

原

多线程中的信号机制--sigwait()函数

2013年11月06日 18:41:29

阅读数：8537

在Linux的多线程中使用信号机制，与在进程中使用信号机制有着根本的区别，可以说是完全不同。在进程环境中，对信号的处理是，先注册信号处理函数，当信号异步发生时，调用处理函数来处理信号。它**完全是异步的**（**我们完全不知到信号会在进程的那个执行点到来！**）。然而信号处理函数的实现，有着许多的限制；比如有一些函数不能在信号处理函数中调用；再比如一些函数read、recv等调用时会被异步的信号给中断(interrupt)，因此我们必须对在这些函数在调用时因为信号而中断的情况进行处理（判断函数返回时 enno 是否等于 EINTR）。

但是在多线程中处理信号的原则却完全不同，它的基本原则是：**将对信号的异步处理，转换成同步处理**，也就是说**用一个线程专门的来“同步等待”信号的到来，而其它的线程可以完全不被该信号中断/打断(interrupt)**。这样就在相当程度上简化了在多线程环境中对信号的处理。而且可以保证其它的线程不受信号的影响。这样**我们对信号就可以完全预测**，因为它不再是异步的，而是同步的（**我们完全知道信号会在哪个线程中的哪个执行点到来而被处理！**）。而同步的编程模式总是比异步的编程模式简单。其实多线程相比于多进程的其中一个优点就是：多线程可以将进程中异步的东西转换成同步的来处理。

1. sigwait函数：

1. sigwait - wait for a signal
2.
3. #include <signal.h>
4. int sigwait(const sigset_t *set, int *sig);
5.
- Description
6. The sigwait() function **suspends** execution of the calling thread until the delivery of one
7. of the signals specified in the signal set *set*. The function **accepts** the signal (removes
8. it from the pending list of signals), and **returns** the signal number in *sig*.
9.
10. The operation of sigwait() is the same as sigwaitinfo(2), except that:
11.
12. * sigwait() only returns the signal number, rather than a siginfo_t structure describing
13. the signal.
14. * The return values of the two functions are different.
15.
16. Return Value
17.
18. On success, sigwait() returns 0. On error, it returns a positive error number.

从上面的man sigwait的描述中，我们知道：sigwait是**同步**的等待信号的到来，而不是像进程中那样是异步的等待信号的到来。sigwait函数使用一个信号集作为他的参数，并且在集合中的任一个信号发生时返回该信号值，解除阻塞，然后可以针对该信号进行一些相应的处理。

2. 记住：

在多线程代码中，总是使用sigwait或者sigwaitinfo或者sigtimedwait等函数来处理信号。而不是signal或者sigaction等函数。因为在多线程中调用signal或者sigaction等函数会改变所以线程中的信号处理函数。而不是仅仅改变调用signal/sigaction的那个线程的信号处理函数。

3. pthread_sigmask函数：

每个线程均有自己的信号屏蔽集（信号掩码）。可以使用pthread_sigmask函数来**屏蔽某个线程对某些信号的响应处理**，仅留下需要处理该信号的线程来接收该信号。实现方式是：利用线程信号屏蔽集的继承关系（在主进程中对sigmask进行设置后，主进程新创建的线程将**继承主进程的掩码**）

1. pthread_sigmask - examine and change the current thread's mask of blocked signals
2.

3. #include <signal.h>

4. `int pthread_sigmask(int how, const sigset_t *set, sigset_t *oldset);`

5.

6. Compile and link with -pthread.

7.

8. DESCRIPTION

9. The pthread_sigmask() function is just like sigprocmask(2), with the difference that **its use**

10. **in multithreaded programs** is explicitly specified by POSIX.1-2001.

11. Other differences are noted in this page.

12. For a description of the arguments and operation of this function, see sigprocmask(2).

13.

14. RETURN VALUE

15. On success, pthread_sigmask() returns 0; on error, it returns an error number.

16. NOTES

17. A new thread **inherits** a copy of its creator's signal mask.

18.

19. (from man sigprocmask:)

20.

