How do you add a timed delay to a C++ program?

Asked 15 years, 3 months ago Modified 3 years, 1 month ago Viewed 688k times

138

I am trying to add a timed delay in a C++ program, and was wondering if anyone has any suggestions on what I can try or information I can look at?

I wish I had more details on how I am implementing this timed delay, but until I have more information on how to add a timed delay I am not sure on how I should even attempt to implement this.

c++ time

Share Edit Follow

edited Apr 21, 2009 at 17:34
17 of 26
27.2k 13 66 85

asked Oct 1, 2008 at 16:46
Matt Pascoe
8,701 17 43 48

2 It depends what platform (OS) and what libraries you have available. – Scott Langham Oct 1, 2008 at 16:50

14 Answers

Sorted by: Highest score (default)

191

An updated answer for C++11:

Use the `sleep_for` and `sleep_until` functions:

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

int main() {
    using namespace std::this_thread; // sleep_for, sleep_until
    using namespace std::chrono; // nanoseconds, system_clock, seconds

    sleep_for(nanoseconds(10));
    sleep_until(system_clock::now() + seconds(1));
}
```

With these functions there's no longer a need to continually add new functions for better resolution: `sleep`, `usleep`, `nanosleep`, etc. `sleep_for` and `sleep_until` are template functions that can accept values of any resolution via `chrono` types; hours, seconds, femtoseconds, etc.

In C++14 you can further simplify the code with the literal suffixes for `nanoseconds` and `seconds`:

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

int main() {
    using namespace std::this_thread;     // sleep_for, sleep_until
    using namespace std::chrono_literals; // ns, us, ms, s, h, etc.
    using std::chrono::system_clock;

    sleep_for(10ns);
    sleep_until(system_clock::now() + 1s);
}
```

Note that the actual duration of a sleep depends on the implementation: You can ask to sleep for 10 nanoseconds, but an implementation might end up sleeping for a millisecond instead, if that's the shortest it can do.

Share Edit Follow

edited Feb 18, 2017 at 9:54

answered Mar 17, 2012 at 5:57

bames53
86.6k 15 180 244

6 User defined literal suffixes. They dug... too deep. – Ciro Santilli OurBigBook.com May 5, 2019 at 8:24

1 @CiroSantilliOurBigBook.com what do you mean? Can you please explain, I'm new to C++ and I like funny comments – crjase Oct 23, 2022 at 10:46

1 @crjase that `10ns` thing is insane. To be more precise, that one is not "user defined" as it does not have a underscore: stackoverflow.com/questions/29017344/... – Ciro Santilli OurBigBook.com Oct 23, 2022 at 21:13

123

In Win32:

```
#include<windows.h>
Sleep(milliseconds);
```

In Unix:

```
#include<unistd.h>
unsigned int microsecond = 1000000;
usleep(3 * microsecond);//sleeps for 3 second
```

`sleep()` only takes a number of seconds which is often too long.

Share Edit Follow

edited Nov 24, 2020 at 15:04

Nav
20.1k 27 93 138

answered Oct 1, 2008 at 16:52

Richard Harrison
19.3k 4 41 67

5 There is also nanosleep() if usleep() doesn't give you enough resolution. – Kristopher Johnson Oct 1, 2008 at 16:57

12 Keep in mind that the argument to these functions are MINIMUM sleep times, and do NOT guarantee you'll come back right away if another process is hogging the CPU at that time. – billjamesdev Oct 1, 2008 at 17:10

3 @Richard Harrison: Which header I have to include to let sleep() work? – Overflowh May 24, 2013 at 15:39

13 @Overflowh on win32 Sleep() is in windows.h ; on unix sleep() in unistd.h – Richard Harrison May 24, 2013 at 17:00

2 @TommasoTheaCioni Ok, I assume you mean, "How can I get the program to restart *exactly* at the delay?", and the answer is... you can't. You don't control the operating system to that degree. If some other program is in the middle of some critical segment, your program just has to wait. – billjamesdev Jun 20, 2018 at 1:32

43

```
#include <unistd.h>
usleep(3000000);
```

This will also sleep for three seconds. You can refine the numbers a little more though.

Share Edit Follow

answered Oct 1, 2008 at 16:50
Samir Talwar
14.3k 3 43 65

Thank you! Simple and clean solution. Worked on Mac compiling with G++ VS Code. – Z41N Jul 3, 2020 at 22:31

36

Do you want something as simple like:

```
#include <unistd.h>
sleep(3);//sleeps for 3 second
```

Share Edit Follow

edited Nov 24, 2020 at 15:05
Nav
20.1k 27 93 138

answered Oct 1, 2008 at 16:47
J.J.
4,864 1 25 29

1 doesn't work on windows ... – PierreBdR Oct 1, 2008 at 16:58

It should. It is almost universal. What language/Compiler did you try it in? Did you import all of the necessary libraries? – J.J. Oct 1, 2008 at 17:12

8

Note that this does not guarantee that the amount of time the thread sleeps will be anywhere close to the sleep period, it only guarantees that the amount of time before the thread continues execution will be at least the desired amount. The actual delay will vary depending on circumstances (especially load on the machine in question) and may be orders of magnitude higher than the desired sleep time.

