Get Unix timestamp with C++

How do I get a `uint` unix timestamp in C++? I've googled a bit and it seems that most methods are looking for more convoluted ways to represent time. Can't I just get it as a `uint` ?

c++ unix timestamp uint

asked May 16 '11 at 2:24
2rs2ts
5,134 4 24 50

6 Answers

`time()` is the simplest function - seconds since Epoch. Linux manpage here.

The cppreference page linked above gives this example:

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

int main()
{
    std::time_t result = std::time(nullptr);
    std::cout << std::asctime(std::localtime(&result))
              << result << " seconds since the Epoch\n";
}
```

edited Sep 22 '15 at 16:32
Willi Mentzel
2,800 10 23 43

answered May 16 '11 at 2:26
Tony D
75.2k 8 98 161

2 "Epoch" is of course the Unix eproch on Unix and Linux, but that's not universal. – MSalters May 17 '11 at 8:09


```
if (dev.isBored() || job.sucks()) {
    searchJobs({flexibleHours: true, companyCulture: 100});
}
// A career site that's by developers, for developers.
```

stackoverflow JOBS

Get started


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

int main()
{
    std::time_t t = std::time(0);  // t is an integer type
    std::cout << t << " seconds since 01-Jan-1970\n";
    return 0;
}
```

answered May 16 '11 at 2:28
wilhelmtell
37.3k 13 78 119

This assumes that `std::time()` gives you seconds since 1970. That's true on a lot of systems (POSIX and Windows, I believe), but it's not guaranteed by the language standard. – Keith Thompson Jun 24 '16 at 2:44

The most common advice is wrong, you can't just rely on `time()`. That's used for relative timing: ISO C++ doesn't specify that `1970-01-01T00:00Z` is `time_t(0)`

What's worse is that you can't easily figure it out, either. Sure, you can find the calendar date of `time_t(0)` with `gmtime`, but what are you going to do if that's `2000-01-01T00:00Z`? How many seconds were there between `1970-01-01T00:00Z` and `2000-01-01T00:00Z`? It's certainly no multiple of 60, due to leap seconds.

answered May 17 '11 at 8:15
MSalters
115k 8 91 238

1 "How do I get a uint unix timestamp in C++?" - given - as you've said - you can call `gmtime()` later to get a readable representation of whatever that timestamp encodes, what functionality requested in the question isn't satisfied by `time()`, regardless of the reference date or suitability for interval calculations? – Tony D Aug 23 '12 at 6:58

3 To get a **UNIX** timestamp on a non-UNIX system, you have to know the difference (in seconds) between the local epoch and `1970-01-01T00:00Z`. There's just no method which does that. – MSalters Aug 23 '12 at 7:48

For some reason I got the impression the question was for code on a UNIX (or Linux etc) machine, but now I see where you're coming from. Curious: have you found any actual system where time_t(0) wasn't 1970-01-01T00:00Z? Would be but a couple minutes work to work out an offset on any given system (take the non-UNIX time_t(0) and get a time_t for it on a UNIX system), but thanks for explaining your concern. – Tony D Aug 23 '12 at 23:35

4 For VS its UNIX time; MSDN states: The time function returns the number of seconds elapsed since midnight (00:00:00), January 1, 1970, Coordinated Universal Time (UTC), according to the system clock. msdn.microsoft.com/en-us/library/1f4c8f33.aspx – Oliver Zendel Mar 8 '14 at 8:15

2 @TonyD: IBM systems apparently are an exception. Not really a surprise since they were in the computer business well before 1-1-1970. – MSalters May 8 '15 at 12:02

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

using namespace std;

int main ()
{
  unsigned long int sec= time(NULL);
  cout<<sec<<endl;
}
```

answered May 16 '11 at 2:29
anijhaw
3,884 2 22 34

`time()` returns a result of type `time_t`, which can in principle be either signed, unsigned, or even floating-point. Why store the result in a `long int`? And why use `<sys/time.h>` rather than the standard `<time.h>`? – Keith Thompson Jun 24 '16 at 2:43

I understand but I think in any posix compliant OS both time.h & sys/time.h are present. Also you are completely correct that time_t can be negative value too as it is represented as seconds elapsed from 1970-01-01T00:00Z. – anijhaw Jun 27 '16 at 20:20

Windows uses a different epoch and time units: see Convert Windows Filetime to second in Unix/Linux

What std::time() returns on Windows is (as yet) unknown to me (;-))

answered Nov 6 '13 at 17:31
davecb
69 3

I created a global define with more information:

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

#define INFO std::cout << std::put_time(std::localtime(&time_now), "%y-%m-%d
%OH:%OM:%OS") << " [INFO] " << __FILE__ << "(" << __FUNCTION__ << ":" << __LINE__
<< ") >> "
#define ERROR std::cout << std::put_time(std::localtime(&time_now), "%y-%m-%d
%OH:%OM:%OS") << " [ERROR] " << __FILE__ << "(" << __FUNCTION__ << ":" << __LINE__
<< ") >> "
```

```
static std::time_t time_now = std::time(nullptr);
```

Use it like this:

```
INFO << "Hello world" << std::endl;
ERROR << "Goodbye world" << std::endl;
```

Sample output:

```
16-06-23 21:33:19 [INFO] src/main.cpp(main:6) >> Hello world
16-06-23 21:33:19 [ERROR] src/main.cpp(main:7) >> Goodbye world
```

Put these lines in your header file. I find this very useful for debugging, etc.

answered Jun 24 '16 at 2:37

xinthose

411 1 3 17