한 줄 한 줄 짜면서 익히는

러스트 프로그래밍

Rust in Action
by Tim Mcnamara

Original English language edition published by Manning Publications, USA.

한 줄 한 줄 짜면서 익히는 러스트 프로그래밍

전자책 1쇄 발행 2022년 7월 21일 **지은이** 팀 맥나마라 **옮긴이** 장연호 **펴낸이** 한기성 **펴낸곳** (주)도서출판인사이트 **편집** 정수진 **등록번호** 제2002-000049호 **등록일자** 2002년 2월 19일 **주소** 서울특별시 마포구 연남로5길 19-5 **전화** 02-322-5143 **팩스** 02-3143-5579 **블로그** http://blog.insightbook.co.kr **이메일** insight@insightbook.co.kr **ISBN** 978-89-6626-366-0

프로그래밍 인사이트

한 줄 한 줄 짜면서 익히는

러스트 프로그래밍

RUST IN ACTION

팀 맥나마라 지음 | 장연호 옮김

인사이트

차례

옮긴이의 글

시대라고 하면 거창하지만, 컴퓨터 엔지니어들은 당시의 문제를 해결하는 데 다양한 해법을 제시해 왔다. 소프트웨어의 여명기에 소프트웨어 위기라는 문제를 객체지향 프로그래밍으로 헤쳐 나왔고, 웹, 모바일이라는 새로운 플랫폼에서는 그에 맞는 언어와 프레임워크로 폭발적인 수요를 감당했다.

그렇다면 이 시대의 당면 과제는 무엇일까? 너무나 많은 문제가 산재해 있지만 물리적 한계에 근접한 칩, 점점 심각하게 받아들여지고 있는 소프트웨어 결함 문제, 개발자 집단에서 증가하는 다양성을 꼽을 수 있겠다.

코어의 속도 향상은 정체기에 접어들었다. 코어 속도가 매년 기하급수적으로 향상되는 것은 더 이상 기대할 수 없다. 멀티 코어를 상정한 개발 없이 성능 향상을 논하기가 어려워졌다.

사용자 부주의나 잘못된 코딩으로 인한 보안 결함은 갈수록 산업에 큰 위협이 되고 있다. 산업의 디지털 전환이 급격하게 이루어지면서 소프트웨어의 수요는 그 어느 때보다 높아졌고 책임은 그에 비례해 무거워지고 있지만, 기존의 도구로는 그 기대를 맞추기 어렵다.

우리 사회가 성별, 문화, 인종 등에 대한 다양성을 수용해 나감에 따라 관련한 개발자 커뮤니티에도 변화가 일어나고 있다. 제법 오랫동안 개발자의 이미지는 몸집이 크고 어딘가 괴짜스러운 남성이라는 고정 관념이 형성되어 있었다. 수년 동안 몇몇 개발 공동체는 고정 관념과 변화된 사회 관념이 충돌하는, 과도기적 갈등이 빚어졌다. 몇몇 컨퍼런스에서는 여성 참가자에 대한 성적 공격이나 비하가 일어나기도 했고, 커밍아웃한 성소수자에 대해 지극히 편향된 시각을 바탕으로 한 인신공격이 일어나기도 했다. 이러한 상황이 완전히 해결되지는 않았지만, 몇몇 커뮤니티에서는 행동수칙(Code of Conduct)이라는 규정을 두어 좀 더 포용적인 방향을 지향하고 있다. 이는 사회가 갖추어야 할 포용성을 커뮤니티 차원에서도 노력을 통해 수용해 나가야 한다는 점을 시사한다.

하나의 언어가 이런 모든 문제의 해답을 가지고 있지는 않다. 다만 러스트는 바

른 방향을 고민하고 건설적인 논의를 통해 다양한 문제를 모범적으로 해결해 나가려는 사람들의 모임을 토대로 발전해 나가고 있다.

사람이 만든 모든 것에는 당시의 문제에 대한 고민과 이를 해결하려는 노력이 고스란히 반영되어 있다. 바로 이 점에서 러스트를 배운다는 것은 단순히 하나의 언어를 배우는 것 이상의 의미가 있으리라 본다.

이 책은 수많은 사람들의 땀과 노고가 있어 나올 수 있었다. 한 권의 책에 많은 내용을 충실히 담고자 노력했던 저자의 노력에 경의를 표한다. 과감하게 러스트를 선택해 주셨고, 번역 과정을 통해 더 깊은 지식을 알 수 있는 기회를 주신 인사이트의 한기성 대표님께 감사 드린다. 편집/교정 과정에서 애써 주신 송우일, 정수진 편집자님과 인사이트의 모든 분에게 감사한다.

그리고 책을 번역하는 내내 격려와 응원을 준 아내 미정과 딸 은채, 은유에게 고마움을 전한다.

머리말

기술 서적을 읽는 데 노력을 들일 가치가 있는지 없는지는 누구도 모른다. 비싸고 따분하고 형편없이 집필된 책일 수도 있다. 설상가상으로 아무것도 배우지 못할 가능성도 있다. 다행히도 이 책은 그런 사실을 이해하는 사람이 썼다.

