Work Diary

Jesper Kjær Nielsen
Aalborg University
Bang & Olufsen

Last compiled: September 13, 2014

Contents

Todo list

Part I

2014

Chapter 1

March

1.1 March 15, 2014

1.1.1 Aim and Usage of the Work Diary

Tag(s):
User guide (userguide)
Tasks (tasks)
Tags (tags)

Author(s):
Jesper Kjær Nielsen (jkn)

File(s):
`addTask.py`
`default.tpl.tex`
`newTag.py`
`changeTag.py`
`newAuthor.py`
`changeAuthor.py`
`newBuild.py`

How often have you returned to an old task wondering what you were thinking the last time you worked on it? Did I really write this piece of code? How did I connect the DUT to the measurement equipment? What did I search for in Google when I found that interesting paper? What did I name that file? Where did I place that piece of paper with the magnificent derivation? Unfortunately, I am guilty of asking myself these and many other questions way too often. Yes, I tend to forget things. Even important things. Things which I am sure that the future me will remember, but he often disappoints me.

I am also guilty of being a happy LaTeX 2ε user [1, 2, 3]. And friendly - to myself at least. Therefore, I decided to create the present LaTeX 2ε-based work diary to the forgetful future me. The diary resembles a good old lab journal, but it is hopefully also much more powerful while still being fairly simple to use. The focus should be on the work - not on managing a diary - because the current me hates bureaucracy.

Sure sure, that all sounds great and all, but how do you manage the diary in two years from now when it consists of 1000 pages? What about in 10 years? Can you write confidential information in the diary and still share your work with an external partner? Yes, you can! Thanks to a tagging system, the diary can be build conditionally on these tags or on the time. If I remember it, I will say much more about this later.

The work diary can also be used as a collaborative tool for multiple forgetful people working on the same project. When used in this way, the diary is a project diary which serves as the documentation of the project.

All right. Enough introduction. Now let us discuss how you actually use this diary.

An Example

Both the current and future me like to learn through examples so therefore I better start with one. One cold morning, Jon Snow shows up at work at Castle Black. He turns on his computer while he tries to recall what he was supposed to do this morning before the pigeon pie is served for lunch. It annoys him since he often finds himself forgetting things. Suddenly he remembers that interesting diary template he accidentally stumbled upon yesterday evening while he was searching for methods to work smarter and more efficiently. Jon quickly looks at the photo of Ygritte next to his screen and decides to try this template. He downloads it and copies the `diary` folder in the zip-archive to the location `c:\path\to\the\diary`.

"Literature review", he finally recalls, "I have to look into how people build and maintain a really tall wall. My partner, Stark Industries, is really interested in this *Tall Wall* project." Jon takes a sip of his Westerosi coffee, flexes his sword hand, hits the Windows button, writes `cmd` in the search field to launch the terminal, and changes the current directory to the diary folder while he silently complains about that the Night's Watch IT policy forces him to use Windows. For a moment he looks angry at the blinking cursor next to the path

```
c:\path\to\the\diary>
```

Then he writes

```
python addTask.py "tallWall, stark" "Literature Review on Building Tall Walls" js
```

Since this is the first time Jon adds an entry to the diary, he is prompted for tag titles for the unknown `tallWall` and `stark` tags. He writes `The Tall Wall Project` and `Stark Industries`. Moreover, this is also the first time that he has provided the optional author initials `js` after the `addTask.py` Python script so he also fills in his name and email address. Immediately after filling in the email address, his favourite LaTeX 2ε editor is launched displaying a file with the following content.

```
% mainfile: ../../../../master.tex
\subsection{Literature Review on Building Tall Walls}
\label{task:20140315_js0}
\tags{tallWall, stark}
\authors{js}
%\files{}
%\persons{}
```

"Great", he thinks and starts reading and typing in all his findings and sources on how to build and maintain a really tall wall. A few hours later he has finished his literature review - and his stomach is rumbling, but before he goes to lunch he wants to compile the diary. To do that, Jon first writes in the terminal

```
python newBuild.py all
```

to create all the necessary build files for LaTeX 2ε to assemble the diary. Then he compiles the diary as he would normally compile a LaTeX 2ε document.

