LanguaMunity

The Official Manual

By Sophia Elizabeth Shapira

(This is an alpha-beta version of the manual.)

For the latest version of this manual, visit:
http://languamunity.org

Contents

Basic Disclaimer

I hope to eventually translate this to Legaleze if someone helps me -- but until then, I must say the following as clearly and definitively as I, a layperson with regards to law, possibly can.

What I must say is that I need to clarify up-front that THIS MANUAL, AS WELL AS ANY SOFTWARE IT REFERENCES, AND ANY COMMANDS IT INSTRUCTS are provided AS-IS and WITHOUT ANY WARRANTY OF ANY KIND - not even the IMPLIED WARRANTY of MERCHANTABILITY or FITNESS FOR ANY PARTICULAR PURPOSE -- nor even any warranty of being SAFE TO USE.

Make no mistake - I make every effort to assure that all this is quality, safe, and useful --- BUT I MUST MAKE THIS DISCLAIMER because I CAN NOT AFFORD TO BE IN A POSITION OF HAVING TO PROVE THAT IN COURT. I just don’t have the resources to fight that battle SO FOR THAT REASON, I MUST MAKE IT CLEAR THAT I AM PROVIDING ALL THIS WITHOUT ANY WARRANTY AT ALL.

Introduction

What Is LanguaMunity?

LanguaMunity is a software package designed to introduce an improvement to how foreign languages are learned. At the heart of the software-end is probably the most refined algorithm available for a flashcard system. However, this flashcard-system is not meant to be used by itself - but is meant to be combined with other aspects of learning - most importantly, with immersion.

I would define **practical use** of a language as any use of a language that you are learning for a purpose other than for learning the language itself. Subsequently, I would define **immersion** as a substantial amount of practical use of a language that you are learning.

Quality immersion of a language must contain at least *some conversational* immersion - but even a modest amount can be supplemented by existing things such as reading. A well-composed LanguaMunity course should guide you through that as well.

About This Manual

This manual is the official user's manual for LanguaMunity. Eventually, a separate developer's manual will be written for the benefit of those preparing courses for LanguaMunity - as well as for those preparing auxiliary tools for LanguaMunity.

Installing LanguaMunity

Instructions on how to get LanguaMunity installed and up-and-running may vary from one system to another. For this reason, instructions here will have to be provided on a system-by-system basis and added as the details for that system are worked out.

Installing on MacOS X

MacOS X is the operating system on which LanguaMunity was first developed - and thereby the first system that these instructions are to be provided for.

Now - though LanguaMunity has the aim of eventually having GUI implementations and even proper mobile implementations - this documentation is for the Prototypical imlementation of LanguaMunity -- which is run from the Command Line. Some MacOS X power users are familiar already with the command line on Mac OS X. But in cas you are not, there are a few sources you could go to for a primer:

- http://blog.teamtreehouse.com/introduction-to-the-mac-os-x-command-line
- http://www.macworld.co.uk/feature/mac-software/get-more-out-of-os-x-terminal-3608274/

Basic Installation of the Software

To install LanguaMunity on MacOS X you first have to install Git (a software package that is used in the installation process, the update process, and in the actual function of LanguaMunity) and then you have to run a special terminal installation command that installs both **chobakwrap** (a framework package that LanguaMunity uses) and LanguaMunity itself.

Installing Git

Git is needed for the installation, for the update, and for the proper function of LanguaMunity.

To install Git on MacOS X, visit the URL https://git-scm.com/download/mac to download the MacOS DMG package. Then, install the package you are given, and after that proceed to the next step.

Running the LanguaMunity Installation Command

The next step in installing LanguaMunity on MacOS X is to simply copy and paste the following block of code to the command-line (accessible through the Terminal) and then press the **[return/enter]** button.

```
curl -f \
http://languamunity.org/scrips/smp/a.php/osx/install-01.sh.txt \
-o ~/to-install-languamunity.sh && sh ~/to-install-languamunity.sh
```

Don't try to copy it by hand unless you are in a position in which you have absolutely no choice -- and in such situation be double, triple, even quadruple careful to avoid even the slightest typographical error. Best, if at all possible, to simply do the copy/paste.

