多线程调试

(一)多线程调试命令

shell的命令：

(1)查看当前运行的进程：ps aux | grep book

(2)查看当前运行的轻量级进程：ps -aL | grep book

(3)查看主线程和子线程的关系：pstree -p 主线程id

gdb的命令：

(1)查看可切换调试的线程：info threads

(2)切换调试的线程：thread 线程id

(3)只运行当前线程：set scheduler-locking on

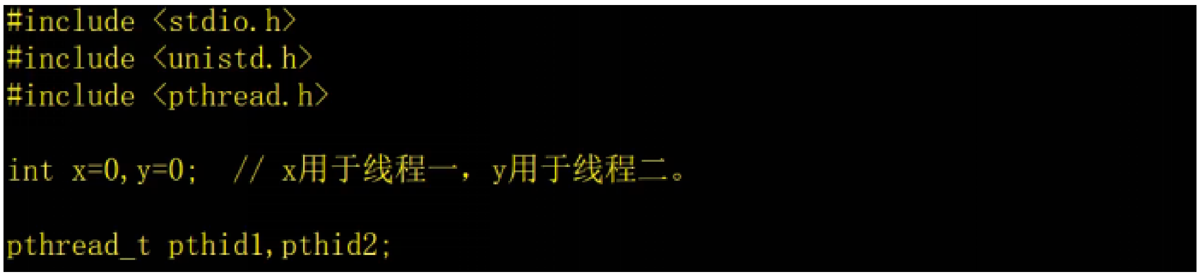
(4)运行全部的线程：set scheduler-locking off

(5)指定某线程执行某gdb命令：thread apply 线程id gdb_cmd

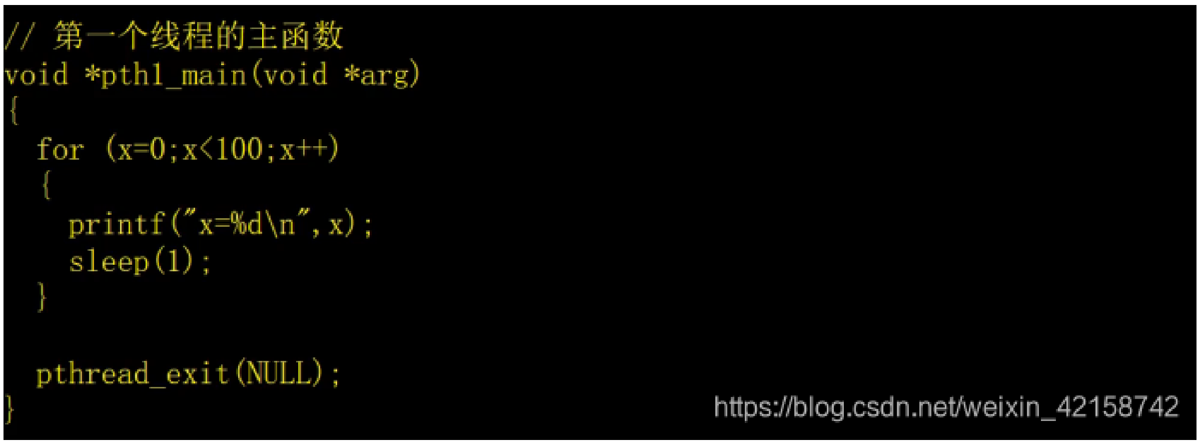
(6)全部的线程执行某gdb命令：thread apply all gdb_cmd

