TaskJuggler 3.x - Project Management for Linux Users

Project Management is more than Gantt Charts

Chris Schlaeger

chris@linux.com

Chemitzer Linux-Tage 2012

(english translation by Sebastian Tramp)

TaskJuggler and Linux

- TaskJuggler was initially developed to support the development unit of a Linux distribution
- It is still in use by some distributions such as Fedora
- It uses a simple project description language
- TaskJuggler works similar to a compiler or LaTeX
- It consists of a few tools which provide the main features and are integrated with other programs

Ruby Runtime Environment

- TaskJuggler 3.x is written in ruby
- The main functions are ruby 1.8.7 compatible
- The server and scripting functions need at least ruby 1.9.3
- We generally suggest to use ruby 1.9.3 since it is 3 times faster than ruby 1.8.7

Gem Installation

- Ruby is available on any Linux distribution
- Unfortunately many distributions provide only ruby 1.8
- RubyGem is the ruby package management tool
- ruby gems are OS and CPU architecture independent
- The installation mostly easy. A single command downloads the gem package as well as its dependencies and installs everything in you system:

```
gem install taskjuggler
```

Ruby 1.9 Installation

- The current version is available at `ruby-lang.org` `http://www.ruby-lang.org/en/downloads/`
- decompress, configure and install

```
tar -Zxvf ruby-X.X.X-*.tar.gz
cd ruby-X.X.X-*
./configure --program-suffix=19
make
sudo make install
ln -s /usr/local/bin/ruby19 ${HOME}/bin/ruby
```

- `${HOME}/bin` has to be part of your `$PATH` varable
- Each major ruby version has its own gem package collection

Ruby 1.9 Installation

- The current version is available at `ruby-lang.org` `http://www.ruby-lang.org/en/downloads/`
- decompress, configure and install

```
tar -Zxvf ruby-X.X.X-*.tar.gz
cd ruby-X.X.X-*
./configure --program-suffix=19
make
sudo make install
ln -s /usr/local/bin/ruby19 ${HOME}/bin/ruby
```

- `${HOME}/bin` has to be part of your `$PATH` varable
- Each major ruby version has its own gem package collection

Ruby 1.9 Installation

- The current version is available at `ruby-lang.org`
  `http://www.ruby-lang.org/en/downloads/`
- decompress, configure and install

```
tar -Zxvf ruby-X.X.X-*.tar.gz
cd ruby-X.X.X-*
./configure --program-suffix=19
make
sudo make install
ln -s /usr/local/bin/ruby19 ${HOME}/bin/ruby
```

- `${HOME}/bin` has to be part of your `$PATH` varable
- Each major ruby version has its own gem package collection

Ruby 1.9 Installation

- The current version is available at `ruby-lang.org`
  `http://www.ruby-lang.org/en/downloads/`
- decompress, configure and install

```
tar -Zxvf ruby-X.X.X-*.tar.gz
cd ruby-X.X.X-*
./configure --program-suffix=19
make
sudo make install
ln -s /usr/local/bin/ruby19 ${HOME}/bin/ruby
```

- `${HOME}/bin` has to be part of your `$PATH` varable
- Each major ruby version has its own gem package collection

Online Help: `tj3man`

- Get help for a single TaskJuggler keyword

  ```
  tj3man <keyword>
  ```

- Use `-html` to load it in the browser

  ```
  tj3man --html <keyword>
  ```

- Get help for the online help tool

  ```
  tj3man --help
  ```

Online Help: `tj3man`

- Get help for a single TaskJuggler keyword

  ```
  tj3man <keyword>
  ```

- Use `-html` to load it in the browser

  ```
  tj3man --html <keyword>
  ```

- Get help for the online help tool

  ```
  tj3man --help
  ```

Online Help: `tj3man`

- Get help for a single TaskJuggler keyword

  ```
  tj3man <keyword>
  ```

- Use `-html` to load it in the browser

  ```
  tj3man --html <keyword>
  ```

- Get help for the online help tool

  ```
  tj3man --help
  ```

The main program: `tj3`

- Calculate a project and create reports

```
tj3 yourproject.tjp
```

- Projects can consist of more than one file

```
tj3 yourproject.tjp reports.tji
```