The above was a simple little example of how the diary can be used. Below you will find the much more elaborate and boring documentation.

Organisation

The work diary is basically a book of ever increasing size and organised by time. In this book, a year is a part, a month is a chapter, and a day is a section. Each day is comprised of a number of tasks. These tasks are all you should care about, because LaTeX 2ε handles the rest.

Tasks

So what do you need to know about these tasks? Well, a new task is created by running the following Python script from a command line[1].

```
python addTask.py "tallWall, stark" "Literature Review on Building Tall Walls"
```

This automatically creates a new task in a separate file with the title *Literature Review on Building Tall Walls* and the tags `tallWall` and `stark`. The file is automatically placed in the `entries` folders which is organised in a year, month, and day structure. Moreover, the newly created file is opened in your default LaTeX 2ε editor. The Python script `addTask.py` takes five input variables of which three are optional.

"tagA, tagB, tagC" The first input variable is required and contains a comma separated list of valid tags. I will say more about these tags later.

"Task Title" The second input variable is also required and is the title of the task.

author initials The third input variable is optional and specifies the author initials. If not set or set to `""`, no author is used. This will be described in more detail later.

template The fourth input variable is also optional and generates the task from a particular template such as a meeting or an agenda. If not set or set to `""`, the default template is used. The creation of new templates is described later.

YYYY-MM-DD The fifth input variable is also an optional input variable which specifies the date of the task. If not set, today is used as the date.

If you actually ran the Python file above with the same input parameters, a new file should open in your default LaTeX 2ε editor with the following content.

```
% mainfile: ../../../../master.tex
\subsection{Literature Review on Building Tall Walls}
\label{task:20140315_0}
\tags{tallWall, stark}
%\authors{}
```

[1] On unix-like systems, you can write the shorter `./addTask.py` instead of `python addTask.py`. Moreover, you can also use single quotes instead of double quotes.

```
%\files{}
%\persons{}
```

The first line enables synctex forward and inverse search support for my editor of choice (gedit) and possible also for other editors. If your editor does not need this line, it can be safely ignored or deleted in the template file `default.tpl.tex` located in the `template folder`. The second line simply prints the task title as a subsection. The third line is a unique label of the task so that is can be referred to from other tasks. The fourth line in the opened file contains the comma separated list of tags which will be described later. The fifth, sixth, and seventh lines are commented out by default, but they can be used to specify the authors of the task entry, which files/folders are created or changed in the task, and which persons are also involved in the task. LaTeX 2_ε automatically generates separate indices in the work diary from the list of authors, files, and persons. These indices can be used to quickly find who wrote which entries and where in the diary you worked on a particular file or had a meeting with a particular person.

Tags

Aside from being based on LaTeX 2_ε, the tag list is really the most compelling feature of the present work diary. It provides you with two powerful possibilities.

- An index is generated for the tags so that tasks labelled with one particular tag can quickly be found in the work diary.
- The work diary can be compiled conditioned on just one or a few tags. Suppose, for example, that you work on various projects with multiple external partners. If one of these partners (partner A, say) wants to see what you have been doing on the project, you probably do not want to send them the entire diary. Instead you can compile your work diary only with the tasks labelled by the `partnerA` tag. Another usage of the tag-dependent compilation is that you can quickly get an overview over previous work labelled with the same tag. I will say more about conditional compilation later.

Before a tag can be used, it must be defined and given a title. This is easily done by running the following python script.

```
python newTag.py tag "Title␣of␣the␣tag"
```

The tag should be a short id which can only contain letters (a-zA-Z) and numbers (0-9). The tag title is a longer description. Both the tag and its title is shown in the margin next to a task. Moreover, they can also be found in the index. Tags can also be renamed by running the following python script.

```
python changeTag.py oldTag newTag "New␣title␣of␣the␣tag"
```

When a tag is renamed, the entire diary is updated with the new name.

