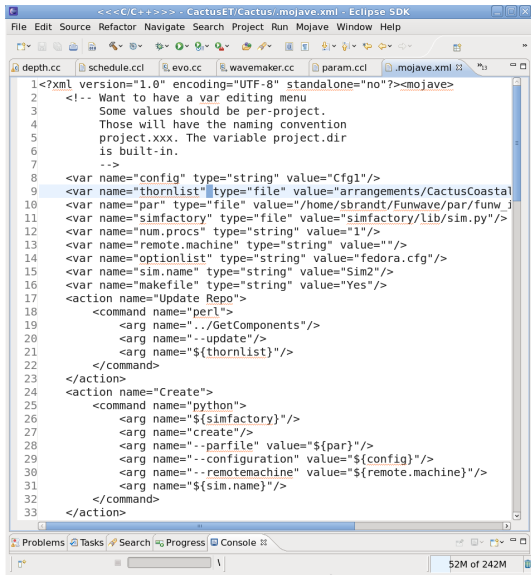
Advanced Configuration of the Mojave Menu

- 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.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.



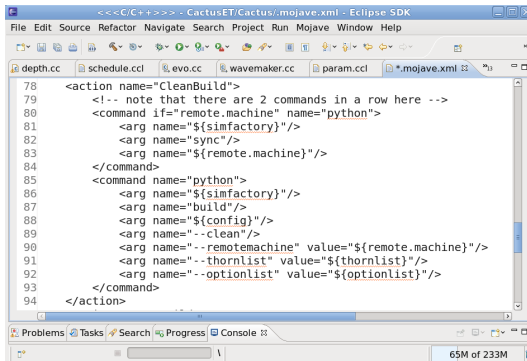


```

1<?xml version="1.0" encoding="UTF-8" standalone="no"?><mojave>
2  <!-- Want to have a var editing menu
3      Some values should be per-project.
4      Those will have the naming convention
5      project.xxx. The variable project.dir
6      is built-in.
7      -->
8  <var name="config" type="string" value="Cfg1"/>
9  <var name="thornlist" type="file" value="arrangements/CactusCoastal
10 <var name="par" type="file" value="/home/sbrandt/Funwave/par/funw_j
11 <var name="simfactory" type="file" value="simfactory/lib/sim.py"/>
12 <var name="num.procs" type="string" value="1"/>
13 <var name="remote.machine" type="string" value=""/>
14 <var name="optionlist" type="string" value="fedora.cfg"/>
15 <var name="sim.name" type="string" value="Sim2"/>
16 <var name="makefile" type="string" value="Yes"/>
17 <action name="Update Repo">
18   <command name="perl">
19     <arg name="../GetComponents"/>
20     <arg name="--update"/>
21     <arg name="${thornlist}"/>
22   </command>
23 </action>
24 <action name="Create">
25   <command name="python">
26     <arg name="${simfactory}"/>
27     <arg name="create"/>
28     <arg name="--parfile" value="${par}"/>
29     <arg name="--configuration" value="${config}"/>
30     <arg name="--remotemachine" value="${remote.machine}"/>
31     <arg name="${sim.name}"/>
32   </command>
33 </action>

```


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



```
<<<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
78 <action name="CleanBuild">
79     <!-- note that there are 2 commands in a row here -->
80     <command if="remote.machine" name="python">
81         <arg name="${simfactory}"/>
82         <arg name="sync"/>
83         <arg name="${remote.machine}"/>
84     </command>
85     <command name="python">
86         <arg name="${simfactory}"/>
87         <arg name="build"/>
88         <arg name="${config}"/>
89         <arg name="--clean"/>
90         <arg name="--remotemachine" value="${remote.machine}"/>
91         <arg name="--thornlist" value="${thornlist}"/>
92         <arg name="--optionlist" value="${optionlist}"/>
93     </command>
94 </action>
```

- 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.