The main program: `tj3`

- Calculate a project and create reports

  ```
  tj3 yourproject.tjp
  ```

- Projects can consist of more than one file

  ```
  tj3 yourproject.tjp reports.tji
  ```

Client and Server: `tj3client` and `tj3d`

- Start the server

  ```
  tj3d -w
  ```

- Query the server status

  ```
  tj3client status
  ```

- (Re-)load a project

  ```
  tj3client add yourproject.tjp
  ```

- Shutdown the server

  ```
  tj3client terminate
  ```

Client and Server: `tj3client` and `tj3d`

- Start the server

```
tj3d -w
```

- Query the server status

```
tj3client status
```

- (Re-)load a project

```
tj3client add yourproject.tjp
```

- Shutdown the server

```
tj3client terminate
```

Client and Server: `tj3client` and `tj3d`

- Start the server

  ```
  tj3d -w
  ```

- Query the server status

  ```
  tj3client status
  ```

- (Re-)load a project

  ```
  tj3client add yourproject.tjp
  ```

- Shutdown the server

  ```
  tj3client terminate
  ```

Client and Server: `tj3client` and `tj3d`

- Start the server

  ```
  tj3d -w
  ```

- Query the server status

  ```
  tj3client status
  ```

- (Re-)load a project

  ```
  tj3client add yourproject.tjp
  ```

- Shutdown the server

  ```
  tj3client terminate
  ```

The Project Description

- All project data is collected in one ore more text files
- Information is structured as *properties* which consists of *attributes*
- A single *property* has always the following structure

  ```
  PROPERTY [ID] "NAME" [ { ATTRIBUTES } ]
  ```
- Elements surrounded by `[]` are optional

The Properties

A project consists of the following *properties*

- `project`: the project head
- `accounts`: accounts for cost calculation
- `resources`: human and non-human resources
- `tasks`: the project tasks
- `reports`: reports to generate

All *properties* have *attributes*, e.g. the *property* `task` has the *attributes* `start`, `duration` as well as other properties.

The Properties

A project consists of the following *properties*

- `project`: the project head
- `accounts`: accounts for cost calculation
- `resources`: human and non-human resources
- `tasks`: the project tasks
- `reports`: reports to generate

All *properties* have *attributes*, e.g. the *property* `task` has the *attributes* `start`, `duration` as well as other properties.

The Properties

A project consists of the following *properties*

- `project`: the project head
- `accounts`: accounts for cost calculation
- `resources`: human and non-human resources
- `tasks`: the project tasks
- `reports`: reports to generate

All *properties* have *attributes*, e.g. the *property* `task` has the *attributes* `start`, `duration` as well as other properties.

The Properties

A project consists of the following *properties*

- `project`: the project head
- `accounts`: accounts for cost calculation
- `resources`: human and non-human resources
- `tasks`: the project tasks
- `reports`: reports to generate

All *properties* have *attributes*, e.g. the *property* `task` has the *attributes* `start`, `duration` as well as other properties.

The Properties

A project consists of the following *properties*

- `project`: the project head
- `accounts`: accounts for cost calculation
- `resources`: human and non-human resources
- `tasks`: the project tasks
- `reports`: reports to generate

All *properties* have *attributes*, e.g. the *property* `task` has the *attributes* `start`, `duration` as well as other properties.

The Properties

A project consists of the following *properties*

- `project`: the project head
- `accounts`: accounts for cost calculation
- `resources`: human and non-human resources
- `tasks`: the project tasks
- `reports`: reports to generate

All *properties* have *attributes*, e.g. the *property* `task` has the *attributes* `start`, `duration` as well as other properties.

