Taskwarrior

From Source to Getting Things Done

Dirk Deimeke

November, 6th 2016

Taskwarrior Academy / OpenRheinRuhr 2016

Prolog

Dirk Deimeke – d5e.org

Project founder: Paul Beckingham

- I started out using Gina Trapani's todo.sh, which was great, but I soon wanted features that would have been difficult to implement in a shell script, so I wrote my own.
- There are many different methodologies people use for managing their work, and Taskwarrior tries to walk a line through the middle of all that, with features for all the different approaches.
- Taskwarrior is intended to scale with the user, from very simple straightforward usage up to quite sophisticated task management.
- **It has a lot of features, but tries to remain simple to use.**

Reasons pro Taskwarrior

Taskwarrior

- is easy to learn.
- grows along with the work.
- is unbelievably powerful.
- is very fast.
- is easily extensible.
- is platform independent:
  - Most flavours of Unix and Linux, including Mac OS X
  - Windows 10 Linux Subsystem
    Other Windows versions with Cygwin (unsupported)
  - Android with Termux
  - Third-Party Apps (Android-Client, GUI based on NodeJS, . . .)
- is actively developed.
- can be influenced by users (feature requests).
- has excellent and very friendly support.

This workshop . . .

This workshop hopefully is a *real workshop*.

It will live from you doing things and asking,

it is not about me talking all of the time.

Nevertheless I will show you every command.

Installation

Installation from source

Attention!

Since some packagers (Debian and Ubuntu as examples) implement their thinking of the place where files have to be without changing the templates, an installation from source is the recommended way.

All you need to compile is

- `GnuTLS` (ideally version 3.2 or newer)
- `libuuid`
  (Darwin, FreeBSD ... include `libuuid` functionality in libc.)
- `CMake` (2.8 or newer)
- `make`
- C++ Compiler supporting C++11
  (`GCC 4.7` or `Clang 3.3` or newer)

Install dependencies

Install the necessary packages with your package manager.

CentOS, Fedora, openSUSE
: gnutls-devel libuuid-devel cmake gcc-c++ # or clang

Debian, Ubuntu
: libgnutls28-dev uuid-dev cmake g++ # or clang

Mac OS X
: Install Xcode from Apple, via the AppStore, launch it, and select from some menu that you want the command line tools.
With Homebrew install the necessary packages:
`brew install cmake git gnutls`

Get the source

Either

```
curl -LO http://taskwarrior.org/download/task-2.5.1.tar.gz
tar xzf task-2.5.1.tar.gz
cd task-2.5.1
```

or

```
git clone --recursive https://git.tasktools.org/scm/tm/task.git task.git
cd task.git

# Updates
git pull --all --recurse-submodules=yes
# git submodule init (if first time)
git submodule update
```

Recent development version

```
git clone --recursive https://git.tasktools.org/scm/tm/task.git task.git
cd task.git

git checkout 2.6.0

# Magic happening here
git submodule init
git submodule update
```

Compile it

Three easy steps . . . hopefully!

1. `cmake -DCMAKE_BUILD_TYPE=release .`
   add the following before the dot (if necessary)
   `-DCMAKE_INSTALL_PREFIX=/home/user/task`
2. `make`
   `export MAKEFLAGS="-j 2"` to speed things up (number of CPUs)
3. `sudo make install`

Test it

```
$ task diagnostics
A configuration file could not be found in

Would you like a sample /home/dirk/.taskrc created, so Taskwarrior can
     proceed? (yes/no) yes

task 2.6.0
   Platform: Linux

Compiler
    Version: 6.2.1 20160916 (Red Hat 6.2.1-2)
       Caps: +stdc +stdc_hosted +LP64 +c8 +i32 +l64 +vp64 +time_t64
 Compliance: C++11

Build Features
      Built: Nov  3 2016 14:14:33
     Commit: bcfebff
     CMake: 3.6.2
   libuuid: libuuid + uuid_unparse_lower
 libgnutls: 3.5.5
Build type: release
```

