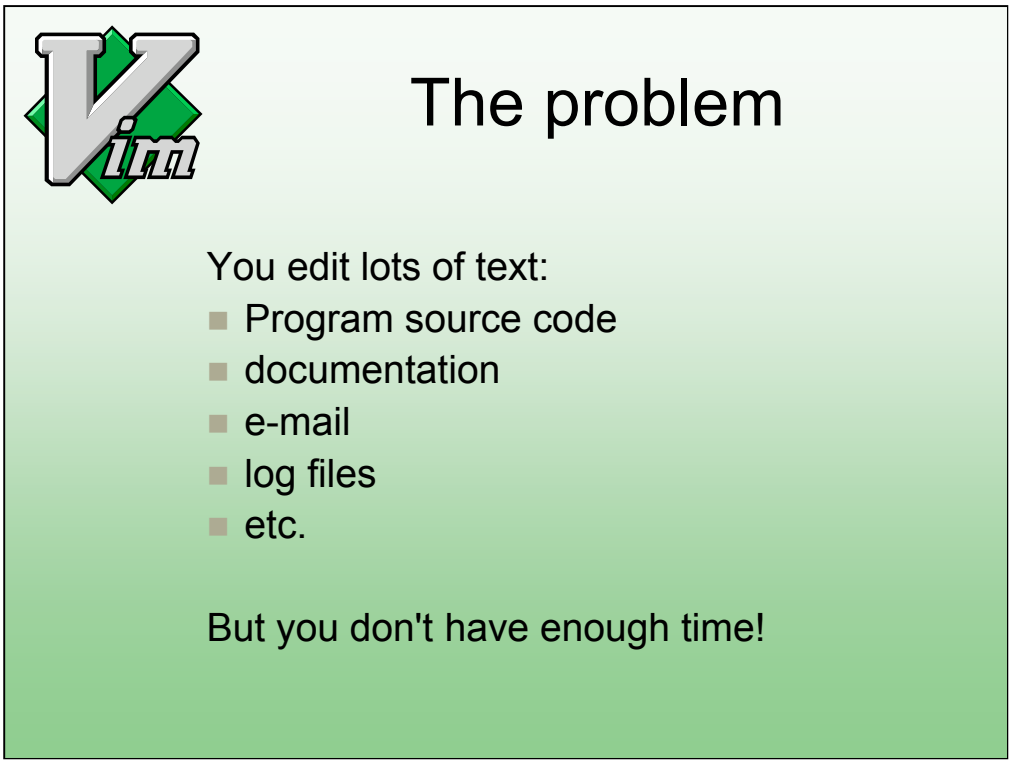
Seven habits of effective text editing 2.0

Bram Moolenaar

www.moolenaar.net

Presentation given by Bram Moolenaar at Google, 2007 February 13

There will be time to ask questions at the end of the presentation.

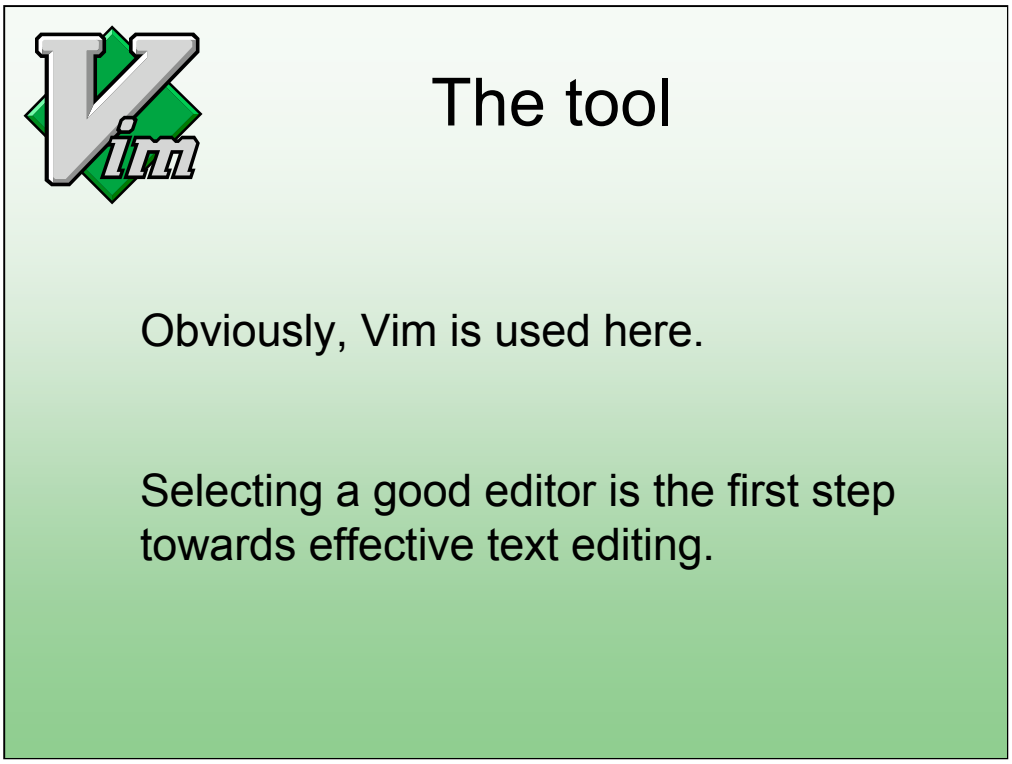


Editing text takes a lot of our time. Looking at myself, I spend more than half my day reading and changing various kinds of text.

Rest of the time is spent in meetings. Perhaps someone can do a “seven habits of effective meetings”?

This presentation is about how to get more work done in less time.

Don’t listen when you are paid by the hour. :-)



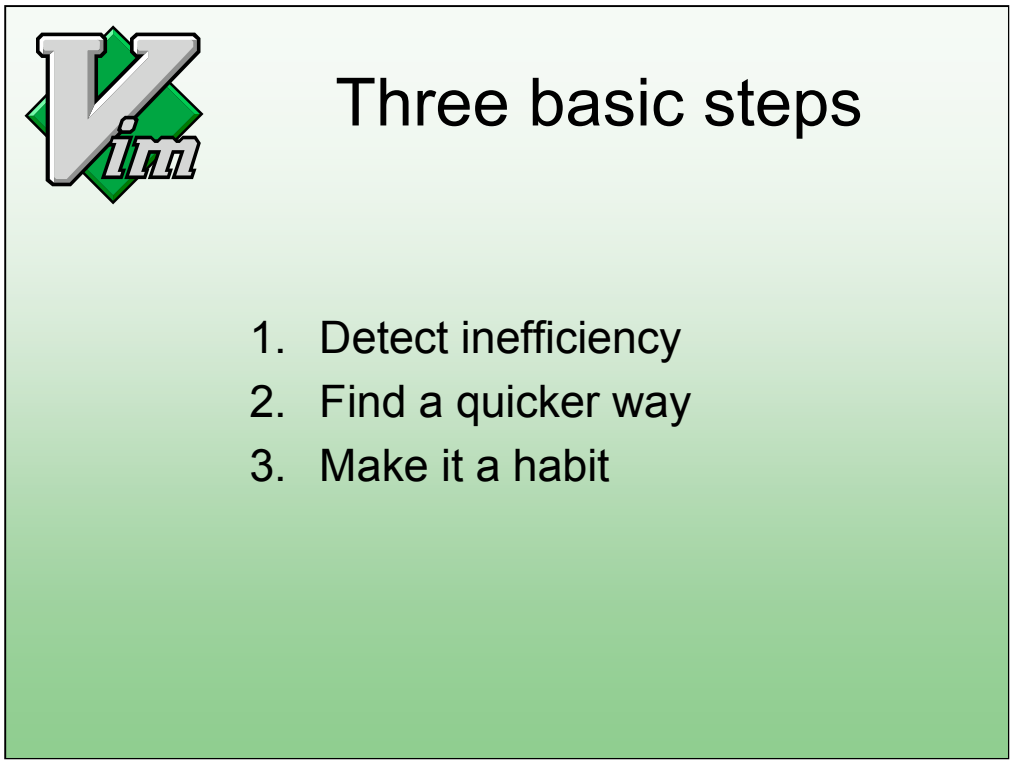
Some people use Notepad and never get past it. Those people can learn a lot today.

You can spend lots of time on evaluating different editors. I use two rules:

- If you are already using an editor and it works very well for you, don't waste time by learning to use another one. However, after this presentation you might wonder if your editor is really good enough.
- Otherwise use Vim. You won't be disappointed.

I'm not going into the discussion which editor is best for you. That would take too much time.

Question: Who of you have never used Vim? Who is using Vim every day?



Keep an eye out for actions that you repeat and/or spend a lot of time on. Lean back and evaluate what you have been doing the past hour. If you have more than a bit of work to do, try to find a pattern in it that repeats itself.

Example: You have a program with several files and have to rename a function that's used in many places. If you have to type the name many times you are inefficient.

Any powerful editor provides you with commands to do your work fast. You just have to find them. If the editor doesn't offer an appropriate command, you can use its script language. Or perhaps you can use an external program.

You have to learn new commands until you use them automatically.

