

Forum:[Hack Forums](#)

Thread: [CPU Usage in Percent](#)

?onsai - November 26, 2014, 4:20 pm

Hi HF,

Anyone knows whats the simpliest way on Win32 C++ to determine the CPU Usage?

I tried to do it with PDH but not successful. He said my CPU has a Loadage of 1.7932e-307 :D

Help please

Cheers ?

sqsgkrldhwg - November 26, 2014, 4:58 pm

Hi,

Where is your code ?

Did you read the MSDN reference :

<http://msdn.microsoft.com/en-us/library/windows/desktop/aa372637%28v=vs.85%29.aspx> ?

Are you aware that the result is actually a time, so you have to get the processor timer and the idle timer if you want to calculate the ratio ?

Did you see this :

<https://code.msdn.microsoft.com/windowsdesktop/Use-PerformanceCounter-to-272d57a1> ?

?onsai - November 27, 2014, 2:19 am

Hi,

This is my code:

Code #include "pdh.h"

```
static PDH_HQUERY cpuQuery;  
static PDH_HCOUNTER cpuTotal;
```

```
void init() {  
    PdhOpenQuery(NULL, NULL, &cpuQuery);  
    PdhAddCounter(cpuQuery, (LPCSTR)"\\Processor(_Total)\\% Processor Time", NULL,  
&cpuTotal);  
    PdhCollectQueryData(cpuQuery);  
}
```

```
long getCurrentValue() {  
    PDH_FMT_COUNTERVALUE counterVal;  
  
    PdhCollectQueryData(cpuQuery);  
    PdhGetFormattedCounterValue(cpuTotal, PDH_FMT_LONG, NULL, &counterVal);  
    return counterVal.longValue;  
}
```

Cheers ?

CrySon - November 27, 2014, 1:20 pm

(Nov 27, 2014 03:19 AM)S1L3NZ Wrote: [\[CITE\]](#) Hi,

This is my code:

Code #include "pdh.h"

```
static PDH_HQUERY cpuQuery;
static PDH_HCOUNTER cpuTotal;

void init() {
    PdhOpenQuery(NULL, NULL, &cpuQuery);
    PdhAddCounter(cpuQuery, (LPCSTR)"\\Processor(_Total)\\% Processor Time", NULL,
&cpuTotal);
    PdhCollectQueryData(cpuQuery);
}

long getCurrentValue() {
    PDH_FMT_COUNTERVALUE counterVal;

    PdhCollectQueryData(cpuQuery);
    PdhGetFormattedCounterValue(cpuTotal, PDH_FMT_LONG, NULL, &counterVal);
    return counterVal.longValue;
}
```

Cheers ?

thanks for this reply :) very usefull for some ppl

Mouse? - November 27, 2014, 4:17 pm

He's posting code that isn't done right. Stop spamming without reading the thread.

?onsai - November 30, 2014, 12:18 pm

Yea stop spamming CrySon.

Nobody knows how to do it?

cheers ?

D4T4B4S3 - November 30, 2014, 1:46 pm

Try this:

Code #include <Windows.h>

```
static float CalculateCPULoad(unsigned long long idleTicks, unsigned long long totalTicks)
{
    static unsigned long long _previousTotalTicks = 0;
    static unsigned long long _previousIdleTicks = 0;

    unsigned long long totalTicksSinceLastTime = totalTicks-_previousTotalTicks;
    unsigned long long idleTicksSinceLastTime = idleTicks-_previousIdleTicks;

    float ret = 1.0f-((totalTicksSinceLastTime > 0) ?
```

```
((float)idleTicksSinceLastTime)/totalTicksSinceLastTime : 0);
```

```
    _previousTotalTicks = totalTicks;  
    _previousIdleTicks = idleTicks;  
    return ret;  
}
```

```
static unsigned long long FileTimeToInt64(const FILETIME & ft) {return (((unsigned long long)(ft.dwHighDateTime))<<32)|((unsigned long long)ft.dwLowDateTime);}
```

```
float GetCPULoad()  
{  
    FILETIME idleTime, kernelTime, userTime;  
    return GetSystemTimes(&idleTime, &kernelTime, &userTime) ?  
    CalculateCPULoad(FileTimeToInt64(idleTime),  
    FileTimeToInt64(kernelTime)+FileTimeToInt64(userTime)) : -1.0f;  
}_____
```