Simple ToDo-Lists

A simple example

```
task add

task list

task <ID> start

task list

task <ID> stop

task list

task <ID> done
```

Choose a theme

Theme

Uncomment the theme you want to use from `~/.taskrc`

```
# Color theme (uncomment one to use)
#include /usr/local/share/doc/task/rc/light-16.theme
#include /usr/local/share/doc/task/rc/light-256.theme
#include /usr/local/share/doc/task/rc/dark-16.theme
#include /usr/local/share/doc/task/rc/dark-256.theme
#include /usr/local/share/doc/task/rc/dark-red-256.theme
#include /usr/local/share/doc/task/rc/dark-green-256.theme
#include /usr/local/share/doc/task/rc/dark-blue-256.theme
#include /usr/local/share/doc/task/rc/dark-violets-256.theme
#include /usr/local/share/doc/task/rc/dark-yellow-green.theme
#include /usr/local/share/doc/task/rc/dark-gray-256.theme
#include /usr/local/share/doc/task/rc/dark-gray-blue-256.theme
#include /usr/local/share/doc/task/rc/solarized-dark-256.theme
include /usr/local/share/doc/task/rc/solarized-light-256.theme
#include /usr/local/share/doc/task/rc/no-color.theme
```

Theme

Packaged Taskwarrior

Your package distributor might have different ideas where the theme files should be.

Check with

```
find / -name no-color.theme -type f 2>/dev/null
```

General

Nearly all commands work on a bunch of tasks

There is a lot more to explore.

Even the commands from the last section are more mighty than they seem.

- task add <mods>
- task <filter> list
- task <filter> start <mods>
- task <filter> stop <mods>
- task <filter> done <mods>

To get an overview, take a look at the cheat sheet (pdf, 145kB) (or come over and grab a printed copy).

task <filter> command <mods>

- Is the basic usage of all task related **write** commands.
- Write commands can operate on one task or a group of tasks or even on all tasks.
- Every command may be **abbreviated** up to the minimum that is necessary to identify a single command.
- Filters can be anything from nothing to simple IDs to regular expressions or Boolean constructs.
- Modifications can be either a change of description, a change of dates or anything else that changes a task.
- In our simple example we already used the write commands **add**, **done**, **start** and **stop**.

Scripts

```
# Scripts shipped with Taskwarrior
ls /usr/local/share/doc/task/scripts/*

# Commandline completion tabtabtabtabtabtab ;-)
source /usr/local/share/doc/task/scripts/bash/task.sh

# Make it persistent
echo source /usr/local/share/doc/task/scripts/bash/task.sh >> .bashrc

# Syntaxhighlighting for vim
[[ -d ~/.vim ]] || mkdir ~/.vim
cp -r /usr/local/share/doc/task/scripts/vim ~/.vim
```

Most important commands

These are the most important commands, just because I use them most ;-)

- **task <filter> modify**
  The name says it, it modifies tasks according to the filter used.
- **task <filter> edit**
  This starts your favourite editor with the tasks you want to change.
  (Remember the syntax highlighting for vim?)
- **task undo**
  Reverts the most recent change to a task.
- **task help**
  Gives an overview of implemented commands and custom reports.

No kidding!

- **man task (taskrc, task-sync)**
  Show the (almighty) man-page(s). Unlike the man-pages of many other programs they are extremely helpful and full of information and examples.
  Try them!

Working with dates

Dateformats (incomplete) – from 'man taskrc'

```
m  minimal-digit month,   for example 1 or 12
d  minimal-digit day,     for example 1 or 30
y  two-digit year,        for example 09
D  two-digit day,         for example 01 or 30
M  two-digit month,       for example 01 or 12
Y  four-digit year,       for example 2009
a  short name of weekday, for example Mon or Wed
A  long name of weekday,  for example Monday or Wednesday
b  short name of month,   for example Jan or Aug
B  long name of month,    for example January or August
V  weeknumber,            for example 03 or 37
H  two-digit hour,        for example 03 or 11
N  two-digit minutes,     for example 05 or 42
S  two-digit seconds,     for example 07 or 47
```

ISO supported

Defined dateformats

The dateformat you define, will be used in **addition** to all the standard supported ISO-8601 formats.

