Subject: Time::Set(int64 scalar) unexpected results
Posted by Alboni on Tue, 10 Sep 2013 18:14:11 GMT
View Forum Message <> Reply to Message

This is about

void Date::Set(int scalar)
Assign a date that is stored in the numeric scalar.

and

void Time::Set(int64 scalar)
Assign a time that is stored in the numeric scalar.

When I assign a unix time with Time::Set, the year is 43 instead of 2013. The origin of that problem seems to be in Date::Set(int scalar)

Is this a bug or is the Upp timesystem not starting in 1970 but in the year 0? scalar values obtained from Time::Get() do work.

I fixed it in my program like this (for now) :
int64 unixtime = ...
Time timestamp;
time.Set(unixtime);
time.year+=1970;

What would be the correct way to import a unix time?

Subject: Re: Time::Set(int64 scalar) unexpected results
Posted by Alboni on Tue, 10 Sep 2013 18:37:11 GMT
View Forum Message <> Reply to Message

Little test: See http://www.unixtimestamp.com/index.php for times

```
#include <Core/Core.h>
#include <time.h>

using namespace Upp;

CONSOLE_APP_MAIN
{
 time_t scalar = time(0); // seconds since 1/1/1970 00:00:00
 Time back;
 back.Set(scalar);

 Cout() << "The time is: " << asctime(localtime(&scalar)) << "\n"; // oldskool functions
```

```
 Cout() << "Scalar = " << scalar << "\n";
 Cout() << "Converted back = " << back << "\n";
}
```

Subject: Re: Time::Set(int64 scalar) unexpected results
Posted by dolik.rce on Tue, 10 Sep 2013 18:52:10 GMT
View Forum Message <> Reply to Message

Hi Alboni,

That is not a bug, it is just how the Time in U++ is designed. Unix timestamp is 32bit number, describing times from 1.1.1970 to someday in 2038. U++ Time works for dates from 4000 B.C. to 4000 A.D. if I remember correctly.

Maybe there is an explicit method to convert the timestamp to Time that I don't know about... But AFAIK the simplest way (at least on POSIX) is to use FileTime, which is just a wrapper around time_t.

So you should be able to doTime t = FileTime(timestamp);

Another possible solution, that should IMHO work correctly even on windows, could be something likeconst Time epoch(1970,1,1);
Time t;
t.Set(epoch.Get()+timestamp);

Best regards,
Honza

PS: You're right about U++ intentionally using year 0 as a center point for time. As it uses signed type, it allows for both negative and positive values so it can accurately represent about 8000 years timespan.

Subject: Re: Time::Set(int64 scalar) unexpected results
Posted by Alboni on Tue, 10 Sep 2013 18:54:42 GMT
View Forum Message <> Reply to Message

time_t can be both 32 or 64 bit

and the rest work ok. just not the year.

Subject: Re: Time::Set(int64 scalar) unexpected results
Posted by Alboni on Tue, 10 Sep 2013 18:57:29 GMT

View Forum Message <> Reply to Message

Thanks Honza.

Subject: Re: Time::Set(int64 scalar) unexpected results
Posted by dolik.rce on Tue, 10 Sep 2013 19:04:28 GMT
View Forum Message <> Reply to Message

Alboni wrote on Tue, 10 September 2013 20:54time_t can be both 32 or 64 bitYes, it is implementation defined. It doesn't even have to start at 1970 :http://en.cppreference.com/w/c/chrono/time_tArithmetic type capable of representing times. Although not defined, this is almost always a integral value holding the number of seconds (not counting leap seconds) since 00:00, Jan 1 1970 UTC, corresponding to POSIX time.So it would be safer not to rely on it too much, especially if you want the app to be platform independent.

Alboni wrote on Tue, 10 September 2013 20:54and the rest work ok. just not the year.All the years are quite similar, unless it is a leap year. I'd say that just changing the year after using Time::Set() with timestamp might mess up something due to the complex rules of leap days (see Date::Set(int64) to get an idea how complex this stuff is).

Honza

Subject: Re: Time::Set(int64 scalar) unexpected results
Posted by Alboni on Tue, 10 Sep 2013 23:25:34 GMT
View Forum Message <> Reply to Message

Yeah, I changed the fix into your example.

I know it's tricky stuff, but doable with some logic thinking. I had the pleasure of spending the whole summer of 1998 changing time related routines in all kinds of alarm related software in olskool C and C++ on Sco Unix 5 and Windows 95 for the company I worked for back then. The boss didn't want to wait till the next year to fix all the millennium bugs stuff. The software packages used something like 6 formats to store time, some of wich relied on bugs to work properly. I ended up writing a new library for it. (and fix hundreds of pointer errors, in the process just because I happened to see them when fixing the time stuff)