21. The behavior of the call is dependent on the value of how, as follows.

22. **SIG_BLOCK**

23. The set of blocked signals is the **union** of the current set and the *set* argument.

24. **SIG_UNBLOCK**

25. The signals in set are removed from the current set of blocked signals. It is permissible

26. to attempt to **unblock** a signal which is not blocked.

27. **SIG_SETMASK**

28. The set of blocked signals is **set** to the argument *set*.

29.

30. If *oldset* is non-NULL, the previous value of the signal mask is stored in *oldset*.

31.

32. If *set* is NULL, then the signal mask is unchanged (i.e., *how* is ignored), but the current

33. value of the signal mask is nevertheless returned in *oldset* (if it is not NULL).

4. pthread_kill函数：

在多线程程序中，一个线程可以使用pthread_kill对同一个进程中指定的线程（包括自己）发送信号。注意在多线程中一般不使用kill函数发送信号，因为kill是对进程发送信号，结果是：正在运行的线程会处理该信号，如果该线程没有注册信号处理函数，那么会导致整个进程退出。

1. #include <signal.h>

2. `int pthread_kill(pthread_t thread, int sig);`

3.

4. Compile and link with -pthread.

5.

6. DESCRIPTION

7. The pthread_kill() function sends the signal *sig* to thread, another thread in the same

8. process as the caller. The signal is asynchronously directed to thread.

9.

10. If *sig* is 0, then no signal is sent, but error checking is still performed; this can be

11. used to check for the existence of a thread ID.

12.

13. RETURN VALUE

- 14. On success, pthread_kill() returns 0; on error, it returns an error number, and no signal
- 15. is sent.
- 16.
- 17. ERRORS
- 18. ESRCH No thread with the ID thread could be found.
- 19. EINVAL An invalid signal was specified.

5. 记住：调用sigwait同步等待的信号必须在调用线程中被屏蔽，并且通常应该在所有的线程中被屏蔽（这样可以保证信号绝不会被送到除了调用sigwait的任何其它线程），这是通过利用信号掩码的继承关系来达到的。

(The semantics of sigwait require that all threads (including the thread calling sigwait) have the signal masked, for reliable operation. Otherwise, a signal that arrives not blocked in sigwait **might be delivered to another thread.**)

6. 代码示例： (from man pthread_sigmask)

```

1. #include <pthread.h>
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include <unistd.h>
5. #include <signal.h>
6. #include <errno.h>
7.
8. /* Simple error handling functions*/
9.
10. #define handle_error_en(en, msg)\
11.     do { errno = en; perror(msg); exit(EXIT_FAILURE); } while(0)
12.
13. static void *      sig_thread(void*arg)
14. {
15.     sigset_t *set=(sigset_t*) arg;
16.     int s, sig;
17.
18.     for (;;) {
19.         s = sigwait(set, &sig);
20.         if (s != 0)
21.             handle_error_en(s, "sigwait");
22.         printf("Signal handling thread got signal %d\n", sig);
23.     }
24. }
25.
26. int      main(int argc, char*argv[])
27. {
28.     pthread_t thread;
29.     sigset_t set;
30.     int s;
31.
32.     /*
33.      Block SIGINT; other threads created by main() will inherit
34.      a copy of the signal mask.
35.      */
36.     sigemptyset(&set);
37.     sigaddset(&set, SIGQUIT);

```

```

38. sigaddset(&set, SIGUSR1);
39. s = pthread_sigmask(SIG_BLOCK, &set, NULL);
40. if (s != 0)
41.     handle_error_en(s, "pthread_sigmask");
42. s = pthread_create(&thread, NULL, &sig_thread, (void*)&set);
43. if (s != 0)
44.     handle_error_en(s, "pthread_create");
45. /*
46.  Main thread carries on to create other threads and/or do
47.  other work
48.  */
49. pause(); /* Dummy pause so we can test program */
50. return 0;
51. }

