ELIP
The Emacs Learning Instruction Program
Installation and Usage Manual

Beta Release 0.611

May 24, 2004

Bob Newell
Santa Fe, New Mexico, USA

Contents

1 What is ELIP? 5

2 Requirements and Compatibility 6

3 Further Information on Learning Methods 7

4 Keystroke Conventions 7

5 Installation 8

6 Creating a New Study Item Database 8

6.1 Creating the Database Files 8

6.2 Creating and Inserting Database Study Items 9

6.2.1 Direct EDB Item Addition 9

6.2.2 Adding Items from a Text File 10

6.2.3 The Ellipsis Method 11

6.2.4 Keeping Your Place With Bookmarks 13

6.2.5 Ellipsis Marking: As Easy as 1-2-3 14

6.2.6 Editing Question Items 15

6.2.7 Editing Question Items While Testing 15

6.3 Sorting the Database 16

6.4 Shuffling New Items 16

6.5 Reversing Questions and Answers 17

6.6 Exporting Questions and Answers 18

6.7 Embedded Elisp Code 18

7 Learning from a Study Item Database **19**

7.1 Loading an ELIP Database 19

7.2 Four Ways to Learn 20

7.2.1 Interval Learning 20

7.2.2 Leitner Learning 20

7.2.3 Text-Mode Learning 21

7.2.4 Flashcard Learning 22

7.3 The Learning Process 23

7.4 Automated Learning in Interval Mode 23

7.5 Learning New Items in Interval Mode 24

7.6 Learning Old Items in Interval Mode 25

7.7 Learning in Leitner Mode 25

7.8 Learning in Text Mode 26

7.9 Learning in Flashcard Mode 26

7.10 Resetting the Database 27

7.11 The Pileup Problem in Interval Mode 27

7.12 Delaying or Advancing Item Due Dates 28

7.12.1 Delaying Items 28

7.12.2 Advancing Items 29

7.13 Changing the Number of Items Learned at Once 29

7.14 Checking a Database for Trojans 30

8 Reports **31**

8.1 Global Reports 31

8.2 Individual Item Statistics 32

8.3 Interval Mode: Workload by Date Report and the Catchup Issue 33

9 Command Summary **34**

10 .emacs Considerations **35**

11 The Interval Algorithm **36**

12 Release Terms and Disclaimer of Warranty **36**

13 Contact and Mailing List **37**

14 Acknowledgements **37**

Notice

ELIP is *beta* software. This is a euphemism for "not fully tested." As of the date of this manual, the user base isn't large enough to ensure quite enough testing and cleanup.

While I use ELIP often for my own study and learning, your mileage will vary. You should use ELIP with at least some expectation of encountering bugs or errors.

1 What is ELIP?

ELIP is the "Emacs Learning Instruction Program." It is something like a super flashcard system for presenting and learning nearly arbitrary study material. ELIP incorporates multiple methods of learning which may be used in any desired combination.

One method, as described later, is based on the concept developed in the *Supermemo* system of "interval recall"— the presentation and repetition of material at just the right intervals to greatly enhance long-term retention. It is amazingly effective. You need to try it out to be convinced; there is some effort involved and some persistence required, but the returns are substantial.

A second method, further described in a later section, is a variation of the *Leitner* flashcard method, which is based on repeating a question until it is answered correctly a number of times in succession.

Yet another method is adapted to the memorization of text material. The text to be memorized is presented with part of it left out, that part to be supplied from memory. The length of the omitted part is successively increased or decreased based on the quality of the answer supplied by the student.

A final method is "plain old flashcards" which simply presents questions and answers in a simple manner, without keeping track of results.

I developed ELIP because it seemed that Emacs was a practical vehicle for both development of study material and the actual learning of it. Emacs fanatics will understand. If you are a Windows Solitaire person, please continue to give Bill your money. ELIP may not be for you.

If ELIP leads you to an appreciation of these various types of learning, you might want to delve further. Other commercial, shareware, and free software products are available which provide rich feature sets and much pre-written study material. (Unfortunately, many of these are Windows-only.)

2 Requirements and Compatibility

You must, of course, be running Emacs, and be reasonably familiar with its use.

ELIP is known to work with Gnu Emacs 19.34, 20.7.2 and 21.2.1 on Linux 2.0.x, 2.2.x, and 2.4.x kernels, and with the same versions of Gnu Emacs on Windows 98 and Windows XP systems. It will probably work with various versions of Gnu Emacs on a Macintosh, on Windows 95, and with Xemacs, but I don't have these platforms available and have not tested on them. If you do try out these or other platforms, please let me know of your results. I doubt very much if ELIP will work with Windows 3.1 or DOS versions of EMACS.

You also must have a working installation of EDB, the Emacs database. You can obtain the last version, 1.21, here:

`ftp://theory.lcs.mit.edu/pub/people/mernst/edb/edb-1.21.tar.gz`

This link has been valid and stable for a number of years, but if you run into difficulty let me know and I'll document an alternative source (even if it's my own server).

I've provided a *critically important* substitute for `db-time.el` in the EDB package. Please copy this over the `db-time.el` that comes with EDB. Do this right after unpacking EDB but before you build or install it. My version has important patches; ELIP may fail to run without these patches. If you have already installed EDB, please copy my version of `db-time.elc` to the directory where you've installed EDB. If you run into any problems, byte-compile my version of `db-time.el` and copy the resultant `db-time.elc` to the directory where you've installed EDB (this is often `/usr/share/emacs/site-lisp` on Linux systems; but please check).

I can't provide tutorials on installation and usage of EDB (much less of Emacs itself). If you are not already an EDB user, please take the time to familiarize yourself with this extraordinary piece of software. It's a bit idiosyncratic and takes a little effort, but it's worthwhile.

Requiring pre-installation of EDB, with patches no less, creates somewhat of an "entry barrier" to ELIP, and I have thought more than once about rewriting ELIP to stand alone. I ultimately decided, however, that EDB offers so many features and conveniences that keeping ELIP in its present (EDB-dependent) form is still the best choice. The time you spend setting up and learning about EDB will be well repaid.

3 Further Information on Learning Methods

One of the learning types featured in ELIP is based to quite a measure on the Supermemo concept. You can find out everthing there is to know about Supermemo here:

`http://www.supermemo.com`

Supermemo is an excellent learning system which has been around for quite a number of years but hasn't seemed to attract massive mainstream attention yet. Again, please learn about Supermemo on your own; it's not something I can or should try to teach here. ELIP does not pretend to compete directly with Supermemo.

Similarly, the Leitner method is well documented on the web; a search for "Leitner" and "flashcards" will yield a great deal of interesting material, including some commercial products.

I have not seen a lot of information about the "text mode" learning method, but something very similar is included in a commercial product called "Scriptorian" which seems mostly used for learning passages from religious texts. This product is also easily found on the web.

4 Keystroke Conventions

In this manual, I use the following standard keystroke representation conventions:

`RET` is the **return** key.

`M` is the **meta** key, usually **escape**. So, `M-x` is "escape-x" or the escape key *followed* by the "x" key.

`C-` stands for the **control** key. So, `C-c` stands for the control key and the "c" key pressed at the *same time*. `C-cx` would stand for the control key and the "c" key pressed at the same time, *followed* by the "x" key.

As an example, `M-x elip-learn RET` means: press the escape key, then press the "x" key, then type "elip-learn" then press the return key.

Many ELIP commands have shortcuts, so the command above could also be actuated by typing `C-cl` or control and "c" at the same time followed by "l".

