2주차 예비보고서

전공: 수학과,컴퓨터공학과 학년: 3학년 학번: 20181256 이름: 김도현

2.

c/c++ 프로그래밍에서 컴파일 과정은 high level로 작성된 코드를 low level로 바꿔 기계에서 실행 되도록 해준다. preprocessing(프리프로세싱), compiling(컴파일링), assembling(어셈블링), linking(링킹)을 통해서 이루어진다.

일반적으로 Unix 및 Linux 등에서 컴파일 시 표준 컴파일러나 다름 없는 GNU C Compiler(GNU Compiler Colleciton)를 이용한다. gcc 는 preprocessing 과 compiling 에 이용하지만 Unix 내에서의 as(assembler) 를 호출해서 assembling 하게 만들고 ld(linker) 를 호출해서 linking 하게 만들어 object 파일 및 실행 파일을 만들도록 도와주기 때문에 컴파일의 전체적인 부분에 관여한다.

Preprocessing(프리프로세싱)은 preprocessor(프리프로세서)를 통해서 이루어진다. 이 과정에서는 컴파일러가 컴파일하기 전에 컴파일이 가능하도록 소스파일(ex. .c, .cc, .cpp)을 수정한다. 타 파일(헤더파일)을 합치거나 소스파일 내 일부 문자열을 컴파일에 용이하도록 저장된 문자열들로 바꾸기도 한다. 또한 c/c++ 은 매크로(확장)도 사용가능하다. #include(헤더파일), #define(매크로). (.c->.i)

Compiling(컴파일링)은 compilier(컴파일러)를 통해서 이루어진다. 이 과정에서는 preprocessing을 통해 확장된 소스파일을 front end에서 검사(lexical, syntax, semantic anaylsis)를 한다.

front end의 lexical(어휘 분석)은 소스파일의 소스코드를 token으로 나누고 syntax(구문 분석)에서

token을 이용하여 parse tree 를 만들고 문법적 에러를 찾아낸다. Semantic(의미 분석)에서는 parse tree 의 의미적 에러를 찾아낸다.( ex. a/0, int+char)만약 구문 분석에서 에러가 존재하면 컴파일 에러가 된다. 그리고 GIMPLE tree 를 만든다.

middle end를 통해 OS 와 같은 환경과 독립적인 최적화를 해서 Register Transfer Lanuage로 back end로 전달해주고 환경에 맞게 최적화를 한다. 이후 assembly 언어로 바꾼다.

‘gcc -S 소스파일’ 을 할 시 Assembling 이 일어나지 않고 Compiling까지만 일어난다. 즉 Assembly 언어로 번역한 .s 파일이 만들어진다.

Assembling(어셈블링)은 Assembler(어셈블러)를 통해서 이루어진다. 이 과정에서는 최종적으로 기계가 이해할 수 있는 low level 단계인 기계어(0,1 같은 것)로 번역한다. 이렇게 object 파일이 만들어진다.(.s->.o)

(unix에서 assembler 는 as)

‘gcc -c 소스파일’ 을 할 시 Linking이 일어나지 않고 Assembling까지만 일어난다. 즉 object 파일(.o)이 만들어진다.

Linking(링킹)은 Linker(링커)를 통해 어셈블링 과정에서 만들어진 object 파일, 코드에 사용된 library 들을 연결한다. 마지막 과정으로 코드를 실제로 실행 가능한 실행 파일로 만든다.(.o,.a->a.out,.exe)

(unix에서 linker 는 ld)

‘gcc 소스파일’ 을 할 시 모든 컴파일 과정이 다 일어난다.