The Project Head

- What is the name of the project?
- When do you start?
- How long is the project?

```
project "Example"  2012-03-01 +4m
```

The Project Head

- What is the name of the project?
- When do you start?
- How long is the project?

```
project "Example"  2012-03-01 +4m
```

Localization

- Date format and time zone localization
- Time format
- Money value localization

```
project "Example"  2012-03-01 +4m {
  timezone "Europe/Berlin"
  timeformat "%d.%m.%Y"
  numberformat "-" "" "" "," 1
  currencyformat "-" "" "" "," 0
  currency "EUR"
}
```

Localization

- Date format and time zone localization
- Time format
- Money value localization

```
project "Example"  2012-03-01 +4m {
  timezone "Europe/Berlin"
  timeformat "%d.%m.%Y"
  numberformat "-" "" "" "," 1
  currencyformat "-" "" "" "," 0
  currency "EUR"
}
```

Different Scenarios

- A scenario describes a variant of the project
- You can write and calculate as much scenarios as you want
- Scenarios can be nested
- Nested scenarios inherit all *attributes* but can overwrite them

```
project "Beispiel"  2012-03-01 +4m {
  scenario plan "Plan" {
    scenario real "Realität"
  }
  now 2012-03-17
}
```

Different Scenarios

- A scenario describes a variant of the project
- You can write and calculate as much scenarios as you want
- Scenarios can be nested
- Nested scenarios inherit all *attributes* but can overwrite them

```
project "Beispiel"  2012-03-01 +4m {
  scenario plan "Plan" {
    scenario real "Realität"
  }
  now 2012-03-17
}
```

Extend the Data Model

- *Properties* can have a set of *attributes*
- You can extend this set with your own *attributes*

```
project example "Example"  2012-03-01 +4m {
  extend resource {
    text Phone "Phone"
  }
  extend task {
    reference Wiki "Wiki"
  }
}
```

Extend the Data Model

- *Properties* can have a set of *attributes*
- You can extend this set with your own *attributes*

```
project example "Example"  2012-03-01 +4m {
  extend resource {
    text Phone "Phone"
  }
  extend task {
    reference Wiki "Wiki"
  }
}
```

Define Resources

- Add an employee to the project

```
resource karl "Karl Mustermann"
```

- Create a team

```
resource ateam "Das A-Team" {
  rate 330.0
  resource karl "Karl Mustermann"
  resource erika "Erika Musterfrau"
}
```

- All nested resources re-use the *attribute* `rate`

Define Resources

- Add an employee to the project

```
resource karl "Karl Mustermann"
```

- Create a team

```
resource ateam "Das A-Team" {
  rate 330.0
  resource karl "Karl Mustermann"
  resource erika "Erika Musterfrau"
}
```

- All nested resources re-use the *attribute* `rate`

Define Resources

- Add an employee to the project

```
resource karl "Karl Mustermann"
```

- Create a team

```
resource ateam "Das A-Team" {
  rate 330.0
  resource karl "Karl Mustermann"
  resource erika "Erika Musterfrau"
}
```

- All nested resources re-use the *attribute* `rate`

Other resources such as tools

- The tool never sleeps but also does not work

```
resource tool "The Tool" {
  efficiency 0.0
  rate 500.0
}
```

The Project Task Structure

```
task "Phase 1" {
  task "Step 1"
  task "Step 2"
}
task "Phase 2" {
  task "Step 1"
  task "Step 2"
}
```

Define Dependencies

```
task p1 "Phase 1" {
  task s1 "Step 1"
  task "Step 2" {
    depends !s1 # relative ID
  }
}
task p2 "Phase 2" {
  task s1 "Step 1" {
    depends p1.s1 # absolute ID
  }
  task "Step 2"
}
```

Workload and Task Lengths

```
task p1 "Phase 1" {
  task s1 "Step 1" {
    duration 2d
  task "Step 2" {
    length 10d
  }
  task "Step 3" {
    effort 5d
    allocate karl
  }
}
```

Lists and Reports

- TaskJuggler supports differnt types of lists and reports
  - Task lists
  - Employee and resource lists
  - Calendar
  - Time sheet forms
  - Status report forms
- TaskJuggler can create lists in different formats
  - HTML
  - CSV
  - TaskJuggler Syntax
  - iCal

Lists and Reports

- TaskJuggler supports differnt types of lists and reports
  - Task lists
  - Employee and resource lists
  - Calendar
  - Time sheet forms
  - Status report forms
- TaskJuggler can create lists in different formats
  - HTML
  - CSV
  - TaskJuggler Syntax
  - iCal