5 Installation

Put all files from the ELIP distribution in your preferred EDB database location, or in an entirely new location dedicated to ELIP; it's up to you. Whatever you do, it's highly recommended (although not mission critical) that you keep your ELIP databases together. At any rate, I use `/data/db` for this and I'll use that as an example. If you are particular about mixing source and data (and I don't blame you if you are; it's bad practice in general), move `elip.elc`, `elip.el`, and `elip-sort.el` to somewhere on your Emacs load path. A typical Linux location is `/usr/share/emacs/site-lisp` but your installation may be different.

Depending on your release of Emacs, you might want, or need, to re-byte-compile `elip.el`. To do this, simply load `elip.el` from the directory where you put it or moved it, and byte-compile as you would any other Elisp file. *Don't* byte-compile `elip-sort.el`. Due to a bug in EDB this won't work and will cause all sorts of problems with ELIP (it won't run, which seems like quite a problem indeed).

This completes ELIP installation. You're ready to start learning!

6 Creating a New Study Item Database

6.1 Creating the Database Files

The files `elip.dat`, `elip.fmt`, and `elip.dba` are templates for new database files. Please don't modify these in any way. If you modify these files, it is likely that new databases will not work correctly.

To create a new database, you **must** always first load the template database:

```
M-x db-find-file RET /data/db/elip.dat RET
```

If you don't do this, you won't initialize important ELIP variables, ELIP commands won't be loaded, and nothing will work, which is not quite what we want.

Then create your database with

```
M-x elip-make-database RET
```

or `C-ck`

You will be prompted for a one-word database name. Name the database appropriately, such as "history" or "esperanto" and press return, and the database will be created. You will

then be transferred to the buffer for the new database. You are then prompted for a brief description of the database. This should be just a few words, such as "Esperanto Grammar" or "American History." Enter your description, and press return.

There is then just one more question to answer. You will be asked if items should be learned randomly (which is the default; the option is sequential). If random item presentation is what you want, answer "yes"; if you prefer sequential presentation, answer "no."

A little elaboration on this point is in order. Items are learned in groups of (usually) 20; if you answer "yes," random presentation will be *within* these groups but will not extend *beyond* each individual group. (For full randomization, see the *shuffle* option described in a section below.) Some material presented randomly or randomly in groups will work just fine. But material in which each item or group of items builds on material learned in earlier items should be presented sequentially. In such a case, you should answer "no." In particular, the *text* learning mode may often be better in sequential mode.

In any event, you're now done. Your new shell database will have been created and you'll have been brought to its buffer. The database contains just a single placeholder item. You will delete this later, but *not* until you've added some "real" items to the database.

Please note that your new database will be created in the same directory in which the template files `elip.dat`, etc., are found. This seemed to be pretty intuitive and logical so I didn't offer an option for other locations. If you wish to move your new database, be sure to move all three files, namely `databasename.dat, databasename.fmt`, and `databasename.dba`, where "databasename" is the one-word name you gave to your database when you created it.

6.2 Creating and Inserting Database Study Items

Creating learning or study items is, of course, the crux of the matter. This is easier with a certain amount of skill and experience, but you should by no means let that stop you. You can create useful items right away and modify them any way you want later on. The Supermemo web site contains some valuable advice about item content. I won't repeat that here; I'll just show three ways of introducing new material into your database.

6.2.1 Direct EDB Item Addition

The most conceptually simple method of adding items is the native EDB method, using the `'a'` command while in View mode. (`C-c C-c` switches to View mode; you'll see the cursor in the upper left-hand corner at that point.) A new database item is added *before* the current item. This may or may not be what you want, so be aware.

Then you just press the TAB key to move to the question field. Type in the question, then press TAB again to move to the answer field. Type in the answer. It's best to leave the date field, which tells when the question should next be asked, as it stands. It initializes to the current date, but for new items, the date is actually ignored. Pressing C-c C-c will then ensure you are back in Database View mode.

Add as many more items as you wish in the same manner, periodically saving the database with C-c C-s to ensure you don't lose your work.

While intuitive and easy, this method can become tedious, and unless you do a lot of moving around among the record positions, the items may end up in reverse order and in the front (or even middle) of the database. You can control this with the EDB jump command (`j`) but there is a better way to add large numbers of items.

6.2.2 Adding Items from a Text File

This is the method you are likely to use most often. Open another buffer, or file, with any descriptive name you wish. In this method, you simply type in questions and answers, in matched pairs, in the new buffer. Each question must have Q. starting in the first column: that's capital Q, period, space. Each answer must have A. starting in the first column. A blank line between question and answer is not required (or recommended) and a blank line between pairs is optional. Here is a not especially inspired example:

```
Q. Should you delete the sample item prior to adding new items?
A. No.

Q. Where can you find information about learning methods?
A. On the web.
```

When you are done, write the file out to disk. Then issue the command M-x elip-import RET or C-cm and when prompted enter the name of the file you've just created, and press return. The new items will be imported to the database, in the order written, at the end of the database. There is no duplicate checking, so if you import the same file or items twice, they will appear twice. (Create a *new* text file each time; don't append to an old one.) You can always recover a complete text file of all item text by using the elip-export or C-cx command.

Remember: ELIP commands are only available if you've already loaded an ELIP database. If EMACS complains about an ELIP command being unknown, load up the sample database elip.dat or any other ELIP database of your choice. See the section below on "Loading an ELIP database."

There is little error checking of any kind, so you are advised to *back up* your database before doing an import. And, after importing, do some checking of your own!

Special note about creating items for "text-mode" learning: we're getting a little ahead of ourselves here, but a special format is recommended for a database that will be studied in "text" mode. Since text-mode is all about memorization of a passage of text, the concept of "question" and "answer" is not the same. When a study item is learned in text-mode, the answer is ignored and only the question is used. However, you still must supply an answer when you make up the study items, but that answer might as well be blank. Here's an example:

```
Q. In the seventeenth century, our foreparents came to this planet with the
goal of establishing a free state based on unsound business principles and
the highest standards of immoral conduct. (attributed to an anonymous
bard)
A.
```

6.2.3 The Ellipsis Method

This is an idea, and a great one, borrowed from Supermemo. It allows making items out of arbitrary text material, such as a web page or a text document. The only requirement is that the original material has to be in plain text format, that is, something that Emacs can load into a text buffer.

However you do it, get the text you want to learn into an editable buffer. You can run W3 and cut and paste, load up a text file, or extract text from a word processor and put it in a file; there are many options.

Within Emacs, start to read through the text. When you find a section of interest, mark it on each side by inserting a pair of triple curly braces— `‘{’ and ‘}’` as in this brief example:

```
The earth is quite far from the sun.  {The distance from the earth to the sun
is about 93 million miles.  This is called an Astronomical Unit (AU).}  It is
often used for other purposes.
```

Be careful not to mark too much text at once or you will end up with long, difficult to read learning items. Within any text file, you can set up as many pairs of curly brackets as you wish. Each such selection will then generate one or more actual question-answer items.

Now, as you go, or later on if you wish, go back and make "ellipsis" style questions and answers from each text selection. You do this with triple square brackets, and an example is worth a thousand words. In the selection above, suppose we mark it like so:

```
{{{The distance from the earth to the sun is about [[[93 million miles]]].
This is called an [[[Astronomical Unit (AU)]]].}}}
```

When processed as later explained, this marking will generate the following question and answer items:

```
Q. The distance from the earth to the sun is about [...].  This
is called an Astronomical Unit (AU).
A. 93 million miles

Q. The distance from the earth to the sun is about 93 million miles.  This
is called an [...].
A. Astronomical Unit (AU)
```

Essentially, you've created a fill-in the blanks set of questions. Beware: if you select a lot of text with curly brackets and within that text, mark a lot of words or phrases with square brackets, you'll generate quite a number of learning items, perhaps more than you wished. Experiment and find the right balance.

