[Google Heap Profiler使用方法](http://blog.csdn.net/jhzhou/article/details/7245992)

标签： [profiler](http://www.csdn.net/tag/profiler)[google](http://www.csdn.net/tag/google)[postscript](http://www.csdn.net/tag/postscript)[allocation](http://www.csdn.net/tag/allocation)[graphviz](http://www.csdn.net/tag/graphviz)[工具](http://www.csdn.net/tag/%e5%b7%a5%e5%85%b7)

2012-02-10 10:12 7801人阅读 [评论](http://blog.csdn.net/jhzhou/article/details/7245992#comments)(1) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/jhzhou/article/details/7245992#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

内存泄露（2） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg google perf tool Google Heap Profiler C/C++（1） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得转载。

目录[(?)[+]](http://blog.csdn.net/jhzhou/article/details/7245992)

最近在查找内存泄露的问题，使用了一些工具来查找问题，定位问题，但是工具都有各自的优劣，在我们实际的使用过程当中，只有各种工具结合使用才能发挥最大的效用。

由于项目需要，最近也开始使用Google Heap Profiler 来帮助定位内存泄露的问题。

这里是它的详细介绍，请参考<http://google-perftools.googlecode.com/svn/trunk/doc/heapprofile.html>

**功能及原理**

先来大概介绍下Google Heap Profiler，大致有三类功能：

* 可以分析出在程序的堆内有些什么东西
* 定位出内存泄露
* 可以让我们知道哪些地方分配了比较多的内存

大概的原理就是使用tcmalloc 来代替malloc calloc new等等，这样Google Heap Profiler就能知道内存的分配情况，从而分析出内存问题。

**用法**

首先需要把tcmalloc链接到我们需要分析的程序中， 当然我们也可以动态load 这个lib，但是为了简单起见，还是推荐大家链接这个lib到自己的程序中。

链接之后，我们接下来的任务就是得到内存分析的dump文件，我们有两种方法：

     1.    静态dump方法：

         直接定义一个环境变量HEAPPROFILE 来指定dump profile文件的位置，如：/tmp/test.log,它将会在/tmp/目录下生成很多类似/tmp/test.log.0003.heap文件名的文件

         env HEAPPROFILE="/tmp/test.log" /test/testprog

     2.    动态dump方法：

我们可以调用Google Heap Profiler的API来控制什么时候dump出内存的profiler文件，这样更加灵活，为此，我们必须包含heap-profiler.h这个头文件。

HeapProfilerStart() 用来开始内存分析

HeapProfilerStop().  用来终止内存分析

这样就只会在开始和结束之间产生dump profiler文件。

**选项**

* HEAP\_PROFILE\_ALLOCATION\_INTERVAL

程序内存每增长这一数值之后就dump 一次内存，默认是1G （1073741824）

* HEAP\_PROFILE\_INUSE\_INTERVAL

程序如果一次性分配内存超过这个数值dump 默认是100K， 待验证

**查看内存dump文件**

这么dump文件生成之后，我们接下来就可以查看内存的分布情况，如：

pprof --pdf /test/testProg /tmp/test.log.0001.heap

就是以pdf的形式来显示这个dump文件，当然我们也可以使用其他的格式来显示。

   --text              Generate text report  
   --callgrind         Generate callgrind format to stdout  
   --gv                Generate Postscript and display  
   --evince            Generate PDF and display  
   --web               Generate SVG and display  
   --list=<regexp>     Generate source listing of matching routines  
   --disasm=<regexp>   Generate disassembly of matching routines  
   --symbols           Print demangled symbol names found at given addresses  
   --dot               Generate DOT file to stdout  
   --ps                Generate Postcript to stdout  
   --pdf               Generate PDF to stdout  
   --svg               Generate SVG to stdout  
   --gif               Generate GIF to stdout  
   --raw               Generate symbolized pprof data (useful with remote fetch)

这就是所有可支持的格式。

*注：如果pprof 运行出错，请检查时候已经正确安装，如果出现sh: dot: command not found 这个错误，就是需要安装yum install graphviz -y*

我们也可以专门focus在一些包含某些关键字的路径上，也可以忽略相关的路径

--focus

--ignore

pprof --pdf --focus=CData /test/testProg /tmp/test.log.0001.heap

**比较dump文件**

为了知道在某一段时间内的内存分布情况，或者需要了解某段时间内有没有内存泄露，我们就需要用到diff我们的dump文件

例如：pprof --pdf --base /tmp/test.log.0001.heap /test/testProg /tmp/test.log.0101.heap

比较了第一个dump文件与第101个文件的差异，而且结果以pdf的形式显示