```

编译运行情况：

```

1. digdeep@ubuntu:~/pthread/learnthread$ gcc-Wall-pthread-o pthread_sigmask pthread_sigmask.c
2. digdeep@ubuntu:~/pthread/learnthread$ ./pthread_sigmask &
3. [1] 4170
4. digdeep@ubuntu:~/pthread/learnthread$ kill-QUIT%1
5. digdeep@ubuntu:~/pthread/learnthread$ Signal handling thread got signal 3
6.
7. digdeep@ubuntu:~/pthread/learnthread$ kill-USR1%1
8. digdeep@ubuntu:~/pthread/learnthread$ Signal handling thread got signal 10
9.
10. digdeep@ubuntu:~/pthread/learnthread$ kill-TERM%1
11. digdeep@ubuntu:~/pthread/learnthread$
12. [1]+ Terminated ./pthread_sigmask
13. digdeep@ubuntu:~/pthread/learnthread$

```

这个例子演示了：通过在主线程中阻塞一些信号，其它的线程会继承信号掩码，然后专门用一个线程使用sigwait函数来同步的处理信号，使其其它的线程不受到信号的影响。

```

[cpp]
1. #include <pthread.h>
2. #include <stdio.h>
3. #include <sys/signal.h>
4. #define NUMTHREADS 3
5. void sighand(int signo);
6. void *threadfunc(void *parm)
7. {
8.     pthread_t      tid = pthread_self();
9.     int             rc;
10.    printf("Thread %u entered\n", tid);
11.    rc = sleep(3);
12.    printf("Thread %u did not get expected results! rc=%d\n", tid, rc);
13.    return NULL;
14. }
15. void *threadmasked(void *parm)
16. {
17.     pthread_t      tid = pthread_self();
18.     sigset_t        mask;
19.     int             rc;
20.    printf("Masked thread %lu entered\n", tid);
21.    sigfillset(&mask); /* Mask all allowed signals */
22.    // sigemptyset(&mask);
23.    rc = pthread_sigmask(SIG_BLOCK, &mask, NULL);
24.    if (rc != 0)
25.    {
26.        printf("%d, %s\n", rc, strerror(rc));
27.        return NULL;

```

```
28.     }
29.     rc = sleep(1);
30.     if (rc != 0)
31.     {
32.         printf("Masked thread %lu did not get expected results! ""rc=%d\n",tid, rc);
33.         return NULL;
34.     }
35.     // sigwait(&mask,&rc);
36.     printf("Masked thread %lu completed masked work\n",tid);
37.     return NULL;
38. }
39. int main(int argc, char **argv)
40. {
41.     int rc;
42.     int i;
43.     struct sigaction actions;
44.     pthread_t threads[NUMTHREADS];
45.     pthread_t maskedthreads[NUMTHREADS];
46.     printf("Enter Testcase - %s\n", argv[0]);
47.     printf("Set up the alarm handler for the process\n");
48.     memset(&actions, 0, sizeof(actions));
49.     sigemptyset(&actions.sa_mask);
50.     actions.sa_flags = 0;
51.     actions.sa_handler = sighand;
52.     rc = sigaction(SIGALRM,&actions,NULL);
53.     printf("Create masked and unmasked threads\n");
54.     for(i=0; i<NUMTHREADS; ++i)
55.     {
56.         rc = pthread_create(&threads[i], NULL, threadfunc, NULL);
57.         if (rc != 0)
58.         {
59.             printf("%d, %s\n", rc, strerror(rc));
60.             return -1;
61.         }
62.         rc = pthread_create(&maskedthreads[i], NULL, threadmasked, NULL);
63.         if (rc != 0)
64.         {
65.             printf("%d, %s\n", rc, strerror(rc));
66.             return -1;
67.         }
68.     }
69.     sleep(5);
70.     printf("Send a signal to masked and unmasked threads\n");
71.     for(i=0; i<NUMTHREADS; ++i)
72.     {
73.         rc = pthread_kill(threads[i], SIGALRM);
74.         rc = pthread_kill(maskedthreads[i], SIGALRM);
75.     }
76.     printf("Wait for masked and unmasked threads to complete\n");
77.     for(i=0; i<NUMTHREADS; ++i) {
78.         rc = pthread_join(threads[i], NULL);
79.         rc = pthread_join(maskedthreads[i], NULL);
80.     }
81.     printf("Main completed\n");
82.     return 0;
83. }
84. void sighand(int signo)
85. {
86.     pthread_t tid = pthread_self();
87.     printf("Thread %lu in signal handler\n",tid);
88.     return;
89. }
90. }
```