As you notice, this block of code spans three lines - although it is only one command. This is because the end of each of the first-two lines is a backward slash ('\') which signifies to the standard Unix-type shell that the command is to continue on the next line. Anyone who is seasoned in working with Unix-type shells (such as, for example, that one accessed through the MacOS X Terminal) will know this - but for those of you who are new to it it is important to learn this right now.

Anyway - once you've done this installation, you will be able to in the future update the software just by typing the command:

```
update-languamunity-software
```

The only way that you will again have to install as though for the first time is if there is an important improvement in the infrastructure for invoking the update/install commands.

Getting Speech Synthesis to Work

As a separate PERL script must be written into the LanguaMunity code for every language for which speech synthesis is a supported feature (although the option of third-party utilities is on the development to-do list). For this reason, not every language has speech synthesis available.

However - if the language you are learning is one of those for which speech synthesis is supported (and if the flashcards you are using are configured to use that feature) then after typing correct-answers to the flash-cards in your target language, you will hear the computerized approximation of a native speaker demonstrating how the answers you type are pronounced.

However - for this to work, you have to download the voices that LanguaMunity requires for that particular language (or else you won't hear anything at all). Here are a few source from which you can learn how to do this:

- Instructions on Apple's web-site: (https://support.apple.com/kb/PH18734?locale=en_US)
- An article in OS X Daily: (http://osxdaily.com/2011/07/25/how-to-add-new-voices-to-mac-os-x/)
- An MacOS Hints page from MacWorld: (http://hints.macworld.com/article.php?story=20110704093645914)

Regarding the supported languages, here are what they are and what voices you need:

Spanish	The Mexican voices **Juan** and **Angelica**

Before you Start Studying

Okay - you've installed LanguaMunity and are ready to check out your first course and start studying.

But wait! Before you begin, there are a few things that you need to check.

Checking the Browser Open Command

Some lessons in LanguaMunity are accompanied with "lectures". These "lectures" are generally not oral, but written - and are in the form of HTML documents. When you add to your study-belt a lesson that is accompanied by such a lecture, LanguaMunity will use one of the **chobakwrap** subcommands to display the lecture in a browser window.

However - the particular subcommand of **chobakwrap** that is used to do this has to be implemented differently for different operating systems. Hopefully, this command, when invoked, will simply do the desired thing, which is displaying the lecture in a browser window. However, depending on what system yours is and how it is set up - it may just provide a notice that it is *attempting* to open a particular HTML file, followed by either the URL or the local-pathname of the file -- in which case, you will have to open that file yourself in a browser window.

In any case - it is important that you go forth knowing what happens on your particular computer when this **chobakwrap** subcommand is invoked. You will find this out by attempting to open a well-known web-page with this subcommand and seeing what happens. Simply type the following command and make note of what happens:

```
chobakwrap -sub browseropen http://google.com
```

And just to be certain - let's try it on another URL -- this time one of the best web-comics of all time (if not the best):

```
chobakwrap -sub browseropen http://egscomics.com
```

Courses and Houses

When you select a LanguaMunity course to learn - LanguaMunity creates a local copy of the course on your machine, as well as a repository of information regarding where you are in the study of that course. At lack of any better word, the term used for all this stuff that LanguaMunity creates each time you select a course is a “house”.

When you make this selection, you specify the URL of the online Git repository of the course as well as the name that you want to give to the house that is created. The URL of the online Git repository must be whatever is provided by the information about the course. However, the local house-name can be anything you choose provided that (a) it contains only alphanumeric characters (b) any and all alphabetic characters in the house’s name are lowercase and (c) it does not conflict with any of the names of houses you already have.

Let’s say you want to learn Spanish with the second experimental Spanish-language module for English-speakers. And let’s suppose you decided to call the local house “spanishmain”. You would create the house with the command:

```
languamunity git spanishmain \
https://github.com/LanguagMunity/course--en--es--experiment002.git
```

Running this command does two things: (1) it creates a local house named “spanishmain” through which you will study the selected course and (2) it selects the house named “spanishmain” as your **Current Default**. The Current Default is the term for the house currently selected by LanguaMunity that will serve as the context for various LanguaMunity commands.

