Question Data Base

Johan Arvelius

March 26, 2014
Revision 208

Contents

1 Introduction

Question Data Base (QDB) is a format to write questions for typesetting in LaTeX. Each question is written in one file and marked up with special commands to include

- full solutions,
- short answers,

- which course it has been used in,
- number of points in exam,
- teachers notes for correction criterias,
- references to a course book where answers are to be found and
- indexing keywords.

The questions may then be used to make collections of questions for studying or exams. Several versions of the same collections are generated simultainously e.g. separate questions answers and solution collections or versions with questions and solutions mixed. QDB generates LaTeX-files for those versions to be further processed to get typeset results. A flow chart for the process is shown in figure 1

For questions that requre only simple answers like a calculated value or a single word an auto correcting web-form can be generated as a perl CGI-script.

A simple randomization scheme is included that can pick a random subset among questions or parts of a question, a set of values for parameters in the question or a random subset of words to match up with their descriptions in scrambled order.

The core part of the system is the program qdbpp (Question DataBase Pre-Processor) that parses the LaTeX-file with the questionaire and includes gather input question-files to produce the different versions of the document.

Three more accompanying perl scripts are provided: qdbbatch to read a list of names and create individual exams to each person in the list, qdbsort to extract and sort a collection of questions from the database according to given criteria and a cgi script called qdbcgi to run a an automatic correcting web questionnair in a webserver.

2 Installation and setup

To install the qdb collection check out a copy of the svn repository provided from `http://kyla.kiruna.se/qdb/trunk`. The programs qdbpp, qdbbatch and qdbsort needs to be executable. They can be called by full or relative path at all time or copied or linked to a directory in your path.

To set up a new questions collection the following layout is recommended. In one new directory create three subdirectories:

- A working directory with arbitrary name where you can put the document files. This directory will be filled with qdbs intermediate files as well as the resulting pdf or postscript files. To this directory a file `settings.tex` shall be copied which will be input to all qdb files with the `%(PREAMBLE)` markup. The example settings files can be copied from e. g. `astronomi/instudering/settings.tex` for documents in the planning style and `astronomi/prov/settings.tex` for documents in the exam style.

fragor to contain the question files. The directory to look for questions is set by default to `../fragor` in qdbpp but may be overridden by the **-d** flag or `%(DIRECTORIES)` markup.

figures to contain all graphics used in the questions. The graphicspath is set by the `%(GRAPHICSPATH)` markup preferrably given in the `settings.tex` file.

For use of qdbsort another directory parallel to fragor with its own `settings.tex` file is recommended.

2.1 Webforms

To make use of the webform feature you'll need a webserver that allows you to run cgi scripts. The cgi script qdbcgi needs to be moved or linked to a directory where the webserver can run the script such as `~/public_html/cgi-bin/`. In the same mother directory where the cgi directory is two new directories must be created:

qdb-answers which is the directory where the answers from the students are going to end up. This directory needs writing permissions to the webserver process. E. g. by giving writing permissions to group www-data for a standard installation of apache.

qdb-questions is the directory to contain the `.web` file from a run of qdbpp. The path to this file is given with the **-w** switch to qdbpp during the preprocessor run. The directory needs only read permissons to the webserver process (but of course write process to the user running qdbpp).

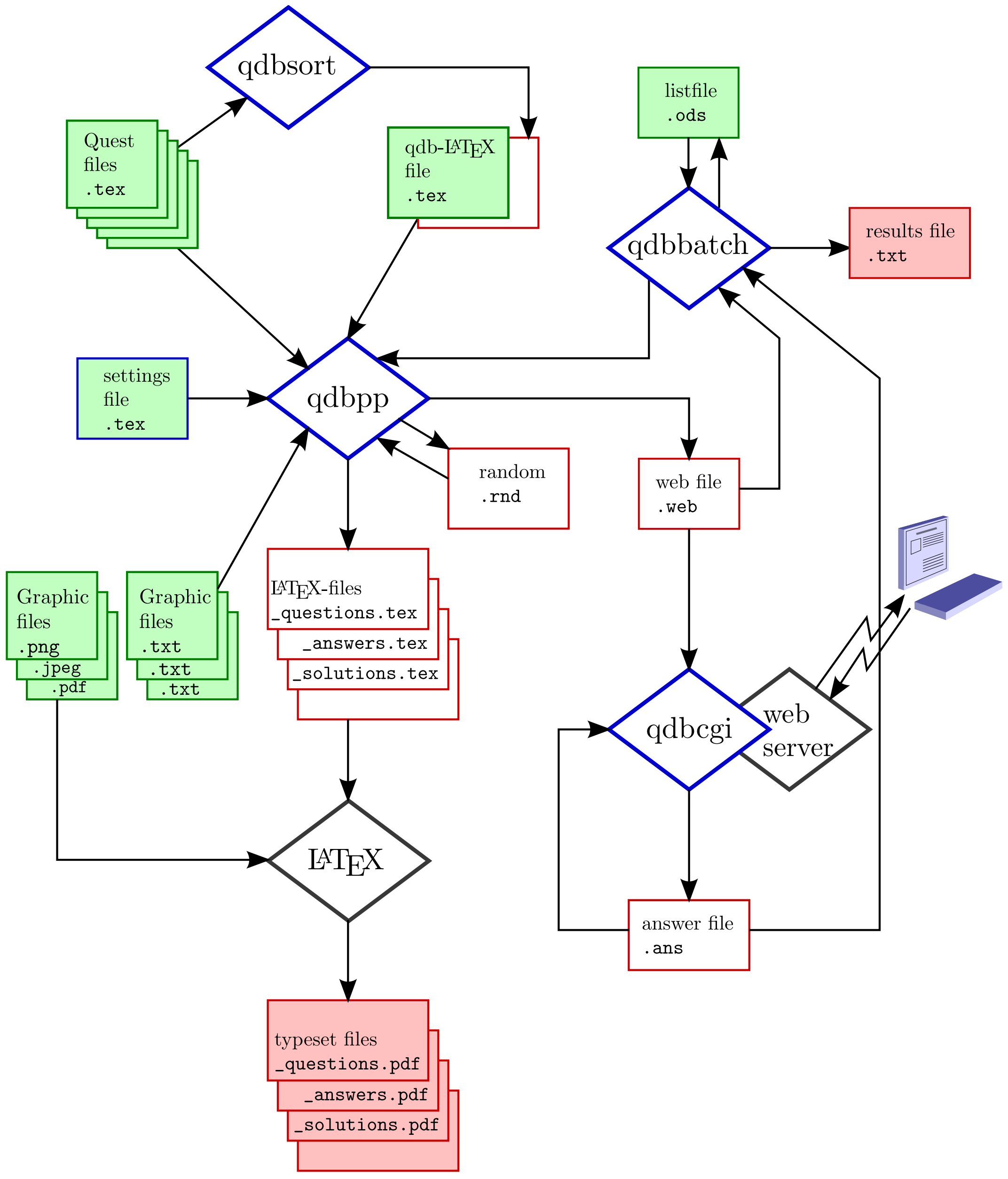
qdbsort
Quest files .tex
qdb-LaTeX file .tex
listfile .ods
qdbbatch
results file .txt
settings file .tex
qdbpp
random .rnd
web file .web
Graphic files .png .jpeg .pdf
Graphic files .txt .txt .txt
LaTeX-files _questions.tex _answers.tex _solutions.tex
qdbcgi
web server
LaTeX
answer file .ans
typeset files _questions.pdf _answers.pdf _solutions.pdf