(二)多线程调试演示

以调试book.c为例，内容如下：

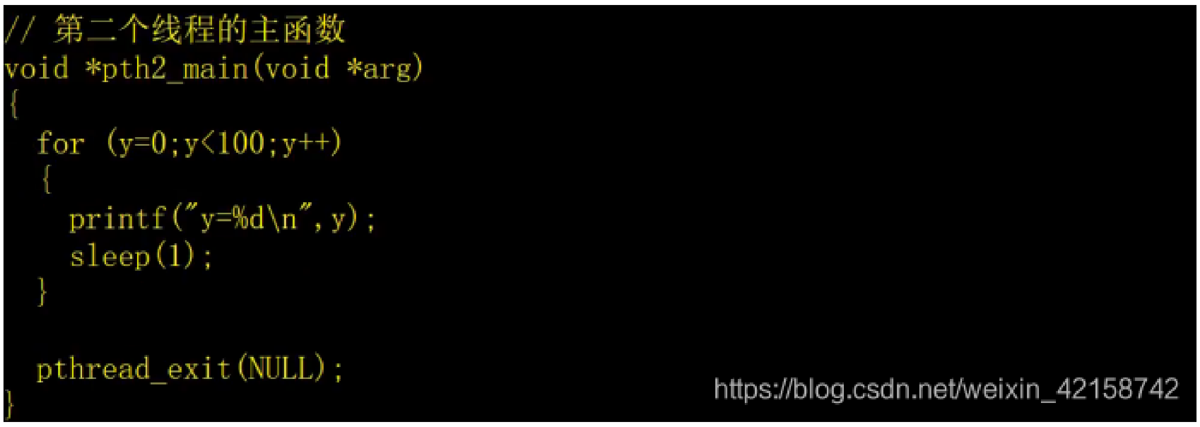


```
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>

int x=0,y=0;  // x用于线程一, y用于线程二。

pthread_t pthid1,pthid2;
```

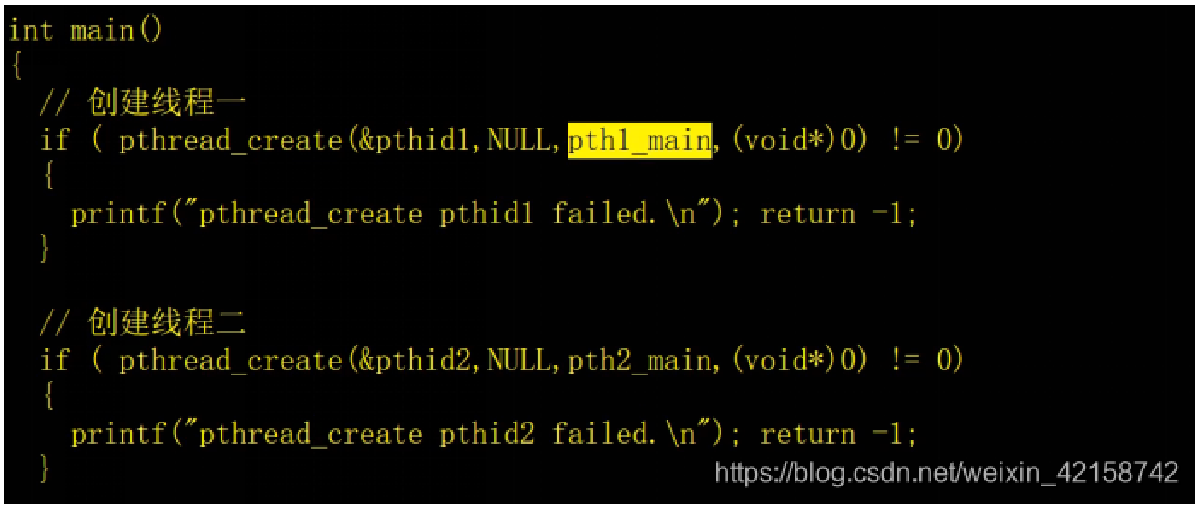


```
// 第一个线程的主函数
void *pth1_main(void *arg)
{
  for (x=0;x<100;x++)
  {
    printf("x=%d\n",x);
    sleep(1);
  }

  pthread_exit(NULL);
}
```

