Concepts required in building an IDE/compiler [closed]

Asked 15 ye

20

Closed. This question needs to be more focused. It is not currently accepting answers.

Want to improve this question? Update the question so it focuses on one problem only by editing this post.

Closed 8 years ago.

Improve this question

When it comes to making an IDE (e.g. SharpDevelop) or a compiler/language parser, what topics of computer science do I need to know? I don't expect a full list of in depth tutorials but just a list of topics which would benefit me in improving.

Am I right in thinking a parser has some rules about the syntax/semantics of a language, and validates the code based on these rules? That seems like a simple approach?

Thanks

parsing compiler-construction ide

Share Edit Follow

edited Aug 7, 2010 at 15:42

Jon Smock
9,621 10 48 49

asked Jan 21, 2009 at 16:04

GurdeepS
66.8k 112 255 394

6 Answers

Sorted by: Highest score (default)

9

An IDE, a compiler and a debugger are three different beasts.

Here's a quick and slightly random selection of some links that I've found interesting or inspiring when thinking about building modeling tools for simulation languages, which is as close as I get to IDE:

- The High Performance GUI
- Magic Ink - programmable information rather than interaction

- Edward Tufte - imagine if your working life was spent looking at information made this beautiful.

- Information Design Patterns - general examples of GUIs
- Problems with tabbed interfaces (though icons aren't much use either when each page of code looks the same; maybe generated icons like the ones here would work)
- Common Sense Suggestions for Developing Multimodal User Interfaces - what if the best way to express your code is to wave your hands around, or give a bug a stern talking to.
- The Pinocchio Problem - similar are to Bracha's post, about patching a running framework

There's somewhat of a bias in those links towards patterns to help reading and browsing rather than writing code, and towards systems the user extends while using them rather than as a separate cycle; if you want a task-oriented interface or static plugins, projects for existing IDEs such as Eclipse are the place to look.

Share Edit Follow

answered Jan 21, 2009 at 16:44

community wiki
Pete Kirkham

6

For implementing a compiler / language, you will need a fundamental understanding of:

- BNF & EBNF - Context-Free Grammers (the syntax rules)
- Lexical Analyzing Techniques & Tools (Lex / Bison)
- Parsing Techniques (eg. Recursive Decent, LL, LR)

Share Edit Follow

edited Jan 21, 2009 at 16:16

answered Jan 21, 2009 at 16:08

mmattax
27.5k 41 117 151

4

Writing Compilers and Interpreters: A Software Engineering Approach [Paperback] Ronald Mak -> this is a great book to get started with. It leads you through the whole process of building a compiler and an ide with a debugger and many other things you need. At the end of the book you will have a good sense of what to do to branch out on your own.

You may also want to look at Language Implementation Patters by PragProg.com publishing too.

Share Edit Follow

answered Mar 31, 2011 at 4:45

Krispy
1,108 1 7 12

well in order to develop a compiler you need to grasp following topics

Answer saved to For later. ×

olue print of your language/compiler)

after done with theoretical part then you need develop following components for you compiler

1. Lexical Analyzer (validate tokens for you language)
2. Parser (validate sentences of you language)
3. Translation Scheme (convert your code (high level) into 3 address code (machine language code))
4. Virtual Machine (code Generation / produce actual out put of your code)

Note : out put of each component is input to next component and input of Lexical Analyzer is your actual code and mostly compiler is developed using Formal Method (a procedural design pattern)

Share Edit Follow

answered Apr 17, 2013 at 10:13

Haider Ali
800 2 9 22

If you're writing a compiler, a good Computer Science course in Theory of Language Translation or something similar is pretty much essential. MIT Open Courseware offers a "Computer Language Engineering" class along those lines. That should teach you the concept that mmattax mentions and provide a good start.

As for an IDE, that's really more of a desktop application project. You might be calling a compiler from your IDE, but you're not actually compiling code (though, to be fair, in a sophisticated IDE, you might be parsing code). So the knowledge required to build an IDE that calls an external compiler/linker would be more centered around the UI toolkit of whatever platform you're using, with perhaps a bit of compiler front-end theory (as you would learn in a compiler course) if you want to parse code.

Share Edit Follow

answered Jan 21, 2009 at 16:21

MattK
10.3k 1 33 41

Sorry but the answer is "The whole of computer science and years of practical experience".

-3

Its too big a subject for ordinary mortals and eclipse, intellij, netbeans and Visual... have the subject pretty well covered.

Lo
int

Answer saved to For later.

Share Edit Follow

answered Jan 21, 2009 at 16:27

James Anderson
27.4k 7 53 80

Why was this downmodded? +1 becaus I agree about the eclipse idea – Nifle Jan 21, 2009 at 18:29

5 -1 for pessimism and discouragement. No-one knows the whole of computer science, and much of it only existed after the first compilers were written. – Pete Kirkham Jan 21, 2009 at 20:17

1 Its not a bad thing to discourage someone from attempting a task that is too big for one individual. The range of skills required : GUI design, syntax parsing, build dependencies, source control etc. etc. might just be within the reach of an lone programmer, but, not a programmer that needed to ask the question. – James Anderson Feb 15, 2012 at 6:13