Newbie-Guideline

해당 문서는 프로그래밍/개발에 입문하려는 뉴비분들(컴퓨터학과 신입생, 비전공 개발 입문자 등)과 새로운 공부 리소스를 찾으려 하는 모든 분들을 위한 지침서입니다.

Contents

    - etc.
- Fundamental Technique
    - Algorithm
        - Academic
        - Coding Interview (Coding Test)
        - Competition
    - Computer Architecture
    - System Programming
    - Design Pattern
    - Operating System
    - Network
    - Web
    - Database
- Professional Technique
    - Android Application
    - iOS Application
    - Game
    - Embedded System
    - System Engineering
    - Software Engineering
    - Security
- Study
    - Roadmap
- Projects
    - 개인 프로젝트
    - 대외 활동
    - 해커톤
    - 대회
- Comments
    - Community
        - 개발 관련 커뮤니티 리스트
        - 커뮤니티 이용법
    - Q&A
    - 조언
        - Web
        - System
        - Embedded
        - Network

        - Game

Prerequisite

프로그래밍은 다른 분야와 비교하여 선수지식이 적은 편이라고 생각하지만, 그럼에도 불구하고 알아두면 좋은 선수지식이 있습니다. 특정 프로그래밍 분야들은 심도 있는 선수지식이 필요한 때도 있긴 하지만, 아래 내용 하나만큼은 일반적으로 도움이 되는 선수지식입니다:

- 영어 : 보통 선수지식으로 수학을 생각하는 분들이 많은데, 의외로 정말 필요한 선수지식은 영어입니다. 수학은 분야에 따라 많이 쓰이기도 하지만 전혀 쓰이지 않을 수도 있습니다. 반면에 영어는 공부를 하다 보면 필수 불가결한 스킬입니다. 책이나 문서 등 양질의 공부 자료들은 대부분 영어로 작성되어 있으며, 무엇보다 공부하는 과정에서 대단히 많은 검색을 하게 될 텐데 대부분의 원하는 결과는 Stack Overflow나 Quora라는 해외 사이트에서 찾을 수 있습니다. 영어 능력에서 오는 차이는 고급 개발자가 될수록 더욱 심화하므로, 본인의 영어 능력이 부족하다고 한국어로 되어있는 책이나 자료만 고집하지 말고 천천히 공부해서 영어 자료들에 친해지도록 합시다.
    - IT 개발자의 영어 필살기 : IT 업계에서 주로 사용하는 표현과 단어들을 접해볼 수 있습니다. 난이도는 굉장히 쉬운편이고 굉장히 얇아서 해당 책으로 새로운 공부를 한다는 느낌은 아니고, 해당 책을 통해서 앞으로 어떤 영어를 어떻게 공부할 지 길라잡이로 사용할 수 있습니다.
    - English for Developers : IT 업계에서 주로 사용하는 표현과 단어들을 어느 정도 익힐 수 있습니다. 난이도가 높진 않지만 위 책보다는 어려운 편이며, 어느 정도 영어 표현들에 대한 학습을 할 수 있습니다.
    - Grammar in use intermediate : 학창 시절 한 번씩은 들어봤을 유명한 영어 문법서입니다. 해당 책을 두세 번 정독하고 나면 영어를 읽고 쓰는데 확실히 도움 되는 것이 체감됩니다.
    - Computer Science and Technology Vocabulary : Computer Science(이후 CS)와 관련된 용어들 리스트와 간단한 뜻이 적혀 있습니다. 한 번씩 눈에 익혀두면 좋습니다.
    - Glossary of computer science : 위와 마찬가지로 용어 리스트와 간단한 뜻들이 적혀 있습니다. 한 번씩 익혀두면 좋습니다.
    - 위 자료들은 굉장히 빠르게 학습할 수 있는 내용입니다. 가볍게 학습하고 나서, 주기적으로 원서를 읽고 Stack Overflow나 Quora 글들을 보며 영어에 친숙해지도록 합시다.

Computer Language

개발 공부를 시작하면서 가장 처음 접하게 되는 것은 아무래도 컴퓨터 언어일 것입니다. (간혹가다가 컴퓨터구조 같은 것들을 공부하며 입문하는 예외적인 케이스도 보긴 했습니다만..) 컴퓨터 언어는 말 그대로 컴퓨터에서 동작하는 소프트웨어(프로그램)를 작성하기 위한 언어입니다. 처음 공부하는 분들의 입장에서는 정말 수많은 컴퓨터 언어 중에 무엇을 공부하는 것이 가장 좋을지 막막할 수도 있습니다. 또, 어떤 분은 대체 왜 이렇게 많은 컴퓨터 언어가 존재하는지 의아해할 수도 있습니다. (이런 의문을 품는 것은 매우 좋습니다)

컴퓨터 언어들은 저마다 고유한 특색을 가지고 있고, 장단점이 있습니다. 이러한 언어의 특성 때문에 특정 컴퓨터 분야에는 특정 언어가 어울리게 됩니다. 예를 들어, 안드로이드 앱 개발을 할 때는 Java와 Kotlin이라는 언어를 주로 쓰게 됩니다. (NDK개발을 한다면 C/C++도 많이 쓰게 됩니다) 웹 개발자라면 Javascript와 Java를 많이 다루게 될 것이고, 게임 개발자라면 C++이나 C#을 많이 사용할 것입니다. 시스템 개발자라면 C와 C++를 주로 쓰게 될 것이며, 윈도우 응용 프로그램 개발자라면 C#과 일부 C++을 쓸 것입니다. 혹은, 정말 어쩌면, 매니악한 분야를 좋아하여 로우 레벨 개발을 하게 된다면 C와 어셈블리를 사용하게 될 것입니다.

이처럼 많은 분야와 많은 언어가 있는데 이들을 모두 공부해야 하는 것은 아닌가 걱정이 될 수 있습니다. 하지만, 언어는 어디까지나 도구일 뿐이며 해당 언어를 사용하여 무언가를 만들어내는 능력이 중요합니다. (물론, 컴파일러나 언어론 자체를 전공하는 예외적인 케이스는 제외합니다) 그리고 보통 하나의 언어에 숙달하고 나면 다른 언어를 익히는 시간은 굉장히 줄어듭니다. 제 경우 처음 공부한 언어가 C언어였는데, 언어를 처음 공부한 시점부터 해당 언어로 무언가 그럴싸한 프로그램을 만들어내기까지 거지는 6개월이라는 시간이 걸렸던 거 같습니다. 반면, 자바를 공부하기 시작한 지 1주일 만에 정부 사업으로 진행한 앱을 만들었고, 파이썬을 공부한 지 이틀 만에 그럴싸한 프로그램을 만들어냈습니다. 이러한 극단적인 시간 축소에는 언어의 특성적인 측면도 있긴 하지만 프로그래밍 자체에 대한 숙련도 증가로 인한 학습 속도 향상이 지대했습니다. 즉, 여러분은 (취미로 여러 언어를 공부하는 경우를 제외하면) 여러 언어를 공부하는 데에 시간을 과투자할 필요가 없습니다.

그렇다고 언어를 익히는 것이 중요하지 않다는 것은 아닙니다. 도구 자체의 사용법을 익히는 것 또한 필요합니다. 개인적으로 주 무기로 사용할 언어 두 개와 보조 무기로 사용할 언어 하나 정도 익혀두는 것을 추천합니다. 자신이 어떤 분야에서 일을 하게 될지, 어떤 분야를 세부적으로 전공할지 미리 정할 수 있다면 그에 맞춰서 주 무기를 구성하는 게 베스트지만, 그렇지 않은 경우가 더 많을 것입니다. 그렇다면 한국에서 일할 것이라면 Java와 C++을 주 무기로 삼고, Python을 보조 무기로 삼는 것을 추천합니다. Java는 한국에서 가장 많이 사용하는 언어이고, C++는 잘 공부해두면 다른 언어들을 쉽게 익힐 수 있으며, Python은 범용적으로 쓸 수 있기 때문입니다. 물론, 자신이 끌리는 언어가 따로 있다면 해당 언어를 주 무기로 삼아도 전혀 무방합니다. 참고로 위에서 주 무기라고 하면 해당 언어로 자신이 원하는 로직을 자유롭게 짤 수 있으며 해당 언어의 장단점이 무엇인지 알고 특색을 살려 프로그래밍 할 수 있는 수준을 말하고, 보조 무기라고 하면 해당 언어로 쓸만한 프로그램을 구현할 수 있는 수준을 말합니다.

그 외에도 어떤 언어로 입문을 할 것인지도 매우 많은 분들이 궁금해하는 토픽 중 하나입니다. 컴퓨터공학과 신입생이라면 학교 커리큘럼 상 처음 배우는 언어로, 비전공자분들 중 원하는 분야가 있다면 해당 분야에 적합한 언어가 있기에 해당 언어로 시작하면 되지만, 그렇지 않은 분이라면 Python으로 입문하는 것을 굉장히 추천합니다. Python은 굉장히 범용적이며 언어에 내장된 기능이 다양하고 프로그램을 빠르게 만들어낼 수 있는 효율적인 언어입니다. 그뿐만 아니라 익히기도 쉽고 코드 생김새 자체가 깔끔하여 입문용으로 제격인 언어입니다.