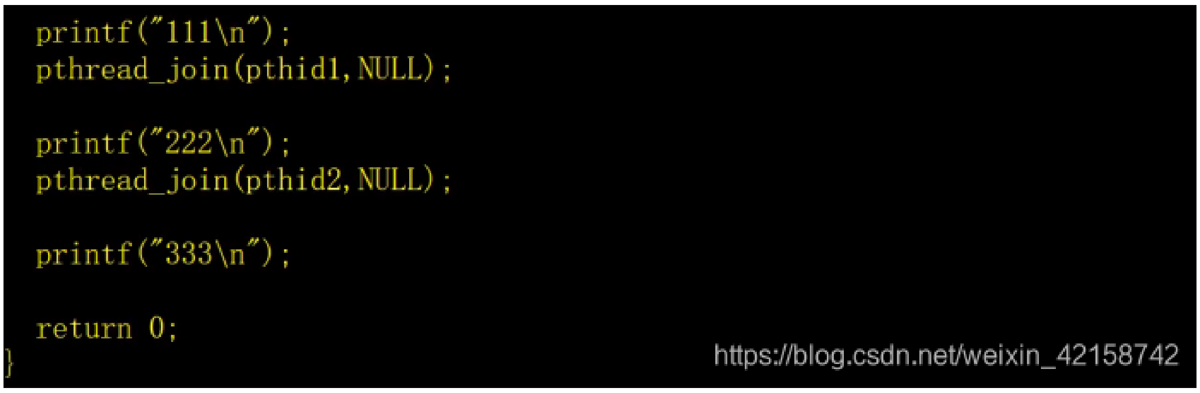
https://blog.csdn.net/weixin_42158742



```
// 第二个线程的主函数
void *pth2_main(void *arg)
{
  for (y=0;y<100;y++)
  {
    printf("y=%d\n",y);
    sleep(1);
  }

  pthread_exit(NULL);
}
```



```
int main()
{
  // 创建线程一
  if ( pthread_create(&pthid1,NULL,pth1_main,(void*)0) != 0)
  {
    printf("pthread_create pthid1 failed.\n"); return -1;
  }

  // 创建线程二
  if ( pthread_create(&pthid2,NULL,pth2_main,(void*)0) != 0)
  {
    printf("pthread_create pthid2 failed.\n"); return -1;
  }
```



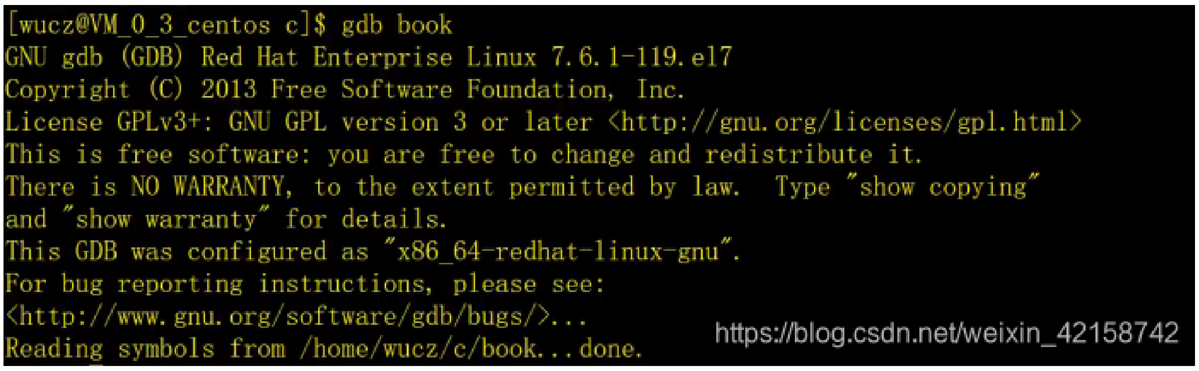
```
  printf("111\n");
  pthread_join(pthid1,NULL);

  printf("222\n");
  pthread_join(pthid2,NULL);

  printf("333\n");

  return 0;
}
```

