|  |  |
| --- | --- |
|  |  |

GDB Improver for IptNMS Circuit

1. **Intro**

This tool is meant to be an integration to GNU gdb and debuggers that use it as backend (ddd, Eclipse, QtCreator, ….) with the intention of simplify debugging of IptNMS Circuit Server. Main difficulties arise by the intrinsic complexity of objects and structures defined in database (Objectivity) and various interfaces used by processes (I38, I36, ICHRH,….). It’s very hard to get useful information by raw data printed by gdb. The work done by the tool is to extract all “informative” data for interfaces structures and some details (names, ids) for databases objects . Means used are python API available by gdb (pretty printing, custom commands,…)

1. **Preliminary Checks**

An important fact to underline is that use of this tool is intended in PIEDE vm environment (not the HUB).

The fundamental thing to check to have a gdb Improver working is (besides the presence of gdb itself) support for python API. This check can be done launching gdb executable and giving the command from the console:

*show configuration*

If the output include a string of kind “—with-python=…” it’s ok. Otherwise it’s necessary to get another gdb build with this support.

1. **Installation**

The whole package must be unpacked in the python gdb directory. Currently it is “/usr/share/gdb/python”, but in future could be changed, so it’s necessary to see it giving the following command in gdb console:

*python print gdb.PYTHONDIR*

The next step is to modify .gdbinit configuration file (read at startup) relative to the superuser (**because gdb must be launched as root**) to include the following scriptlet:

*python*

*scriptPath = gdb.PYTHONDIR + "/CCDbgImprover/main.py"*

*command = "source " + scriptPath*

*gdb.execute(command)*

*end*

This is necessary to automate sourcing of gdb Improver python scripts at startup.

1. **Preparation for Use**

Gdb needs debug information embedded in executable files and libraries to work. The fundamental step is to compile statically the process. In this way you have debug information for process itself and for libraries relative to database and interfaces under debug in one executable. For example, if you want debug I38 structures the reference library is:

libI38.so.1 (found in /home/nmsdev/GERRIT/IptNMS-C/mv38\_nmdev/dev/NMDEV/LIBMSG/I38).

The reference library for database objects, instead, is

libooDnNm.so.1 (found in /home/nmsdev/GERRIT/IptNMS-C/OO38)

Obviously the target executable must be copied in the installation path. Moreover, Gdb Improver needs debug information for at sure two basic libraries. They are:

1. libKernelCommon.so.1 (found in /home/nmsdev/GERRIT/IptNMS-C/mv38\_ext\_lib/dev/NMCOM/common/)
2. libKernelSYS.so.1 (found in /home/nmsdev/GERRIT/IptNMS-C/mv38\_ext\_lib/dev/NMCOM/SYS/)

They must be copied in the library directory of the installation, too.

These steps are important because executables and libraries supplied by rpm Circuit installation don’t have enough debug information.

A confirm that executables and libraries self-compiled have debug info embedded is that executing the following command on any object file (one of \*.o files) that composes executable or library

*readelf -S <object file> | grep debug*

you have some lines like these:

[ 4] .debug\_abbrev PROGBITS 00000000 000034 000269 00 0 0 1

[ 5] .debug\_info PROGBITS 00000000 00029d 001541 00 0 0 1

[ 6] .rel.debug\_info REL 00000000 002a90 0007c8 08 15 5 4

[ 7] .debug\_line PROGBITS 00000000 0017de 0002cc 00 0 0 1

[ 8] .debug\_pubnames PROGBITS 00000000 001aaa 00002c 00 0 0 1

[ 9] .rel.debug\_pubnam REL 00000000 003258 000008 08 15 8 4

[10] .debug\_str PROGBITS 00000000 001ad6 000b28 01 MS 0 0 1

Pay attention to set permission to “555” (with chmod) and user:group to “nmcman:nmc” (with chown) to all processes and libraries copied.

The check for the correct importation of the tool is the gdb command:

*info pretty-printer*

The output must list the following series of printers:

global pretty-printers:

Basic Types Printers

BitStringPrinter

BoolPrinter

C++StdStringPrinter

ChStringPrinter

FloatPrinter

IntTypePrinter

NullTypePrinter

OctectStringPrinter

General Composite Printer

I36Printer

I38Printer

ICHRHPrinter

IHFeederrinter

ILHPrinter

LMPrinter

Reference Types Printer

1. **Use with gdb**

The tool is usable launching a new gdb command named “*powerprint*” as:

*powerprint [object]*

Object specified can be:

* a simple var (int,char,….) : in this case *powerprint* is equivalent to the traditional gdb print; in this category there are reference c++ objects (the tool prints the referenced object) and c++ std::string;
* a database object (handle, iterator, named iterator): the tool prints the object name and its id;
* an interface object: the tool prints every member in the structure, with name and contents;

Printing of interface objects makes use of pretty printers defined in python scripts of the tool. They can be listed with the command shown above (*info pretty-printer)*.

Pretty printers can be enabled/disabled with commands:

*enable/disable pretty-printer global <name>*

The “global” parameter refers to the fact that printers inside this environment are handled globally and not for single programs or objects. The <name> parameter is complete name in the form “<printer name>;<subprinter name>” . Printer and subprinter names can be distinguished by the different level of indentation in the output of *info pretty-printer.* So, for example, taking into account the list above, “boolPrinter” is a subprinter of “Basic Types Printers” printer; if you want to disable it, it’s necessary to give the command:

*disable pretty-printer global Basic;boolPrinter*

(<printer name> and <subprinter name> can be even subsets of complete names, like “Basic” is a univocal subset of the complete name “Basic Types Printers”)

Pretty printers can be totally deregistered in gdb using the simple script “remove-pretty-printers” contained in the main directory of the tool (/usr/share/gdb/python/CCDbgImprover) and giving the following command:

*source /usr/share/gdb/python/CCDbgImprover/remove-pretty-printers.py*

1. **Use with graphical debuggers**

The use with other graphical debuggers like ddd and Eclipse, for example, is identical because it’s sufficient to use gdb console embedded; so commands are the same.