OpenGrade

Benjamin Crowell *www.lightandmatter.com*

Contents

1 Introduction

purpose

OpenGrade is software for teachers to keep track of grades. It can put the students' grade reports on a web server, and allow the students password-protected access to them.

legalities

2 Installing

generic instructions for Unix

You will need a Perl interpreter, which is available as free software. Perl 5.8 and later will work. You will also need the free gcc compiler installed.

First, download the `opengrade-...tar.gz` file from `lightandmatter.com`. If your browser doesn't automatically unpack the file, you can unpack it by using the Unix command `tar -zxvf opengrade...tar.gz`.

Next you need to install some libraries that OpenGrade depends on: `Clone`, `Term::ReadKey`, `Date::Calc`, `Digest::SHA1`, `Digest::Whirlpool`, and `Tk`.

If your distribution includes a sane method for distributing software, you should

use that; explicit directions are given in subsections below for Debian and FreeBSD. If not, then you can use a script that's included with OpenGrade to install them via CPAN's crufty interface.

Next, do the command `make install` to install the program. This will install the software in a subdirectory of `/usr/local/lib`, and will make a symbolic link to the software in `/usr/local/bin`.

After all this, you can run OpenGrade by invoking the program from the command line:

```
% opengrade &
```

or

```
% opengrade my.gb &
```

The .gb extension stands for gradebook. (That was a description of how to run the graphical user interface. There is also a text-based user interface, which you can run by giving the -t option, and omitting the final &.)

Install the "sox" program if you want support for sound.[1]

installing the libraries via CPAN

If you run Linux or BSD, and can't find a better option than installing the libraries via CPAN, here's how to do it. While logged in as root, use the command `make depend` to run a script that will download some open-source software from `cpan.org` that is required in order for OpenGrade to work. The installer script will check whether each of these is already installed, and if it's not, it will download and install them. You need to have an active internet connection for this to work.

Debian Linux

Follow the generic instructions given above for Unix. The dependencies are available as Debian packages: perl-tk, libdate-calc-perl, libdigest-sha1-perl, libclone-perl, libterm-readkey-perl, sox.

You will also need to install Perl's Digest::Whirlpool library, which, as far as I know is not yet (September 2005) available as a Debian package, so you have to use the `get_dependencies_from_cpan.pl` script.

FreeBSD

Follow the generic instructions given above for Linux and BSD, with the following modifications.

First, make sure you have Perl 5.8 or later installed. If you upgrade to a newer version of Perl, follow up with the command

```
use.perl port
```

so that the system will actually use the new version by default.

Install the libraries:

```
cd /usr/ports/devel/p5-Clone ; make install
cd /usr/ports/devel/p5-Date-Calc ; make install
cd /usr/ports/devel/p5-Term-ReadKey ; make install
cd /usr/ports/security/p5-Digest-SHA1 ; make install
cd /usr/ports/audio/sox ; make installcd /usr/ports/x11-toolkits/p5-T
```

You will also need to install Perl's Digest::Whirlpool library, which, as far as I know is not yet (September 2005) available as a FreeBSD package, so you have

[1] OpenGrade uses the external "play" command to play the little "chk" sound when you select a student from the keyboard. The "play" command is just a wrapper for sox. This works on Debian, but didn't work on FreeBSD the last time I tested it, ca. 2004. OpenGrade installs some sound files in /usr/share/apps/opengrade/sounds. If you don't like the default sounds, you can record different sounds and replace the files with your own.

to use the `get_dependencies_from_cpan.pl` script.

If you're switching from an earlier version of Perl to 5.8, and you already had some of these ports installed, you'll have to go to each port's directory, remove the "work" directory, then do "make", "make deinstall", and "make reinstall". (The same problem occurs if you install the packages rather than the ports.)

Finally, go back to the directory that has OpenGrade's stuff in it. Edit the first line of the file 'opengrade.pl' so that it matches the output of the command 'which perl', and then do

```
make install
```

Windows

I do not officially support OpenGrade on Windows, but my wife reports that it works.

First you need to download Perl from `http://www.activestate.com/`. Starting from their homepage, follow the links for downloading ActivePerl. They ask you for some registration information, which is optional, and then you download ActivePerl. I've gotten reports that ActivePerl 5.8 breaks the installer script for OpenGrade, so you should probably download and install 5.6 instead.[2]

Once you have ActivePerl installed, download OpenGrade, and unpack the archive using WinZip or similar software — double-clicking on the file will probably work. Reboot, make sure you have an active internet connection, and then double-click on the file `install_win.pl`.[3] This file will download and install a bunch of open-source software that's required in order for OpenGrade to work; the last piece of software it downloads, Tk, is large, and will take a long time to download over a modem connection. Wait until it's done, and reboot again.

