DEVELOPER ESSENTIALS – v1.0

What Every Aspiring 'Software Engineer' Must Know?

Santosh S Malagi

santoshsmalagi@gmail.com

WHAT THIS IS ABOUT?

Hi Folks!

Hope everyone is keeping safe during these tough times. It is a kind of a 'messy situation' for both students and teachers alike. Students are worried about completing their studies in time, those in the final/pre-final years are tensed about what the future holds for them. Though I cannot give you a definitive answer, of how all of this will unfold, all I can say is you will make a career, sooner or later - if you keep at it. Find your interests and pursue them. Invest this time in getting better and better, learn one domain and do it very well. Use the time at hand, explore resources and try seeking out break out opportunities.

Maybe I can't get you jobs, but sure I can tell you what not to do! and what to do!!

This is one such effort in that direction, and I call it 'Developer Essentials'. A collection of the basic skills very aspiring engineer must develop. Trust me – majority of the careers and domains you will choose in the future, each of these skills I discuss here will manifest in one form or the other. The first 3 weeks to 6 months will be spent in acquiring these skills, and you will continue to add neat little tricks as you go further. So invest time in gaining these skills and you will be well rewarded. You don't have to spend a single penny, just Google around, search YouTube, scrape through StackOverFlow and Reddit. You will have all the answers you need. If still you are in doubt, reach out to me, be patient I will reply.

Let's get you started!

Warm Regards,
Santosh S Malagi, 2KA09EC041
santoshsmalagi@gmail.com, 9th Oct 2020

WHO THE HELL I AM TO TELL YOU ALL THIS?

Well,

- I graduated from the Dept. of ECE, SKSVMACET in 2013
- Worked as a QA Engineer testing Embedded Firmware at Mphasis, Ltd
- Graduated with an MSc in Computer Engineering from TU Delft, the Netherlands in 2018 (one of the top 20 universities in EEE in the world)
- Worked as a Research Student at IMEC, Leuven Belgium (one of the best place in the world for advanced research in microelectronics)
- 2+ years of experience working as an R&D engineer for an EDA major developing some cool technology, which is used by major semiconductor design houses to make chips which go in your cars

- Passionate geek
- Had to pay a heavy cost, because I did not know these skills
- Learnt the skills mentioned in this guide the hard way

HOW TO USE THIS GUIDE?

1. There are 4 skills in this guidebook. You need to start at the beginning and go through each of them, sometimes in parallel.
2. Every step outlines a theme – e.g. Learn Shell Scripting.
3. Now you need to probe deeper and search the various online resources available and start picking up this knowledge.
4. There is no set syllabus, but I try to outline the major topics.
5. I have also provided some pointers to good YouTube playlists which you can use.
6. You can't learn anything by just reading, you need to implement stuff!
7. Maybe form a small study group and encourage each other?
8. If you don't understand any terms in this guidebook, do some homework – search around.
   E.g. what is a 'Systems Programmer'

Why this guide does not talk about programming languages, data structures and algorithms?

Well, my friend! My goal is to teach you – 'Developer Essentials'. The very essential skills and tools you will need as a programmer, developer, tester – basically as an Engineer! Whether you plan to write software which runs on supercomputers, or plan to be a Full Stack developer, or want to work as a VLSI Design Engineer the tools mentioned in this guide will be your workhorses. So make sincere efforts to be comfortable with them.

Mind you – don't try to master them, it is impossible! but try to build a portfolio of the major and most essential topics from each of these.

GET COMFORTABLE WITH THE LINUX CLI

The most basic knowledge which you need to acquire (most students have this already) is to work on the Linux Command Line Interface (CLI). You need to be hands on with the CLI, and try avoiding the GUI based Window mentality! Of course this is from a 'Systems Programmer' perspective. Many Android, and Swift, Front-End roles etc. tend to use a GUI based environment, but this should be true for them as well.

