GiDpost

Download

Please click here in order to download the library.

Introduction

GIDpost is a set of functions (library) for writing postprocess results for GiD in ASCII or binary format. GiD version 6.1.4b or higher is required to read the ASCII postprocess files. GiD version 7.2 or higher is required to read the binaries postprocess files.

 This description asumes that the readear is familiar with the postprocess terminology. For futher details please check the online help available in GiD (Postprocess data files chapter).

The library was implemented taking into account two of the must widely used development environments: C/C++ and FORTRAN.

Here we are going to describe how to compile and use the library. At the end the reference of the library functions can be found.

Compiling

- **Unix platform:**
  A `makefile` is provided inside the `unix` directory. Invoke `make` to create the library:

  ```
  make [target=release|debug]
  ```

  This will create the library `unix/release/gidpost.a` if the option is `target=release` or `unix/debug/gidpostd.a` if the option is`target=debug`. If no option is provided the release version is created by default.

- **Microsoft Windows platform:**
  A file named `Makefile.vc` is provided inside `gidpost\win`to compile with Microsoft Visual C++. To compile, open a DOS console, change the current directory to `gidpost\win`. First run vcvars32.bat prior namake execution to ensure enviroment PATH is ok. Then invoke:

  ```
  nmake -f Makefile.vc [CFG="Release"|"Debug"]
  ```

  This will create the library `gidpost.lib` inside the subdirectory `win\Release` if the option is `CFG="Release"` or inside `win\Debug`) if the option is `CFG="Debug"`. If no

option is provided the release version is created by default.

- **Other platforms:**
  All the source code to generate the library is included, it can be compiled in other platforms.
  The third party code, "`cfortran.h`", is also provided as a link between C and FORTRAN. It has its own distribution policy. Please, read the file "`cfortran.doc`" about the licence terms of this code.

Using the library

- **C / C++ language:**
  Include the file header file `gidpost.h` to use the libray `gidpost`

  ```
  #include "gidpost.h"
  ```

  A small example, called `testpost.c`, is provided to show the use of the library from C/C++.

- **FORTRAN language:**
  A small example, called `testpostfor.f`, is provided to show the use of the library with FORTRAN.

You can link your code either with the release or debug version of the library. In the release version there is almost no check in the use of the library in order to provide no overhead in using it. If link your code with the debug version then you will get some extra checking in the calling of the functions and you will get more information if something is wrong but with an extra overhead in time.

Functions references

Mesh file functions

```
int GiD_OpenPostMeshFile(char* FileName,GiD_PostMode Mode);

Description: Open a new post mesh file

Parameters:

char* FileName
      name of the mesh file (*.flavia.msh)
GiD_PostMode Mode
      GiD_PostAscii=0 for ascii output
      GiD_PostAsciiZipped=1 for compressed ascii output
```

GiD_PostBinary=2 for compressed binary output

Example:
C/C++
```
GiD_OpenPostMeshFile( "testpost.flavia.msh", GiD_PostAscii);
```
FORTRAN
```
CALL GID_OPENPOSTMESHFILE('testpost.flavia.msh',0)
```

```
int GiD_ClosePostMeshFile();
```

Description: Close the current post mesh file

Parameters:
None

Example:
C/C++
```
GiD_ClosePostMeshFile();
```
FORTRAN
```
CALL GID_CLOSEPOSTMESHFILE
```

```
int GiD_BeginMeshGroup(char* Name);
```

Description: This function open a group of mesh.
This enable specifying multiples meshes withing the group.

Parameters:
char* Name

Name of the group. This name can be used later when givin the set of results that
apply to this group, see GiD_OnMeshGroup.

Example:
C/C++

```
GiD_BeginMeshGroup("steps 1, 2, 3 and 4" );
```

FORTRAN

```
CALL GID_BEGINMESHGROUP("steps 1, 2, 3 and 4" )
```