编译，生成可执行文件

$ gcc -g -o book book.c

开始调试命令

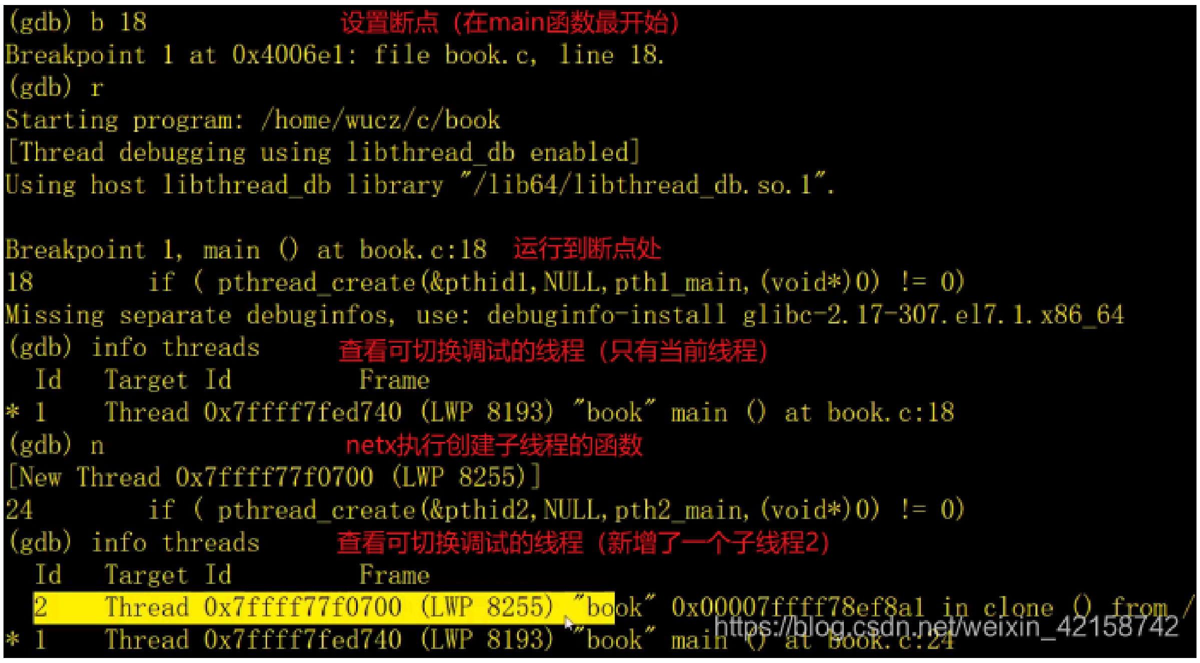
$ gdb book



```
[wucz@VM_0_3_centos c]$ gdb book
GNU gdb (GDB) Red Hat Enterprise Linux 7.6.1-119.el7
Copyright (C) 2013 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 "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /home/wucz/c/book...done.
```

