The AcroWeb script collection

Robert Mařík

October 25, 2007

1 Introduction

1.1 What is AcroWeb

The teacher of mathematics can use pdfLaTeX and AcroTeX eDucation bundle[1] to prepare interactive tests for students. Tests produced in this way are unique in several points.

- Each test is a PDF file. Only free Adobe Reader and no extra software and no Internet access is required when working out the test.
- The author of the test has a possibility to ask not only multichoice questios, but also fill-in questions, where the answer is a mathematical formula (function, equation, vector).
- The author of the test can change the function which is used to compare the correct and user's answers. Thus you can ask questions where the answer is not unique, but unique up to an aditive constant factor (useful in integral calculus, for example).

For more details about AcroTeX see `http://www.acrotex.net`. The AcroWeb bundle is a collection of PHP and perl scripts and LaTeX templates which

- read a plain text database of mahematical problems with multichoice questions or fill-in questions (the answer is not a string, but mathematical formula!)
- select randomly given number of problems from this database (databases)
- build LaTeX file which is compiled by pdfLaTeX and sent to the user's www browser.

Roughly speaking, AcroWeb is an interface to build AcroTeX tests on web from a database of problems.

The templates of PHP and Perls scripts are included in the AcroWeb collection and the maintainer of the database need not to know these languages.

[1] LaTeX package written by D. P. Story

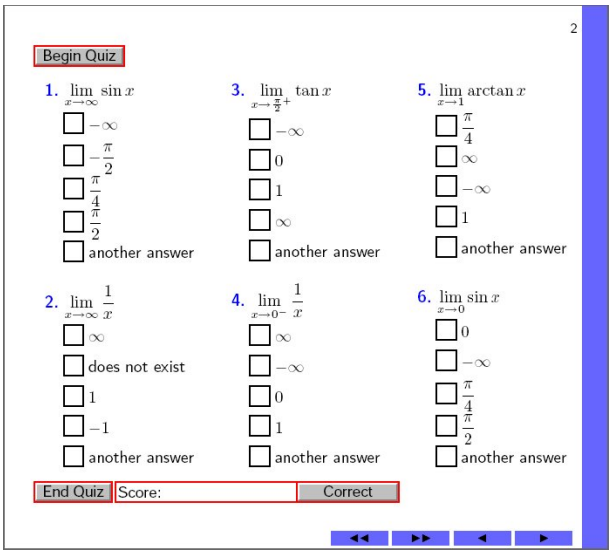

Figure 1: Multichoice test

The structure of AcroTEX commands is hidden to the author of the database, but the author of the test has to put some commands relating these tests into preamble of the TEX file and he should be familiar with AcroTEX. However, he/she can start with customizingthe templates distributed with AcroWeb.

1.2 How AcroWeb looks like

The demo tests distributed with AcroWeb bundle are shown on Figures 1 and 2. The user clicks the Begin Quizz button, answers the questions and clicks End quizz button. The user's score (the number of correct answers) appears in the field Score. If the user clicks Corect button, he can see correct answers to all questions.

1.3 How AcroWeb works

We assume that the author of educational tests runs http server with PHP, LATEX and Perl. The user clicks a link on author's web page which points to the corresponding php script and this php script starts the job:

- A temporary directory in `/tmp` is created. The name of this directory is unique for each process and starts with word `acrotex`, e.g. `/tmp/acrotex6892/`.
- TEX file `test.tex` is created in temporary directory. This file is produced by perl script which reads the database of problems, translates the problems into notation suitable for AcroTEXand selects randomly given