이제 아래에는 대표적인 언어들을 공부하기 위한 소스들을 추천해드릴 것입니다. 그런데 이는 세부 설명을 보면 알겠지만, 입문서부터 고급서까지 다양하기 때문에, 필요에 따라 선택하여 학습하면 됩니다. 해당 언어로 입문하여 주 무기로 사용할 예정이라면 전부 읽으면 좋겠지만, 보조 무기로 사용할 것이라면 기초 서적만 봐도 무방합니다. 혹은, 이미 프로그래밍에 익숙해졌는데 특정 언어를 사용하게 됐다면 입문서를 건너뛰고 레퍼런스 북을 볼 수도 있습니다.

Python

```
print('Hello world!')
```

파이썬은 바로 위에 말한 대로 굉장히 범용적으로 쓰이는 언어입니다. 귀찮은 수작업을 자동화하기 위한 개인 툴부터 요즘 인기 많은 AI 분야나 대규모 네트워크 프로그램을 만드는 데에도 사용됩니다. 또한, IT 회사 입사를 위한 코딩 테스트에서 사용하면 가장 효율적이고 유리한 언어 중 하나입니다. 굉장히 장점이 많은 언어긴 하지만, 몇 가지 한계점 때문에 보조 무기로 익혀 두는 것을 추천합니다.

- 점프 투 파이썬 : 한국에서 전통적으로 유명한 파이썬 입문 자료입니다. Do It 종이책 시리즈로도 출판되었는데, 웹에서 무료로 볼 수 있습니다. 기초적인 파이썬 내용을 굉장히 쉽게 설명하므로 입문 자료로 사용하기 제격이며, 프로그래밍을 처음 하는 사람도 어렵지 않게 따라 할 수 있습니다. 다만, 파이썬의 세세한 내용을 설명하지는 않기 때문에 입문서 이상의 역할을 하기는 어렵습니다. 프로그래밍을 처음 접하는 분들에게 추천하며, 다른 언어를 공부한 경험이 있는 분은 바로 뛰어넘어서 아래 책으로 시작하는 것을 추천합니다.
- 처음 시작하는 파이썬 : 세계적으로 유명한 파이썬 입문서 중 하나입니다. 점프 투 파이썬보다는 세세하게 설명하고 있으며 파이썬을 이용하여 응용 프로젝트를 만들어내는 내용도 조금 담고 있습니다. 파이썬 언어를 처음 공부하는 분들에게 추천해 드리는 서적입니다. 다만, 해당 책으로 프로그래밍 공부에 입문했는데 내용이 어려워 이해하기 힘들다면 위의 점프 투 파이썬을 한번 보고 오는 것을 추천해 드립니다.
- 파이썬 챌린지 : 여러 아이비리그 대학들의 CS 입문 과목 용으로 사용하는 파이썬 교재입니다. 내용이 많이 압축 되어 있어서 200장으로 얇은데 핵심적인 내용이 많이 담겨있습니다. 누군가에게 파이썬을 가르치는 용도 혹은 이미 다른 언어를 할 줄 아는 상태에서 파이썬 언어를 처음 공부해야할 때 굉장히 추천합니다.
- 러닝 파이썬 : 파이썬에 대해 정말 세세하게 설명한 서적입니다. 위 서적이나 다른 자료들로 파이썬을 어느 정도 공부한 분들이 파이썬이라는 언어를 좀 더 깊게 공부하는 용도 혹은 레퍼런스 용도로 적합합니다. 비슷한 성격의 유명하고 좋은 서적으로 파이썬 완벽 가이드가 있는데, 러닝 파이썬 서적이 조금 더 최신 내용까지 커버하고 있습니다.
- 전문가를 위한 파이썬 : 파이썬을 좀 더 파이썬답게(Pythonic) 쓸 수 있도록 해주는 전문서입니다. 파이썬 언어에 대한 숙련도가 어느 정도 있으며, 파이썬으로 어느 정도 프로젝트를 진행해본 경험이 있는 분이 파이썬을 전문적으로 쓰기 위해서 공부할 자료입니다. 비슷한 성격의 좋은 서적으로 파이썬 코딩의 기술이 있습니다.

C

```c
#include <stdio.h>

int main() {
    puts("Hello world!");
    return 0;
}
```

C언어는 태생이 UNIX라는 운영체제를 만들기 위하여 태어난 언어입니다. C언어가 세상에 나온 이후 오랜 시간 동안 많은 개발자에게 사랑을 받아왔으며, 지금도 임베디드나 시스템 분야에서 압도적인 점유율을 자랑하고 있습니다. 이후에 생겨난 매우 많은 언어들에 영향을 미쳤으며, 따라서 C언어를 익히고 나면 그로 인해 파생된 많은 언어를 좀 더 쉽게 익힐 수 있습니다. 또한, 굉장히 컴팩트한 문법 체계를 추구하여 언어의 문법 자체는 간결하기 때문에 많은 사람이 C언어를 입문용 언어로 배웠으며, 지금도 많은 대학에서 신입생들에게 첫 언어로 C언어를 가르치고는 합니다. 하지만 이 컴팩트한 문법 때문에, 사실 함정이 매우 많은 언어입니다. C언어로 프로그램을 작성할 때 자칫 잘못하면 내 컴퓨터 환경에서의 동작 결과와 친구 컴퓨터 환경에서의 동작 결과가 달라질 수 있습니다. 초보자 때는 별로 신경 쓰이는 내용이 아닐 수도 있지만, 사실 이런 것은 산업적으로 매우 큰 문제를 야기할 수 있기 때문에, C언어를 배운다면 다른 언어들보다 좀 더 신경 써서 정확하게 배워야 합니다. 이와 관련되어 좋은 포스트가 하나 있기에 링크를 첨부하는데, C언어를 공부할 분이라면 꼭 한 번쯤 읽어보길 권장합니다. C언어로는 주로 디바이스 드라이버, Board Support Package, OS, 미들웨어, 시스템 소프트웨어 등을 만들게 됩니다.

- C Programming : A Modern Approach : C언어를 공부하기 가장 적합한 책입니다. 아쉽지만 번역서가 없기 때문에 원서로 공부해야 합니다. '나는 영어를 잘 못 해서 한국어로 된 책으로 공부하고 싶은데...'라고 생각하시는 분이 있다면 위 선수 지식 영어 파트를 다시 한번 보시기 바랍니다. 정말 아쉽게도, 국내 서적 중에 C언어를 정확하게 설명하고 있는 입문서를 본 적이 없습니다. 따라서 C언어를 공부하고 싶다면 힘들더라도 어느 정도 영어를 익혀서 해당 책으로 공부하기를 권장합니다. (국내 입문서 중 C언어를 정확하게 설명하고 있는 책이 새로 나왔다고 한다면 이슈로 달아주시기 바랍니다. 그리고 나는 대충 공부하고 넘길거니까 그냥 국내 서적으로 볼거야 라는 생각하시는 분이 있다면, 잘못 알고있는 것이 모르는 것보다 위험하다는 것은 인지해두기를 바랍니다.)
- The C Programming Language : C언어의 창시자들이 쓴 책입니다. 보면 알겠지만, 생각보다 굉장히 얇습니다. 하지만 그 안에는 C언어의 초기 설계 철학이 담겨있어서 C언어라는 언어 자체를 좀 더 깊이 이해하기에 적합한 책입니다. 그런데 사실 내용이 좀 구식이기 때문에, 단순 언어 학습을 위해서는 오히려 위의 C Programming 책을 더 추천합니다. 그 외에도 zlib같은 전통 있는 소스 코드를 보다 보면 옛날 스타일의 C언어를 접하게 되는데, 이런 소스 코드를 이해하는데에도 큰 도움을 줍니다.
- C언어 펀더멘탈 : 국내 유일하게 C언어를 정확히 설명한 책이지만, 절대 입문서는 아니고 더군다나 절판입니다. 추천 대상 독자는 '학교 다닐 때 C언어를 국내 서적으로 공부했는데, 어쩌다 보니 C언어로 일할 일이 생겼다.' 하시는 분이 자신이 알고 있는 C언어가 어떻

게 잘못되어있는지 알기 위해 보면 정말 적절합니다. 다만 절판이다 보니 도서관에서 빌려봐야겠지요.