I am not sure if thats is what you're looking for so here is the second choice (you can edit the source) ? _____

CpuUsage.h:
Code #pragma once
#include <windows.h>

```
class CpuUsage{  
public:  
    CpuUsage(void);  
    short GetUsage();  
private:  
    ULONGLONG SubtractTimes(const FILETIME& ftA, const FILETIME& ftB);  
    bool EnoughTimePassed();  
    inline bool IsFirstRun() const { return (m_dwLastRun == 0); }  
    FILETIME m_ftPrevSysKernel;  
    FILETIME m_ftPrevSysUser;  
    FILETIME m_ftPrevProcKernel;  
    FILETIME m_ftPrevProcUser;  
    short m_nCpuUsage;  
    ULONGLONG m_dwLastRun;  
    volatile LONG m_lRunCount;  
};
```

Now the main cpp file:
Code #include "StdAfx.h"
#include <windows.h>
#include "CpuUsage.h"

```
CpuUsage::CpuUsage(void)  
:m_nCpuUsage(-1)  
, m_dwLastRun(0)  
, m_lRunCount(0){  
  
    ZeroMemory(&m_ftPrevSysKernel, sizeof(FILETIME));  
    ZeroMemory(&m_ftPrevSysUser, sizeof(FILETIME));  
    ZeroMemory(&m_ftPrevProcKernel, sizeof(FILETIME));
```

```

ZeroMemory(&m_ftPrevProcUser, sizeof(FILETIME));
}

short CpuUsage::GetUsage(){
    short nCpuCopy = m_nCpuUsage;
    if (::InterlockedIncrement(&m_IRunCount) == 1){
        if (!EnoughTimePassed()){

            ::InterlockedDecrement(&m_IRunCount);
            return nCpuCopy;

        }

        FILETIME ftSysIdle, ftSysKernel, ftSysUser;
        FILETIME ftProcCreation, ftProcExit, ftProcKernel, ftProcUser;

        if (!GetSystemTimes(&ftSysIdle, &ftSysKernel, &ftSysUser) ||
            !GetProcessTimes(GetCurrentProcess(), &ftProcCreation,
                            &ftProcExit, &ftProcKernel, &ftProcUser)){

            ::InterlockedDecrement(&m_IRunCount);
            return nCpuCopy;
        }

        if (!IsFirstRun()){
            ULONGLONG ftSysKernelDiff =
                SubtractTimes(ftSysKernel, m_ftPrevSysKernel);
            ULONGLONG ftSysUserDiff = SubtractTimes(ftSysUser, m_ftPrevSysUser);
            ULONGLONG ftProcKernelDiff = SubtractTimes(ftProcKernel, m_ftPrevProcKernel);
            ULONGLONG ftProcUserDiff = SubtractTimes(ftProcUser, m_ftPrevProcUser);
            ULONGLONG nTotalSys = ftSysKernelDiff + ftSysUserDiff;
            ULONGLONG nTotalProc = ftProcKernelDiff + ftProcUserDiff;
            if (nTotalSys > 0){

                m_nCpuUsage = (short)((100.0 * nTotalProc) / nTotalSys);

            }

        }

        m_ftPrevSysKernel = ftSysKernel;
        m_ftPrevSysUser = ftSysUser;
        m_ftPrevProcKernel = ftProcKernel;
        m_ftPrevProcUser = ftProcUser;
        m_dwLastRun = GetTickCount64();
        nCpuCopy = m_nCpuUsage;

    }
    ::InterlockedDecrement(&m_IRunCount);
    return nCpuCopy;
}

ULONGLONG CpuUsage::SubtractTimes(const FILETIME& ftA, const FILETIME& ftB){

