

```

doc_GThreadManager.cpp

GThreadManager: h.GThreadGroup, h.GThreadGroup.cpp,
#ifndef _GTHREAD_MANAGER_
#define _GTHREAD_MANAGER_
#include "GThreadGroup.h"
#include "GThread.h"
class GThreadManager
{
public: GThreadGroup

public:
static GThreadManager* getInstance();
GThreadGroup* CreateGroup( const char *skey );
static void DeleteInstance();
private:
static GThreadManager *pManager;
static const char *sGroupNames;
GThreadManager( const char *skey ) : GThreadGroup( skey ) {}
virtual ~GThreadManager() {}
};
#endif
#ifndef GTHREADGROUP_
#define GTHREADGROUP_
#include "GThreadBase.h"
#include "Runnable.h"
#include "Thread.h"
#include "String.h"
#include "Mutex.h"
#include "TList.h"
#include "Time.h"
class GThreadGroup
{
public: GThreadBase

public:
virtual int AddThread(
const char *skey
, Runnable *pObj
, int _nsetPriority = THREAD_PRIORITY_DEF
, int _nsetPolicy = SCHED_OTHER
, int _nsetScope = THREAD_SCOPE_SYSTEM
);
virtual int AddGroup( GThreadGroup *pObject );
virtual int StartThread( int _nInterval )
{
return this->ExecAll100ject( EXEC_START_THREAD, _nInterval );
}
virtual int StopThread( int _nMode )
{
return this->ExecAll100ject( EXEC_STOP_THREAD, _nMode );
}
virtual int ActiveCount( int &nCount )
{
return this->ExecAll100ject( EXEC_ACTIVE_COUNT, _nCount );
}
virtual int ThreadCount( int &nCount )
{
return this->ExecAll100ject( EXEC_THREAD_COUNT, _nCount );
}
virtual int CheckThread()
{
int nMany = 0;
return this->ExecAll100ject( EXEC_CHECK_THREAD, nMany );
}
virtual int GetMaxPriority( int &nPriority )
{
return this->ExecAll100ject( EXEC_MAX_PRIORITY, _nPriority );
}
virtual int Join()
{
int nMany = 0;
return this->ExecAll100ject( EXEC_THREAD_JOIN, nMany );
}
protected:
static const int EXEC_START_THREAD = 0;
static const int EXEC_STOP_THREAD = 1;
static const int EXEC_ACTIVE_COUNT = 2;
static const int EXEC_THREAD_COUNT = 3;
static const int EXEC_CHECK_THREAD = 4;
static const int EXEC_MAX_PRIORITY = 5;
static const int EXEC_THREAD_JOIN = 6;
Mutex String: GThreadBase > oThreadMap;
Mutex String: oMutex;
const String: sGroupNames;
friend class GThreadManager;

```

```

doc_GThreadManager.cpp

GThreadGroup( const char *sGroupNames );
virtual ~GThreadGroup();
virtual int ExecAll100ject( int _nExec, int &nArg );
virtual int DeleteObject( const char *skey );
virtual int Delete100ject();
};
#endif
#ifndef GTHREADGROUP_CPP_
#define GTHREADGROUP_CPP_
#include "TList.h"
#include "Mutex.h"
#include "String"
static const char *sProc[] = {
"Start thread"
, "Stop thread"
, "Check active count"
, "Check thread count"
, "Check thread status"
, "Get max priority"
, "Thread join"
};
GThreadGroup::GThreadGroup(
const char *sGroupNames
)
: sGroupNames( sGroupNames )
{
}
GThreadGroup::~GThreadGroup(
void
)
{
this->DeleteAll100ject();
}
int GThreadGroup::AddThread(
const char *skey
, Runnable *pObject
, int _nsetPriority
, int _nsetPolicy
, int _nsetScope
)
{
int nStatus;
GThreadBase *pThread;
String skey;
static const char *sKey = "AddThreadKey[ %s ]YGroup[ %s ]";
nStatus = 0;
pThread = NULL;
NULL_RETURN( nStatus );
NULL_RETURN( _pObject );
pThread = new( std::nothrow )GThread(
_pObject
, _nsetPriority
, _nsetPolicy
, _nsetScope
);
};
ALLOC_RETURN( pThread );
skey = skey;
nStatus = this->oMutex.Lock();
ERR_RETURN( nStatus );
nStatus = this->oThreadMap.Put( skey, pThread );
if( ERR_CHECK( nStatus ) )
{
this->oMutex.Unlock();
return nStatus;
}
nStatus = this->oMutex.Unlock();
ERR_RETURN( nStatus );
return RT_S_SUCCESS;
}
int GThreadGroup::AddGroup(
GThreadGroup *pObject
)
{
int nStatus;
const char *sKey = "AddGroupKey[ %s ]YGroup[ %s ]";
nStatus = 0;
NULL_RETURN( _pObject );
nStatus = this->oMutex.Lock();
ERR_RETURN( nStatus );
nStatus = this->oThreadMap.Put( _pObject->sGroupNames , _pObject );
if( ERR_CHECK( nStatus ) )
{
this->oMutex.Unlock();
}
}

```