- Linux Kernel : 위에 말한 듯이 C언어는 애초에 운영체제를 만들기 위하여 태어난 언어입니다. 이러한 C언어를 요즘 가장 잘 쓰고 있는 곳 중 하나가 바로 Linux라는 운영체제입니다. Linux 운영체제의 상당한 부분은 C언어로 작성되어 있습니다. 소스 코드를 보다 보면 요즘 스타일의 C언어를 어떻게 쓰고 있는지 확실히 알 수 있습니다. 예를 들어 시스템마다 유동적으로 사용될 구조체에 관련된 자료들과 함께 그 자료들을 제어할 함수 포인터를 두고, 해당 시스템에서 함수 포인터에 제어 함수를 등록하는 방식으로 흔히 객체 지향과 비슷한 방식의 소스 코드를 작성하는 모습도 굉장히 자주 볼 수 있습니다. 링크로는 커널 코드 중 공부용으로 참고하기 좋은 sysfs 소스 코드 부분을 달아두겠습니다.

C++

```C++
#include <iostream>

auto main() -> int {
    std::cout << "Hello world!\n";
    return 0;
}
```

처음에는 C언어에 Simula 언어의 객체지향 개념을 추가하여 만들어진 언어입니다. 그런데 이 언어는 시간이 지나며 너무 많은 기능을 추가하려고 하였고 결국 괴물이 되었습니다. 이제는 절대로 단순히 C언어에 Class를 추가한 언어가 아닌, 각종 패러다임을 집어넣은 만능 지향 언어라고 생각해야 합니다. 특히 C++11이라는 표준안부터는 정말 많은 신개념 패러다임과 기능들이 탑재되어 Modern C++이라고 불립니다. 언어적인 기능이 너무나도 방대하며 언어 자체의 난이도가 상당하기 때문에 입문용 언어로는 추천하지 않습니다만, 이 언어 하나만 정말 잘하면 다른 언어들을 좀 더 쉽게 익힐 수 있습니다. C++은 주로 성능이 중요시되는 프로그램을 만들 때 사용합니다. 영상 처리, 게임 제작, 고성능 서버, 웹 브라우저, GUI 데스크톱 환경 등 정말 다양한 곳에 사용합니다.

- C++ Primer : C++을 처음 공부하는 입문서로 가장 좋은 책이라고 생각합니다. 1000페이지 정도의 두꺼운 책이지만 입문서가 맞습니다. (그만큼 C++의 기본 기능이 방대합니다) 간혹 C++ Primer Plus라는 책과 혼동되기도 하는데, (실제로 해당 서적도 좋은 책인 것은 맞습니다) 이 책이 더욱 C++를 자세히 설명해줄 뿐 아니라 올바르게 사용하는 방법부터 최신 기능을 왜 사용해야 하는지 등 더 추천할만합니다. 특히 저자인 Stanley Lippman은 C++ 창시자인 Bjarne옹의 제자(?)입니다.
- Programming : Principles and Practice Using C++ : C++ 창시자인 Bjarne Stroustrup이 학부생을 가르치기 위하여 쓴 서적입니다. 저자인 비야네옹은 프로그래밍 입문을 위한 서적이라고 말씀하시지만, 솔직히 입문서로 쓰기엔 다소 어렵습니다. 추천하는 대상은 프로그래밍을 조금 공부해본 학부생 1~3학년 정도 수준일 때 프로그래밍 스킬을 조금 더 업그레이드시키는 겸 C++을 공부하려고 하면 이 책이 굉장히 좋습니다. 분명 입문서인데 연습문제가 잘 풀리지 않아도 정상이니 좌절하지 마시기 바랍니다. (해당 책의 연습문제들을 다 풀 수 있다면 프로그래밍 스킬에 굉장히 숙련되어 있을 것입니다.)

- Effective C++ : 구형 C++을 효율적으로 쓰기 위한 필독서입니다. Effective 시리즈의 시조 같은 책인데, 'C++라는 언어를 이렇게는 쓰면 안 되고 이런 식으로 써야 한다'는 것을 조금이나마 깨닫게 해주는 책입니다. 추천하는 대상은 C++ 언어로 일을 해야 하거나 C++ 언어를 잘하고 싶은 모든 분입니다. (사실 이 책 이후에 더욱더 깊게 공부하는 More Effective C++이라는 책도 있지만, Modern의 시대가 열려서 MEC++은 Optional 합니다.)
- Effective Modern C++ : Modern C++을 사용할 분들이 필수적으로 봐야 하는 필독서입니다. C++로 일을 하거나 C++ 언어를 잘 쓰고 싶다는 분들은 예외 없이 봐야 할 책입니다.
- The C++ Programming Language : C++ 창시자인 비야네옹이 쓴 책으로, 레퍼런스 서적입니다. 사실 C++는 아래 사이트에 레퍼런스가 잘 되어있기 때문에 해당 책은 좀 Optional하긴 하지만, C++ 언어에 어느 정도 익숙해 졌을 때 이 언어의 철학을 느끼며 내용을 정리하려는 분들이 공부하기 좋은 책입니다.
- cppreference : C++ 레퍼런스 사이트입니다. 예를 들어 C++의 std::vector에 무슨 기능이 있고 어떻게 사용해야 하는지 찾아야 한다면 다음 링크를 참조하면 됩니다.
- C++로 만든 프로그램으로는 구글 크롬을 포함한 수많은 구글 어플리케이션, 비트코인, 리그 오브 레전드, QT, KDE, IE/Office/그림판 등 마이크로소프트 어플리케이션 등이 있습니다.

Java

```java
public class Main {
        public static void main(String[] args) {
                System.out.println("Hello world!");
        }
}
```

Java는 대표적인 객체 지향 프로그래밍 언어입니다. JVM이라는 가상 머신 위에서 돌아가기 때문에 크로스 플랫폼 언어라고도 합니다. 이 언어의 가장 주요한 특징으로는 한국에서 압도적으로 많이 사용하는 언어라는 것입니다. 한국에서 개발자로 취업하기 위해서는 몇 분야를 제외하고는 거의 필수적으로 잘 알아야 하는 언어입니다. (특히 Spring Framework와 함께 잘 알아둬야 합니다) 자바는 주로 웹 서비스를 개발하거나 안드로이드 애플리케이션을 개발할 때 사용합니다.

- Java의 정석 : 한국에서 전통적인 Java 입문서입니다. 비슷한 성격의 책으로는 이것이 자바다나 자바의 신이 있습니다. Java의 정석 책이 주변에서는 평이 가장 좋긴 한데, 서점이나 도서관에서 대충 훑어보며 셋 중 가장 마음에 드는 책으로 골라도 크게 상관은 없습니다. 어차피 기본서의 한계가 있기 때문에 문법을 확실히 익히고 난 후에 다른 자료들로 더욱 공부할 필요가 있습니다.
- Effective Java : 유명한 Effective 시리즈 중 하나로 Java를 더욱 효율적으로 사용하기 위한 전문서입니다. Java 또한 C++11 이후에 큰 변화가 있던 것처럼 7버전 이후에 점점 변화가 생겨왔습니다. 해당 서적에는 이런 변화까지 포함하여 Java를 어떻게 해야 잘 쓸 수 있는지 가이드해줍니다. 사실상 Java로 일을 하기 위한 필독서입니다.

- 토비의 스프링 : 사실 Java라는 언어가 아닌 Spring Framework에 관련된 서적입니다. 하지만 사실 한국에서 Java로 일을 하게 되면 대부분 해당 프레임워크를 사용할 것이기에 함께 공부해두면 좋습니다. 국내 Spring 서적 중 가장 유명하고 자세한 서적이 아닌가 싶습니다만 신간이 3.1 구버전임을 고려해야 합니다.

Javascript

```
console.log("Hello world!");
```

포탈의 메인 화면에서 광고 배너가 움직이고, SNS에서 스크롤을 할 때마다 새로운 글이 계속해서 나타나는 모습을 보신 적 있으신가요? Javascript는 그러한 동적인 웹 페이지를 개발하기 위해 만들어진 프로그래밍 언어입니다.
처음엔 웹 브라우저 위에서만 작동했었으나, Node.js (Javascript 런타임 환경)가 등장하며 환경의 제약에서 벗어나게 되었습니다. 그로 인해 서버, 데스크톱 앱, 모바일 앱 등 다양한 분야에서 사용되고 있습니다.
자바와는 이름과 구문의 유사성만 있을 뿐, 직접적인 관련이 없는 별개의 언어입니다.
(Mocha -> LiveScript -> Javascript 로 여러번의 개명이 있었습니다. 이유가 궁금하시다면 여기를 참고하세요.)

웹 페이지에서 작동하는 Javascript에 대해 더 알고 싶으시면 여기를 참고하세요.
Node.js에 대해 더 알고 싶으시면 여기를 참고하세요.