Figure 1: Flow chart for a QDB-project. Diamonds are programs and rectangles are other files. Files with green borders are user-provided, blue borders are provided in QDB and red borders are auto-generated in the process. Files with green backgrounds are thought to be edited by user and files with pink backgrounds are end products while those with white backgrounds are intermediate files. The LaTeX diamond is just a representation of the whole LaTeX system which may be pdflatex, bibtex, xindy and whatever else needed.

2.2 Dependencies

There are certainly many dependencies on webserver, LaTeX-installation and perl libraries to get the system to run, which are not investigated. The requirement on perl itself it version ≥ 5.10. The perl modules needed to be installed especially for the running of qdb on an Ubuntu lucid lynx system has been found to be:

- Set/IntSpan.pm in package libset-intspan-perl
- Data/Random.pm in package libdata-random-perl
- String/Random.pm in package libstring-random-perl
- AppConfig.pm in package libappconfig-perl
- package perl-doc

The system is written for a posix environment and no special efforts to make the code portable to other platforms has been made. It is reported successfully running in perl 5.14 under OS X.

3 qdbpp

Preprocessor to make LaTeX-files of question collections and exams containing questions from database in special format for this preprocessor.

SYNOPSIS

qdbpp [–**auxfiles** *citekey=filename*] [–**commandsfile**=*path/to/file*] [**-c**|–**configdirs**=*directory1[,directory2[,...]]*] [–[**no-**]**decimalcomma**] [**-d**|–**directories**=*directory1[,directory2[,...]]*] [**-e**|–**extra**=*string*] [**-o**|–**output**=*filebasename*] [**-q**|–**quiet**] [**-r**|–**randomize**] [**-l**|–[**no-**]**restlist**] [–**solutioncol**=*colourarg*] [–**sourcedirectories**=*directory1[,directory2[,...]]*] [–**sourcesuffixes**=*suffix1[,suffix2[,...]]*] [**-s**|–**suffixes**=*suffix1[,suffix2[,...]]*] [–**texreferences** *citekey=bibtexcitekey*] [–**urlbase**=*URL*] [**-v**|–**verbose**] [–[**no-**]**warning**] [**-w**|–**web**[=*webdir*]] sourcefile

qdbpp **-?**|–**help**

qdbpp –**man**

DESCRIPTION

qdbpp is a preprocessor in a system to build a database of questions for student each question in one file and use them in LaTeX documents by inserting special preprocessor commands formatted as LaTeX-comments in the *.tex* file. LaTeX ready *.tex* files are built all at once, typically a question version, one file with answers, one with solutions and a teacher version contining extra information like comments and filenames to the problems. There are different types of question files that shall be called by different command in the source file, in some cases similar question files may be called by different commands for example to typeset an example with inline solution inbetween exercises.

The same questions can be used in different types of documents like exams and study material using different profiles. Each profile to be accompanied by a settings file that loads the right LaTeX class, loads necessary modules and defines LaTeX commands used in the LaTeX code generated by the profile. Three such profiles are provided:

exam

Used for exams provides number of points per question and gives a solution and a teachers version for correction of exams.

planning

Used for study material. Answers and solution files are provided as raw LaTeX code to be inserted in the end of the document with hyperlinks from questions to the solutions part and back.

collection

Used for collection of questions e.g. sorted out by qdbsort. This profile provides only one outputfile *filebasename_questions.tex* in a manner rather similar to the *filebasename_teachers.tex* file in the exam profile, i.e. question followed by solution and comments. The format is adopted to the purpose of making teachers collections of questions.

The exam profile is also used to generate *.web* files for webforms provided with the qdbcgi cgi script. All questions included in a file using the webform must exclusively provide answers using `%(WEB_VALUE)` or `%(WEB_VALUE)` markup. see the package documentation for the sytax of those.

A typical use cycle:

Prepair a directory with questions *questions*

Prepair an exam file *exam.tex*

`cqdbpp -d questions exam.tex`

Outputs e.g. *exam_questions.tex*, *exam_solutions.tex*, *exam_answers.tex*, *exam_teachers.tex*

`latex exam_questions.tex`

`latex exam_teachers.tex`

`bibtex exam_teachers`

Bibtex teachers file as it ususally contains references to where in a course book to find aswers to the question. Other files ofcourse needs bibtex if the sourcefile contains citations.