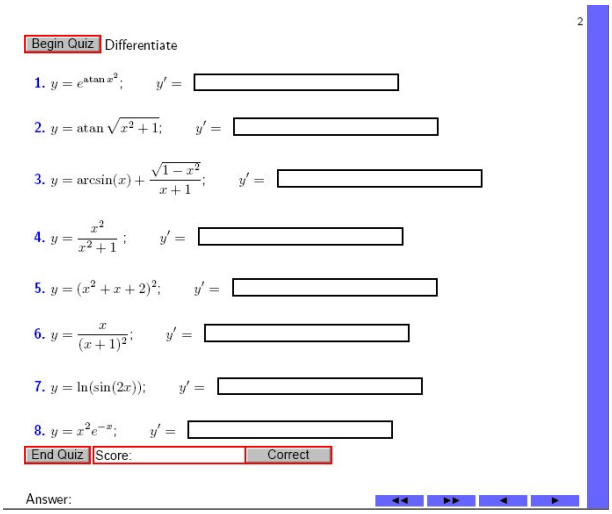

Figure 2: Fill-in test

number of problems and (for multichoice questions) a given number of incorrect answers which accompany a correct answer.

- The file `test.tex` is compiled by pdfLaTeX and the resulting PDF file is opend in the user's browser.
- Finally we clean the temporary directory and write simple log information. These two steps are performed by the function `compile_acroweb` defined in the file `acroweb.php`.

The demo page `demo.html` contains links to php scripts which build the PDF file and to several examples how the database and TeX files look like.

1.4 How to try AcroWeb

The demo site is `http://www.mendelu.cz/user/marik/acroweb`. This site is in Czech, but it contains link to demo file in English. This demo file is included also in the AcroWeb distribution.

1.5 How to install AcroWeb

Installation of Acroweb is pretty simple. The administrator has to put the `acroweb` directory (contains `*.html`, `*.pl`, `*.pm`, `*.head`, `*.tail`, `*.txt` and `*.tex` files) into directory visible from http server, unzip the `phpfiles.zip` also into this directory (this extracts `acroweb.php`, `demo1.php`, `demo2.php`, `template2.php` and `template.php` files) and open the `demo.html` file in browser. For example, if you put the `acroweb` directory to `/var/www/`, open the page `http://your.server.domain/acroweb/demo.html` in your browser. If you put

the directory to local computer (running apache server and php), open the page `http://localhost/acroweb/demo.html`[2] A table with several demo tests appears. If you click links pointing to php file (the second column), a PDF file `test.pdf` should compile and appear in your browser or in Adobe Reader (depending on the configuration of your computer).[3] If the PDF file is not compiled, check that PHP and LaTeX with all necessary packages are installed. If this does not help, read the section Troublesooting which describes how to catch the resulting TeX file and the output of pdfLaTeX compiler. Finally, check that users from the Internet have write access to the `log` directory and file `acroweb.log` in this directory. This file is used to store simple information with time, date, IP address and the descripton of the test.

Works? You can create your own database of problems now.

2 Writing database and customizing scripts for compilation

2.1 Multichoice question

2.1.1 Database of problems

The format of the database is inspired by the format which is used in eLearning section of University information system at Mendel University in Brno[4]:

```
The first problem - on one line (which can be very very long)
+correct answer
-incorrect answer number 1
-another incorrect answer
-yet another incorrect answer
-the fourth incorrect answer
-the last incorrect answer
<one blank line>
The second problem - no more than one line again
-incorrect answer number 1
-another incorrect answer
-yet another incorrect answer
+correct answer
-the fourth incorrect answer
-the last incorrect answer
<one blank line>
```

[2] If the page is not displayed in your browser, check that you have the files in correct path and that your apache server is running.

[3] Do *not* open the PDF file in *another PDF viewer* than Adobe Reader, since Adobe Reader is these days the only PDF browser capable to handle java scripts in the tests.

[4] And also at Technical University in Bratislava and other universities with the same information system.

Here `<one blank line>` means *blank line, no spaces and no tabs.* The order of answers is arbitrary, the number of answers can be bigger or equal to the number of answers used in the test. The file should end with one blank line again. You can use LaTeX commands including mathematics, text faces and also commands defined in the head of the TeX file which produced (see below and files `*.head` in the distribution) during he compilation.

2.1.2 PHP script

The philosophy is the following

- We have a Perl script which reads the database, selects randomly given numer of problems with given number of answers and sends the TeX file to the standard output.
- We have a PHP script which calls the Perl script from the preceeding item and redirects output to a TeX file in temporary directory. This script runs pdfLaTeX compiler and sends the PDF file to the browser using `header` and `readfile` directives.