```

```

    LARGE_INTEGER a, b;
    a.LowPart = ftA.dwLowDateTime;
    a.HighPart = ftA.dwHighDateTime;
    b.LowPart = ftB.dwLowDateTime;
    b.HighPart = ftB.dwHighDateTime;
    return a.QuadPart - b.QuadPart;
}

bool CpuUsage::EnoughTimePassed(){
    const int minElapsedMS = 250;//milliseconds
    ULONGLONG dwCurrentTickCount = GetTickCount64();
    return (dwCurrentTickCount - m_dwLastRun) > minElapsedMS;
}

DWORD WINAPI EatItThreadProc(LPVOID lpParam);
DWORD WINAPI WatchItThreadProc(LPVOID lpParam);
CpuUsage usage;
int _tmain(int argc, _TCHAR* argv[]){

    CreateThread(NULL, 0, EatItThreadProc, NULL, 0, NULL);
    CreateThread(NULL, 0, EatItThreadProc, NULL, 0, NULL);
    CreateThread(NULL, 0, WatchItThreadProc, NULL, 0, NULL);
    CreateThread(NULL, 0, WatchItThreadProc, NULL, 0, NULL);
    while (true){

        Sleep(1000);

    }
    return 0;
}

DWORD WINAPI WatchItThreadProc(LPVOID lpParam){
    while (true){
        short cpuUsage = usage.GetUsage();
        printf("Thread id %d: %d%% cpu usage\n", ::GetCurrentThreadId(), cpuUsage);
        Sleep(1000);
    }
}

DWORD WINAPI EatItThreadProc(LPVOID lpParam){

    ULONGLONG accum = 0;
    while (true){

        accum++;
    }
    printf("%64d\n", accum);
}

```

Give these a shot and tell me what you think ?

0xDEAD10CC - November 30, 2014, 2:31 pm

Could be because you stole it from this page without really understanding what is being done here:
<http://stackoverflow.com/questions/63166/how-to-determine-cpu-and-memory-consumption-from-ins>

[ide-a-process](#)

And the reason you're getting the weird value is also most likely because you don't call `init()` before calling `getCurrentValue()`...

?onsai - December 1, 2014, 3:57 am

(Nov 30, 2014 02:46 PM)Førtune Wrote: [\[CITE\]](#) Try this:

Code `#include <Windows.h>`

```
static float CalculateCPULoad(unsigned long long idleTicks, unsigned long long totalTicks)
{
    static unsigned long long _previousTotalTicks = 0;
    static unsigned long long _previousIdleTicks = 0;

    unsigned long long totalTicksSinceLastTime = totalTicks - _previousTotalTicks;
    unsigned long long idleTicksSinceLastTime = idleTicks - _previousIdleTicks;

    float ret = 1.0f - ((totalTicksSinceLastTime > 0) ?
((float)idleTicksSinceLastTime)/totalTicksSinceLastTime : 0);

    _previousTotalTicks = totalTicks;
    _previousIdleTicks = idleTicks;
    return ret;
}
```

```
static unsigned long long FileTimeToInt64(const FILETIME & ft) {return (((unsigned long
long)(ft.dwHighDateTime))<<32)|((unsigned long long)ft.dwLowDateTime);}
```

```
float GetCPULoad()
{
    FILETIME idleTime, kernelTime, userTime;
    return GetSystemTimes(&idleTime, &kernelTime, &userTime) ?
CalculateCPULoad(FileTimeToInt64(idleTime),
FileTimeToInt64(kernelTime)+FileTimeToInt64(userTime)) : -1.0f;
}_____
```

I am not sure if thats is what you're looking for so here is the second choice (you can edit the source) ?_____

CpuUsage.h:

