

doc_thread.cpp

GThreadBase.h, Thread.h, Thread.cpp

```
#ifndef _GTHREADBASE_
#define _GTHREADBASE_
#include <pthread.h>
#include <sched.h>
class GThreadBase
{
public:
    static const int THREAD_STOP_NOMAL    = 0;
    static const int THREAD_STOP_JOIN     = 1;
    static const int THREAD_STOP_TERM     = 2;
    static const int THREAD_STOP_KILL     = 3;
protected:
    static const int THREAD_PRIORITY_MAX  = 5;
    static const int THREAD_PRIORITY_HIGH = 4;
    static const int THREAD_PRIORITY_MID  = 3;
    static const int THREAD_PRIORITY_LOW  = 2;
    static const int THREAD_PRIORITY_MIN  = 1;
    static const int THREAD_PRIORITY_DEF  = 1;
    static const int THREAD_POLICY_DEF    = SCHED_OTHER;
    static const int THREAD_POLICY_FIFO   = SCHED_FIFO;
    static const int THREAD_POLICY_RR     = SCHED_RR;
    static const int THREAD_SCOPE_SYSTEM  = PTHREAD_SCOPE_SYSTEM;
    static const int THREAD_SCOPE_PROCESS = PTHREAD_SCOPE_PROCESS;
    virtual ~GThreadBase() {}
    virtual int StartThread( int _nInterval ) = 0;
    virtual int StopThread( int _nMode ) = 0;
    virtual int ActivCount( int &_nCount ) = 0;
    virtual int ThreadCount( int &_nCount ) = 0;
    virtual int CheckThread() = 0;
    virtual int GetMaxPriority( int &_nPriority ) = 0;
    virtual int Join( void ) = 0;
};
#endif
#ifdef _THREAD_H_
#define _THREAD_H_
#include <pthread.h>
#include <sched.h>
#include <signal.h>
#include <errno.h>
#include "rt_sts.h"
#define THREAD_SUCCESS      RT_S_SUCCESS
#define THREAD_ERROR        RT_E_ERROR
#define THREAD_ALREADYACTIVE -10
#define THREAD_SIGNAL_INVALID -21
#define THREAD_NOT_FOUND    -22
#define THREAD_DEADLOCK     -23
#define THREAD_PRIORITY_MAX  5
#define THREAD_PRIORITY_HIGH  4
#define THREAD_PRIORITY_MID   3
#define THREAD_PRIORITY_LOW   2
#define THREAD_PRIORITY_MIN   1
#define THREAD_PRIORITY_DEF   1
#define THREAD_POLICY_DEF     SCHED_OTHER
#define THREAD_POLICY_FIFO    SCHED_FIFO
#define THREAD_POLICY_RR      SCHED_RR
#define THREAD_SCOPE_SYSTEM   PTHREAD_SCOPE_SYSTEM
#define THREAD_SCOPE_PROCESS  PTHREAD_SCOPE_PROCESS
#ifdef MS_SOLARIS
extern "C" {
    void *_run_thread( void *_pObject );
}
#endif
class Thread
{
public:
    Thread( int _nSetPriority    = THREAD_PRIORITY_DEF,
            int _nSetPolicy      = SCHED_OTHER,
            int _nSetScope       = THREAD_SCOPE_SYSTEM
    );
    virtual int StopLoop( void );
    virtual int StartThread( void );
    virtual int StopThread( void );
    virtual int SetPolicy( int _nSetPriority, int _nSetPolicy );
    static pthread_cond_t thread_condition;
    static pthread_mutex_t thread_cond_mutex;
    static int DeleteCondition( void );
    int nPolicy;
    int nScope;
    struct sched_param oParam;
protected:
    virtual void* RunThread( void ) = 0;
};
```