# 使用gdb调试nginx

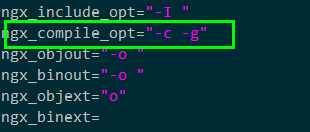
调试nginx能够更方便的了解nginx的运行状态，从而达到熟悉nginx的目的。工欲善其事，必先利其器。君子性非异也，善假于物也。在linux下，可以选择使用gdb器调试nginx。下面我们简单的介绍下如何使用gdb调试nginx。

## gdb的基本命令

attach, run, continue, bt, frame, break, next, step, print, list, delete, info等命令不再做详细介绍，具体使用可以是用help attach命令进行查看，或者直接百度一下。实际上，平时用到的也不多，边用边学，边学边用就好了。

## 编译nginx

NOTE: 在编译nginx的时候，修改auto/cc/conf文件，增加-g编译选项，从而增加gdb的调试信息。如下图所示：

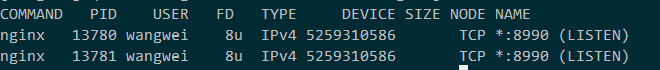


编译nginx，生成可执行文件。

## 利用gdb调试nginx

（STEP 1）为了调试方便，我们把nginx的子进程数设置为1个，这样，nginx在启动的时候就会生成1个master进程和一个worker进程，因此，我们的所有请求都会有这1个worker进程去处理。

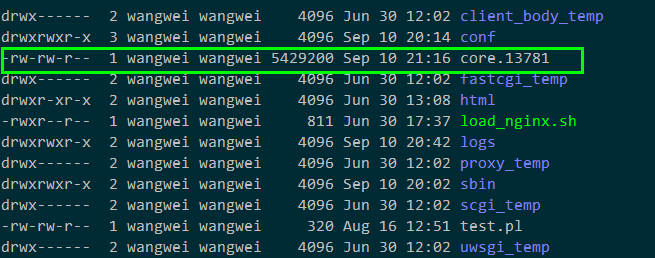
利用/usr/sbin/lsof –i:8990查看，发现nginx启动了2个进程。



cat logs/nginx.pid发现主进程号为13780，因此nginx的worker进程为13781。

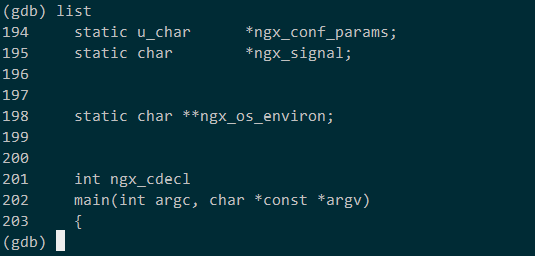
此处可以用ps aux | grep nginx来查看nginx的启动情况，由于我的机器上开启了6个nginx，因此直接使用lsof命令去查看监听端口的进程信息。

（STEP 2）gcore 13781生成core文件，如下图所示：

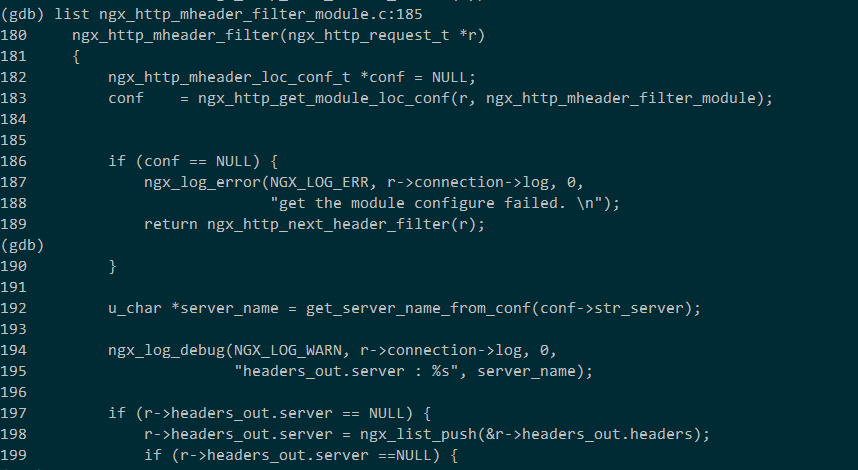


（STEP 3）停止nginx：./sbin/nginx –s stop

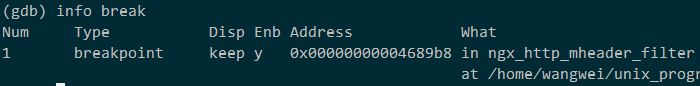
(STEP 4) gdb ./sbin/nginx core.13781 进入gdb调试模式。



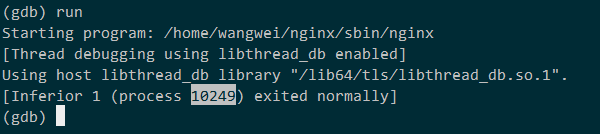
（STEP 5）在需要设置断点的地方设置断点，break的使用请查看相关资料。此处，以调试我写的扩展模块ngx\_http\_mheader\_filter\_module.c为例子进行调试。首先查看该模块的相关信息，如下图：



为了在进入进入ngx\_http\_mheader\_filter函数之后，查看该函数的执行情况，使用如下命令在该模块的182行设置一个断点： break ngx\_http\_mheader\_filter\_module:182。

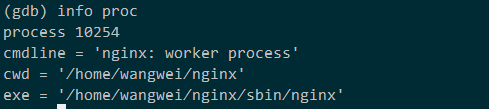


（STEP 6）使用run命令在gdb中运行nginx

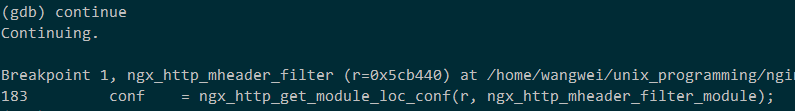


此时利用ps aux | grep nginx命令应该可以发现，nginx已经启动了。我的nginx的子进程为10254。

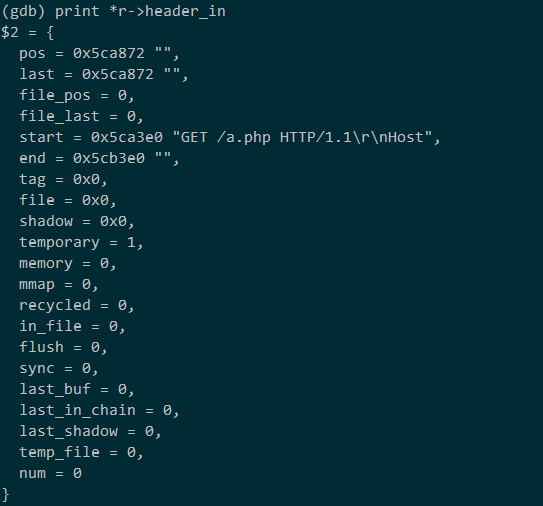
（STEP 7）利用attach命令把nginx的worker进程attach到gdb中：



（STEP 8）前端访问该nginx，然后输入continue，请求会在断点出暂停：



然后就可以利用print命令去查看此处的任意变量了。如下图所示。



如果想调试程序的每一步执行情况可以使用next或者step去单步执行，再次不做介绍了，感兴趣的可以去尝试。

调试完毕后，为了让请求完成，可以再次输入continue完成请求的处理。

还可以给进程发送各种信号，如ctrl+c发送SIGINT信号从而终端改进程。

（STEP 9）退出gdb。