이 책의 첫 번째 목표는 러스트를 가르치는 것이다. 이 책은 학습을 돕기 위해 규모가 있고 실무에 도움이 되는 프로젝트를 제공한다. 책을 읽으면서 데이터베이스, CPU 에뮬레이터, 운영 체제 커널, 기타 여러 가지 흥미로운 프로젝트를 작성하게 될 것이다. 생성 예술(generative art)에도 손을 댄다. 각 프로젝트는 각자에게 맞는 속도로 러스트 프로그래밍 언어를 탐색할 수 있도록 설계되었다. 러스트를 거의 모르는 독자들이라도 어떤 방향을 선택하든 프로젝트를 확장할 수 있는 많은 기회가 주어질 것이다.

그런데 프로그래밍 언어 학습은 단지 문법과 의미를 공부하는 데서 그치지 않는다. 여러분은 커뮤니티에도 참여하게 된다. 불행히도 일정하게 자리잡은 커뮤니티는 지식, 용어, 관행을 공유하기 때문에 신규 진입자에게 보이지 않는 장벽을 만들 수 있다.

많은 새내기 러스트 프로그래머가 부딪히는 장벽 중 하나가 시스템 프로그래밍 개념이다. 해당 분야에 대한 배경 지식 없이 러스트에 입문하는 프로그래머가 많다. 이를 보완하기 위해 이 책은 두 번째 목표로 시스템 프로그래밍을 가르친다. 그리고 이 책의 12개 장에서 메모리, 디지털 시간 기록, 장치 드라이버 작동 방식에 대해 배울 것이다. 이를 통해 러스트 커뮤니티의 구성원이 될 때 더 편안하게 느낄 수 있기를 바란다. 그리고 우리는 여러분이 필요하다!

우리 사회는 소프트웨어에 의존하고 있는데 심각한 보안 허점이 일상적이고 어쩌면 불가피한 것으로 받아들여지고 있다. 러스트는 그렇지 않다는 것을 보여 준다. 게다가 우리의 컴퓨터는 필요 이상의 에너지를 소모하는 비대한 애플리케이션으로 가득 차 있다. 러스트는 이러한 유한한 자원을 덜 요구하는 소프트웨어를 개발하는 실행 가능한 대안을 제공한다.

이 책은 개발자에게 능력을 더해 주는 책이다. 이 책의 궁극적인 목적은 여러분에게 그 사실을 확신시키는 것이다. 러스트는 선별된 전문가 그룹을 위한 전유물이 아니다. 누구나 사용할 수 있는 도구다. 다른 학습 과정을 거쳐 여기까지 오느라 수고했다. 여러분을 이제 몇 걸음 더 나아가게 할 수 있게 되어 기쁘다.

감사의 말

내가 무너지는 것을 막아 주고 넘어졌을 때 나를 일으켜 준 Katie에게 감사한다. 아빠가 글을 쓰느라 놀아 주지 못했을 때도 포옹과 미소를 보내 준 Florence와 Octavia에게도 감사한다.

너무나 많은 사람에게 빚을 지고 있어서 일부만 열거하는 것이 불공평하다고 생각한다. 이 책 집필을 지원해 준 러스트 커뮤니티의 많은 구성원이 있다. 수천 명의 독자가 책이 집필되는 동안 라이브북(liveBook)을 통해 수정 사항, 질문, 제안 사항을 제출했다. 모든 기여는 내가 텍스트를 다듬는 데 도움이 되었다. 감사드린다.

친구가 된 몇몇 독자에게 특히 감사드린다. Aï Maiga, Ana Hobden, Andrew Meredith, Andréy Lesnikov, Andy Grove, Arturo J. Pérez, Bruce Mitchener, Cecile Tonglet, Daniel Carosone, Eric Ridge, Esteban Kuber, Florian Gilcher, Ian Battersby, Jane Lusby, Javier Viola, Jonathan Turn, Lachezar Lechev, Luciano Mammino, Luke Jones, Natalie Bloomfield, Oleksandr Kaleniuk, Olivia Ifrim, Paul Faria, Paul J. Symonds, Philipp Gniewosz, Rod Elias, Stephen Oates, Steve Klabnik, Tannr Allard, Thomas Lockney, William Brown. 지난 4년간 여러분과 교류한 것은 특별한 영광이었다.

책의 검토자들인 Afshin Mehrabani, Alastair Smith, Bryce Darling, Christoffer Fink, Christopher Haupt, Damian Esteban, Federico Hernandez, Geert Van Laethem, Jeff Lim, Johan Liseborn, Josh Cohen, Konark Modi, Marc Cooper, Morgan Nelson, Ramnivas Laddad, Riccardo Moschetti, Sanket Naik, Sumant Tambe, Tim van Deurzen, Tom Barber, Wade Johnson, William Brown, William Wheeler, Yves Dorfsman에게 감사를 전한다. 여러분의 의견을 모두 읽었다. 책 집필 후반 단계에서 적용한 많은 개선 사항은 여러분의 사려 깊은 피드백 덕분이었다.

매닝의 두 팀원이 보여 준 인내, 전문성, 적극성은 칭찬받아 마땅하다. Elesha Hyde와 Frances Buran은 수많은 초안을 거쳐 가며 이 책의 집필을 인도해 주었다.

Bert Bates, Jerry Kuch, Mihaela Batinić, Rebecca Rinehart, René van den Berg,

Tim van Deurzen을 포함한 나머지 편집자에게도 감사드린다. Benjamin Berg, Deirdre Hiam, Jennifer Houle, Paul Wells를 비롯한 편집자에게도 감사를 전한다.

