Timewarrior – Where did my time go?

Introduction

Dirk Deimeke

Taskwarrior academy

FrOSCon 2016

Content

Introduction

Installation

Introduction

- Setup
- Stopwatch
- Tags
- Help
- Historical
- Hints
- Charts and Reports
- Configuration

Corrections

Integration in Takswarrior

Dirk Deimeke (that's me)

- Born 1968 in Wanne-Eickel
- Linux since 1996
- Emigrated 2008 to Switzerland
- Taskwarrior Team since 2010

Entry point for more `https://d5e.org/`

Today is the day!

Timewarrior

Prerequisites

All you need to compile is

- `libuuid`
- `CMake` (2.8 or newer)
- `make`
- A C++ Compiler (`GCC 4.7` or `Clang 3.3` or newer)

Get the source

Installation from source

Setup

As a new Timewarrior user, there is no configuration needed. Once you have installed the software, the first time you run it, a data/configuration directory is created for you if necessary, and if you allow this action.

```
$ timew
Create new database in /Users/sample/.timewarrior? (yes/no) yes
There is no active time tracking.
$
```

You are told that there is no active time tracking, and that’s right, because we’ve done nothing yet.

Data directory

```
$ ls -1 .timewarrior
data
extensions
timewarrior.cfg
$
```

By default a `.timewarrior` directory is created in your home directory, and contains a sub-directory for data, and another for extensions. There is also an empty configuration file created.

Just like Taskwarrior, an empty configuration file means we are using all the default settings. If you want to use an alternate location, you can either symlink it to `~/.timewarrior`, or use the `TIMEWARRIORDB` environment variable to specify any location.

Stopwatch

The most basic tracking can be done by using the stopwatch features. Just like a stopwatch, you start and stop a clock, and you can see the elapsed time. Start the clock when you begin work work, and stop it when you are done. First let's see if the clock is running:

```
$ timew
There is no active time tracking.
$
```

No it's not. You can always run timewarrior with no arguments to see if the clock is running, and no data is modified. Let's start the clock.

```
$ timew start
Tracking
  Started 2016-08-07T20:19:42
  Current                 42
  Total             0:00:00
$
```

Stopwatch (2)

The clock is now running. In reality there is no clock and all that happened was that the start time was recorded, so don't be concerned about using system resources, as none are being used. Now when we check, we see a summary of the time recorded so far:

```
$ timew
Tracking
  Started 2016-08-07T20:19:42
  Current               20:34
  Total              0:00:52
$
```

Once our work is complete, we stop the clock, and look at the summary.

```
$ timew stop
Recorded
  Started 2016-08-07T20:19:42
  Ended                 21:12
  Total              0:01:30
$
$ timew summary

Wk  Date       Day Tags  Start    End      Time    Total
W31 2016-08-07 Sun       20:19:42 20:21:12 0:01:30 0:01:30
                                                   -------
                                                   0:01:30

$
```

Stopwatch (3)

The summary report shows time that was tracked today by default.

There is a `continue` command that re-starts the previous tracking, and is useful if you stopped the clock during lunch, or overnight.

```
$ timew continue
Tracking
  Started 2016-08-07T20:22:06
  Current                 06
  Total              0:01:30
$
$ timew stop
Recorded
  Started 2016-08-07T20:22:06
  Ended                   18
  Total              0:01:42
$
$ timew summary

Wk  Date       Day Tags   Start    End      Time   Total
W31 2016-08-07 Sun      20:19:42 20:21:12 0:01:30
                        20:22:06 20:22:18 0:00:12 0:01:42
                                                  -------
                                                  0:01:42

$
```

See how every time the clock is started, there is a new line in the `summary` report?

Stopwatch (4)

It is important to note that it is possible, and quite likely,that you will start the clock and leave it running by mistake. This is in fact one of the most annoying problems with any time tracking solution that relies on human interaction. While some tools will detect inactivity and stop the clock for you, Timewarrior does not. Instead it deals with this problem in two ways, first by making it easy to correct tracked time, and second by way of exclusions, which we will cover later.