The PHP script `template2.php` looks as follows:

```
<?php
require ("./acroweb.php");
$directory="/tmp/acrotex".getmypid();
system("mkdir $directory");
system("perl template2.pl > $directory/test.tex");
compile_acroweb("some name",$adresar);
?>
```

On the first line we read the file `acroweb.php` which defines `compile_acroweb` function. On the second line we derive the name of the working directory from the number of the process running and on the third line we create this directory. The next line calls perl script which outputs the TeX file – the output is redirected to the file `test.tex` in the working directory. On the last line we call the function `compile_acroweb` which runs pdfLaTeX, sends the PDF file to browser and writes simple log information containing date, time, IP address and the name of the test in the form

```
08.Oct.2007, 22:12:17, IP: 127.0.0.1, test: some name
```

where "the name of the test" is the first parameter of the `compile_acroweb` function. This function also removes the working directory. However, if the script fails for some reason (due to errors in compilatoin, for example), the temporary directory remains on the server. It is suggested to clean `/tmp` directory regularly in a cron script.

2.1.3 Perl script – multichoice tests

The task for the perl script is to read the problems from database, to select given number of problems from this database and send the TeX file to the standard

output. (The output is redirected to a file in the corresponding PHP script.) The perl script `template2.pl` looks as follows.

```
use acroweb_eng;
system("cat template2.head");
@outA=genrandom("template.txt",4,4,0);
@outB=genrandom("template2.txt",4,4,0);
@out=(@outA, @outB);
print permutemyfield(@out);
system("cat template2.tail");
```

On the first line we read the library with macros. We build the TEX file from three parts: the first part which precedes the questions is stored in the file `template2.head`. The `*.head` file should end with `\begin{questions}` command and we send the contetns of this file to standard output using `cat` command. Then we write the second part of TEX file, where we transform selected problems of the database into the form which may looks as follows.

```
\problem{This is question number $6$}
\begin{answers}{1}
\Ans0 incorrect answer number 1\\[5pt]
\Ans0 incorrect answer number 2\\[5pt]
\Ans0 incorrect answer number 4\\[5pt]
\Ans1 this is correct answer\\[5pt]
\Ans0 incorrect answer number 5
\end{answers}

\problem{This is question number $3$, series $2$}
\begin{answers}{1}
\Ans0 incorrect answer number 3\\[5pt]
\Ans0 incorrect answer number 1\\[5pt]
\Ans0 incorrect answer number 4\\[5pt]
\Ans0 incorrect answer number 2\\[5pt]
\Ans1 this is correct answer
\end{answers}
```

We finish the TEX file by `template2.tail` (this file should start with `\end{questions}`)

Now we describe in more details how the database transforms into second part of the TEX file. The library `acroweb_eng.pm` defines three functions: `genrandom`, `filtruj` and `permutemyfield`. The function `genrandom("template.txt",4,4,0);` reads the file `template.txt` and returns 4 randomly selected problems (the second parameter). Each parameter has one correct and 4 incorrect answers (the third parameter).

The last parameter is a flag which takes value 0 or 1 or 2. The value 0 means: we select 4 incorrect answers, one correct answer and write them in random order. The value 1 works in a similar way, but the last answer is replaced by the phrase "another answer". The value 2 is used for fill-in questions (see below).

The function `genrandom` calls the function `filtruj` first. This function reads

the database and writes the problems with solutions into the form suitable for AcroTEX. After this we randomly select given number of problems and return these problems as strings in a field. If we have one source of the problems (one file) we print this field and finish the file with `*.tail` file. In `template2.pl` we read problems from two files and mix them up. From this reason we put these problems into one field named `@out` and call

```
print permutemyfield(@out);
```

function to print elements of this field in random order.