When you're done marking the text, or you've marked as much as you wished, go back to the point in the text buffer where you started working. This might be the beginning of the file; go there with `M-<`. If it's a previously edited file, and you're working on it in stages, go back to the point where you began work in this session. (See the next section for using bookmarks to help manage this.)

Now give the command `M-x elip-scan-text` or `C-cc`. You will be prompted for a buffer name to receive the generated item question and answer text. Type in a name and press `RET`.

It is advisable to go to the newly generated buffer and check the items over to see if they're what you want. Edit as you see fit. For instance, in the example above, you might make an improvement:

```
Q. The distance from the earth to the sun is about 93 million miles.  This
is called an [...] (name and abbrevation).
A. Astronomical Unit (AU)
```

This makes it clear that in the answer you're expected to give both the full name and the abbreviation of the correct term.

When you're satisfied, write the new buffer out to disk with `C-x C-w`, typing in the file name when prompted. Now, go back to your ELIP data file. Use the previously discussed `M-x elip-import-file` or `C-cm` command to import your newly generated learning items. This is a very fast method of creating items and works extremely well in a practical sense.

Making backups of your database is always a good thing to do *before* importing or making major changes.

Note that the ellipsis method is really not compatible with "text-mode" learning, as will become apparent later on. The ellipsis method works fine with any of the other methods, however.

(Previous releases of ELIP used single, rather than triple, brace and bracket markers. However this was too easily confused with actual text content; hence the shift to triple markers which are much less likely to be found in most text passages.)

It's worth repeating: ELIP commands are only available if you've already loaded an ELIP database. If EMACS complains about an ELIP command being unknown, load up the sample database `elip.dat` or any other ELIP database of your choice. See the section below on "Loading an ELIP database."

6.2.4 Keeping Your Place With Bookmarks

When creating ellipsis style questions and answers, this is important, so it's worth repeating. In your text buffer, when you give the `elip-scan-text` or `C-cc` command, the scanning is done from the *current cursor position* forward to the end of file. This allows you to prepare long texts in batches, but it also means you must pay attention. Just remember to have the cursor in the right place when you give the scan command; you don't want to either miss items or generate them twice.

ELIP provides a simple bookmark facility for this purpose (among other purposes).

You can use the `M-x elip-set-bookmark RET` or `C-cb` command to help keep your place in a file. This command "remembers" a point in your text file. It would be a good idea, at the end of an ellipsis editing session, to put a bookmark at the place where you left off.

Then, next time, you can use the `M-x elip-goto-bookmark RET` or `C-cj` command to return to where you stopped last time. The bookmark remains valid as long as you don't edit anything *above* the bookmark. After you edit *downwards* in the file, you can pop back up to your bookmark and begin your elip text scan. After that, just re-set the bookmark at the

point where you left off work durring this latter session.

A little explanation: a bookmark is just a simple position marker, counting the number of characters from the start of the file. So, if you edit *above* the bookmark that count is thrown off, and the bookmark will be invalid. Editing *downwards* or *below* the bookmark won't disturb the position count.

6.2.5 Ellipsis Marking: As Easy as 1-2-3

Ellipsis style questions and answers are a very powerful learning method. "Pure" EMACS facilities allow for relatively easy marking of text with triple braces and triple brackets as required by this method. However, ELIP includes an extension to provide shortcut keys to make marking large stretches of text quite a bit easier. This extension is available as long as ELIP commands have already been loaded, as described just above.

Step one: when you find a region of text that is of interest for generating one or more question and answer pairs, mark it off with the EMACS "mark" and "point" facilities. That is, at one end (beginning or end, it makes no difference) of the interesting text, set the EMACS "mark" with `M-x set-mark RET` or, on most systems **control-spacebar**. (Some systems use **control-shift-2** instead.)

Now, move the cursor to the other end of the text of interest. On some systems, you don't even have to set the "mark" — you can just hold down the left button of the mouse and highlight the text.

Then, give the ELIP command `M-x elip-mark-area` or `C-c1`. Everything else except the text of interest will now disappear, and you'll notice that the triple curly brackets appear at both ends. That's step one.

Step two is to mark off short portions of text to be replaced by the ellipsis; in other words, to create a question and answer item as described above. Set the "mark" and move the cursor as before, or use the mouse. Now give the ELIP command `M-x elip-mark-question RET` or `C-c2`. Square brackets will appear at both ends of the small text area in question. Repeat this process as many times as you wish with different short text portions.

Step three is just the ELIP command `M-x elip-done-marking` or `C-c3`. The whole text file will reappear and you go back to step one, to find and select new text areas of interest.

Save the file, position the cursor to the starting point of your work, scan it with `M-x elip-text-scan RET` or `C-cc` as described in the previous section, check it over, and import it to your database with `M-x elip-import-file RET` or `C-cm`. This is all described in more detail in the previous section.

You'll find that with minimal practice, the 1-2-3 method is a very fast and effective way of generating ellipsis style questions.

6.2.6 Editing Question Items

Any item in the database can be changed at any time by using the EDB editing facilities. Simply go to the item you wish to change, and `TAB` to the appropriate field (question or answer; you should not be changing the other fields unless you really know what you're doing, and why you're doing it).

When you're on the appropriate field, just edit as you wish and save the results as usual with `C-x C-s`. Similary, you can delete entire items if you wish. In general, though, consistency is an important part of this learning methodology, and making too many changes or deletions will not be beneficial.

6.2.7 Editing Question Items While Testing

It does happen at times that, when you're reviewing old items or learning new items, you may see that there is an error in the item or that the item could use some changes or qualifications. ELIP has a feature to allow editing of items in the middle of a learning session. While we're getting a bit ahead of ourselves, this option is best documented here. If any of this doesn't make sense quite yet, read the rest of the manual and then return here.

During a testing session, if you find an item that really needs immediate editing attention, when you are prompted for an item score, enter the letter `e` instead of a number. This will tell ELIP that you want to edit this item right away. Your request will be acknowledged, you will be re-prompted for an item score, and after you enter the score, you will be put in EDB edit mode for the item in question.

When you are done editing, enter either the keystrokes `Esc C-c` (that's escape control-c) or the command `exit-recursive-edit` to return to testing exactly where you left off.

Beware of certain "features" of this process:

- Timers related to how long it takes to answer questions are suspended while you are in edit mode.
- Your edits are automatically saved so be careful what you do.
- Edits are done in EMACS "recursive-edit" mode. You can do anything you want in this mode, including get in a lot of trouble or irretrievably harm your database. Don't

do anything except edit the single item in question! If you start another ELIP session, edit other items, or do other unnatural things, there is no telling what might happen. Of course, please have a backup for your database just in case something goes wrong.

- Be sure you exit this mode as described above with `Esc C-c`. You don't want to be building up layer after layer of recursion within EMACS.

6.3 Sorting the Database

There are several reasons that you might wish to sort an ELIP database according to a series of special sort rules. ("Ordinary" sorting could be problematic and is best avoided; ELIP builds in a few safeguards to ensure that if you do sort the database, you get a sensible result.)