Tags

The examples so far did not use tags. Tags are optional, but when you do make use of tags, you start tracking time spent on *different* activities.

```
$ timew start 'Using Tags' Software
Tracking Software "Using Tags"
  Started 2016-08-07T20:23:08
  Current                 08
  Total             0:00:00
$
```

This example is tracking time using two different tags. The first is *Using Tags*, the second is *Software*. The first tag is two words, and because of the space between them, the quotes are needed to keep those two words together in one tag. The second tag is a single word and needs no quotes.

Tags (2)

```
$ timew stop
Recorded Software "Using Tags"
  Started 2016-08-07T20:23:08
  Ended                   48
  Total             0:00:40
$
$ timew summary

Wk  Date       Day Tags                 Start    End      Time    Total
W31 2016-08-07 Sun                   20:19:42 20:21:12 0:01:30
                                     20:22:06 20:22:18 0:00:12
                   Software, Using Tags 20:23:08 20:23:48 0:00:40 0:02:22
                                                                  -------
                                                                  0:02:22

$
```

You can see that using tags is useful, but optional. Once you are using tags, you can use them to filter reports, such as the `summary` report.

```
$ timew summary Software

Wk  Date       Day Tags                 Start    End      Time    Total
W31 2016-08-07 Sun Software, Using Tags 20:23:08 20:23:48 0:00:40 0:00:40
                                                                  -------
                                                                  0:00:40

$
```

Tags (3)

There is a `tags` command, which will show you all the tags you have used.

```
$ timew tags

Tag         Description
Software    -
Using Tags  -

$
```

You may wonder how is that *Description* column in the report used? It is the first example of tag metadata, which you can configure.

```
$ timew config tags.Software.description 'Learning about new software'
Are you sure you want to add 'tags.Software.description' with a value of 'Learning about new software'? (yes/no) yes
Config file /Users/sample/.timewarrior/timewarrior.cfg modified.
$
```

This is not currently used, but does represent how Timewarrior will be extended to include tag metadata in future releases.

Help command

Although we have just begun, it is important to mention the built-in help system. Let's look at the help for the `continue` command.

```
$ timew help continue

Syntax: timew continue

Resumes tracking the most recently closed interval. For exqample:

  $ timew track 9am - 10am tag1 tag2
  $ timew continue

The 'continue' command creates a new interval, starting now, and using the tags
'tag1' and 'tag2'.

This command is a convenient way to resume work without re-entering the tags.

See also 'start', 'stop'.

$
```

You can see that the help system contains examples, and introduces new commands and other help topics. From the help text, we learn that the `continue` command will not only resume tracking but use the same set of tags.

Man page

In addition to the built-in help, there is a man page which contains the same information plus a lot more.

```
$ man timew
timew(1)                                    User Manuals                                    timew(1)

NAME
       timew - A command line time tracker.

SYNOPSIS
       timew <command> [<arg> ...]
```

Historical

We have seen how to use the stopwatch feature, and combine it with tags. Additionally we can record time *ex post facto*. So to track time I spent earlier in the day (but forgot to record), I use the `track` command.

```
$ timew track 9:00 - 11:00 'Outline the tutorial topics'
Recorded "Outline the tutorial topics"
  Started 2016-08-07T09:00:00
  Ended              11:00:00
  Total               2:00:00
$
```

When Timewarrior sees a time like `9:00` it always assumes it is in the past, because Timewarrior is a tool for recording what you have done or are doing, and is not a forward-looking planning tool.

This is the opposite of Taskwarrior, which always looks forward, because tasks are generally going to be completed in the future.