Also, you don't list why you need to sleep but you should generally avoid using delays as a method of synchronization.

Share Edit Follow

answered Oct 1, 2008 at 16:55
Wedge
19.6k 7 48 71

6

You can try this code snippet:

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

int main(){
    std::this_thread::sleep_for(std::chrono::nanoseconds(10));
    std::this_thread::sleep_until(std::chrono::system_clock::now() +
std::chrono::seconds(1));
}
```

Share Edit Follow

edited Sep 12, 2014 at 13:04
ekostadinov
6,892 3 29 48

answered Sep 12, 2014 at 12:25
Jayesh Rathod
61 1 1

4

You can also use select(2) if you want microsecond precision (this works on platform that don't have usleep(3))

The following code will wait for 1.5 second:

```
#include <sys/select.h>
#include <sys/time.h>
#include <unistd.h>`

int main() {
    struct timeval t;
    t.tv_sec = 1;
    t.tv_usec = 500000;
    select(0, NULL, NULL, NULL, &t);
}
```

`

Share Edit Follow

edited Oct 1, 2008 at 16:56

answered Oct 1, 2008 at 16:49
Mathias Brossard
3,706 2 27 30

4

I found that `"_sleep(milliseconds);"` (without the quotes) works well for Win32 if you include the `chrono` library

E.g:

```
#include <chrono>

using namespace std;

main
{
    cout << "text" << endl;
    _sleep(10000); // pauses for 10 seconds
}
```

Make sure you include the underscore before sleep.

Share Edit Follow

edited Sep 12, 2014 at 12:43
NetVipeC
4,412 1 17 19

answered Sep 12, 2014 at 12:07
ARoberts
49 1

This function has been superceded by newer library or operating system functionality. Consider using Sleep instead. – ByWaleed Apr 8, 2020 at 19:44

3

The top answer here seems to be an OS dependent answer; for a more portable solution you can write up a quick sleep function using the ctime header file (although this may be a poor implementation on my part).

```
#include <iostream>
#include <ctime>

using namespace std;

void sleep(float seconds){
    clock_t startClock = clock();
    float secondsAhead = seconds * CLOCKS_PER_SEC;
    // do nothing until the elapsed time has passed.
    while(clock() < startClock+secondsAhead);
    return;
}
int main(){

    cout << "Next string coming up in one second!" << endl;
    sleep(1.0);
    cout << "Hey, what did I miss?" << endl;
    return 0;
}
```

Share Edit Follow

edited Jan 18, 2020 at 21:12

answered Jan 15, 2020 at 23:11

Milan Donhowe
193 1 9

3

Yes, sleep is probably the function of choice here. Note that the time passed into the function is the smallest amount of time the calling thread will be inactive. So for example if you call sleep with 5 seconds, you're guaranteed your thread will be sleeping for at least 5 seconds. Could be 6, or 8 or 50, depending on what the OS is doing. (During optimal OS execution, this will be very close to 5.)
Another useful feature of the sleep function is to pass in 0. This will force a context switch from your thread.

Some additional information:
http://www.opengroup.org/onlinepubs/000095399/functions/sleep.html

Share Edit Follow

answered Oct 1, 2008 at 17:01

community wiki
Marcin

2

to delay output in cpp for fixed time, you can use the Sleep() function by including windows.h header file syntax for Sleep() function is Sleep(time_in_ms) as

```
cout<<"Apple\n";
Sleep(3000);
cout<<"Mango";
```

OUTPUT. above code will print Apple and wait for 3 seconds before printing Mango.

Share Edit Follow

answered Apr 18, 2017 at 17:09

Abhishek Rathore
1,076 1 11 14

0

Many others have provided good info for sleeping. I agree with Wedge that a sleep seldom the most appropriate solution.

If you are sleeping as you wait for something, then you are better off actually waiting for that thing/event. Look at Condition Variables for this.

I don't know what OS you are trying to do this on, but for threading and synchronisation you could look to the Boost Threading libraries (Boost Condition Varriable).

Moving now to the other extreme if you are trying to wait for exceptionally short periods then there are a couple of hack style options. If you are working on some sort of embedded platform where a 'sleep' is not implemented then you can try a simple loop (for/while etc) with an empty body (be careful the compiler does not optimise it away). Of course the wait time is dependant on the specific hardware in this case. For really short 'waits' you can try an assembly "nop". I highly doubt these are what you are after but without knowing why you need to wait it's hard to be more specific.

Share Edit Follow

answered Oct 2, 2008 at 4:43
Michael

0

On Windows you can include the windows library and use "Sleep(0);" to sleep the program. It takes a value that represents milliseconds.

Share Edit Follow

answered Oct 2, 2008 at 4:52
Zee JollyRoger
399 4 15

0

Syntax:

void sleep(unsigned seconds);

sleep() suspends execution for an interval (seconds). With a call to sleep, the current program is suspended from execution for the number of seconds specified by the argument seconds. The interval is accurate only to the nearest hundredth of a second or to the accuracy of the operating system clock, whichever is less accurate.

Share Edit Follow

answered Oct 1, 2008 at 16:48
GEOCHET
21.2k 15 76 98