To run OpenGrade, double-click on the file `opengrade.pl`.

Starting with version 2.7.0, OpenGrade no longer has full support for sound on Windows.

MacOS X

The text-based user interface runs fine on MacOS X — that's the system I developed it on. I haven't tried getting the GUI to run in X Windows on MacOS X. You need to install the Developer Tools, and also X Windows, which is now distributed with MacOS X. Since MacOS X is Unix, the procedure for installing OpenGrade itself is similar to the Linux procedure.

3 Upgrading

The process of upgrading is similar to the process of installing, but if you're just upgrading a minor version (e.g., 2.4.12 to 2.4.13), you shouldn't need to redo the part with `install_win` (Windows) or `get_dependecies_from_cpan.pl` (Unix). If you're upgrading a major version (e.g., 2.4.13 to 2.5.0), then you should redo that part. (It won't have to download all the huge stuff again — it will just look for stuff that the new version needs that the old version didn't.)

[2] I found it confusing navigating their web site. Although they change the site from time to time, here's what I had to do the last time I went through the process. From the ActiveState home page, click on ActivePerl under DOWNLOADS on the right side of the screen. Next, click on the red rectangle that says Download. Click through the registration step using the Next button. Click on MSI to download the Windows version of ActivePerl 5.6.

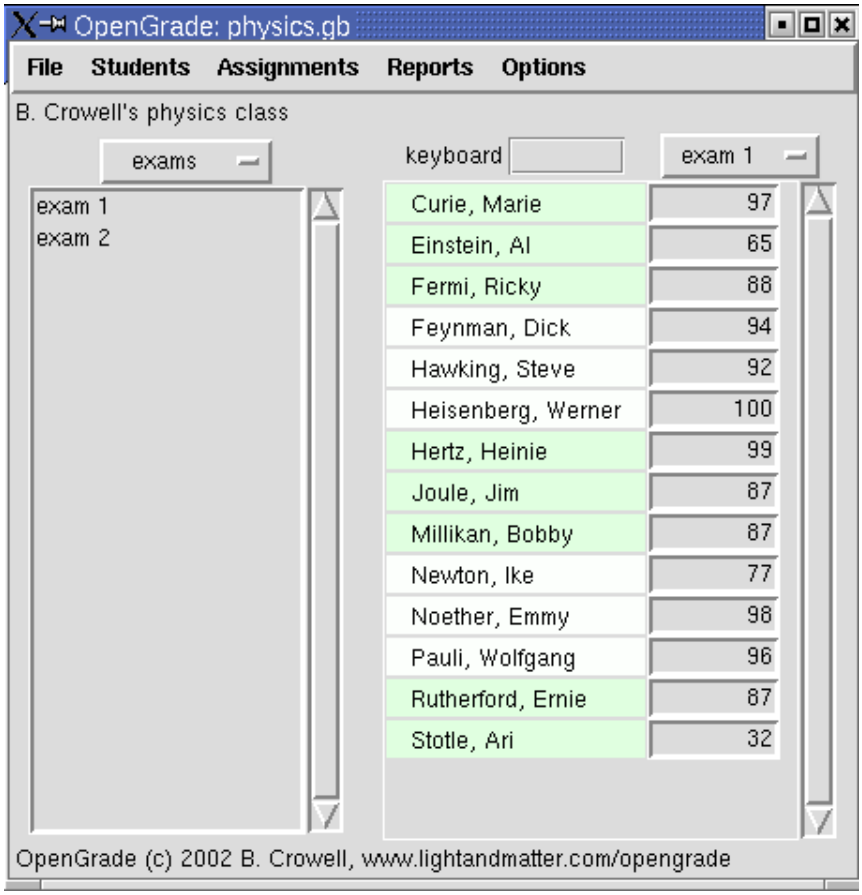
[3] If you don't have privileges for administering your computer, you may find that installing ActivePerl didn't correctly associate the .pl extension with ActivePerl. If this is the case, then you'll have to run the installer by dragging it onto the file C:\Perl\bin\perl.exe, and the same drag-and-drop method will be required in order to run OpenGrade later.

This applies in particular to the upgrade to version 2.5. (If this means anything to you, 2.5 adds a dependency on Perl's Clone module.)