Conditional Compilation

So how do you compile the work diary for just one or a few tags? You simple run the following python script.

```
python newBuild.py "tagA,tagB" "dateA,dateB" "taskLabel1,taskLabel2"
```

When this python script is run, a build file and a tag dictionary are made so that the next time LaTeX 2ε compiles the diary, only the relevant tasks are kept. If no input parameters are given, the default build file contains every task from the last 90 days. If you want to compile the diary for just 2014, you could write

```
python newBuild.py all 2014
```

If you want to compile the diary for just March and April 2014, you could write

```
python newBuild.py all "2014-03, 2014-04"
```

If you want to compile the diary for only some specific task labels, you can write

```
python newBuild.py all all "20140315_0,20140316_0"
```

And so on. Aside from specifying dates and tags, you can also write

```
python newBuild.py all all
```

to include everything in your diary. A few years from now, however, compiling the diary for this build profile might be a time consuming process.

You can also exclude particular tags by preceding the tag name by an exclamation mark. If you for example have defined a tag with the tage name name confidential, you can compile a non-confidential version of your diary by writing

```
python newBuild.py "all,!confidential" all
```

Multiple Authors

When multiple authors are working simultaneously on the same diary, every new task should have at least one author attached to it. This makes it easier for the authors to see which person completed which task and is also essential in order to avoid file name conflicts. A task by the author `js` can be created as

```
python addTask.py "tagA,tagB" "Task title" js
```

Before an author can be attached to a task, he or she must be in the author database. To add the author (e.g., Jon Snow (js@nightswatch.wes) with initials js) to the author database, simply run

```
python addAuthor.py js "Jon Snow" "js@nightswatch.wes"
```

An author can also be changed by running

```
python changeAuthor.py oldInitials newInitials "New name" "New email address"
```

The last two options are optional. If not provided, only the initials will be updated.

New templates

New templates can also easily be added by creating them in the `template` folder. A new template such as a meeting agenda can be added by using the *default* template as a starting point. A new template should have the file name `<someName>.tpl.tex`. A new task can be created from a template by writing, e.g.,

```
python addTask.py "tagA,tagB" "Task␣title" "" myTemplate
```

This will create a task based on the template `myTemplate.tpl.tex`. When you create a new template, make sure that it is saved with UTF-8 encoding. By default, all Python script assumes that this is the case.

System Requirements

You need to have a LaTeX 2ε distribution and Python installed. The diary has been developed and tested on a 64 bit Ubuntu 12.04.4-12.04.5 system with the versions of Python (2.7.3 and 3.2.3) and TexLive (2009-15) from the repositories installed. Moreover, all unit tests also pass under a 32 bit Windows 7 with TexLive 2014, Python 2.7.8, and Python 3.4.1 installed (remember to add Python to the path when you install it). I do not have a Mac so I have not tested the diary on OSX. By default, the various operating systems use different encoding for standard text files such as `.tex` files. To avoid problems with special characters such as the Danish æ,ø, and å, I have therefore decided that all files are created and read with the UTF-8 encoding. I you observe any strange problems when trying to compile the diary, a file saved in the wrong encoding might be the cause.

Compiling the Diary

When I compile the diary, I run the following commands.

```
pdflatex -shell-escape -file-line-error -synctex=1 master.tex
bibtex master.aux
splitindex master.idx
pdflatex -shell-escape -file-line-error -synctex=1 master.tex
pdflatex -shell-escape -file-line-error -synctex=1 master.tex
```

You can find these commands in the Makefile which you can use on Linux.

Anything else?

Nope! Or not that I can think of at the moment. To start your own work/project diary, just copy the contest of the diary folder in the zip archive you have downloaded. As I claimed earlier, the diary should be very simple to use. If you lack any functionality, you are very welcome to change the code as you see fit.