`latex exam_teachers.tex`

...

The output files may be free-standing LaTeX documents to run typically questions and teachers versions or raw LaTeX code output of just the inclusion to be inserted by a LaTeX `\insert` command, typically answers part.

Apart from the latex files the qdbpp also produces a *.rnd*-file that stores the random output for all randomization processes that occur during the run. This file is read in if present (unless **-r** switch is given) to ensure reproducable output.

Also a *.dep* file is produced. This file includes a list of all files that the job is dependent on. The files in the list is written with searchpath to where they were found by the preprocessor. This file is not used by qdbpp but merely an extra information that can be used e.g. to make a tarball of the sources.

OPTIONS

–auxfiles *citekey=filename*

For crossreferencing, provide filename to .aux file to lookup reference labels for document *citekey*

–commandsfile

Filename of commands file. Default ../../qdb/commands.tex

–debug

Print debug messages. Useless.

-c *directory1[,directory2[,...]]*, –configdirs=*directory1[,directory2[,...]]*

Directories to search for config files

–[no-]decimalcomma

Set output of floatingpoint numers using decimal comma (Swedish style), rather than decimal point. Default is TRUE!

-d *directory1[,directory2[,...]]*, –directories=*directory1[,directory2[,...]]*

Directories to search for questions.

-e *string*, –extra=*string*

String to be inserted by variable substitution in sourcefile. The option may be given several times and the strings are available as \#1, \#2 etc. for the qdb sourcefile in the order they are specified.

-h, –help

Print help message.

–man

Print the whole manpage.

–sourcesuffixes=*suffix1[,suffix2[,...]]*

Suffixes for meta files containing copyright and licencing data to illustrations to look for in priority order. First match for each question will be included. Default is *.txt*.

-o *filebasename*, –output=*filebasename*

Use *filebasename* instead of source file name to generate output filenames e.g. *filebasename_questions.tex*.

-q, –quiet

Don't print messages to STDOUT.

-r, –randomize

Do a new randomization and create a new .rnd file even if one is present since earlier.

-l, –[no-]restlist

Toggles printing of a list of questions where no corresponding file is found to STDOUT. Default is on.

–solutioncol=*colourarg*

Sets the colour for the text of example solutions and solutions included in teachers version to questions marked with –nosolution. *colourarg* should be in the form for the \definecolor command of the latex color package. Default is {rgb}{0,0,0.6}

–sourcedirectories=*directory1[,directory2[,...]]*

Directories to search for source code for programming tasks.

–sourcesuffixes=*suffix1[,suffix2[,...]]*

Suffixes for source code files to programming tasks to look for in priority order. First match for each question will be included. Default is *.cpp*.

-s *suffix1[,suffix2[,...]]*, –suffixes=*suffix1[,suffix2[,...]]*

Suffixes for question files to look for in priority order. First match for each question will be included. Default is *.tex*.

–texreferences *citekey=bibtexcitekey*

Exchange TEXREF *citekey* with BibTeX *bibtexcitekey* in TEXREF markup.

–urlbase=*URL*

Give an URL to the directory containing program source code examples. Used to concatenate with filenames.

-v, –verbose

Print verbose output. Several calls increase verbosity.

–[no]warning

Toggles printing of warning messages to STDOUT. Default is on.

-w [*webdir*], –web=[*webdir*]

Output *.web* file for `qdbcgi` in *webdir*.

BUGS AND LIMITATIONS

The randomization files *.rnd* are only overwritten by an explicit **-r** option. This is stable in the sense that corrections may be made e.g. to the solutions that go to the teachers version after the exams has been printed and reprocessing will always give the same values as the exams. If new values or questions with values are added qdbpp must be read with **-r** for these to appear.

4 qdbbatch

Script to read a class list and generate individual exams to students using qdb.

SYNOPSIS

qdbbatch [**-c**|–**cols**=*column1[,column2[,...]]*] [**-p**|–**print**[=*printer*]] [**-r**|–**rows**[=*rowinterval*]] [**-t**|–**table**[=*table*]] [**-u**|–**user**[=*usercolumn*]] [**-v**|–**verbose**] tablefile qdbsourcefile

qdbbatch **-m**|–**mark**[=*column1[,column2[,...]]*] | **-o**|–**outfile**=*outputfile* [**-a**|–**answerdir**[=*answerdirectory*]] [**-c**|–**cols**=*column1[,column2[,...]]*] [**-r**|–**rows**[=*rowinterval*]] [**-t**|–**table**[=*table*]] [**-u**|–**user**[=*usercolumn*]] [**-v**|–**verbose**] [**-w**|–**webdir**[=*webdirectory*]] tablefile

qdbbatch **-h**|–**help**

qdbbatch –**man**

DESCRIPTION

qdbbatch reads a list from an open document spreadsheet file (.ods) and creates or marks previously created individual exams for users in thelist.

In creation mode it takes output filenames from the column given by **-u** switch in the row interval given by **-r** switch and makes randomised runs with qdbpp. Other tabulated material in the same table of the tablefile can be sent as extra-arguments to be input in the qdbsourcefile.

In mark mode, ran with the **-m** or **-o** switches, answerfiles from a qdbcgi are compared to correct answers to the individual. Files are searched in the directories given by the **-a** and **-w** switches unless given by those switches *qdb-answers* and *qdb-questions* are searched if they exist. Filenames for the files to compare are constructed from the column given by **-u** switch in the row interval given by **-r** switch. With **-o** switch the results are written to a textfile *outfile* as a list with names taken from colums given by the **-c** switch and the number of correctly given answers to each question. The **-m** switch writes results back to listfile, use at own risk and on a copy of listfile.