이 책은 MEAP(Manning early access program) 과정에서 16번 릴리스했는데 많은 사람의 지원 없이는 불가능했을 것이다. Aleksandar Dragosavljević, Ana Romac, Eleonor Gardner, Ivan Martinović, Lori Weidert, Marko Rajkovic, Matko Hrvatin, Mehmed Pasic, Melissa Ice, Mihaela Batinic, Owen Roberts, Radmila Ercegovac, Rejhana Markanovic에게 감사드린다.

Branko Latincic, Candace Gillhoolley, Cody Tankersley, Lucas Weber, Stjepan Jureković를 포함한 마케팅 팀원에게도 감사드린다. 여러분은 엄청난 격려를 내게 보내 주었다.

매닝의 다른 팀도 매우 신속하게 대응하고 도움을 주었다. 책을 만드는 동안 도움을 준 Aira Dučić, Andrew Waldron, Barbara Mirecki, Branko Latincic, Breckyn Ely, Christopher Kaufmann, Dennis Dalinnik, Erin Twohey, Ian Hough, Josip Maras, Julia Quinn, Lana Klasic, Linda Kotlyarsky, Lori Kehrwald, Melody Dolab에게 감사드린다. 그리고 인생을 바꾸는 이 모든 과정을 시작하도록 해 준 Mike Stephens에게 감사드린다. 어려운 일이 될 것이라고 내게 경고했는데 그 말이 옳았다.

책 소개

이 책은 주로 온라인에서 러스트를 설명한 무료 자료를 찾아본 다음에 '이제 뭘 보지?"라고 스스로에게 질문한 사람들을 대상으로 한다. 이 책에는 수십 가지 흥미로운 예제가 들어 있고, 창의력과 시간이 허락된다면 이 예제들을 확장할 수 있을 것이다. 이러한 예를 통해 러스트의 생산적인 부분과 생태계에서 중요한 많은 서드파티 라이브러리를 다룰 작정이다.

코드 예제는 우아하고 관용적인 러스트 방식보다는 초보자를 위한 접근성을 강조했다. 어느 정도 러스트에 익숙한 프로그래머라면 예제의 일부 스타일 결정에 동의하지 않을 수 있다. 이 점은 배우는 사람들을 위함이니 양해 바란다.

이 책은 포괄적인 참고용 교과서가 아니다. 언어와 표준 라이브러리의 일부 내용이 빠져 있다. 빠진 부분들은 고도로 전문화되어 특별히 다뤄야 하는 것들이 대부분이다. 대신 이 책은 독자들에게 필요한 경우 전문적인 주제를 배울 수 있는 충분한 기본 지식과 자신감을 제공하는 것을 목표로 한다. 이 책은 거의 모든 예제가 마이크로소프트 윈도우에서 작동하기 때문에 시스템 프로그래밍 책의 관점에서도 독특하다고 할 수 있다.

누가 이 책을 읽어야 하는가

러스트에 관심이 있거나 현실적인 예를 적용하여 배우고 싶거나 러스트가 시스템 프로그래밍 언어라서 꺼려지는 사람이라도 이 책을 즐길 수 있다. 이미 프로그래밍 경험이 있는 독자는 몇 가지 컴퓨터 프로그래밍 개념이 필요한 부분에서 가장 큰 혜택을 얻을 수 있다.

이 책의 구성: 로드맵

이 책은 2부로 구성되어 있다. 1부에서는 러스트 문법과 몇 가지 독특한 특징을 소개한다. 2부에서는 1부에서 얻은 지식을 여러 프로젝트에 적용한다. 각 장에서는 새로운 러스트 개념을 한두 가지 소개한다. 즉, 1부는 러스트에 입문하는 속성 과정이다.

- 1장 '러스트 소개'에서는 왜 러스트가 나왔는지, 그리고 러스트로 프로그래밍을 시작하는 방법을 설명한다.
- 2장 '러스트 언어의 기초'는 러스트 문법을 구체적으로 설명한다. 예제는 망델브로 집합 렌더러와 grep 클론이다.
- 3장 '복합 데이터 타입'에서는 러스트 데이터 타입과 오류 처리 기능을 구성하는 방법을 설명한다.
- 4장 '수명, 소유권, 대여'에서는 데이터 접근이 항상 유효한지 확인하는 메커니즘에 대해 설명한다.

2부에서는 러스트를 입문 수준 시스템 프로그램 영역에 적용한다.

- 5장 '데이터 심화'에서는 숫자의 근삿값을 어떻게 계산하는지 중점적으로 살펴보면서 디지털 컴퓨터에서 정보를 표현하는 방법을 다룬다. 예제로는 맞춤형 숫자 형식과 CPU 에뮬레이터가 있다.
- 6장 '메모리'에서는 참조, 포인터, 가상 메모리, 스택, 힙이라는 용어를 설명한다. 예제로는 메모리 스캐너와 생성 예술 프로젝트가 있다.
- 7장 '파일과 저장소'에서는 데이터 구조를 저장 장치에 저장하는 프로세스를 설명한다. 예제는 십육진 덤프 프로그램과 실제 동작하는 데이터베이스다.
- 8장 '네트워킹'에서는 추상화 계층을 제거해 나가면서 HTTP를 여러 번 다시 구현하는 과정을 통해 컴퓨터가 통신하는 방법을 설명한다.
- 9장 '시간과 시간 관리'에서는 디지털 컴퓨터 내에서 시간을 추적하는 프로세스를 알아본다. 예제로는 NTP 클라이언트를 만들어 본다.
- 10장 '프로세스, 스레드, 컨테이너'에서는 프로세스, 스레드와 관련 추상화에 대해 설명한다. 터틀 그래픽 애플리케이션과 병렬 분석기 예제로 살펴본다.
- 11장 '커널'에서는 운영 체제의 역할과 컴퓨터 부팅 방법에 대해 알아본다. 자체 부트로더와 운영 체제 커널을 컴파일하는 예제로 살펴본다.
- 12장 '시그널, 인터럽트, 예외'에서는 외부 세계가 CPU, 운영 체제와 통신하는 방법을 설명한다.