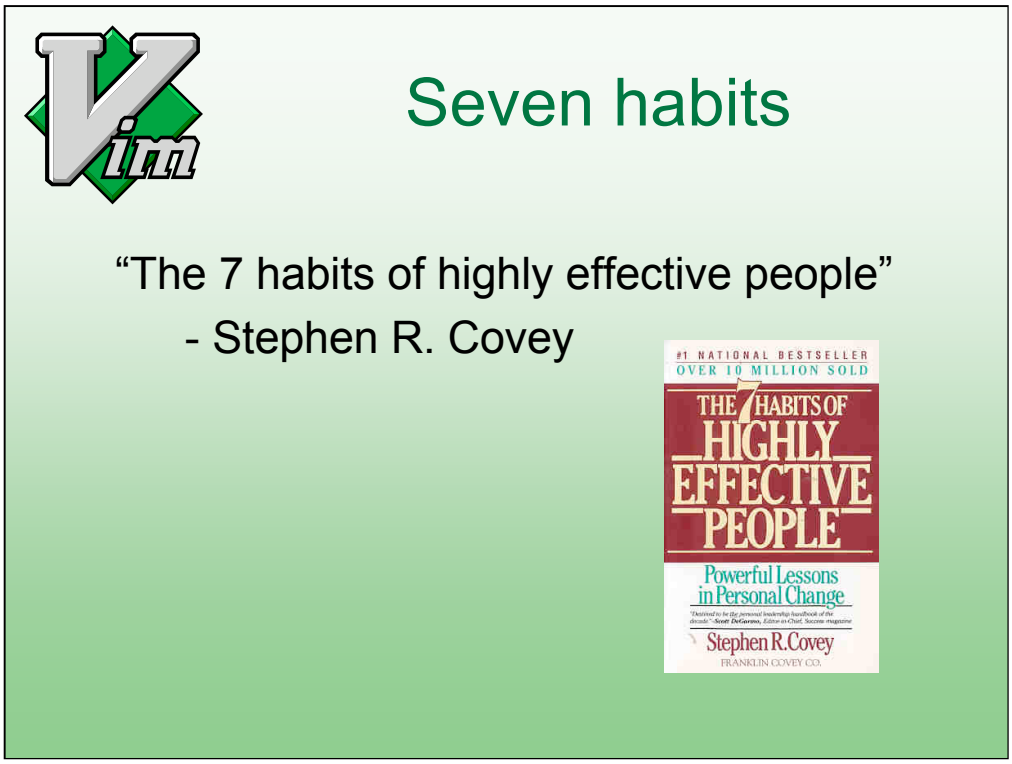
Summary:

1. See the problem

2. find a solution

3. use it

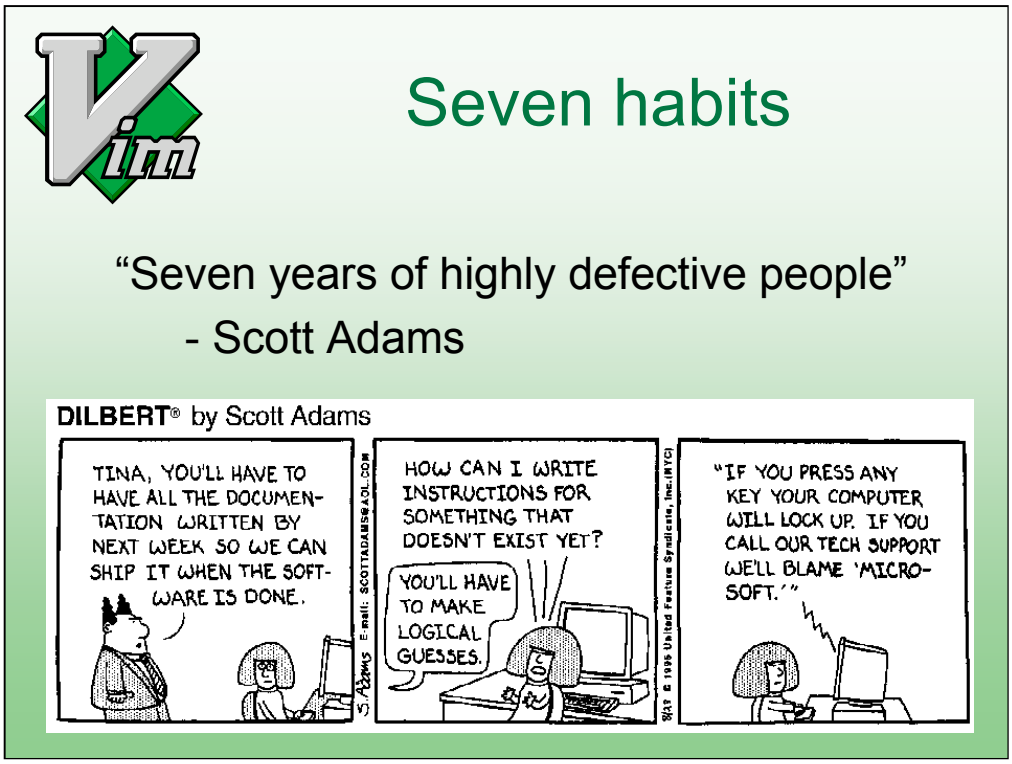
That sounds easy, doesn't it? And so it is, you just have to do it.



I will give seven examples. Why seven? To match the title. The title is inspired by the book from Stephen Covey.

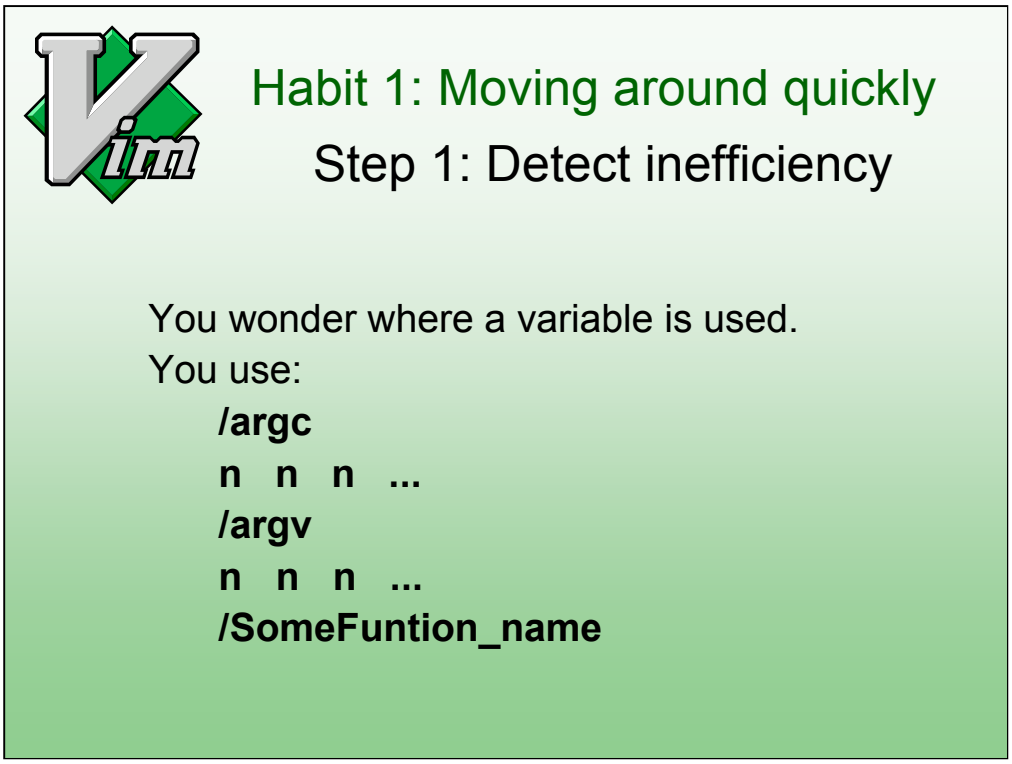
The Seven habits book is a very good one. I can recommend it to anyone who wants to improve his life. And who doesn't?

Although, the presentation might as well be based on Dilbert….

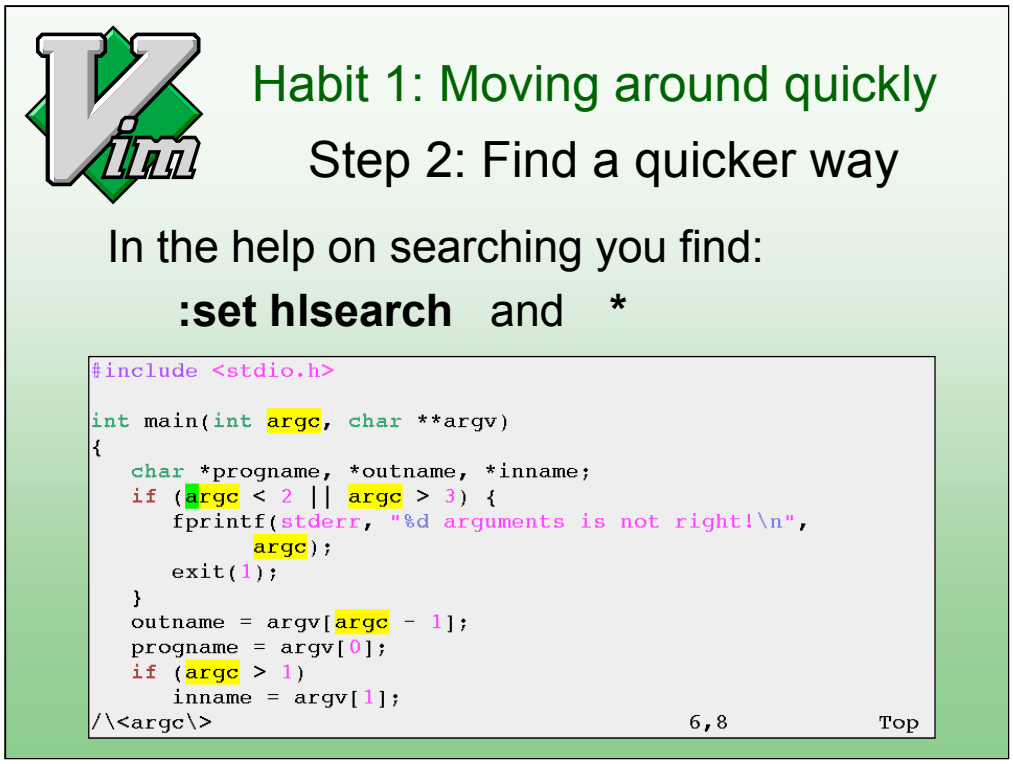


The Dilbert book helps you to enjoy life. Read it when you take a break.

These two books make a great combination. See **http://www.iccf.nl/click2.html** for more information (and help orphans in Uganda at the same time).