Historical(2)

There are other ways to specify time in the past, for example:

```
$ timew track 9am for    2h   'Outline the tutorial topics'
$ timew track 9am to     11am 'Outline the tutorial topics'
$ timew track 2h  before 11am 'Outline the tutorial topics'
$
```

Historical(3)

And there are many more, which can be seen in the help system.

```
$ timew help interval

An interval defines a block of time that is tracked. The syntax for specifying
an interval is flexible, and may be one of:

  [from] <date>
  [from] <date> to/- <date>
  [from] <date> for <duration>
  <duration> before/after <date>
  <duration> ago
  [for] <duration>

Examples are:

  from 9:00
  from 9am - 11am
  from 9:00:00 to 11:00
  from 9:00 for 2h
  2h after 9am
  2h before 11:00
  2h ago
  for 2h

An interval is said to be 'closed' if there is both a start and end, and 'open'
if there is no end date.

$
```

Historical(4)

In addition to time, you can specify date and time, so one equivalent command would use an ISO datetime.

```
$ timew track 2016-07-30T09:00:00 - 2016-07-30T11:00:00 'Outline the tutorial topics'
...
```

Again, you can see all the date formats listed using the help system.

```
$ timew help date
...
```

Using date synonyms you can track time for a whole month.

```
$ timew track june - july Training
...
```

But that command will track all 30 days, all 24 hours each in June, including weekends, holidays and lunch breaks. Or does it? This is discussed later.

Hints

Many commands support hints, which are words that start with a `:` and are convenient representations to save time. Here is the `:quiet` hint, being used to suppress all feedback:

```
$ timew track 9am - 10am 'Walk the dog' :quiet
$
```

The `:quiet` hint is the same as disabling verbosity, but is easier to specify, and temporary. Another hint is `:yes`, which is used to override confirmation, by automatically answering *yes* to the question.

Some hints are shortcuts specifiers for date ranges. For example, the `:yesterday` hint is a date range representing all day yesterday. Similarly, `:lastweek` is also a date range. That makes the following two commands identical (assuming that today is the 6th):

```
$ timew track :yesterday Hiking
$ timew track 5th - 6th Hiking
```

Hints (2)

The help system lists all the supported hints.

```
$ timew help hints

Timewarrior supports hints, which are single-word command line features that
start with a colon like this:

  :week

Hints serve several purposes. This example is a shortcut for the date range
that defines the current week. Other hints, such as:

  :quiet

are ways to control the behavior of Timewarrior, in this case eliminating all
forms of feedback, for purposes of automation. The supported hints are:

  :quiet         Turns off all feedback. For automation
  :debug         Runs in debug mode, shows many runtime details
  :yes           Overrides confirmation by answering 'yes' to the questions

...
```

Charts

Timewarrior has a built-in chart that can show blocks of time by day. This is a text-based chart so it is not high resolution and has no drill-down capabilities. (Incidentally such a chart would be possible using the extension API, but by default, Timewarrior just has simple charting).

There are three charts, which are really just three variations of the same chart, all controlled by configuration. We'll take a look at these charts, but first we need some sample data to look at. Let us first track a couple of days of data, to illustrate how the charts work.

```
$ timew track 2016-08-06T09:00 - 2016-08-06T12:00 'Staff meeting'  :quiet
$ timew track 2016-08-06T12:45 - 2016-08-06T14:00 Training         :quiet
$ timew track 2016-08-06T14:15 - 2016-08-06T16:00 'Project review' :quiet
$ timew track 2016-08-06T16:15 - 2016-08-06T17:20 Research         :quiet
$ timew track 2016-08-07T08:45 - 2016-08-07T12:30 Research         :quiet
$ timew track 2016-08-07T13:15 - 2016-08-07T17:30 Research         :quiet
$
$
```

Charts (2)

We have tracked six separate intervals, and the summary report shows just that.

```
$ timew summary yesterday - now

Wk  Date       Day Tags           Start      End    Time    Total
W31 2016-08-06 Sat Staff meeting   9:00:00 12:00:00 3:00:00
                   Training       12:45:00 14:00:00 1:15:00
                   Project review 14:15:00 16:00:00 1:45:00
                   Research       16:15:00 17:20:00 1:05:00  7:05:00
W31 2016-08-07 Sun Research        8:45:00 12:30:00 3:45:00
                   Research       13:15:00 17:30:00 4:15:00  8:00:00
                                                            --------
                                                            15:05:00

$
```