(ps:由上面的几篇文章可知, 线程中的信号处理机制和进程中的信号处理机制是不同的, 或者说是完全不同的, 一般在多线程中使用信号的, 或单独启动一个线程类处理信号, 在主进程中发送信号, 在主进程中使用了pthread_mask()函数来处理信号屏蔽信息, 这个时候在主进程中使用kill, 这个时候线程也是可以接受到信号的原因是pthread_mask继承了主进程的信号屏蔽信息, 这个时候也是可以在使用pthread_kill给相应的线程发送相应的信号, 但是在有的程序中, 在线程中使用pthread_mask, 这个使用就必须使用pthread_kill来发送信号了;但是在进程中的信号机制没有这么复杂, 进程中只需要注册一个函数, 就静静的等待信号处理。不过在进程中使用的方法也是可以在线程中使用的)

文章标签: sigwait 信号机制

个人分类: linux多线程编程

查看更多>>

想对作者说点什么？

我来说一句

线程控制-sigwait函数和相关函数解释

首先看这篇文章学习线程和信号：http://www.cnblogs.com/clover-toeic/p/4126594.html 然后我写了一个小程序 #include #include #...

 big_bit

2016-05-12 09:06:47

阅读数：753

sigwait()函数

线程可以通过


 u012317833

2014-09-16 17:10:42

阅读数：1012

linux中使用信号--sigwait()和pthread_sigmask()

1. sigwait函数：sigwait等一个或者多个指定信号发生。 它所做的工作只有两个：第一，监听被阻塞的信号；第二，如果所监听的信号产生了，则将其从未决队列中移出来（这里实时信号和非实...

 yusiguyuan

2013-11-06 17:29:54

阅读数：8223

sigwait

int sigwait(const sigset_t *restrict sigmask, int* restrict signo); sigwait函数一直阻塞直到*sigmas...

 qq_17308321

2016-05-28 21:22:59

阅读数：429

sigwait说明

sigwait函数详解 int sigwait(const sigset_t *restrict sigmask, int* restrict signo); s...

 u010154760

2015-03-04 17:08:59

阅读数：325

sigwait函数

刚开始看sigwait函数，只是知道它是用来解除阻塞的信号，可是使我疑惑的是那么解除了以后为什么线程收到终止信号SIGINT的时候还是没能终止呢？于是网上找了一些资料，总的理解如下所示：...


 king16304

2016-08-16 14:47:17

阅读数：150

UNIX sigwait函数的用法

#include int sigwait(const sigset_t *restrict set, int *restrict signop); Returns: 0 if OK, e...

 xiaoshengqdlg

2015-03-18 14:53:47

阅读数：537

Libev源码分析06：异步信号同步化--sigwait、sigwaitinfo、sigtimedwait和signalfd

一：信号简述 信号是典型的异步事件。内核在某个信号出现时有三种处理方式：a：忽略信号，除了SIGKILL和SIGSTOP信号不能忽略外，其他大部分信号都可以被忽略；b：捕捉信号，...


 gqtcgq

2015-11-06 21:37:33

阅读数：2677

线程之间的通信（thread signal）

线程通信的目的是为了能够让线程之间相互发送信号。另外，线程通信还能够使得线程等待其它线程的信号，比如，线程B可以等待线程A的信号，这个信号可以是线程A已经处理完成的信号。通过共享对象通信有一个简单的实...

 suifeng3051 2016-07-08 17:57:04 阅读数：7126

Linux多线程中使用信号-1

在Linux的**多线程**中使用**信号机制**，与在进程中使用**信号机制**有着根本的区别，可以说是完全不同。在进程环境中，对**信号**的处理是，先注册**信号处理函数**，当**信号**异步发生时，调用处理**函数**来处理**信号**。它完全是异步的（...

