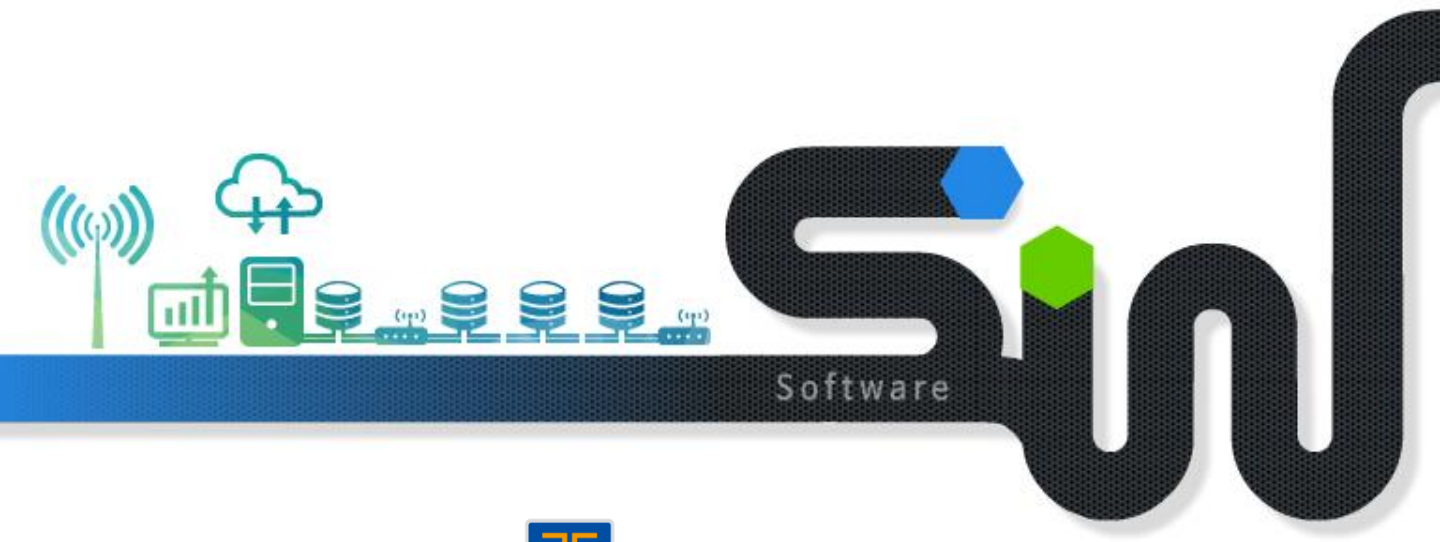
응용 SW 기초 활용 기술 part 1

리눅스 컴파일러와 디버거



한국기술교육대학교
온라인평생교육원

학습목표

- gcc 컴파일러
- gdb/ddd 디버거

- gcc 컴파일러의 개념을 설명하고 gcc 컴파일러를 사용하여 프로그램 컴파일을 할 수 있다.
- gdb와 ddd 디버거의 개념을 설명하고, 프로그램을 디버깅 할 수 있다.

gcc 컴파일러

1 gcc 컴파일러 개요

1) gcc(GUN C Compiler) 컴파일러 소개

gcc (GNU C Compiler)	GNU에서 만든 C 컴파일러

지원 CPU 아키텍처	ARM, DEC, AVR, i386, PPC, SPARC, M
gcc 컴파일 가능 언어	C, Fortran, Ada, Java, Objective-C
설치 방법	gcc 명령 실행