OPTIONS

-a *answerdirectory* –answerdir=***answerdirectory***

Directory to search for answer files in markmode. Default is qdb-answers if it excists.

-c *column1[,column2[,...]]* –cols=***column1[,column2[,...]]***

For the specified columns the rows defined by **-r** is read, for each call to qdbpp the contents of the cells is sent to qdbpp by it's -e switch. Columns may be given by a comma separated list or intervals (e.g. 1-3).

-m [***column1[,column2[,...]]***] –**mark**[=***column1[,column2[,...]]***]

Runs qdbbatch in markmode and prints results to columns indicated in tablefile. **Use with care.**

-o *outputfile* –**outfile** =***outputfile***

Runs qdbbatch in markmode and prints results to tab separated list in *outputfile* textfile.

-p[***printer***] –**print**[=***printer***]

Print resulting files by issuing `dvips` *user*`_questions.dvi -o - |lpr -P printer`.

-r *rowinterval* –rows=***rowinterval***

Row interval to read from tablefile.

-t *table* –table=***table***

Table to read in the tablefile.

-u *usercolumn* –user=***usercolumn***

Column number for the column where the output filenames are stored.

-v|–**verbose**

Print verbose output several calls increase verbosity.

-w *webdirectory* –webdir=***webdirectory***

Directory to search for files with correct answers in markmode. Default is qdb-questions if it excists.

5 qdbsort

Script to sort out a collection of questions from a qdb database.

SYNOPSIS

qdbsort **-b**|–**book**=*reference* [options] [filepattern]
 qdbsort –**texreference**=*reference* –**auxfile**=*file* [options] [filepattern]
 qdbsort **-h**|–**help**
 qdbsort –**man**

DESCRIPTION

qdbsort reads all files in questiondirectory that contains an underscore (_) and ends with *.tex*. The questions in the file are chosen and sorted according to the roles set by the arguments given and output is written in a *.tex* file with preprocessor commands for qdbpp. This output file may be included in another qdb *.tex* file using the `%(INPUT)` preprocessor command.

The mandatory argument must be a valid regular expression that matches filenames that shall go in the list and sets the base part of the filename (without suffix) to $1. Default is (.*_.*).tex$

REQUIRED ARGUMENTS

In order for a sorting to make sense a primary sorting kriteria is needed, either of two options is required

-b *reference*, –book=*reference*

Sort out questions referencing to *reference*. *reference* is the string used as the bibtex citekey in the questions. Sorting is done by page numbers as given in the `%(REFERENCE)` markup in the question filess.

or

–texreference=*reference*

Sort out questions referecing to *reference*. *reference* is the string identifying the LaTeX source file for a file containing the text referred. Sorting is done on the `\label` marks in the source file as referred in the `%(TEXREF)` markup in the question files. The job must have been run and the `.aux` file available. Must be used together with

–auxfile=*file*

The `.aux` file to lookup labelmarks from *reference*.

OPTIONS

-a *name*, –author=*name*

Sort out questions with the author *name*.

-c *number*, –clearpage=*number*

Clearpagecommand to be set in between questions. *number* shall be in range 0-2 where

1. means questions following each other in paragraph mode,
2. that each question start a new page (`\clearpage`),
3. that each question start on a new odd sided page, i.e. a new sheet of paper in hardcopy (`\cleardoublepage`).

-d *directory1[,directory2[,...]]*, –directories=*directory1[,directory2[,...]]*

Directories to search for questions. If not given the default directory is ../fragor/.

-h, –help

Print help message.

-i [*set*], **–intersectpages=**[*set*]

Filter out questions for which the page set in the question file intersects with the *set*. I.e. all question that has any reference to *set*. The set of pages may be a combination of comma separated intervals indicated by -. Open intervals may be given by finaling the open end by paranthesis e.g. ”1,3-5,8-)”.

-l [*set*], **–lastpage=**[*set*]

Filter out questions for which the lastpage in the set in the question file is within *set*. I.e. all question that is likely to fit while reading pages *set*.

–man

Print the whole manpage.

-o [***file***], **–output=**[***file***]

Output *.tex* file. Default is *sortedquestions.tex*.

-r [***interval***], **–revisions=**[***interval***]

Sort out questions with revision numbers in interval *interval*. The interval may be open ended in the end by writing ”5-)” for revision numbers 5 or higher. Revisions are matched on the `%(SVNREVISION){$Revision$}` markup where the revision number is set by subversion. I.e. this switch only works on subversion controlled question files with the property Revision set.

-s [*set*], **–subsetpages=**[*set*]

Filter out questions for which the page set in the question file is a subset of the *set*. I.e. all question that may be answered by reading *set*.

-u, –union

Options -a, -i, -l, -r and -s are usually limiting the list of question to the intersection of the three lists. With the -u switch the union of the lists are used instead i.e. the questions written either by the author *name*, OR in the revision set *set* OR fulfilling the page criteria.

-v, –verbose

Print verbose output. Several calls increase verbosity.

–[no]warning

Toggles printing of warning messages to STDOUT. Default is on.

