Converting between Windows FILETIME and Unix time_t without having to type the magic number 116444736000000000

devblogs.microsoft.com/oldnewthing/20220602-00

June 2, 2022

Raymond Chen

Windows tracks time in 100ns units since January 1, 1601. Unix tracks time in 1s units since January 1, 1970. Is there an easy way to convert between them?

The Windows documentation offers a helper function to perform the conversion from `time_t` to `FILETIME`: It converts the units from seconds to 100ns by multiplying against the magic number 10000000, and then adds the second magic number 116444736000000000.

Is there a way to do the conversion without having to hard-code these magic numbers? Maybe somebody else has written a conversion that we can use?

Well, here's one place: C++/WinRT.

The `winrt::clock` class represents the Windows Runtime `DateTime` clock, and also provides a number of helpers to convert to and from other formats. The Windows Runtime `DateTime` has the same internal format as a `FILETIME`, so you can treat them as basically the same thing, just in different wrapping. And since C++/WinRT represents the Windows Runtime `DateTime` as a C++ `std::chrono::time_point` object, you have all of the C++ standard library facilities available.

```
// from Unix time to FILETIME
auto datetime = winrt::clock::from_time_t(unix_time_seconds);
FILETIME filetime = winrt::clock::to_file_time(datetime);

// or combined into one line
FILETIME filetime = winrt::clock::to_file_time(
               winrt::clock::from_time_t(unix_time_seconds));
```

And you can just run everything in reverse to go the other way.

```
// from FILETIME to Unix time
auto datetime = winrt::clock::from_file_time(filetime);
time_t unix_time_seconds = winrt::clock::to_time_t(datetime);

// or combined into one line
time_t unix_time_seconds = winrt::clock::to_time_t(
                     winrt::clock::from_file_time(filetime));
```

Of course, once you reach the `std::chrono::time_point`, you can stop and enjoy the scenery before moving onward to your final destination.

```
auto datetime = winrt::clock::from_file_time(filetime);

// move forward 3 minutes
datetime += 3min;

time_t unix_time_seconds = winrt::clock::to_time_t(datetime);
```

Unix time is represented in the C++ standard library as a `std::chrono::system_clock`, so you can convert your Unix timestamps into a `sys_time<Duration>` (or use one of the pre-made types like `sys_seconds`), and then do your work in the world of C++ `std::chrono` before converting at the last moment to a Windows `FILETIME`.

`time_t`	
⇵	

`sys_time` = `time_point<system_clock>`
⇵
`DateTime` = `time_point<winrt::clock>`
	`std::chrono::time_point`

⇵	
`FILETIME`	

To get in and out of the box through the top:

```
// time_t to sys_seconds
auto n_seconds = std::chrono::sys_seconds(std::chrono::seconds(N));

// sys_seconds to time_t
auto unix_ticks = seconds.time_since_epoch().count();
```

To convert between `sys_time` and `winrt::clock`:

```
auto winrt = winrt::clock::from_sys(sys);
auto sys = winrt::clock::to_sys(winrt);
```

And to get in and out through the bottom, use the `to_file_time` and `from_file_time` methods, as noted earlier.

Raymond Chen

Follow