**参考文献：**

**Linux内核进程管理之死锁原因分析**

**知乎公众号：玩转Linux内核**

**一，死锁原因**

二，代码实例

三，第一种情况

四，第二种情况

五，第三种情况

六，死锁总结

**一，死锁原因**

死锁问题大概可以分为3种情况：

1. 接口卡主锁不能及时释放，导致别的线程卡主，这类不算真正意义的死锁。

2. 同一个线程锁多次导致死锁。

3. 两个锁交叉使用导致死锁。

这三种情况的定位方法都是一样，都是根据锁的owner顺藤摸瓜，比如线程1在等锁，必然有别的线程在占有锁，可以查看线程1中的锁的owner找到线程2，依次类推，知道把锁的关系理清楚就知道死锁原因了。

**二，代码实例**

下面就根据测试代码分析这三种情况。

**#include** <stdio.h>

**#include** <stdlib.h>

**#include** <iostream>

**#include** <unistd.h>

**#include** <list>

**#include** <pthread.h>

**#include** <assert.h>

**#include** <semaphore.h>

pthread\_mutex\_t g\_mutex **=** PTHREAD\_MUTEX\_INITIALIZER;

pthread\_mutex\_t g\_mutex1 **=** PTHREAD\_MUTEX\_INITIALIZER;

**void** **\*threadProc1**(**void** **\***)

{

printf("threadProc1 enter = %d\n", getpid());

pthread\_mutex\_lock(**&**g\_mutex);

**while**(1);

pthread\_mutex\_unlock(**&**g\_mutex);

printf("threadProc1 unlock pid = %d\n", getpid());

}

**void** **\*threadProc2**(**void** **\***)

{

printf("threadProc2 enter = %d\n", getpid());

pthread\_mutex\_lock(**&**g\_mutex);

pthread\_mutex\_unlock(**&**g\_mutex);

printf("threadProc2 unlock pid = %d\n", getpid());

}

**void** **fun3**()

{

printf("fun3 begin\n");

pthread\_mutex\_lock(**&**g\_mutex1);

pthread\_mutex\_unlock(**&**g\_mutex1);

printf("fun3 end\n");

}

**void** **fun4**()

{

printf("fun4 begin\n");

pthread\_mutex\_lock(**&**g\_mutex);

pthread\_mutex\_unlock(**&**g\_mutex);

printf("fun4 end\n");

}

**void** **\*threadProc3**(**void** **\***)

{

printf("threadProc3 enter = %d\n", getpid());

pthread\_mutex\_lock(**&**g\_mutex);

sleep(1);

fun3();

pthread\_mutex\_unlock(**&**g\_mutex);

printf("threadProc3 unlock pid = %d\n", getpid());

}

**void** **\*threadProc4**(**void** **\***)

{

printf("threadProc4 enter = %d\n", getpid());

pthread\_mutex\_lock(**&**g\_mutex1);

fun4();

pthread\_mutex\_unlock(**&**g\_mutex1);

printf("threadProc4 unlock pid = %d\n", getpid());

}

**void** **fun5**()

{

printf("fun5 begin\n");

pthread\_mutex\_lock(**&**g\_mutex);

pthread\_mutex\_unlock(**&**g\_mutex);

printf("fun5 end\n");

}

**void** **\*threadProc5**(**void** **\***)

{

printf("threadProc5 enter = %d\n", getpid());

pthread\_mutex\_lock(**&**g\_mutex);

fun5();

pthread\_mutex\_unlock(**&**g\_mutex);

printf("threadProc5 unlock pid = %d\n", getpid());

}

**void** **Thread1**()

{

pthread\_t tid;

pthread\_create(**&**tid, NULL, threadProc1, NULL);

pthread\_detach(tid);

}

**void** **Thread2**()

{

pthread\_t tid;

pthread\_create(**&**tid, NULL, threadProc2, NULL);

pthread\_detach(tid);

}

**void** **Thread3**()

{

pthread\_t tid;

pthread\_create(**&**tid, NULL, threadProc3, NULL);

pthread\_detach(tid);

}

**void** **Thread4**()

{

pthread\_t tid;

pthread\_create(**&**tid, NULL, threadProc4, NULL);

pthread\_detach(tid);