If you've added new items manually somewhere in the middle of a database, or scattered them throughout the database, you might wish to clean this up. Or, if you want to preview new items, it might be nice to have them in one part of the database. Finally, you might want to review old items in due-date order, looking at the ones that will come up the soonest before the ones that come up later on.

The command `elip-sort` or `C-cs` automates all of this. When you give this command, the database gets sorted as follows:

1. New items go to the beginning of the database.
2. New items come in the order they currently appear in the database.
3. Old items go after new items.
4. Old items will appear in order of due date, with items due sooner before those due later (which implies that overdue items will come first of all).
5. If items have the same due date, their current order in the database is maintained as-is.

Normally, no harm can come from sorting a database in this manner. Even if the database has been previously *shuffled* (see next section), the shuffling of new items will not be affected. But if you are unsure, as always, make a backup first.

6.4 Shuffling New Items

If a large number of new items exist, or especially if they have been entered all at once, when you give the command `elip-learn-new` or `C-cn` they will be presented in groups of 20 at a

time. This is not a problem except that the groups of 20 that appear earlier in the database will always be presented *before* groups of 20 that appear later in the database. This may or may not be what you wish to happen. (Of course, if you are using the default *random* option, the items *within* each group of 20 will be presented in an arbitrary order. But this randomization doesn't extend *beyond* each individual group of 20 items.)

In order to completely randomize the order in which any (large) number of new items is presented, use the command `elip-shuffle` (no shortcut keys). This command will do two things:

1. It will completely reorder the entire database in a random manner.
2. It will then sort the database (as described in the previous section).

The end result is that all new items will be sorted to the top of the database, but they will be in a completely random sequence. Old items will be sorted after new items according to due date. New items will still be learned in groups of 20, but those 20 items will be effectively taken at random from the entire collection of new items.

Use this feature with some discretion. Some curricula work better if certain (easier) items are learned before other (more difficult) items. Complete randomization of hundreds of items may not give you what you really want. This may be especially true for databases meant to be studied in text-mode.

6.5 Reversing Questions and Answers

In some types of learning, especially language learning, having questions and answers exist both in the normal and reversed form is highly useful. (Think of language flashcards, where you can present either side and try to supply the answer on the other side). ELIP contains a facility that avoids the need to write out each question and answer pair twice.

This is what you do: create and save a set of questions and answers in a text file, as described above. Then use the command `M-x elip-reverse RET` or `C-cv`. When prompted, enter "f" to reverse items from a flat file (text file). Supply the name of the text file you've created, and a name for the reversed questions and answers (in which the answer becomes the question and the question becomes the answer). That's it. The reversed file will be created. Check it for correctness.

You can now use the `M-x elip-import RET` or `C-cm` command twice, to import each of the files (normal and reverse). This will place the normal questions all together, followed by all of the reversed questions. If you want to mix them all up, use the `elip-shuffle` command described above.

You can also directly reverse items which are already in a database without the intermediate export step. Go to the buffer in which the database in question is open. Give the `M-x elip-reverse RET` or `C-cv` command. When prompted, type "c" to reverse items from the current database. Then give the name of the export file. You're done. Be sure to check the export file for correctness before importing it with the `M-x elip-import RET` or `C-cm` command.

I deliberately do *not* provide a facility for automatic item import. You really do need to make at least a quick manual check of the item file. This can save you a great deal of grief from a possibly trashed database. And, always back up your database before making major changes, even if you are sure they are safe.

6.6 Exporting Questions and Answers

At times you may want to dump out all the questions and answers in a given database. To do this, simply give the command `M-x elip-export RET` or `C-cx` and then supply the name of an output file when requested. The database items will be written out to a buffer in the text file (Q. and A.) format. You must save the buffer to disk on your own, after looking it over and making any changes you may desire.

If you are trying to start over on a given database, exporting all the questions and importing them to a new database isn't necessary. See the section about resetting for a more effective method.

6.7 Embedded Elisp Code

Here we introduce a powerful feature that is also a great way to get into a lot of trouble. If you hang your session, blow away your database, or cause geomagnetic field fluctuations, you have been pre-warned.

Arbitrary Emacs Lisp (Elisp) code can be placed at the end of any question or at the end of any answer, and when this code is encountered, it will be interpreted by Emacs. Anything below a line that has five hashmarks at the beginning, like this:

```
#####
```

will be interpreted as Elisp code.

One purpose of this could be to present audio or graphic material. For instance, the following (in the right environment) would show a picture as part of an answer:

```
The A-minor scale is depicted as follows:
#####
(shell-command "xv aminorscale.jpg &")
```

Obviously there are many possibilities and many perils. This is a feature to be used with caution. It is easy to hang Emacs by, for instance, launching a `tty` program that expects keyboard input, by forgetting the "`&`" to spawn processes asynchronously, and so on. You can casually blow away ELIP variables and counters, and commit any mayhem which Elisp allows (which is virtually without limit).

Of course, there are Bad Guys out there, and although unlikely, you *might* some day receive an ELIP database from someone which has harmful or "Trojan" Elisp code in it. To deal with this possibility, a form of trojan protection has been added; it doesn't really check anything but it does at least give the student the option to deny execution of Elisp code, or dump the code out for examination. See the separate section below, where this is described in more detail. But there is a simple "moral to the story" which is: only use ELIP databases that come from trusted sources.

7 Learning from a Study Item Database

7.1 Loading an ELIP Database

To learn from an ELIP database, load it into Emacs with

```
M-x db-find-file RET
```

after which you will be prompted for the database name. Be sure to enter the full name, including the "`.dat`" suffix. The database will appear and you are ready to go.

A word of caution: while you can work with as many ELIP databases as you wish at any one time, and even multiple instances of the same database if you so desire, don't try to do question and answer learning from more than one at a time. (It may actually not be possible but don't do it anyways!) This will mix up ELIP's internal variables and the results are unpredictable, and very likely not at all what you want. Learning from a single database at a time is in any case likely to be a lot more effective and efficient.

7.2 Four Ways to Learn

As briefly discussed in the introduction, ELIP provides four different ways to learn. You can use any one or more of them in any combination you wish (although not every combination makes sense).

What follows is a brief description of each learning method, followed by directions on how to operate each method. Some repetition is inevitable, so bear with me.

7.2.1 Interval Learning

Interval learning is based on Supermemo concepts. The basic idea is that items are presented for study and review at intervals which are timed to ensure best long-term learning of the item content.

In this method, new items are presented in groups of (usually) 20, while review items are presented as they come due on a daily basis. In any given daily study session, each item is repeated until a "passing" score is obtained. (This is usually 4 or better on a 0 to 5 scale.) When an item is finally "passed" it is scheduled for review based on how many times it has come up for review, and how well the item has been learned. Quality of learning is based on percent correct answers (this differs from Supermemo). An item that has been answered incorrectly numerous times will be scheduled for repetition sooner than an item that has been answered correctly most of the time.

Repetition intervals increase with time. If an item is brand new and is answered correctly right away, it will be repeated in 4 days. If it is answered correctly after 4 days, it will be repeated in 7 more days, and so on until an item will not be repeated for many months.

However, if a very low score is obtained when doing a review (usually 2 or lower on the 5 point scale) the item reverts to an effectively "new" status and the repeat cycle starts from the beginning, with frequent repetitions.

This method is extremely effective and you really need to try it out to see just how effective it can be, especially with language learning.

7.2.2 Leitner Learning

Leitner learning is an advanced form of flashcard learning. It is based somewhat on Dr. Leitner's book, "So Lernt Man Lernen" which for better or worse is only available in German at present. (I live in the US, and ordered a copy online from amazon.de. I paid twice as much for *surface* shipping than the cost of the book itself!)

