[gdb调试多进程和多线程命令](http://blog.csdn.net/pbymw8iwm/article/details/7876797)

标签： [多线程](http://www.csdn.net/tag/%e5%a4%9a%e7%ba%bf%e7%a8%8b)[thread](http://www.csdn.net/tag/thread)[debugging](http://www.csdn.net/tag/debugging)[null](http://www.csdn.net/tag/null)[centos](http://www.csdn.net/tag/centos)[file](http://www.csdn.net/tag/file)

2012-08-17 12:27 27902人阅读 [评论](http://blog.csdn.net/pbymw8iwm/article/details/7876797#comments)(2) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/pbymw8iwm/article/details/7876797#report)

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

linux（59） 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。

1. 默认设置下，在调试多进程程序时GDB只会调试主进程。但是GDB（>V7.0）支持多进程的**分别以及同时**调试，换句话说，GDB可以同时调试多个程序。只需要设置follow-fork-mode(默认值：parent)和detach-on-fork（默认值：on）即可。

      follow-fork-mode  detach-on-fork   说明

parent                   on               只调试主进程（GDB默认）  
child                     on               只调试子进程  
parent                   off              同时调试两个进程，gdb跟主进程，子进程block在fork位置  
child                     off              同时调试两个进程，gdb跟子进程，主进程block在fork位置

   设置方法：set follow-fork-mode [parent|child]   set detach-on-fork [on|off]  
  
   查询正在调试的进程：info inferiors  
   切换调试的进程： inferior <infer number>  
   添加新的调试进程： add-inferior [-copies n] [-exec executable] ,可以用file executable来分配给inferior可执行文件。  
   其他：remove-inferiors infno， detach inferior  
  
2. GDB默认支持调试多线程，跟主线程，子线程block在create thread。  
   查询线程：info threads  
   切换调试线程：thread <thread number>  
  
例程：

#include <stdio.h>  
#include <pthread.h>  
  
void processA();  
void processB();  
void \* processAworker(void \*arg);  
  
int main(int argc, const char \*argv[])  
  {  
  int pid;  
  
  pid = fork();  
  
  if(pid != 0)  
    processA();  
  else  
    processB();  
  
  return 0;  
  }  
  
void processA()  
  {  
  pid\_t pid = getpid();  
  char prefix[] = "ProcessA: ";  
  char tprefix[] = "thread ";  
  int tstatus;  
  pthread\_t pt;  
  
  printf("%s%lu %s\n", prefix, pid, "step1");  
  
  tstatus = pthread\_create(&pt, NULL, processAworker, NULL);  
  if( tstatus != 0 )  
    {  
    printf("ProcessA: Can not create new thread.");  
    }  
   
  processAworker(NULL);  
  sleep(1);  
  }  
  
void \* processAworker(void \*arg)  
  {  
  pid\_t pid = getpid();  
  pthread\_t tid = pthread\_self();  
  char prefix[] = "ProcessA: ";  
  char tprefix[] = "thread ";  
  
  printf("%s%lu %s%lu %s\n", prefix, pid, tprefix, tid, "step2");  
  printf("%s%lu %s%lu %s\n", prefix, pid, tprefix, tid, "step3");  
  
  return NULL;  
  }  
  
void processB()  
  {  
  pid\_t pid = getpid();  
  char prefix[] = "ProcessB: ";  
  printf("%s%lu %s\n", prefix, pid, "step1");  
  printf("%s%lu %s\n", prefix, pid, "step2");  
  printf("%s%lu %s\n", prefix, pid, "step3");  
  
  }

输出：

[cnwuwil@centos c-lab]$ ./test

ProcessA: 802 step1  
ProcessB: 803 step1  
ProcessB: 803 step2  
ProcessB: 803 step3  
ProcessA: 802 thread 3077555904 step2  
ProcessA: 802 thread 3077555904 step3  
ProcessA: 802 thread 3077553008 step2  
ProcessA: 802 thread 3077553008 step3

