VS2015中控制台程序获取系统时间的方法

/////////////////////////////////////////////////////////////////////////////

**方法1：**

在VS2015的控制台程序中，可以通过C++标准库函数、Windows API函数和C++11的函数实现。

1 C++标准库函数

通过C++标准库函数time()和localtime\_s()获取系统当前时间。

1.1 time()函数

1.1.1 功能

该函数的作用是获取系统当前时间距离1970年1月1日零时的时间，以秒作为单位。

1.1.2 格式

该函数的格式为

time\_t time(time\_t\* time);

其中，参数time是一个指针，获取到的时间就保存到该指针指向的内容中。如果该参数是NULL，则不保存获取到的时间。该函数的返回值即为获取到的时间。time\_t是一个long integer类型。

1.2 localtime\_s()函数

1.2.1 功能

该函数的功能是将time()函数获取到的秒数转换为当前的年月日时分秒。

1.2.2 格式

该函数的格式为

errno\_t localtime\_s(struct tm\* \_tm, const time\_t \*time);

其中，参数\_tm是tm结构的指针，转换后的时间就保存在该指针指向的内容中。

tm的结构定义如下

struct tm {

int tm\_sec; // seconds after the minute - [0, 60] including leap second

int tm\_min; // minutes after the hour - [0, 59]

int tm\_hour; // hours since midnight - [0, 23]

int tm\_mday; // day of the month - [1, 31]

int tm\_mon; // months since January - [0, 11]

int tm\_year; // years since 1900

int tm\_wday; // days since Sunday - [0, 6]

int tm\_yday; // days since January 1 - [0, 365]

int tm\_isdst; // daylight savings time flag

};

该结构中包含了时间信息。参数time是“1.1 time()函数”中获取的时间秒数。如果转换成功，则localtime\_s()函数的返回值是0，否则返回错误代码。

1.3 相关代码

获取当前系统时间的代码如下所示

time\_t tt = time(NULL);

tm t;

localtime\_s(&t, &tt);

printf("%d-%02d-%02d %02d:%02d:%02d",

t.tm\_year + 1900, t.tm\_mon, t.tm\_mday, t.tm\_hour, t.tm\_min, t.tm\_sec);

需要注意的是，使用time()函数与localtime\_s()函数，需要添加time.h头文件。

//////////////////////////////////////////////////////////////////////////////

**方法2：**

2 Windows API函数

通过Windows API的GetLocalTime()函数获取当前系统时间。

2.1 函数格式

void WINAPI GetLocalTime(

  \_\_out  LPSYSTEMTIME LpSystemTime

）；

其中，参数LpSystemTime是SYSTEMTIME结构的指针，用于保存获取到的时间

typedef struct \_SYSTEMTIME {

WORD wYear;

WORD wMonth;

WORD wDayOfWeek;

WORD wDay;

WORD wHour;

WORD wMinute;

WORD wSecond;

WORD wMilliseconds;

} SYSTEMTIME, \*PSYSTEMTIME;

从以上定义可以看出，SYSTEMTIME不仅包含年月日时分秒，还包含更小的毫秒单位。

2.2 相关代码

通过GetLocalTime()函数获取当前系统时间的代码如下所示：

SYSTEMTIME st;

GetLocalTime(&st);

printf("%d-%02d-%02d %02d:%02d:%02d:%03d",

st.wYear, st.wMonth, st.wDay, st.wHour, st.wMinute, st.wSecond, st.wMilliseconds);

2.3 GetLocalTime()与GetSystemTime()的区别

GetSystemTime()的使用方法与GetLocalTime()函数相同。区别在于，GetLocalTime()获取到的时间是本机时间，而GetSystemTime()获取到的时间是格林尼治时间。如图3所示。

需要注意的是，在使用Windows API函数时，需要包含Windows.h头文件。

////////////////////////////////////////////////////////////////////////////

**方法3：**

3 C++11新功能

在C++11中，使用std::chrono库中的system\_clock类来获取当前系统时间。

3.1 获取当前系统时间

system\_clock类表示了系统范围内的实时挂钟。相关代码如下所示：

auto start = chrono::system\_clock::now();

time\_t start\_time = chrono::system\_clock::to\_time\_t(start);

printf("%s\n", ctime(&start\_time));