4 Using the Graphical User Interface

entering grades

To see how OpenGrade works in everyday use, run the program as described above, and open the file `sample.gb` that was included among the files you downloaded. Enter `secret` for the password. (OpenGrade uses passwords for authentication, but there is no encryption. See page 8 for more information on how this works.) Click on exam 1 in the list on the left, and you should see something like the window shown below.



On the right, you have a list of the students enrolled in the class. Their scores on exam 1 have already been entered, but you can change them now if you like. If you click on exam 2 from the list, you'll see their results from that exam.

Note the popup menu at the top of the column of scores. With this menu, you can choose to see each student's average test grade, or each student's overall grade in the course. The two tests are the only assignments in the file right now, so the exam averages are the same as the overall grades.

If you explore the other popup menu, above the list of assignments, you'll see that there is another category, homework, with no assignments in it. Let's add a new homework assignment. Select the homework category from the popup menu, and then select New Homework from the Assignments menu. Type 1 in the first space: this assignment will be known as Homework 1. Type a number in

the second space for the maximum number of points possible on this assignment. Click on OK. A blank column of scores pops up in the main window, and you can enter the students' grades.

When entering grades, make sure you have your Num Lock turned on, or else OpenGrade won't respond to numbers from the keypad. (It tries to detect this situation, and will normally beep at you if it appens.) When you hit the Return key (on either the keypad or the main keyboard), the grade will be entered, and the next student will automatically be selected. (You can also use the down-arrow key as a synonym for Return, and the up-arrow to select the preceding student.) This allows you to enter a list of grades quickly if you already have them in alphabetical order.

If you have a stack of papers that is not in alphabetical order, and you're entering all the grades, the quickest way to do it is to use the keyboard to select each student. Here's how it works. Normally when you're entering grades, the keyboard focus is in one of the little grade boxes on the right-hand column of the spreadsheet. If you hit the space bar, however, you'll see that now the student's name is highlighted rather than the student's grade. Once the focus is in the left-hand (names) column of the spreadsheet, you can use the keyboard to select a different student by typing the first few letters of her last name. The characters you've typed will appear above the top of the list. Now hit the tab key, and you can enter the grade. Note that if you keep on typing, the software assumes you're trying to be more specific about what student you have in mind rather than selecting a different student. For instance, if you have students named Jarret, Johnson, and Olson, then typing `jo` means that you're selecting Johnson; it doesn't mean you hit j, then changed your mind and wanted to select Olson instead. If you hit j and then change your mind and want to select Olson instead, then you need to hit the backspace key to erase the j. (The software will also forget about the letters you've typed when you hit tab and take the keyboard focus away from the list of students.) If you get a unique match after typing the first three letters, OpenGrade will make a little "shk" sound to let you know it worked.[4] Once you get used to hearing it, you'll start to notice when you don't hear it, which happens when you have two students with similar last names; that's a warning that, e.g., you may have selected Wilberforce (the first one in alphabetical order) when you wanted Wilson.

When entering grades, the + key is a shortcut for .5. For instance, a score of 3.5 points can be entered as 3+.

creating a new file

To create a new gradebook file, choose New from the File menu and fill in the form. You've created a new gradebook file, which is empty. Now you need to get it set up correctly.

If you use a fixed set of grading standards, you can now enter them by choosing Grading Standards from the Options menu.

Next you'll probably want to set up all your categories of assignments, such as exams, homework, and quizzes. To do this, choose New Category from the Assignments menu. The first thing you're asked for is a short name for the category. It doesn't matter too much what you pick; in the graphical user interface, you'll never even see it again. However, you will see this short name if you ever use the text-based user interface, or if you examine the gradebook file directly in an editor. Just pick something short, like e for exams, or q for quizzes. Make sure you don't use the same short name for two different categories. Most of the rest

[4] This is only if you have beeping enabled in your preferences, and if you have the optional Audio::Data and Audio::Play Perl modules installed.

of the inputs are fairly self-explanatory, except for the space at the bottom for the weight of the category. (To see this space, you may have to use the scrollbar on the right-hand side of the window to scroll down to it.) This is optional. If your student's grades are simply going to be based on the sum of their points in the various categories, then you should always leave this space blank. On the other hand, if you want each category to have a certain weight, then you should enter a weight for every category you create. See page 10 for more information on weighted grading.

Once you've created your categories, you can enter your students' names using Add from the Students menu. If you already have your roster in a text file, you may find it more convenient simply to open your gradebook file in a text editor, paste in your roster, and then put it in the form OpenGrade requires, as documented in section 11.

reports

A variety of reports is available from the Reports menu. The reports offered in this menu will depend on what student, category, and assignment you've selected. When you choose a report from the menu, it's displayed on the screen. You can then hit OK if all you wanted to do was look at it, or Save to save it to disk as a plain text file, e.g. for printing.

