**定时器池v0.9**

**研发三部**

模块名：<weedong/core/timer/timer.h>

依赖: <weedong/core/time/time.h>

简介(详细请看<weedong/core/timer/timer.h> )：

在指定日期时间执发相应注册的函数指针

结构(类)：

struct WDSTimer; // 定时器基本结构

enum WDETimeLoopType; // 定时器循环定义间隔执行类型

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description: // 定时器时间状态更新后触发的回调函数

Param: // pvParam : 用户在注册时所指定的参数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

typedef void (\_\_cdecl\* FTimerCallback)( void \*pvParam ); //定时器触发函数指针类型

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description: // 创建一个定时器池

Return: // 返回一个新定时器池对象

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

extern "C" WDSTimer\* timer\_create( void );

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description: // 锁毁一个定时器池

Input: // pTimer : 定时器对象

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

extern "C" void timer\_dispose( WDSTimer \*pTimer );

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description: // 更新一个定时器时间状态

Input: // pTimer : 定时器对象

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

extern "C" void timer\_update( WDSTimer \*pTimer );

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description: // 增加一个定时循环

Input: // pTimer : 定时器对象

// pszTimerset : 执行开始日期, 可接受以下格式

"2011-10-01 10:00:00" -> 完整格式

"2011-10-01" -> 接口会补充当前系统时间作为pszTimerset 的完整格式, 例如: 目前时间是17:06:31, 补全后, "2011-10-01 17:06:31"

"10:00:00" -> 接口会补充当前系统日期作为pszTimerset 的完整格式, 例如: 目前日期是2011-10-24, 补全后, "2011-10-24 10:00:00"

// eType : 定时器执行间隔类型

// uiDelay : 定时器执行间隔值

// uiLoopCount : 定时器执行次数, uiLoopCount == 0 将无限循环执行, uiLoopCount > 0 时为该时间循环的执行次数

// pfCallback : 定时器时间状态更新后执行的回调函数接口

// pvParam : 定时器时间状态更新后执行的回调函数参数

//

Return // 返回定义定时执行的唯一标识(Guid)

Other: // 注意, 定义的pszTimerset 如果在当前系统时间之前的所有执行点将被忽略( uiLoopCount > 0 时不扣取执行次数)

当系统时间改变后调用timer\_update 接口更新时间状态时, 跨个以上(含个)执行点产生时, 会被逐一执行并逐一扣除uiLoopCount

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

extern "C" uint32 timer\_add\_loop( WDSTimer \*pTimer, char \*pszTimeset, WDETimeLoopType eType,

uint32 uiDelay, uint32 uiLoopCount,

FTimerCallback pfCallback, void \*pvParam );

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description: // 增加一个延时循环

Input: // pTimer : 定时器对象

// uiSeconds : 在当前系统时间uiSeconds 秒后被执行

// pfCallback : 定时器时间状态更新后执行的回调函数接口

// pvParam : 定时器时间状态更新后执行的回调函数参数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

extern "C" uint32 timer\_add\_call(WDSTimer \*pTimer, uint32 uiSeconds,

FTimerCallback pfCallback, void \*pvParam );

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Description: // 删除一个定时执行

Input: // pTimer : 定时器对象

// uiLoopId : 定时器执行Id, timer\_add\_loop 和timer\_add\_call 的返回值

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

extern "C" void timer\_del( WDSTimer \*pTimer, uint32 uiLoopId );

/\*===============================示例============================\*/

#include <weedong/core/timer/timer.h>

void myFunc(void\* pvData)

{

printf( “name:%s”, (char\*)pvData );

}

main()

{

WDSTimer \*pTimer = timer\_create();

//增加一个指定日期开始的定时触发器, 每分钟触发一次, 共触发次

//uiLoopId 保存时间触发器的唯一Id

uint32 uiLoopId = timer\_add\_loop(

pTimer, //定时器池对象

"2011-10-24 17:35:00", //触发点

etlMinute, //触发间隔类型

2, //触发间隔值

10, //触发次数

myFunc, //触发函数指针

“Alex” //触发函数参数指针

);

Sleep( 120/\*秒\*/ );

//更新定时器池的时间状态

timer\_update( pTimer );

//删除一个定时触发器

timer\_del( pTimer, uiLoopId );

//释放定时器池

timer\_dispose( pTimer );

}