}

**void** **Thread5**()

{

pthread\_t tid;

pthread\_create(**&**tid, NULL, threadProc5, NULL);

pthread\_detach(tid);

}

*//回调接口卡主导致其他线程等锁*

**void** **test1**()

{

printf("test1 begin\n");

Thread1();

Thread2();

printf("test1 end\n");

}

*//两个锁交叉使用导致死锁*

**void** **test2**()

{

Thread3();

Thread4();

}

*//同一锁被一个线程锁多次*

**void** **test3**()

{

Thread5();

}

**int** **main**(**int** argc, **char** **\***argv[])

{

test1();

*//test2();*

*//test3();*

**while**(1)

{

sleep(1);

}

**return** 0;

}

**编译：**

baoqingti@Cpl**-**IBP**-**Product:**~/**data**/**work**/**toolSH**/**tool**+**sh2$ g**++** **-**o deadlock deadlock.cpp **-**lpthrea

**三，第一种情况**

**首先分析回调接口卡主导致锁等待**

执行deadlock结果：

baoqingti@Cpl**-**IBP**-**Product:**~/**data**/**work**/**toolSH**/**tool**+**sh2$ .**/**deadlock

test1 begin

test1 end

threadProc2 enter **=** 2036

threadProc1 enter **=** 2036

可以看到线程1是死循环卡住了，占有了g\_mutex, 而线程2进入的时候需要等待g\_mutex可用，那么结果，就是线程2一直在等待。当然这里从代码就能看出问题了，如果在真实环境中，这种情况不加分析是不清楚是死锁还是卡主，现在通过gdb大致分析下。

GNU **gdb** (GDB) 7.6.1

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-unknown-linux-gnu".

For bug reporting instructions, please see:

**<**http:*//www.gnu.org/software/gdb/bugs/>...*

Reading symbols from **/**data1**/**data\_baoqingti**/**work**/**toolSH**/**tool**+**sh2**/**deadlock...done.

(gdb) r

Starting program: **/**data1**/**data\_baoqingti**/**work**/**toolSH**/**tool**+**sh2**/**deadlock

[Thread debugging **using** libthread\_db enabled]

Using host libthread\_db library "/lib/x86\_64-linux-gnu/libthread\_db.so.1".

test1 begin

[New Thread 0x7ffff6fec700 (LWP 32361)]

threadProc1 enter **=** 32357

[New Thread 0x7ffff67eb700 (LWP 32362)]

test1 end

threadProc2 enter **=** 32357

**^**C

Program received signal SIGINT, Interrupt.

0x00007ffff75be0ad in nanosleep () from **/**lib**/**x86\_64**-**linux**-**gnu**/**libc.so.6

(gdb) info **thread**

Id Target Id Frame

3 Thread 0x7ffff67eb700 (LWP 32362) "deadlock" 0x00007ffff7bcb89c in \_\_lll\_lock\_wait ()

from **/**lib**/**x86\_64**-**linux**-**gnu**/**libpthread.so.0

2 Thread 0x7ffff6fec700 (LWP 32361) "deadlock" threadProc1 () at deadlock.cpp:16

**\*** 1 Thread 0x7ffff7fef740 (LWP 32357) "deadlock" 0x00007ffff75be0ad in nanosleep () from **/**lib**/**x86\_64**-**linux**-**gnu**/**libc.so.6

(gdb) t 3

[Switching to **thread** 3 (Thread 0x7ffff67eb700 (LWP 32362))]

**#0 0x00007ffff7bcb89c in \_\_lll\_lock\_wait () from /lib/x86\_64-linux-gnu/libpthread.so.0**

(gdb) bt

**#0 0x00007ffff7bcb89c in \_\_lll\_lock\_wait () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#1 0x00007ffff7bc7065 in \_L\_lock\_858 () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#2 0x00007ffff7bc6eba in pthread\_mutex\_lock () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#3 0x000000000040091e in threadProc2 () at deadlock.cpp:23**

**#4 0x00007ffff7bc4e9a in start\_thread () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#5 0x00007ffff75f238d in clone () from /lib/x86\_64-linux-gnu/libc.so.6**