Code `#pragma once`
`#include <windows.h>`

```
class CpuUsage{
public:
    CpuUsage(void);
    short GetUsage();
private:
    ULONGLONG SubtractTimes(const FILETIME& ftA, const FILETIME& ftB);
    bool EnoughTimePassed();
    inline bool IsFirstRun() const { return (m_dwLastRun == 0); }
    FILETIME m_ftPrevSysKernel;
    FILETIME m_ftPrevSysUser;
```

```

    FILETIME m_ftPrevProcKernel;
    FILETIME m_ftPrevProcUser;
    short m_nCpuUsage;
    ULONGLONG m_dwLastRun;
    volatile LONG m_lRunCount;
};

```

Now the main cpp file:

```
Code #include "StdAfx.h"
```

```
#include <windows.h>
```

```
#include "CpuUsage.h"
```

```
CpuUsage::CpuUsage(void)
```

```
:m_nCpuUsage(-1)
```

```
, m_dwLastRun(0)
```

```
, m_lRunCount(0){
```

```
    ZeroMemory(&m_ftPrevSysKernel, sizeof(FILETIME));
```

```
    ZeroMemory(&m_ftPrevSysUser, sizeof(FILETIME));
```

```
    ZeroMemory(&m_ftPrevProcKernel, sizeof(FILETIME));
```

```
    ZeroMemory(&m_ftPrevProcUser, sizeof(FILETIME));
```

```
}
```

```
short CpuUsage::GetUsage(){
```

```
    short nCpuCopy = m_nCpuUsage;
```

```
    if (::InterlockedIncrement(&m_lRunCount) == 1){
```

```
        if (!EnoughTimePassed()){
```

```
            ::InterlockedDecrement(&m_lRunCount);
```

```
            return nCpuCopy;
```

```
        }
```

```
    FILETIME ftSysIdle, ftSysKernel, ftSysUser;
```

```
    FILETIME ftProcCreation, ftProcExit, ftProcKernel, ftProcUser;
```

```
    if (!GetSystemTimes(&ftSysIdle, &ftSysKernel, &ftSysUser) ||
```

```
        !GetProcessTimes(GetCurrentProcess(), &ftProcCreation,
```

```
        &ftProcExit, &ftProcKernel, &ftProcUser)){
```

```
        ::InterlockedDecrement(&m_lRunCount);
```

```
        return nCpuCopy;
```

```
    }
```

```
    if (!IsFirstRun()){
```

```
        ULONGLONG ftSysKernelDiff =
```

```
        SubtractTimes(ftSysKernel, m_ftPrevSysKernel);
```

```
        ULONGLONG ftSysUserDiff = SubtractTimes(ftSysUser, m_ftPrevSysUser);
```

```
        ULONGLONG ftProcKernelDiff = SubtractTimes(ftProcKernel, m_ftPrevProcKernel);
```

```
        ULONGLONG ftProcUserDiff = SubtractTimes(ftProcUser, m_ftPrevProcUser);
```

```
        ULONGLONG nTotalSys = ftSysKernelDiff + ftSysUserDiff;
```

```
        ULONGLONG nTotalProc = ftProcKernelDiff + ftProcUserDiff;
```

```
        if (nTotalSys > 0){
```

```

        m_nCpuUsage = (short)((100.0 * nTotalProc) / nTotalSys);

    }

}

m_ftPrevSysKernel = ftSysKernel;
m_ftPrevSysUser = ftSysUser;
m_ftPrevProcKernel = ftProcKernel;
m_ftPrevProcUser = ftProcUser;
m_dwLastRun = GetTickCount64();
nCpuCopy = m_nCpuUsage;

}
::InterlockedDecrement(&m_lRunCount);
return nCpuCopy;
}

ULONGLONG CpuUsage::SubtractTimes(const FILETIME& ftA, const FILETIME& ftB){

    LARGE_INTEGER a, b;
    a.LowPart = ftA.dwLowDateTime;
    a.HighPart = ftA.dwHighDateTime;
    b.LowPart = ftB.dwLowDateTime;
    b.HighPart = ftB.dwHighDateTime;
    return a.QuadPart - b.QuadPart;
}

bool CpuUsage::EnoughTimePassed(){
    const int minElapsedMS = 250;//milliseconds
    ULONGLONG dwCurrentTickCount = GetTickCount64();
    return (dwCurrentTickCount - m_dwLastRun) > minElapsedMS;
}

DWORD WINAPI EatItThreadProc(LPVOID lpParam);
DWORD WINAPI WatchItThreadProc(LPVOID lpParam);
CpuUsage usage;
int _tmain(int argc, _TCHAR* argv[]){

    CreateThread(NULL, 0, EatItThreadProc, NULL, 0, NULL);
    CreateThread(NULL, 0, EatItThreadProc, NULL, 0, NULL);
    CreateThread(NULL, 0, WatchItThreadProc, NULL, 0, NULL);
    CreateThread(NULL, 0, WatchItThreadProc, NULL, 0, NULL);
    while (true){

        Sleep(1000);

    }
    return 0;
}

DWORD WINAPI WatchItThreadProc(LPVOID lpParam){
    while (true){