- 생활코딩! HTML+CSS+자바스크립트 : 생활코딩으로 유명한 이고잉님의 웹 프론트 입문서입니다. 프로그래밍을 처음 시작하는 분 중에 웹 개발로 시작해보고 싶은 분에게 굉장히 추천합니다.
- 인사이드 자바스크립트 : 자바스크립트 동작 원리를 좀 더 자세하게 설명해주는 책입니다. 책이 굉장히 얇음에도 불구하고 자바스크립트 핵심 개념들을 명확하게 설명해줍니다. 자바스크립트를 공부한 적이 있는 분이 좀 더 자세한 언어 동작 원리를 공부하려고 할 때 추천합니다.
- 자바스크립트 완벽 가이드 : 자바스크립트의 거의 모든 것을 설명해 둔 레퍼런스 책입니다. 자바스크립트 코뿔소 책이라는 명칭으로 유명합니다. 자바스크립트라는 언어의 기능과 동작 원리를 정말 상세하게 공부하고 싶은 분에게 추천합니다.
- 자바스크립트는 왜 그 모양일까? : 자바스크립트의 대가 더글러스 크락포드의 저서입니다. 저자의 다른 저서로 자바스크립트 핵심 가이드라는 좋은 책이 있는데, 절판입니다. 해당 서적은 자바스크립트를 어떻게 해야 잘 쓸 수 있는지 가이드해주는 책으로 자바스크립트로 개발하려고 하는 분들은 필수적으로 봐야 할 서적입니다. 위의 코뿔소 책은 Optional이라면 해당 책은 자바스크립트로 개발하려는 분들에게 필수입니다.
- 자바스크립트는 모든 곳에 존재한다 : 최근 사용처가 늘어난 자바스크립트를 어떻게 다양하게 쓸 수 있는지 가이드해주는 책입니다. 자바스크립트로 개발을 하려는 모든 분들에게 굉장히 추천하는 책입니다.
- 모던 자바스크립트 입문 : 자바스크립트의 동작 원리를 이해하고 사용하는데 도움을 주는 책입니다. 자바스크립트를 깊이 이해하고 싶다면 이 책을 추천합니다.

C#

```
using System;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello world!");
        }
    }
}
```

C#은 마이크로소프트에서 개발한 객체지향 언어입니다. C 계열의 언어는 맞지만, 오히려 Java 언어와 비슷합니다. 여러 언어의 장점을 흡수해 언어적인 완성도가 우수하며, 지속해서 업데이트 되는 새로운 기능 또한 MS DOCS에서 쉽게 확인할 수 있습니다. 태생이 .NET이라는 프레임워크의 한 부분으로 만들어진 언어여서 .NET 플랫폼에 크게 의존하지만, Unity 등의 스크립트 언어로도 사용합니다. 따라서, C#은 주로 .NET 프레임워크를 이용하여 윈도우 애플리케이션을 개발하거나, Unity에서 게임을 개발할 때 사용합니다. 물론, Mono와 .NET Core 덕분에 범용성이 많이 증가하여 다양한 곳에서 사용할 수는 있습니다.

- 이것이 C#이다 : 최신 버전인 C# 9.0을 반영한 입문서입니다. 비슷한 성격의 서적으로는 시작하세요! C# 9.0 프로그래밍이 있으니 서점 등에서 비교하여 마음에 드는 책으로 골라 공부해도 좋습니다. C#을 처음 공부하려는 분들에게 추천합니다.
- C# 6.0 완벽 가이드 : C# 언어를 정말 상세하게 설명한 서적으로 언어의 세부적인 기능을 자세하게 공부할 수 있습니다. 6.0 버전까지만 커버하지만, 해당 내용으로도 언어의 깊은 내용을 알 수 있습니다. 언어의 기능을 상세하게 공부하고 싶거나 레퍼런스로 사용하려는 분들에게 추천합니다.
- 이펙티브 C# : 이번에도 유명한 Effective 시리즈 중 하나로 C#을 더욱 효율적으로 쓸 수 있게 해주는 전문서입니다. 위의 완벽 가이드 책은 Optional 하지만, 해당 책은 C#으로 개발을 하려는 분들에게 필수입니다. C#을 이미 어느 정도 공부한 적이 있는 분들이 이 언어로 일을 하기 위해 언어 숙련도를 높이려고 할 때 추천합니다.

etc.

- Go : Google에서 만든 프로그래밍 언어로, 언어 설계자인 Thompson, Pike, Griesemer가 말하길 C++의 복잡성이 싫어 해당 언어를 만들게 되었다고 합니다. Garbage Collection 기능이 있고 병행성을 지원합니다. Go 언어를 사용하는 프로젝트로는 도커와 쿠버네티스, Cloudflare, MongoDB, Netflix, Twitch 등이 있습니다. 클라우드 계열에서 많이 사용하는 언어입니다. 공부할만한 서적으로는 Head First Go와 Go in Action, Tucker의 Go 언어 프로그래밍이 있습니다.
- Rust : Mozilla에서 만든 프로그래밍 언어로, 뛰어난 언어 완성도를 자랑합니다. (컴파일러 전공하는 지인들이 굉장히 극찬하는 언어입니다.) Performance, Reliability,

Productivity를 슬로건으로 내세우고 객체 지향 프로그래밍, 병렬 프로그래밍, 함수형 프로그래밍, 명령형(Imperative) 프로그래밍 패러다임을 지원합니다. 특이한 점으로는 변수의 Lifetime과 Ownership을 컴파일 타임에 추적하여 생성과 소멸 시기를 결정합니다. 즉, Java나 C#같이 Garbage Collection으로 인한 런타임 코스트 없이 안전하게 메모리를 관리합니다. 대신 언어 난이도 자체가 상당한 편이므로 해당 언어를 사용하여 개발을 진행하기 전에는 단단한 각오가 필요할 것입니다. Rust를 사용하고 있는 프로젝트에는 디스코드, 파이어폭스 등이 있습니다. 공부할 서적으로는 러스트 프로그래밍 공식 가이드가 있습니다.

- Swift : Apple사의 플랫폼(macOS, iOS, watchOS, tvOS)을 위한 프로그래밍 언어입니다.
- Kotlin : JetBrains 에서 개발한 JVM 기반의 언어입니다. 2017년에 안드로이드 공식 언어로 채택되었습니다. 장황한 문법을 지닌 Java에 비해 비교적 간결한 문법을 제공합니다.

Fundamental Technique

컴퓨터공학/컴퓨터과학의 뼈대를 이루는 기술들이 있습니다. 흔히 컴퓨터 학과 학부 과정의 전공과목인 알고리즘, 컴퓨터구조, 운영체제 등으로 이를 기반 기술이라고 하겠습니다. 사실 요새는 개발 프레임워크 / 개발 플랫폼 등이 너무나도 많이 발전하고 하드웨어 및 최적화 기술의 연혁나 향상에 힘입어 과장한 추상화된 컴퓨터들을 가지고 간편하게 개발할 수 있게

README.md

것만 신기술에 좀 더 관심을 두는 경향이 보입니다. 그런 말이 틀렸단 것은 아니지만, 기본의 중요성을 간과하지 말았으면 합니다. 후술할 공부 방식 중 Top-Down 방식의 학습을 한다면 기본적인 내용을 뛰어 넘어가며 필요할 때 필요한 내용을 공부하게 될 텐데, 그렇다 하더라도 최종적으로 기본들에 대한 지식이 없다면 속이 빈 강정이 되어 속히 말하는 야매 개발자가 될 확률이 높습니다. 그리고 사실 최신 기술들도 하늘에서 뚝 떨어진 것이 아니라 이런 기반 기술들로부터 영향을 받아 만들어지고 발전한 것입니다. 알고리즘 공부를 열심히 하다가 인공지능을 공부하면 똑같은 내용이 그대로 등장하며, 컴파일러를 공부하고 자연어 처리를 보면 친숙한 내용이 반갑게 맞아줍니다. 운영체제를 공부하며 숨 쉬듯이 보던 컨셉들을 응용 프로그램 개발에서 그대로 볼 수 있습니다. 그러니 최신 기술들에 현혹되어 컴퓨터학과에 입학하신 분 중 이러한 과목들을 공부하며 재미가 없다고 실망하시는 분이 있다면 결국 이러한 내용이 기반이 되는 것들이므로 생각을 다잡으시기 바랍니다.

Algorithm

알고리즘은 어떠한 문제를 해결하기 위한 절차의 표현으로 Computer Science의 큰 기초가 되는 과목 중 하나입니다. 효율적인 프로그래밍의 근간이 되어주고, AI와 같은 다양한 분야의 근본적인 기술입니다. 특히나 MS와 Google로 시작한 Coding Interview(코딩 테스트)가 국내에도 크게 유행하고 있기에 입사를 위해서 신경 써 공부해야 할 과목입니다. 그런데 학술적인 연구를 위한 알고리즘, 실무에서 사용하는 알고리즘, PS(Problem Solving) 대회를 위한 알고리즘, 코딩 테스트를 위한 알고리즘 등 분야별로 다루는 내용이 크게 달라지기 때문에 목적에 맞춰서 효율적인 공부가 필요합니다. 실무에서 사용하는 알고리즘은 어떤 회사에서 어떤 제품을 개발하는지에 따라 천차만별이기에 제외하고 아래에는 학술, 코딩 테스트, 대회를 위한 공부 리소스들을 설명하겠습니다. 사실 코딩 테스트와 대회를 위한 알고리즘은 공부하는 내용이 굉장히 유사하지만, 대회를 위한 공부가 훨씬 넓은 범위를 깊게 파고들어야 한다는 차이가 있습니다.