**#6 0x0000000000000000 in ?? ()**

(gdb) f 2

**#2 0x00007ffff7bc6eba in pthread\_mutex\_lock () from /lib/x86\_64-linux-gnu/libpthread.so.0**

(gdb) p g\_mutex

$1 **=** {\_\_data **=** {\_\_lock **=** 2, \_\_count **=** 0, \_\_owner **=** 32361, \_\_nusers **=** 1, \_\_kind **=** 0, \_\_spins **=** 0, \_\_list **=** {\_\_prev **=** 0x0,

\_\_next **=** 0x0}}, \_\_size **=** "\002\000\000\000\000\000\000\000i~\000\000\001", '\000' **<**repeats 26 times**>**, \_\_align **=** 2}

(gdb) GNU gdb (GDB) 7.6.1

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-unknown-linux-gnu".

For bug reporting instructions, please see:

**<**http:*//www.gnu.org/software/gdb/bugs/>...*

Reading symbols from **/**data1**/**data\_baoqingti**/**work**/**toolSH**/**tool**+**sh2**/**deadlock...done.

(gdb) r

Starting program: **/**data1**/**data\_baoqingti**/**work**/**toolSH**/**tool**+**sh2**/**deadlock

[Thread debugging **using** libthread\_db enabled]

Using host libthread\_db library "/lib/x86\_64-linux-gnu/libthread\_db.so.1".

test1 begin

[New Thread 0x7ffff6fec700 (LWP 32361)]

threadProc1 enter **=** 32357

[New Thread 0x7ffff67eb700 (LWP 32362)]

test1 end

threadProc2 enter **=** 32357

**^**C

Program received signal SIGINT, Interrupt.

0x00007ffff75be0ad in nanosleep () from **/**lib**/**x86\_64**-**linux**-**gnu**/**libc.so.6

(gdb) info **thread**

Id Target Id Frame

3 Thread 0x7ffff67eb700 (LWP 32362) "deadlock" 0x00007ffff7bcb89c in \_\_lll\_lock\_wait ()

from **/**lib**/**x86\_64**-**linux**-**gnu**/**libpthread.so.0

2 Thread 0x7ffff6fec700 (LWP 32361) "deadlock" threadProc1 () at deadlock.cpp:16

**\*** 1 Thread 0x7ffff7fef740 (LWP 32357) "deadlock" 0x00007ffff75be0ad in nanosleep () from **/**lib**/**x86\_64**-**linux**-**gnu**/**libc.so.6

(gdb) t 3

[Switching to **thread** 3 (Thread 0x7ffff67eb700 (LWP 32362))]

**#0 0x00007ffff7bcb89c in \_\_lll\_lock\_wait () from /lib/x86\_64-linux-gnu/libpthread.so.0**

(gdb) bt

**#0 0x00007ffff7bcb89c in \_\_lll\_lock\_wait () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#1 0x00007ffff7bc7065 in \_L\_lock\_858 () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#2 0x00007ffff7bc6eba in pthread\_mutex\_lock () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#3 0x000000000040091e in threadProc2 () at deadlock.cpp:23**

**#4 0x00007ffff7bc4e9a in start\_thread () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#5 0x00007ffff75f238d in clone () from /lib/x86\_64-linux-gnu/libc.so.6**

**#6 0x0000000000000000 in ?? ()**

(gdb) f 2

**#2 0x00007ffff7bc6eba in pthread\_mutex\_lock () from /lib/x86\_64-linux-gnu/libpthread.so.0**

(gdb) p g\_mutex

$1 **=** {\_\_data **=** {\_\_lock **=** 2, \_\_count **=** 0, \_\_owner **=** 32361, \_\_nusers **=** 1, \_\_kind **=** 0, \_\_spins **=** 0, \_\_list **=** {\_\_prev **=** 0x0,

\_\_next **=** 0x0}}, \_\_size **=** "\002\000\000\000\000\000\000\000i~\000\000\001", '\000' **<**repeats 26 times**>**, \_\_align **=** 2}

(gdb)

线程32362，在等锁，锁是被32361线程占用了，而32361线程回调是死循环，g\_mutex一直不会释放，那么32362

就一直在等待。这种情况分析32361线程回调卡主原因基本上就能解决问题了。

**四，第二种情况**

同一个线程多次lock导致死锁。对应test3

执行test3运行结果：

baoqingti@Cpl**-**IBP**-**Product:**~/**data**/**work**/**toolSH**/**tool**+**sh2$ .**/**deadlock

threadProc5 enter **=** 6612

fun5 begin

这里是fun5函数里面在等待g\_mutex，通过gdb看下g\_mutex被谁占用了，

GNU gdb (GDB) 7.6.1

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-unknown-linux-gnu".