Set dateformat

```
task show dateformat

task config dateformat YMD
task config dateformat.annotation YMD
task config dateformat.report YMD

# my dateformat once was YMD-HN

task show dateformat

grep dateformat ~/.taskrc
```

Set weekstart

```
task show weekstart

task config weekstart Monday

task show | wc -l # nearly everything can be customized
235
```

Special dates (1)

Relative wording

```
task ... due:today
task ... due:yesterday
task ... due:tomorrow
```

Day number with ordinal

```
task ... due:23rd
task ... due:3wks
task ... due:1day
task ... due:9hrs
```

At some point or later (sets the wait date to 1/18/2038)

```
task ... wait:later
task ... wait:someday
```

Special dates (2)

Start / end of . . . (remember weekstart setting)

```
task ...due:sow/eow # week
task ...due:soww/eoww # workweek
task ...due:socw/eocw # current week
task ...due:som/eom # month
task ...due:soq/eoq # quarter
task ...due:soy/eoy # year
```

Next occurring weekday

```
task ...due:fri
```

Due and wait

```
task add due:20161231 "Celebrate␣Sylvester"
task add due:Sunday "Drive␣home"

task list

task x modify wait:20161107

task list
```

Urgency and next

Based on your tasks attributes especially – but not limited to – the due date, Taskwarrior calculates an urgency value which will be used by some reports to sort the tasks.

You can increase urgency by adding the `+next` tag.

This is a very complex topic and goes beyond the scope of this introductory workshop.

Recurrence

```
task waiting
task x modify due:eom recur:monthly

task list

task recurring

# task id changed from x (task modify) to y
# try task x edit
```

Recurrence modifiers (1)

hourly	Every hour.
daily, day, 1da, 2da, . . .	Every day or a number of days.
weekdays	Mondays, Tuesdays, Wednesdays, Thursdays, Fridays and skipping weekend days.
weekly, 1wk, 2wks, . . .	Every week or a number of weeks.
biweekly, fortnight	Every two weeks.
monthly	Every month.

Recurrence modifiers (2)

quarterly, 1qtr, 2qtrs, . . .
: Every three months, a quarter, or a number of quarters.

semiannual
: Every six months.

annual, yearly, 1yr, 2yrs, . . .
: Every year or a number of years.

biannual, biyearly, 2yr
: Every two years.

Recurrence based on hours

No alarm!

Nothing is wrong with setting a recurrence to hours or minutes, but please keep in mind that Taskwarrior is not and never will be a calendar application or an alarm clock.

If you want to get notified, you are on your own.

Until and entry

```
task add due:eom recur:monthly until:20161231 "Pay␣installment␣for␣
     credit"

task add "Prepare␣slides␣for␣workshop"
task x modify entry:yesterday

task list
```

Holiday

Attention!

Holiday has nothing in common with the German words *Ferien* or *Urlaub* (this would be vacation). (Public) Holiday means *Feiertag*.

You can add holidays by either adding them via `task config` on the commandline or by adding them directly to the `~/.taskrc`-File or by including an external holiday definition.

On holidata.net you find a growing list of holiday dates, licensed CC-BY and offered by volunteers. Service was introduced by the Taskwarrior team, who is responsible for hosting and conversion to different formats.

