I’ve spoken about Unix epoch before, so this is a follow up post with the next few things about epoch time that I find particularly interesting.

What is Unix Epoch time?

Unix Epoch time is the number of seconds elapsed since 00:00:00 UTC, Thursday, 1 January 1970.

It’s an interesting concept, because the point in time is specific time we can easily recognise and process – it’s a specific time on a certain day way back in 1970. But the Epoch time is not a point in time, but a large counter – specifically, a number of seconds elapsed since the beginning on Unix time.
Does Epoch Time differ with timezones?

This is one of the most interesting things about Unix time. Even though it’s based on a specific timezone (UTC), the Unix time itself is not timezone specific. So one server referring to a specific Unix time will be referring to exactly the same point in time as another server in a different timezone – for both of them the number of seconds elapsed since UTC will be the same.

It is a great question though, because Unix Epoch time is commonly converted back to human-friendly time and date stamps, which at that point become timezone specific. So if you decode Epoch time value on two servers in different timezones, you will end up with a different human-friendly timestamp.

Will there be an end of Epoch Time?

Apparently, current representation of Epoch time means some 32-bit implementations will have a problem similar to Y2K, only a bit later: sometime on January 19th, 2038. On that day, the time counter will reach its maximum value for a 32-bit signed number, meaning next valid value will have to be 0 – meaning both the end and the start of the Unix Epoch time.

This 2038 problem sounds like such a fascinating issue that I’ll have a separate post on it soon!
See Aso

- Unix Epoch
- Converting Epoch time in Perl
- Unix scripting: date and time
- atime, ctime and mtime