For bug reporting instructions, please see:

**<**http:*//www.gnu.org/software/gdb/bugs/>...*

Reading symbols from **/**data1**/**data\_baoqingti**/**work**/**toolSH**/**tool**+**sh2**/**deadlock...done.

(gdb) r

Starting program: **/**data1**/**data\_baoqingti**/**work**/**toolSH**/**tool**+**sh2**/**deadlock

[Thread debugging **using** libthread\_db enabled]

Using host libthread\_db library "/lib/x86\_64-linux-gnu/libthread\_db.so.1".

[New Thread 0x7ffff6fec700 (LWP 6655)]

threadProc5 enter **=** 6651

fun5 begin

**^**C

Program received signal SIGINT, Interrupt.

0x00007ffff75be0ad in nanosleep () from **/**lib**/**x86\_64**-**linux**-**gnu**/**libc.so.6

(gdb) info **thread**

Id Target Id Frame

2 Thread 0x7ffff6fec700 (LWP 6655) "deadlock" 0x00007ffff7bcb89c in \_\_lll\_lock\_wait ()

from **/**lib**/**x86\_64**-**linux**-**gnu**/**libpthread.so.0

**\*** 1 Thread 0x7ffff7fef740 (LWP 6651) "deadlock" 0x00007ffff75be0ad in nanosleep () from **/**lib**/**x86\_64**-**linux**-**gnu**/**libc.so.6

(gdb) t 2

[Switching to **thread** 2 (Thread 0x7ffff6fec700 (LWP 6655))]

**#0 0x00007ffff7bcb89c in \_\_lll\_lock\_wait () from /lib/x86\_64-linux-gnu/libpthread.so.0**

(gdb) bt

**#0 0x00007ffff7bcb89c in \_\_lll\_lock\_wait () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#1 0x00007ffff7bc7065 in \_L\_lock\_858 () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#2 0x00007ffff7bc6eba in pthread\_mutex\_lock () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#3 0x0000000000400a64 in fun5 () at deadlock.cpp:62**

**#4 0x0000000000400aab in threadProc5 () at deadlock.cpp:70**

**#5 0x00007ffff7bc4e9a in start\_thread () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#6 0x00007ffff75f238d in clone () from /lib/x86\_64-linux-gnu/libc.so.6**

**#7 0x0000000000000000 in ?? ()**

(gdb) f 2

**#2 0x00007ffff7bc6eba in pthread\_mutex\_lock () from /lib/x86\_64-linux-gnu/libpthread.so.0**

(gdb) p g\_mutex

$1 **=** {\_\_data **=** {\_\_lock **=** 2, \_\_count **=** 0, \_\_owner **=** 6655, \_\_nusers **=** 1, \_\_kind **=** 0, \_\_spins **=** 0, \_\_list **=** {\_\_prev **=** 0x0,

\_\_next **=** 0x0}}, \_\_size **=** "\002\000\000\000\000\000\000\000\377\031\000\000\001", '\000' **<**repeats 26 times**>**, \_\_align **=** 2}

(gdb)

可以看出fun5等待的锁就是被fun5所在的线程占用了，这种一般是优先考虑优化流程，如果不好优化可以采用递归锁，如果别人的代码，修改时一定要谨慎，所有锁使用的地方都要考虑下。

**五，第三种情况**

两个锁交叉使用导致死锁，对应test2,

执行test2运行结果：

baoqingti@Cpl**-**IBP**-**Product:**~/**data**/**work**/**toolSH**/**tool**+**sh2$ .**/**deadlock

test1 begin

test1 end

threadProc2 enter **=** 2036

threadProc1 enter **=** 2036

可以看到线程1是死循环卡住了，占有了g\_mutex, 而线程2进入的时候需要等待g\_mutex可用，那么结果，就是线程2一直在等待。当然这里从代码就能看出问题了，如果在真实环境中，这种情况不加分析是不清楚是死锁还是卡主，现在通过gdb大致分析下。

GNU gdb (GDB) 7.6.1

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-unknown-linux-gnu".

