

```

doc_thread.cpp

#include "Runnable.h"
extern "C" void* _runnable(
    void * _pObject
);
class GThread
    : public GThreadBase
{
public:
    virtual int StartThread( int _nInterval );
    virtual int StopThread( int _nMode );
    virtual int Join();
    virtual int CheckThread();
    int ActivCount( int &_nCount )
    {
        _nCount += ( this->pObj->isRunning() == true ? 1 : 0 );
        return RT_S_SUCCESS;
    }
    int ThreadCount( int &_nCount )
    {
        _nCount += 1;
        return RT_S_SUCCESS;
    }
    int GetMaxPriority( int &_nPriority )
    {
        if( this->oParam.sched_priority > _nPriority )
        {
            _nPriority = this->oParam.sched_priority;
        }
        return RT_S_SUCCESS;
    }
protected:
    bool                bRunning;
    pthread_t           nThreadID;
    int                 nPolicy;
    int                 nScope;
    struct sched_param  oParam;
    Runnable            *pObj;
    GThread( Runnable   *_pObject
            , int        _nSetPriority
            , int        _nSetPolicy
            , int        _nSetScope
    );
    virtual ~GThread() {}
    virtual int KillThread( int nSignal );
    friend class GThreadGroup;
private:
    GThread( const GThread &_oCopy );
    GThread &operator = ( const GThread &_oCopy ) { return *this; }
};
#endif
#pragma ident "@(#)GThread.cpp 1.4 10/09/11 11:37:34 @Symbolic Technology."
#include "GThread.h"
#include <string.h>
#include <signal.h>
#include <errno.h>
#include <time.h>
extern "C" void* _runnable(
    void * _pObject
)
{
    void* pResult;
    pResult = NULL;
    if( _pObject == NULL )
    {
        return ( void* )( RT_F_NULL_POINTER );
    }
    Runnable *pThread = reinterpret_cast< Runnable* >( _pObject );
    pThread->bRunning = true;
    pResult = pThread->RunThread();
    pThread->bRunning = false;
    return pResult;
}
GThread::GThread(
    Runnable   *_pObject
    , int       _nSetPriority
    , int       _nSetPolicy
    , int       _nSetScope
)
: pObj( _pObject )
, nPolicy( _nSetPolicy )
, nScope( _nSetScope )
{

```