gdb调试插件使用说明

目录

gdb调试corefile是每个程序员的必备能力，但原始gdb工具对stl库的支持不好，一大堆字符串堆在一起，可读性太差，在此安利一个小插件，可以更为方便的查看各种常用数据结构

网上的轮子

https://sourceware.org/gdb/wiki/STLSupport

本文介绍的是**gdb-stl-views**，为了防止后续网页不可用，插件附在此处

stl-views-1.0.3.gdb
17KB

安装

看一下自己的机器上是否有~/.gdbinit文件，没有的话把上述文件拷过去，改成这个名字

就！安！装！完！了！

本身有这个文件的话，相信你看完这篇文章，也就知道怎么处理了

使用

1. 使用gdb加载corefile
2. 运行

```
source ~/.gdbinit
```

3. 运行下列命令查看相关变量即可

http://www.yolinux.com/TUTORIALS/GDB-Commands.html#STLDEREF

De-Referencing STL Containers:

Displaying STL container classes using the GDB "`p variable-name`" results in an cryptic display of template definitions and pointers. Use the following `~/.gdbinit` file (V1.03 09/15/08). Now works with GDB 4.3+.
(Archived versions: [V1.01 GDB 6.4+ only])
Thanks to Dr. Eng. Dan C. Marinescu for permission to post this script.

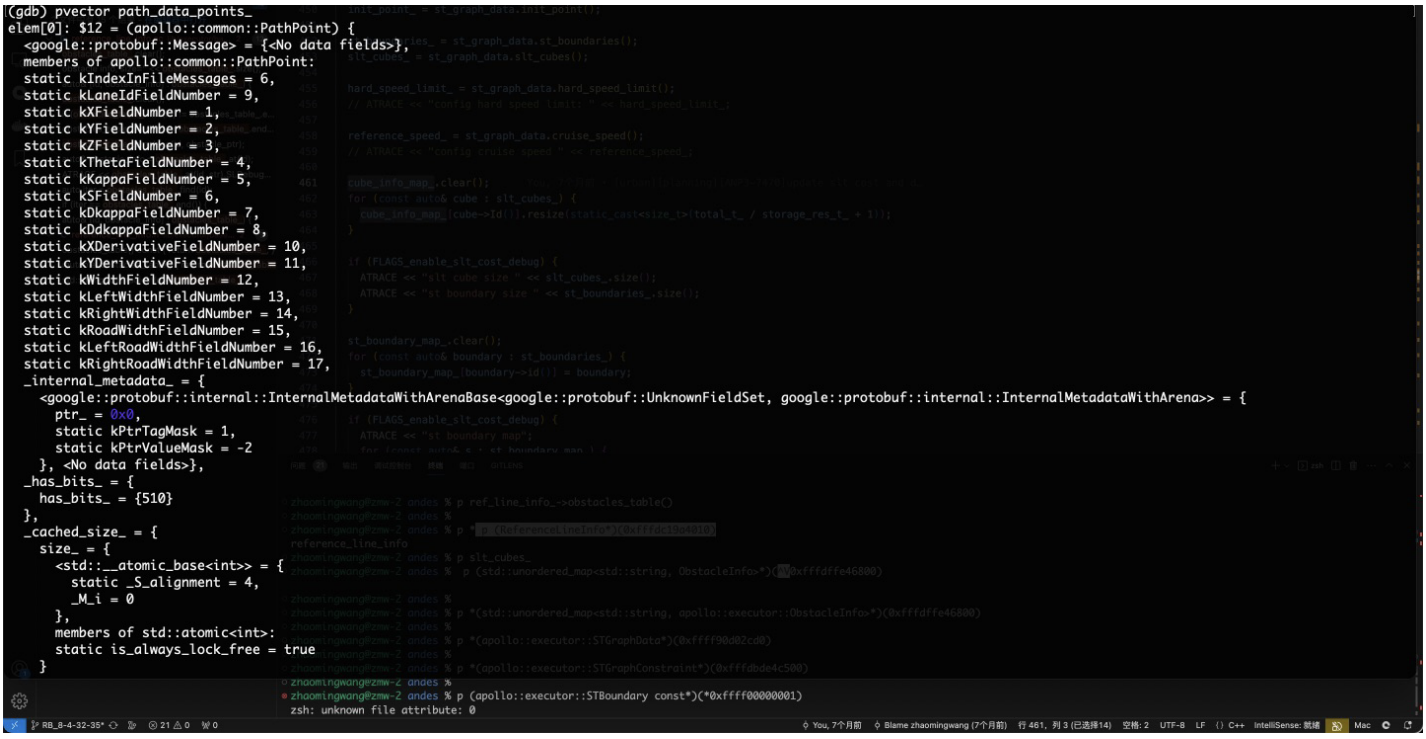
Use the following commands provided by the script:

Data type	GDB command
std::vector<T>	pvector *stl_variable*
std::list<T>	plist *stl_variable* T
std::map<T,T>	pmap *stl_variable*
std::multimap<T,T>	pmap *stl_variable*
std::set<T>	pset *stl_variable* T
std::multiset<T>	pset *stl_variable*
std::deque<T>	pdequeue *stl_variable*
std::stack<T>	pstack *stl_variable*
std::queue<T>	pqueue *stl_variable*
std::priority_queue<T>	ppqueue *stl_variable*
std::bitset<n>td>	pbitset *stl_variable*
std::string	pstring *stl_variable*
std::widestring	pwstring *stl_variable*

Where T refers to native C++ data types. While classes and other STL data types will work with the STL container classes, this de-reference tool may not handle non-native types.
Also see the YoLinux.com STL string class tutorial and debugging with GDB.