For bug reporting instructions, please see:

**<**http:*//www.gnu.org/software/gdb/bugs/>...*

Reading symbols from **/**data1**/**data\_baoqingti**/**work**/**toolSH**/**tool**+**sh2**/**deadlock...done.

(gdb) r

Starting program: **/**data1**/**data\_baoqingti**/**work**/**toolSH**/**tool**+**sh2**/**deadlock

[Thread debugging **using** libthread\_db enabled]

Using host libthread\_db library "/lib/x86\_64-linux-gnu/libthread\_db.so.1".

test1 begin

[New Thread 0x7ffff6fec700 (LWP 32361)]

threadProc1 enter **=** 32357

[New Thread 0x7ffff67eb700 (LWP 32362)]

test1 end

threadProc2 enter **=** 32357

**^**C

Program received signal SIGINT, Interrupt.

0x00007ffff75be0ad in nanosleep () from **/**lib**/**x86\_64**-**linux**-**gnu**/**libc.so.6

(gdb) info **thread**

Id Target Id Frame

3 Thread 0x7ffff67eb700 (LWP 32362) "deadlock" 0x00007ffff7bcb89c in \_\_lll\_lock\_wait ()

from **/**lib**/**x86\_64**-**linux**-**gnu**/**libpthread.so.0

2 Thread 0x7ffff6fec700 (LWP 32361) "deadlock" threadProc1 () at deadlock.cpp:16

**\*** 1 Thread 0x7ffff7fef740 (LWP 32357) "deadlock" 0x00007ffff75be0ad in nanosleep () from **/**lib**/**x86\_64**-**linux**-**gnu**/**libc.so.6

(gdb) t 3

[Switching to **thread** 3 (Thread 0x7ffff67eb700 (LWP 32362))]

**#0 0x00007ffff7bcb89c in \_\_lll\_lock\_wait () from /lib/x86\_64-linux-gnu/libpthread.so.0**

(gdb) bt

**#0 0x00007ffff7bcb89c in \_\_lll\_lock\_wait () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#1 0x00007ffff7bc7065 in \_L\_lock\_858 () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#2 0x00007ffff7bc6eba in pthread\_mutex\_lock () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#3 0x000000000040091e in threadProc2 () at deadlock.cpp:23**

**#4 0x00007ffff7bc4e9a in start\_thread () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#5 0x00007ffff75f238d in clone () from /lib/x86\_64-linux-gnu/libc.so.6**

**#6 0x0000000000000000 in ?? ()**

(gdb) f 2

**#2 0x00007ffff7bc6eba in pthread\_mutex\_lock () from /lib/x86\_64-linux-gnu/libpthread.so.0**

(gdb) p g\_mutex

$1 **=** {\_\_data **=** {\_\_lock **=** 2, \_\_count **=** 0, \_\_owner **=** 32361, \_\_nusers **=** 1, \_\_kind **=** 0, \_\_spins **=** 0, \_\_list **=** {\_\_prev **=** 0x0,

\_\_next **=** 0x0}}, \_\_size **=** "\002\000\000\000\000\000\000\000i~\000\000\001", '\000' **<**repeats 26 times**>**, \_\_align **=** 2}

(gdb) GNU gdb (GDB) 7.6.1

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-unknown-linux-gnu".

For bug reporting instructions, please see:

**<**http:*//www.gnu.org/software/gdb/bugs/>...*

Reading symbols from **/**data1**/**data\_baoqingti**/**work**/**toolSH**/**tool**+**sh2**/**deadlock...done.

