Sleep for milliseconds

I know the POSIX `sleep(x)` function makes the program sleep for x seconds. Is there a function to make the program sleep for x *milliseconds* in C++?

c++ linux sleep

edited Feb 4 '15 at 10:30

asked Nov 15 '10 at 12:49

Prasanth Madhavan

3,395 9 39 79

5 You should be aware that, in Windows anyway, `Sleep()` has millisecond *precision*, but it's *accuracy* can be orders of magnitude higher. You may think your sleeping for 4 milliseconds, but actually sleep for 400. – John Dibling Nov 15 '10 at 13:09

4 @John Dibling: I think he's using POSIX `sleep` , not win32 `Sleep` given "x seconds". – Charles Bailey Nov 15 '10 at 13:14

1 Although C and C++ have different name mangling, which can be a source of bugs and incompatibilities, in most cases it's fine to use C headers in C++. However, if you want to be absolutely sure that nothing goes wrong, `#include` the C header inside an `extern "C" {}` block. Also, if you have C and C++ source files in the same project, it's highly recommended that you do this in order to avoid any problems, especially if you include the same headers in both kinds of source files (in which case this is necessary). If you have a purely C++ project, it might just work with no problem at all. – adam10603 Mar 17 '15 at 12:56

15 Answers

Note that there is no standard C API for milliseconds, so (on Unix) you will have to settle for `usleep` , which accepts microseconds:

```
#include <unistd.h>

unsigned int microseconds;
...
usleep(microseconds);
```

edited May 2 '14 at 20:23

Tshepang

5,104 12 62 107

answered Nov 15 '10 at 12:52

Niet the Dark Absol

223k 40 279 409

2 It wouldn't hurt if the answer was in the form of a MWE =) – puk Nov 6 '13 at 0:17

2 Is it a busy sleep? I need to yield to another thread during sleep; can I use `usleep` for that? – Michael Oct 8 '15 at 18:30

4 It's not a busy wait stackoverflow.com/a/8156644/1206499, and `nanosleep` may be a better choice since `usleep` is obsolete. – jswetzen Dec 3 '15 at 14:55

1 Niet, please consider mentioning @HighCommander4's answer which is more portable if you have a C++11 compiler. – einpoklum Nov 14 '16 at 15:54

In C++11, you can do this with standard library facilities:

```
std::this_thread::sleep_for(std::chrono::milliseconds(x));
```

Clear and readable, no more need to guess at what units the sleep function takes.

like stated by Rhubbarb, you will need:

```
#include <chrono>
#include <thread>
```

edited Apr 6 '14 at 17:11
ooga
10.2k 2 12 18

answered May 16 '12 at 7:11
HighCommander4
19.1k 12 80 142

4 Does std::this_thread::sleep_for define an interruption point? Like boost::this_thread_sleep does? – Martin Meeser Jul 18 '13 at 9:28

25 This is the right way to do it. Period. Thread is cross platform as well as chrono. – Void Nov 19 '13 at 18:36

32 @Void. A very good way certainly, but "the" and "period" are awfully strong words. – Mad Physicist Jan 4 '15 at 7:12

1 @Valen: Is there another method that avoids such delays? – HighCommander4 Feb 17 '15 at 7:55

2 @Michael: It's not a busy sleep, it will yield to other threads. – HighCommander4 Oct 9 '15 at 2:32

To stay portable you could use Boost::Thread for sleeping:

```
#include <boost/thread/thread.hpp>

int main()
{
    //waits 2 seconds
    boost::this_thread::sleep( boost::posix_time::seconds(1) );
    boost::this_thread::sleep( boost::posix_time::milliseconds(1000) );

    return 0;
}
```

This answer is a duplicate and has been posted in this question before. Perhaps you could find some usable answers there too.

answered Nov 15 '10 at 12:52
MOnsDaR
4,646 5 31 57

6 Keep in mind that - in a multi-threaded environment - `boost::this_thread::sleep` adds an interruption point to your code. boost.org/doc/libs/1_49_0/doc/html/thread/… – Martin Meeser Jul 18 '13 at 9:25

Depending on your platform you may have `usleep` or `nanosleep` available. `usleep` is deprecated and has been deleted from the most recent POSIX standard; `nanosleep` is preferred.

answered Nov 15 '10 at 12:54
Charles Bailey
410k 60 497 570

Note that while `usleep()` is declared in `<unistd.h>` , confusingly, `nanosleep()` is declared in `<time.h>` / `<ctime>` . – gbmhunter May 7 '14 at 5:14

In Unix you can use usleep.