Academic

- Foundations of Algorithms : 아래 책과 함께 학부 알고리즘 수업에 가장 많이 사용되는 바이블입니다. 사실 알고리즘 공부를 한다고 하면 대부분 학술적인 내용보다는 코딩 테스트나 대회를 위한 알고리즘 공부를 할 테지만, 순수하게 알고리즘을 공부해보고 싶은 분들에게는 해당 서적을 추천합니다.
- Introduction to Algorithms : CLRS로도 불리는 이 책은 오랜 세월 동안 많은 대학의 교재로 사용되어 왔습니다. 알고리즘의 바이블이지만 개인적으로는 독학용으로는 위의 Foundations of Algorithms를 더 추천해 드리며, 해당 서적은 레퍼런스로 사용하는 것을 추천해 드립니다.
- The Art of Computer Programming : Donald Knuth 교수가 집필 중인 서적으로, 알고리즘의 끝판왕이라고 할 수 있습니다. 해당 서적을 다 이해하기만 하더라도 빌 게이츠가 마이크로소프트에 특채하겠다고 할 정도로 극악무도한 난이도로 유명합니다. 컴퓨터과학의 근간은 수학이라는 것을 여실히 보여주는 것처럼 굉장히 수학적인 내용으로 구성되어 있습니다. 솔직히 말해서 순수한 학습용은 아닌 것 같습니다.. (저는 보는 것을 포기하여 책장 장식용으로 사용하고 있습니다) 알고리즘을 전공할 분들에게 추천해 드려봅니다.
- Project Euler : 유명한 PS 플랫폼으로 여타 다른 플랫폼과는 다르게 수학 문제를 해결하는 사이트입니다. 한국어 번역 사이트도 있지만 몇 기능과 문제들이 넘어오지 않았습니다. 수학 문제들을 프로그래밍을 통해 해결해보고 싶은 분들에게 추천합니다.

Coding Interview (Coding Test)

- 파이썬 알고리즘 인터뷰 : 코딩 테스트를 위한 알고리즘 공부 입문으로 굉장히 좋은 서적입니다. 비슷한 성격의 서적으로는 이것이 취업을 위한 코딩 테스트다 with 파이썬이 있습니다. 알고리즘 공부를 전혀 해보지 않은 상태에서 인터넷에서 유명한 백준이나 프로그래머스 문제 풀이부터 하려고 하면 굉장히 막막할 것입니다. 이러한 서적을 한번 보면서 어떤 자료구조와 알고리즘들을 어떤 식으로 사용하는지 형식을 잡고 들어가면 굉장히 효율적으로 공부할 수 있습니다.
- 프로그래머스 : 국내에서 많이 사용하는 코딩 테스트 플랫폼 사이트입니다. 많은 회사들이 해당 플랫폼을 사용하여 코딩 테스트를 진행하기 때문에 미리 익숙해지면 굉장히 좋

습니다. 질 좋은 문제들과 카카오 기출 문제들을 풀어볼 수 있으며 Level 3, 4 정도의 문제를 풀 수 있으면 코딩 테스트 준비가 잘 되었다고 생각할 수 있습니다. 유일한 단점으로는 문제 수가 좀 적습니다.

- LeetCode : 유명한 해외 PS 플랫폼입니다. 양질의 문제가 매우 많으며 구글, 아마존, 페이스북, 넷플릭스 등 해외 기업들의 기출 문제들도 풀 수 있습니다. 위의 파이썬 알고리즘 인터뷰 서적 또한 해당 플랫폼에 있는 문제들을 풀이하며 설명합니다. 흔히 코딩 테스트 양치기를 하기에 가장 좋은 플랫폼 중 하나입니다.

Competition

- 알고리즘 문제 해결 전략 세트 : 종만북으로 유명한 이 책은 알고리즘 대회를 준비하는 분들에게 필독서가 아닌가 싶습니다. 저자의 노하우가 책에 잘 녹아있고 대회에 필요한 알고리즘들을 잘 설명해줍니다. 하지만 굉장히 어려운 난이도로 많은 사람들에게 좌절을 안겨준 서적이기도 합니다. 단순히 입사를 위한 코딩 테스트 준비로 이 서적을 보는 것은 솔직히 크게 추천드리지는 않습니다. 알고리즘 대회를 준비하는 분들이나 자신의 PS 실력을 향상하고 싶은 분이 단단한 뼈대를 만들고 싶을 때 공부하는 것을 추천합니다.
- 백준 온라인 저지 : 국내에서 가장 유명한 PS 플랫폼입니다. 문제 양이 정말 굉장히 많으며 지원하는 프로그래밍 언어도 매우 많습니다. 국내외 알고리즘 대회 기출 문제나 삼성 코딩 테스트 기출 문제도 풀 수 있습니다. 대회를 준비하시는 분들에게는 거의 필수적인 플랫폼입니다. (사실 대회 준비하시는 분이 해당 사이트를 모를 수가 없긴 합니다.)
- CodeForces : 이 또한 세계적으로 유명한 PS 플랫폼인데, 코딩 테스트보다는 대회 준비에 좀 더 적합한 사이트입니다. 백준 사이트와 연동이 가능하며 국제 대회를 준비하는 경우 필수적으로 이용하게 되는 사이트입니다. 전반적인 문제 난이도가 높습니다.
- 알고스팟 : 알고리즘 문제 해결 전략 서적이 해당 사이트의 문제들을 풀이하며 설명합니다. 전반적인 문제 난이도가 굉장히 높습니다. 튜토리얼을 보면 알겠지만 초보자용 문제도 꽤나 난이도가 있는 편입니다.

Computer Architecture

- Computer Organization and Design : 많은 대학에서 사용하는 컴퓨터 구조의 바이블입니다. RISC-V 버전도 나와 있습니다.
- 프로그래머가 몰랐던 멀티코어 CPU 이야기 : 컴퓨터 구조 이론들을 굉장히 쉽게 풀어 써준 서적입니다. 현대 컴퓨터 구조를 이해하는데 필요한 내용을 이해하기 쉽게 설명해준 좋은 서적인데 정말 아쉽게도 절판입니다. 개인적으로 컴퓨터구조를 공부한 적이 없는 채로 프로그래밍을 하는 모든 분이 읽어봤으면 하는 서적입니다. 도서관에서 빌려서라도 볼 수 있다면 좋겠습니다.
- 프로세서를 지탱하는 기술 : 위의 멀티코어 CPU 이야기와 약간 비슷한 성격의 서적인데 좀 더 프로세서의 근간이 되는 내용을 설명해줍니다. 마찬가지로 굉장히 좋은 서적이지만 아쉽게도 절판입니다. 혹시 도서관에서 빌려볼 수 있다면 한번 읽어보기를 추천합니다.

System Programming

- UNIX 고급 프로그래밍 : APUE라고 불리는 이 책은 시스템 프로그래밍 전통의 바이블입니다. 전반적인 UNIX-like 환경에서의 시스템 콜 프로그래밍과 주의사항 등을 설명합니다. 시스템 개발을 공부하려는 분들에게 사실상 필독서입니다.
- 리눅스 API의 모든 것 : 원제 The Linux Programming Interface인 이 책은 위의 APUE와 비슷하지만 좀 더 리눅스 환경에 집중되어있고, 내용이 신식입니다. 비록 고 Stevens의 APUE가 공신력으로는 가장 알아주지만, 해당 서적도 리눅스 매뉴얼 페이지를 작성한 Kerrisk의 저서로 그에 못지 않은 편입니다. APUE도 그렇지만 몇 구현 연습문제 난이도가 꽤 상당합니다. 리눅스 환경에서의 시스템 프로그래밍을 공부하고 싶은데 APUE가 싫다면 해당 서적을 추천합니다.
- Advanced! 리눅스 시스템 네트워크 프로그래밍 : 굉장히 좋은 시스템 프로그래밍 실무 지침서로 저자분의 노하우가 잘 들어가 있습니다. 개념서로 활용해도 좋지만, 위의 APUE나 TLPI 서적으로 개념들을 익히고 해당 서적으로 이러한 개념들을 어떻게 실무 개발에 적용할지 익히는 게 가장 좋을 거 같습니다. 개인적으로, APUE와 해당 서적을 한 번씩 정독한 후 공부한 내용을 활용하여 다중 클라이언트에 동시 서비스되는 웹 프록시 서버를 하나 만들 수 있다면 시스템 프로그래밍에 어느 정도 익숙해졌다고 생각해도 될 것 같습니다.