The summary report gives accurate time values, so this should be the preferred report for this reason. Let's look at the first chart, the `day` report.

```
$ timew day

Sun  7 0    1    2    3    4    5    6    7    8  Research          1Research          18   19   20   21   22   23


       Tracked        8:00:00
       Available     16:00:00
       Total         24:00:00

$
```

Charts (3)

Like the `summary` report, the `day` report shows data for today by default. You can make it show multiple days like this:

```
$ time day 4th - now
```

A better option is to use the `week` report.

```
$ cat foo

          0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23    Total
W31 Mon   1
    Tue   2
    Wed   3
    Thu   4
    Fri   5
    Sat   6                                             Staff meeting     Trainin Project  Resea                                    7:05
    Sun   7                                        Research               Research                                                  8:00
                                                                                                                                   ------
                                                                                                                                   15:05

          Tracked       15:05:00
          Available    152:55:00
          Total        168:00:00

$
```

Charts (4)

There is also a `month` report that looks the same, but is longer.

Again, the only difference between these reports is configuration, and you can override any of this to customize these charts. See `man timew` for full configuration details.

Reports

We have seen the `summary` and `tags` reports, but there is another useful report that shows the untracked time in the day.

```
$ timew summary

Wk  Date       Day Tags      Start      End    Time   Total
W31 2016-08-07 Sun Research   8:45:00 12:30:00 3:45:00
                   Research  13:15:00 17:30:00 4:15:00 8:00:00
                                                       ------
                                                       8:00:00


$
$ timew gaps

Wk  Date       Day    Start      End    Time   Total
W31 2016-08-07 Sun   0:00:00   8:45:00 8:45:00
                    12:30:00  13:15:00 0:45:00
                    17:30:00   0:00:00 6:30:00 16:00:00
                                                -------
                                               16:00:00

$ _
```

The `gaps` report is useful for finding time in the day where you were not tracking time. In the example the gaps correspond well to time that was not spent working, so there is no need for adjustments.

All reports in Timewarrior can be filtered by time interval and tags, but all have a default time interval.

Themes

Timewarrior has color themes that are mostly used by the charts to color the different parts of the display. To use a color theme, add this line to your `~/.timewarrior/timewarrior.cfg` file with a text editor:

```
import /path/to/themes/dark_green.theme
```

Note that the path `/path/to/themes` is a placeholder. Your installation will likely use a path more like this, but it should be noted that this path varies depending on platform and the wishes of the packager.

```
import /usr/local/share/doc/timew/doc/themes/dark_green.theme
```

There are a few simple themes available initially, but this collection will grow and improve. Timewarrior is also likely to make greater use of themes in future releases.

Holidays

Timewarrior can also make use of Holiday files. In the same way that a color theme was imported into the configuration file, a holiday file can also be used:

```
import /usr/local/share/doc/timew/doc/holidays/holidays.en-US
```

Again, that path is platform-dependent, so use the appropriate path for your system.

When a holiday file is used, Timewarrior knows that there are some days in the year that are not work days. While this changes nothing about your ability to track time, it does affect some automatic tracking features, which we will cover next.

There is a `README` document and a `refresh` script provided with the holiday file, which explains how to update the holiday data, and how to obtain holiday files for other locales. Note that only the `en-US` locale is included by default.

Exclusions

Exclusions are a very powerful Timewarrior feature, and make automatic time tracking possible. An exclusion – much like a holiday – represents a block of time where you do not expect to work.