2.2 Math fill-in test

The format ot the database for fill-in questions is

```
Question on one line
+ correct solution (math formula for acrotex)
- interval where we check the solution
<blank line>
Question on one line
+ correct solution (math formula for acrotex)
- interval where we check the solution
<blank line>
```

(see also `demo2.tex`). We use flag 2 in the `genrandom` function in the corresponding perl script and The other parts of the compilatoin process are the same as for multichoice problems. For example, the file `demo2.pl` has only four rows.

```
use acroweb_eng;
system("cat demo2.head");
print genrandom("demo2.txt",8,0,2);
system("cat demo2.tail");
```

Using flag 2, the database is compiled into the form which may look as follows.

```
\problem{${(x^2+1)^3} $}
\interval{[1,2]}\correctanswer{6*x*(x^2+1)^2}

\problem{${(x+1)\ln(x^2+1)}$}
\interval{[1,2]}\correctanswer{ln(x^2+1)+(x+1)*2*x/(x^2+1)}

\problem{$\cos \left( 2\,x-1 \right) $}
\interval{[0,1]}\correctanswer{-2*sin(2*x-1)}
```

It is supposed that user defines macros `\problem`, `\interval`, `\correctanser` which build the question and the field for student's answer — see the file `demo2.head` or invoke

```
perl demo2.pl > /tmp/file.tex
```

and check the file `/tmp/file.tex`.

3 Questions and troubleshooting

3.1 How to check the TEX file produced?

The directory with TEX file is removed when the script is finished. Hence you cannot view this file. If you wish to see the tex file, change the line which calls the perl script in the PHP file into something like

```
system("perl template2.pl > $directory/test.tex && cp $directory/test.tex /tmp");
```

and the file `test.tex` will be copied into `/tmp` directory.

3.2 I get no PDF file, how to find what gets wrong?

1. You can use the preceding hint and check that the file `test.tex` appears in the `/tmp` directory and this file can be compiled with pdfLATEX.

2. You can change the line of the PHP script into

   ```
   system("perl template2.pl");
   ```

   and the TEX file appears in your browser. But remember that the line-breaks in the TEX file are destroyed.

3. You can call the perl script in command line, redirect the output into `tex` file and compile by hand. For example, you can issue the following commands

   ```
   perl demo1.pl > myfile.tex && pdflatex myfile.tex
   ```

4. The output of the pdfLATEX compiler is redirected into a file by default. You can send the output of this compilaion to the browser. To do this change the line in `acroweb.php` from

   ```
   $command="cd $adresar && pdflatex test.tex>nothing && .......";
   ```

   into

   ```
   $command="cd $adresar && pdflatex test.tex && .......";
   ```

   and the output of the compilation should appear in the browser. If not, you may need to call pdfLATEX with full path, e.g. `/opt/texlive2007/bin/i386-linux/pdflatex`.

5. Check the log file of appache server, e.g. check the file `/var/log/appache2/errors.log` on Debian server.

3.3 How to print the whole set of problems?

To print the whole set of problems with all answers run the script `printacroweb.pl` from the `script` subdirectory. You have to redirect the standard input from the database and redirect the standard output into a TEX file and compile. You can run e.g.

```
perl printacroweb.pl < ../demo1.txt > all.tex && pdflatex all.tex
```

from the `script` subdirectory. The script `printacroweb.pl` work for multichoice questions, a similar script `printacrowebmath.pl` works for file with math fill-in questions.

3.4 What is the future of AcroWeb?

1. The scripts which read the database of problems should be improved. We should allow to write the formulation of the problem on more than one line, we should treat more blank lines as single blank line and treat lines with blank spaces as blank lines. However, these restrictions are marginal for me. Thus if I get no feedback from users, I may keep the scripts in the current form.

2. The log file from compilation by pdfLATEX should be parsed automatically and if an error occurs, the log file should be sent to the maintainer of the database as e-mail.

4 Feedback from AcroWeb users

Do you have any suggestions to AcroWeb? Do you use these scripts on your site? Do you have some other comments, ideas or hints? Let me know, please. My email address is `mailto:marik@mendelu.cz`.

5 Changelog

version 0.79 October 2007: the first public version.