Design Pattern

- Head First Design Patterns : 유명한 디자인 패턴 입문서입니다. 사실 개인적으로 Head First 시리즈를 선호하는 편은 아니지만, 디자인 패턴만큼은 패턴들을 이해하고 익히기 좋은 예제로 잘 구성되어 있어서 예외로 두고 있습니다. 아래 GoF 서적이 디자인 패턴의 바이블이긴 하지만, 내용이 딱딱하고 일부 이해하기 쉽지 않아서 디자인 패턴을 처음 공부할 시에는 해당 서적으로 공부하는 것을 추천합니다. 참고로 디자인 패턴을 적용할 일이 많은 java로 작성되어 있습니다.
- GoF의 디자인 패턴 : 디자인 패턴의 바이블입니다. 참고로 GoF는 Gang of Four이라고 불리는 Gamma, Helm, Johnson, Vlissides를 줄인 것입니다. 디자인 패턴을 처음 공부하시는 분들보다는, 디자인 패턴을 한 번 정도 공부한 적이 있고 이제 실제 코드에 적용하려고 할 때 레퍼런스가 필요한 경우 해당 서적을 참고하는 것을 추천해 드립니다. 참고로 C++ 언어를 기반으로 작성되어 있습니다.

Operating System

- Operating System Concepts : 소위 공룡책으로 유명한 OS 개념 바이블 서적으로 많은 대학들에서 수업 교재로 사용하고 있습니다. 안타깝게도 최신 운영체제들의 기술들을 반영하고 있지는 못하지만, 근본적인 개념을 익히기엔 여전히 가장 좋은 책 중 하나입니다. (사실 본격적인 운영체제 공부를 처음 하는 독자들을 대상으로 최신 기술들을 가르치는 것도 너무 복잡하기에 무리가 있습니다.) 굉장히 역사가 깊은 책이라 여러 판이나 버전이 있는데, 에센셜 버전은 역서도 나와있습니다. 해당 서적은 최신 운영체제 기술을 공부하는 목적이 아닌 운영체제 개념을 처음 공부하는 분들에게 추천하며, 8판 이후 아무 버전 혹은 에센셜 버전으로 봐도 무방합니다.
- 64비트 멀티코어 OS 원리와 구조 : OS를 부트로더부터 GUI까지 전부 만들어보는 서적입니다. OS를 만들어보는 컨텐츠를 여럿 봤지만 해당 서적이 가장 범용적이고 내용 구성이

좋았습니다. OS를 전문적으로 공부해보고 싶은 분에게 굉장히 추천드리는 책입니다. 참고로 내용이 정말 어렵습니다. 간신히 따라가다가도 APIC 같은 챕터에서 막힐 수밖에 없습니다. 그래도 포기하지 않고 내용을 다 이해하신다면 시스템에 대한 이해도가 크게 향상될 것입니다.

Network

- TCP/IP가 보이는 그림책 : 네트워크라는 분야를 아예 처음 공부하는 사람이 보기에 좋은 책입니다. 그림책이라는 이름 답게 정말 기초적인 내용을 쉽게 설명해주고 있습니다. 깊이 있는 학습을 할 수는 없지만, 네트워크에에서 사용하는 기본적인 개념 지식들을 탑재하여 이 후 공부를 수월하게 할 수 있습니다.
- 후니의 쉽게 쓴 시스코 네트워킹 : 한국에서 정말 전통적으로 쓰인 네트워크 기본 서적입니다. 저 또한 해당 서적으로 네트워크 분야에 입문했는데 당시에 초록 책으로 유명하던 서적이 파란 색으로 리뉴얼 되어서 조금 놀랐습니다. 사실 개발자분들보다는 네트워크 엔지니어분들에게 필요한 내용들을 학습할 수 있는 서적이긴 하지만, 범용적인 네트워크 기본 지식을 공부하고 싶은 분들에게 추천합니다.
- IT 엔지니어를 위한 네트워크 입문 : 개발 공부를 하다가 네트워크에 대한 전문적인 지식이 필요해졌을 때 보기 좋은 서적입니다. 혹은 위의 후니 책을 보고 복습 겸 좀 더 전문적인 내용까지 다루고 싶다면 해당 서적을 읽으면 좋습니다. 사실 이 글을 보는 대부분의 분들은 네트워크 엔지니어보다는 개발자를 목표로 할 것으로 예상되는데, 이러한 경우엔 위 후니 책보다 해당 서적을 좀 더 추천드립니다.
- TCP/IP Illustrated : 고 Stevens가 쓰신 TCP/IP 네트워크의 바이블입니다. 총 3권으로 구성되어 있는데, 특히 1권은 네트워크 관련 개발을 하려는 모든 분들이 꼭 봐야할 필수 서적입니다. 참고로 번역된 적이 있긴 한데, 번역의 퀄리티가 정말 좋지 않아서 꼭 원서로 봐야합니다.

Web

사실 잡부인 필자가 시스템 개발, 네트워크 개발, 보안 솔루션 개발, BSP 개발 등의 경력은 있지만 웹개발을 전문적으로 한 적이 없기 때문에 자료의 공신력이나 신빙성을 위하여 시니어 개발자 분들의 Pull Request를 기다리겠습니다.

- HTTP 완벽 가이드
- 모던 웹 애플리케이션 개발

Database

- Real MySQL : 데이터베이스를 학습함에 있어서 DBMS와 SQL 학습은 필수불가결합니다. 해당 서적은 세계에서 가장 많이 쓰이는 RDBMS인 MySQL을 국내에서 가장 상세하게 설명한 책입니다. 다만 굉장히 상세하고 실전적이라 처음 데이터베이스를 공부하는 분에게는 조금 버거울 수 있습니다. 그런 분들은 모두의 SQL같은 입문서를 한번 빠르게 보거나 인터넷에 있는 쿼리문 예제들을 실습해보며 쿼리문을 조금 익히고 해당 서적을 보는 것을 추천합니다.

- 파워 오브 데이터베이스 : 세계적으로 유명한 데이터 베이스 설계 지침서인 '가장 쉬운 데이터베이스 설계'의 개정판 입니다. 이 서적은 특정한 DBMS나 SQL에 종속된 내용이 아닌 데이터베이스를 어떻게 설계하고 이용해야 하는 지에 대한 가이드를 잡아줍니다. 즉, 데이터베이스를 공부해본 적이 있어서 DBMS를 사용할 줄 아는 분이 데이터베이스를 어떻게 설계해야하는 지에 대하여 학습할 때 굉장히 추천드립니다.
- 데이터베이스 인터널스 : 읽어보지 못한 서적이지만 초고수분들이 해당 서적으로 스터디 하는 것을 목격하여 추가해봅니다. 읽어보신 분들의 PR을 기다리겠습니다.

Professional Technique

Android Application

- Android Developers Guide : 세상 모든 안드로이드 앱 개발자 분들이 참고할 수 밖에 없는 레퍼런스 사이트이자 학습 사이트입니다. 안드로이드의 특성 상 업데이트가 굉장히 빠르기 때문에 출간되는 서적이 최신 내용을 따라잡을 수가 없습니다. 따라서 안드로이드를 한번이라도 공부한 적이 있다면 실시간으로 업데이트되는 해당 사이트를 이용하여 학습하는 것이 좋습니다. 다만, 안드로이드 앱 개발이 아예 처음이신 분은 아무래도 레퍼런스식 가이드가 접근하기 어려울 수 있습니다. 그런 분들은 서점에 있는 안드로이드 신간들 중 마음에 드는 것을 골라 한번 따라한 후에 해당 사이트를 참조하는 것이 좋습니다. 글 쓰는 시점을 기준으로는 안드로이드 프로그래밍, Do it! 안드로이드 앱 프로그래밍, 이것이 안드로이드다 with 코틀린, Step by Step 안드로이드 프로그래밍과 같은 서적들이 나와있습니다.
- 실무에 바로 적용하는 안드로이드 프로그래밍 : 위와 같은 사정에도 불구하고 참고하면 좋은 안드로이드 서적들이 몇권 있는데, 그 중 하나입니다. 해당 서적은 제목과 같이 실무에서 사용할만한 토픽들을 중점적으로 다룹니다. 안드로이드를 공부해본적 있는 분이 실제 플레이스토어에 배포할만한 앱을 만들고 싶을 때 굉장히 유용합니다.
- 프로페셔널 안드로이드 : 구글 안드로이드 프레임워크 팀원이 작성한 서적으로 안드로이드 SDK를 좀 더 상세하게 공부하고 싶은 분들이 꼭 봐야할 서적입니다. 안드로이드를 공부해본 경험이 있는 분이 위의 서적과 함께 참고하면 매우 좋습니다.

iOS Application

안타깝게도 필자가 iOS쪽은 개발 경험이 없어 시니어 개발자 분들의 PR을 기다리겠습니다.

- Do it! 스위프트로 아이폰 앱 만들기 입문
- Apple Developer

Game

안타깝게도 필자가 게임 개발 경력이 없기에 시니어 개발자 분들의 PR을 기다리겠습니다.

- Real-Time Rendering : 실시간 렌더링 분야에서 가장 유명한 책입니다. 이 책을 통해 렌더링 기본을 읽히시고 쉐이더나 언리얼 실무서를 보시면 더욱 이해가 빠를 것입니다.