6 Format of the source LATEXfile

The qdbpp preprocessor only reacts to the special qdb markup in the file. A source LATEX-file with no qdb markup will result in a set of output LATEX-files that are identical to input or empty. It’s up to the writer to make the file a valid LATEX document. In pracical use were qdb markup is included in the input file it is advised to load the corresponding settings file with a `\include{settings.tex}` or a modified version of the same code. Each markup must start a new line, not proceeded even by whitespace, except where noted. The full markup must be contained in one row of input. Apart from the markup of section 9 the supported markup are:

`%(PROFILE)`{***profile***} Choose profile to use. Must be the first command in the file. Choices for *profile* are:

exam Uses Philip Hirchhorn’s exam document class. Numbered questions with two classes of points written to the right. Outputs questions (the examination handout with answerlines and boxes to fill in), solutions (questions and solutions intermixed), answers (separate answers file), teachers (verbose version of solutions with inputfilenames and comments). Randomization is stored to a .rnd file with possibility to recall the same randomly choosen questions and parts.

planning A special planning format where constructed for a course divided in lectures or whatever other units each in one section containing reading instructions, practical exercises, problems and textsection. Problems continously numbered Q# throughout the document using an own counter. Support for dot-marked and starred and muliply starred versions for special problems, e.g. that solutions are to be delivered to teacher or to mark difficult questions. Both answers and solutions output to inclusion type files that may be included in the questions file by an `\include` command. Gives internal hyperlinks two ways between solution part and question with printed page number reference to solution and one way hyperlinks from answers to questions. Support embedded examples with questions numbered "Exempel #" and got the solutions directly in the text in coloured text.

collection Used for collection of questions e.g. sorted out by qdbsort. This profile provides only one outputfile *filebasename*`_questions.tex` in a manner rather similar to the *filebasename*`_teachers.tex` file in the exam profile, i.e. question followed by solution and comments. The format is adopted for the purpose of making teachers collections of questions.

`%(AUTHOR)`{***Author Name***} Provide the name of the author of the compilation. For included questions and graphics the attributions and licensing information will be supressed in most cases for contributions by the author himself.

`%(PREAMBLE)` includes the `settings.tex` file as well as other latex commands for the preamble to the output files. `settings.tex` is parsed by inclusion in by the parser for the qdb file.

`%(DIRECTORIES)`{***paths***} sets the path to search for question files. Overrides the default path (../fragor/) and may be overridden by the qdbpp command line argument --directories The directories in *paths* are separated by semicolon (;).

`%(GRAPHICSPATH)`{***paths***} inserts the LaTeX `\graphicspath` command and also provides the same path for qdbpp search for metadata in textfiles for the included graphics. The directories in *paths* are separated by semicolon (;).

`%(BEGIN_QUESTIONS) ... %(END_QUESTIONS)` Questions environment into which all of the following commands are to be contained in.

`%(RANDOM_INCLUDE)`{***#includes***}{***#questions***} Randomly include *#includes* among the following *#questions* questions.

`%(INCLUDE_QUESTION)`{***file***} Includes question in file *file*. Takes options:

-e, --example Include example with solution. *Planning profile only.*

--exercise Include as exercise. This effects the numbering scheme and makes sure the exercise is numbered as an exercise and changes the label to `ovn:`*filename. Makes difference for planning profile only.*

-g, --gpoints *#points* Number of basic points for question. Overrides parameter in questionfile. *Exam profile only.*

-m, --mark *mark* Put *mark* in front of question number. E.g. '`*`' or '`\textbullet`'. *Planning profile only.*

-n, --nolines Supress the printing of lines or space for giving answers.

-n, --nosolution Supress the printing of solutions to the solution output. In exam mode solution is anyway printed to teacher output.

-p, --parts *partnumbers* The parts of the question to use, or if -r is also given randomly chose from.

-r, --random *#parts* Chose *#parts* parts randomly. If -p is also given parts are randomly chosen in the subset specified by -p.

--sourcecodeurl {*URL*} Complete URL to source code linked in the question.

--urlbase {*URL*} Locally override baseurl set in qdbpp.

--values {*value1*}{*value2*}... Provide own values to a question that has numbered values substitution.

--valueset *set* Use the *set*th set of values in a question using numbered values substitution.

-v, --vgpoints *#points* Number of extra points for harder questions. *Exam profile only.*

`%(INCLUDE_EXAMPLE)`{***file***} Alias for `%(INCLUDE_QUESTION)`{*file*} -e

`%(INCLUDE_EXERCISE)`{***file***} Alias for `%(INCLUDE_QUESTION)`{*file*} --exercise

`%(BEGIN_PAIRING)`{***#pairs***} `...%(END_PAIRING)` Pairing environment for `%(PAIR)` commands. Randomly chooses *#pairs* pairs from random list if necessary. *NB syntax subject for change to option in future releases.*

`%(PAIR)`{***file***}{***use***}{***random***} Include pairs of concepts and their descriptions from special pair file *file*. The concepts in list *use*, given as comma separated values (1,3) or intervalls (2-4) or a mix, are included. The descriptions in list *random* are included to the random list of the pairing environment. Pairs may be anything and is not restricted to concepts and descriptions.

With exam profile descriptions are given in order, each marked with a letter in one column, and the descriptions with an empty box for answer randomly shuffled in another column. With web output turned on the answer form is one column only containing descriptions i. e. similar to second column in the question version.

With planning profile the concepts only are put in the questions file and the explanations in the answersfile.

Must be in a pairing environment *NB syntax subject for change to options in future releases.*

`%(INPUT)`{***file***} preprocessor equivalent to latex `\input` command. File *file* is opened and parsed by the preprocessor at the spot the command is given.

`%(AUTHOR_ATTRIB)` prints out an attribution to authors of questions that have been used in the compilation. The author of the compilation is usually supressed.

7 Format of the question files

As for the question file, each markup must start a new line, not proceeded even by whitespace, except where noted. The full markup must be contained in one row of input. Also the markup of sections 8 and 9 may be used.

There are two kinds of markup for the question files, there are begin and end matched pairs that works like LaTeX environments. There are four of that kind which devide the question file in four regimes. They should come in the following order:

`%(BEGIN_QUESTION)...%(END_QUESTION)` This part is all text that goes to the questions version. Usually only contains the question and if applicable an `\answerline` or `\fillwithlines` command for answers.

`%(BEGIN_SOLUTION)...%(END_SOLUTION)` Is supposed to contain full solutions to questions that needs calculations to give an answer. Contents of this part goes to the teachers and solutions versions. If a question only have an answer that needs no longer solution this part may safely be left out.