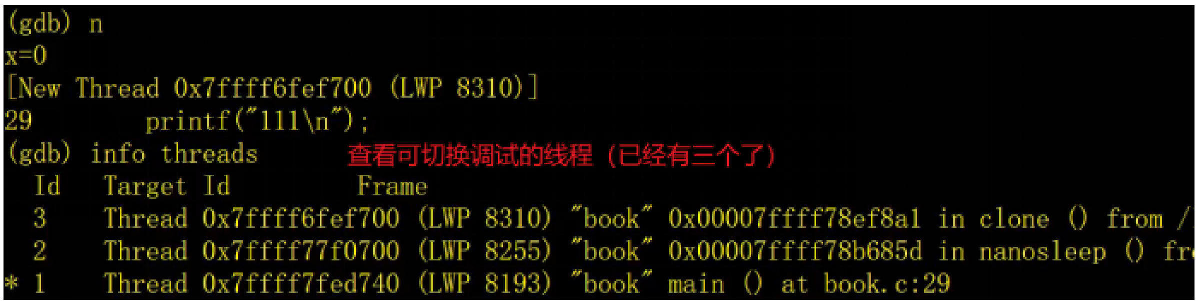
1.查看/切换调试的线程

开始调试

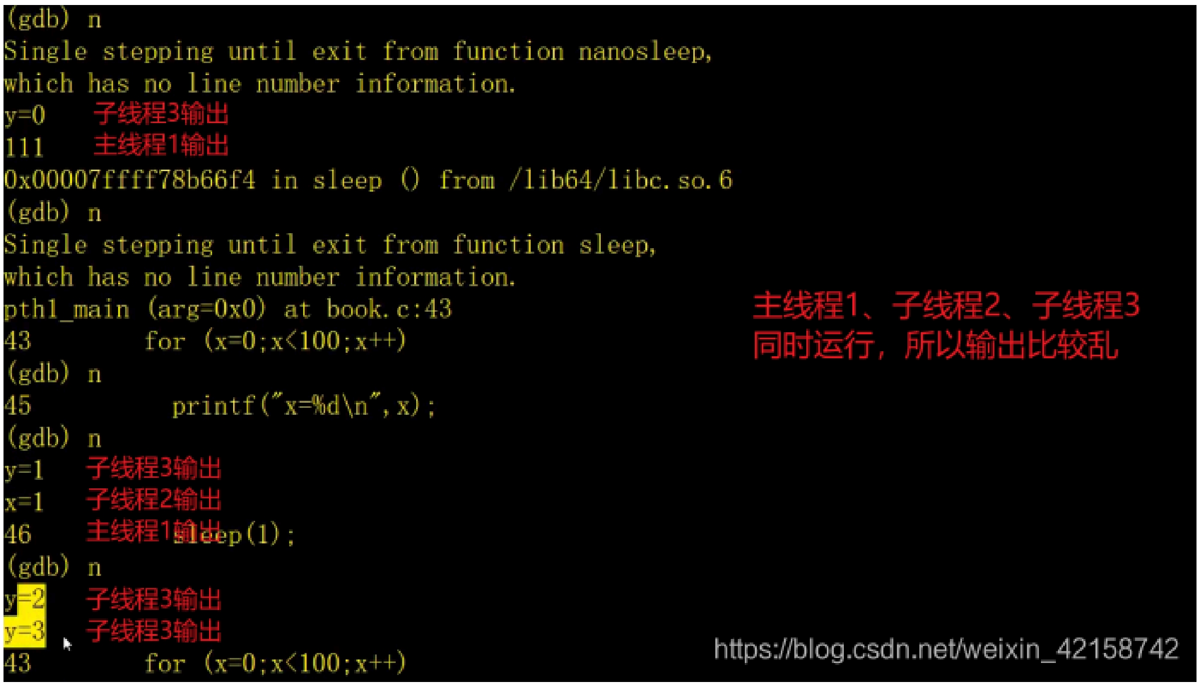


```
(gdb) b 18                设置断点（在main函数最开始）
Breakpoint 1 at 0x4006e1: file book.c, line 18.
(gdb) r
Starting program: /home/wucz/c/book
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".

Breakpoint 1, main () at book.c:18  运行到断点处
18          if ( pthread_create(&pthid1,NULL,pth1_main,(void*)0) != 0)
Missing separate debuginfos, use: debuginfo-install glibc-2.17-307.el7.1.x86_64
(gdb) info threads        查看可切换调试的线程（只有当前线程）
  Id   Target Id         Frame
* 1    Thread 0x7ffff7fed740 (LWP 8193) "book" main () at book.c:18
(gdb) n                   netx执行创建子线程的函数
[New Thread 0x7ffff77f0700 (LWP 8255)]
24          if ( pthread_create(&pthid2,NULL,pth2_main,(void*)0) != 0)
(gdb) info threads        查看可切换调试的线程（新增了一个子线程2）
  Id   Target Id         Frame
  2    Thread 0x7ffff77f0700 (LWP 8255) "book" 0x00007ffff78ef8a1 in clone () from /
* 1    Thread 0x7ffff7fed740 (LWP 8193) "book" main () at book.c:24
```

https://blog.csdn.net/weixin_42158742



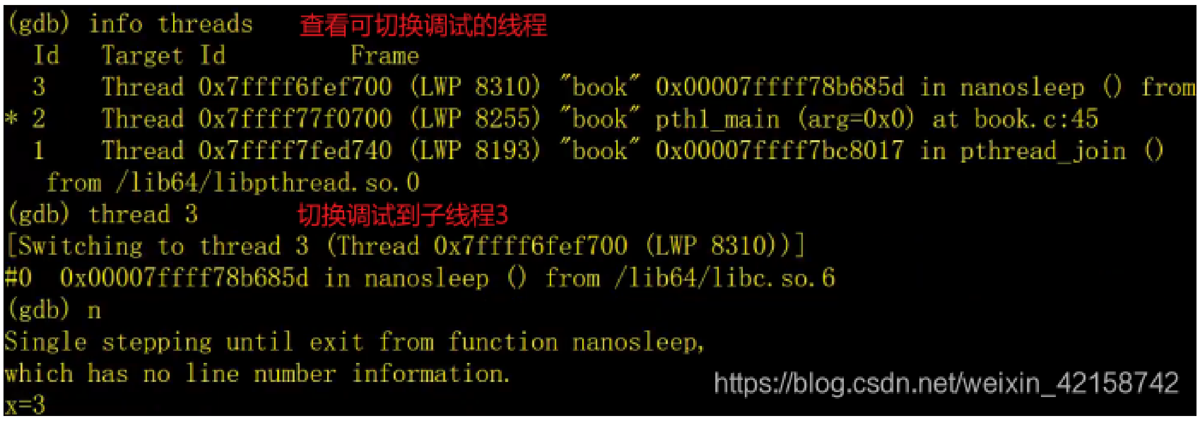
```
(gdb) n
x=0
[New Thread 0x7ffff6fef700 (LWP 8310)]
29          printf("111\n");
(gdb) info threads        查看可切换调试的线程（已经有三个了）
  Id   Target Id         Frame
  3    Thread 0x7ffff6fef700 (LWP 8310) "book" 0x00007ffff78ef8a1 in clone () from /
  2    Thread 0x7ffff77f0700 (LWP 8255) "book" 0x00007ffff78b685d in nanosleep () fr
* 1    Thread 0x7ffff7fed740 (LWP 8193) "book" main () at book.c:29
```



```
(gdb) n
Single stepping until exit from function nanosleep,
which has no line number information.
y=0    子线程3输出
111    主线程1输出
0x00007ffff78b66f4 in sleep () from /lib64/libc.so.6
(gdb) n
Single stepping until exit from function sleep,
which has no line number information.
pth1_main (arg=0x0) at book.c:43
43          for (x=0;x<100;x++)
(gdb) n
45            printf("x=%d\n",x);
(gdb) n
y=1    子线程3输出
x=1    子线程2输出
46     主线程1输出 sleep(1);
(gdb) n
y=2    子线程3输出
y=3    子线程3输出
43          for (x=0;x<100;x++)
```

