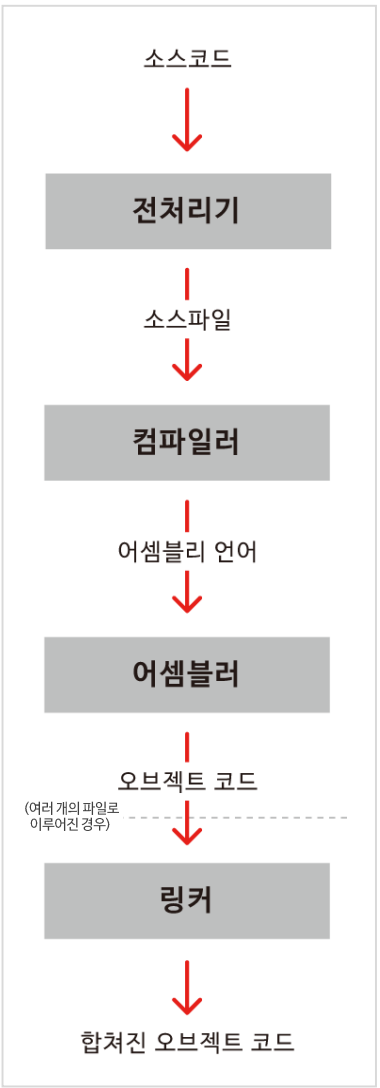
4.1 컴파일

개요

컴파일은 **소스 코드를 오브젝트 코드로 변환시키는 과정입니다.** 여기서 소스 코드는 여러분이 C언어와 같은 프로그래밍 언어로 작성한 코드이고, **오브젝트 코드**는 **기계어**라고도 알려져 있는데, **0과 1로 이루어져 있으며 컴퓨터에게 프로그램이 어떻게 실행되어야 하는지 알려주는 코드**입니다. make 명령어 자체는 컴파일러가 아니고, clang이라는 컴파일러를 호출해서 C 소스 코드를 오브젝트 코드로 컴파일 하도록 합니다.

핵심개념

* 컴파일
* 컴파일러
* 오브젝트 코드
* 전처리
* 어셈블
* 링크



전처리(Precompile)

컴파일의 전체 과정은 네 단계로 나누어볼 수 있습니다. 그 중 첫 번째 단계는 **전처리**인데, 전처리기에 의해 수행됩니다. # 으로 시작되는 C 소스 코드는 전처리기에게 **실질적인 컴파일이 이루어지기 전에 무언가를 실행**하라고 알려줍니다.

예를 들어, #include는 전처리기에게 다른 파일의 내용을 포함시키라고 알려줍니다. 프로그램의 소스 코드에 #include <stdio.h>와 같은 줄을 포함하면, 전처리기는 새로운 파일을 생성하는데 이 파일은 여전히 C 소스 코드 형태이며 stdio.h 파일의 내용이 #include 부분에 포함됩니다.

컴파일(Compile)

전처리기가 전처리한 소스 코드를 생성하고 나면 그 다음 단계는 **컴파일**입니다. **컴파일러**라고 불리는 프로그램은 **C 코드를 어셈블리어라는 저수준 프로그래밍 언어로 컴파일**합니다.

어셈블리는 C보다 연산의 종류가 훨씬 적지만, 여러 연산들이 함께 사용되면 C에서 할 수 있는 모든 것들을 수행할 수 있습니다. C 코드를 어셈블리 코드로 변환시켜줌으로써 컴파일러는 컴퓨터가 이해할 수 있는 언어와 최대한 가까운 프로그램으로 만들어 줍니다. 컴파일이라는 용어는 소스 코드에서 오브젝트 코드로 변환하는 전체 과정을 통틀어 일컫기도 하지만, 구체적으로 전처리한 소스 코드를 어셈블리 코드로 변환시키는 단계를 말하기도 합니다.

어셈블(Assemble)

소스 코드가 어셈블리 코드로 변환되면, 다음 단계인 **어셈블** 단계로 **어셈블리 코드를 오브젝트 코드로 변환**시키는 것입니다. 컴퓨터의 중앙처리장치가 프로그램을 어떻게 수행해야 하는지 알 수 있는 명령어 형태인 **연속된 0과 1들로 바꿔주는 작업**이죠. 이 변환작업은 **어셈블러**라는 프로그램이 수행합니다. 소스 코드에서 오브젝트 코드로 컴파일 되어야 할 파일이 딱 한 개라면, 컴파일 작업은 여기서 끝이 납니다. 그러나 그렇지 않은 경우에는 링크라 불리는 단계가 추가됩니다.

링크(Link)

만약 프로그램이 (math.h나 cs50.h와 같은 라이브러리를 포함해) **여러 개의 파일로 이루어져 있어 하나의 오브젝트 파일로 합쳐져야 한다면** 링크라는 컴파일의 마지막 단계가 필요합니다. 링커는 여러 개의 다른 오브젝트 코드 파일을 실행 가능한 하나의 오브젝트 코드 파일로 합쳐줍니다. 예를 들어, 컴파일을 하는 동안에 CS50 라이브러리를 링크하면 오브젝트 코드는 GetInt()나 GetString() 같은 함수를 어떻게 실행할 지 알 수 있게 됩니다.