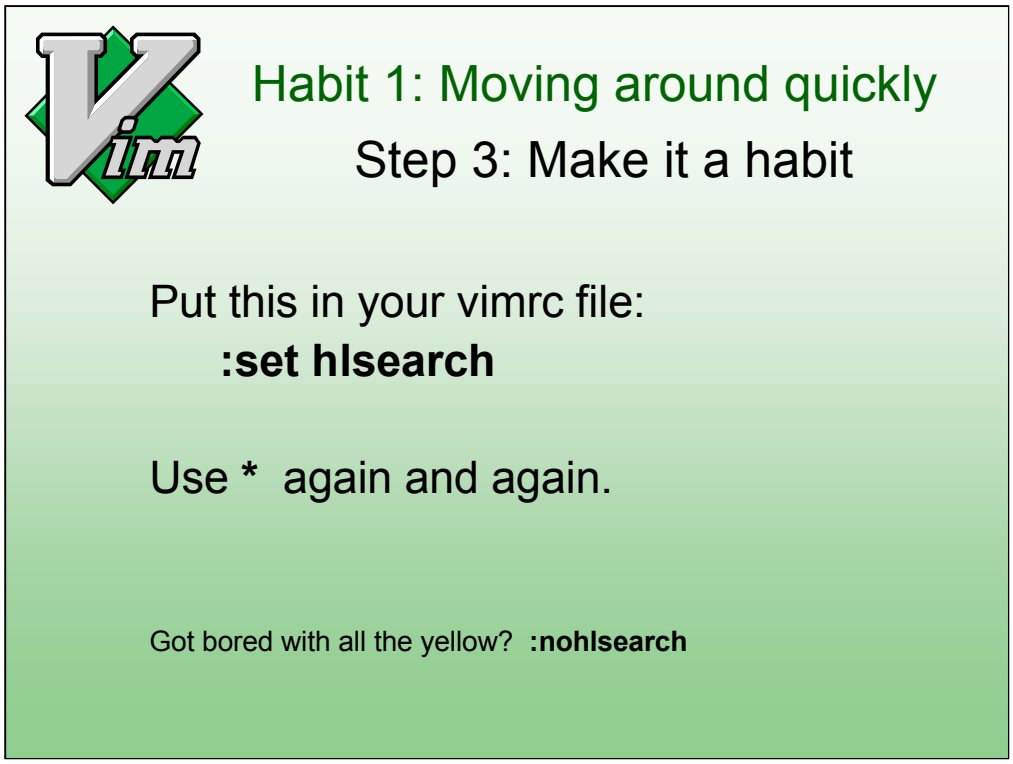
While editing a program you often have to check where a variable is set and where it's used. Currently you use the search command to find each location. And if you type it wrong you can't find it.



You look in the help for searching commands. You find references to two items that appear to be useful:

The hlsearch option shows all matches with a pattern. You don't have to find each match, you can see them right away.

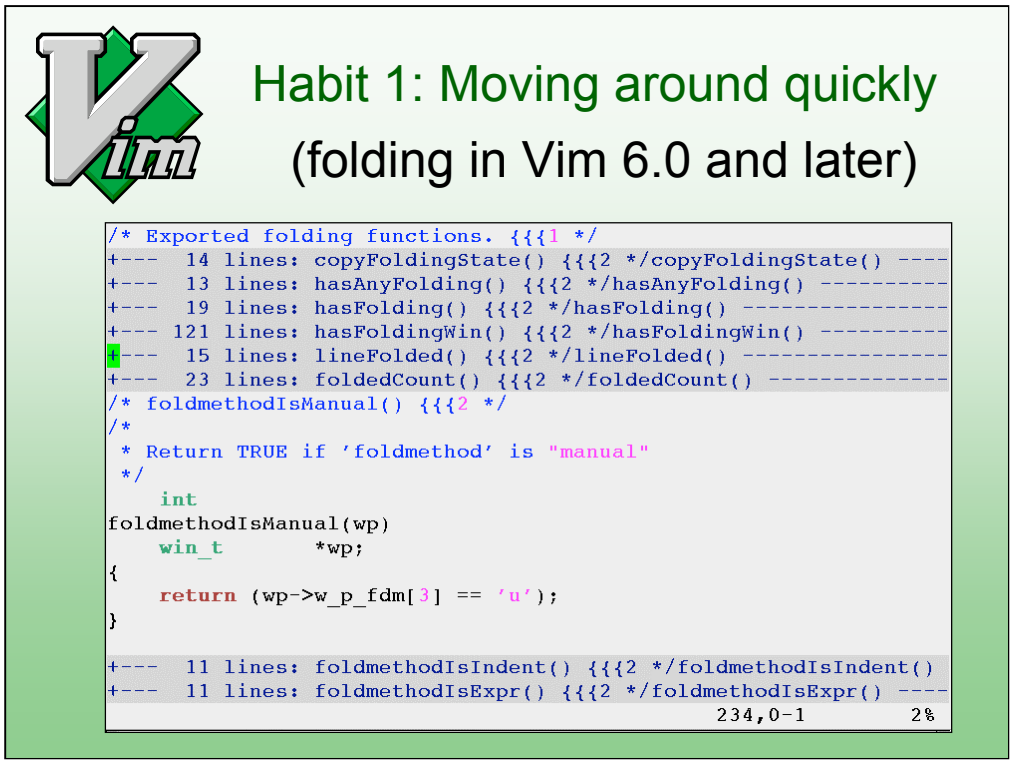
The * command searches for the word under the cursor. You don't have to type the word.



You now start using the "*" command.

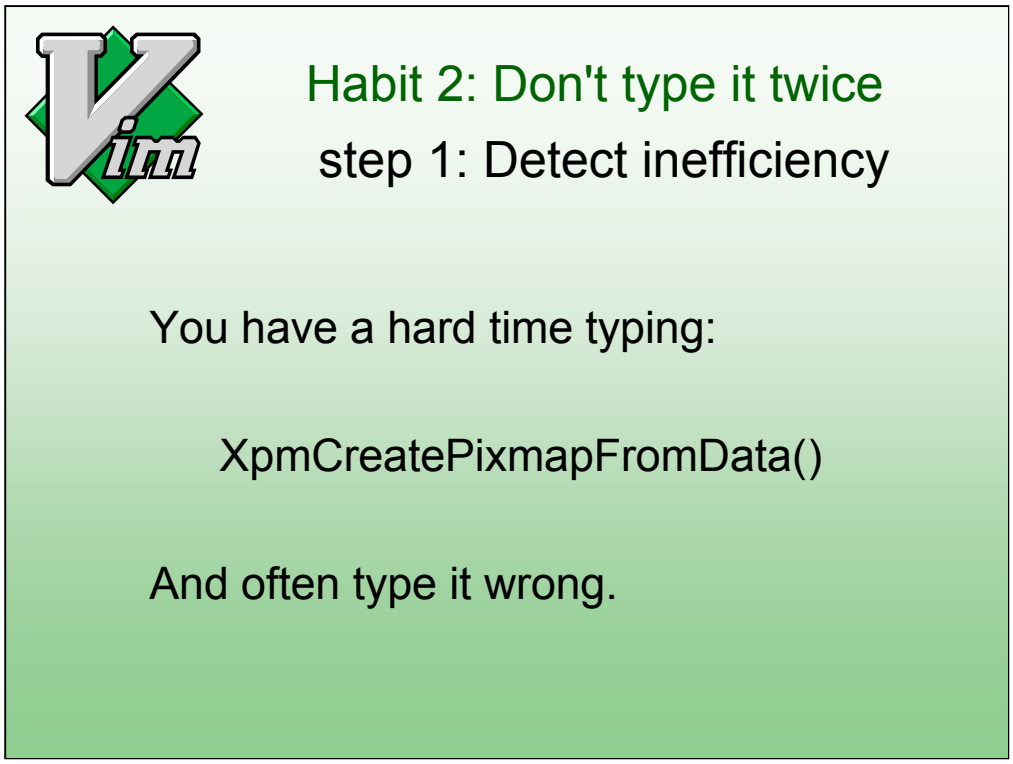
Every editor offers many ways to move around. It's not a bad idea to read the documentation to find useful commands.

Note: ":nohlseach" can be abbreviated to ":noh". Don't forget to use command line completion (using <Tab>).



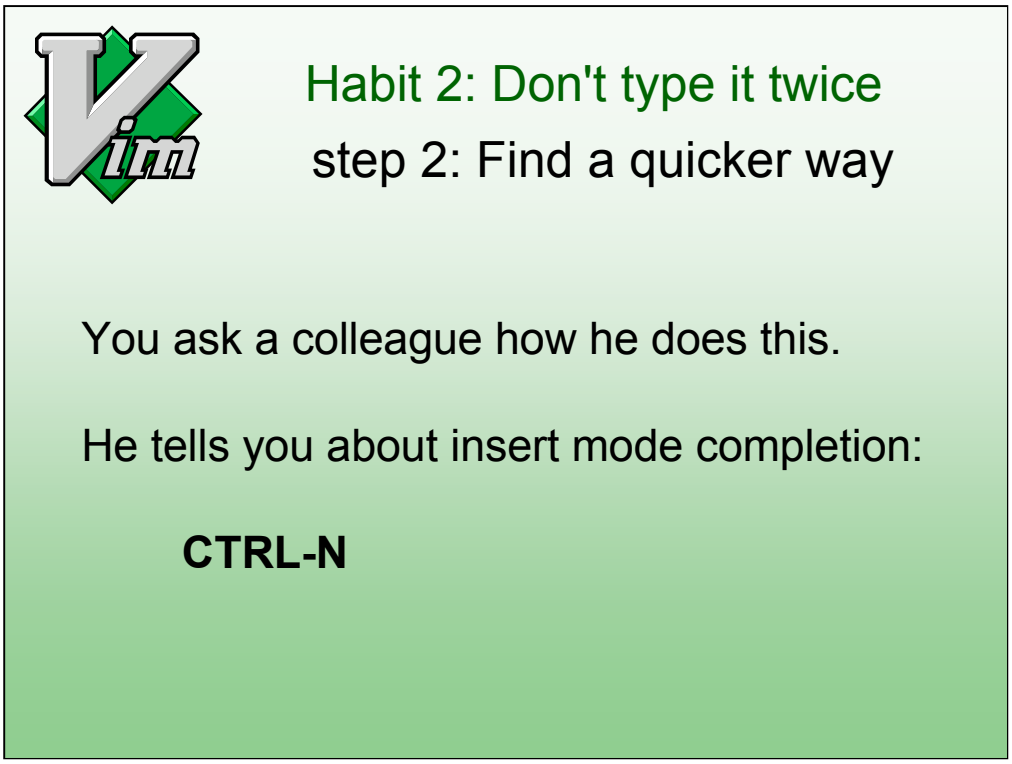
The folds contain a block of lines. In this example each fold contains a function.

Closing all the folds makes it easy to locate a function, quickly move to it and open the fold. That's a clever way to move around quickly.