To see a list of all houses on your account on the system, type the command:

```
languamunity houses
```

To change your Current Default to any of the houses shown on this list, type the command:

```
languamunity select name-of-house
```

For example, if you wish to return to the “spanishmain” house we just created, the command would be:

```
languamunity select spanishmain
```

Last but not least, if you need to completely clean-slate a house (for example, if something in the house’s structure is hopelessly awry) and make the house exactly as it would be were it just created right now (which means you will lose any progress you have made in it’s study), type the following command:

```
languamunity reinit name-of-house
```

For example, if you want to do this to the “spanishmain” house that we just created, the command would be:

```
languamunity reinit spanishmain
```

This, too, I believe, will also set the newly reinitialized house as the Current Default.

Beginning Your Study

So now you've installed LanguaMunity, checked how it looks when **chobakwrap** attempts to open a document in a browser window, and checked out your first course into a local house. It is time to start actually studying.

Adding your First Lesson to the Study Belt

The first thing you have to do is add a lesson to your Study Belt - that is, the sequence of lessons that LanguaMunity assemble each round of flashcards. Before you do this, you need to know what lessons are available to add that are not already added. You find this with the following command:

```
languamunity eligible
```

Once you have selected the course you want to add to the belt from the provided list, you do so using the following command:

```
languamunity addlcn lesson-id-code
```

In the example of the Spanish course used earlier in examples, the only lesson eligible to add at this point is the one labeled "newbie-001". So in this case, the command would look as follows:

```
languamunity addlcn newbie-001
```

The Lesson's Lecture

In some cases, a lecture page will appear in a browser window when you add a lesson. (We mentioned that earlier.) As stated, it probably isn't literally an oral lecture, but is rather a written document in HTML. If there is such a lecture, you should read it now before proceeding any further. However, if not all of it sticks in your memory, you can always re-read it later by typing the command:

```
languamunity lcnlect -idcd lesson-id-code
```

In the case of this example, that would be:

```
languamunity lcnlect -idcd newbie-001
```

Of course, there are multiple reasons why one might want to later on review the lecture.

- You want to refresh your memory of something - or look back on something you are uncertain about.
- There may be links to various online resources you want to check out.
- After an update of the course material, you may want to see if there is anything new in the lecture.
- And sometimes, even, after an update of the course material, a lesson that previously didn't have a lecture might suddenly have one.

Currently, there is no way to get a comprehensive lists of lessons in the course that have lectures - but that is on the development to-do list.

Invoking the Quiz Session

Once you have added the lesson, and read the lecture material if there is any (which in the above-example of the “newbie-001” lesson there should be) you are ready to begin your first five-minute round of flashcards. You do that with the following command:

```
languamunity quiz
```

When you complete the five-minute session, you will be shown a message something that looks somewhat like this:

```
        Start time: Wed Nov 16 14:57:26 EST 2016
          End time: Wed Nov 16 15:03:04 EST 2016
Round Within Level: 2
   Status In Round: 0:3 - 59
   START QUIZ-SIZE: 1044
     END QUIZ-SIZE: 1006
     Wrong Answers: 2
Rehashes Requested: 0
 NET CARDS CLEARED: 38
```

The line labeled “Rounds Within Level” says what rounds of flashcards you are on. No matter how easy a time you are having answering all of the cards you are shown, it is not advised to add yet another lesson to your Lesson Belt until this number reaches a value of 4 or 5.

Commands Within the Quiz

While you are in the midst of a quiz, there are three general kinds of prompts you can be presented with. One is the Question Prompt that occurs within the context of a flashcard. The other is the Transitional Prompt that you are presented with after completing each flash-card before proceeding to the next one.

The Transitional Prompt prompt looks as follows:

```
["h"/"x"/..] JUST [enter] TO CONTINUE:>
```

The third kind of prompt is a Special Prompt - which only will be presented to you in response to you issuing certain commands.

