[nginx源码分析--GDB调试](http://blog.csdn.net/yusiguyuan/article/details/28006269)

标签： [NGINX](http://www.csdn.net/tag/NGINX)

2014-06-01 22:33 1044人阅读 [评论](http://blog.csdn.net/yusiguyuan/article/details/28006269#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/yusiguyuan/article/details/28006269#report)

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

nginx源码分析（53） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

利用gdb[i]调试nginx[ii]和利用gdb调试其它程序没有两样，不过nginx可以是daemon程序，也可以以多进程运行，因此利用gdb调试和平常会有些许不一样。当然，我们可以选择将nginx设置为非daemon模式并以单进程运行，而这需做如下设置即可：  
daemon off;  
master\_process off;

这是第一种情况：  
这种设置下的nginx在gdb下调试很普通，过程可以[iii]是这样：  
执行命令：  
lenky@lenky-desktop:/usr/local/nginx/sbin$ sudo gdb ./nginx  
当前目录是在/usr/local/nginx/sbin，该目录下有执行程序nginx，上条命令也就是用gdb来开始调试nginx，进入gdb命令行，直接输入r即可执行nginx：  
(gdb) r  
Starting program: /usr/local/nginx/sbin/nginx  
src/core/ngx\_conf\_file.c 1163 : 1000000  
===============================  
ngx\_timer\_resolution 1000000  
因为nginx以前台形式运行，键盘输入被nginx接管，此时无法输入gdb命令，因此需要按Ctrl+C退到gdb命令行模式，而nginx被中断暂停在\_\_kernel\_vsyscall调用里，输入bt命令可以查看到调用堆栈信息：  
(gdb) r  
Starting program: /usr/local/nginx/sbin/nginx  
src/core/ngx\_conf\_file.c 1163 : 1000000  
===============================  
ngx\_timer\_resolution 1000000  
^C  
Program received signal SIGINT, Interrupt.  
0xb7f29430 in \_\_kernel\_vsyscall ()  
(gdb) bt  
#0  0xb7f29430 in \_\_kernel\_vsyscall ()  
#1  0xb7e081a8 in epoll\_wait () from /lib/tls/i686/cmov/libc.so.6  
#2  0x08073ea3 in ngx\_epoll\_process\_events (cycle=0x860ad60, timer=4294967295,  
flags=0) at src/event/modules/ngx\_epoll\_module.c:405  
#3  0x080668ee in ngx\_process\_events\_and\_timers (cycle=0x860ad60)  
at src/event/ngx\_event.c:253  
#4  0×08071618 in ngx\_single\_process\_cycle (cycle=0x860ad60)  
at src/os/unix/ngx\_process\_cycle.c:283  
#5  0x0804a926 in main (argc=1, argv=0xbfd2ae44) at src/core/nginx.c:372  
(gdb)  
好，来给nginx设置断点，这很简单，比如这里给函数ngx\_process\_events\_and\_timers设置个断点，再输入命令c继续执行nginx：  
#4  0×08071618 in ngx\_single\_process\_cycle (cycle=0x860ad60)  
at src/os/unix/ngx\_process\_cycle.c:283  
#5  0x0804a926 in main (argc=1, argv=0xbfd2ae44) at src/core/nginx.c:372  
(gdb) b ngx\_process\_events\_and\_timers  
Breakpoint 1 at 0x80667d6: file src/event/ngx\_event.c, line 200.  
(gdb) c  
Continuing.  
Breakpoint 1, ngx\_process\_events\_and\_timers (cycle=0x860ad60)  
at src/event/ngx\_event.c:200  
200  {  
(gdb)  
结果我这里gdb马上提示断点中断，这是因为恰好调到这个函数而已，也许在你那并不会马上中断，从而gdb就一直停在Continuing后面，此时我们就要主动去触发事件（或者等待，时间可能很久，根据你的nginx设置）使得nginx调用这个函数，比如利用浏览器去访问nginx监听的站点，使得nginx有事件发生。  
断点中断后，利用s、c等gdb常规命令进行我们的调试，跟踪，这些无需多说：  
200  {  
(gdb) s  
\_\_cyg\_profile\_func\_enter (this=0x80667d0, call=0×8071618)  
at src/core/my\_debug.c:65  
warning: Source file is more recent than executable.  
65      my\_debug\_print(“Enter\n%p\n%p\n”, call, this);  
(gdb)  
66    }  
(gdb) c  
Continuing.  
ngx\_timer\_resolution 1000000  
第二种情况：  
当以daemon模式（即配置选项：daemon on;）单进程运行nginx时，上面那种方法不凑效了，原因很简单，当nginx以daemon模式运行时，这个启动过程中nginx将fork好几次，而那些父进程都早已直接退出了，因此看到的会这样：  
lenky@lenky-desktop:/usr/local/nginx/sbin$ sudo gdb ./nginx  
GNU gdb 6.8-debian  
Copyright (C) 2008 Free Software Foundation, Inc.  
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>  
This is free software: you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law.  Type “show copying”  
and “show warranty” for details.  
This GDB was configured as “i486-[**Linux**](http://lib.csdn.net/base/linux)-gnu”…  
(gdb) r  
Starting program: /usr/local/nginx/sbin/nginx  
src/core/ngx\_conf\_file.c 1163 : 1000000  
===============================  