5 Goodies

This section documents various features of OpenGrade that you may or may not need. Almost everything in this section applies equally to the GUI and the text-based interface.

extra credit

If an assignment is going to be for extra credit, you can simply set it up with a maximum point value of zero, but enter scores that are greater than zero. Normally, the software beeps at you and asks for confirmation when you try to enter a grade that is higher than the nominal maximum for the assignment. But when the maximum is zero, it will accept nonzero values without complaining.

Sometimes you have an assignment that is not completely extra credit, but it may be possible for students to do something extra that gives them more than the nominal maximum number of points. When you enter a score that's above the maximum, the software will pop up a dialog box asking you if that's what you really want. To avoid this dialog box, you can just type in the score with a trailing letter x, e.g. `101x` on an assignment with a nominal maximum score of 100.

dropping a student

When you drop a student, the student's grades will still be in the gradebook file, so that if you need to, you can add her back in later. In the GUI, you drop a student by clicking on her name in the main window, and then selecting Drop from the Students menu; to reinstate her, select Reinstate from the same menu.

scores that don't count

You may wish to record scores that don't count at all in a student's grade. (For instance, you might give a diagnostic test, or record whether the student has completed some safety training.) There is an option for this when you set up the category.

dropping some scores

When you're setting up categories, the software will ask you how many scores to drop in this category. For instance, you could choose not to count each student's two lowest homework scores.

cloning a file

The cloning feature allows you to make a new version of a gradebook file, with all the students and grades deleted. This is useful when you're starting a new semester, and want to get going with all the same categories and grading stan-

dards you used last time.

reconciling files

If you enter grades both at home and at work, you have to keep synchronizing your gradebook files between the two computers. If you forget to synchronize them, you can end up with two inconsistent versions of the file, and it may not be obvious what the differences are. To fix this, use Reconcile from the File menu. It asks you to select two files, the first being the file you want to change, and the second being the one from which you want to fold in changes. Let's say these files are a.gb and b.gb, respectively. Basically the result is to form the union of the two sets and put it in a.gb, i.e., after you're done, a.gb will contain any scores you entered in either of the two files. It works the same way for students, categories, and assignments. If a grade for a particular student is inconsistent in the two files, the program will ask you whether to change a.gb so that it has the score from b.gb, or leave a.gb alone. Grading standards, as well as properties of students, categories, and assignments are left as in a.gb, so for example, if a student is marked as having been dropped in a.gb but not in b.gb, the student is still marked as dropped.

6 Using the Text-Based User Interface

The user interface is menu-driven, and is fairly self-explanatory.

Often you'll need to select a particular student. Typically you'll just enter the first few characters of the student's last name, and hit return. As you go along, the software will display a list of all students whose names match the characters you've typed so far. When you see that there's only one student on the list, then you know you're ready to hit return to select that student. Capitalization is ignored here.

There is a time-saving trick that you can use when you have more than one student with the same last name. For instance, my school has a large Vietnamese population, and my classes usually have several students named Nguyen. Suppose my class includes both Phong Nguyen and Thu Nguyen. To select Phong Nguyen, I can type `ng`, which narrows it down to these two students, then hit the tab key, and then type `p`.

7 Safety Features

Grades are valuable information, and OpenGrade has a whole repertoire of methods to help you make sure you never lose any of them.

auto-save

Let's say you're editing a gradebook file called `foo.gb`. OpenGrade will save a copy of your work once per minute[5] under the filename `#foo.gb#` (the same naming convention as the one used by the Emacs editor). If your computer crashes in the middle of an intense grade-recording session, you should be able to recover most of your work from this file. (You may want to wait one minute before rebooting, in case the program is still running and there are changes it hasn't saved yet.) The auto-save file is deleted when you eventually save your file in the normal way, so your directory should not get cluttered up with auto-save files. Note that if you're recovering from a crash, you should copy the auto-save file before starting OpenGrade again, because the auto-save file will be overwritten when you open the original file.

[5] If you're using the text-based interface, it saves it every time you cycle through the main menu.

auto-backup When you save the file `foo.gb`, OpenGrade will preserve the old version of the file under the name `foo.gb~` (again, the name is based on the Emacs convention). This takes care of the situation where you save your file, and then realize you've messed something up.