각 언어의 컴파일러가 설치되어 있어야 함

GCC (GNU Compiler Collection)	다양한 컴파일러를 포함하는 컴파일러 묶음

gcc, c++, Java, Ada, Fortran, Objective-C

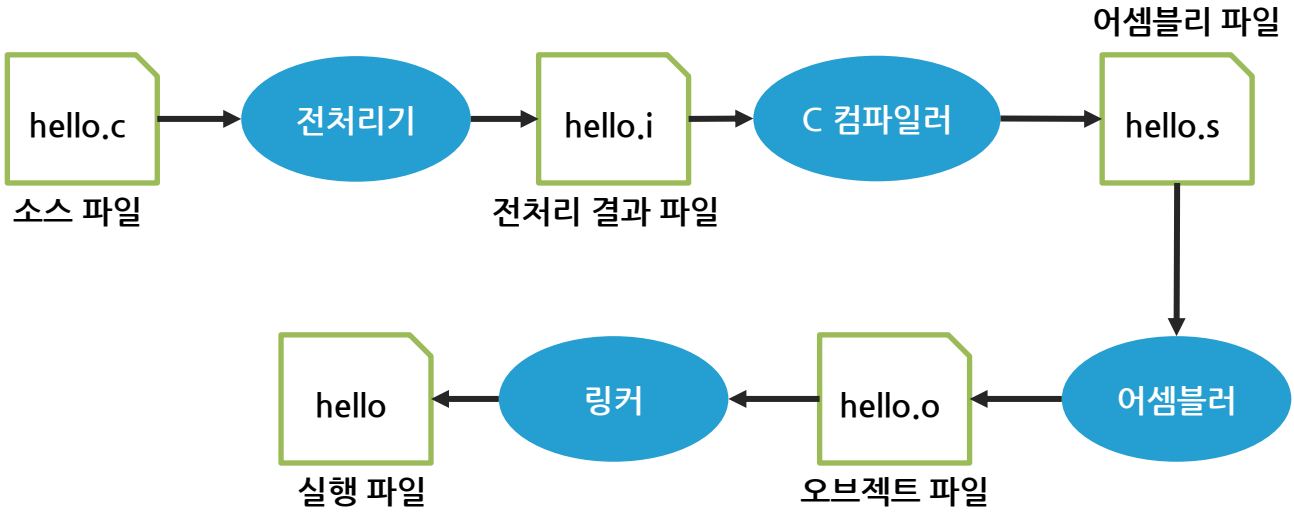
GCC와 gcc는 다른 의미로 사용함

gcc 컴파일러

1 gcc 컴파일러 개요

2) gcc 컴파일 과정

gcc 컴파일러로 c 언어를 컴파일하는 과정



1 gcc 컴파일러 개요

3) 컴파일 드라이버(Compile Driver)

전처리기, 컴파일러, 어셈블러, 링커를 호출하여 소스가 컴파일되도록 하는 역할

gcc 컴파일러가 아닌, C 컴파일 드라이버가 정확한 개념임

관행적으로 gcc 컴파일러라고 부름

컴파일러 종류

컴파일러	설명
cc1	C 컴파일러
cc1plus	C++ 컴파일러
collect2	링커
f951	포트란 95 컴파일러
f771	포트란 77 컴파일러
jc1	자바 컴파일러
cc1obj	Objective-C 컴파일러

컴파일 드라이버 종류

컴파일 드라이버	설명
cc	C 컴파일 드라이버
gcc	C 컴파일 드라이버
c++	C++ 컴파일 드라이버
g++	C++ 컴파일 드라이버

1 gcc 컴파일러 개요

4) gcc 컴파일러 옵션

1. C 컴파일 드라이버 옵션
2. 전처리기 옵션
3. 컴파일러 옵션
4. 어셈블러 옵션
5. 링커 옵션

1 gcc 컴파일러 개요

4) gcc 컴파일러 옵션

1 C 컴파일 드라이버 옵션

■ 컴파일 진행 과정에 대한 옵션

옵션	설명
-E	전처리 과정 화면 출력, 전처리 과정만 수행
-S	어셈블리 파일 저장(*.s), 어셈블리 과정까지만 수행
-c	오브젝트 파일 저장(*.o), 컴파일 과정까지만 수행
-v	컴파일 과정 화면 출력
-save-temps	컴파일 전 과정의 중간 생성 파일 저장

2 전처리기 옵션

■ 헤더 파일과 define 매크로와 관련된 옵션

옵션	설명
-I	헤더 파일의 탐색 디렉토리 추가
-D	매크로 정의를 외부에서 할 경우 사용(#define)
-M	Make를 위한 소스 파일의 모든 종속 항목 출력
-MM	Make를 위한 기본 include 디렉토리에 있는 헤더 파일을 제외하고 종속 항목 출력
-nostdinc	기본 include 디렉토리에서 헤더 파일 탐색을 하지 않고, -I 옵션에 추가한 디렉토리에서만 헤더 파일을 탐색