The simplest exclusion is a day off work. Suppose you took a day off, on August 4th. You can define this day as an exclusion.

```
$ timew config exclusions.days.2016_08_04 off :yes
Config file /Users/sample/.timewarrior/timewarrior.cfg modified.
$
```

Exclusions (2)

The date must be formatted in this precise way. Note the use of the `:yes` hint to override confirmation. Now we see on the `week` report that the day is marked as non-working.

```
$ timew week

          0    1    2    3    4    5    6    7    8    9    10   11   12   13   14   15   16   17   18   19   20   21   22   23    Total
W31 Mon   1
    Tue   2
    Wed   3
    Thu   4
    Fri   5
    Sat   6
    Sun   7

                                                                                                                                    0:00

          Tracked          0:00:00
          Available      156:00:00
          Total          156:00:00

$
```

Exclusions (3)

Similarly, if you worked on a holiday, you can define that day as a work day, and therefore available for automatic tracking.

```
$ timew config exclusions.days.2016_01_01 on :yes
Config file /Users/sample/.timewarrior/timewarrior.cfg modified.
$
```

Here we have (re)defined January 1st as a working day, which was previously defined as a holiday in the `en-US` locale.

Exclusions (4)

In addition to whole days working or not working, you can define exclusions for each day of the week to represent your work schedule. Suppose you work a regular weekly schedule, that starts at 8:30am, Monday to Friday, with weekends off. You take a 45-minute lunch break each day, and leave work at 5:30pm. On Fridays you leave early.

```
$ timew config exclusions.monday    '<8:30 12:30-13:15 >17:30' :yes
$ timew config exclusions.tuesday   '<8:30 12:30-13:15 >17:30' :yes
$ timew config exclusions.wednesday '<8:30 12:30-13:15 >17:30' :yes
$ timew config exclusions.thursday  '<8:30 12:30-13:15 >17:30' :yes
$ timew config exclusions.friday    '<8:30 12:30-13:15 >16:30' :yes
$ timew config exclusions.saturday  '>0:00'                    :yes
$ timew config exclusions.sunday    '>0:00'                    :yes
$
```

Exclusions (5)

You can view this in the `week` report, and here we will use the `:blank` hint to remove all the tracked data from the report, leaving only the exclusions.

```
$ timew week :blank

          0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23    Total
W31 Mon   1
    Tue   2
    Wed   3
    Thu   4
    Fri   5
    Sat   6
    Sun   7                                                                                                                          ──────
                                                                                                                                     0:00

          Tracked          0:00:00
          Available       32:00:00
          Total           32:00:00

$
$
```

Your whole work week is defined. While you are not at all constrained by this defined schedule, it does control automatic time tracking.

Automatic Tracking

Once you have defined exclusions for your workweek, the tracked time will conform to the set boundaries. Suppose this is our work week:

```
$ timew week august

          0    1    2    3    4    5    6    7    8    9    10   11   12   13   14   15   16   17   18   19   20   21   22   23    Total
W31 Mon   1
    Tue   2
    Wed   3
    Thu   4
    Fri   5
    Sat   6
    Sun   7                                                                                                                          ──────
                                                                                                                                     0:00

          Tracked          0:00:00
          Available       32:00:00
          Total           32:00:00

$
```

Automatic Tracking (2)

There is no tracked time, but there are 40:15:00 hours available to be tracked. If we worked the whole week on our *Research* project, we can track all that time with one command:

```
$ timew start monday Research
Tracking Research
  Started 2016-08-01T00:00:00
  Current         07T21:38:29
  Total             165:38:29
$
$ timew week august

          0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23    Total
W31 Mon   1                                        Research Research Research Research Research Research                            8:15
    Tue   2                                        Research Research Research Research Research Research                            8:15
    Wed   3                                        Research Research Research Research Research Research                            8:15
    Thu   4
    Fri   5                                        Research Research Research Research                                              7:15
    Sat   6
    Sun   7                                                                                                                        ------
                                                                                                                                    32:00

          Tracked       32:00:00
          Available      0:00:00
          Total         32:00:00

$
```

We see that the exclusions are automatically subtracted from the time, and fill the whole week, leaving no available time.