 QQ276592716 2012-03-06 16:07:30 阅读数：16939

多线程中使用信号机制 pthread_sigmask()

在Linux的**多线程**中使用**信号机制**，与在进程中使用**信号机制**有着根本的区别，可以说是完全不同。在进程环境中，对**信号**的处理是，先注册**信号处理函数**，当**信号**异步发生时，调用处理**函数**来处理**信号**。它完全是异步的（...

 I_am_JoJo 2012-05-22 19:31:52 阅读数：13084

100-多线程与信号

不知道你是否还记得之前在进程中的**信号**处理时，提到过阻塞**信号**集与未决**信号**集的概念，如果你已经忘记了，请参考《阻塞**信号**与未决**信号**》一文回忆一下。1. **多线程**程序中的**信号**在**多线程**中，每一个线程都有属于自己的...

 q1007729991 2017-03-17 15:15:56 阅读数：678


线程与信号

类UNIX**信号**以前是专为进程设计的，它比线程的出现早了很多年。当线程模型出现后，专家们试图也在线程上实现**信号**，这导致了一个问题：如果要在线程模型中保持原来在进程中**信号**语意不变，是相当困难的。避免信...

 zhangfulin_hwatop 2013-02-03 21:44:23 阅读数：7924

线程中解决sigpipe信号问题

有时候线程会出现SIGPIPE错误： (gdb) bt #0 0x00110416 in __kernel_vsyscall () #1 0x0092f918 in send () f...

 suifengpiao_2011 2016-07-06 12:13:56 阅读数：1514

多线程信号总结

linux **多线程信号**总结(一) 1. 在**多线程**环境下，产生的**信号**是传递给整个进程的，一般而言，所有线程都有机会收到这个**信号**，进程在收到**信号**的的线程上下文执行**信号**处理**函数**，具体是哪个线程执行的难...

 u012007928 2015-01-30 15:56:01 阅读数：1173

每个线程都有自己的信号屏蔽字，但是信号的处理时进程中所有线程共享的http://bbs.csdn.net/topics/350222437

新手小白看《unix环境高级编程》有点困惑 一个进程不是只有一个**信号**屏蔽字吗？一个进程可以创建多个线程，那么多个线程共享一个**信号**屏蔽字呀，为什么书上12.8节里说“每个线程都有自己的...

 wangyin159 2015-10-02 16:52:40 阅读数：702


多线程与信号

参考地址：<https://www.cnblogs.com/coding-my-life/p/4782529.html>（很大帮助）http://blog.csdn.net/zhangfulin_hwatop...

 kyang_823 2017-11-10 10:45:05 阅读数：80

多线程中的信号机制--sigwait()函数

<http://blog.csdn.net/yusiguyuan/article/details/14237277> comments: **sigwait**和**sigsuspend** 一样，都是hold在...

 fdsafwagdagdg6576 2015-04-11 21:24:20 阅读数：282

sigsuspend函数和sigwait函数

#include int sigsuspend(const sigset_t *mask); **sigsuspend函数**是等待某个**信号**到达，然后调用处理**函数**之后才返回的，否则会处于阻塞等待状态...

 Tiny_210 2014-03-08 14:42:13 阅读数：536

sigsuspend 与sigwait 的区别

sigsuspend 与sigwait 的区别 sigsuspend(const sigset_t *mask) : 设置阻塞信号为mask，等待其他信号（除mask之外的信号）的发生，若信号发生且对应...

ctthunagchneg 2013-01-16 13:24:10 阅读数：2125

Qt中的信号与槽机制解析

注：要想使用Qt的核心机制信号与槽，就必须在类的私有数据区声明Q_OBJECT宏，然后会有moc编译器负责读取这个宏进行代码转化，从而使Qt这个特有的机制得到使用。所谓信号槽，简单来说，就像是插销...

bzhxuexi 2015-05-04 17:44:02 阅读数：4483

linux多线程信号处理

在linux下，每个进程都有自己的signal mask，这个信号掩码指定哪个信号被阻塞，哪个不会被阻塞，通常用调用sigmask来处理。同时每个进程还有自己的signal action，这个行为集合...