(gdb) r

Starting program: **/**data1**/**data\_baoqingti**/**work**/**toolSH**/**tool**+**sh2**/**deadlock

[Thread debugging **using** libthread\_db enabled]

Using host libthread\_db library "/lib/x86\_64-linux-gnu/libthread\_db.so.1".

test1 begin

[New Thread 0x7ffff6fec700 (LWP 32361)]

threadProc1 enter **=** 32357

[New Thread 0x7ffff67eb700 (LWP 32362)]

test1 end

threadProc2 enter **=** 32357

**^**C

Program received signal SIGINT, Interrupt.

0x00007ffff75be0ad in nanosleep () from **/**lib**/**x86\_64**-**linux**-**gnu**/**libc.so.6

(gdb) info **thread**

Id Target Id Frame

3 Thread 0x7ffff67eb700 (LWP 32362) "deadlock" 0x00007ffff7bcb89c in \_\_lll\_lock\_wait ()

from **/**lib**/**x86\_64**-**linux**-**gnu**/**libpthread.so.0

2 Thread 0x7ffff6fec700 (LWP 32361) "deadlock" threadProc1 () at deadlock.cpp:16

**\*** 1 Thread 0x7ffff7fef740 (LWP 32357) "deadlock" 0x00007ffff75be0ad in nanosleep () from **/**lib**/**x86\_64**-**linux**-**gnu**/**libc.so.6

(gdb) t 3

[Switching to **thread** 3 (Thread 0x7ffff67eb700 (LWP 32362))]

**#0 0x00007ffff7bcb89c in \_\_lll\_lock\_wait () from /lib/x86\_64-linux-gnu/libpthread.so.0**

(gdb) bt

**#0 0x00007ffff7bcb89c in \_\_lll\_lock\_wait () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#1 0x00007ffff7bc7065 in \_L\_lock\_858 () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#2 0x00007ffff7bc6eba in pthread\_mutex\_lock () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#3 0x000000000040091e in threadProc2 () at deadlock.cpp:23**

**#4 0x00007ffff7bc4e9a in start\_thread () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#5 0x00007ffff75f238d in clone () from /lib/x86\_64-linux-gnu/libc.so.6**

**#6 0x0000000000000000 in ?? ()**

(gdb) f 2

**#2 0x00007ffff7bc6eba in pthread\_mutex\_lock () from /lib/x86\_64-linux-gnu/libpthread.so.0**

(gdb) p g\_mutex

$1 **=** {\_\_data **=** {\_\_lock **=** 2, \_\_count **=** 0, \_\_owner **=** 32361, \_\_nusers **=** 1, \_\_kind **=** 0, \_\_spins **=** 0, \_\_list **=** {\_\_prev **=** 0x0,

\_\_next **=** 0x0}}, \_\_size **=** "\002\000\000\000\000\000\000\000i~\000\000\001", '\000' **<**repeats 26 times**>**, \_\_align **=** 2}

(gdb)

线程32362，在等锁，锁是被32361线程占用了，而32361线程回调是死循环，g\_mutex一直不会释放，那么32362

就一直在等待。这种情况分析32361线程回调卡主原因基本上就能解决问题了。

4.第二种情况

同一个线程多次lock导致死锁。对应test3

执行test3运行结果：

baoqingti@Cpl**-**IBP**-**Product:**~/**data**/**work**/**toolSH**/**tool**+**sh2$ .**/**deadlock

threadProc5 enter **=** 6612

fun5 begin

这里是fun5函数里面在等待g\_mutex，通过gdb看下g\_mutex被谁占用了，

GNU gdb (GDB) 7.6.1

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-unknown-linux-gnu".

For bug reporting instructions, please see:

**<**http:*//www.gnu.org/software/gdb/bugs/>...*

Reading symbols from **/**data1**/**data\_baoqingti**/**work**/**toolSH**/**tool**+**sh2**/**deadlock...done.

(gdb) r

Starting program: **/**data1**/**data\_baoqingti**/**work**/**toolSH**/**tool**+**sh2**/**deadlock

[Thread debugging **using** libthread\_db enabled]

Using host libthread\_db library "/lib/x86\_64-linux-gnu/libthread\_db.so.1".

[New Thread 0x7ffff6fec700 (LWP 6655)]

threadProc5 enter **=** 6651

fun5 begin

**^**C

Program received signal SIGINT, Interrupt.