1 gcc 컴파일러 개요

4) gcc 컴파일러 옵션

3 컴파일러 옵션(C 언어 옵션)

- C 언어 종류와 표준 관련 옵션

옵션	설명
-ansi	ANSI-C 표준으로 문법 체크
-std=[표준명]	c89 : ANSI-C와 동일 c99 : ANSI-C 업그레이드 gnu89 : 기본 gnu99 : gnu89 업그레이드
-traditional	Traditional C 문법

4 컴파일러 옵션(경고 옵션)

- 컴파일 경고 수위를 조절하는 옵션

옵션	설명
-Wall	모든 모호한 문법에 대한 경고 메시지 출력
-W	합법적이지만 모호한 문법에 대한 경고 메시지 출력
-w	모든 경고 메시지 제거
-Werror	모든 경고를 에러로 처리

- 두 옵션을 같이 사용하면 아주 사소한 경고도 모두 출력됨
- 소스코드의 엄격성을 보장할 경우에 사용

1 gcc 컴파일러 개요

4) gcc 컴파일러 옵션

5 컴파일러 옵션(최적화 옵션)

- 소스 코드를 최적화하는 옵션

옵션	설명
-O[최적화 레벨]	최적화 레벨 지정
-O0	최적화 수행 안 함
-O1	기본 최적화
-O2	일반 애플리케이션 또는 커널 컴파일 시 사용
-O3	모든 최적화 실행
-Os	사이즈 최적화(임베디드 시스템 등)

6 컴파일러 옵션(디버깅 옵션)

- 애플리케이션 디버깅 옵션

옵션	설명
-g	■ 디버깅 정보를 실행 파일에 삽입함 ■ gdb 디버거로 소스를 보면서 디버깅 가능
-g0	■ 디버깅 정보 삽입 안 함
-g2	■ -g 옵션과 동일
-g3	■ 디버깅 정보를 가장 많이 삽입 ■ 개발 단계에서만 사용

1 gcc 컴파일러 개요

4) gcc 컴파일러 옵션

7 컴파일러 옵션(어셈블러 옵션)

■ 어셈블리 코드 관련 옵션

옵션	설명
-Wa,-al	어셈블된 인스트럭션을 보여줌
-Wa,-as	정의된 심볼을 보여줌

8 컴파일러 옵션(링커 옵션)

■ 오브젝트 코드 생성과 관련된 라이브러리 지정 옵션

옵션	설명
-L[라이브러리 디렉토리]	링킹 시 참고할 라이브러리가 있는 디렉토리 지정
-l[라이브러리 이름]	링킹 시 참고할 라이브러리 파일 이름 지정

gcc 컴파일러

2 gcc 컴파일러 활용

1) gcc 컴파일 명령(옵션 없이 컴파일)