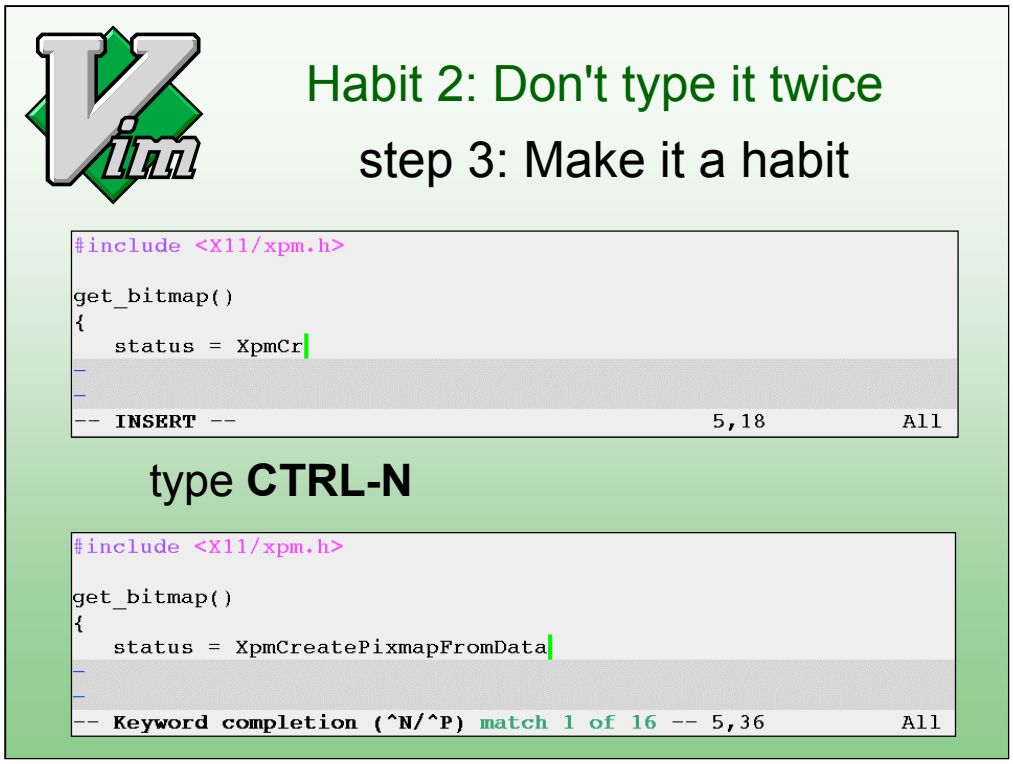
Function names can be very difficult to type. It takes a lot of key strokes and it's easy to make a mistake.

I see people use copy/paste for these names, but that still is not very effective.



This time you ask someone else if he knows a quicker way. He has run into the same problem before and found a good solution: Insert mode completion.

It's always a good idea to ask someone else, you don't have to invent the solution yourself.



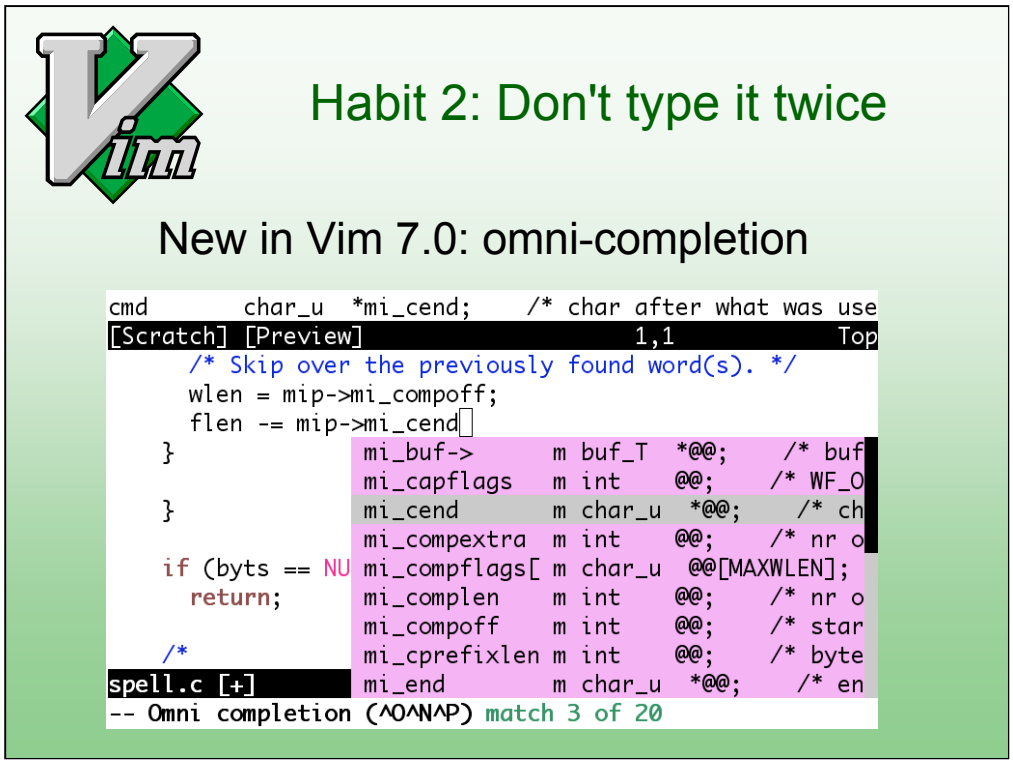
You start using it and find out what you need to type to complete it quickly.

If you don't type enough you get too many matches. Type CTRL-P to go back to the original word, type an extra letter and hit CTRL-N again.

If you don't get a match you forgot the #include line.

After doing this for a while you hardly type any long words.

It's also useful when typing unusual names in e-mail. You can create a dictionary for specific words you use and set the 'complete' option to use it.

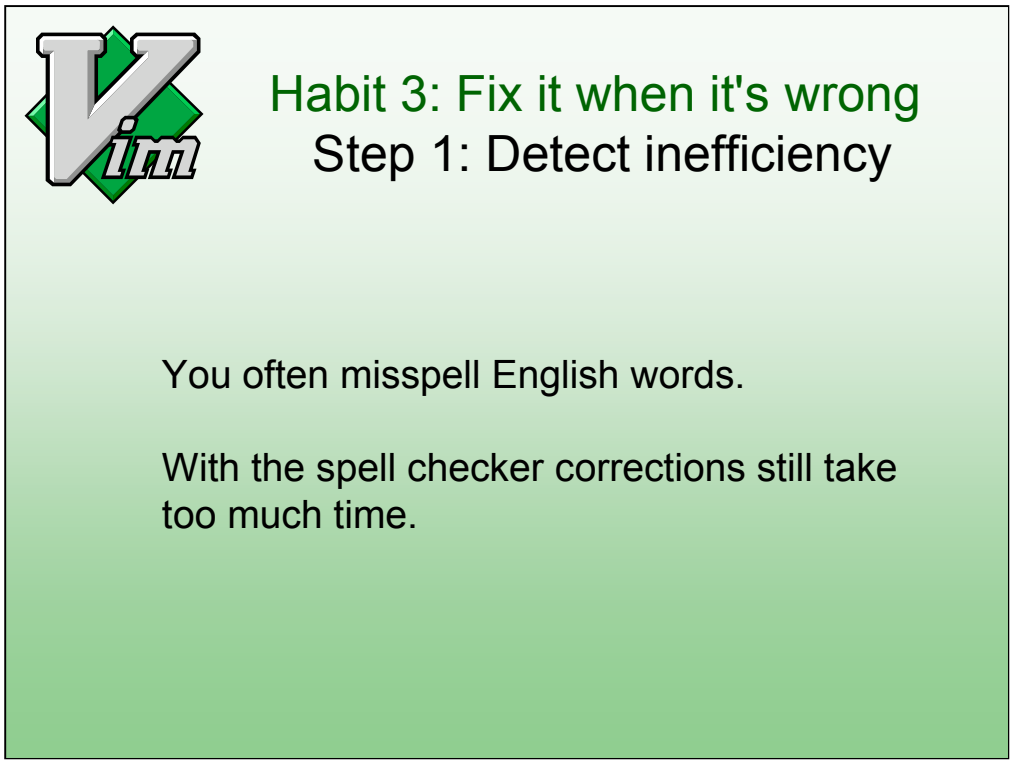


In Vim 7.0 a more advanced, context-sensitive completion has been added.

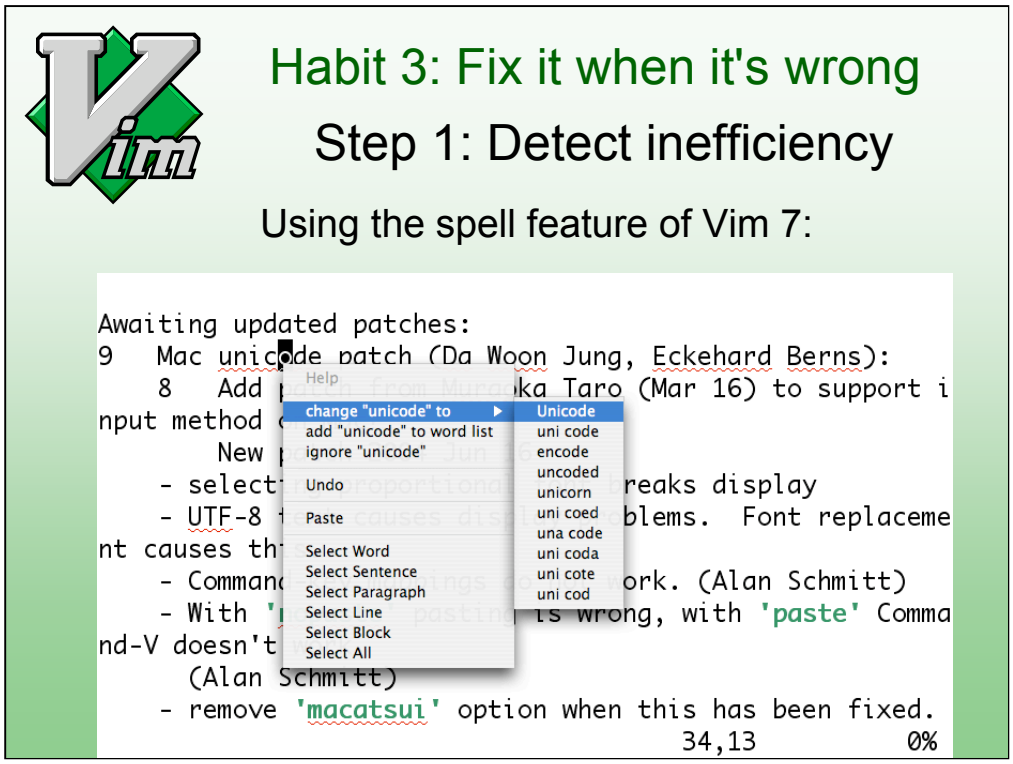
This needs to understand the language you are editing. That makes it slower, but more precise.