8 Passwords and Authentication

Every OpenGrade file has a digital watermark that is based on the password you choose for the file. Suppose one of your students somehow gets access to your gradebook files. Of course, this shouldn't happen, because you should store your gradebook files on a computer account that nobody has access to without logging in, but let's assume that you got up from your desk without logging out, and a student has walked up and is trying to tamper with your files. If she tries to open one of your gradebook files using OpenGrade, it will demand the password, and will refuse to open it without it. (But don't leave your desk while you have a file open!) If she opens the file using a text editor, she could start messing around with its contents. However, when you get back to your computer and open the file in OpenGrade, it will detect the fact that the digital watermark is no longer valid. OpenGrade does not store your password anywhere, so it assumes that the problem with the watermark is because *you* are an unauthorized person trying to tamper with the file! It will refuse to open the file for you. This is your warning that the file has been tampered with. OpenGrade will also let you view a report that shows more information about the tampering, e.g., which lines were tampered with, deleted, or added.

In everyday use, you should go by the following rules to protect your files. Make frequent backups and printouts. Don't store your gradebook files on a computer that people can get access to without logging in. Never leave a file open in OpenGrade if you're getting up from your desk. Use a good password: one that is at least eight characters long, isn't a dictionary word, and contains both letters and digits. Don't write your password down, and don't use passwords people can guess, such as your license plate number, your kid's birthday, or your pet's name. Don't type your password if someone can look at your hands and see your keystrokes. Don't give anyone access to copies of your gradebook file, because a determined hacker might be able to determine your password by writing a program to guess millions of possibilities.

Here are some real-life situations that can arise, and methods for dealing with them:

You try to open the file, and OpenGrade doesn't accept your password. Don't panic! The most likely cause of this is that you've made a mistake typing in your password. Just try again.

You've forgotten your password, and you need to edit the file. OpenGrade does not store your password anywhere, so there is no way to get it back. Use Change Password under the File menu to change the file's password.

You need to look at an old OpenGrade file from years ago, and can't remember the password The easiest solution is to examine the file in a text editor. Otherwise, change the password as described above.

A student tampers with your file using a text editor. Once you've convinced yourself that you didn't just make a mistake typing in your password, here's what you should do. Click on the button to view the report on the tampering, and save the report to a file. Next, make a copy of the file. You'll probably

also want to print out both of these files. Next, restore your gradebook file from your latest backup. (If you've been uploading your grade reports to a server, then you should have a recent backup copy there.) By examining the report on the tampering, you should be able to repair some or all of the damage, and you may be able to figure out who the tamperer might be, e.g., if all the changes were increases in a particular student's scores.

A student changes the password of your file. This could happen if the student has read this documentation! You'll find out that the file has been tampered with, because OpenGrade will refuse to open it with your password. The procedures for handling this are similar to the ones described in the preceding paragraph. However, you won't be able to get a line-by-line report on the tampering.

This is too much security for you. You don't want to be bothered with passwords. Just use a password with no characters in it. All you have to do is click on the OK button whenever OpenGrade asks for a password.

This isn't enough security for you. Download Gnu Privacy Guard (also known as GPG or GnuPG), and use it to encrypt your gradebook files.

9 Posting Grades on the Web

This section describes OpenGrade's machinery for posting grades on the web.

To post grades on the web, you'll need access to a server where you can install files and CGI software yourself. Install the `ServerOG.cgi` script distributed with OpenGrade. To view their grades, students will interact with a second CGI script, `Spotter.cgi`, which is distributed with the Spotter software, also on the `lightandmatter.com` site. You will need to set up a directory tree on your server as described in the Spotter documentation, in the section titled "Interfacing."

Once this is all set up, you can upload grades by selecting Post Grades from the Server menu.

10 How Grades Are Calculated

There may be situations where it's not obvious to you how the software is calculating students' grades. This may happen when you have categories in which you drop some low scores, or when you have assignments that aren't due yet, but which some students have already turned in. OpenGrade has definite mathematical rules, documented in this section, for handling such situations. Although the rules are a little complex, there is one general principle that will usually keep things clear in your head: the number of possible points is always the same for every student. This principle has two advantages: (1) If we want to know who is doing better in the course, student A or student B, we don't have to specify whether we mean better in the sense of total points or better in percentage terms. (2) It makes sense to compute statistics like averages and standard deviations for the class as a whole, and these statistics can be computed in terms of points, not just in terms of percentages.

dropping low scores

When dropping low scores, we choose them so as to maximize the student's total points, not the student's percentage. The maximum number of possible points in the category is calculated by leaving out the assignments that have the lowest

number of points possible. Thus, the numbers deleted from the list of scores do not necessarily correspond to the numbers deleted from the list of possible points. If there is extra credit on some assignments (i.e. the student has scores higher than the maximum), then these extra credit points are skimmed off the top before any of the calculations are carried out, and then added back on at the end.

due dates

When you create a new assignment, the default is to leave the due date blank; the software then assumes that the assignment has already fallen due.