0x00007ffff75be0ad in nanosleep () from **/**lib**/**x86\_64**-**linux**-**gnu**/**libc.so.6

(gdb) info **thread**

Id Target Id Frame

2 Thread 0x7ffff6fec700 (LWP 6655) "deadlock" 0x00007ffff7bcb89c in \_\_lll\_lock\_wait ()

from **/**lib**/**x86\_64**-**linux**-**gnu**/**libpthread.so.0

**\*** 1 Thread 0x7ffff7fef740 (LWP 6651) "deadlock" 0x00007ffff75be0ad in nanosleep () from **/**lib**/**x86\_64**-**linux**-**gnu**/**libc.so.6

(gdb) t 2

[Switching to **thread** 2 (Thread 0x7ffff6fec700 (LWP 6655))]

**#0 0x00007ffff7bcb89c in \_\_lll\_lock\_wait () from /lib/x86\_64-linux-gnu/libpthread.so.0**

(gdb) bt

**#0 0x00007ffff7bcb89c in \_\_lll\_lock\_wait () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#1 0x00007ffff7bc7065 in \_L\_lock\_858 () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#2 0x00007ffff7bc6eba in pthread\_mutex\_lock () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#3 0x0000000000400a64 in fun5 () at deadlock.cpp:62**

**#4 0x0000000000400aab in threadProc5 () at deadlock.cpp:70**

**#5 0x00007ffff7bc4e9a in start\_thread () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#6 0x00007ffff75f238d in clone () from /lib/x86\_64-linux-gnu/libc.so.6**

**#7 0x0000000000000000 in ?? ()**

(gdb) f 2

**#2 0x00007ffff7bc6eba in pthread\_mutex\_lock () from /lib/x86\_64-linux-gnu/libpthread.so.0**

(gdb) p g\_mutex

$1 **=** {\_\_data **=** {\_\_lock **=** 2, \_\_count **=** 0, \_\_owner **=** 6655, \_\_nusers **=** 1, \_\_kind **=** 0, \_\_spins **=** 0, \_\_list **=** {\_\_prev **=** 0x0,

\_\_next **=** 0x0}}, \_\_size **=** "\002\000\000\000\000\000\000\000\377\031\000\000\001", '\000' **<**repeats 26 times**>**, \_\_align **=** 2}

(gdb)

可以看出fun5等待的锁就是被fun5所在的线程占用了，这种一般是优先考虑优化流程，如果不好优化可

以采用递归锁，如果别人的代码，修改时一定要谨慎，所有锁使用的地方都要考虑下。

5.第三种情况

两个锁交叉使用导致死锁，对应test2,

执行test2运行结果：

baoqingti@Cpl**-**IBP**-**Product:**~/**data**/**work**/**toolSH**/**tool**+**sh2$ .**/**deadlock

threadProc3 enter **=** 11853

threadProc4 enter **=** 11853

fun4 begin

fun3 begin

看到两个线程都卡主了，gdb分析

GNU gdb (GDB) 7.6.1

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-unknown-linux-gnu".

For bug reporting instructions, please see:

**<**http:*//www.gnu.org/software/gdb/bugs/>...*

Reading symbols from **/**data1**/**data\_baoqingti**/**work**/**toolSH**/**tool**+**sh2**/**deadlock...done.

(gdb) r

Starting program: **/**data1**/**data\_baoqingti**/**work**/**toolSH**/**tool**+**sh2**/**deadlock

[Thread debugging **using** libthread\_db enabled]

Using host libthread\_db library "/lib/x86\_64-linux-gnu/libthread\_db.so.1".

[New Thread 0x7ffff6fec700 (LWP 21676)]

threadProc3 enter **=** 21672

[New Thread 0x7ffff67eb700 (LWP 21677)]

threadProc4 enter **=** 21672

fun4 begin

fun3 begin

**^**C

Program received signal SIGINT, Interrupt.