Embedded System

- 사물인터넷을 위한 리눅스 프로그래밍 with 라즈베리 파이 : 임베디드쪽 공부를 처음 시작한다면 아마 높은 확률로 라즈베리파이를 가지고 공부하게 될 것입니다. 국내에 좋은 라즈베리파이 서적이 많은데, 그 중에서도 특히 해당 서적이 굉장히 풍부한 내용을 담고 있습니다. 라즈베리파이 기초부터 리눅스 기초 프로그래밍과 좀 더 응용한 주제들까지 다루고 있는데, 본인의 수준에 맞게 골라볼 수도 있게 구성되어 있습니다. 임베디드 분야를 처음 공부하시는 분에게 굉장히 추천드립니다.
- Raspberry Pi User Guide : Programming the Raspberry Pi와 함께 세계적으로 유명한 서적 중 하나입니다. 두 서적 모두 굉장히 얇기 때문에 편하게 볼 수 있습니다. 라즈베리파이로 프로젝트를 진행하게 되어 라즈베리파이 자체를 공부하고 싶다면 해당 서적들을 추천하지만, 그 외의 경우엔 굳이 그럴 필요는 없다고 생각합니다.
- 임베디드 엔지니어 교과서 : 임베디드 시스템의 전반적인 내용을 간략하게 소개해줍니다. 임베디드 소프트웨어부터 RTOS의 특성이나 실무에서 사용하게 되는 임베디드 리눅스인 yocto까지 어느정도 소개해줍니다. 위의 사물인터넷을 위한 리눅스 프로그래밍 서적으로 입문을 마쳤다면, 해당 서적을 통해 임베디드 시스템의 개괄을 학습하시길 추천드립니다.
- 임베디드 OS 개발 프로젝트 : KLDP에서 유명한 나빌레라님의 저서로, 임베디드 시스템에서 동작할만한 RTOS를 직접 만들어보는 서적입니다. OS 제작이라고 하더라도 책이 굉장히 얇고 내용이 깊지는 않습니다. 임베디드 시스템에 대한 이해도를 높이기에 좋고, 가볍게 볼 수 있어서 굉장히 추천드립니다.
- 임베디드 RTOS 입문서 세트 : 비단 RTOS 뿐만 아니라 임베디드 시스템을 다루면서 필요한 OS의 개념들을 예제 소스코드를 통해 굉장히 잘 설명해줍니다. 다만, 20년이 된 서적이므로 내용이 굉장히 구식입니다. (윈도우 2000에서 볼란드C를 사용해 만들었으니 말 다 했습니다.) 그래도 핵심 개념 자체는 지금도 어느정도 통용되는 내용이라 임베디드 시스템에 대하여 좀 더 깊게 알고싶다면 해당 서적을 공부할만한 가치가 있습니다.
- Yocto 프로젝트를 활용한 임베디드 리눅스 개발 : 실제로 임베디드 시스템을 구축하는 경우 주로 buildroot나 yocto project를 사용하게 됩니다. 해당 서적은 그 중 yocto project의 구성 요소들과 구축 방법에 대하여 설명해줍니다. TV에 들어가는 LG webOS나 BMW, 벤츠 등에 들어가는 IVI 등 많은 임베디드 시스템이 yocto project를 통해 구성됩니다. 사실 그걸 넘어서 임베디드 업계의 가장 큰 손인 Qualcomm을 포함한 거의 모든 SoC 업체들이 yocto project를 기반으로 BSP를 배포합니다. 실무에 투입되기 전에 조금이라도 실무 내용을 미리 접하고 싶다면 yocto project에 대한 학습을 권장합니다.

System Engineering

- 컨테이너 인프라 환경 구축을 위한 쿠버네티스/도커 : 믿을 수 있는 저자님의 책이 출간되어서 우선 추가 하였습니다. 아직 책을 보진 못했기에 리뷰나 설명을 남기기는 어려운데, 읽어보신 분이 계시다면 PR 부탁드립니다.
- 쿠버네티스 인 액션 : 아무래도 요즘 인프라 구축에 컨테이너는 빠질 수 없는 존재가 되었습니다. 가장 대표적이고 널리 쓰이는 컨테이너는 모두가 잘 알다시피 도커이며, 쿠버네티스는 이를 강력하게 관리해줍니다. 해당 서적은 쿠버네티스에 대한 자세한 설명을 넘어 실제로 마주칠 수 있는 다양한 상황에서의 대응 기술들을 알려줍니다. 입문서로도

사용할 수 있지만, 내용이 어렵다면 아리수님의 쿠버네티스 입문 같은 서적을 먼저 보면 좋습니다.

- 쿠버네티스 모범 사례 : 책의 설명대로 쿠버네티스의 창시자들이 쓴 쿠버네티스 적용 가이드입니다. 책이 생각보다 얇기에 가볍게 볼 수 있으며, 내용 또한 챕터 별로 독립적이라 주 관심 대상을 선별하여 읽을 수도 있습니다.
- 아마존 웹 서비스 AWS Discovery Book : 마찬가지로 요즘 인프라 구축에 클라우드 서비스는 반 필수적인 요소가 되었습니다. AWS, Azure, GCP 등의 서비스들이 있지만 아무래도 국내에서는 여전히 AWS를 가장 많이 사용합니다. 해당 서적은 클라우드 컴퓨팅에 대한 기본 개념부터 AWS 이론과 실전까지 잘 설명하여 입문용으로 평이 굉장히 좋습니다.
- 따라하며 배우는 AWS 네트워크 입문 : 해당 서적은 AWS 네트워크에 초점을 맞춰 설명한 서적입니다. 아무래도 AWS를 사용하다 보면 네트워크 관련 내용들을 가장 어려워하기 마련입니다. 이 서적은 그런 AWS 네트워킹에 대한 개념 설명과 함께 실무 상황에서 마주칠 수 있는 상황에 대한 실습을 통해 AWS 네트워킹에 대한 이해를 도와줍니다. 다만 책 이름에 입문이 들어가 있다고 하여 AWS 입문 서적은 절대 아니고, AWS 네트워크에 대한 입문이니 참고 바랍니다.
- DevOps와 SE를 위한 리눅스 커널 이야기 : 시스템 엔지니어를 위한 리눅스 커널 서적입니다. 커널 서적이라고 해서 커널 소스코드를 분석하는 것은 아니고, 시스템 엔지니어링에 필요한 커널 개념들을 소개하며 이를 기반으로 커널 파라미터 등을 시스템에 맞게 어떻게 설정할 것인지 가이드해줍니다. 또한, 시스템 엔지니어링 시 유용한 명령어들도 설명해줍니다. Greg옹의 리눅스 커널 IN A NUTSHELL과 병행하면 시너지 효과를 기대할 수 있을거 같습니다. (물론 Greg옹의 서적은 커널 버전 2.6을 기준으로 한 구식 버전이긴 합니다.)
- 카프카, 데이터 플랫폼의 최강자 : 카프카는 분산 스트리밍 플랫폼 중 메시지 브로커 시스템으로 주로 데이터 파이프 라인을 만들 때 사용합니다. 대규모 인프라 구축 시 자주 사용되며, 카카오 리소스 모니터링 구성도를 보아도 핵심적으로 사용되는 것을 볼 수 있습니다. 사실 이러한 인프라 구축을 단순히 독학으로 익히기는 어렵지만, 이런 인프라 구축에 들어가는 주요 시스템을 한 번 맛보는 것에 의의를 두고 학습하면 좋을것 같습니다.
- 엔터프라이즈 데이터 플랫폼 구축 : 위와 같은 인프라 구축에 관련된 서적입니다. 데이터베이스 파트에서 언급한 데이터베이스 인터널즈와 마찬가지로 읽어보지 못한 서적이지만 고수분들이 해당 서적으로 공부하던 것을 보고 추가합니다. 마찬가지로 읽어보신 분들의 PR을 기다리겠습니다.

Software Engineering

- 리팩토링 : 리팩토링의 뜻은 위키피디아에 따르면 '결과의 변경 없이 코드의 구조를 재조정함'으로, 쉽게 말하면 코드를 이해하기 쉽고 관리하기 쉽도록 고치는 것입니다. 여기서 중요한건 기능을 추가하거나 버그를 잡는 행위가 아닌 유지보수를 위해 코드의 구조를 고친다는 것입니다. 이렇게 코드를 보기 좋게 고치는 과정에서 까딱 잘못하면 예상치 못한 버그가 새로이 생기게 되고, 그러면 다시 디버깅을 하다가 또 코드가 더러워지는 일이 생길 수 있습니다. 해당 서적은 이러한 리팩토링 과정을 체계적으로 수행하여 이러한 실수를 최소화하는 동시에 좀 더 코드를 깔끔하게 구성할 수 있도록 가이드해줍니다. '일단

기능 구현이나 제대로 하고 나중에 리팩토링 해야지'라고 말하고서는, 정작 나중에 거대해진 코드를 어떻게 리팩토링해야 할 지 모르겠는 분들에게 굉장히 추천합니다.

