<https://blog.csdn.net/cslqm/article/details/53239135>

版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/CSLQM/article/details/53239135

目录[(?)](https://blog.csdn.net/cslqm/article/details/53239135" \o "系统根据文章中H1到H6标签自动生成文章目录)[[+]](https://blog.csdn.net/cslqm/article/details/53239135)

# pthread\_exit

## 关于线程的终止

单个线程可以通过下列三种方式退出，在不终止整个进程的情况下停止它的控制流。

(1)线程只是从例程中返回，返回线程的退出码。

(2)线程可以被同一进程中的其他线程取消。

(3)线程调用pthread\_exit。

## 头文件

#include <pthread.h>

## 函数原型

void pthread\_exit(void \*retval);

## 参数

retval是一个无类型指针，与传给启动例程的单个参数类似。同一进程中的其他线程可调用pthread\_join函数访问这个指针。调用了pthread\_join的线程会一直阻塞直到它指定的线程调用pthread\_exit、从启动例程中返回或者被取消。如果线程从启动它的例程返回，retval将包含返回码。若线程被取消则retval指定的内存空间会被置为PTHREAD\_CANCELED。

## unix环境高级编程

**[cpp]** [view plain](https://blog.csdn.net/cslqm/article/details/53239135) [copy](https://blog.csdn.net/cslqm/article/details/53239135)

1. #include <pthread.h>
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include <unistd.h>
5. #include <sys/types.h>
7. **void** \*thr\_fn1(**void** \*arg)
8. {
9. printf("thread 1 returning\n");
10. **return**((**void** \*)1);
11. }
13. **void** \* thr\_fn2(**void** \*arg)
14. {
15. printf("thread 2 exiting\n");
16. pthread\_exit((**void** \*)2);
17. }
19. **int** main(**void**)
20. {
21. **int** err;
22. pthread\_t tid1, tid2;
23. **void** \*tret;
25. err = pthread\_create(&tid1, NULL, thr\_fn1, NULL);
26. **if**(err != 0)
27. {
28. printf("can't create thread 1: %s\n", strerror(err));
29. }
30. err = pthread\_create(&tid2, NULL, thr\_fn2, NULL);
31. **if**(err != 0)
32. {
33. printf("can't create thread 2: %s\n", strerror(err));
34. }
35. err = pthread\_join(tid1, &tret);
36. **if**(err != 0)
37. {
38. printf("can't join with thread 1: %s\n", strerror(err));
39. }
40. printf("thread 1 exit code %d\n", (**int**)tret);
41. err = pthread\_join(tid2, &tret);
42. **if**(err != 0)
43. {
44. printf("can't join with thread 2: %s\n", strerror(err));
45. }
46. printf("thread 2 exit code %d\n", (**int**)tret);
47. exit(0);
48. }

# 线程清理

## 头文件

#include <pthread.h>

## 函数原型

int pthread\_cancel(pthread\_t thread);  
void pthread\_cleanup\_push(void (\*routine)(void \*),  
                                 void \*arg);  
void pthread\_cleanup\_pop(int execute);

## 功能

线程可以调用pthread\_cancel函数来请求取消同一进程中的其他线程。在默认情况下pthread\_cancel函数会使由thread标识的线程的行为表现为如同调用了参数为PTHREAD\_CANCELED的pthread\_exit函数，必须特别注意的是，pthread\_cancel并不等待线程的终止。它仅仅提出请求。后两个函数被称为线程清理处理函数，线程可以建立多个清理处理函数。处理程序记录在栈空间，所以它们的执行顺序与它们的注册时的顺序相反。

## 说明

当线程执行动作(1)调用pthread\_exit时，(2)响应取消请求时，(3)用非零execute参数调用pthread\_cleanup\_pop时调用清理函数，调用参数arg,清理函数routine的调用顺序是由pthread\_cleanup\_push函数来安排的。

如果execute参数置为0，清理函数将不被调用。

## unix环境高级编程的例子

**[cpp]** [view plain](https://blog.csdn.net/cslqm/article/details/53239135) [copy](https://blog.csdn.net/cslqm/article/details/53239135)

1. #include <pthread.h>
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include <unistd.h>
5. #include <sys/types.h>
7. **void** cleanup(**void** \*arg)
8. {
9. printf("cleanup: %s\n", (**char** \*)arg);
10. }
12. **void** \*thr\_fn1(**void** \*arg)
13. {
14. printf("thread 1 start\n");
15. pthread\_cleanup\_push(cleanup, "thread 1 first handler");
16. pthread\_cleanup\_push(cleanup, "thread 1 second handler");
17. printf("thread 1 push complete\n");
18. **if**(arg)
19. {
20. **return**((**void** \*)1);
21. }
22. pthread\_cleanup\_pop(0);
23. pthread\_cleanup\_pop(0);
24. **return**((**void** \*)1);
25. }
27. **void** \*thr\_fn2(**void** \*arg)
28. {
29. printf("thread 2 start\n");
30. pthread\_cleanup\_push(cleanup, "thread 2 first handler");
31. pthread\_cleanup\_push(cleanup, "thread 2 second handler");
32. printf("thread 2 push complete\n");
33. **if**(arg)
34. {
35. pthread\_exit((**void** \*)2);
36. }
37. pthread\_cleanup\_pop(0);
38. pthread\_cleanup\_pop(0);
39. }
41. **int** main(**void**)
42. {
43. **int** err;
44. pthread\_t tid1, tid2;
45. **void** \*tret;
47. err = pthread\_create(&tid1, NULL, thr\_fn1, (**void** \*)1);
48. **if**(err != 0)
49. {
50. printf("can't create thread 1: %s\n", strerror(err));
51. }
53. err = pthread\_create(&tid2, NULL, thr\_fn2, (**void** \*)1);
54. **if**(err != 0)
55. {
56. printf("can't create thread 2: %s\n", strerror(err));
57. }
59. err = pthread\_join(tid1, &tret);
60. **if**(err != 0)
61. {
62. printf("can't join with thread 1: %s\n", strerror(err));
63. }
65. printf("thread 1 exit code %d\n", (**int**)tret);
67. err = pthread\_join(tid2, &tret);
68. **if**(err != 0)
69. {
70. printf("can't join with thread 2: %s\n", strerror(err));
71. }
73. printf("thread 2 exit code %d\n", (**int**)tret);
74. exit(0);
75. }