이 책은 순서대로 읽도록 되어 있다. 이전 장에서 배운 지식을 알고 있다고 가정하고 이후 장이 진행된다. 그러나 각 장의 프로젝트는 독립적으로 실행할 수 있다. 그러니 다루고 싶은 주제가 있다면 자유롭게 앞뒤로 오가며 봐도 괜찮다.

코드에 대해

이 책의 코드 예제는 러스트 2018 에디션으로 작성되었으며 윈도우와 우분투 리눅스에서 테스트했다. 러스트를 설치하는 것 외에는 다른 특별한 소프트웨어가 필요하지 않다. 설치 방법은 2장에 나와 있다.

이 책에는 소스 코드 행 번호를 매기는 방식과 일반 본문 사이사이에 끼워 넣는 방식으로 예제 소스 코드를 담았다. 두 경우 다 소스 코드는 **고정폭 글꼴**로 서식이 지정되어 일반 텍스트와 구분된다. 기존 코드에 새 기능이 추가되는 경우에는 코드를 **굵게 표시**하여 같은 장의 이전 단계에서 변경된 코드를 강조하여 표시한다.

많은 경우 원본 소스 코드를 재정리했다. 책에서 사용 가능한 지면에 맞추기 위해 줄 바꿈을 추가하고 들여쓰기를 했다. 드물지만 이것으로도 충분하지 않으면 줄 연속 표시를 넣었다. 또한 코드가 본문에 설명되어 있는 경우에는 소스 코드 주석을 제거했다. 중요한 개념을 강조해야 할 때는 코드에 주해를 제공했다.

다른 온라인 자료들

이 책의 지은이인 팀은 소셜 미디어에서 @timClicks로 찾을 수 있다. 주요 채널은 트위터(*https://twitter.com/timclicks*), 유튜브(*https://youtube.com/c/timclicks*), 트위치(*https://twitch.tv/timclicks*)다. *https://discord.gg/vZBX2bDa7W*에서 그가 운영하는 디스코드 서버에 가입할 수도 있다.

책 표지 그림에 대해

이 책의 표지에 있는 그림은 'Le maitre de chausson' 또는 'The boxer'라고 한다. 삽화는 여러 예술가의 작업을 루이 퀴메아(Louis Curmer)가 편집해 1841년 파리에서 발간한 작품집에서 가져온 것이다. 작품집 제목은 《LesFrançais peints par eux-mêmes》이며, 번역하면 '프랑스 사람들이 직접 그린 자신들의 모습'이라는 뜻이다. 각 삽화는 손으로 섬세하게 그리고 채색했는데 작품집의 풍부한 그림을 보면 불과 200년 전에 세계적으로 서로 다른 지역이나 마을에 사는 주민들이 문화적으로 얼마나 분리되었는지 생생하게 떠올릴 수 있다. 사람들은 서로 고립되어 서로 다른 방언과 언어를 사용했다. 거리나 시골 풍경으로 그들이 어디에 살았는지 그리고 옷차림으로 그들의 직업이나 삶의 위치가 어땠는지 쉽게 식별할 수 있다.

이후로 드레스 코드가 바뀌었고 그 당시 풍부했던 지역별 다양성은 사라졌다. 이제 다른 도시나 지역은 고사하고 다른 대륙의 주민들을 구별하기도 어렵다. 아마도

우리는 문화적 다양성을 더 다양한 개인 생활, 특히 더 다양하고 빠르게 변화하는 기술 생활과 맞바꾸었을 것이다.

컴퓨터 책들을 서로 구별하기 어려운 시기에 매닝은 2세기 전 지역 생활의 풍부한 다양성을 바탕으로 한 책 표지로 컴퓨터 비즈니스의 독창성과 주도성을 기리며 이 작품집의 그림을 통해 이를 되살리고자 한다.

1장

Rust in Action

러스트 소개

이 장에서 배울 내용

- 러스트의 특징과 목표
- 러스트 문법 살펴보기
- 러스트를 사용할 경우와 사용하지 말아야 할 경우
- 첫 번째 러스트 프로그램 만들기
- 객체 지향 언어 및 그 외 언어와 러스트 비교 설명

개발 능력을 더해 주는 언어, 러스트 세상에 온 것을 환영한다. 한번 그 속으로 파고들어 가 보면 러스트가 비할 데 없는 속도와 안전성을 가졌으며 사용하기에도 즐거운 언어임을 알 수 있을 것이다.

러스트로 프로그래밍을 시작한다면 이후에도 계속 쓰고 싶을 것이다. 그리고 이 책은 러스트 프로그래머로서 여러분의 자신감을 길러 줄 것이다. 다만 프로그래밍을 처음부터 가르치지는 않는다. 이 책은 러스트를 다음번에 사용할 언어로 고려하는 사람들과 실용적인 예제를 구현하는 것을 즐기는 사람들을 대상으로 썼다. 이 책에서 다루는 몇 가지 주요한 예제는 다음과 같다.