In the example you can also see the preview window, where the context of the currently selected match is displayed.

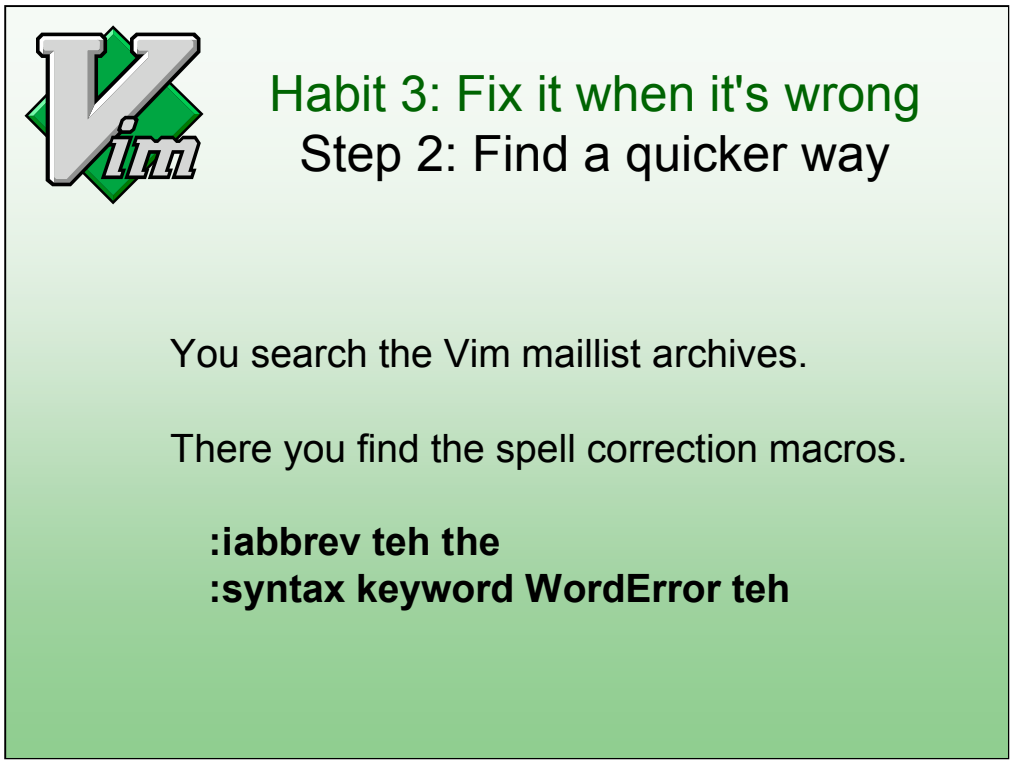
The menu is nice to see the overview of matches and quickly jump though them. It also works in a terminal, that's why it looks quite ugly.



When typing English text you often make mistakes. You have to proofread your text carefully or use the spell checker.



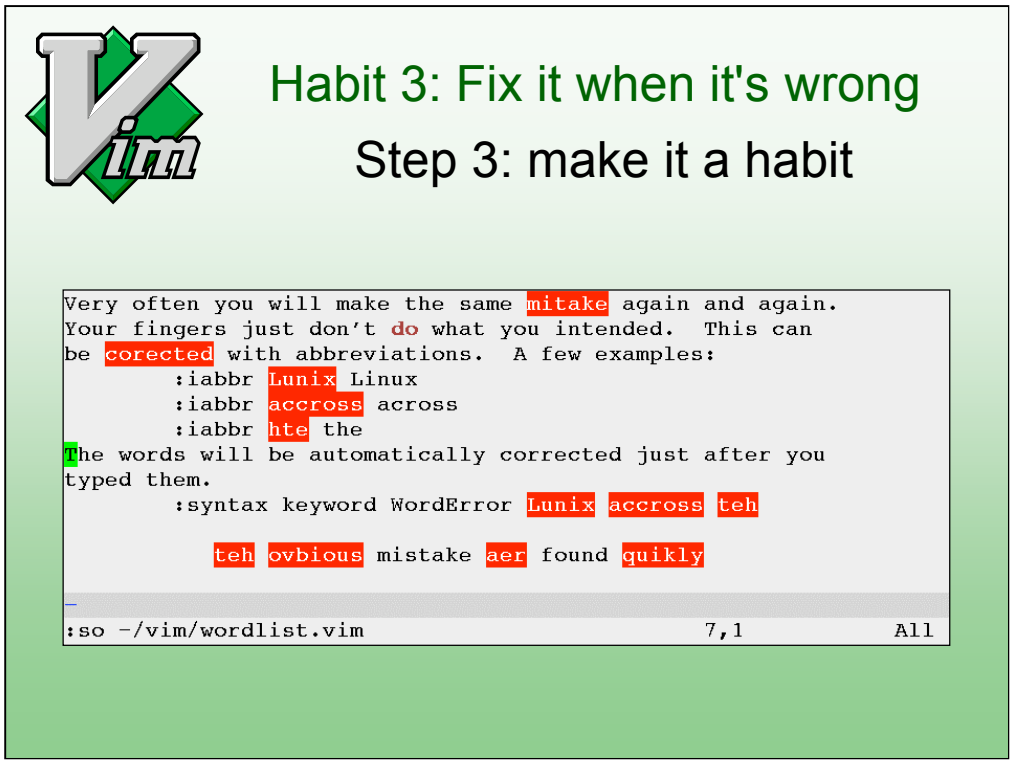
Now you spot typing mistakes when proofreading text. But you still need to perform a sequence of actions to correct them. That is OK for infrequent errors, but clumsy for mistakes you make all the time.



This time you look for a solution on the internet. The Vim maillist contains questions, answers and announcements for scripts that people made. You can find info about it on the Vim website:

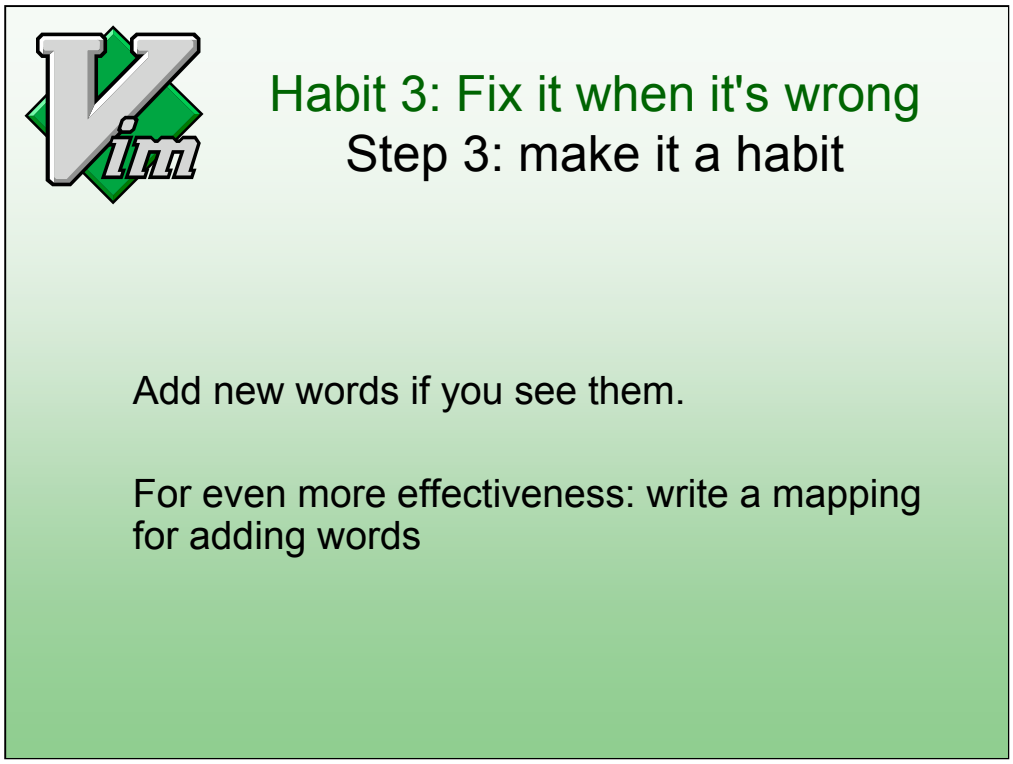
http://www.vim.org/maillist.php

This is a very useful resource, many questions have been asked and answered already.



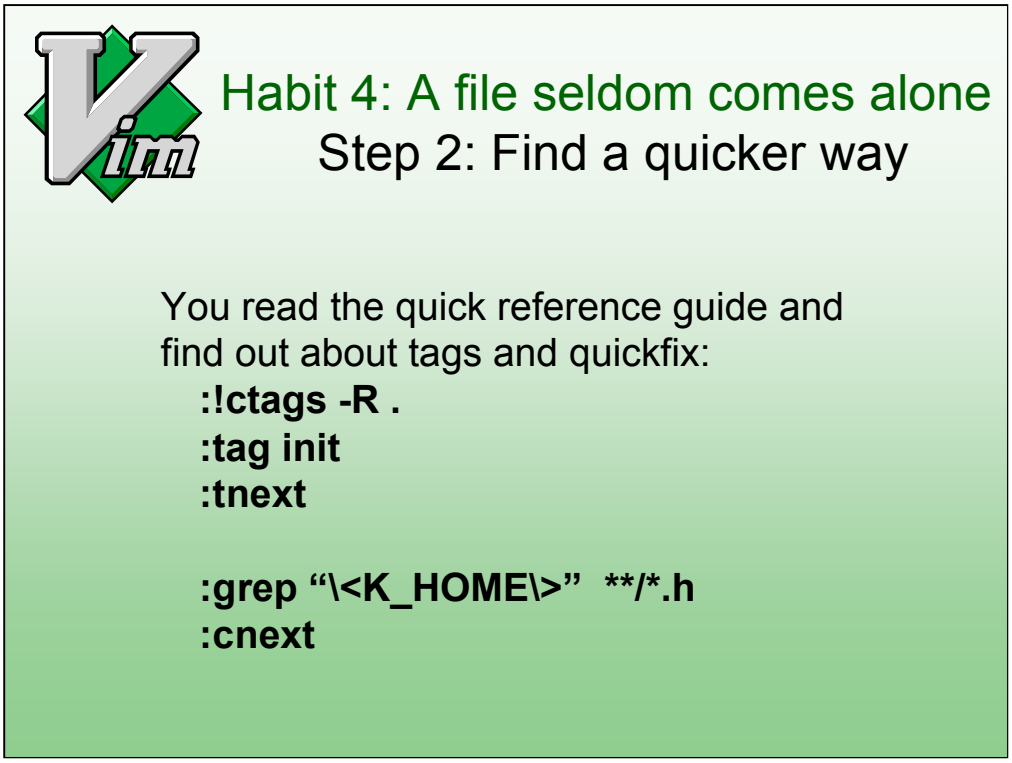
Now you spot typing mistakes while editing text. And they are automatically corrected when you type them.

