HelloWorld

Gabrielle Allen

Date: 2002/06/04 12:51:15

Abstract

The Hello World thorn is the simplest possible example of programming an application with the Cactus Framework

1 Purpose

The thorn demonstrates how to implement the typical first example of all programming languages with Cactus — printing the words `Hello World!` to the screen.

2 Tutorial

2.1 Source Code

Cactus allows you to write thorns using any mixture of Fortran 77, Fortran 90, C and C++, in this case there is only one source code routine in the thorn, and we have chosen to write this in C. Later we will see how this routine can be *scheduled* to executed when Cactus is run.

The actual code which prints is in the file

`CactusExamples/HelloWorld/src/HelloWorld.c`

Note that this file is listed in the thorn make file

`CactusExamples/HelloWorld/src/make.code.defn`

which informs Cactus to include this file when building executables.

The `HelloWorld.c` file contains the lines

```
C1.  #include "cctk.h"
C2.  #include "cctk_Arguments.h"
C3
C4.  void HelloWorld(CCTK_ARGUMENTS)
C5.  {
C6.    DECLARE_CCTK_ARGUMENTS
C7.
C8.    CCTK_INFO("Hello World !");
C9.
C10.   return;
C11. }
```

Lets go through this line by line, and compare it to a standalone Hello World program written to match the lines of the Cactus routine.

```
S1.  #include <stdio.h>
S2.
S3.
S4.  int main()
S5.  {
S6.
S7.
S8.    printf("Hello World !\n");
S9.
S10.   return 0;
S11. }
```

Include Files (C1/C2): All source files which make use of Cactus infrastructure need to include `cctk.h`, this sets up the Cactus data types and prototypes Cactus functions. All scheduled functions also need to include `cctk_Arguments.h` which contains information about the data variables which are available to the function.

Function Name (C4): Scheduled functions are always void, and the argument contains the macro `CCTK_ARGUMENTS` which is expanded during compilation to contain information about the data variable available to the function. In our simple case there are actually no data variables.

Arguments Declaration (C6): This line defines any data variables available to the routine, from this thorn and inherited from other thorns.

Cactus Info (C8): Finally, the line to print to screen. We could have just used `printf("Hello World !\n")` here as in the standalone function, but here we actually use a Cactus function `CCTK_INFO` which formats output to screen to include the name of the thorn. That isn't so useful here, but in cases where there are many thorns it helps for understanding the screen output. Also, for more complicated applications running in parallel, the `CCTK_INFO` function controls which of the processors actually writes to screen (having 1024 processors each writing `Hello World !` could look confusing!).

3 Configuration Files

Cactus Configuration (or `CCL`) files are the way in which the Cactus Flesh find out information about your thorn. Cactus isn't interested in all the private stuff of your thorn, but it needs to know about things like variables, routines, parameters which will interface with other thorns.

In the Hello World example these files are trivial, since we have no variables, only one routine to schedule, and no parameters (to learn about parameters you could try to add a parameter to this thorn to switch off the output, or to change the number of times the message is printed). The contents of our CCL files are

interface.ccl

```
implements: helloworld
```

Each thorn has to give a name to what it does, here we say that we do is called "helloworld". (If you look further into the notion of implementations, you will find that this mechanism allows you to switch between different thorns which provide the same implementation, but for our example this isn't very useful).

param.ccl

There are no parameters in the HelloWorld thorn, so this file is empty.

schedule.ccl

```
schedule HelloWorld at CCTK_EVOL
{
  LANG: C
} "Print message to screen"
```

This file instructs Cactus to run the routine called HelloWorld in our thorn during the evolution timebin. Additionally it tells Cactus that the source codee is written in C, and provides a descipion of what the routine does.

4 Screen Output

Here we list the actual output from running the Hello World parameter file. Note that as we develop the framework this output may look a little different by the time you see it.

```
[allen@gullveig CactusClean]$ ./exe/cactus_world arrangements/CactusExamples \
  /HelloWorld/par/HelloWorld.par
--------------------------------------------------------------------------------
       10
  1   0101       ************************
  01  1010 10      The Cactus Code V4.0
 1010 1101 011      www.cactuscode.org
  1001 100101    ************************
    00010101
     100011     (c) Copyright The Authors
      0100      GNU Licensed. No Warranty
      0101

--------------------------------------------------------------------------------
Cactus version: 4.0.b12
Compile date:   May 01 2002 (10:26:44)
Run date:       May 01 2002 (10:28:35)
Run host:       gullveig.aei.mpg.de
Executable:     ./exe/cactus_world
Parameter file: arrangements/CactusExamples/HelloWorld/par/HelloWorld.par
--------------------------------------------------------------------------------
Activating thorn Cactus...Success -> active implementation Cactus
Activation requested for
--->HelloWorld<---
Activating thorn HelloWorld...Success -> active implementation helloworld
--------------------------------------------------------------------------------
  if (recover initial data)
    Recover parameters
  endif

  Startup routines

  Parameter checking routines

  Initialisation
    if (NOT (recover initial data AND recovery_mode is 'strict'))
    endif
    if (recover initial data)
    endif
    if (checkpoint initial data)
```

```
    endif
    if (analysis)
    endif
    Do periodic output of grid variables

  do loop over timesteps
    Rotate timelevels
    iteration = iteration + 1
    t = t+dt
    HelloWorld: Print message to screen
    if (checkpoint)
    endif
    if (analysis)
    endif
    Do periodic output of grid variables

  do loop over timesteps
    Rotate timelevels
    iteration = iteration + 1
    t = t+dt
    HelloWorld: Print message to screen
    if (checkpoint)
    endif
    if (analysis)
    endif
    Do periodic output of grid variables
  enddo

  Termination routines

  Shutdown routines
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
INFO (HelloWorld): Hello World !
INFO (HelloWorld): Hello World !
INFO (HelloWorld): Hello World !
INFO (HelloWorld): Hello World !
INFO (HelloWorld): Hello World !
INFO (HelloWorld): Hello World !
INFO (HelloWorld): Hello World !
INFO (HelloWorld): Hello World !
INFO (HelloWorld): Hello World !
INFO (HelloWorld): Hello World !
--------------------------------------------------------------------------------
Done.
```