The only time you really need to enter a due date for an assignment is when it's something that some students will turn in early. You only need to enter the month and day, not the year; the year is figured out based on the date you gave for the beginning of the term when you set up the file. If an assignment isn't due yet, then it doesn't count in students' grades at all. This is in keeping with the general principle that the number of points possible is the same for all students. Thus, a student may turn in an important assignment early, know she had a certain grade on it, and know that her grade in the course is going to change a lot on the assignment's due date; but the effect will not show up in the software until the due date.

An assignment is considered to have fallen due at the beginning of the day it was due. For instance, if June 3 is the day you'll be turning in final grades for your course, then you can set the due date on a term paper to June 3, and it will be included in the grades you calculate on June 3.

weighted grading

You can set up a gradebook file for either weighted or point-total grading; this is determined when you make your first category.

With weighted grading, if the total number of points possible in a particular category is zero, then that category is treated as if it doesn't exist, and the other categories then become more important. Because of this, there can be no such thing as an extra credit category when you use weighted grading.

11 Format of Gradebook Files

OpenGrade's files are in a plain text format that you can edit in a text editor. The following is a small example.

```
1  class
2    .title "B. Crowell's Physics 205/210, spring 2002"
3    .staff "bcrowell"
4    .days "MW"
5    .time "13:30"
6    .term "2002-1"
7    .dir   "bcrowell/s2002/205"
8    .standards "A:90","B:80","C:70","D:60","F:0"
9
10 preferences
11   .file "/home/bcrowell/.OpenGrade_prefs"
12
13 categories
14   .e "catname:exam,exams","max:120"
15   .hw "catname:homework,homework","drop:1"
16   .q "catname:quiz,quizzes","ignore:true"
17
```

```
18  roster
19    .newton_ike "id:1234"
20    .einstein_al "id:5678","dropped:true"
21    .oflaherty_karen "id:3333","last:O'Flaherty"
22
23  assignments
24    .e.1
25    .e.2 "max:135","name:midterm"
26
27  grades
28    .newton_ike.e "1:119","2:135"
29    .einstein_al.e "1:120"
30    .oflaherty_karen.e "1:120","2:135"
```

Lines 2-6 give general information about the class. The title is placed at the top of web reports. The staff list gives usernames of the instructors and TA's who may make changes to the gradebook. The list of days the class meets is used to make it easier to enter certain dates, and likewise for the line specifying the date on which the current term began. The days, time, and term fields are all optional, and can be left as null strings.

Line 7 gives a subdirectory of the directory `cgi-bin/spotter` in which this class's grades will be stored; this is only needed if you're posting grades on the web and letting students view them with Spotter. Line 8 sets your grading standards for this class. On reports, these symbols will be displayed along with the numerical scores. If you don't want any grade symbols displayed, you can just leave the rest of the line blank after `.preferences`. The symbols are entirely arbitrary.

Lines 10-11 tell where your preferences file is located. OpenGrade tries to put this in the customary location for your operating system.

Lines 13-16 specify that there are three categories of graded work in this class: exams, homework, and quizzes. The short tags `e`, `hw`, and `q` are used internally, and are also as headings in some tables. The `catname` parameter gives two forms of the name, which are usually singular and plural. The first one is used in constructing strings like `exam 1` for use in reports, and the second one is for headings in reports. In line 14, we specify that exams are worth a maximum of 120 points by default; this is optional. In line 15, the parameter `drop:1` says that we drop the lowest homework score. In line 16, `ignore:true` says that quizzes don't count toward the students' grades at all. The category parameters `ignore` and `max` are just defaults that are applied when new assignments are created, and these defaults can be overridden for individual assignments.

Lines 18-21 give the roster. Normally the names are given simply as database keys, in the format `lastname_firstname`. When the name is displayed, the first character of each name is capitalized. When this is not the correct capitalization, or when the name contains punctuation, the correct form can be indicated separately, as in line 21. Line 20 indicates that Al Einstein has been dropped from the course, so he will not show up on reports. However, his data are still in the file, and he can be added back in at any time.

Lines 23-25 list the assignments that have been scored so far: two exams. The first exam is worth 120 points, the default for the exam category. The second exam is worth a little more, and instead of being called "exam 2," it will be labeled on resports as "midterm."

Lines 27-30 list the students' grades.