`%(BEGIN_ANSWER)...%(END_ANSWER)` Here only the final answer to the question should go. These go to the answers part. If no solutions are provided the answer is also going to the teachers and solutions versions in the exam profile.

`%(BEGIN_JUDGEMENT)...%(END_JUDGEMENT)` This part goes only to the teachers version. It's intended to give judgement criteria to questions, most useful for questions that may render in several points.

If the question is divided in parts each of these must contain a nested environment

`%(BEGIN_PARTS)...%(END_PARTS)` that should go into each of the four above if present and contain a similar list of parts, there is no way to relate parts in the four regimes exept from the order in which they are written. If parts are selected by the `--parts` switch the same parts are selected from each of the regimes.

The parts enviroment must contain a list of parts followng the syntax:

`%(PART)`{***points1***}{***points2***} Start a part of a question. In the question it may be given with one or two arguments for the number of points (one or two categories of points). qdbpp may be configured to display the points next to each subquestion, sum up for the question or both. The use of points to the parts are encouraged over the -g and -v switches for the `%(INCLUDE_QUESTION)` command as it can be used to automatically recalculate the number of points when only some parts are used.

Finally there is one regime that is not divided in parts and that must be present even if present as it is used to trigger writing of some metadata to the teachers part and in the colletion profile.

`%(BEGIN_NOTES)...%(END_NOTES)` May contain any notes to the teacher about the question. In the example files this regime contains an `%INDEX%` markup that from the beginning has been thougt to contain strings to an index. This feature is not yet implemented but may be in an undefined future.

Before those five regimes the following markup may be used:

`%(TITLE)`{***title***} A title for the question.

`%(REFERENCE)`{***cite key***}{***pages***} Reference to a source where the answer or instructions to find the solution can be found. Printed only in the teachers version. *cite key* is the key that should have gone in a LaTeX `\cite` command, the corresponding bibtex file is `../references.tex`.

`%(TEXREF)`{***cite key***}{***label***} Reference to another latexfile as source, treated similar to `%(REFERENCE)`. *cite key* should be a key that can translate to a bibtex cite key provided by the `texreferences` option to qdbpp. For the reference to resolve a `.aux` file from the referred file must be provided to qdbpp and/or qdbsort. See further in 10.3.

`%(POINTS)`{***points1***}{***points2***} Set number of points for the whole question. In the question it may be given with one or two arguments for the number of points (one or two categories of points). For multipart questions points to the individual parts are encouraged.

To make it possible to make the same question come out with different lexical or numerical values the following three markups are provided.

`%(VALUES)`{***value#1***}**...** Any number of values may be given in a values row and then referenced in the question as `#1` etc. Several rows of values may be given of which only one will be used in the question. This may be choosen in the calling file.

`%(CALCULATE)`{***expression***}{***format***} Calculates the perl *expression* which after substitution for variables is evalutated as eval(*expression*) in the preprocessor. The result is formatted according to *format* to a textstring stored in a new variable that can be substituted from `#C`nr where nr is a counter of the calculated values increasing from 1 for the first.

##expression##format## Calculates *expression* and formats in the same way as `%(CALCULATE)` but inserts the result on spot where it is called and does not store a new variable. *This markup may be given anywhere in the text (not necessary starting a new line) but must not cross a new-line border. Unfortunatly only once in each row, filed as a bug*

Finally, especially for questions written for use with the webforms the answer regime may contain special markup for the webforms, that also provides the normal answers for the answerpart.

`%(WEB_VALUE)`{***text***}{***target value***}{l_1}{l_2}{***unit***}{***text***} Used in the answer environment to give the answer to a question asking for a numerical value. In the answer part the will be written "*text* \unit[*target value*]{*unit*} *text*".

For webforms the proceeding and preceeding texts and the unit will be printed while the value itself will be a textbox to fill in. Values between a lower and an upper limit will be considered correct. There are three ways to enter these limits:

1. Both l_1 and l_2 are numbers providing the limits.
2. l_1 is the codeword 'term' and l_2 is a number. Limits are calculated as *target value* $\pm l_2$
3. l_1 is the codeword 'factor' and l_2 is a number. Limits are calculated as *target value* $\cdot (1 \pm l_2)$.

`%(WEB_TEXT)`{***text***}{***answer***}{***synonyms***}{***text***} Used in the answer environment to give the answer to a question asking for a text answer. In the answer part the text will be *text answer text*. In a webform the *answer* will be replaced by a textbox. The *answer* as well as the semicolon (;) separated {*synonyms*} are each considered a correct answer, comparison is made case independent.

7.1 Subversion markup

The collection of questions is not necessarily put under subversion control, but it gives some extra possibilities with extra markup. These markup should be written litterally as below and subversion will fill up. See section 14 for instructions on how to use subversion. At the moment only revision has a practical use to filter with qdbsort (see section 5).

`%(SVNDATE)`{**`$Date$`**} For the date of last commit to the database.

`%(SVNREVISION)`{**`$Revision$`**} For the revision number.

`%(SVNAUTHOR)`{**`$Author$`**} For the username of the user that performed the last commit. *This is just the technical user for the last commit. The author(s) of the question and copyright holder should be given by the %(AUTHOR) markup.*

`%(SVNURL)`{**`$URL$`**} URL to the file in the repository.

8 Copyright, attributions and licenses

To give the author a reasonable chance to compile a set of questions from different contributors or with graphical figures from different sources without braking copyright a system with markup for questions and graphics is provided. Both question files and special metadata files with similar names as the graphics files except for the suffix that should be `.txt` (may be changed in settings to qdbpp) will be parsed for the following commands. The necessary attributions to authors, photographers, illustrationists and authors will be given by using the `%(CAPTION)` or `%(COPYRIGHTNOTICE)` markup in the main, included, task, or question file. Markup in the questions file will be used to include the information in the document automatically or by the `%(AUTHOR_ATTRIB)` markup in the main file.