Add holiday / Configure calendar

```
task config holiday.swissnationalday.name Swiss National Day
task config holiday.swissnationalday.date 20170801

# Holiday is not highlighted by default
task cale 08 2017

task show calendar
task config calendar.holidays full

task cale 08 2017
```

Calendar with due tasks

```
task config calendar.holidays sparse
task config calendar.details full

task cale
```

Getting sorted

Project and subproject

```
task x modify pro:openrheinruhr
task y modify pro:openrheinruhr.workshop
task z modify pro:private

task list
```

Projects

```
task projects

task pro:openrheinruhr ls

task x done
```

Tags

```
task x modify +banking
task y modify +banking

task list

task x mod -banking +oberhausen

task +oberhausen list
```

Priorities

```
task long

task x modify pri:H # can be either (H)igh, (M)edium or (L)ow

task long
```

Annotations

```
task x annotate "Do␣not␣forget␣your␣head"

task y annotate "Use␣wifes␣account"

task list

task y denotate "Use␣wifes␣account"
```

Dependencies

Dependency, part 1

```
task add "Send␣letter␣to␣Fritz"

task add "Write␣letter"

task x modify depends:y

task blocked

task unblocked
```

Dependency, part 2

```
task x done

task list
```

Undo

```
task undo
```

Dependency, part 3

```
task x,y done
```

```
task blocked
```

Reports

Predefined reports (from task reports), part 1

These reports were already used.

- **blocked** Lists all blocked tasks matching the specified criteria
- **list** Lists all tasks matching the specified criteria
- **long** Lists all task, all data, matching the specified criteria
- **projects** Shows a list of all project names used, and how many tasks are in each
- **recurring** Lists recurring tasks matching the specified criteria
- **unblocked** Lists all unblocked tasks matching the specified criteria
- **waiting** Lists all waiting tasks matching the specified criteria

Predefined reports (from task reports), part 2

New ones:

- **active** Lists active tasks matching the specified criteria
- **all** Lists all tasks matching the specified criteria, including parents of recurring tasks
- **blocking** Blocking tasks
- **burndown.daily** Shows a graphical burndown chart, by day
- **burndown.monthly** Shows a graphical burndown chart, by month
- **burndown.weekly** Shows a graphical burndown chart, by week
- **completed** Lists completed tasks matching the specified criteria

Predefined reports (from task reports), part 3

And more:

- **ghistory.annual** Shows a graphical report of task history, by year
- **ghistory.monthly** Shows a graphical report of task history, by month
- **history.annual** Shows a report of task history, by year
- **history.monthly** Shows a report of task history, by month
- **information** Shows all data and metadata for specified tasks
- **ls** Minimal listing of all tasks matching the specified criteria
- **minimal** A really minimal listing
- **newest** Shows the newest tasks
- **next** Lists the most urgent tasks

Predefined reports (from task reports), part 4

The leftovers:

- **oldest** Shows the oldest tasks
- **overdue** Lists overdue tasks matching the specified criteria
- **ready** Most urgent actionable tasks
- **summary** Shows a report of task status by burndown-dailyoject
- **tags** Shows a list of all tags used

26 reports in total (as told by `task reports`)

Test the reports

```
task burndown.daily

task ghistory.annual
task ghistory.monthly

task history.monthly

task ls

task minimal

task summary
```

Report definitions

```
task show report.minimal
task show report.list

task show report # to see all definitions
```

Dirks former task list

```
echo "
report.ll.description=Dirks␣task␣list
report.ll.columns=id,project,priority,due,due.countdown,tags,description
report.ll.labels=ID,Project,Pri,Due,Countdown,Tags,Description
report.ll.sort=due+,priority-,project+,description+
report.ll.filter=status:pending
" >> ~/.taskrc

task ll
```

Set default command

```
task show default

task config default.command ll
task
```

Filtering

Filtering in general

You can filter for any modifier. If you don't use a modifier description is searched for the term, which may be a regular expression, on the command line. Filters may be combined.

The following attribute modifiers maybe applied as well. Names in brackets can be used alternatively.

So a filter can look like `attribute.modifier:value`.