Now, in front of any kind of prompt (except for the Special Prompts, which are reserved specifically for you to provide information needed by the already-issued commands) there are different commands that you can issue.

Commands for the Question Prompts

All commands that are eligible for the Question Prompts begin with a double-asterik (‘**’) so as to distinguish them from actual attempts to answer the question.

**diff

This command is presently implemented for the context of simple-text flash-cards, but not for the context of drill-sequence flash-cards (such as those used in conjugational exercises).

Let us suppose you answered a question and got a message that your answer is wrong - but you can not see where the answer you provided is different from the answer you were supposed to have given. This command prompts for expanded output that clarifies exactly where your answer and the correct one differ.

**out

This command is written specifically for Question Prompts - but (unlike other Question Prompt commands) it is also valid in the context of Transitional Prompts (complete with the double-asterik prefix).

Let’s say you started a quiz-session but realize, upon doing so, that your head is in La-La Land and you are repeatedly missing questions that you know the answer very well for. This command can be used to invoke an emergency exit to the quiz in which none of the changes in status (neither the little progress nor the extensive misses) are saved.

By the way - don’t use this command just for missing one question that you should have known the answer to. Brain-farts can happen any time. Rather, use it just when you are *repeatedly* missing questions that you know the answer to full well -- in short, when it’s clear that your head just isn’t in the game.

Also - don’t use this on account of missing questions that you just learned or otherwise are shaky on. In those cases, records of a wrong-answer may be exactly what you need.

Commands for the Transitional Prompts

rvu

This command adds rehash-pairs for the flash-card you just completed. It invokes a Special Prompt so that you can use this command to order anywhere from one to ten rehash-pairs.

For each rehash-pair you order, one copy of the just-finished flashcard is added onto the short-term rehash pile - and another copy is added onto the long-term rehash pile.

It should be noted that whenever you provide a wrong-answer to a question, a rehash-pair is automatically added. (In the case of drill-sequence cards, this calculation is done based on whichever item on the list you guess wrong the most times.) So if you had gotten wrong-answers on the previous flash-card prior to (finally) getting the right answer, any rehash-pairs you request through this command will be in *addition* to those incurred by wrong answers.

x

This command indicates that your quiz must end now because you don't have time to spend the full five-minutes on it. It is different from the '****out**' command which doesn't save your status. Rather, this command saves your status just as the program would if it were ending the quiz because your five minutes are up.

As a matter of fact, when the quiz is ended by this '**x**' command, it does everything exactly as it would if the five minutes were up - only it does so early.

House Maintenance

Now, having discussed the commands that you will find useful during a quiz session - we go back to LanguaMunity commands that you would use at other times from the system's terminal shell. Specifically, it is time to discuss commands you need for the maintenance of the house.

Keeping the House Up-to-Date

There is a good chance that the course you are taken is a work-in-progress - or at least undergoes minor improvements and/or updates once in awhile. For this reason, it is important that, from time to time, you update the contents of the house.

To update contents of the Current Default house, type the following command:

```
languamunity update
```

This updates the in-house copy of the course you are taking. However, it should be noted that (in order to prevent potentially problematic inconsistencies between the flash-cards of different versions of the course) this command also clears out and empties the rehash piles - both the long-term rehash pile and the short-term rehash piles.

For this reason, as the rehash piles are important to the learning process, it is recommended that you not be too frequent in updating the house's contents. Generally speaking, unless you are part of a course's development team, updating once a week is frequent enough.

However - it is also important to make sure that you never wait longer than five weeks between house updates. This is because sometimes updates make changes in the structure of the course-mapping, and it is important that the previous version of the course be recent enough that the instruction-codes for adjusting the structure go back to that previous version. The longer the developer has to maintain such a backward-compatibility, the more burdensome it can become. That is why it is asked that users not wait more than five weeks between updates to a house's contents - or else you could potentially find yourself in a position of having to reinitialize the house.

While it is true that developers are expected to wait longer than five weeks before allowing such backward compatibilities to be broken, any time beyond five weeks is to be seen as grace period for error margins and the like, and not time that the user should take for granted.
--