
Subject: Re: Thread::GetCurrentThreadId() and Thread::GetCurrentThreadHandle()
new methods

Posted by [tojocky](#) on Wed, 12 Jan 2011 11:36:52 GMT

[View Forum Message](#) <> [Reply to Message](#)

mirek wrote on Wed, 12 January 2011 10:49

I mean we are planning to have:

```
static Handle GetCurrentHandle(){  
static Id GetCurrentId(){
```

but we only have now

```
    Handle GetHandle();
```

is not

```
    Id GetId();
```

missing?

You are right,

Thank you for explanation.

Finally, The class Thread should have api callback.

header code:

```
class Thread : NoCopy {  
#ifdef PLATFORM_WIN32  
    HANDLE    handle;  
    DWORD    thread_id;  
#endif  
#ifdef PLATFORM_POSIX  
    pthread_t handle;  
    pthread_t thread_id;  
#endif  
public:  
    bool    Run(Callback cb);
```

```

void    Detach();
int     Wait();

bool    IsOpen() const    { return handle; }

#ifdef PLATFORM_WIN32
typedef HANDLE Handle;
typedef DWORD Id;
#elseif
#ifdef PLATFORM_POSIX
typedef pthread_t Handle;
typedef pthread_t Id;
#elseif
typedef qword IdCustom;
Handle    GetHandle() const { return handle; }
Id        GetId() const {return thread_id;}

void      Priority(int percent); // 0 = lowest, 100 = normal

static void Start(Callback cb);

static void Sleep(int ms);

static bool IsST();
static bool IsMain();
static int  GetCount();
static void ShutdownThreads();
static bool IsShutdownThreads();
#ifdef PLATFORM_WIN32
static Handle GetCurrentHandle(){
    return GetCurrentThread();
}
static inline Id GetCurrentId(){
    return ::GetCurrentThreadId();
};
#elif defined(PLATFORM_POSIX)
static Handle GetCurrentHandle(){
    return pthread_self();
}
static inline Id GetCurrentId(){
    return pthread_self();
};
#endif
#endif

Thread();
~Thread();

private:

```

```

void operator=(const Thread&);
Thread(const Thread&);
};

```

and cpp code:

```

Thread::Thread()
{
    sMutexLock();
#ifdef PLATFORM_WIN32
    handle = 0;
    thread_id = 0;
#endif
#ifdef PLATFORM_POSIX
    handle = 0;
    thread_id = 0;
#endif
}

void Thread::Detach()
{
#ifdef PLATFORM_WIN32
    if(handle) {
        CloseHandle(handle);
        handle = 0;
        thread_id = 0;
    }
#elif defined(PLATFORM_POSIX)
    if(handle) {
        CHECK(!pthread_detach(handle));
        handle = 0;
        thread_id = 0;
    }
#endif
}

bool Thread::Run(Callback _cb)
{
    AtomicInc(sThreadCount);
    if(!threadr)
#ifdef CPU_BLACKFIN
        threadr = sMain = true;
#else
    {
        threadr = true;
        //the sMain replacement

```

```

#ifdef PLATFORM_POSIX
    pthread_t thid = pthread_self();
    vm.Enter();
    if(threadsv.Find(thid) < 0){
        //thread not yet present, mark present
        threadsv.Add(thid);
    }
    else
        RLOG("BUG: Multiple Add in Mt.cpp");
    vm.Leave();
#endif
}
#endif
Detach();
Callback *cb = new Callback(_cb);
#ifdef PLATFORM_WIN32
    handle = (HANDLE)_beginthreadex(0, 0, sThreadRoutine, cb, 0, ((unsigned int *)&thread_id));
#endif
#ifdef PLATFORM_POSIX
    if(pthread_create(&handle, 0, sThreadRoutine, cb))
        handle = 0;
    thread_id = handle;
#endif
return handle;
}

```

mirek wrote on Wed, 12 January 2011 10:49

As for TLS method, without such GetId() you definitely do not need to add it as part of interface... You can define TLS variable anywhere you need to do this fast And it is true that system Id has some expected meaning...

I agree with you!

And Thank you for a good cooperation.