`%(COPYRIGHT)`{***name***}{***project***}{***subproject***} shall be used to notice the copyright owner to the file. *name* may be either a full name in which case the other arguments are not given or a username at database from which the file was loaded. *project* is the keyword for the project in the case of a username, at the moment only "commons", "wikipedia" and "flickr" are supported. The first two leading to the Wikimedia projects. For Wikipedia the language code shall be given by the *subproject* argument, for commons no last argument is needed. The keyword "link" makes the name a link to the adress given in the *subproject* argument.

`%(PHOTO)`{***name***}{***project***}{***subproject***} similar for photographer.

`%(AUTHOR)`{***name***}{***project***}{***subproject***} similar for author.

`%(ILLUSTRATION)`{***name***}{***project***}{***subproject***} similar for illustrator.

`%(CCBYSA)`{***version***}{***juris***} indicates the file is released according to Creative Commons Attribution Share Alike (copyleft) version *version* license. The second argument may contain the country code for the jurisdiction.

`%(CCBY)`{***version***}{***juris***} similar for Creative Commons Attribution (non-copyleft) license.

`%(CCBYNC)`{***version***}{***juris***} similar for Creative Commons Attribution Non-Commercial license, (non-copyleft).

`%(CCBYNCND)`{***version***}{***juris***} similar for Creative Commons Attribution Non-Commercial No-Derivatives license, (non-copyleft).

`%(LGPL)`{***version***} indicates the file is released according to GNU Lesser General Public Licence (or Library General Public Licence for version 2.0) version *version* license. *version* may be 2.0, 2.1 or 3.0.

`%(COMMONS)`{***filename***} is to print the web adress or create a link to the file at Wikimedia commons.

`%(FLICKR)`{***user***}{***picturenumber***} is to print the web adress or create a link to the file at Flickr. It is only used to produce the link as `http://www.flickr.com/photos/`*user*`/`*picturenumber*. The *user* field may be even other photostreams in which the picture is included.

`%(WEB)`{***adress***} is to print a reference to any other webpage. *adress* should be a valid URL to the webpage.

9 Markup common to all files

Some markup is common and have exactly the same meaning in the main file (and included files in main) as well as in task and question files, those are:

`%(CAPTION)`{***caption text***}{***metadata file***} is a replacement for the LaTeX `\caption` command. The *caption text* is inserted in the caption together with attributions and licensing info fetched from the *metadata file* which should be a filename found in the graphics path. *NB the matching of the arguments is crude and does not allow curly braces in the caption. Use `%(COPYRIGHTNOTICE)` without arguments instead, if the caption contains curly braces.*

`%(COPYRIGHTNOTICE)`{***metadata file***}{***attribution***}{***license***}{***source***} gives a more detailed control than `%(CAPTION)` on which information to be printed. The *metadata file* should be a filename found in the graphics path to the metadata file and the *attribution*, *license* and *source* arguments are flags which may be 1 or 0 to include or not the attribution to the author, photographer, illustrator or copyright owner, the license information and information on from which source the file was taken. Leftout or empty arguments will use default values.

`%(INCLUDE_SOURCECODE)`{***file***} Includes and pretty-prints source code from file *file*. Takes option:

--link Include link to file rather than typsetting.

--sourcecodeurl *URL* override URL by same option from qdbpp or main file.

--urlbase *URL* override URL by same option from qdbpp or main file.

10 Citation and refererencing

The user may use the normal way of latex internal referencing using `\label` and `\ref` and citations using `\cite` markup. There are, however, several automatically defined labels to use internally and some extra markup using citations and references.

10.1 Internal referencing

In planning mode the questions included by the `%(INCLUDE_QUESTION)` markup each got a label to get the references to the answers part work that may also be used to reference the question in the text. Those labels are set to `fra:`*filename where filename is the name of the included file without suffix. The use of --exercise switch or `%(INCLUDE_EXERCISE)` markup changes the labels to `ovn:`*filename.

10.2 Citations

In a qdb-latex file an ordinary reference to a printed source may be given by the `%(REFERENCE)`{*cite key*}{*pages*} markup, where *cite key* is the key that should have gone in a LaTeX `\cite` command and must be found i a used bibtex file.

10.3 Cross citation to other documents

To enable references to evolving latex documents where final pagenumbers are not set the markup `%(TEXREF)`{*cite key*}{*label*} (section 7) refers to another latexfile as source, treated similar to `%(REFERENCE)`. *cite key* should be a key that can translate to a bibtex cite key provided by the `texreferences` option to qdbpp. For the reference to resolve a `.aux` file from the referred file must be provided to qdbpp and/or qdbsort. That is done by the `auxfile` switch to those programs or by the corresponding entry in a config file, see section12.

The *cite key* in `%(TEXREF)` should be a key to a file and if the *label* is in an included file to the referred document the name of the included file should be given after a colon mark such as `citation_to_main_document:included_fil` At the moment only the first part is used and in practice the included files must be included by the qdb `%(INPUT)` or latex `\input` markup. Future versions *may* include sorting by included files by qdbsort and/or support ordinary latex `\include` commands.

The labels are searched in the auxfile from the run of the file. That file must be made available by the auxfile. References is put by qdbpp and sorting is done by qdbsort to the pagenumber found in the `.aux` file.

11 Webform with qdbcgi

For questions with answers written in the special commands `%(WEB_VALUE)` or `%(WEB_TEXT)` a cgi script providing automatic marking can be set up. The script itself is qdbcgi. Installed properly (see section 2.1) qdbcgi reads the .web file generated by qdbpp and displays a self correcting webform.

Until a username and password combination is provided a login page asking for username and password is displayed. The username is the base name for the `.web` file and the password is compared to the password given in `.web` file.