Repetition intervals are not used in the modified version of the Leitner method used in ELIP. Instead, as implemented in ELIP, 20 learning items ("flashcards") are chosen from the database (or as many as are available if less than 20 are in the database). These are all placed in "slot 1" and a single run-through of these items is done. Each item that gets a passing grade moves to "slot 2" (think of these as pigeonholes in which you put cards or letters). In the next runthrough, correct items move to slot 3; wrong items that were in slot 2 go back to slot 1.

As the run-throughs continue, items work their way up or are moved back depending on the quality of the answer. "Passing" items (usually with a score of 4 or 5 on a 0 to 5 scale) always move up one slot. Items with (usually) a score of 3 move back one slot. Items with (usually) a score of 2 or less are moved all the way back to slot 1.

When the last pigeonhole is reached (there are usually 5 but you can change this to whatever you like), and the item is answered correctly (i.e. with a passing score), the item is considered "complete" and removed from active use for this learning method. A new item, if available, is then placed in slot 1 in order to make up the full complement of 20 items.

Eventually all items will be completed by moving out of the top slot, and the database will be considered "learned." Notice how this works: if there are five slots, you need to answer an item with a passing score five times *in a row* before the item is "complete." Items that are missed fairly often might get repeated many times before completion.

There is no need to try to learn the entire database at one sitting; for large databases, this could take quite a long time. Most databases will be learned over a period of days, comprising a number of learning sessions. ELIP tracks your progress and lets you pick up in a new session where you left off in the previous session.

When the database is finally completed with this method, it must be *reset*, as later described, before Leitner learning can be repeated.

This method is very thorough but I find it can become quite tedious at times. I tend to prefer interval learning, but styles and preferences will vary, and Leitner learning provides a very comprehensive drill in an "over-learning" style.

Leitner and interval learning can be done at the same time, if you so desire. The status of a given learning item is maintained separately for each mode.

7.2.3 Text-Mode Learning

Text-mode learning is usually done on a database which is prepared specially for this mode of study. Such a database is generally suitable only for text-mode learning and not for interval or Leitner learning, although the software will allow you to mix these modes if you wish.

Text-mode learning is generally applied to the memorization of passages of text. This will likely be most often used for religious texts but any text material can be used at will.

In text-mode learning, 20 items are chosen from the database (or as many as are available in smaller databases). If the items are at all lengthy, a smaller increment might be chosen, such as 10 or even 5 (see command descriptions below to see how to do this).

Each item is then presented with some of the text removed. It's the job of the student to supply that text from memory. When an item first comes up, it's shown with 90 percent of the text intact (which means 10 percent has been removed). The removed text is replaced, letter for letter, by underscores. If the item is answered with a passing score (usually 4 or 5 on a 0 to 5 scale), the item will be presented in the next run-through with 80 percent of the text shown (20 percent removed). As the item continues to get a passing score, 10 percent of the text is removed each successive time, until only 20 percent of the text is shown (80 percent removed). If the item is passed at that level, it is considered "complete" and is replaced with an untried item (if available) at the 90 percent showing, or starting level.

When an item is missed at a score of 3, the percentage of text shown is *increased* by ten percent for the next repetition. If the item is missed at a score of 2 or less, it is set to "start over" at a 90 percent text-shown level.

When all items have been successfully answered at the 20 percent shown level, the database is considered "learned" and would need a reset to do any more study in text-mode.

Percentage of text shown is calculated from the number of words in the study item. All words are included in this count, including articles such as "a" and "the." (There is room for improvement here in a future release!) Words are removed *at random* so one repetition at an 80 percent level will likely not be the same as another repetition at an 80 percent level.

This method is quite effective in promoting memorization of text passages, even passages of considerable length. Like the Leitner method, it can become somewhat tedious at times. Learning hundreds, or even dozens of long text passages by this method will probably take considerable time; but it will likely be much faster than any other method of memorization.

7.2.4 Flashcard Learning

Flashcard learning mode is simply the old-fashioned method. 20 items are chosen at random from the database (or as many as are available in smaller databases) and they are presented as usual. Each item will be repeated until a passing score (usually 4 or 5) is obtained. In this informal mode, no records or statistics are kept. More details on some of the options for this mode of learning are given in a section below.

Flashcard learning can be done independent of any other learning mode. It's something that

can be used for quick review or if you just want to "cruise" a little.

7.3 The Learning Process

Whatever the learning method, and whether you are learning old items or new items (both of which are described below) the process is the same. For each item, you will be taken to a buffer which shows nothing but a question. Answer the question mentally and then press `RET`. The answer is then displayed. Score the answer, enter the score, and press `RET`.

Scoring is done as follows (this scale is similar to the one used in Supermemo):

0 You are clueless, you have no idea of the answer

1 Wrong answer and not really familiar

2 Completely wrong, but when you saw the answer it came back

3 A near miss, or partly wrong, but familiar

4 Correct, but you hesitated

5 Completely correct response provided reasonably quickly

Scoring your answers accurately is very important. In interval or flashcard mode, all items that don't score 4 or 5 will be repeated during this session, over and over until this "passing" score is achieved. In addition, items that score 2 or less are considered "flubs." In all modes, they will be restarted from the beginning, cancelling out any progress you had previously made with that item (however, correct/incorrect statistics will be retained).

Score as correctly and honestly as possible. Fooling yourself by entering higher scores than are really justified doesn't help you learn.

Questions are presented in groups of (up to) 20 at a time. If you don't want to finish learning any particular group, just do an Emacs interrupt by pressing `C-g`, or pressing `q RET` when prompted to answer. Only questions completed to date will be scored, and that only if you *save* the database. (The database is saved automatically if you complete a group of questions, but not if you interrupt.)

7.4 Automated Learning in Interval Mode

If you just give the command `M-x elip-learn RET` or `C-cl` ELIP will take you through its idea of a cycle of "automated" learning.

At the moment, this concept is not terribly sophisticated. The logic is something like this:

1. You first learn "old" items; that is, items due (or overdue) for repetition (if any). You are continually prompted until *all* overdue items are complete.
2. If you have learned 20 or less "old" items, you will be offered up to 20 "new" or not yet learned items, if available.
3. In any case, as long as you didn't "quit" during 1 or 2 above, you will be offered a flashcard-style drill on the hardest items so far encountered in the database.

Of course, you can quit at any time you wish. Generally, learning sessions above 15-30 minutes will not be very productive. Cramming doesn't work so well in interval learning; the important thing seems to be do daily study.

This method for deciding when to add new items, based strictly on the size of the backlog, isn't very advanced. As you learn more and more items, and have repetition workloads that steadily increase, you may reach a point at which new items are not offered very often if at all. At that point you might just wish to manually add new items, as described in the next section.

Suggestions for better methodologies are earnestly sought.

It is not necessary to learn in automated mode. You can learn just new items, or just old items, as described in the next two sections.

7.5 Learning New Items in Interval Mode

If there are new items to learn (items that you haven't yet attempted), you may use the command `M-x elip-learn-new RET` or `C-cn`. If you have more than 20 new items in your database, a very common situation, you will be presented with the first 20. (You can change this number, as described in a later section, but it seems about right, at least for interval and Leitner modes.) Upon completion of these 20 items, if more new items are available, you will be asked if you wish to learn them. This continues until there are no more new items or until you decide to quit.

There is no need to cram in new items; 20 a day or even every couple of days may be enough. You decide; but ELIP *will* prompt you to learn more items if they are available. A word of advice: if you have something like 1,000 new items available (as in a new database), limit your ambitions for the first day's learning!