```

```

    short cpuUsage = usage.GetUsage();
    printf("Thread id %d: %d%% cpu usage\n", ::GetCurrentThreadId(), cpuUsage);
    Sleep(1000);
}
}
DWORD WINAPI EatItThreadProc(LPVOID lpParam){

    ULONGLONG accum = 0;
    while (true){

        accum++;
    }
    printf("%64d\n", accum);
}

```

Give these a shot and tell me what you think ?
 Hey Fortune... i love you ?

First Source worked fine as fuck !

Cheers ?

Mouse? - December 1, 2014, 12:32 pm

And not a single thing was learned that day.

?onsai - December 1, 2014, 1:07 pm

(Dec 1, 2014 01:32 PM)Rodaxoleaux Wrote: [\[CITE\]](#) And not a single thing was learned that day.
 How do you know that? I read step by step the examples and implemented the first one in my project. Now i know how to calculate the CPU loadage.

Please stop with these senseless posts.

_Stoner - December 1, 2014, 1:10 pm

(Dec 1, 2014 01:32 PM)Rodaxoleaux Wrote: [\[CITE\]](#) And not a single thing was learned that day.
 You seem to be unaware of the fact that not everybody goes to school to learn, but rather to collect the tools they need in order to do what they want.

The same applies to this forum, and the rest of the entire internet.

Now you have learned something Rodaxoleaux.

EDIT: It's also worth mentioning that different people have different learning methods. Some learn by example and I would say most people learn coding by example.

?onsai - December 2, 2014, 8:50 am

(Dec 1, 2014 02:10 PM)_Stoner Wrote: [\[CITE\]](#) You seem to be unaware of the fact that not everybody goes to school to learn, but rather to collect the tools they need in order to do what they want.

The same applies to this forum, and the rest of the entire internet.

Now you have learned something Rodaxoleaux.

EDIT: It's also worth mentioning that different people have different learning methods. Some learn

by example and I would say most people learn coding by example.

Yep. True Story. Learning by example is for me the most efficient way to learn specific stuff. If i want to learn basics and stuff in a language i read books but in this case its much easier to get some examples to experiment with.

Cheers ?

0xDEAD10CC - December 2, 2014, 6:08 pm

Regardless of whether a person wants to learn or not... I still think that the statement above holds true. In other words, I doubt that OP learned anything here. That first code was copied from StackOverflow and the issue was obviously a result of not understanding what the code did or how to use it. I would say the same thing about whatever the code posted within this thread too (in regards to understanding -- supposedly he/she got by with the usage this time), I didn't really look through it.

If OP did, then he/she would be able to explain how the code worked.

Furthermore, S1L3NZ, I doubt you understand what _Stoner is saying here. He's really implying that you're not here to learn as far as I can tell... Yet you come back with some statement about "learning by example"... (?)

Mouse? - December 4, 2014, 10:56 am

All I "learned" was that people are lazy and take advantage of others to get what they want, and so I didn't learn anything, because I already knew that. I just don't like it. It's disgusting.