In Windows there is Sleep.

edited Aug 19 '14 at 20:41
Peter Mortensen
11.1k 15 76 109

answered Nov 15 '10 at 12:55
INS
6,963 3 39 78

2 and the Windows call is in milliseconds. – shindigo Sep 13 '13 at 14:06

6 You have to include <unistd.h> or <Windows.h> respectively. – gbmhunter May 7 '14 at 5:13

`nanosleep` is a better choice than `usleep` - it is more resilient against interrupts.

answered Nov 15 '10 at 12:54
Johan Kotlinski


16.4k 5 54 88

1 I was familiar with `usleep` , but not `nanosleep` . You should provide an example of using it on Linux. – jww Aug 14 '16 at 3:10

Why don't use time.h library? Runs on Windows and POSIX systems:

```
#include <iostream>
#include <time.h>

using namespace std;

void sleepcp(int milliseconds);

void sleepcp(int milliseconds) // cross-platform sleep function
{
    clock_t time_end;
    time_end = clock() + milliseconds * CLOCKS_PER_SEC/1000;
    while (clock() < time_end)
    {
    }
}
int main()
{
cout << "Hi! At the count to 3, I'll die! :)" << endl;
sleepcp(3000);
cout << "urrrrggghhhh!" << endl;
}
```

corrected code - now CPU stays in IDLE state [2014.05.24]:

```
#include <iostream>
#ifdef WIN32
#include <windows.h>
#else
#include <unistd.h>
#endif // win32

using namespace std;

void sleepcp(int milliseconds);

void sleepcp(int milliseconds) // cross-platform sleep function
{
    #ifdef WIN32
    Sleep(milliseconds);
    #else
    usleep(milliseconds * 1000);
    #endif // win32
}
int main()
{
cout << "Hi! At the count to 3, I'll die! :)" << endl;
sleepcp(3000);
cout << "urrrrggghhhh!" << endl;
}
```

edited May 23 '14 at 22:37

answered May 3 '14 at 23:38


Bart Grzybicki
137 1 5

9 One of the problems with that code is that it is a busy loop, it will continue using the 100% of a single processor core. The sleep function is implemented around an OS call that will put to sleep the current thread and do something else, and only will wake up the thread when the specified time expires. – Ismael May 20 '14 at 22:53

You're right - it will consume 100% of a one CPU core. So here is rewritten code using system sleep functions - and it's still cross-platform: – Bart Grzybicki May 23 '14 at 22:25

If using MS Visual C++ 10.0, you can do this with standard library facilities:

```
Concurrency::wait(milliseconds);
```

you will need:

```
#include <concrt.h>
```

edited Mar 8 '16 at 12:24


Tanuva
43 7

answered Jan 29 '15 at 18:59


Metronit
61 1 1

Syntax:

```
Sleep (  __in DWORD dwMilliseconds   );
```

Usage:

```
Sleep (1000); //Sleeps for 1000 ms or 1 sec
```

answered Apr 14 '14 at 13:34

foobar
1,360 1 13 33

2 What do you need to include for this? – ʇoleəz əɥʇ qoq Jun 25 '14 at 4:21

`#include <WinBase.h>` – foobar Jun 25 '14 at 5:40

4 No, you need to `#include <windows.h>` – ʇoleəz əɥʇ qoq Jun 26 '14 at 2:39

The way to sleep your program in C++ is the `Sleep(int)` method. The header file for it is `#include "windows.h"` For example:

```
#include "stdafx.h"
#include "windows.h"
#include "iostream"
using namespace std;

int main()
{
    int x = 6000;
    Sleep(x);
    cout << "it has been 6 seconds" << endl;
    return 0;
}
```

The time it sleeps is measured in milliseconds and has no limit.

```
Second = 1000 milliseconds
Minute = 60000 milliseconds
Hour = 3600000 milliseconds
```

edited Nov 2 '16 at 20:04

meJustAndrew
2,162 3 12 43

answered Jul 9 '14 at 18:39

genius
78 1 9

2 What do you mean it has no limit? It surely has limit which is 0xFFFFFFFE. Waiting for 0xFFFFFFFF will just not time out (which means it will wait till program ends). – Izzy Jan 16 '15 at 9:31

I didn't mean it like that Izzy, sorry for our misunderstanding. I meant that you can enter any positive number of milliseconds. So it will wait that many milliseconds to close the program. If you do not understand please say so, I shall explain to you more. – genius Jan 18 '15 at 16:54

Select call is a way of having more precision (sleep time can be specified in nanoseconds).

answered Nov 15 '10 at 13:17

Madhava Gaikwad
31 2

On platforms with `select` function (POSIX, Linux, Windows) you could do:

```
void sleep(unsigned long msec) {
    timeval delay = {msec / 1000, msec % 1000 * 1000};
    int rc = ::select(0, NULL, NULL, NULL, &delay);
    if(-1 == rc) {
        // Handle signals by continuing to sleep or return immediately.
    }
}
```

However, there are better alternatives available nowadays.

answered Mar 8 '16 at 12:38

Maxim Egorushkin
59.5k 4 67 119

Use boost asio threads, Sleep for x millisec;

```
#include <boost/thread.hpp>
#include <boost/asio.hpp>

boost::thread::sleep(boost::get_system_time()+boost::posix_time::millisec(1000));
```

answered Aug 11 '16 at 5:33

Anum Sheraz
160 11

reason for down-vote ? – Anum Sheraz Aug 11 '16 at 10:18

for gcc/g++ :

```
#include <unistd.h>
```

see http://linux.die.net/man/3/sleep

answered Sep 17 '16 at 4:35

Tanguy
802 7 8

As a Win32 replacement for POSIX systems:

```
void Sleep(unsigned int milliseconds) {
    usleep(milliseconds * 1000);
}

while (1) {
    printf(".");
    Sleep((unsigned int)(1000.0f/20.0f)); // 20 fps
}
```

answered Mar 17 at 16:02

lama12345
1,177 1 11 13

protected by Community ♦ Aug 11 '15 at 17:58

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?