1.1.2 Initial Code Base

Tag(s):
Development (development)

Author(s):
Jesper Kjær Nielsen (jkn)

The current version of the work diary is based on my own version which is tailored to my set up. Therefore, various shell scripts must be rewritten in Python. Moreover, some of the features described above must be implemented and tested. Finally, many small refinements must be made.

1.2 March 16, 2014

1.2.1 Automatic Generation of Tasks from Templates

Tag(s):
Development (development)
Tasks (tasks)

Author(s):
Jesper Kjær Nielsen (jkn)

File(s):
`addTask.py`

As described in yesterday's user guide, a Python script should be written for generating and opening a new task in the user's default LaTeX 2ε editor. The script should do the following.

1. The input parameters must be validated and, if not provided, the optional input parameters must be set to their default values. The default is today for the date, empty for the author, and

```
% mainfile: ../../../../master.tex
\subsection{<task title>}
\label{task:YYYYMMDD_IIIX}
\tags{<tags>}
%\authors{}
%\files{}
%\persons{}
```

   for the template. The provided tags must be compared to the tag database. If not found, the user must be asked whether he/she wants to create it.

2. The missing year, month, and/or day directories must be created in the `entries` folder.

3. The task index corresponding to the date must be found and a file name must be generated. The structure of a file name for a task is `YYYYMMDD_IIIX.tex` where `III` are the author initials (if any) and `X` is the task index. The author initials is added to the file name to prevent a possible conflict when multiple authors are working on the same diary.

4. The task file must be created and stored from the specified template.

5. The current active build configuration (if any) must be updated.

6. The created file must be opened in the default LaTeX 2ε editor.

I have implemented everything in `addTask.py` with the exception of tag validation and author validations. This functionality will be implemented when the Python scripts for creating new tags and new authors have been written.

1.2.2 Adding Tags

Tag(s):
Development (development)
Tags (tags)

Author(s):
Jesper Kjær Nielsen (jkn)

File(s):
`newTag.py`

The user guide also describes how tags should be added using a Python script. The script should do the following.

1. The input variables should be validated. The tag can only consist of letters (a-zA-Z) and numbers (0-9). Moreover, if the tag has already been defined, an error must be given.

2. The tag and the tag list must be stored in a cross-platform database. An SQLite database looks like a simple and cross platform solution for the database.

1.3 March 20, 2014

1.3.1 SQLite Database

Tag(s):
Development (development)
Database (database)
Tags (tags)

Author(s):
Jesper Kjær Nielsen (jkn)

File(s):
`logic/DiaryDatabaseWrapper.py`

I have started implementing a number of functions for creating, adding, retrieving, updating, and deleting tags and authors. Only a few of the functions have been written, and I have not tested anything yet.

1.4 March 22, 2014

1.4.1 SQLite 3 Database Wrapper

Tag(s):
Development (development)
Database (database)

Author(s):
Jesper Kjær Nielsen (jkn)

File(s):
`logic/DiaryDatabaseWrapper.py`

I have finished writing the simple database wrapper class to interface with the SQLite 3 database. Functionality such as inserting, updating, selecting, and deleting entries in the database have been implemented and tested.

1.5 March 23, 2014

1.5.1 Adding Tags

Tag(s):
Development (development)
Tags (tags)
Database (database)

Author(s):
Jesper Kjær Nielsen (jkn)

File(s):
`newTag.py`
`newTask.py`

I have now implemented the functionality for adding a tag to the database as described in task 1.2.2. Moreover, I have updated the Python script for adding the new tasks so that the provided tag(s) is/are validated against the diary database. If a provided tag is not in the diary database, the user is offered to add this tag to the database.

1.6 March 24, 2014

1.6.1 Creating a New Build (Conditional Compilation)

Tag(s):
Development (development)
Conditional Compilation (building)

Author(s):
Jesper Kjær Nielsen (jkn)

File(s):
`newBuild.py`

As described in the user guide in task 1.1.1, it should be possible to compile the diary only for specific tags and dates. The Python script `newBuild.py` should handle this by doing the following.