- 클린 코드 + 클린 아키텍처 : 주로 협업을 하는 프로젝트에서 어떻게 코드를 유지보수하기 좋게 작성할지 가이드해주는 서적입니다. 유지보수뿐만 아니라 대형 프로젝트에 참여할 때 주의해야할 점들을 일깨워줍니다. 사실 이런 내용들은 정작 실제 업무에 치이다 보면 마음 속의 우선순위가 내려가 신경쓰기 어렵긴 하지만 이런 내용을 알고 모르고에 따라 코드 결과물을 보면 퀄리티 차이가 나기 마련입니다. 어느 정도 원하는 기능을 구현할 수 있는 수준에서 코드의 아키텍처를 개선하고 싶은 분들에게 추천합니다.
- 실용주의 프로그래머 : 개발자가 참고할만한 칼럼 모음집으로 기술서보다는 교양서의 느낌입니다. 학부 4학년 ~ 신입 정도의 시기에 교양 느낌으로 시간 날 때 한번 읽어보길 권장합니다.
- 조엘 온 소프트웨어 : 조엘 아저씨의 자서전 같은 느낌으로, 소프트웨어를 관리 혹은 개발하는데 필요한 이야기들을 엮은 서적입니다. 좀 오래되었고, 조엘 아저씨가 마이크로소프트 등 해외에서 일할 때의 경험을 기반으로 쓴 내용이기에 사실 국내 사정과는 잘 맞지 않기는 합니다. 그래도 읽어보면 입담이 굉장히 재밌고 기술적인 내용도 아니라서 교양서로 읽어보면 좋습니다. 물론 프로젝트 매니징 하는데 있어 도움도 됩니다.

Security

컴퓨터 학과에 입학하신 분들 중에 보안/해킹 쪽 진로를 생각하며 오신 분들도 적지 않을 겁니다. 일반적으로 보안 분야는 공부하는 내용이나 방법이 미묘하게 달라집니다. 예를 들어 (정말 간단하게 설명하자면) C언어의 함수를 공부할 때만 해도 개발자라면 함수는 언제 사용하며, 어떻게 만드는 것이 좋고, 프로토타입은 어떻게 정의해야 효율적인지 등을 중점적으로 공부할 겁니다. 그런데 보안쪽에 진로를 잡고 있다면 그보다 함수 스택 프레임은 어떻게 구성되는지, 호출자와 피호출자간에 정보 전달은 어떤 식으로 이루어지는지, 콜링 컨벤션에 따라 메모리가 어떻게 변화하는지 등에 초점을 맞추게 됩니다.
그 외에도, 디자인패턴이나 개발방법론 같은 내용은 사실 별 도움이 되지 않을 수 있습니다. 소스 코드 오디팅하는 데에는 조금이나마 도움이 될 수도 있겠지만, 그 시간에 다른 공부를 하는 것이 훨씬 이득입니다. (물론 보안 솔루션을 개발하는 쪽으로 간다면 예외입니다.)
하지만 보안쪽을 공부한다고 해도 CS 기본기는 매우 도움이 됩니다. 포너블을 하는데 있어 시스템 프로그래밍이나 운영체제에 대한 지식은 기본입니다. 마찬가지로 웹해킹을 하는데 기본적인 웹이나 데이터베이스에 대한 이해가 없으면 진행하기가 어렵습니다. 따라서 기반기술 내용만큼은 잘 공부해두시면 도움이 됩니다.
아래에는 보안쪽 공부를 하는데 있어 도움이 될만한 자료나 사이트들을 몇 가지 소개해드리겠습니다. 사실 필자는 와우해커나 해커즈랩이 주력이던 때에 달고나 문서 등으로 공부했던 지라 (소위 말하는 틀인지라..) 최신 트렌드를 잘 모르기 때문에 고수분들이 PR을 주시면 굉장히 감사드리겠습니다.

- 드림핵
- 해커스쿨
- 라젠카 테크노트
- 루비야 블로그
- 웹해킹KR

- 포너블KR

Study

Roadmap

로드맵은 직접 작성하기에는 너무 케이스가 많고 양이 많다보니 이미 시니어 개발자분들이 만들어주신 로드맵 중 개인적으로 추천하는 로드맵의 링크들을 걸어두겠습니다. 만약 추후에 시간이 된다면 시스템 개발 분야 로드맵은 직접 작성도 해보도록 하겠습니다.

- 웹 개발자 로드맵
- 안드로이드 앱 개발자 로드맵
- iOS 앱 개발자 로드맵
- 게임 개발자 로드맵

Projects

여러가지 프로젝트나 대외 활동 등에 대하여 설명하는 챕터입니다.

개인 프로젝트

대외 활동

- 차세대 보안리더 양성 프로그램 : 한국정보기술연구원(KITRI)에서 진행하는 정보보안 기술과 산업의 미래를 이끌어 나갈 차세대 리더를 양성하는 프로그램입니다. 총 3단계의 교육과 취약점 분석, 디지털 포렌식, 보안 컨설팅, 보안제품개발 총 4개의 트랙으로 구성되며, 정보보안 분야 최고 전문가로 구성된 멘토단이 이끌고 있습니다. 전공 교육, 팀 프로젝트 교육을 거쳐 우수 교육생을 대상으로 3단계 경연 교육을 통한 최종 BEST10 선발 및 인증으로 진행됩니다. 교육생 지원사항으로 학습공간, 프로젝트 활동 멘토링 및 비용 지원, 기숙사 지원 등이 있습니다. 기수 별로 운영되며 매년 5월 경에 모집하고 있습니다.
- SW 마에스트로 : 창의도전형 SW인재 육성으로 SW산업의 미래를 선도하는 정부지원 사업입니다. 예비 연수과정, 창의도전형 과정을 거쳐 우수한 연수생(상위 10%)에게는 글로벌 SW역량 교육 프로그램을 지원합니다. 연수생 지원 사항으로 IT기기 구입비 지원(1회), 교육과정 중 (6개월) 매월 100만원 지급, 멘토링, 프로젝트 개발비 등이 있습니다.
- Microsoft Learn Student Ambassadors : Microsoft에서 운영하는 학생 기술 리더 네트워크 프로그램입니다. 전 세계 학생들과 함께 온라인으로 코딩 기술을 학습하는 데 도움을 주고, 가상 해커톤을 주최하여 실제 문제를 해결하고, 디지털 커뮤니티를 구축합니다. 전 세계에서 16세 이상, 대학 등의 교육기관에서 공부하는 학생을 대상으로 하며 매년 3월, 9월 경에 모집합니다. 활동 수준에 따라 알파, 베타, 골드의 3가지 단계로 구성되어 있으며 알파 이상의 스튜던트 앰베서더에게는 150 USD 월간 Azure 크레딧, Visual Studio Enterprise 구독권, MTC 인증 시험 바우처, studentambassadors.com 도메인, Camtasia 등의 소프트웨어 지원 등의 혜택이 있습니다. 베타 이상의 앰베서더에겐 행사에 대한 지

원, swag 물품 지급, 멘토십, 해외 써밋 이용이 가능합니다. 골드 앰베서더의 경우, 컨퍼런트 발표 및 참여가 가능하고 파일럿 프로그램 참여, Microsoft의 MVP 멘토쉽과 함께 MVP 프로그램에 초대될 수 있습니다.

- Github Campus Experts : 깃헙에서 운영하는 캠퍼스 기술 커뮤니티 리더 프로그램입니다. 기술을 배우고 경험을 공유하고 프로젝트를 함께 구축하는 활동을 합니다. 2월과 9월경에 모집하며 지원 제출 후 합격하면 5분 분량의 지원 동기, 커뮤니티의 비전과 목표에 대한 영상을 제출해야 합니다. 온오프라인 컨퍼런스와 모임 및 해커톤을 선도하며 깃헙에서 연설 스피킹, 테크니컬 라이팅, 커뮤니티 리더십, 소프트웨어 개발 등의 교육과 지원을 받게 됩니다.
- Google Summer of Code

해커톤

대회

- ACM ICPC
- Google Codejam
- Facebook HackerCup
- Kakao Code Festival
- SCPC

Comments

기타 프로그래밍 공부를 하며 알아두면 좋은 조언들을 적는 챕터입니다.

Community

커뮤니티 리스트 혹은 이용법 등에 관한 챕터입니다.

개발 관련 커뮤니티 리스트

- Stack Overflow
- Quora
- Github Community
- KLDP
- OKKY
- 생활코딩

커뮤니티 이용법

- 질문하는 방법

Q&A

ISSUE에 올라온 질문 등에 대해 대답하는 챕터입니다.

조언

각 분야의 시니어 개발자 분들이 주니어 개발자 혹은 뉴비분들에게 드리고 싶은 조언란입니다.
시니어 개발자님들의 PR 감사히 받겠습니다.

ex)

> 한국에서 취업하려면 자바, 스프링, AWS 공부하고 코테 준비 잘 하세요 - 익명

> 이곳은 고인물 파티야, 도망치세요 - 익명

> 프로그래밍은 단거리 경주가 아닌 마라톤입니다 - 익명

Web

System

Embedded

Network

Game

Releases

No releases published

Packages

No packages published