This listing is different from the OpenGrade files you'd typically see, because it lacks authentication codes. Authentication adds some data to the file which is preceded by a # sign wherever it occurs. Every line has some authentication information at the end, and there are some separate authentication lnes at the beginning and end of the file. The format of the authentication codes is described in the source code of the `LineByLine` module. If you cut and paste this example into a file, and then try to open it with OpenGrade, it will refuse because it doesn't have valid authentication codes. You would have to do Change Password from the File menu to add authentication codes.

I haven't written a formal grammar for the file format, but here's some practical information about some of the possible ambiguities.

Sections are typically separated by a single blank line, but the number of blank lines between sections is not significant, and will not be preserved when OpenGrade reads a file and then writes it back out. The order of the sections is arbitrary, but to help make the files more readable to humans, it's recommended to start with the following sections in this order: class, preferences, categories, roster, assignments, grades. Other sections will be added in the future, and an application that reads and writes the format is responsible for preserving any sections that it doesn't understand.

Although the authentication code at the end of each line is marked with a pound sign, it's not treated the same as comments are typically treated in Unix configuration files: it's required, not optional; it's not preserved verbatim when a file is read and rewritten; and it's not free-form. The remainder of the discussion applies to what's left of the line after the authentication code has been checked and stripped off. No significance is attached to leading or trailing whitespace, but for human readability, each section should be indented. Programs that read and write the format can and should indent their output consistently, disregarding the indentation of the original input. A line that ends with a comma continues onto the next line as if there was no newline after the comma.

12 Future Improvements

general All client-server transactions should be wrapped in a subroutine. The subroutine should check whether the password works, and if not, prompt for the right one. If not connected, should give an error message. It should then reconcile the server and client: server's e-mail overrides client's, and all other data from client overrides server's: disabled accounts,...? Should also handle errors. Need to deal with cases where the keys for the same student on the server and client become inconsistent.

It would be nice to be able to export to a spreadsheet in tab-delimited or comma-delimited format, which could then be imported into Excel, OpenOffice, etc. I would also be cool to be able to make custom-formatted report cards; I think the right way to do this is probably to use the facilities built into a word-processor (OpenOffice?) for merging data into a form.

It should be possible to get stats like what percentage of the class has A's, etc. This could be done by adding more outputs to the Crunch::stats subroutine.

Downloading grades: Needs more flexibility for scoring.

MyWords.pm has a mechanism for defaulting to English if the translation in the

preferred language isn't available. This should be generalized so that the fallback isn't always English, and it should be in Words, not MyWords.

Some server operations should be made more efficient. When uploading grades, compression could be used. When listing work from the server, the algorithm that runs on the server shouldn't read each file more than once.

When downloading work, due date should be easier to edit, and it should be harder to forget to set it correctly; should round off the time to the nearest hour or half-hour, and should set the date to the most recent class meeting by default. Could also have the meeting time of the class stored in the gradebook file. The algorithm is a kludge: AND together the "find" parts.

GUI There should be an easier way to look at your grading standards. You currently can't look at them without going through a couple of steps as if you were going to modify them.

A bunch of the menu items should have "..."

The routine `ExtraGUI::save_plain_text_to_file` should offer a better dialog box for choosing where to put your file.

A bunch of the dialog boxes that you pop up from the server and reports menu have blank or bogus titles.

When you select a student using the keyboard, the roster should scroll if necessary.

The Browser.pm module is in severe need of refactoring. It was only supposed to contain GUI-related stuff, but it's become a messy mixture of lots of different stuff.

In the dialog box for adding a new category, there's some extra help that the TermUI gives when weights are enabled, and the GUI should give that help, too.

When creating a new file, there needs to be a way to set whether weighted grading will be used. (TermUI does this when you add the first category, but don't do it that way in the GUI, too awkward. The decision really needs to be made when they open the file.) There is no warning when you create a new file on top of an old one.

In FileDialog: (1) Inputs should look sunken, not raised. (2) When creating a new file, the focus should be in the right place to start with. (3) There's no easy way to set activeBackground; in KDE, the default is lame, and this causes buttons to become totally black when the cursor is over them.

Missing functionality: change order of assignments or categories, delete a category (after deleting all assignments).

The assignments listbox should start out positioned at the end, not the beginning, when you select a new category. This seems to be pretty difficult to do, however! Selecting and then unselecting the last item doesn't work, and it doesn't have set() and get() methods like a Scrollbar widget.

The dialog box for creating a new file doesn't give all the help that the text-based version does. It doesn't check whether the days of the week are legal. Actually, I think I can do this by building new types and options in to the Input class.