0x00007ffff75be0ad in nanosleep () from **/**lib**/**x86\_64**-**linux**-**gnu**/**libc.so.6

(gdb) info **thread**

Id Target Id Frame

3 Thread 0x7ffff67eb700 (LWP 21677) "deadlock" 0x00007ffff7bcb89c in \_\_lll\_lock\_wait ()

from **/**lib**/**x86\_64**-**linux**-**gnu**/**libpthread.so.0

2 Thread 0x7ffff6fec700 (LWP 21676) "deadlock" 0x00007ffff7bcb89c in \_\_lll\_lock\_wait ()

from **/**lib**/**x86\_64**-**linux**-**gnu**/**libpthread.so.0

**\*** 1 Thread 0x7ffff7fef740 (LWP 21672) "deadlock" 0x00007ffff75be0ad in nanosleep () from **/**lib**/**x86\_64**-**linux**-**gnu**/**libc.so.6

(gdb) t 2

[Switching to **thread** 2 (Thread 0x7ffff6fec700 (LWP 21676))]

**#0 0x00007ffff7bcb89c in \_\_lll\_lock\_wait () from /lib/x86\_64-linux-gnu/libpthread.so.0**

(gdb) bt

**#0 0x00007ffff7bcb89c in \_\_lll\_lock\_wait () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#1 0x00007ffff7bc7065 in \_L\_lock\_858 () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#2 0x00007ffff7bc6eba in pthread\_mutex\_lock () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#3 0x0000000000400958 in fun3 () at deadlock.cpp:31**

**#4 0x00000000004009d7 in threadProc3 () at deadlock.cpp:47**

**#5 0x00007ffff7bc4e9a in start\_thread () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#6 0x00007ffff75f238d in clone () from /lib/x86\_64-linux-gnu/libc.so.6**

**#7 0x0000000000000000 in ?? ()**

(gdb) t 3

[Switching to **thread** 3 (Thread 0x7ffff67eb700 (LWP 21677))]

**#0 0x00007ffff7bcb89c in \_\_lll\_lock\_wait () from /lib/x86\_64-linux-gnu/libpthread.so.0**

(gdb) bt

**#0 0x00007ffff7bcb89c in \_\_lll\_lock\_wait () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#1 0x00007ffff7bc7065 in \_L\_lock\_858 () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#2 0x00007ffff7bc6eba in pthread\_mutex\_lock () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#3 0x0000000000400986 in fun4 () at deadlock.cpp:38**

**#4 0x0000000000400a2a in threadProc4 () at deadlock.cpp:55**

**#5 0x00007ffff7bc4e9a in start\_thread () from /lib/x86\_64-linux-gnu/libpthread.so.0**

**#6 0x00007ffff75f238d in clone () from /lib/x86\_64-linux-gnu/libc.so.6**

**#7 0x0000000000000000 in ?? ()**

(gdb) p g\_mutex

$1 **=** {\_\_data **=** {\_\_lock **=** 2, \_\_count **=** 0, \_\_owner **=** 21676, \_\_nusers **=** 1, \_\_kind **=** 0, \_\_spins **=** 0, \_\_list **=** {\_\_prev **=** 0x0,

\_\_next **=** 0x0}}, \_\_size **=** "\002\000\000\000\000\000\000\000\254T\000\000\001", '\000' **<**repeats 26 times**>**, \_\_align **=** 2}

(gdb) p g\_mutex1

$2 **=** {\_\_data **=** {\_\_lock **=** 2, \_\_count **=** 0, \_\_owner **=** 21677, \_\_nusers **=** 1, \_\_kind **=** 0, \_\_spins **=** 0, \_\_list **=** {\_\_prev **=** 0x0,

\_\_next **=** 0x0}}, \_\_size **=** "\002\000\000\000\000\000\000\000\255T\000\000\001", '\000' **<**repeats 26 times**>**, \_\_align **=** 2}

(gdb)

线程21676在等待g\_mutex1，g\_mutex1被21677线程占用；线程21677在等待g\_mutex，而g\_mutex被21676占用，双方都对用了对方需要的锁，导致死锁。这类问题的解决要避免锁交叉互锁的。

## 六，死锁总结

上面的三种情况是只是单一的情况，实际应用中不会这么单一，可能上上面情况组合，比如线程1在等锁，所在线程2中占用，而线程2和线程3形成死锁。对于线程1、2来说就是第一种情况，对应线程2、3来说就是第三种情况；分析的方法都是一样的，根据锁的owner一步步去把线程理清楚就知道锁的占用关系了，基本上原因就能知道了。