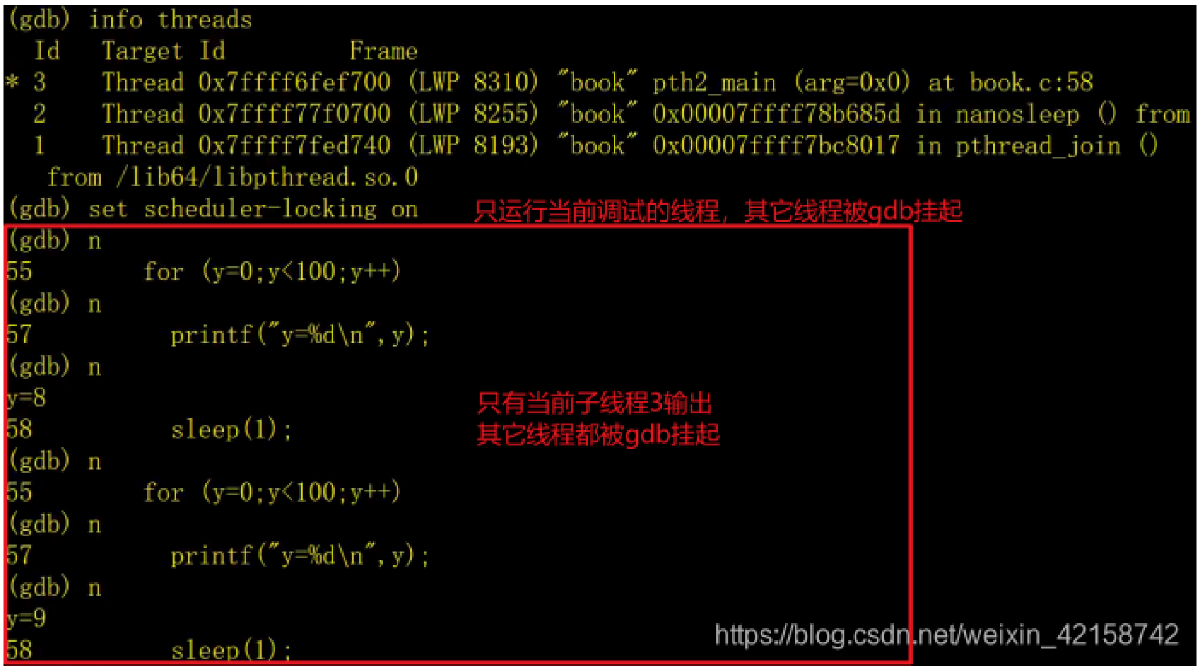
主线程1、子线程2、子线程3
同时运行，所以输出比较乱

https://blog.csdn.net/weixin_42158742

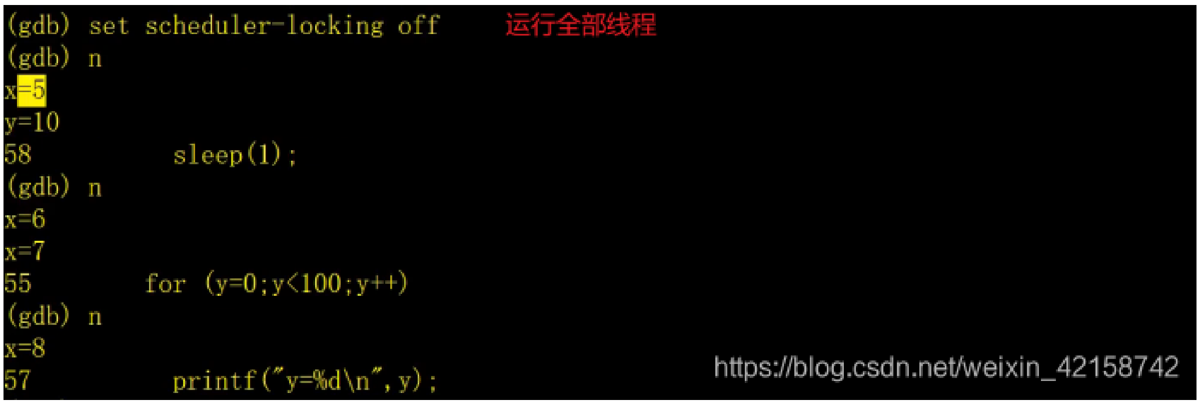


```
(gdb) info threads      查看可切换调试的线程
  Id   Target Id         Frame
  3    Thread 0x7ffff6fef700 (LWP 8310) "book" 0x00007ffff78b685d in nanosleep () from
* 2    Thread 0x7ffff77f0700 (LWP 8255) "book" pth1_main (arg=0x0) at book.c:45
  1    Thread 0x7ffff7fed740 (LWP 8193) "book" 0x00007ffff7bc8017 in pthread_join ()
   from /lib64/libpthread.so.0
(gdb) thread 3          切换调试到子线程3
[Switching to thread 3 (Thread 0x7ffff6fef700 (LWP 8310))]
#0  0x00007ffff78b685d in nanosleep () from /lib64/libc.so.6
(gdb) n
Single stepping until exit from function nanosleep,
which has no line number information.                https://blog.csdn.net/weixin_42158742
x=3
```

