<https://www.cnblogs.com/kex1n/p/5588845.html>

http://www.linuxidc.com/Linux/2013-09/90156.htm

今天在网上查了一下Linux中对进程的挂起与恢复的实现，相关资料少的可怜，大部分都是粘贴复制。也没有完整详细的代码。故自己整理了一下

程序流程为：主线程创建子线程（当前子线程状态为stop停止状态），5秒后主线程唤醒子线程，10秒后主线程挂起子线程，15秒后主线程再次唤醒子线程，20秒后主线程执行完毕等待子线程退出。

代码如下：  
#include  
#include  
#include  
#include  
#include

#define RUN 1  
#define STOP 0

pthread\_mutex\_t mut = PTHREAD\_MUTEX\_INITIALIZER;  
pthread\_cond\_t cond = PTHREAD\_COND\_INITIALIZER;

int status = STOP;  
void \* thread\_function(void)  
{  
    static int i = 0;  
    while (1)   
    {    
        pthread\_mutex\_lock(&mut);  
        while (!status)  
        {  
            pthread\_cond\_wait(&cond, &mut);  
        }  
        pthread\_mutex\_unlock(&mut);  
      
        printf("child pthread %d\n", i++);  
        if (i == 20)   
            break;  
        sleep(1);  
    }    
}

void thread\_resume()  
{  
    if (status == STOP)  
    {    
        pthread\_mutex\_lock(&mut);  
        status = RUN;  
        pthread\_cond\_signal(&cond);  
        printf("pthread run!\n");  
        pthread\_mutex\_unlock(&mut);  
    }    
    else  
    {    
        printf("pthread run already\n");  
    }    
}

void thread\_pause()  
{  
    if (status == RUN)  
    {    
        pthread\_mutex\_lock(&mut);  
        status = STOP;  
        printf("thread stop!\n");  
        pthread\_mutex\_unlock(&mut);  
    }    
    else  
    {    
        printf("pthread pause already\n");  
    }  
}

int main()  
{  
    int err;  
    static int i = 0;  
    pthread\_t child\_thread;

#if 0  
    if (pthread\_mutex\_init(&mut, NULL) != 0)  
        printf("mutex init error\n");  
    if (pthread\_cond\_init(&cond, NULL) != 0)  
        printf("cond init error\n");  
#endif

    err = pthread\_create(&child\_thread, NULL, (void \*)thread\_function, NULL);  
    if (err != 0 )  
        printf("can't create thread: %s\n", strerror(err));  
    while(1)  
    {  
        printf("father pthread %d\n", i++);  
        sleep(1);  
        if (i == 5)  
            thread\_resume();  
        if (i == 10)  
            thread\_pause();  
        if (i == 15)  
            thread\_resume();  
        if (i == 20)  
            break;  
    }  
    if (0 == pthread\_join(child\_thread, NULL))  
        printf("child thread is over\n");  
    return 0;  
}

**相关阅读：**

对Linux中多线程编程中pthread\_join的理解 <http://www.linuxidc.com/Linux/2013-09/89931.htm>

Linux多线程编程时如何查看一个进程中的某个线程是否存活 <http://www.linuxidc.com/Linux/2013-09/89930.htm>

有关Linux下线程的创建 <http://www.linuxidc.com/Linux/2013-08/88530.htm>

Linux内核线程死锁或死循环之后如何让系统宕机重启 <http://www.linuxidc.com/Linux/2013-04/82063.htm>

Linux下C语言实现多线程文件复制 <http://www.linuxidc.com/Linux/2013-03/81373.htm>