Corrections

With or without the use of exclusions, there is always the need to make corrections to the tracked time. As mentioned earlier, it is quite likely that the clock would be left running by mistake, or that the tracking was started or stopped at the wrong time. We will create some incorrect time tracking, and then correct it.

```
$ timew track 9am - 10am ProjectA
Recorded ProjectA
  Started 2016-08-07T09:00:00
  Ended              10:00:00
  Total               1:00:00
$ timew track 10:12am - 10:30am projectB
Recorded projectB
  Started 2016-08-07T10:12:00
  Ended                30:00
  Total               0:18:00
$ timew start 12pm projectC
Tracking projectC
  Started 2016-08-07T12:00:00
  Current            21:41:18
  Total               9:41:18
$ timew summary

Wk  Date       Day Tags      Start    End      Time    Total
W31 2016-08-07 Sun ProjectA   9:00:00 10:00:00 1:00:00
                   projectB  10:12:00 10:30:00 0:18:00
                   projectC  12:00:00        - 9:41:24 10:59:24
                                                      --------
                                                      10:59:24

$
```

Corrections (2)

I actually started work at 8:30am, but forgot to start the clock at the beginning. I also did not take a break at 10am, I again forgot to start the clock. Then I was working on projectC since about 11am, but didn't record it properly.

These are typical mistakes, and are easy to fix, as there are several commands for making adjustments like this.

To make an adjustment to an interval, we need first to identify that interval. The `summary`, `day`, `week` and `month` reports all support the `:ids` hint for this purpose. If we take a look at the `summary` report with the hint:

```
$ timew summary :ids

Wk  Date       Day ID Tags      Start    End      Time    Total
W31 2016-08-07 Sun @3 ProjectA  9:00:00 10:00:00 1:00:00
                   @2 projectB 10:12:00 10:30:00 0:18:00
                   @1 projectC 12:00:00       - 9:41:54 10:59:54
                                                        ========
                                                        10:59:54

$
```

Corrections (3)

Notice how the intervals now have IDs: `@3` for the oldest interval. Let's make some corrections:

```
$ timew move @3 8:30am
Moved @3 to 2016-08-07T08:30:00
$ timew lengthen @3 30mins
Lengthened @3 by 0:30:00
$ timew move @1 11:00
Moved @1 to 2016-08-07T11:00:00
$
$ timew summary :ids

Wk  Date       Day ID Tags      Start    End      Time    Total
W31 2016-08-07 Sun @3 ProjectA  8:30:00 10:00:00  1:30:00
                   @2 projectB 10:12:00 10:30:00  0:18:00
                   @1 projectC 11:00:00        - 10:42:39 12:30:39
                                                          ________
                                                          12:30:39

$
```

Corrections (4)

There is still a problem, interval `@2` needs to occupy the slot between 10am and 11am. There is a `:fill` hint that does this for us.

```
$ timew @2 move 10:02 :fill
Backfilled @2 to 2016-08-07T10:00:00
Filled @2 to 2016-08-07T11:00:00
Moved @2 to 2016-08-07T10:00:00
$
$ timew summary

Wk  Date       Day Tags      Start      End     Time   Total
W31 2016-08-07 Sun ProjectA   8:30:00 10:00:00  1:30:00
                   projectB 10:00:00 11:00:00  1:00:00
                   projectC 11:00:00        - 10:43:15 13:13:15
                                                       --------
                                                       13:13:15

$
```

The `@1` ID always represents the newest interval. Note that if my corrections have changes the order, then the IDs would be different, and you would need to run `timew summary :ids` again to see the new IDs.

