Background

I work as a professional software developer for about six years. During this period, I gained enough broad experience. I worked on different projects targeting different platforms:

- Linux Desktop apps
- iOS apps
- web services deployed on Linux (RedHat, Debian), and once on FreeBSD

Being opened for new technologies, I became acquainted (to a different extent) with programming languages from diverse families:

- C, C++, ObjC, ObjC++, Swift, Rust, Java, C#
- Bash, Perl, Ruby, Python, JavaScript, Lua, PHP
- Scala, OCaml, Scheme, Haskell, Elm

Interests

Despite the fact that I mostly worked on user-oriented apps I have a keen interest in 'low-level' programming, especially in the area of compilers and developer tools.

About three years ago I started diving deeper into compiler theory consolidating my knowledge by practicing on LLVM and Clang. Since then I made couple contributions to Clang. Also, I began working on my pet project: Mutation Testing System based on LLVM.

Links:

- `https://github.com/llvm-mirror/clang/commits/master?author=AlexDenisov`
- `https://lowlevelbits.org/circular-containers-in-objective-c`
- `https://lowlevelbits.org/nsvalue-and-boxed-expressions`
- `https://lowlevelbits.org/mutation-testing`
- `https://github.com/mull-project/mull`

Public Activity

Sometimes I give technical talks at various meetups and conferences. For example:

- "Compilation Process" at CocoaHeads Berlin, 2015
- "Magic Behind Xcode" at Mobile Warsaw, 2015
- "Using LLVM C API from Swift" at FOSDEM, LLVM room, 2016
- "Getting started with LLVM using Swift" at Russian Internet Technologies, 2016
- "Mutation Testing: Leaving the Stone Age" at FOSDEM, LLVM dev-room, 2017

Besides, that I organize LLVM user group in Berlin.

Links:

- `https://speakerdeck.com/alexdenisov/compilation-process`
- `https://speakerdeck.com/alexdenisov/magic-behind-xcode`
- `https://archive.fosdem.org/2016/schedule/event/llvm_c_swift`
- `http://appsconf.ru/2016/abstracts/2165`
- `https://www.meetup.com/LLVM-Social-Berlin/`
- `https://www.youtube.com/watch?v=YEgiyiICkpQ`

Personal Development

I firmly believe that the most valuable resource is Knowledge. Having this in mind I try to share what I know with others via blogging:

- "Low Level Bits" has different topics from iOS development to hacking on Clang and LLVM
- "System Under Test" describes how various Open Source projects (LLVM, FreeBSD, etc.) are tested.

Blogging obviously helps to learn, but this is not enough. To fill the gaps I read technical literature (books, scientific papers, articles) and participate in MOOCs, mostly on Coursera.

Blogs:

- `https://lowlevelbits.org`
- `https://systemundertest.org`

Tech books I read recently:

- Introduction to Automata Theory, Languages, and Computation
- Growing Object-Oriented Software, Guided by Tests
- The Deadline: A Novel About Project Management

Favorite papers:

- The Correctness-Security Gap in Compiler Optimization
  `https://nebelwelt.net/publications/files/15LangSec.pdf`
- The Linux Scheduler: a Decade of Wasted Cores
  `https://www.ece.ubc.ca/~sasha/papers/eurosys16-final29.pdf`
- Simulating Physics with Computers
  `https://people.eecs.berkeley.edu/~christos/classics/Feynman.pdf`

MOOCs I have finished:

- Compilers
  `http://web.stanford.edu/class/cs143`
- Functional Programming Principles in Scala
  `https://www.coursera.org/learn/progfun1`
- Automata
  `http://online.stanford.edu/course/automata`
- From Nand to Tetris
  `https://www.coursera.org/learn/build-a-computer`
- Synapses, Neurons and Brains
  `https://www.coursera.org/learn/synapses`

Delight

I didn't create anything innovative, though I have built few tools I'm proud of.

First one, xcconf, is a small program that converts YAML into Objective-C. Its purpose is to handle iOS/OS X app configuration for a specific environment, e.g. different API secret and different server URL for 'staging' and 'production'.

The second tool called Components. Its purpose is to provide a stable, robust, and flexible way to manage dependencies. The system is very simple: there is a directory with a bunch of makefiles, shell script goes over those makefiles and runs 'make install'. This way installation is controlled by each makefile and 'format' agnostic, i.e. makefile could just fetch sources and copy them into a project directory, or fetch a binary, or fetch sources and build them, or whatever else maintainer can imagine. It's very similar to the Ports system in *BSD world.

I used both tools for ~2 years and had no problems at all, unlike other solutions I used to solve same problems.

Another minor thing I'm happy about is that one of the features I brought to Clang was presented at Apple's WWDC session "What's new in LLVM".

Links:

- `https://github.com/alexdenisov/xcconf`
- `https://github.com/alexdenisov/components`
- `https://developer.apple.com/videos/play/wwdc2016/405`
- `https://lowlevelbits.org/yaml-based-configuration-for-objc-projects`
- `https://lowlevelbits.org/components`
- `https://lowlevelbits.org/circular-containers-in-objective-c`

Future Plans

My long-term goal is to work on low-level stuff and developer tools.

My mid-term goal is to refresh algorithms, data structures, and CS in general since I feel that I have many gaps there.

My short term goal is to finally finish reading the book "How to Read and Do Proofs."