举个栗子

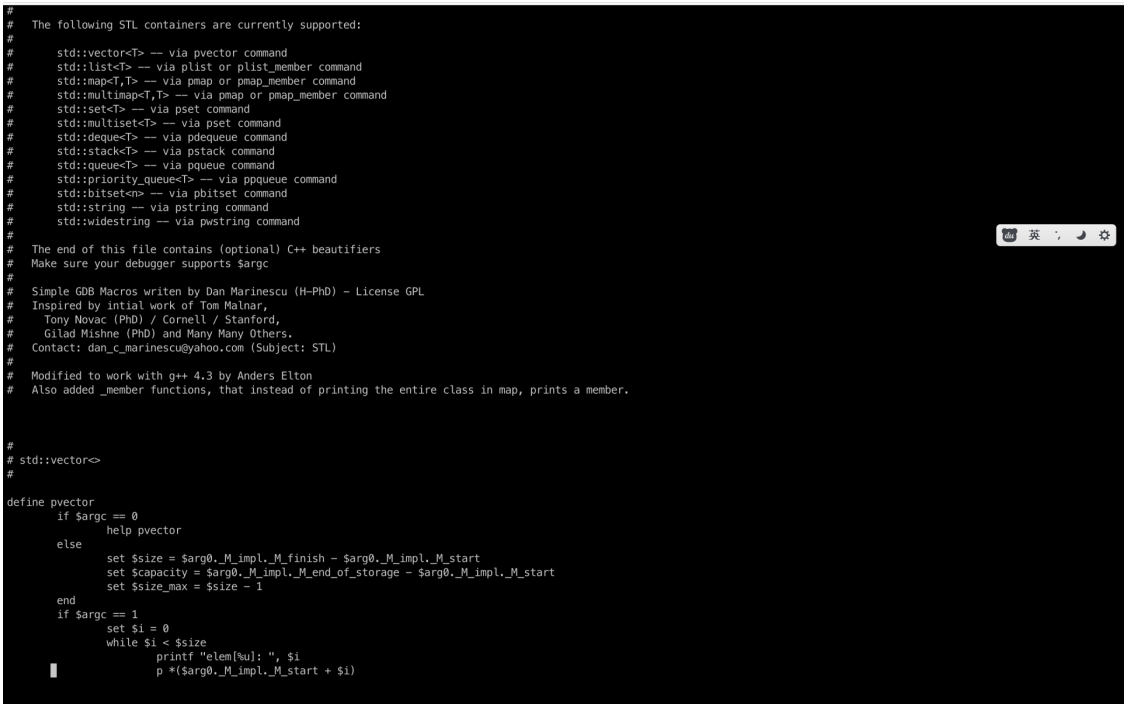
使用pvector命令打印个vector变量，可以看到可读性明显提升



写在最后

这个插件实际就是个文本文件

用vim打开，可以发现里面就是一些脚本函数，实际上就是依赖这些内容对gdb原始的变量打印做字符串处理和格式化输出，提升可读性



```
#
#   The following STL containers are currently supported:
#
#       std::vector<T> -- via pvector command
#       std::list<T> -- via plist or plist_member command
#       std::map<T,T> -- via pmap or pmap_member command
#       std::multimap<T,T> -- via pmap or pmap_member command
#       std::set<T> -- via pset command
#       std::multiset<T> -- via pset command
#       std::deque<T> -- via pdequeue command
#       std::stack<T> -- via pstack command
#       std::queue<T> -- via pqueue command
#       std::priority_queue<T> -- via ppqueue command
#       std::bitset<n> -- via pbitset command
#       std::string -- via pstring command
#       std::widestring -- via pwstring command
#
#   The end of this file contains (optional) C++ beautifiers
#   Make sure your debugger supports $argc
#
#   Simple GDB Macros writen by Dan Marinescu (H-PhD) - License GPL
#   Inspired by intial work of Tom Malnar,
#     Tony Novac (PhD) / Cornell / Stanford,
#     Gilad Mishne (PhD) and Many Many Others.
#   Contact: dan_c_marinescu@yahoo.com (Subject: STL)
#
#   Modified to work with g++ 4.3 by Anders Elton
#   Also added _member functions, that instead of printing the entire class in map, prints a member.



#
# std::vector<>
#

define pvector
        if $argc == 0
                help pvector
        else
                set $size = $arg0._M_impl._M_finish - $arg0._M_impl._M_start
                set $capacity = $arg0._M_impl._M_end_of_storage - $arg0._M_impl._M_start
                set $size_max = $size - 1
        end
        if $argc == 1
                set $i = 0
                while $i < $size
                        printf "elem[%u]: ", $i
                        p *($arg0._M_impl._M_start + $i)
```

所以，大家有余力的可以尝试自己定制，一起提升我们的工具链能力