Actually, it was a bit difficult to create this example, since the mistakes were automatically corrected when I typed them.



Whenever you spot a wrong word that isn’t detected yet you add it to the dictionary. If you do this often you can make a mapping for it.

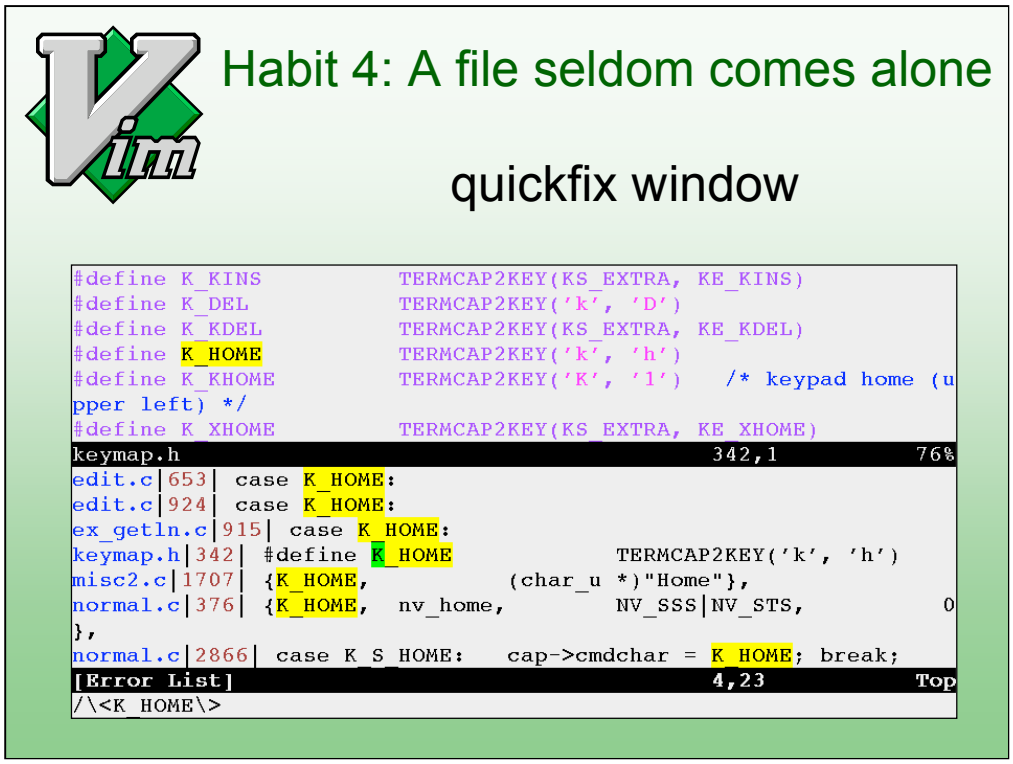
I have this problem every day. Especially now that I work for Google.



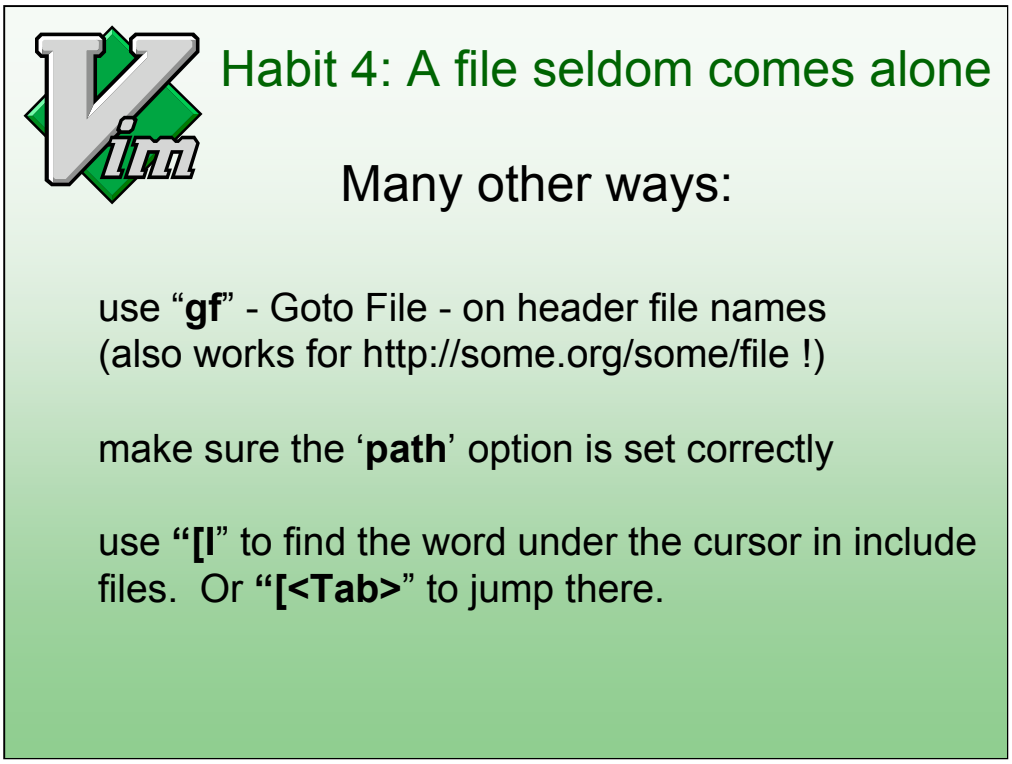
A tag file can be used to jump to where an item is **_defined_**. Exuberant ctags is recommended http://ctags.sourceforge.net/

Question: who of you hear about using tags for the first time?

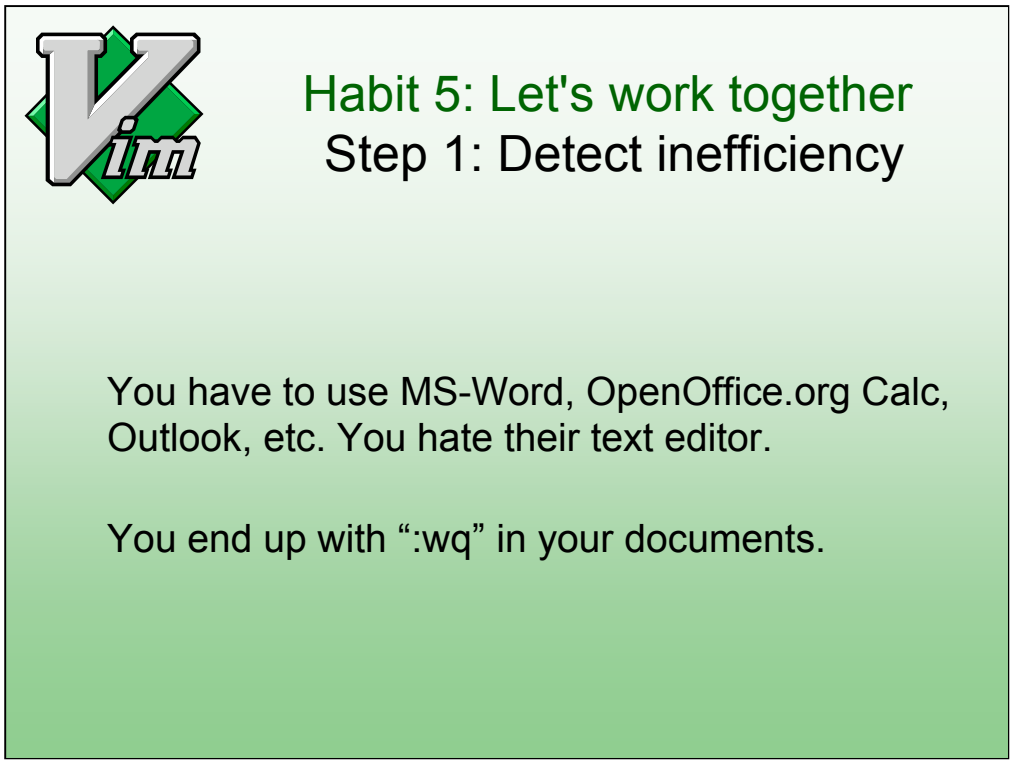
When searching for all places where a variable or function is **_used_**, the ":grep" command finds them. You can jump to each next item with ":cn".



The quickfix window lists all items from a ":make" or ":grep" command. This gives you an overview and allows quickly jumping to the location you want to see: Hitting <Enter> on a line displays the line with that error in the other window.



These are useful commands when browsing source code. They don’t require a tags file or anything, only that the ‘path’ option is set to find include files. Like the list of directories that the compiler uses.



Real power comes from programs working together. There are many programs that work with formatted text but have a bad editor, Vim is a good editor but does not do layout. Connecting the two will use the best of both.

Habit 5: Let's work together
Step 2: Find a quicker way

Check the MS-Word/OpenOffice.org/Outlook docs:
Can you select another editor? No.
You ask the Vim maillist if someone knows a solution. Response:

Edit the text in Vim with
:set tw=0 wrap linebreak

Copy the text between application and Vim through the clipboard.