2.只运行当前线程

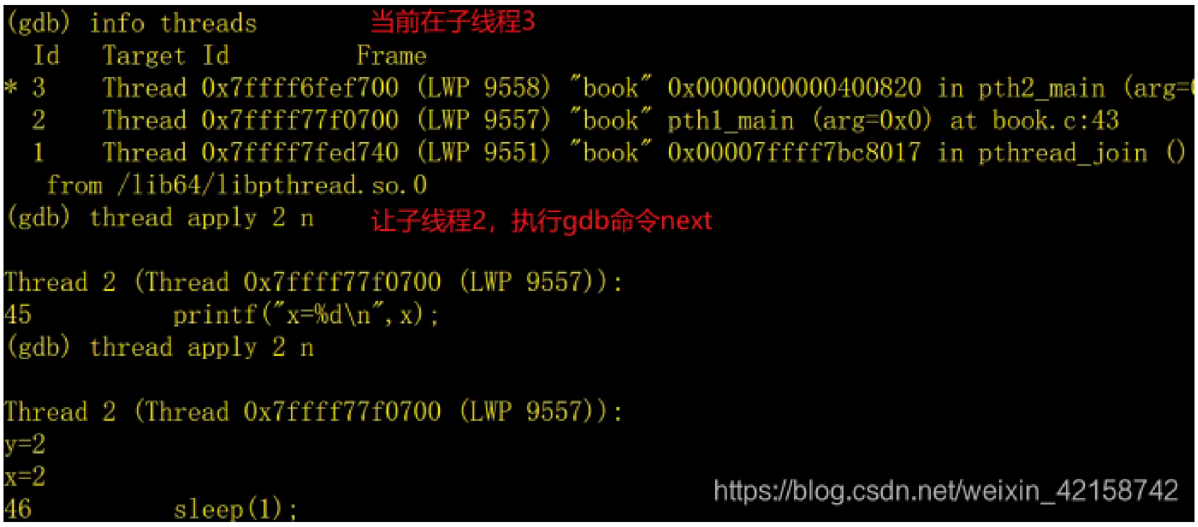


```
(gdb) info threads
  Id   Target Id         Frame
* 3    Thread 0x7ffff6fef700 (LWP 8310) "book" pth2_main (arg=0x0) at book.c:58
  2    Thread 0x7ffff77f0700 (LWP 8255) "book" 0x00007ffff78b685d in nanosleep () from
  1    Thread 0x7ffff7fed740 (LWP 8193) "book" 0x00007ffff7bc8017 in pthread_join ()
   from /lib64/libpthread.so.0
(gdb) set scheduler-locking on     只运行当前调试的线程，其它线程被gdb挂起
(gdb) n
55          for (y=0;y<100;y++)
(gdb) n
57            printf("y=%d\n",y);
(gdb) n
y=8                                只有当前子线程3输出
58            sleep(1);            其它线程都被gdb挂起
(gdb) n
55          for (y=0;y<100;y++)
(gdb) n
57            printf("y=%d\n",y);
(gdb) n
y=9
58            sleep(1);                              https://blog.csdn.net/weixin_42158742
```