1. The input parameters must be validated and default values for the non-specified parameters must be set.
    - If no parameters are given, the default build of all tasks within the last 90 days should be selected.
    - If only tags are given, tasks labelled with these tags from all time should be selected.
    - If only dates are selected, all tags and tasks from these dates should be selected.
2. Scan all files in the entries directory satisfying the selected dates. Create a list of those tasks which are labelled with the selected tags and/or task labels. Moreover, retrieve all other tags as well.
3. From the list of extracted tags, create the tag dictionary in the `buildFiles` folder.
4. From the list of selected tasks, create the task list file in the `buildFiles` folder. In addition to including the filenames of the paths, the task list file must also format years, months, and days as parts, chapters, and sections.

1.7 March 29, 2014

1.7.1 Creating a New Build (Conditional Compilation)

Tag(s):
Development (development)
Conditional Compilation (building)

Author(s):
Jesper Kjær Nielsen (jkn)

File(s):
`newBuild.py`

I have started implementing input validation in the function for creating new build profiles. The functionality of this function is described in task 1.6.1

Chapter 2

April

2.1 April 2, 2014

2.1.1 Creating a New Build (Conditional Compilation)

I have finalised the input validation and the initialisation of the default values for those input parameters which have not been set. This is the first point described in task 1.6.1.

Tag(s):
Development (development)
Conditional Compilation (building)

Author(s):
Jesper Kjær Nielsen (jkn)

File(s):
`newBuild.py`

2.2 April 4, 2014

2.2.1 Creating a New Build (Conditional Compilation)

The script `newBuild.py` has now been implemented as described in task 1.6.1, except for building conditioned on task labels. The list of tag names and tag titles are written to the file `tagDictionary.tex` in the `buildFiles` folder. The list of valid tasks is written to the file `taskList.tex` also in the `buildFiles` folder. Notice that the tags has been defined in a rather strange way to ensure that numbers can be used in the tag names.

Tag(s):
Development (development)
Conditional Compilation (building)

Author(s):
Jesper Kjær Nielsen (jkn)

File(s):
`newBuild.py`
`buildFiles/tagDictionary.tex`
`buildFiles/taskList.tex`

Chapter 3

August

3.1 August 16, 2014

3.1.1 Excluding Tags in a Build (Conditional Compilation)

I have implemented conditional compilation for the excluding tags. An excluding tag is preceded by an exclamation mark such as in this example

```
1 python newBuild.py "all,!confidential" all
```

It can be used to compile the diary for all entries except for one or more tags.

Tag(s):
Development (development)
Conditional Compilation (building)
Tags (tags)

Author(s):
Jesper Kjær Nielsen (jkn)

File(s):
newBuild.py

3.2 August 17, 2014

3.2.1 Conditional Compilation for Particular Task Labels

I have implemented conditional compilation for task labels. Thus, the diary can now be compiled for only a few tasks by running

```
1 python newBuild.py all all "20140315_1,20140817_0"
```

Tag(s):
Conditional Compilation (building)
Development (development)

Author(s):
Jesper Kjær Nielsen (jkn)

File(s):
newBuild.py

3.3 August 18, 2014

3.3.1 Adding Authors

I have written the function for adding a new author in the author database. The next step is to include the author information in the diary as well.

Tag(s):
Development (development)
Authors (author)

Author(s):
Jesper Kjær Nielsen (jkn)

File(s):
newAuthor.py

3.4 August 20, 2014

3.4.1 Including Author Information in the Diary

Tag(s):
Development (development)
Authors (author)

Author(s):
Jesper Kjær Nielsen (jkn)

File(s):
addTask.py

The Python script for adding a task has been updated so that author initials are now validated against the author initials in the author database. Moreover, multiple author initials can now be specified as a comma separated list from the command line when a task is created. However, only the first author initials is used in the file name and the task label.