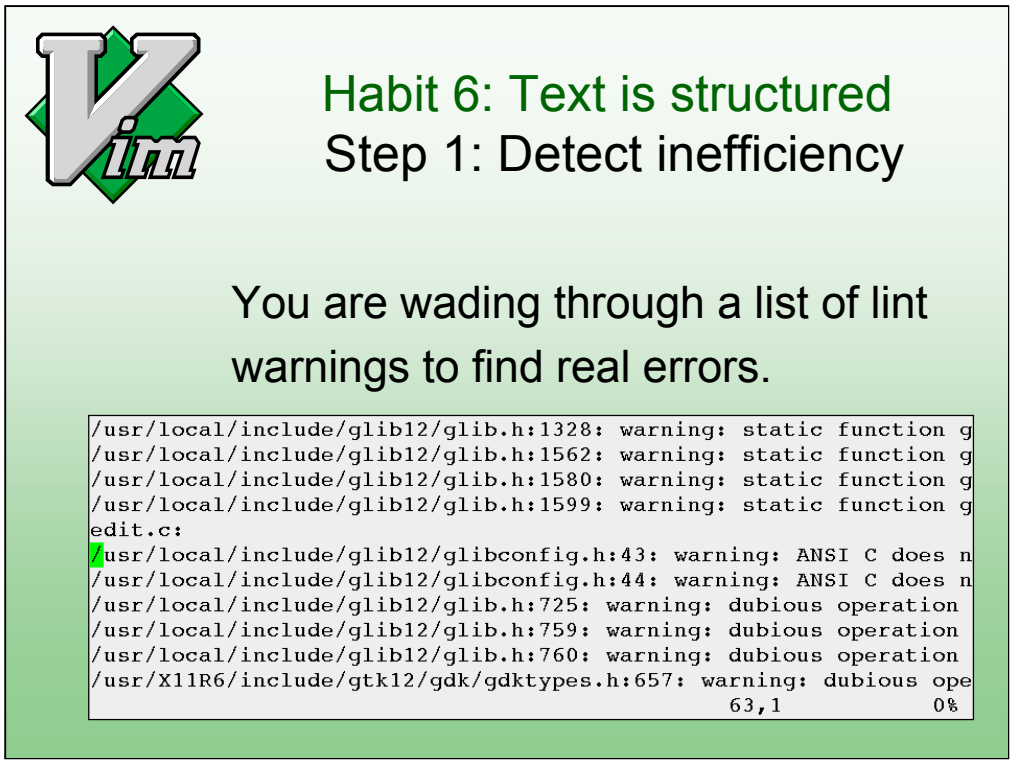
Unfortunately, you can't find an existing solution for this.

Second best is to copy/paste the text and edit it in Vim. You need to avoid inserting hard line breaks.

Another solution would be to remove the line breaks when copying to the clipboard. You can make a mapping for this.

Habit 5: Let's work together
Step 3: Make it a habit

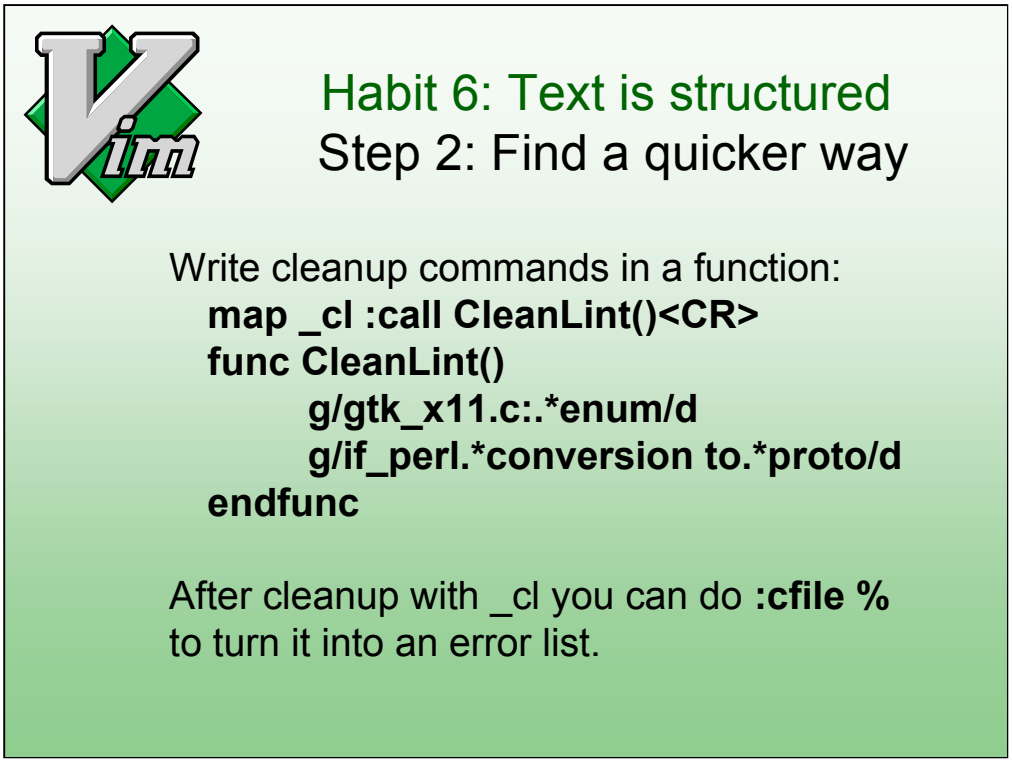
Still get :wq in documents...



Even though files contain plain text, it is often structured. You can use this to make your editing more productive.

If you have warnings in include files you can't avoid them. It makes it difficult to find the serious warnings that you need to take care of.

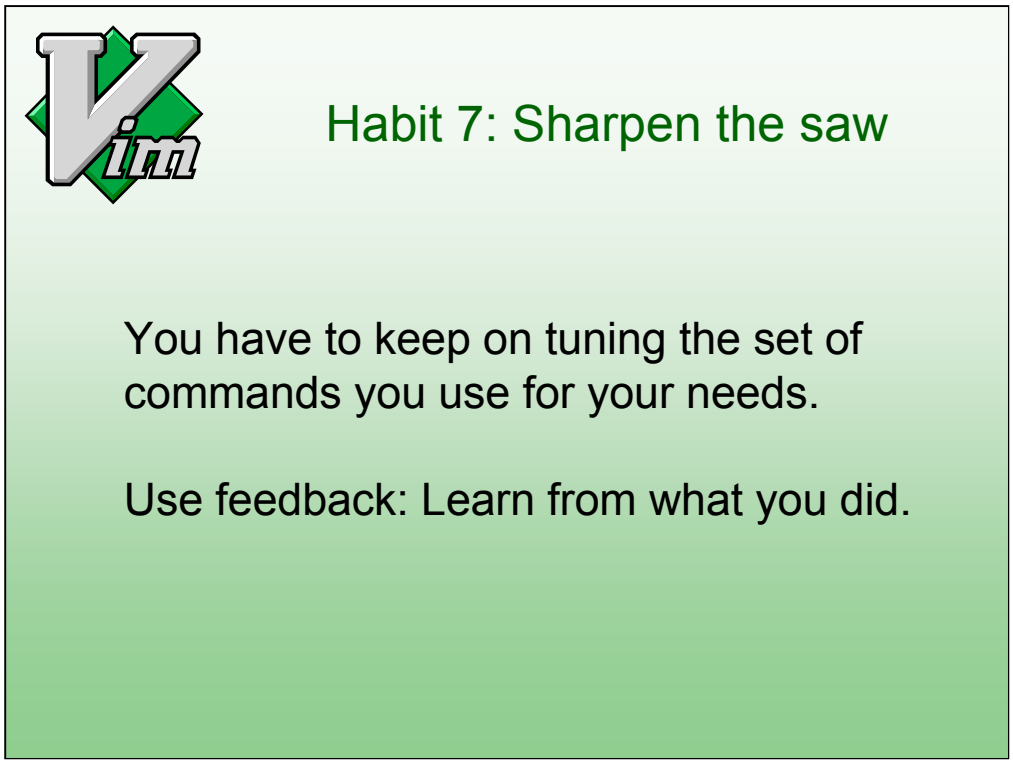
Logs files have a similar issue.



This time you decide to use the features the editor offers to extend its functionality. Most editors offer some script language. Vim has Vim script, which uses the same Ex commands that you type interactively.

Put these commands in your .vimrc file.

The trick here is to use the right patterns to only match the lines of harmless warnings. You need to tune this to avoid that serious warnings get deleted, or that harmless warnings clobber the list.

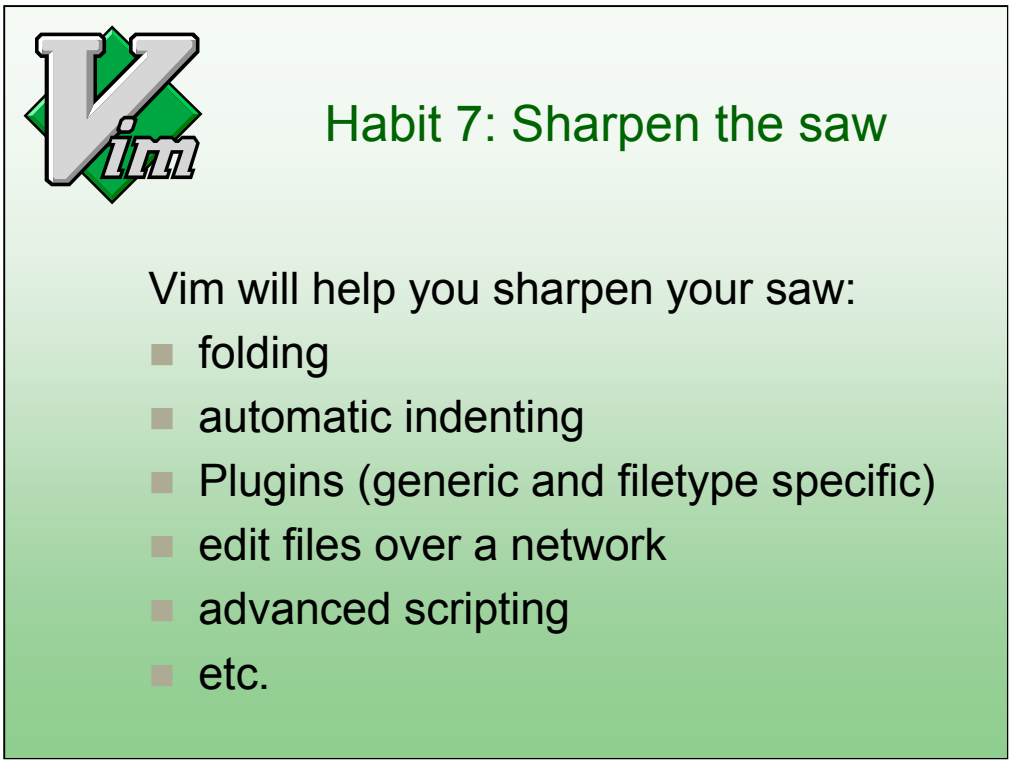


The third step seems to come down to “do it”. You might think that’s obvious and not important. The contrary is true. Only when you have made the task a habit will you be able to work at high speed.