Program exited normally.  
(gdb)  
而事实上nginx并没有退出，退出的是其父（或祖父）进程，而gdb没有跟着fork跟踪下去：  
lenky@lenky-desktop:/usr/local/nginx/sbin$ pidof nginx  
7908  
gdb考虑了这种情况，因此也提供了相应的follow-fork-mode选项：  
其用法为：  
set follow-fork-mode [parent|child]  
说明：  
parent: fork之后继续调试父进程，子进程不受影响。  
child: fork之后调试子进程，父进程不受影响。  
设置了该选项后，其它调式操作和之前介绍的基本一样，相关示例如下：  
lenky@lenky-desktop:/usr/local/nginx/sbin$ sudo gdb ./nginx -q  
(gdb) r  
Starting program: /usr/local/nginx/sbin/nginx  
src/core/ngx\_conf\_file.c 1163 : 1000000  
===============================  
Program exited normally.  
(gdb) shell pidof nginx  
9723  
(gdb) shell kill -9 9723  
(gdb) set follow-fork-mode child  
(gdb) b ngx\_process\_events\_and\_timers  
Breakpoint 1 at 0x80667d6: file src/event/ngx\_event.c, line 200.  
(gdb) r  
Starting program: /usr/local/nginx/sbin/nginx  
src/core/ngx\_conf\_file.c 1163 : 1000000  
===============================  
[Switching to process 9739]  
Breakpoint 1, ngx\_process\_events\_and\_timers (cycle=0x8ff0d60)  
at src/event/ngx\_event.c:200  
200  {  
(gdb) s  
[tcsetpgrp failed in terminal\_inferior: No such process]  
\_\_cyg\_profile\_func\_enter (this=0x80667d0, call=0×8071618)  
at src/core/my\_debug.c:65  
warning: Source file is more recent than executable.  
65      my\_debug\_print(“Enter\n%p\n%p\n”, call, this);  
(gdb) c  
Continuing.  
上面我们先设置断点，这样执行r命令后nginx将自动被中断返回到gdb命令行，如果不这样做，那么将无法返回gdb命令行，即按Ctrl+C没有效果，因为nginx是以daemon模式运行的，输入输出已经被重定向，nginx没有接收到这个Ctrl+C输入。当然也并不是没有其它办法，比如另外一个shell窗口，利用kill命令给nginx进程发送一个SIGINT信号即可，这和按Ctrl+C是一样的，如下（假定9897为nginx进程id号）：  
lenky@lenky-desktop:~$ sudo kill -2 9897  
第二种情况：  
第三种情况就是不管nginx运行模式了，是通用的，即利用gdb的attach、detach命令。  
假定当前nginx的相关配置这样：  
daemon on;  
master\_process on;  
worker\_processes  1;  
即daemon模式，多进程运行nginx。  
lenky@lenky-desktop:/usr/local/nginx/sbin$ sudo gdb -q  
(gdb) shell ./nginx  
src/core/ngx\_conf\_file.c 1163 : 1000000  
===============================  
(gdb) shell pidof nginx  
10246 10245  
(gdb)  
可以看到一共有两个nginx进程，这里10245为master进程，而10246为worker进程。我们要调试worker 10246可以如下：  
(gdb) attach 10246  
Attaching to program: /usr/local/nginx/sbin/nginx, process 10246  
Reading symbols from /lib/tls/i686/cmov/libcrypt.so.1…done.  
Loaded symbols for /lib/tls/i686/cmov/libcrypt.so.1  
Reading symbols from /lib/libpcre.so.3…done.  
Loaded symbols for /lib/libpcre.so.3  
Reading symbols from /usr/lib/libz.so.1…done.  
Loaded symbols for /usr/lib/libz.so.1  
Reading symbols from /lib/tls/i686/cmov/libc.so.6…done.  
Loaded symbols for /lib/tls/i686/cmov/libc.so.6  
Reading symbols from /lib/ld-linux.so.2…done.  
Loaded symbols for /lib/ld-linux.so.2  
Reading symbols from /lib/tls/i686/cmov/libnss\_compat.so.2…done.  
Loaded symbols for /lib/tls/i686/cmov/libnss\_compat.so.2  
Reading symbols from /lib/tls/i686/cmov/libnsl.so.1…done.  
Loaded symbols for /lib/tls/i686/cmov/libnsl.so.1  
Reading symbols from /lib/tls/i686/cmov/libnss\_nis.so.2…done.  
Loaded symbols for /lib/tls/i686/cmov/libnss\_nis.so.2  
Reading symbols from /lib/tls/i686/cmov/libnss\_files.so.2…done.  
Loaded symbols for /lib/tls/i686/cmov/libnss\_files.so.2  
0xb8016430 in \_\_kernel\_vsyscall ()  
(gdb) b ngx\_process\_events\_and\_timers  
Breakpoint 3 at 0x80667d6: file src/event/ngx\_event.c, line 200.  
(gdb) c  
Continuing.  
Breakpoint 3, ngx\_process\_events\_and\_timers (cycle=0x8f43d68) at src/event/ngx\_event.c:200  
200  {  
(gdb) detach  
Detaching from program: /usr/local/nginx/sbin/nginx, process 10246  
(gdb)  
总的来看，还是第三种方法最顺手了。

[i] http://www.gnu.org/software/gdb/

[ii] nginx是打开了-g选项编译的。

[iii] 意思是说下面的方法只是众多方法之一。