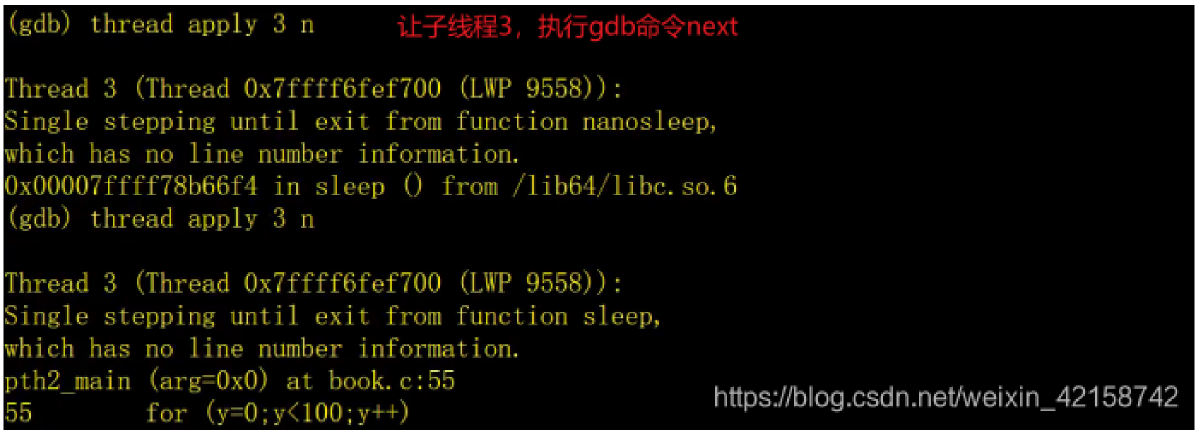


```
(gdb) set scheduler-locking off     运行全部线程
(gdb) n
x=5
y=10
58            sleep(1);
(gdb) n
x=6
x=7
55          for (y=0;y<100;y++)
(gdb) n
x=8
57            printf("y=%d\n",y);                    https://blog.csdn.net/weixin_42158742
```

3.指定某线程执行gdb命令



```
(gdb) info threads        当前在子线程3
  Id   Target Id         Frame
* 3    Thread 0x7ffff6fef700 (LWP 9558) "book" 0x0000000000400820 in pth2_main (arg=
  2    Thread 0x7ffff77f0700 (LWP 9557) "book" pth1_main (arg=0x0) at book.c:43
  1    Thread 0x7ffff7fed740 (LWP 9551) "book" 0x00007ffff7bc8017 in pthread_join ()
   from /lib64/libpthread.so.0
(gdb) thread apply 2 n    让子线程2，执行gdb命令next

Thread 2 (Thread 0x7ffff77f0700 (LWP 9557)):
45            printf("x=%d\n",x);
(gdb) thread apply 2 n

Thread 2 (Thread 0x7ffff77f0700 (LWP 9557)):
y=2
x=2
46            sleep(1);
```



```
(gdb) thread apply 3 n      让子线程3，执行gdb命令next

Thread 3 (Thread 0x7ffff6fef700 (LWP 9558)):
Single stepping until exit from function nanosleep,
which has no line number information.
0x00007ffff78b66f4 in sleep () from /lib64/libc.so.6
(gdb) thread apply 3 n

Thread 3 (Thread 0x7ffff6fef700 (LWP 9558)):
Single stepping until exit from function sleep,
which has no line number information.
pth2_main (arg=0x0) at book.c:55
55          for (y=0;y<100;y++)
```

下一篇：GDB多进程调试（调试命令+调试演示）

个人公众号：拾一札记

参考：

码农有道

www.freecplus.net

https://www.bilibili.com/video/BV1ei4y1V758?p=4

https://freecplus.net/b72113dda88a43b48728e0552fd8a74c.html

如果文章有错别字，或者内容有错误，或其他的建议和意见，请您联系我指正，非常感谢！！