# Debugging brief info

William Oquendo

## Contents

## 1 General Info

**Check the wikis at :** `https://bitbucket.org/iluvatar/scientific-computing-part-01/wiki/Debugging`

## 2 GDB (Gnu debugger)

### 2.1 Interfaces (graphical)

#### 2.1.1 Tui mode

gdb -tui mode: Ncurses gui

#### 2.1.2 Inside emacs

When having a file openned in emacs, press `M-x` and write gdb, press enter. You will be presented with an example command to run gdb, modify it as you wish. Then, emacs will be yor debugger, and you can move from frame to frame by using the cursor or `C-x o` .

#### 2.1.3 DDD : Data Display Debugger

Although dated, its great strenght is the ability to plot data with the command `grap plot dataarrayname` , and check its evilution as the you debug.

#### 2.1.4 Others

Codeblocks, geany, clion, etc all them have interfaces for gdb.

### 2.2 Modern tools

#### 2.2.1 Python

Gdb version >= 7.x supports python. You can use to explore more info about your program, pretty print data, and even plot (although blocking).

To load a c++ var into a python do, inside gdb,

```
py my_array = gdb.parse_and_eval("my_array")
py print my_array.type
```

Even more, to plot some data structure with matplotlib, you can as follows (Based on `https://blog.semicolonsoftware.de/debugging-numerical-c-c-fortran-code-with-gdb-and-numpy/`)

```c
int main(int argc, char **argv) {
  const int nrow = 4;
  const int ncol = 3;
  double x[nrow * ncol];
  for(int i = 0; i < nrow; i++) {
    for(int j = 0; j < ncol; j++) {
  x[i * ncol + j]  = i * j;
    }
  }
  // BREAK here
}
```

Compile as

```
gcc ex-01.c -g -ggdb
```

and now, run gdb and put the following commands

```
py import gdb_numpy
py x = gdb_numpy.to_array("(double *)x", shape=(4 * 3,)).reshape(4, 3)
py print x
py import matplotlib.pyplot as plt
py plt.imshow(x)
py plt.show()
```