To find out how many new items are available, use the global report feature as described below.

It generally isn't the greatest plan to test on new items without having first spent some time looking them over. You can do this from database View mode by simply using the `'n'` and `'p'` commands to view the next or previous database entries respectively. Spend some time looking at the questions and answers to get an idea of the material before using the `elip-learn-new` command. Items are learned from the top of the database on downward, so the new items will generally begin just above the already-learned items. (There can be instances where this isn't the case, such as if you inserted new items into the middle of the database. Sorting the database is then recommended, using `elip-sort` as described above.)

Similarly there is nothing to prevent you from reviewing old items if you wish. ELIP is, after all, a self-study methodology and there is no concept of "cheating." Learn in the manner that suits you best.

7.6 Learning Old Items in Interval Mode

There may be "old" items to learn; these are really review items that you've previously learned but that the scheduling algorithm has determined are due for repetition. To learn these items use the command `M-x elip-learn-old RET` or `C-co`.

Again, if there are many, these will be presented in blocks of (usually) 20. But in the case of review items, you really need to do them all, or at least a lot of them, or you will build up an unbearable backlog. So, after you do the first 20 you'll be asked if you want to learn some more. You decide when to quit; there *is* a realistic limit as to how much you might want to learn at once.

To find out the number of pending items, use the global report feature described below.

7.7 Learning in Leitner Mode

In Leitner mode, since the interval algorithm isn't active, there is no need to separate "old" and "new" items. New items are brought into play as old items are completed, and all of this is automatic. To learn in Leitner mode just use the command `M-x elip-learn-leitner RET` or `C-ce`. Leitner learning is described in detail above.

Sorting or shuffling the database has no impact on Leitner mode, unless sequential learning is in effect.

7.8 Learning in Text Mode

In text mode, since the interval algorithm isn't active, there is no need to separate "old" and "new" items. New items are brought into play as old items are completed, and all of this is automatic. To learn in text mode just use the command `M-x elip-learn-text RET` or `C-ct`. Text learning is described in detail above.

Sorting or shuffling the database has no impact on text mode, unless sequential learning is in effect.

Again, it's worth repeating one more time that text-mode and other modes are not particularly compatible. Usually text-mode databases are specially prepared for use in that mode.

7.9 Learning in Flashcard Mode

It may happen that you want or need to do a review or give yourself a quiz, and the review items aren't due yet. Or you might want to try out some new items without scoring them or setting off the repetition cycle.

While learning new items in this way isn't really consistent with the ELIP interval learning philosophy, and reviewing old items on non-due dates will certainly influence the accuracy of the repetition algorithm, I've provided a flashcard option for use in case of need (or for that matter, desire). This allows you to completely ignore the repetition interval feature of ELIP and just use it as a pure flashcard program. (Not that I'm making any sort of value judgement here...) On the other hand, casual flashcard learning is not terribly inconsistent with Leitner learning, as it increases the degree of over-learning, which by some is considered a good thing (and by others, a bad thing). Flashcard learning isn't consistent with text-mode learning, but text-mode learning is a method unto itself.

To learn in this manner, use the command

`M-x elip-learn-flashcards RET` or `C-cf`

after which you'll be prompted to specify whether you want to drill on old (previously learned) items, new (not yet learned) items, the whole database, or the hardest items. ("Old" and "new" item designation applies only to interval mode; "hardest" items are defined as those items answered incorrectly the most times, in all modes.)

A group of up to 20 items, depending on availability, will be chosen completely at random from the population you specified. (But if you selected the "hardest" option you're limited to one-fifth of the database for databases smaller than 100 items. This is because otherwise

you're too likely to see many items that you've only missed once, which really undermines the concept of "hardest items.")

You'll then work through these questions just as you would if you were learning old or new items in the regular manner, but statistics will *not* be saved for this drill. You still are asked to score the questions, but the scores and other stats won't be saved. However, as always, you are asked to repeat missed questions until you get them right.

If you wish to learn more than 20 items, just repeat the `elip-learn-flashcards` command as many times as you wish. At present, there is no "coverage" option allowing you to drill on the entire chosen population. To do that sort of review, you are better off first *sorting* the database, and then simply flipping through the items with the usual EDB `'n'` (next) and `'p'` (previous) commands.

7.10 Resetting the Database

It may happen that, for a given database, you simply want to start over. To do this, give the command `M-x elip-reset RET` (no shortcut keys!). You will be able to reset a single individual mode, or the whole database. If you reset an individual mode, counts for that mode will be restarted, but item correct/incorrect statistics will be maintained. If you reset the whole database, everything gets reset. You will be asked if you really want to do this, as there is no turning back once you've reset the whole database. (Make a backup first.)

Individual mode resets are available for interval mode, Leitner mode, and text mode.

7.11 The Pileup Problem in Interval Mode

It can happen in large databases, when learning in interval mode, if you've learned a lot of new items in a short period of time, the workload can really pile up on subsequent days. And, as you continue to learn new items along with review of old items, the situation will only get worse. After a period of time, as the older items cycle into longer and longer repeat intervals, this may ease up a bit, but you'll still encounter groups of days with a high workload.

There are a few ways to deal with this. One way is to not keep adding new items until you're secure with old items, and they've started to cycle into longer repeat intervals. You can see exactly where you are with the `elip-workload` command. Another way is to simply not do all the repeats on the day they are due. Do a few groups of 20 and leave it at that. The rest will go into the backlog. As mentioned elsewhere, though, you have to deal with it all eventually. (I realize that my advice on this point is internally inconsistent. It *is* best to keep up if you possibly can.)

Another factor is this: your definition of "unbearable" and "pileup" is likely to change as you get experience with ELIP learning. A group of 20 questions can usually be tossed off in two minutes or less once you get into a rhythym. So even if you have 100 questions to deal with, you're looking at maybe 10-15 minutes of really concentrated effort. (And if the questions are so complex that they take more than 5-10 seconds to process, they are likely to be malformed.)

In the largest databases, which run over a thousand items, you *are* inevitably going to have a lot of work to do at times. This is not surprising; after all, you're learning a lot of material, and learning it well with excellent long-term recall.

So, the best advice is to simply learn to manage your workload. Adjust your study time as needed. You can still do a tremendous amount of learning in half an hour of honest-to-goodness effort.

There *is* another way, however; see the following section.

7.12 Delaying or Advancing Item Due Dates

Let me say straight away that I do not recommend frequent, if any, use of the features described in this section. However, there may be times when you wish to do so. Just keep in mind that you are essentially overriding and defeating the principles of spaced interval recall.

7.12.1 Delaying Items

Let's say you are going on an extended vacation, sans laptop or other means of accessing your ELIP learning databases. You come back after a month in Tahiti and fire up EMACS, bring up your first ELIP database— and you've got 187 items backlogged.

As mentioned above, I believe you *can* deal with this by taking a couple of days to get caught up. But if it's just all too much for you, you can make use of the `M-x elip-delay RET` or `C-cd` command. By the way, this is best done *before* your holidays, rather than after.

`elip-delay` or `C-cd` will ask you how many days you want to delay *all* old, or previously learned, items in the database. (Delay of individual items is not an option that makes sense here; if you wish to do that, just edit the item in EDB edit mode.) So, if you're going on a 30 day vacation, it might make sense to delay 30 days, or perhaps a bit more. ELIP will ask for confirmation; after all, this is a pretty drastic move on your part.