u010064842 2016-07-04 20:28:50 阅读数：952

Java并发编程系列之三十二：丢失的信号

这里的丢失的信号是指线程必须等待一个已经为真的条件，在开始等待之前没有检查等待条件。这种场景其实挺好理解，如果一边烧水，一边看电视，那么在水烧开的时候，由于太投入而没有注意到水被烧开。丢失的信号指的就...

u011116672 2016-04-14 16:35:29 阅读数：3434

Linux信号机制分析和信号处理函数

【摘要】本文分析了Linux内核对于信号的实现机制和应用层的相关处理。首先介绍了软中断信号的本质及信号的两种不同分类方法尤其是不可靠信号的原理。接着分析了内核对于信号的处理流程包括信号的触发/注册/执...

zhangchaoq 2016-04-20 16:57:53 阅读数：3071

QT 信号与槽不在同一个线程 connect

主线程中发出一个信号，另一个线程槽函数没有响应，这个问题搞了好几个小时，才发现原来是connect时候的第5个参数写错了，在这里做下备注吧。connect用于连接qt的信号和槽，在qt编程过程...

u012830675 2016-12-06 09:45:15 阅读数：5923

LINUX环境高级编程 第五章 信号

2012年08月31日 690KB 下载



android多线程并发协调semaphore机制

在项目开发过程中我们难免用到多线程机制，但不可否认的是如果对公共资源共同访问时候没有处理好线程同步的话很容易出现非线程安全的问题，会带来不可预知的错误，在java中进行线程同步的话一般都用wait和n...

limaofang 2016-04-20 17:45:23 阅读数：716

Pyqt5系列(七)-信号与槽机制

信号和槽机制是QT的核心机制，要精通QT编程就必须对信号和槽有所了解。信号和槽是一种高级接口，应用于对象之间的通信，它是QT的核心特性，也是QT区别于其它工具包的重要地方。在l...

zhulove86 2016-09-14 12:01:27 阅读数：10642

Linux并发（异步信号）

Linux下的信号是诸多进程间通信（IPC）中唯一一种异步通信方式，这决定了信号的处理跟其他的IPC有本质差别。拓展：一般情况下，进程什么时候会收到信号、收到什么信号是无法事...

vincent040 2016-03-21 11:00:36 阅读数：1810

linux下多线程编程原以及互锁、条件变量、读写锁、信号等通信机制

一、多线程编程 线程简单的说就是让多个函数同时执行，线程间通信是为了让他们有序的运行。 1.1创建一个线程就是执行一个函数。

线程 ...

 tbadolph 2017-08-04 17:43:22 阅读数：467

linux signal信号处理过程与机制--完全实例讲解

信号是Linux编程中非常重要的部分，本文将详细介绍信号机制的基本概念、Linux对信号机制的大致实现方法、如何使用信号，以及有关信号的几个系统调用。 信号机制是进程之间相互传递消息的一种方法...

 lee244868149 2014-08-20 17:08:34 阅读数：3184

Linux 多线程编程 与 信号处理

原来在main函数中把几个子线程启动后就睡10分钟后开始清理子线程后退出。现在想改成子线程启动后主线程进入无限睡眠，直到收到SIGTERM或SIGINT。主程序如下： 其他头文件 #inclu...

 yangyangye 2014-02-24 11:40:13 阅读数：2459

多线程编程（三）——线程同步——信号量

线程同步主要为了协调线程间工作，尤其是数据的使用， 第一个例子——生产者消费者： 设置： #库存最多1，有库存消费者才消费，没则等待；没库存生产者才生产，没则等待。 #并没有用semA本身做...

 huqinwei987 2016-03-17 16:34:51 阅读数：1818

Linux信号（signal）机制和Linux信号量（semaphore）机制的区别

Linux信号（signal）机制

 langjian2012 2014-10-01 13:27:59 阅读数：5712

C++多线程-第五篇-同步机制

Call_once 使用call_once包装的函数在多线程中只会被执行一次。 Void call_once(once_flag&flag, Callable && func,Args &&.....