- 망델브로 집합 도식화
- grep 프로그램과 똑같은 복제본
- CPU 에뮬레이터
- 생성 예술
- 데이터베이스

- HTTP, NTP 그리고 십육진 덤프 프로그램
- 로고(LOGO) 언어 인터프리터
- 운영 체제 커널

이 목록에서 알 수 있듯이 이 책을 읽으면 러스트뿐 아니라 더 많은 것을 익힐 수 있다. 시스템 프로그래밍과 저수준 프로그래밍도 소개한다. 이 책을 공부해 나가면서 운영 체제의 역할, CPU 동작 방식, 컴퓨터가 시간을 유지하는 방법, 포인터의 개념, 데이터 타입의 개념 등을 배울 것이다. 컴퓨터의 내부 시스템이 상호 작용하는 방식도 이해하게 될 것이다. 문법 이외에도 러스트가 왜 탄생했는지 그리고 어떤 문제를 해결해 왔는지도 보게 될 것이다.

1.1 러스트는 어디에 사용되는가?

러스트는 스택오버플로에서 해마다 실시하는 개발자 대상 설문에서 2016~2021년에 '가장 사랑받는 프로그래밍 언어'로 선정되었다. 아마도 그 덕분에 다음과 같은 대형 기술 선도 업체에서 러스트를 도입했을 것이다.

- AWS(Amazon Web Services)는 2017년부터 서버리스 컴퓨팅 서비스인 AWS 람다(Lambda)와 AWS 파게이트(Fargate)에 러스트를 사용해 왔다. 이후 러스트는 아마존에서 더욱 널리 쓰이게 되었다. 아마존은 러스트로 EC2(Elastic Compute Cloud) 서비스를 위한 보틀로켓(Bottlerocket) 운영 체제와 AWS 니트로(Nitro) 시스템을 만들었다.[1]
- 클라우드플레어는 공개 DNS, 서버리스 컴퓨팅, 패킷 검사 등 많은 서비스를 러스트로 개발했다.[2]
- 드롭박스는 엑사바이트 저장 장치를 관리하는 백엔드를 러스트로 다시 구축했다.[3]
- 구글은 안드로이드 블루투스 모듈을 개발하는 데 러스트를 사용했다. 러스트는 크롬(Chrome) OS의 `crosvm` 컴포넌트에도 사용되었으며 구글의 새로운 운영 체제인 퓨셔(Fuchsia)를 개발하는 데 주요 언어로 채택됐다.[4]

1 "How our AWS Rust team will contribute to Rust's future successes," *http://mng.bz/BR4J*
2 "Rust at Cloudflare," *https://news.ycombinator.com/item?id=17077358*
3 "The Epic Story of Dropbox's Exodus From the Amazon Cloud Empire," *http://mng.bz/d45Q*
4 "Google joins the Rust Foundation," *http://mng.bz/ryOX*

- 페이스북은 러스트를 사용하여 웹, 모바일, API 서비스뿐 아니라 핵(Hack) 프로그래밍 언어에서 사용하는 HHVM(HipHop virtual machine)의 일부분을 개발했다.[5]
- 마이크로소프트는 사물 인터넷 서비스용 보안 데몬을 비롯한 애저 플랫폼 컴포넌트를 러스트로 개발했다.[6]
- 모질라는 코드가 1500만여 줄에 달하는 파이어폭스 웹 브라우저를 개선하는 데 러스트를 사용했다. 모질라가 파이어폭스에 러스트를 처음 적용한 프로젝트는 MP4 메타데이터 분석기와 텍스트 인코더/디코더였으며, 전반적인 성능과 안전성 향상을 이루었다.
- 깃허브 자회사인 npm은 '일일 13억 개 이상의 패키지'를 사용자에게 제공하는 데 러스트를 이용했다.[7]
- 오라클은 고(Go) 참조 구현으로 인한 문제를 해결하기 위해 컨테이너 런타임을 러스트로 개발했다.[8]
- 스마트싱스(SmartThings)라는 삼성 계열사에서는 사물 인터넷 서비스를 위한 펌웨어 백엔드인 '허브(Hub)'에 러스트를 사용했다.

또한 러스트는 빠르게 움직여야 하는 스타트업에서 제품과 서비스를 배포하기에 충분히 생산적이다. 다음은 그 예다.

- 소스그래프(Sourcegraph)는 지원하는 모든 언어에 대한 문법 강조(syntax highlight)에 러스트를 이용한다.[9]
- 피그마는 다중 사용자 서버의 핵심 성능을 좌우하는 컴포넌트에 러스트를 사용했다.[10]
- 패러티(Parity)는 이더리움 블록체인 프로그램을 러스트로 개발했다.[11]

5 HHVM 4.20.0 and 4.20.1," *https://hhvm.com/blog/2019/08/27/hhvm-4.20.0.html*
6 *https://github.com/Azure/iotedge/tree/master/edgelet*
7 "Rust Case Study: Community makes Rust an easy choice for npm," *http://mng.bz/xm9B*
8 "Building a Container Runtime in Rust," *http://mng.bz/d40Q*
9 "HTTP code syntax highlighting server written in Rust," *https://github.com/sourcegraph/syntect_server*
10 "Rust in Production at Figma," *https://www.figma.com/blog/rust-in-production-at-figma/*
11 "The fast, light, and robust EVM and WASM client," *https://github.com/paritytech/parity-ethereum*

1.2 러스트를 실무에서 추천하기

직장에서 러스트를 추천한다면 어떨까? 처음 난관을 극복하면 잘되는 경향이 있다. 다음에 인용한 2017년 토론은 좋은 일화다. 구글의 크롬 OS 팀 개발자 중 한 명이 프로젝트에 러스트를 도입하는 과정에 대해 이야기한 것이다.[12]

2017년 9월 27일 indy 씀:

러스트가 구글에서 사용 승인을 받을 수 있을까요?