Corrections (5)

Finally, I actually stopped for lunch at 12:30 for 45 minutes:

```
$ timew stop 12:30pm
Recorded projectC
  Started 2016-08-07T11:00:00
  Ended             12:30:00
  Total              1:30:00
$
$ timew continue
Tracking projectC
  Started 2016-08-07T21:43:53
  Current                 53
  Total              1:30:00
$
$ timew summary

Wk  Date       Day Tags      Start    End      Time    Total
W31 2016-08-07 Sun ProjectA   8:30:00 10:00:00 1:30:00
                   projectB  10:00:00 11:00:00 1:00:00
                   projectC  11:00:00 12:30:00 1:30:00
                   projectC  21:43:53        - 0:00:07 4:00:07
                                                      ------
                                                      4:00:07

$
$ timew move @1 1:15pm
Moved @1 to 2016-08-07T13:15:00
$
```

Corrections (6)

Let’s keep going, even though this example has already exceeded credibility, to demonstrate more. I need to change that *projectB* interval to use *projectB1* and *projectB2* tags, and divide the time between the two. We will split the interval, the re-tag it.

```
$ timew summary :ids

Wk  Date       Day ID Tags      Start    End      Time    Total
W31 2016-08-07 Sun @4 ProjectA  8:30:00 10:00:00 1:30:00
                   @3 projectB 10:00:00 11:00:00 1:00:00
                   @2 projectC 11:00:00 12:30:00 1:30:00
                   @1 projectC 13:15:00        - 8:29:42 12:29:42
                                                         ________
                                                         12:29:42

$ timew @3 split
Split @3
$
$ timew summary :ids

Wk  Date       Day ID Tags      Start    End      Time    Total
W31 2016-08-07 Sun @5 ProjectA  8:30:00 10:00:00 1:30:00
                   @4 projectB 10:00:00 10:30:00 0:30:00
                   @3 projectB 10:30:00 11:00:00 0:30:00
                   @2 projectC 11:00:00 12:30:00 1:30:00
                   @1 projectC 13:15:00        - 8:29:54 12:29:54
                                                         ________
                                                         12:29:54
```

Corrections (7)

```
$ timew untag @4 @3 projectB
Removed projectB from @4
Removed projectB from @3
$ timew tag @4 projectB1
Added projectB1 to @4
$ timew tag @3 projectB2
Added projectB2 to @3
$ timew summary :ids

Wk  Date       Day ID Tags       Start      End    Time    Total
W31 2016-08-07 Sun @5 ProjectA    8:30:00 10:00:00 1:30:00
                   @4 projectB1  10:00:00 10:30:00 0:30:00
                   @3 projectB2  10:30:00 11:00:00 0:30:00
                   @2 projectC   11:00:00 12:30:00 1:30:00
                   @1 projectC   13:15:00        - 8:30:19 12:30:19
                                                           --------
                                                           12:30:19

$ _
```

Notice how the `split` command just divided `@3` into two even-sized intervals. But we're still not done - I brought lunch to work and ate at my desk while working so let's eliminate that lunch break.

Corrections (8)

```
$ timew join @2 @1
Joined @2 and @1
$
$ timew summary

Wk  Date       Day Tags      Start      End      Time   Total
W31 2016-08-07 Sun ProjectA   8:30:00 10:00:00  1:30:00
                   projectB1 10:00:00 10:30:00  0:30:00
                   projectB2 10:30:00 11:00:00  0:30:00
                   projectC  11:00:00       - 10:46:10 13:16:10
                                                       ________
                                                       13:16:10

$
```

Now whether this report now accurately represents your day, or whether it is a fiction you need to report (no judgement here), Timewarrior supports it, but let's stop - the example can't take much more.

We saw the `stop` command with a specific end time, the `move`, the `lengthen`, `split`, `join`, `tag` and `untag` commands and the `:fill` hint being used. There are also the `shorten`, `cancel`, and `delete` commands. See `man timew` for full details.

Extensions

Timewarrior reports are not sophisticated, they are minimally functional and focus on simply displaying the data. What if you need a report broken down by tags, with weekly subtotals?

Timewarrior supports extension reports, via the extension API. This is mechanism that allows you to write a report using any language you choose. Don't like the way the summary report shows the data? Write your own. Better yet, share the result, and we'll build a list of 3rd party reports.