 hffhjh111 2016-11-12 16:46:25 阅读数：832


Linux-信号机制详解（一）

之前有写过SystemV的信号量机制，现在是信号。这里的信号和前面的信号量是不同的。这里的信号是进程给操作系统或进程的某种信息，让操作系统或者其他进程做出某种反应。 信号是进程间通信机...

 leex_brave 2016-07-17 00:29:05 阅读数：4695

pyqt4 利用信号槽在子线程里面操作Qt界面

```
#-*- coding:utf-8 -*- ##### from PyQt4.QtCore import * from PyQt4.QtGui import * import sys import...
```

 caobin0825 2016-05-04 18:22:18 阅读数：721

利用信号量机制解决进程同步和互斥问题

利用信号量机制解决进程同步和互斥问题 在讨论如何用信号量机制解决这个问题之前，我们应该先了解进程同步和互斥间的一些概念。 首先是进程间的两种关系：同步和互斥。所谓同步就是把异步环境下的一组并...

 sinat_14840443 2014-10-27 15:35:40 阅读数：1898

Qt 信号和槽机制 优点 效率的详解

一、信号和槽机制 Qt提供了信号和槽机制用于完成界面操作的响应，是完成任意两个Qt对象之间的通信机制。 其中，信号会在某个特定情况或动作下被触发，槽是等同于接收并处理信号的函数。 二、信号...

 qq_21334991 2017-09-23 20:41:48 阅读数：1753

java多线程机制

一、进程与线程 1.1、任务调动 大部分操作系统(如Windows、Linux)的任务调度是采用时间片轮转的抢占式调度方式，也就是说一个任务执行一小段时间后强制暂停去执行下一个任务，每个任务轮流执...

 makeprogresszb

2016-01-10 15:39:13

阅读数：1769

个人资料



鱼思故渊

关注

原创	粉丝	喜欢	评论
854	1325	58	348

等级： 博客 日 访问：247万+

积分：3万+ 排名：173

勋章： 

最新文章

- 我为什么那么喜欢呆在实验室，因为我知道我错过就再也没有机会
- STL源码剖析--vector
- 二叉树中和为某一值的路径
- 求一个数组中的逆序对
- 32位系统和64位系统的区别

博主专栏

- | | | |
|---|--------------|------|
|  | muduo源码分析 | |
| | 阅读量：26534 | 14 篇 |
|  | libevent源码分析 | |
| | 阅读量：82401 | 31 篇 |
|  | TCP/IP详解 | |
| | 阅读量：81153 | 23 篇 |
|  | linux内核分析 | |
| | 阅读量：6358 | 1 篇 |

个人分类

- | | |
|----------------|------|
| ubuntu开发环境常见问题 | 34篇 |
| windows开发环境搭建 | 8篇 |
| c程序设计语言 | 27篇 |
| C语言 | 69篇 |
| C++ | 106篇 |

展开

归档

- | | |
|----------|----|
| 2015年11月 | 1篇 |
| 2015年9月 | 7篇 |
| 2015年8月 | 8篇 |

2015年7月	6篇
2015年6月	1篇
展开	

热门文章

- TCP/IP详解--TCP连接中TIME_WAIT状态过多

阅读量：54086
- ubuntu 下出现E: Sub-process /usr/bin/dpkg returned an error code

阅读量：53171
- epoll机制:epoll_create、epoll_ctl、epoll_wait、close

阅读量：48794
- shell脚本--if判断（数字条件、字符串条件）

阅读量：41841
- win7下安装JDK并且配置环境变量

阅读量：33122

最新评论

- TCP/IP详解--TCP连接中T...

wgslucky：讲的非常详细，也容易很懂。如果服务遇到ddos攻击该如何处理呢？
- linux字符设备驱动程序scul...

cky070406：楼主，模块退出的时候难道不用释放kfree(mem_devp[i].data)吗？
- 智能指针 shared_ptr 的...

wzw88486969：这样测试感觉不对[reply]sodawaterer[/reply]
- C程序设计语言--全局/局部变量、...

qq_42222102：谢谢！很有用
- 我为什么那么喜欢呆在实验室，因为我...

amoscykl：博主是西工大的吧~我离你不远，西电hnhh

联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

👤 QQ客服 💬 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心