Upon return from your vacation, you will then find that your ELIP database effectively went

on vacation also, and is in approximately the state you left it, but with a 30 day (or whatever you entered) time shift. Unfortunately, though, your brain will have gone 30 days without learning from this database, and the recall intervals will be much less than optimal. But it's your choice as to whether you wish to use this feature.

7.12.2 Advancing Items

There is the even less desirable `M-x elip-advance RET` or `C-ca` option; in contrast to elip-delay, this option brings all items in the database forward (i.e. makes them due sooner rather than later). You might conceivably wish to do this to get some items out of the way prior to a short vacation; as the advance period is made longer, though, strange things can happen, such as a lot of items being advanced from a date *later* than your vacation to a date *within* your vacation.

This option is used in just the same manner as elip-delay; you enter the number of days to advance every database item, and confirmation is required.

This option is provided for the sake of completeness, and for those few instances in which you may wish to use it, but in general it is not at all recommended.

7.13 Changing the Number of Items Learned at Once

It may be that, as a designer of a learning database, you think that groups of 20 items at once are too large or even too small. Or, as a student, you may wish to modularize the learning sessions or make them "bite-size" or even "super size." This especially applies to text-mode learning.

To change this value, there is the undocumented command

`M-x elip-set-max-items-at-once RET` (no shortcut keys)

which is "undocumented" only in not being listed in the on-line help screen or the command summary in this manual.

Whatever new value you give is saved permanently with this database (but, of course, not with other databases). You can always change it back to 20 or to some other value as you desire.

This option is not particularly recommended outside of text mode, but is provided for convenience. Groups of 20 seem to be effective for well designed questions. If you find that only

10 or even 5 at once is more than enough in interval or Leitner modes, the questions in the database are very likely overly long or complex.

Note: you really shouldn't have to use this command very often. A database designer, particularly when designing a database to be learned in text mode, should pre-establish a good group size.

7.14 Checking a Database for Trojans

Since, as documented above, an ELIP database can contain executable Elisp code of arbitrary content, a nefarious and dastardly bungert could put in something foomatic or harmful. ELIP offers two safeguards against this.

The first time a database encounters executable Elisp code, you will be asked if you trust the database and wish to allow the code to run. If you answer "yes" then *all* Elisp code in that database will be considered "trusted" forevermore. This is true even if you upgrade the database to a newer version, as long as it has the same name and is in the same directory.

Conversely, if you answer "no" the same considerations apply except that this database and its successors and namesakes are forever considered "untrusted" and no Elisp code will ever be run from them.

But you *can* change your mind if you wish. Your choices are stored in the file `.eliprc` in your home directory. (Your home directory is likely to be something like `/home/loginname` on Linux or just `C:` on Windows. You may have to check around a little.) Edit this file and look for the name (full pathname) of the database in question. Just change "OK" to "NOT OK" or vice-versa to either allow or disallow Elisp code for that particular database. A new choice won't take effect, however, until you start a new Emacs session.

The second measure of protection lies in the command

`M-x elip-dump-elisp RET` (no shortcut keys)

which will search the database for Elisp code, and, if it finds any, will dump it to a special report buffer which you can examine to determine if the Elisp code is safe to run. Obviously, you'll need some knowledge, perhaps a lot of knowledge, of Elisp in order to make best use of this option.

Just like e-mail attachments and downloaded programs, the best protection lies in knowing the source of the database. Does it come from someone or someplace you can rely upon and trust? If you aren't sure, answer "no" to that first question and never run untrusted Elisp code. Please keep in mind that Elisp is a *very* powerful programming language and the amount of damage that can be done is almost unlimited.

8 Reports

8.1 Global Reports

At any time when you are visiting an ELIP database, you can ask for a summary, or global, report with the command `M-x elip-report RET` or `C-cr`. You will be taken to a report buffer which will list things such as:

- The number of items in the database.
- The number of new items left to study in interval mode.
- The number of items left to start in text mode.
- The number of items completed in text mode.
- The number of items left to start in Leitner mode.
- The number of items completed in Leitner mode.
- The number of items currently in each Leitner box ("box fill").
- The number of items overdue for repetition in interval mode.
- The number of items due to repeat today in interval mode.
- The number of items anticipated for review tomorrow in interval mode.
- The number of items studied so far in interval mode.
- The total number of attempts for all items.
- The total number of correct responses.
- The total number of incorrect responses.
- The overall percentage of correct answers.
- The average time per attempt.
- The average score per attempt.
- Your estimated letter grade for this material.

I expect some people will find the letter grade pretty annoying, particularly if you test on items "cold" and then get most of them wrong a number of times due to complete unfamiliarity. You'll likely wind up with a grade of 'F' and won't be able to raise it much higher in any reasonable amount of time. Cold testing is a poor idea; please consult the section above on how to study from ELIP databases.

Global reporting provides a very useful glimpse at how you are progressing through the database. One extremely practical use is to find out how many items are due today or tomorrow.

The global report is provided in the `*databasename.report*` buffer, where "databasename" is the name of your database, e.g. `history.dat`.

Note: reissuance of the `elip-report` or `C-cr` command will *overwrite* any previously generated report. If you want to save the report, write it out to disk before doing anything else.

8.2 Individual Item Statistics

Standing in opposition to global reports, the command `M-x elip-item-stats RET` or `C-ci` will show a potentially quite long report about each indiviual item. A new untried question has no stats as yet, so only the question and answer will be shown. For a question which has already been attempted, information is shown similar to that in the global report, but it's just for the item itself. The report is provided in the `*databasename.items*` buffer, where "database" name is the name of your database, e.g. `hawaiian.dat`.

Information provided includes:

- The item number. (An item can be directly accessed, from the database buffer, with the EDB `'j'` or *jump* command.)
- The date that this item is next due for repetition in interval mode.
- The number of times the item has been attempted.
- The number of correct responses.
- The number of incorrect responses.
- The percentage correct.
- The total time spent on this item, in seconds.
- The average time spent on this item per attempt, in seconds.

- The total score for this item.
- The average score per attempt for this item.
- The Leitner "box" for this item in Leitner mode (0 if never tried in Leitner mode).
- The current text-shown percentage for this item in text-mode (0 if never tried in text mode).
- The item question.
- The item answer.

Note: as for the global report, reissuance of the `elip-item-stats` or `C-ci` command will *overwrite* any previously generated report. If you want to save the report, write it out to disk before doing anything else.

This command also creates a spreadsheet-like buffer with all of the stats listed above, less the question and answer text. The buffer is titled `*databasename.table*`, where "databasename" is of course the name of your database, e.g., `spanish.dat`. You can output this buffer to your favorite spreadsheet program, or do sorting and analysis from within Emacs, with the commands `elip-table-sort`. (This command is currently somewhat cryptic. You need to supply a sort field number. The fields start at 1 and go to 12, from left to right, corresponding to the column headings at the beginning of the buffer.)

In the table, a Leitner box number of 99 means "complete." Similarly, a percentage of -1 for text mode also means "complete."

8.3 Interval Mode: Workload by Date Report and the Catchup Issue

The command `M-x elip-workload RET` or `C-cw` gives a very useful report, sorted by date, as to how many questions are due to be answered in interval mode on each and every date that currently has questions assigned to it. (The questions are not identified, just the total count of questions.)

You can use this report to look ahead, for instance, at the workload for the coming week—or if you are behind, you'll see how old some of the backlog may be!