When logged in qdbpp reads the `.ans` file providing the previously stored answers from the same user. Each answer in the `.ans` file is compared to the right answer in the `.web` file and depending on the results either

1. an answerline with an input field according to `.web` file is produced,
2. a message that the provided answer was not correct and a new answerline with an input field according to `.web` file or,
3. an answerline with input field substituted with the correct provided answer.

When the submission button in the end of the page is pressed the submitted values are stored to the `.ans` file which is re-read.

12 Configuration

All the values for the command line options for qdbpp may also be given in configurationfiles. Those should be named 'config' and the default searchpath for them is '../../qdb' and '.' i.e. with values in the local file overloading those in the global file in the qdb directory. Values in both files are given by the options given at the command line. The syntax of the configurationfiles is that accepted by the appconfig::file module.

13 Makefiles

Handling of the commands to produce the final output files may be greatly simplified by using make. In order to use make makefiles are needed. As a start a makefile for use with exams is provided. This file is `qdb/exam.mk` and may be linked, copied or included in makefiles to help setting up exams. The following targets are working

`make` *filebasename*`_questions.pdf`

`make` *filebasename*`_solutions.pdf`

`make` *filebasename*`_teachers.pdf`

`make` *filebasename*`.tar` builds all the tree final targets, makes a tar file with them and all the sourcefiles they depended on to build.

`make clean` removes all files produced.

`make mostlyclean` leaves the pdfs and randomize file (`.rnd`)

`make cleantexauto` removes all intermediate files produced by latex.

`make cleanqdbauto` removes all intermediate files produced by qdbpp.

14 Subversion

How datafiles for the database is stored is up to each author to decide. There are however some extra features is avaliable for subversion controlled question files. The markup for those is explained in section 7.1. The idea is that subversions keyword substitution is used to keep trac of the last commit, with information on who committed, at which time and in which revision. To use subversion for version control of the files a subversion repository must be set up. The primary subversion repository may be used for contents released freely, see section 15.3. The setup of a subversion repository is explained e.g.

in Pilato et al. [2008]. To make use of the version control awareness of qdbsvn and qdbpp any repository is sufficient as long as the keywords are set.

In order to work the corresponding properties have to be set on the files. This is done with the command:

```
svn propset svn:keywords "property1 [property2 [...]]" filename
```

For a full introduction to properties in subversion see Pilato et al. [2008] p. 48ff. In practice if the filenaming convention that questions has underscore in their name but other files like templates don't, the command

```
svn propset svn:keywords "Date Revision Author HeadURL" *_*.tex
```

may be given in the directory containing the questions. Files that already have these properties set is not affected.

15 Database and distribution

QDB is available together with a common database of questions written in the format from different contributiors as a subversion repository at `https://kyla.kiruna.se/qdb/trunk`. The repository is unfortunatly closed for anonymous access, both checkout and commits. This is not a preferred situation that has come up due to two reasons:

1. The primary author Johan Arvelius does not have enough time available to expand the amount of questions for pupils tests for the frequently given courses to let them free to the pupils.
2. It is believed that qdbcgi contains sequrity holes of the kind that cheating is possible, alot easier if the sourcecode was publicly available.

Both of these should be dealt with ASAP to release the code for checkout. Meanwhile subsets of the repository must be released each time a derivative containing copyleft material (almost any content in the database) due to copyright issues.

15.1 Database structure

The database distributed with QDB is structured with the programs and a common setup file that is thought to be needed for all latex output files in a directory `qdb/`. Questions, tests, collections and figures are held together in subjects in such a way that in most cases one folder could be checked out individually. In some cases (e. g. astronautkunskap) the subject folder contains tests with questions from other directories, and some (e. g. geofysik) is not a school subject but a common collection of questions from other subjects. The searchpath for such cross inclusion of questions and pictures must be set up by the `%(DIRECTORIES)` and `%(GRAPHICSPATH)` commands.

In the present database graphics files are in formats to be included by pdflatex (pdf, jpeg, png...) but there is no obsticles to make an independent collection with EPS figures and run latex.

15.2 File naming conventions

The questions in most subjects are named according to a convention where the the first letter describing the purpose for which the question was originally written:

p question for examination.

i question for studying.

w webquestions written for use with qdbcgi.

e examples.

k pairing questions.

After an underscore comes an abreviation for the school where the question is written and after another underscore a descriptive name.

Another convention is used in mathematics (matematik) where all questions are used for examination.

As templates to new questions two files are present in the question folders `quest.tex` and `zpair.tex`. As long as no other questions are written starting with `q` or `z` these templates are easily inserted by e. g. `Ctrl-x i q Tab` in emacs. For the attribution files to the graphics files there is a similar template in the figures folder called `qopyright.tex`.

These are only conventions and have no technical meaning, however a common conventions within a subject may be of great use together with qdbsort to sort questions into categories.

15.3 Commits to the database

Anyone who gets a user to the database are mostly welcome to contribute to it. Adoption to the file naming conventions are appreciated. The database is thought to be opened for public access and therefore all uploads must fulfill the copyright requirements. All files must have their own copyright disclaimer in a comment and with proper markup in the beginning of text files and with an accompanying file to graphic files. All authors are strongly encouraged to use copyleft licenses (preferably cc-by-sa) both for their own works and included graphic files when available. **The absolute requirement for commits to the database is an explicit allowance to redistribute the works in a free maner** even if not copyleft. New sorting criteria to qdbsort to enable sorting out non copylefted material is in the wish list.

Any improvements, bugfixing and feature expansion are welcome, especially of features listed in the wish list. Feel free also to make branches in the repository as apropriate.

References

C. Michael Pilato, Ben Collins-Sussman, and Brian W. Fitzpatrick. *Version control with Subversion.* O'Reilly, 2 edition, 2008. ISBN 978-0-596-51033-6. URL `http://svnbook.red-bean.com/`.