

Re: function for clockticks since 1980?

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2007-07/msg03626.html

- *From:* "Peter J. Holzer" <hjp-usenet2@xxxxxx>
 - *Date:* Sun, 29 Jul 2007 13:14:45 +0200
-

On 2007-07-29 01:16, nsa.usa@xxxxxxxxxx <nsa.usa@xxxxxxxxxx> wrote:

On Jul 29, 2:58 am, Ben Bacarisse <ben.use...@xxxxxxxxxx> wrote:

```
#include <time.h>
time_t time(time_t *timer);
```

The time function determines the current calendar time. The encoding of the value is unspecified.

so you can't rely on much variability there (it *may* be fine, but the solution would not be portable).

Yeah, this one seems to be down to the second only, and so when the app is run 100 times (or even just 2 times) within the same second, then I got a prob =:-)

Since there is not ideal portable solution, your best bet may well be to accept that go fully platform dependent. A small 'get_random_seed' function will be a couple of lines long and can be written easily for (reasonable) platform on which a web application might run. Post in a newsgroup for your which every platform you are currently using for ideas about getting a good random seed.

I might try the /dev/random for linux platform as was suggested by Xu earlier.

/dev/random is guaranteed to return random numbers – if there is not enough entropy in the pool it will block until more entropy is collected. On a busy web server you will probably rapidly running out of entropy, and you probably don't need that level of randomness, so you should use /dev/urandom instead.

Re: function for clockticks since 1980?

Just worried about the overhead on that one though. App is only planned to run on linux on i32 or platform, so I only need it to work on this (for now..).

Linux and other POSIX-compatible systems have the `gettimeofday` function, which returns the time in seconds and microseconds since the epoch. Be warned that the actual resolution of the clock can be much coarser: Linux/x86 actually uses a microsecond clock, but some other Unixes have resolutions of something between 1/60 to 1/1024 of a second.

Im pretty sure there is an interrupt to give this amount of milliseconds since 1980 on that cpu, but then again I'd hate to stick in cpu specific stuff..

These things are rarely CPU-specific. They are, however, OS-specific and in some cases system specific. (BTW, the epoch on POSIX systems is 1970, not 1980)

Somehow I thought this function would be standard in C (I'm not so used to C)... oh well :-)

The C standard specifies a baseline which all implementations must conform to. It is therefore very conservative and doesn't mandate functions which might be impossible to implement on some systems (e.g., not every system has a clock with millisecond resolution). You will notice that the standard doesn't make any guarantees about the resolution of `time_t`: It may have a resolution of nanoseconds, or only of minutes.

hp

--

_ | Peter J. Holzer | I know I'd be respectful of a pirate
|_| | Sysadmin WSR | with an emu on his shoulder.
|| | hjp@xxxxxx |
_/ | <http://www.hjp.at/> | -- Sam in "Freefall"

.