This gives rise to the question of what to do if the workload for a given day or days is higher than you might like or even might have time to deal with. I had contemplated a sophisticated feature (such as in Supermemo) which, on command, would spread the load over a series of future days, to reduce a large workload for any given day. After some thought, I decided

not to do this. Since questions are answered in groups of 20, you need never do more than 20 questions on any one day. Any additional questions remain in the backlog and come up the next time you learn old items from the database. In this way you can distribute the workload as you wish; you just do 20 (or 40 or 60 or as many as you wish) items and do the rest on a following day. Of course, you *do* have to catch up at some point. You might have to pick a day when you have more time and work your way through the outstanding items.

I did, however, add a simple feature that allows a delay (or advance) of *all* items in the database by a *fixed* number of days per item. That way, if you're going on a 30 day vacation, you can push everything back 30 days or so. This is not particularly recommended, but it's an option.

9 Command Summary

You can always get a help screen repeating this brief command summary, plus the question scoring guidelines, with the `M-x elip-help RET` command. The current set of ELIP commands, with shortcuts, is:

```
elip-learn              C-cl  Do some interval learning
elip-learn-new          C-cn  Learn new items in interval mode
elip-learn-old          C-co  Learn 'old/due' items in interval mode
elip-learn-leitner      C-ce  Learn Leitner flashcard style
elip-learn-text         C-ct  Learn text memorization style
elip-learn-flashcards   C-cf  Do a flashcard-style drill
elip-report             C-cr  Report on summary learning stats
elip-workload           C-cw  Report on items due by date (workload)
elip-version            C-cz  Show ELIP version number
elip-import             C-cm  Import questions/answers from flat file
elip-export             C-cx  Export questions/answers to flat file
elip-reverse            C-cv  Reverse questions and answers to/from flat files
elip-item-stats         C-ci  Show statistics with each item
elip-sort               C-cs  Sort an ELIP database
elip-table-sort         C-cT  Sort an ELIP item table
elip-shuffle            C-cS  Randomly shuffle ELIP items
elip-reset                    Reset the database and restart learning
elip-delay              C-cd  Delay due dates of all items
elip-advance            C-ca  Advance due dates of all items
elip-scan-text          C-cc  Prepare questions from marked text file
elip-mark-area          C-c1  Mark area of interest for ellipsis questions
elip-mark-question      C-c2  Mark text for an ellipsis question
elip-done-marking       C-c3  Finish with an area for ellipsis questions
```

```
elip-set-bookmark       C-cb  Set bookmark in text file
elip-goto-bookmark      C-cj  Go to bookmark in text file
elip-backup             C-cu  Not implemented
elip-restore            C-cy  Not implemented
elip-make-database      C-ck  Create a new ELIP database
elip-dump-elisp         C-cp  Dump ELISP code in an ELIP database
elip-quit               C-cq  Quit from ELIP database
```

Note that upper and lower case *does* matter!

Most of these commands must be given when you are viewing an ELIP database buffer, or in a buffer from which ELIP can deduce the correct ELIP database buffer to process from. The exceptions are `elip-table-sort`, which must be given from a *table* display, and `elip-scan-text`, which must be given from the text buffer you wish to scan. In practice, all of this should work out to be pretty intuitive (famous last words).

10 .emacs Considerations

At present, there is no *requirement* to do anything in your `.emacs` file other than what you already did to set up EDB. It may be that you wish to load an ELIP database on startup; the command

```
(db-find-file "name-of-database")
```

is the way to do this. (This must, of course, appear *after* the code that sets up calls to EDB.)

In fact it may not be a bad idea, if you plan to do question and answer editing in a text buffer, or to use the ellipsis method of making questions, to load the shell database `elip.dat` right away. This is because no ELIP commands are available until a database is loaded. So, somewhere *after* your EDB setup, you might put

```
(db-find-file "/correctpathto/elip.dat")
```

substituting the appropriate path name for "correctpathto" which points to the directory containing your ELIP databases.

Note: `.emacs` might be known as `.emacs-local` or `.gnu-emacs-custom` or maybe even a different name, depending on your local version and installation of EMACS.

11 The Interval Algorithm

The *interval algorithm* controls how soon a given item will be repeated in interval mode, and attempts to optimize the learning process. For instance, in the supplied ELIP implementation, a new question that is missed will be repeated the next day; if it's not missed, the first repetition will be about four days later. After four days, if the question is correctly answered, the next repetition will be seven days after that, and so on. If, however, a question is "flubbed" with a score of two or less at any point, the cycle is restarted from the beginning: the question will be presented the next day, and then in four days, and so on.

A table (actually a vector) of repetition invervals, in days, appears in the `elip.el` code as the variable `elip-cycle-days`. This table presents *maximum* repetition intervals, as follows.

If a repetition would be scheduled for 12 days, that 12 day number is multiplied by the percent correct for this item. So, if the item thus far has been answered correctly on 75% of the attempts, the next repetition would actually take place in 9 days (75% of 12).

If you wish to modify the algorithm, it is contained in the function `elip-find-nextask`. You can study this function to find out about the calculations and the values that need to be set. I would be most interested in knowing of any improvements you might make.

If additional information about individual items needs to be kept, record variables `elip-aux5` through `elip-aux8` still are available in the *current* ELIP implementation. (Variables `elip-aux1` through `elip-aux4` and `elip-aux9` have already been gobbled up.)

The algorithm started out based on both the Supermemo pencil-and-paper version, and the published Supermemo algorithm SM-2 (see the Supermemo website for a description). SM-2, as does ELIP, calls for a difficult item to be repeated more often. SM-2 bases difficulty on response scores. ELIP bases difficulty on percentage of correct responses. I made this change because I don't believe the scores assigned by the learner are a particularly accurate measure of difficulty. I do believe that correct and incorrect response counts are a very accurate measure, and so I use this instead.

Making changes to the algorithm is easily done and may prove very interesting to those so inclined. Please share your ideas and improvements.

12 Release Terms and Disclaimer of Warranty

ELIP is released under the terms of the GNU General Public License. All terms of that license pertain to ELIP.

ELIP comes *without warrranty of any kind* including warranty of fitness for intended purpose. NO SUPPORT IS OFFERED OR PROVIDED. The author disclaims all responsibility for damages of any type, whether direct, indirect, special, consequential, or otherwise *even if the author has been made aware of the possibility of such damages.* You are responsible for evaluating the utility, suitability, and safety of ELIP in your own environment.

In other words, if something bad happens, that's your problem. But if something good happens, credit would be appreciated.

13 Contact and Mailing List

While no support is provided and certainly no promises are made, please let me know about bugs, difficulties, or inadequacies in ELIP. While I don't make any commitment to do anything about it, I likely will want to. Similarly, ideas about changes and enhancements are also appreciated.

If you develop interesting learning material, you might want to share it. I will consider hosting such material on my server, provided you offer it under terms equivalent to the GNU General Public License (it has to be free, for one thing).

I can be reached at `chungkuo@chungkuo.org` and I expect this address to be stable for some time to come.

I have set up a mailing list for anyone interested in any aspect of ELIP. At present, the "user population" is rather limited, so the activity on the mailing list mainly consists of my announcing the availability of updates or new databases. The list is not archived.

To subscribe to the list, send an email to

```
majordomo@chungkuo.org
```

with a blank subject line and just a single line in the body of the message:

```
subscribe elip
```

14 Acknowledgements

Many thanks to Chris Lowis for feature suggestions, and for patiently awaiting their implementation. Chris also supplied a Japanese language learning database. Thanks also to

anyone who sent in a bug report, or even just a note to say that they've discovered the existence of ELIP.