> **2017년 9월 27일 zaxcellent 씀:**
>
> 작성자: 구글에서 공식적으로 승인되지 않았지만 러스트를 사용하는 사람들이 있습니다. 이 컴포넌트(KVM-A)에 러스트를 사용할 수 있었던 비결은 회사 내에 적합한 다른 언어가 없다고 동료들을 설득한 것이었습니다.
>
> 사실 러스트가 크롬 OS 빌드 환경에서 제대로 돌아가도록 많은 작업이 필요했습니다. 러스트 쪽 사람들이 내 질문에 답해 주고 큰 도움을 주었습니다.
>
> > **2017년 9월 27일 ekidd 씀:**
> >
> > > 이 컴포넌트(KVM-A)에 러스트를 사용할 수 있었던 비결은 회사 내에 적합한 다른 언어가
> > > 없다고 동료들을 설득한 것이었습니다.
> >
> > 제 프로젝트 중에 비슷한 경우가 하나 있었는데요. 복잡한 바이너리 데이터를 분석하는 vobsub 자막 디코더라는 프로그램으로, 언젠가는 웹 서비스로 실행하고 싶었습니다. 그러다 보니 내 코드에 취약점이 없는지 확인하고 싶었어요.
> >
> > 러스트로 코드를 작성한 다음 `cargo fuzz`를 사용하여 취약점을 찾아보았습니다. 10억(!) 번의 퍼지 반복을 실행한 후 버그 다섯 개를 발견했습니다(https://github.com/rust-fuzz/trophy-case 참고).
> >
> > 다행히 이러한 버그 중 어느 하나도 실제로 실제 악용되지는 않았습니다. 각 경우에 러스트의 다양한 런타임 검사를 통해 문제를 성공적으로 포착하여 패닉이 통제되도록 했습니다. 실제로 이렇게 하면 웹 서버가 깨끗하게 다시 시작돼요.
> >
> > 그래서 여기서 내가 얻은 교훈은 (1) GC(garbage collection)가 없고 (2) 보안이 중요한 상황에서 신뢰할 수 있는 언어를 원할 때 러스트는 탁월한 선택이라는 점입니다. 고(GO)처럼 리눅스 바이너리를 정적으로 연결할 수 있다는 사실은 좋은 장점입니다.

12 "Chrome OS KVM—A component written in Rust," *https://news.ycombinator.com/item?id=15346557*

> 2017년 9월 27일 Manishearth 씀:
> > 다행히 이러한 버그 중 어느 하나도 실제로 실제 악용되지는 않았습니다. 각 경우에
> > 러스트의 다양한 런타임 검사를 통해 문제를 성공적으로 포착하여 패닉이
> > 통제되도록 했습니다.
>
> 그냥 도움이 될까 덧붙이자면, 파이어폭스에서 러스트 코드 퍼징을 했던 경험이기도 한데요. 퍼징으로 다수의 패닉(그리고 디버그 단언(assertion), '안전한' 오버플로 단언 등)을 찾아냈습니다. 비슷한 파이어폭스 렌더링 엔진인 게코(Gecko) 코드에서 약 10년 동안 눈에 띄지 않았던 버그를 실제로 발견한 적도 있습니다.

이 글을 통해 러스트가 비교적 작은 프로젝트에서 기술적 도전을 극복하려는 엔지니어들에 의해 '상향식'으로 채택되었음을 알 수 있다. 이러한 성공에서 얻은 경험은 더 야심 찬 일에 착수하는 데 증거로 사용되었다.

2017년 후반부터 러스트는 지속적으로 더 성숙해졌으며 강력해졌다. 이로써 구글의 기술 지형에 받아들여졌으며, 현재는 안드로이드와 퓨셔 운영 체제 프로젝트에서 공식 승인까지 받았다.

1.3 언어 맛보기

이 절에서는 러스트를 직접 경험해 보기로 한다. 컴파일러 사용법을 이해하는 데서 시작해 간단한 프로그램을 작성하고 이 장의 끝에서는 완전한 프로젝트를 만들어 보겠다.

러스트를 설치하려면 *https://rustup.rs/*에서 제공하는 공식 설치 프로그램을 사용하자.

1.3.1 "Hello, world!" 프로그램을 편법으로 만들어 보기

대부분의 프로그래머들이 새로운 프로그래밍 언어를 접했을 때 가장 먼저 하는 일은 "Hello, world!"를 콘솔 화면에 출력하는 법을 익히는 것이다. 여러분도 같은 일을 할 텐데, 대신 요령을 좀 부릴 것이다. 성가신 오류에 맞닥뜨리기 전에 모든 것이 제대로 되어 있는지 확인할 것이다.