Task Lists

```
taskreport "Tasks" {
  formats html
  hidetask ~isleaf()
  sorttasks plan.end.up
}
```

Employee Lists

```
resourcereport "Employees" {
  formats html
  sortasks plan.id.up
  columns no, name, email
}
```

Task lists with employees

```
taskreport "Tasks" {
  formats html
  hidetask ~isleaf()
  sortasks plan.effort.up
  hideresource 0
  columns no, name, weekly
}
```

Text Reports

- Text reports consists of up to five customizable text fields
- header, left, center, right, bottom

```
textreport "Report" {
  formats html
  left "Left Margin"
  center "Center Area"
  right "Right Margin"
}
```

Text Reports

- Text reports consists of up to five customizable text fields
- header, left, center, right, bottom

```
textreport "Report" {
  formats html
  left "Left Margin"
  center "Center Area"
  right "Right Margin"
}
```

RichText Markup

- Most text attributes are interpreted as *RichText* markup
- TaskJuggler uses a subset of MediaWiki's markup …
- … and extends this subset for
  - text colors,

    `<fcol:green>Green</fcol>`
  - navigation menus,

    `<[navigator id='my_menu']>`
  - value queries and

    `<-query ...->`
  - inline reports

    `<[report id='my_report']>`

Compound Reports

```
taskreport r1 "" {
  columns name, chart
}
resourcereport r2 "" {
  columns name, phone
}
textreport "Report" {
  formats html
  left "<[report id=’r1’]>"
  right "<[report id=’r2’]>"
}
```

HTML Navigation Menus

```
navigator menu
textreport "" {
  header "<[navigator id='menu']>"
  formats html
  taskreport "Tasks" {
    columns name, start, end, chart
    hideresource 0
  }
  resourcereport "Employees" {
    columns name, email
  }
  purge formats
}
```

Project Management Cycle

After starting the project, the following tasks should be re-run on a regularly:

- Ask for done workload for current tasks
- Ask for workload or time left for current tasks
- Update project description based on this data
- Freeze project history

timesheets and statussheets

- Report of done and left workload and task times can be organized semi-automatically with timesheetreports
- Hierarchical status reports be organized semi-automatically with statussheetreports
- This topic is too much complex for this slides but is documented well in the user documentation.

Freeze the Project History

- After upgrading the project description with the reported data, the project history should be friezed.

  ```
  tj3 --freeze yourproject.tjp
  ```

- This will create or update the following two files:
  - `yourproject-header.tjp`
  - `yourproject-bookings.tjp`
- These files must be included to the project with `include`

Freeze the Project History

- After upgrading the project description with the reported data, the project history should be friezed.

  `tj3 --freeze yourproject.tjp`
- This will create or update the following two files:
  - `yourproject-header.tjp`
  - `yourproject-bookings.tjp`
- These files must be included to the project with `include`

Freeze the Project History

- After upgrading the project description with the reported data, the project history should be friezed.

  ```
  tj3 --freeze yourproject.tjp
  ```

- This will create or update the following two files:
  - `yourproject-header.tjp`
  - `yourproject-bookings.tjp`
- These files must be included to the project with `include`

Track Key Values with `tracereport`

- With `tracereports` you can track important key values over the whole projects lifetime.
- Key values are exported to CSV explicitly on demand
- `tj3 –add-trace yourproject.tjp`
- Can be used to create burndown charts
- `tracereports` is the main feature of the upcoming version (3.2)

Follow Up Links

- TaskJuggler on the web: `http://taskjuggler.org`
- Online Documentation: `http://taskjuggler.org/tj3/manual/`
- Questions?

Follow Up Links

- TaskJuggler on the web: `http://taskjuggler.org`
- Online Documentation: `http://taskjuggler.org/tj3/manual/`

- Questions?

Follow Up Links

- TaskJuggler on the web: `http://taskjuggler.org`
- Online Documentation: `http://taskjuggler.org/tj3/manual/`
- Questions?