unique.timestamp version 0.9.4

Background

A unique.timestamp provides a universally unique identifier which is also a human-readable timestamp. It represents **UTC** time, and it returns the time at the moment it was invoked.

```
The format of a unique.timestamp is
```

```
YYYYmmdd_HHMM_SS_xxxxxx.zzzzz.cccc
```

Where:

```
YYYY is the year
mm is the month
dd is the day of the month
HH is the hour
MM is the minute
SS is the second
xxxxxx is the milliseconds
zzzzz is a process pid
cccc is a count for that pid
```

For example,

20090612_0608_56_510702.002621.0000

Provided a host machine's system clock continues to move forward only, this timestamp is guaranteed to be unique for that host.

If you have a collection of different hosts all generating unique.timestamps, then there is a statistical probability that some will end up being the same. This probability is dependent on the number of hosts, and the frequency with which the timestamps are being generatied. It is possible to extend unique.timestamp in a straight forward fashion so that this probability is as close to zero as is required.

Executables in the unique.timestamp package

unique.timestamp

syntax:

unique.timestamp

returns a unique.timestamp as a string to stdout.

ts2secs

syntax:

ts2secs unique.timestamp

Converts the given unique timestamp to the number of seconds since epoch.

tsdiff

syntax:

tsdiff unique.timestamp1 unique.timestamp2

Returns the number of seconds between the two unique timestamps

tsOffset

syntax:

tsOffset *seconds*

Where *seconds* is an integer, which can be negative.

Produces something that resembles a unique.timestamp, but is offset from the current time by *seconds* seconds.

offset.timestamp

syntax:

offset.timestamp seconds [timestamp]

Where *seconds* is an integer, which can be negative, and *timestamp* is a string that resembles a unique.timestamp. *timestamp* defaults to the current timestamp.

Produces something that resembles a unique.timestamp, but is offset from the given timestamp by seconds seconds.

mask.timestamp

syntax:

mask.timestamp mask [timestamp]

Where *mask* is an arbitrary string, but would normally look like the beginning of a unique.timestamp string.

timestamp is also an arbitrary string, which would also normally look like a unique.timestamp. If not given, it defaults to the current timestamp.

The result consists of the *timestamp*, with the *mask* replacing the left-most characters. An occurence of the character . in *mask* will not have an effect on the character in that position.

For example:

```
mask.timestamp '....0601' 20100304_0421_01_736278.008979.0000
```

Returns:

20100601_0421_01_736278.008979.0000

Example 2:

Produce a timestamp representing the year 2012, with the month, day of the month and time (UTC) being the same as now:

mask.timestamp 2012

Produce a timestamp representing the first of June, for this current year, and with the time (UTC) being the same as now:

```
mask.timestamp '....0601'
```

future.timestamp

A shell script, which produces something that resembles a unique.timestamp, but represents a time other than the present. It could be in the future, or the past, according to the arguments given.

syntax:

```
future.timestamp list-of-time-specification
```

Where *list-of-time-specifications* is one or more time-specifications consisting of a single number (integer or decimal, with optional sign), followed by a single character, where:

```
s signifies seconds
m signifies minutes
h signifies hours
d signifies days
w signifies weeks
```

The result is a timestamp, similar in format to a unique.timestamp, but offset from the current time by the time interval represented by the sum of the individual time-specifications.

For example:

future.timestamp 1w -2d 0.5m

will produce a timestamp for a date one week, less two days, plus half a minute from the current time. The time is represented as **UTC**.

makeTimeInterval.pl

A Perl script which converts a human-readable time specification to a number of seconds.

syntax:

```
makeTimeInterval.pl time-specification
```

Where *time-specifications* is a single number (integer or decimal, with optional sign), followed by a single character, where:

```
s signifies seconds
```

```
m signifies minutes
```

- h signifies hours
- d signifies days
- w signifies weeks

The result is the equivalent number of seconds.

For example:

makeTimeInterval.pl 1w

The result is 604800, being 604,800 seconds.

Notes on useage

Of all these executables, only unique.timestamp is designed for uniqueness of the result. Use of future.timestamp, mask.timestamp and offset.timestamp will generate strings that resemble unique.timestamp in format, but have a reduced probability of universal uniqueness. Nevertheless, there are applications where it is useful to generate such strings.

mask.timestamp and future.timestamp/offset.timestamp complement each other. mask.timestamp can be used to produce timestamps representing specific dates and times, according to the calendar, future.timestamp and offset.timestamp can be used to produce timestamps offset from a particular time.

Because mask.timestamp and offset.timestamp both accept a timestamp as an optional second argument, you can combine their useage to satisfy more complex requirements. (For example, to produce a future timestamp for July 1, but one hour earlier than the current time).

C functions

The following C functions are available to use in your own code. They are made available both as source code, and in the form of a library.

The header file is unique.timestamp.h.

uniquetimestamp

```
int uniquetimestamp(char *ts)
```

Generates a unique timestamp as a string, and stores it at the specified address ts

The returned value is the result of a call to the standard library function gettimeofday.

uniquetimestamp_pidcount

int uniquetimestamp_pidcount(char *ts, int pid, int *count)

Produces a unique.timestamp, and stores it in *ts*. The value for pid is specified explicitly in *pid*, and the count is taken from a value pointed to by *count*.

Use this function when you want to specify the pid explicitly, and maintain the count yourself.

The returned value is the result of a call to the standard library function gettimeofday.

uniquetimestamp2time

time_t uniquetimestamp2time(char *ts)

Scans a unique.timestamp, given as a string, and converts it to a time_t value. the *microseconds*, *pid* and *count* parts are ignored.

uniquetimestamp_offset

int uniquetimestamp_offset(char *ts, int secs)

Produces a string resembling a unique.timestamp, representing a time offset *secs* seconds from the current time.

Stores the string in *ts*

The returned value is the result of a call to the standard library function gettimeofday.