We've included one extension report with Timewarrior, and will add more. The one provided is:

```
$ ls -l /usr/local/share/doc/timew/ext/totals.py
-rw-r--r--  1 root  admin  3606 Jul 30 15:34 /usr/local/share/doc/timew/ext/totals.py
$ _
```

Again, that path is platform-dependent, so use the appropriate path for your system.

Extensions (2)

Note that this is a Python script, and to use this you'll need to have Python installed.

To install and use this extension, or any other, simply copy it to your `~/.timewarrior/extensions` directory and make sure it is executable.

```
$ cp /usr/local/share/doc/timew/ext/totals.py ~/.timewarrior/extensions
$ chmod +x ~/.timewarrior/extensions/totals.py
$
```

Caution

Extensions are dangerous things.

As with any downloaded program, be careful – you are giving execute permission to software that may harm you.

Fortunately this is the world of open source, and while that does not mean the software is safe, it does mean you have the necessary access to audit the code and prove it is safe.

If you find the code safe, use it. If you are unsure, don’t.

Once the extension is in the `extensions` directory, and executable, it should be visible to the `extensions` command.

```
$ timew extensions

Extensions located in:
  /Users/sample/.timewarrior/extensions

Extension Status
totals.py Active

$
```

Caution (2)

Additionally, the `diagnostics` command will report the presence and status of this extension.

Once an extension is ready to use, it is used in the same way that the `summary` report is used, with date range and/or tag filtering. The command you use is compared to the name of the script, and if unique, is a match. All of these commands are equivalent:

```
$ timew totals.py
$ timew totals.p
$ timew totals.
$ timew totals
$ timew total
```

Caution (3)

Here is the report run with no filter, and therefore against all recorded data, which is not much in this example.

```
$ timew totals

Total by Tag, for 2016-08-07 12:30:00 - 2016-08-08 01:52:44

Tag          Total
ProjectA     1:30:00
projectB1    0:30:00
projectB2    0:30:00
projectC    10:52:44

Total       13:22:44

$
```

Extensions are not restricted to emitting text, they could for example output HTML, DOT, PDF, PNG, JPEG …

Integration

Timewarrior integrates with Taskwarrior by means of a Taskwarrior `on-modify` hook script. Once installed, this means that whenever a task is active, Timewarrior is used to track the time.

To install the script, copy it into your Taskwarrior directory:

```
$ cp /usr/local/share/doc/timew/ext/on-modify.timewarrior ~/.task/hooks
$ chmod +x ~/.task/hooks/on-modify.timewarrior
$
```

Again, that path is platform-dependent, so use the appropriate path for your system.

Integration (2)

Now we can create a task, start it, complete it and automatically track it.

```
$ task add 'Investigate Timewarrior' +software
Created task 174.
Warning: no project was assigned.
$
$ task 174 start
Starting task 7dbf5537 'Investigate Timewarrior'.
Started 1 task.
Tracking "Investigate Timewarrior" software
  Started 2016-08-07T21:58:02
  Current                 02
  Total              0:00:00
$
$
$ task 174 done
Completed task 7dbf5537 'Investigate Timewarrior'.
Completed 1 task.
Recorded "Investigate Timewarrior" software
  Started 2016-08-07T21:58:02
  Ended                   12
  Total              0:00:10
$
$ timew summary

Wk  Date       Day Tags                          Start    End      Time     Total
W31 2016-08-07 Sun ProjectA                       8:30:00 10:00:00  1:30:00
                   projectB1                     10:00:00 10:30:00  0:30:00
                   projectB2                     10:30:00 11:00:00  0:30:00
                   projectC                      11:00:00 21:57:25 10:57:25
                   Investigate Timewarrior, software 21:58:02 21:58:12  0:00:10 13:27:35
                                                                               ________
                                                                               13:27:35

$
```

Links

This talk was shamelessly converted from the Online Tutorial, there are more detailed Online Docs.

Starting today, there is XXX as well.

Feedback is highly appreciated!