윈도우 사용자라면 러스트를 설치한 후 시작 메뉴에 있는 명령 프롬프트를 연다. 그리고 다음 명령을 입력한다.

```
C:\> cd %TMP%
```

리눅스나 macOS 사용자라면 터미널 창을 열고 다음과 같이 입력한다.

```
$ cd $TMP
```

앞으로는 모든 운영 체제에서 동일한 명령을 사용한다. 러스트를 제대로 설치하고 다음 명령을 실행하면 "Hello, world!"가 화면에 출력된다(물론 다른 출력도 같이 나온다).

```
$ cargo new hello
$ cd hello
$ cargo run
```

마이크로소프트 윈도우에서 cmd.exe를 실행했을 때 전체 과정은 다음과 같다.

```
C:\> cd %TMP%

C:\Users\Tim\AppData\Local\Temp\> cargo new hello
     Created binary (application) `hello` project

C:\Users\Tim\AppData\Local\Temp\> cd hello

C:\Users\Tim\AppData\Local\Temp\hello\> cargo run
   Compiling hello v0.1.0 (file:///C:/Users/Tim/AppData/Local/Temp/hello)
    Finished dev [unoptimized + debuginfo] target(s) in 0.32s
     Running `target\debug\hello.exe`
Hello, world!
```

리눅스나 macOS라면 콘솔 화면은 다음과 같을 것이다.

```
$ cd $TMP

$ cargo new hello
     Created binary (application) `hello` package

$ cd hello

$ cargo run
   Compiling hello v0.1.0 (/home/tsm/hello)
    Finished dev [unoptimized + debuginfo] target(s) in 0.26s
     Running `target/debug/hello`
Hello, world!
```

여기까지 성공했다면 잘했다! 여러분은 러스트 코드를 한 줄도 작성하지 않고 처음으로 실행했다. 이제 어떤 일이 있었는지 살펴보자.

러스트의 카고(cargo) 도구는 빌드 시스템이면서 패키지 관리자다. 즉, 카고는 러스트 코드를 실행 가능한 바이너리로 변환하며, 프로젝트의 의존성을 다운로드하고 컴파일하는 프로세스도 관리할 수 있다.

`cargo new` 명령은 표준 템플릿을 따르는 프로젝트를 만든다. `tree` 명령을 쓰면 `cargo new` 실행 후 생성되는 프로젝트 구조와 파일을 볼 수 있다.

```
$ tree hello
hello
├── Cargo.toml
└── src
    └── main.rs

1 directory, 2 files
```

카고로 만든 러스트 프로젝트의 구조는 전부 동일하다. 기본 디렉터리에는 Cargo.toml이라는 파일이 있는데 이 파일에는 프로젝트의 이름, 버전, 의존성 같은 메타데이터가 들어 있다. 소스 코드는 src 디렉터리에 있다. 러스트 소스 코드의 파일 확장자는 .rs다. `cargo new`로 만들어지는 파일을 보려면 `tree` 명령을 쓴다.

그다음에 실행했던 명령은 `cargo run`이다. 이 부분은 이해하기 쉬워 보이지만, 카고는 실제로 보이는 것 이상으로 많은 일을 했다. 우리는 카고에 프로젝트를 실행하도록 명령했다. 해당 명령을 실행했을 때 실제로는 실행할 수 있는 것이 전혀 없었다. 그래서 최대한 많은 오류 관련 정보를 제공하도록 사용자 대신 코드를 디버그 모드로 컴파일하게 설정한다. 여기까지 진행되면 src/main.rs 파일은 항상 "Hello, world!" 문구를 출력하는 코드를 포함하게 된다. 컴파일 결과 hello(또는 hello.exe)라는 파일이 생성된다. hello 파일이 실행되면 해당 결과가 화면에 출력된다.

`cargo run`을 실행하면 프로젝트에 새로운 파일과 디렉터리가 추가된다. 이제 Cargo.lock 파일과 target/ 디렉터리가 프로젝트의 기본 디렉터리에 추가되었다. 이 파일과 디렉터리는 카고에 의해 관리된다. 이는 컴파일 과정에서 만들어지는 것들로 이 파일과 디렉터리를 건드릴 필요는 없다. Cargo.lock은 모든 의존성에 대한 정확한 버전 번호를 지정하는 파일로, Cargo.toml 파일이 변경되기 전까지는 이후에도 동일한 방식으로 빌드가 정확히 이루어진다.

`tree`를 다시 실행하면 hello 프로젝트를 컴파일하기 위해 `cargo run`을 호출해서 생성된 새로운 구조가 나타난다.

```
$ tree --dirsfirst hello
hello
├── src
│   └── main.rs
├── target
│   └── debug
│       ├── build
│       ├── deps
│       ├── examples
│       ├── native
│       └── hello
├── Cargo.lock
└── Cargo.toml
```

여러 작업을 진행했고 실행도 했다. 잘했다! 지금까지 "Hello, world!" 프로그램을 편법으로 만들어 봤으니 이제는 더 긴 방법으로 같은 일을 해 보자.

1.3.2 첫 번째 러스트 프로그램

첫 번째 프로그램으로 다음과 같이 여러 언어로 된 문장을 출력하는 프로그램을 만들려고 한다.

```
Hello, world!
Grüß Gott!
안녕, 세상아!
```

첫 번째 줄은 앞서 보았던 것이다. 다른 두 줄은 러스트의 특징을 강조하기 위해 넣었다. 바로 쉬운 반복과 기본 지원되는 유니코드다. 이 프로그램을 만드는 데 앞에서처럼 카고를 이용한다. 다음은 프로젝트를 만드는 데 따라야 할 순서다.