3.5 August 22, 2014

3.5.1 Building an Author Dictionary

Tag(s):
Development (development)
Authors (author)

Author(s):
Jesper Kjær Nielsen (jkn)

File(s):
newBuild.py
authorDictionary.tex

The author dictionary is now build and placed in the `buildFiles` folder. Moreover, the author names are now shown in the margin of the diary and in the index. If an author name is clicked, an email can be written to him or her.

3.6 August 24, 2014

3.6.1 Changing Tags

Tag(s):
Development (development)
Tags (tags)

Author(s):
Jesper Kjær Nielsen (jkn)

File(s):
changeTag.py

The tag name or title can be changed by running the command

```
1 python changeTag.py oldTag newTag "New␣title␣of␣the␣tag"
```

The Python script for doing this should do the following.

1. The input parameters must be validated and, if not provided, the optional input parameter (the tag title) must be set to its default value (the old tag title). The old tag should be looked up in the database. If not found, the script should tell the user and abort.

2. The database should be updated with the new tag and the new tag title. If a new tag title is not set, the old tag title is kept.

3. All diary tasks must be scanned. If the old tag is found, it must be replaced by the new tag. The step can be omitted if the old and new tags are identical (only the tag title is changed).

3.7 August 26, 2014

3.7.1 Changing Tags and Authors

Tag(s):
Development (development)
Tags (tags)
Authors (author)

Author(s):
Jesper Kjær Nielsen (jkn)

File(s):
changeTag.py
changeAuthor.py

I have written the Python scripts for changing an author or an tag. The implementations are based on the description in task 3.6.1.

3.8 August 27, 2014

3.8.1 Truncating Long File Names

Some file names are so long that they cannot fit in the margin. To avoid that the file name is overlapping the text, I have included the truncate package in the preamble so that too long file names are truncated.

Tag(s):
Development (development)

Author(s):
Jesper Kjær Nielsen (jkn)

File(s):
aReallyReallyReallyReallyRe ...

Chapter 4

September

4.1 September 1, 2014

4.1.1 Python 3 Compatibility

Tag(s):
Development (development)

Author(s):
Jesper Kjær Nielsen (jkn)

File(s):
addTask.py
newBuild.py
newTag.py
newAuthor.py
changeTag.py
changeAuthor.py

I have updated all the Python scripts so that they now work with Python 2.X and 3.X. I had to change three things to make all the scripts work with Python 3.X.

- The print statement has been changed from

```
1 print "Some text to print."
```

  to

```
1 print("Some text to print.")
```

- All strings are stored in UTF-8 in Python 3 so the encoding and decoding, which is used in Python 2 to convert between byte strings and unicode strings, is not necessary in Python 3.
- The `raw_input()` function is changed to `input()` in Python 3.

4.2 September 3, 2014

4.2.1 Python 3 and Windows Compatibility

Tag(s):
Development (development)

Author(s):
Jesper Kjær Nielsen (jkn)

File(s):
newBuild.py
addTask.py
changeAuthor.py
changeTag.py

I have discovered that many of the problems that I observed with cross-platform compatibility were causes by different encoding of files. Therefore, I have decided to store and read all files with the UTF-8 encoding using the `io` module. In addition to updating the Python scripts with this, I have also updated the preamble and the user guide with a warning about changing the encoding.

4.3 September 5, 2014

4.3.1 Testing the Diary on Windows and OSX

Tag(s):
Testing the Diary (testing)

Author(s):
Jesper Kjær Nielsen (jkn)

Person(s):
Martin Weiss Hansen
Christer Peter Volk
Nicolai Bæk Thomsen

I have emailed the diary Martin, Christer, and Nicolai and asked them to test it thoroughtly under Windows and OSX.

Bibliography

[1] Lars Madsen. *Introduktion til LaTeX*. http://www.imf.au.dk/system/latex/bog/. 2010.

[2] Frank Mittelbach. *The LATEX companion*. 2. ed. Addison-Wesley, 2005.

[3] Tobias Oetiker. *The Not So Short A Introduction to LaTeX2e*. http://tobi.oetiker.ch/lshort/lshort.pdf. 2010.

People Index

File Index

Author Index

Tag Index