Compare to learning to drive a car. The first few lessons you have to think about how to get it in the next gear. Only after practicing a while you can do it automatically and have time to think about where you are going.

Learning to use an editor is similar, with one important difference: It has many more commands. Too many to learn them all. You need to learn one at a time, when you need it.

On a meta level you should think about what you did with the editor. How much time did you waste on not using the best commands? How much time did you waste on finding a better way? How much time did you win by writing that macro? Use this to know what you need to do in the future.



Vim has many more features. All of them have been asked for by users. Thus it should make editing more effective for you too.

Automatic indenting is very flexible. You can define indenting for your specific needs. This avoids manually adjusting the indent. This uses plugins, thus can be changed easily.

Plugins make it easy to add functionality and exchange scripts between users. It's a matter of dropping the plugin in the right directory. They exist for generic use and specific to a certain filetype.

Editing files over a network saves you the manual commands to make a copy of a file and write it back later. This is actually done by a plugin. Since Vim 7.0 it also browses remote directories.

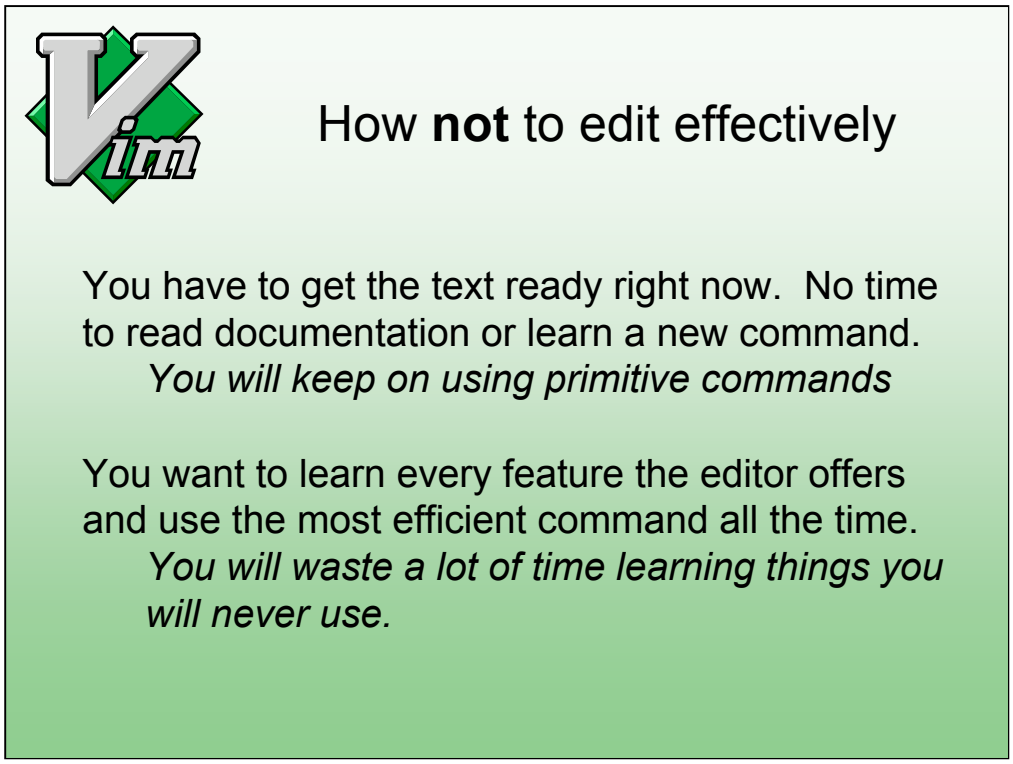
Vim 7 is currently available with all of this. Current work is mainly maintenance. Vim 7.1 will be a bugfix release.

Summary

Step 1: Detect inefficiency
- Find out what you waste time on

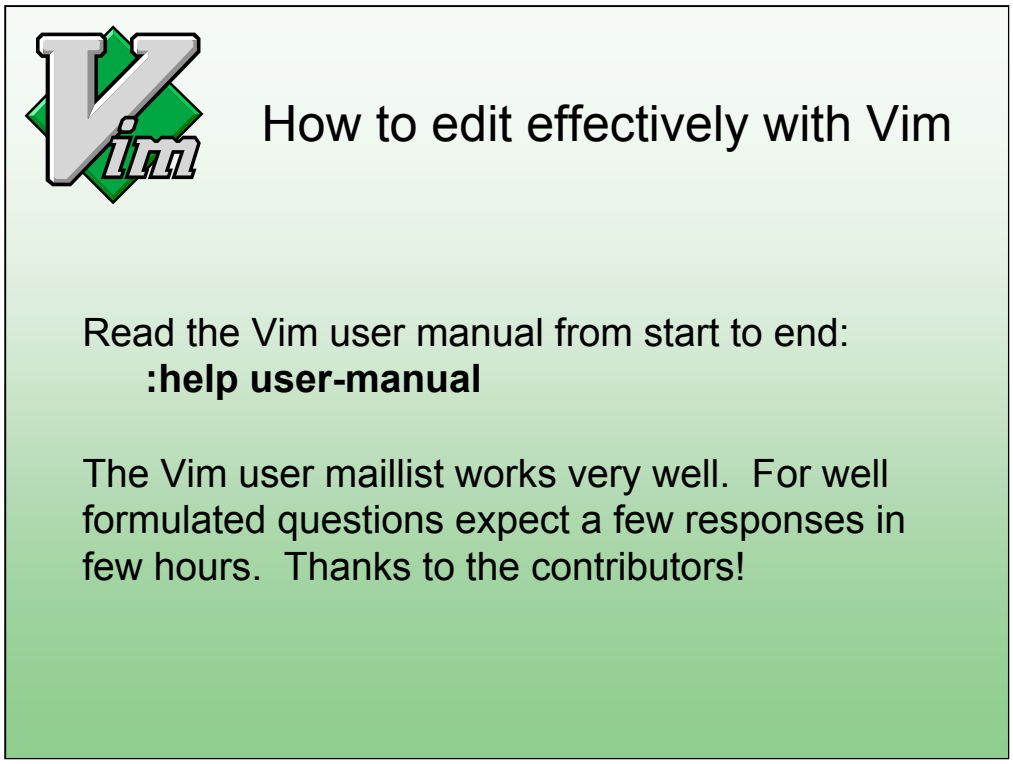
Step 2: Find a quicker way
- read the on-line help
- read the quick reference, books, etc.
- ask friends and colleagues
- search the internet
- do it yourself

Step 3: Make it a habit
- do it
- keep on improving



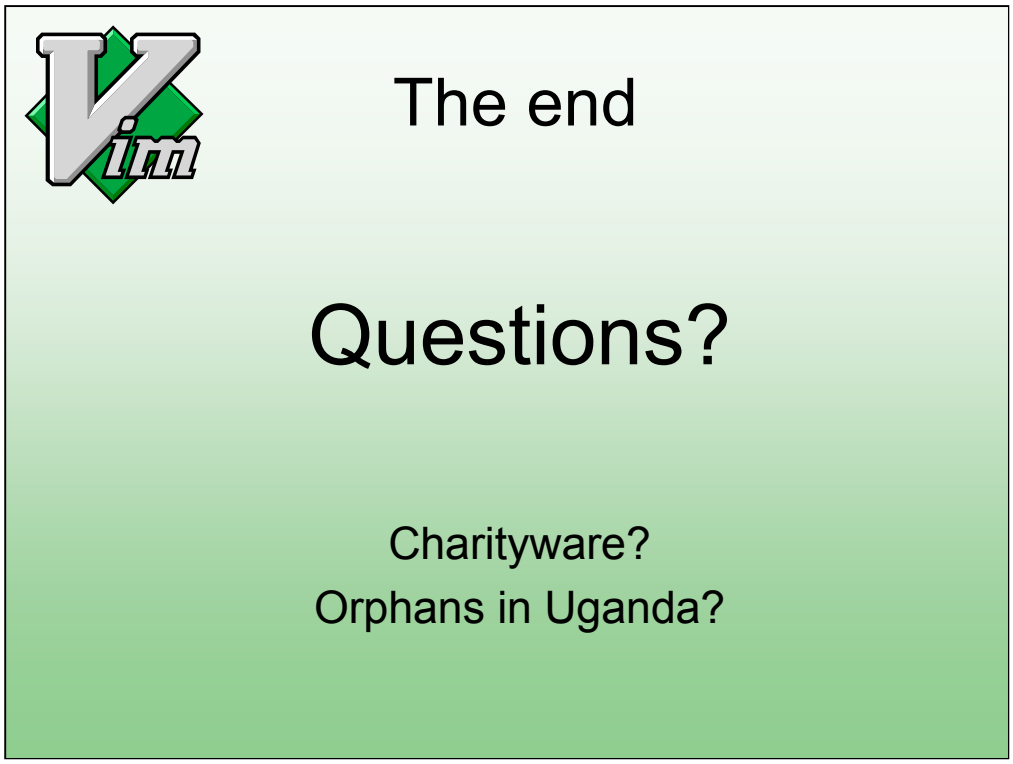
The first remark is obvious: If you don't learn you will remain primitive.

I must warn for a pitfall when overdoing it. Learning every feature of Vim might make you the great wizard of Vim, but it will not be very effective. And it will be impossible to make everything a habit.



The user manual is written in a way that it should be read from start to end. It started out as the first 20 chapters of Steve Oualline's Vim book. Images were removed, corrections made and new commands added.

I like to thank all the people that answer questions on the Vim maillist. They make many Vim users happy!



I will be around for more information.

I have some extra info about Charityware and helping orphans in Uganda.