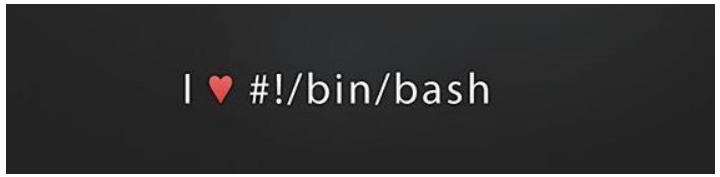
- Step – I : setup a Virtual Machine using Oracle VirtualBox or VMware Player
  - this is better than dual booting, because if you break something, you can fix it easily
- Step – II : pick one Linux OS (*either Ubuntu or CentOS, but you are free to try others) and stick with it.
- Step III : now learn everything mentioned below:
  - The Linux philosophy, OS – constituents, various flavors and branches, what is the kernel, GUI systems etc.
  - What is a shell? different types of shells available, terminal
  - Basic navigation – ls, pwd, cd, cp, mv, mkdir, rm, touch, date, time, file, type, which, where, alias etc.
  - Getting help – man pages, apropos,
  - File system hierarchy, file permission management – chmod, umask, su, sudo, chown,
  - User environment configuration and startup files – i.e. bashrc, the prompt and modifying it
  - Installing software and package management, processes in Linux – ps, kill etc.
  - Searching for files – locate, find,
  - Text redirection and manipulation – redirecting text, pipes, cat, grep, more, less, head, tail, tee, sort, uniq, sed
  - Expansions – tilde, pathname, brace, double and single quotes
  - Mounting and unmounting devices,
  - Archives and backup – gzip, unzip, tar, rsync etc.
  - Resources to get you started:
    - http://linuxcommand.org/
    - https://www.codecademy.com/learn/learn-the-command-line
    - https://www.youtube.com/playlist?list=PLS1QulWo1RIb9WVQGJ_vh-RQusbZgO_As
    - https://www.youtube.com/playlist?list=PLy7Kah3WzqrHPrgkBgwzXyfDDCvthdUfl
    - https://www.youtube.com/playlist?list=PLypxmOPCOkHV4vnWkQCasSjBOI8ZpkmMY
    - https://www.youtube.com/playlist?list=PLW5y1tjAOzI2ZYTIMdGzCV8AJuoqW5IKB

LEARN SHELL SCRIPTING

Once you are good with the CLI, the next logical step is to string those command together and start creating shell scripts. If you have learnt the CLI well then, this task is half done, all you need to do is to acquire some knowledge about the shell script syntax itself. Again beware that the syntax changes slightly from shell to shell, though the overall concept remains the same. e.g. setting an alias in bash is slightly different from doing it in csh. So the trick here is to stick with something universal – say bash.

- Basically everything mentioned below:
  - Brace expansion
  - Regular expression basics:
  - What are shell scripts?
  - String, array and number representation in shell scripts
  - Functions
  - Branching with if-else, and case
  - Looping with while, until and for
  - Reading keyboard input
  - Positional parameters

  - Resources to get you started:
    - https://www.shellscript.sh/
    - https://www.learnshell.org/
    - https://www.youtube.com/playlist?list=PLS1QulWo1RIYmaxcEqw5JhK3b-6rgdWO_
    - https://www.youtube.com/playlist?list=PL2qzCKTbjutJRM7K_hhNyvf8sfGCLklXw
      https://www.youtube.com/playlist?list=PL2qzCKTbjutJRM7K_hhNyvf8sfGCLklXw



PICK A TEXT EDITOR (vim or emacs) AND STICK WITH IT!

While you learn to write shell scripts, the next very important thing to do is to learn a powerful text editor. I know many of you might have used Notepad++, Sublime or Atom. Or the adventurous amongst you might have already learnt the basics of vim (new avatar of vi) or emacs. If not, then this is the time to do so! In a Linux CLI environment, these text editors will provide you with a host of features and that way God will save you from being a slave to a particular IDE.

- Well, I am a Vim fan, and I love it! I have no idea of Emacs, please seek some online resources, the following is for Vim only:
  - vim philosophy, modal editing, opening and closing files, navigation, yank
  - search and substitution
  - file manager – netrw
  - tabs and splits
  - undo and redo
  - visual mode
  - configuring vim - .vimrc
  - advanced (only for the more adventurous) – registers, buffers, macros, mapping keys, navigating code base with ctags
  - very advanced – vimscript (things I don't know!)

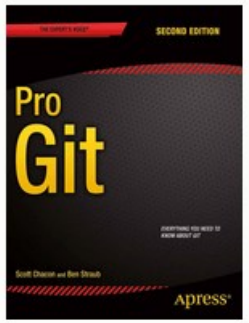
  - Resources to get you started:
    - https://www.youtube.com/playlist?list=PLy7Kah3WzqrEjsuvhT46fr28Q11oa5ZoI
    - https://www.openvim.com/
    - https://github.com/iggredible/Learn-Vim

VERSION CONTROL - git

Okay, any serious programmer now needs to start learning about version control. This way you can also start developing some cool stuff and start contributing to open source projects – one of the best ways for you to get noticed as a young kid on the block.

- Udacity's course on git
  - https://www.youtube.com/playlist?list=PLAwxTw4SYaPk8_-6IGxJtD3i2QAu5_s_p

- Pro Git – free book
  - https://git-scm.com/book/en/v2



2nd Edition (2014)

the end is a new beginning.....