2주차 예비보고서

전공: 컴퓨터공학과 학년: 4학년 학번: 20192135 이름: 윤영인

**1.**

UNIX 상에서 제공하는 C/C++ 관련 도구를 사용하여 컴파일 과정을 공부하여 본다.

**2.**

도표이(가) 표시된 사진

자동 생성된 설명

그림 1. 컴파일 과정

C의 컴파일 과정은 위의 그림과 같으며, 컴파일하고 디버깅할 때 사용하는 도구에는 gcc, gdb, make가 있다. 이 도구들에 대해 알아보도록 하자.

먼저, gcc는 preprocessor와 compiler의 역할을 하지만 내부적으로 UNIX의 assembler인 as와 linker인 ld를 호출하여 컴파일을 진행한다. 가장 단순하게 gcc로 컴파일 할 경우, gcc 뒤에 c 소스 파일을 열거한다. 만약 다른 목적이 있으면 옵션을 추가하는데, gcc -o “name”의 옵션을 사용하면 name의 이름을 가진 실행파일이 생성된다. 만약 이름을 지정하지 않는다면 gcc는 기본적으로 a.out을 생성한다. -c 옵션은 linking 이전의 과정만 수행하여 c 소스 파일의 목적 파일인 .o 파일을 생성한다.

다음으로, gdb는 디버깅을 유용하게 할 수 있는 환경을 제공한다. gdb로 디버깅 하기위해선 “gcc -o test -g test.c”로 디버깅 정보를 포함하여 컴파일을 해줘야 한다. “gdb test”와 같이 디버깅할 프로그램을 인자로 주어 명령어를 입력하면 gdb로 디버깅을 실행할 수 있다. 단축키인 ‘l’을 입력하면 list 명령이 실행되며, 기본적으로 실행 부분(test.c)의 10줄의 코드를 나열한다. breakpoint를 설정하고 run 명령어를 실행하면 해당 지점에서 프로그램이 중단된 후 gdb 프롬포트가 나타난다. breakpoint 명령어로는 “b (중단시킬 지점의 라인)” 또는 “b (함수명)”을 입력하면 된다. display를 사용하면 원하는 변수의 값을 확인할 수 있으며 next 또는 step 명령어를 사용하여 다음 라인의 코드를 실행할 수 있다. 이러한 방식으로 프로그램의 흐름이나 변수의 값을 확인하여 디버깅할 수 있다.

마지막은 make는 다수의 소스 파일들을 컴파일한다. 하지만 어플리케이션의 구성 방법을 자동으로 인식하지 않기 때문에 makefile을 작성하여 구성 방법을 make에게 알려줘야 한다. makefile은 의존성과 규칙으로 구성되며 target은 실행파일이다. 의존성은 생성되는 파일인 target과 이 target이 의존하는 소스 파일을 포함하며, 최종적인 어플리케이션에서 각 파일이 소스 파일과 관련되는 방법을 지정한다. 예를 들어, “myapp : main.o dog.o turtle.o”라면 myapp이 main.o, dog.o, turtle.o에 의존하는 것을 의미한다. 규칙은 의존하는 파일로부터 target을 생성하는 방법을 설명한다. 예를 들어,

myapp : main.o dog.o turtle.o

gcc -o myapp main.o dog.o turtle.o

라면 myapp이 main.o, dog.o, turtle.o에 의존하며 gcc -o 옵션을 사용하여 출력되는 파일의 이름을 myapp으로 지정한다.

또한 makefile은 매크로를 지원하여 특정 코드를 간단하고 변경하기 쉽게 작성할 수 있게 한다. 특정 문자로 끝나는 파일이 있을 때 다른 문자로 끝나는 파일을 생성하기 위해 접미어를 사용하는데, 가장 기본적으로 .c파일을 .o파일로 생성하는 “.SUFFIX: .c .o” 접미어를 사용한다.

.c.o:

gcc -c -o $@ $(CFLAG) $<

의 규칙을 사용한다면, 확장자가 .c인 파일을 .o 파일을 생성한다는 의미이다. 여기서 -c와 -o 옵션을 사용하여 c 소스파일을 사용하여($<) 오브젝트 파일을 ($@) 생성한다.

**3. 참조**

1) 강의자료, 07\_UNIX-2\_C\_VS10.pdf