调试：  
1. 调试主进程，block子进程。

(gdb) set detach-on-fork off  
(gdb) show detach-on-fork  
Whether gdb will detach the child of a fork is off.  
(gdb) catch fork  
Catchpoint 1 (fork)  
(gdb) r  
[Thread debugging using libthread\_db enabled]  
  
Catchpoint 1 (forked process 3475), 0x00110424 in \_\_kernel\_vsyscall ()  
Missing separate debuginfos, use: debuginfo-install glibc-2.12-1.47.el6.i686  
(gdb) break test.c:14  
Breakpoint 2 at 0x8048546: file test.c, line 14.  
(gdb) cont  
[New process 3475]  
[Thread debugging using libthread\_db enabled]  
  
Breakpoint 2, main (argc=1, argv=0xbffff364) at test.c:14  
Missing separate debuginfos, use: debuginfo-install glibc-2.12-1.47.el6.i686  
(gdb) info inferiors  
  Num  Description       Executable         
  2    process 3475      /home/cnwuwil/labs/c-lab/test  
\* 1    process 3472      /home/cnwuwil/labs/c-lab/test

2. 切换到子进程：

(gdb) inferior 2  
[Switching to inferior 2 [process 3475] (/home/cnwuwil/labs/c-lab/test)]  
[Switching to thread 2 (Thread 0xb7fe86c0 (LWP 3475))]  
#0  0x00110424 in ?? ()  
(gdb) info inferiors  
  Num  Description       Executable         
\* 2    process 3475      /home/cnwuwil/labs/c-lab/test  
  1    process 3472      /home/cnwuwil/labs/c-lab/test  
(gdb) inferior 1  
[Switching to inferior 1 [process 3472] (/home/cnwuwil/labs/c-lab/test)]  
[Switching to thread 1 (Thread 0xb7fe86c0 (LWP 3472))]  
#0  main (argc=1, argv=0xbffff364) at test.c:14  
(gdb) info inferiors  
  Num  Description       Executable         
  2    process 3475      /home/cnwuwil/labs/c-lab/test  
\* 1    process 3472      /home/cnwuwil/labs/c-lab/test

3. 设断点继续调试主进程，主进程产生两个子线程：

(gdb) break test.c:50  
Breakpoint 3 at 0x804867d: file test.c, line 50. (2 locations)  
(gdb) cont  
ProcessA: 3472 step1  
[New Thread 0xb7fe7b70 (LWP 3562)]  
ProcessA: 3472 thread 3086911168 step2  
  
Breakpoint 3, processAworker (arg=0x0) at test.c:50  
(gdb) info inferiors  
  Num  Description       Executable         
  2    process 3475      /home/cnwuwil/labs/c-lab/test  
\* 1    process 3472      /home/cnwuwil/labs/c-lab/test  
(gdb) info threads  
  3 Thread 0xb7fe7b70 (LWP 3562)  0x00110424 in \_\_kernel\_vsyscall ()  
  2 Thread 0xb7fe86c0 (LWP 3475)  0x00110424 in ?? ()  
\* 1 Thread 0xb7fe86c0 (LWP 3472)  processAworker (arg=0x0) at test.c:50

4. 切换到主进程中的子线程，注意：线程2为前面产生的子进程

(gdb) thread 3  
[Switching to thread 3 (Thread 0xb7fe7b70 (LWP 3562))]#0  0x00110424 in \_\_kernel\_vsyscall ()  
(gdb) cont  
ProcessA: 3472 thread 3086911168 step3  
ProcessA: 3472 thread 3086908272 step2  
[Switching to Thread 0xb7fe7b70 (LWP 3562)]  
  
Breakpoint 3, processAworker (arg=0x0) at test.c:50  
(gdb) info threads  
\* 3 Thread 0xb7fe7b70 (LWP 3562)  processAworker (arg=0x0) at test.c:50  
  2 Thread 0xb7fe86c0 (LWP 3475)  0x00110424 in ?? ()  
  1 Thread 0xb7fe86c0 (LWP 3472)  0x00110424 in \_\_kernel\_vsyscall ()  
(gdb) thread 1