- before, after
- none, any
- is (equals), isnt (not)
- has (contains), hasnt
- startswith (left), endswith (right)
- word, noword

Searches

```
task

task pay

task /[Pp]ay/
```

Attribute modifiers

```
task due.before:20161130

task project.not:taskwarrior

task project:openrheinruhr +banking
task status:completed project:openrheinruhr
task status:completed project:openrheinruhr completed

task show report.ll.filter
```

Or . . .

```
task list

task \( pro:taskwarrior or pro:private \) list
# Brackets must be escaped for the shell
```

Search configuration

```
task show search

task show regex
```

Filter in reports

```
task show filter
```

Contexts

Context is a user-defined filter, which is automatically applied to all commands that filter the task list. In particular, any report command will have its result affected by the current active context.

- `task context define <name> <filter>`
- `task context delete <name>`
- `task context <name>` – sets active context
- `task context show` – shows active context
- `task context list` – lists available contexts
- `task context none` – clears active context

Miscellanous

Virtual Tags (1)

- **ACTIVE** – Task is started
- **ANNOTATED** – Task has annotations
- **BLOCKED** – Task is blocked
- **BLOCKING** – Task is blocking
- **CHILD** – Task has a parent
- **COMPLETED** – Task has completed status
- **DELETED** – Task has deleted status
- **DUE** – Task is due
- **LATEST** – Task is the newest added task
- **MONTH** – Task is due this month
- **ORPHAN** – Task has any orphaned UDA values
- **OVERDUE** – Task is overdue
- **PARENT** – Task is a parent
- **PENDING** – Task has pending status

Virtual Tags (2)

- **PRIORITY** – Task has a priority
- **PROJECT** – Task has a project
- **READY** – Task is actionable
- **SCHEDULED** – Task is scheduled
- **TAGGED** – Task has tags
- **TODAY** – Task is due today
- **TOMORROW** – Task is due sometime tomorrow
- **UDA** – Task has any UDA values
- **UNBLOCKED** – Task is not blocked
- **UNTIL** – Task expires
- **WAITING** – Task is waiting
- **WEEK** – Task is due this week
- **YEAR** – Task is due this year
- **YESTERDAY** – Task was due sometime yesterday

This is by far not all

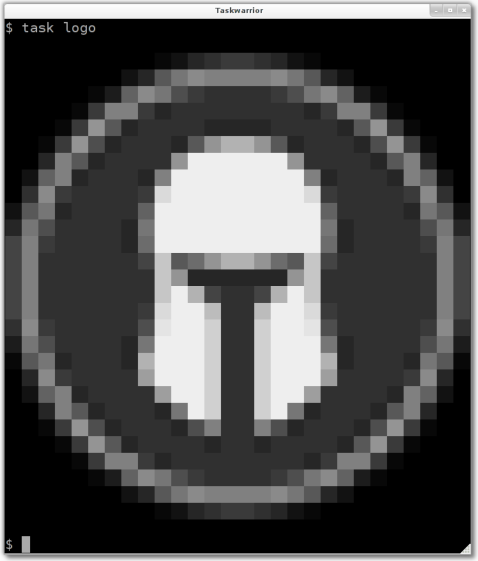
task log
: for logging a task after it is already done.

task diag
: to help support for diagnostic purpose.

UDA
: User defined attributes.

. . .
: and many more!

Questions?



```
$ task logo

$
```

Epilog

Getting Help

There are several ways of getting help:

- Submit your details to our Q & A site, then wait patiently for the community to respond.
- Email us at support@taskwarrior.org, then wait patiently for a volunteer to respond.
- Join us IRC in the #taskwarrior channel on Freenode.net, and get a quick response from the community, where, as you have anticipated, we will walk you through the checklist above.
- Even though Twitter is no means of support, you can get in touch with @taskwarrior.
- We have a User Mailinglist which you can join anytime to discuss about Taskwarrior and techniques.
- The Developer Mailinglist is focussing on a more technical oriented audience.

Thanks for your patience!

Dirk Deimeke, Taskwarrior-Team, 2016, CC-BY

dirk@deimeke.net – d5e.org