其中，now()函数得到类型为

chrono::time\_point<std::chrono::system\_clock>的返回值start；to\_time\_t()函数是system\_clock类的静态函数，将now()函数的返回值转换为time\_t类型的对象start\_time；最后，通过转换后的start\_time构造一个ctime类的对象，并打印出来，如图4所示。

3.2 获取时间间隔

还通过system\_clock类的now()函数获取时间间隔。代码如下所示

auto start = std::chrono::system\_clock::now();

printf("打印信息所需的时间是:\n");

auto end = std::chrono::system\_clock::now();

chrono::duration<double> elapsed\_seconds = end - start;

printf("%lf秒.\n", elapsed\_seconds.count());

通过now()函数获取打印信息前后的系统时间start和end，计算出打印信息所需的时间。

**3.3 根据GetLocalTime获取的时间获取时间间隔**

|  |
| --- |
| *cout* << "-------------------------------" << *endl*;  *cout* << "获取时间间隔：" << *endl*;  //根据CTime中GetCurrentTime也可以获取时间  //Sleep(100);  *SYSTEMTIME* sysEndTime = {0};  *GetLocalTime*(&sysEndTime);    char strStartTime[64] = {0};  *sprintf\_s*(strStartTime, "%d-%02d-%02d %02d:%02d:%02d", sysStartTime.*wYear*, sysStartTime.*wMonth*, sysStartTime.*wDay*, sysStartTime.*wHour*, sysStartTime.*wMinute*, sysStartTime.*wSecond*);  *cout* << "开始时间：" << strStartTime << *endl*;  char strEndTime[64] = { 0 };  *sprintf\_s*(strEndTime, "%d-%02d-%02d %02d:%02d:%02d", sysEndTime.*wYear*, sysEndTime.*wMonth*, sysEndTime.*wDay*, sysEndTime.*wHour*, sysEndTime.*wMinute*, sysEndTime.*wSecond*);  *cout* << "结束时间：" << strEndTime << *endl*;  //CTime相减得到CTimeSpan;  *COleDateTime* cStartTime(sysStartTime);  *COleDateTime* cEndTime(sysEndTime);  *COleDateTimeSpan* cCastTimeSpan = cEndTime - cStartTime;  int nCastTime = cCastTimeSpan.*GetTotalSeconds*();  *cout* << "时间间隔是：" << nCastTime << "s" << *endl*; |

/////////////////////////////////////////////////////////////////////////

#include <iostream>

#include <windows.h>

#include <psapi.h>

#include<ctime>

#pragma comment(lib,"psapi.lib")

using namespace std;

// 获取系统内存

void showMemoryInfo(void)

{

HANDLE handle = GetCurrentProcess();

PROCESS\_MEMORY\_COUNTERS pmc;

GetProcessMemoryInfo(handle, &pmc, sizeof(pmc));

cout << "内存使用:" << pmc.WorkingSetSize / 1000 << "K/" << pmc.PeakWorkingSetSize / 1000 << "K + " << pmc.PagefileUsage / 1000 << "K/" << pmc.PeakPagefileUsage / 1000 << "K" << endl;

}

// 主函数

int main(int argc, char\* argv)

{

// ①：

showMemoryInfo();

cout << "回收所有可回收的内存" << endl;

EmptyWorkingSet(GetCurrentProcess());

showMemoryInfo();

cout << "开始动态分配内存" << endl;

char\* buf[5];

for (int i = 0; i < sizeof(buf) / sizeof(char\*); i++)

{

buf[i] = new char[102400];

showMemoryInfo();

}

cout << "开始释放内存" << endl;

for (int i = 0; i < sizeof(buf) / sizeof(char\*); i++)

{

delete buf[i];

buf[i] = NULL;

showMemoryInfo();

}

cout << "回收所有可回收的内存" << endl;

EmptyWorkingSet(GetCurrentProcess());

showMemoryInfo();

cout << endl;

// ②：获取系统时间

time\_t begin, end;

double ret;

begin = clock();

cout << "Begin time is: " << double(begin) / CLOCKS\_PER\_SEC << endl;

for (int i = 0; i < 100000; i++) {

//cout << "you are a good child!" << endl;

cout << " ";

}

end = clock();

cout << "End time is: " << double(end) / CLOCKS\_PER\_SEC << endl;

ret = double(end - begin) / CLOCKS\_PER\_SEC;

cout << "runtime:" << ret << endl;

return 0;

}