Should allow the user to view dropped students. Have a note in `add_or_drop()` about this.

Input class needs 'date' type.

Resizing needs to be handled better. Some widgets have hardcoded sizes, which I'd like to avoid.

text interface

The terminal-based interface should figure out the width of your terminal window, rather than defaulting to 80 columns.

Emacs-style tab completion is goofy in a case where one student's name is the same as the beginning of another's (Ng and Nguyen) — you need to type the underbar!

In the statistics report, there is no user interface to say if you want stats within a category, and the report routine doesn't support that, either. The formatting of the report could be improved a lot.

When selecting a student, should be able to type in first name. The display is also kind of messed up during the process of selecing a student. And the underbar should be hidden.

Give some message when they make a choice that's not on the menu, or not allowed because there is no open file, etc. In main loop, can make hashes to control the behavior of the commands.

Menus should allow you to type multiple characters in a row and then hit return so you go into a sub-menu. (Don't just take keystrokes without hitting return, because some commands like "revert" are more than one character.) Beep when they do an illegal command, and flush the rest of the string out.

13 Bugs

general

Some errors should be checked for when reading in the gradebook file: two students with the same key; mangled syntax in parameter lists; grades for a student who isn't on the roster; more than one assignment in a single-assignment category.

When you create a new category, it should require you to give a weight if you're already using weights, or prohibit you from doing so if you aren't already. This should really be selected when you create the file.

With some files, I couldn't do any client-server stuff until I touched the client-server settings. It didn't matter that I wasn't even changing the settings.

The time zone correction in `ServerOG.cgi` when downloading grades may be incorrect if the client and server are in different countries that have different definitions of daylight savings time. It will also not work correctly if either time zone is a non-integer. It might make more sense to have the time zone as part of the Spotter file tree, so Spotter can do the time zone correction when recording the grades.

On Windows, the preferences file is always saved in My Documents. Since the My Documents folder could be on the C drive or the D drive, we search those drives for the directory. This should work for must users, but if it doesn't find My Documents on C or D, it just silently fails to create the preferences file, which is a bug. I don't know much about Windows, but I think preferences should probably go in Documents and Settings.

terminal-based user interface

The "u" command for uploading grades in the terminal-based interface is untested, and may not actually work.

GUI

It's not possible to resize the window and cause the list of students to be resized.

Sometimes a dialog box pops up as a small blank square instead of having the right size, shape, and contents. This may be triggered by certain WM actions such as tabbing the window with another window in Fluxbox, or moving the window to a different virtual desktop. The only workaround currently is to restart the program.

Certain network operations can freeze, e.g., uploading grades. This seems to be because the network connection is flaking out. In this situation, it should time out with an error message.

When you select a student with the keyboard, and that student's name is scrolled off the top or bottom of the list, the list should scroll automaticaly to make that student visible.

When you change your client/server settings, you have to exit the program and start again before the changes take effect.

There is some GUI stuff relating to passwords in BrowserData::new().

I've run into one problem that occurs only on a particular system running FreeBSD 4.8 (and doesn't occur on other FreeBSD 4.8 machines!). Right-justified grade entries are displayed incorrectly – they're positioned too far off to the right, so you can't see them; this is the reason I changed the default to left-justification, and added an item to the preferences menu.

The dialog box for setting standards is too tall when you use ABCDF+- grading.

When you change the name of an assignment, the popup window that shows the category above it is temporarily displayed incorrectly.

I wrote stuff into the Browser.pm package at the outside level, assuming there would only ever be one window open. This needs to go into BrowserWindow. Likewise, what happens with stuff like `grades_queue()`, which is inside a `BEGIN{}` block? Will its variable get properly initialized when I make a new BrowserWindow object? Will each BrowserWindow object really have its own `%queue`?

14 Programming Stuff

version history

See freshmeat.net for a history of what changes have been made to the code over time.

internals

You can use `make internals` to make the file `internals.html` from the POD documentation.

contributing

If you'd like to make improvements to the software, please let me know! You can find my current e-mail address at `www.lightandmatter.com/area4author.html`. I haven't set up OpenGrade in CVS, but I would be happy to do so if other people were really started doing a lot of work on the code. For now, I think it would be easiest if contributors e-mailed me patches. My current wish list/to-do list is given in the Improvements and Bugs sections above. I would rather not make any changes to the code that would add more dependencies on libraries, complicate the installation process, complicate the user interface, or break cross-platform compatibility; if the feature you want would violate one of these criteria, maybe it would be better to contribute a separate utility to perform the function you have in mind.