1. 콘솔 창을 연다.
2. 마이크로소프트 윈도우에서는 `cd %TMP%`, 다른 운영 체제(리눅스나 macOS)에서는 `cd $TMP`를 실행한다.
3. `cargo new hello2`를 실행해서 새로운 프로젝트를 만든다.
4. `cd hello2`를 실행해서 프로젝트의 루트 디렉터리로 이동한다.
5. src/main.rs 파일을 텍스트 편집기로 연다.
6. 해당 파일의 내용을 예제 1.1의 내용으로 바꾼다.

다음 예제의 코드는 소스 코드 저장소에 있다. ch1/ch1-hello2/src/main.rs를 열도록 한다.

예제 1.1 "Hello, World!"를 세 가지 언어로 출력하기

```
01 fn greet_world() {
02   println!("Hello, world!");                ❶
03   let southern_germany = "Grüß Gott!";      ❷
04   let korean = "안녕, 세상아!";               ❸
05   let regions = [southern_germany, korean]; ❹
06
07   for region in regions.iter() {   ❺
08     println!("{}", &region);       ❻
09   }
10 }
11
12 fn main() {
13   greet_world();   ❼
14 }
```

❶ 느낌표 표시는 매크로를 의미한다. 나중에 알아보겠다.

❷ 러스트에서의 할당문이다. 좀 더 정확히 말하자면 let 키워드를 이용한 변수 바인딩이다.

❸ 유니코드가 기본으로 제공된다.

❹ 배열 리터럴을 표현할 때에는 대괄호([])를 쓴다.

❺ 많은 타입에 반복자(iterator)를 반환하는 iter() 메서드가 있다.

❻ 앰퍼샌드(&) 기호는 영역 내의 값을 읽기 전용으로 대여(borrow)할 때 사용된다.

❼ 함수를 호출한다. 괄호가 함수명에 붙어 있는 것에 유의한다.

이제 코드를 고쳤으니 cargo run을 hello2/ 디렉터리에서 실행한다. 다음과 같이 카고 자체에서 나오는 일련의 출력과 함께 세 가지 인사말이 출력된다.

```
$ cargo run
   Compiling hello2 v0.1.0 (/path/to/ch1/ch1-hello2)
    Finished dev [unoptimized + debuginfo] target(s) in 0.95s
     Running `target/debug/hello2`
Hello, world!
Grüß Gott!
안녕, 세상아!
```

잠시 시간을 내어 예제 1.1에 나온 러스트의 흥미로운 요소 몇 가지를 살펴보겠다.

눈에 띄는 것 중 하나는 러스트에서 다양한 범주의 문자를 사용할 수 있다는 점이다. 문자열(string)은 UTF-8로 인코딩되어 있다. 이는 비영어권 언어를 상대적으로 쉽게 사용할 수 있다는 것을 의미한다.

println 뒤에 붙는 느낌표는 어울리지 않아 보일 수 있다. 루비를 사용해 봤다면 이를 파괴적 연산(destructive operation: 객체를 변경하는 연산)을 나타내기 위해

사용했을 수도 있다. 러스트에서는 매크로를 사용했음을 알려 줄 때 사용한다. 지금은 매크로를 일종의 멋진 함수 정도로 생각하면 된다. 매크로를 사용하면 비슷비슷하게 중복되는 코드 조합(boilerplate code)을 쓰지 않아도 된다. `println!`을 예로 들면, 어떤 데이터 타입이라도 화면에 출력할 수 있도록 내부적으로 타입을 탐지하는 기능이 들어 있다.

1.4 이 책의 소스 코드 다운로드

이 책의 예제를 실제로 따라 하려면 예제의 소스 코드가 필요할 것이다. 모든 예제의 소스 코드는 다음 사이트에서 편리하게 내려 받을 수 있다.

- *https://manning.com/books/rust-in-action*
- *https://github.com/rust-in-action/code*

1.5 러스트의 생김새와 느낌

러스트는 해스켈 개발자나 자바 프로그래머와도 잘 어울리는 언어다. 해스켈과 자바같은 표현력이 풍분한 고수준 언어이면서, 동시에 저수준 언어에서 볼 수 있는 베어 메탈(bare-metal) 수준의 성능을 보여준다.

1.3절에서 몇 가지 "Hello, world!" 예제를 살펴보았으니 러스트의 특징을 좀 더 느낄 수 있도록 약간 더 복잡한 것에 도전해 보자. 예제 1.2는 기본적인 텍스트 처리를 러스트에서 어떻게 하는지 간단히 보여 준다. 이 코드는 ch1/ch1-penguins/src/main.rs 파일에 있다. 주목해야 할 몇 가지 부분은 다음과 같다.

- 일반적인 흐름 제어 메커니즘 — `for` 반복문과 `continue` 키워드를 포함한다.
- 메서드 문법 — 러스트는 객체 지향 언어가 아니어서 상속 등을 지원하지는 않지만, 객체 지향 언어에 있는 메서드 관련 요소를 가져왔다.
- 고차 프로그래밍 — 함수는 인자로도, 반환값으로도 쓰일 수 있다. 예를 들어 19행(`.map(|field| field.trim())`)은 익명 함수 또는 람다(λ) 함수로 알려진 클로저(closure)를 포함하고 있다
- 타입 애너테이션(type annotation) — 상대적으로 적게 쓰이지만 이따금 컴파일러에 일종의 힌트를 줄 때 필요하다(`if let Ok(length)`로 시작하는 27행을 보라).