옵션 없이 컴파일	# gcc hello.c

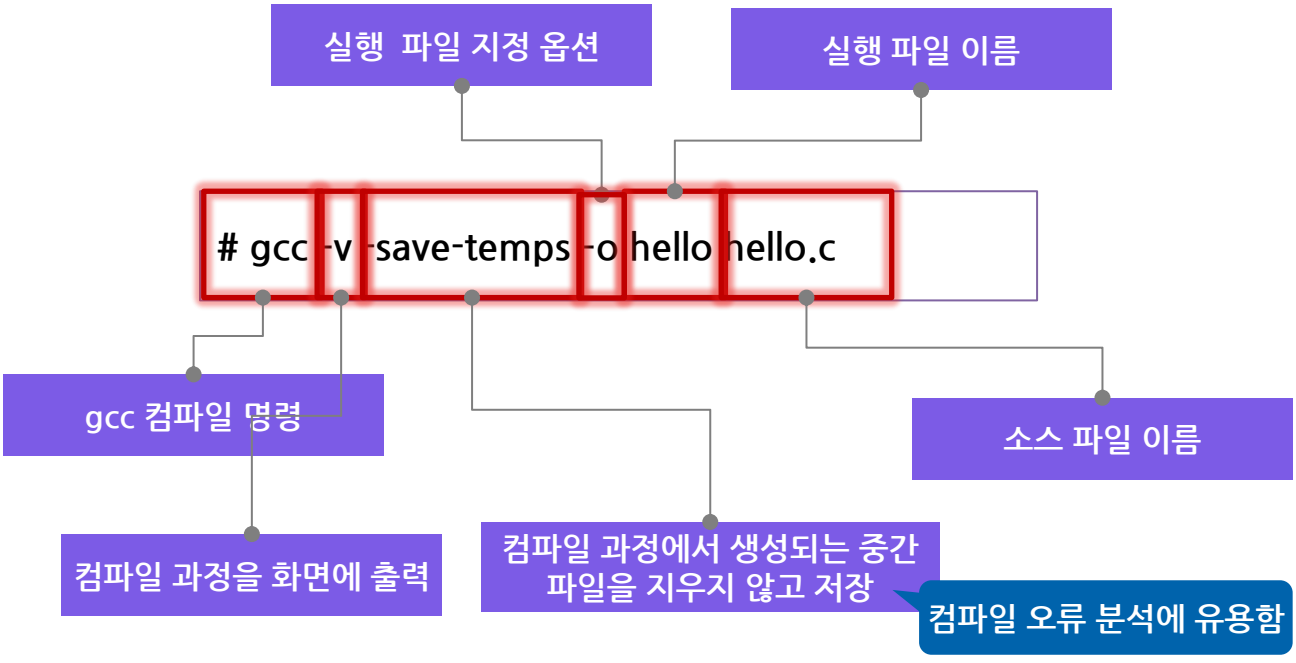
중간 생성 파일은
저장되지 않음

실행 파일	a.out

gcc 컴파일러

2 gcc 컴파일러 활용

2) gcc 컴파일 명령(옵션 사용)



gdb/ddd 디버거

1 gdb 디버거

1) gdb 디버거 소개

1. 매우 강력한 기능을 가지고 있으며, 리눅스 운영체제에서 거의 표준에 가까움
2. GNU에서 만든 디버거
3. CLI(Command Line Interface)를 통해서 정보 제공
   - 제공 정보 : 메모리, 스택, 레지스터 등(임베디드 시스템 개발)
4. 다양한 언어 지원
5. 다양한 CPU 아키텍처 지원

gdb/ddd 디버거

1 gdb 디버거

2) gdb 디버거를 위한 gcc 컴파일

필수 옵션	-g

최적화 옵션은 사용하지 않음

원래의 소스 자체를 디버깅할 필요가 있음

1 gdb 디버거

3) gdb 디버거 시작과 종료

시작 방법

명령	예제
# gdb [실행 파일명]	# gdb hello
# gdb [실행 파일명] [core 파일명]	# gdb hello core.1356
# gdb [실행 파일명] [실행 중인 프로세스의 PID]	# gdb hello 2435

종료 방법

명령
(gdb) q
(gdb) [ctrl] + [d]

gdb/ddd 디버거

1 gdb 디버거

4) gdb 디버거 명령

설명	명령
소스 보기 명령	l(list)
브레이크 포인트 설정 명령	b(Break), rb(Recursive Break)
브레이크 포인트 삭제 명령	cl(Clear), d(Delete)
브레이크 포인트 활성화/비활성화 명령	enable, disable
프로그램 실행 및 종료 명령	r(Run), k(Kill)
프로그램 진행 명령	s(Step), n(Next), c(Continue), u, finish, return, advance
와치 포인트 설정 명령	watch, rwatch(Read watch), awatch(All watch)
변수 디버깅 명령	p[Print], display, undisplay

브레이크 포인트 : 프로그램 실행 중 멈추는 위치

와치 포인트 : 어떤 변수에 값이 지정되거나 읽혀질 때마다 브레이크를 설정할 경우에 사용

gdb/ddd 디버거

2 ddd 디버거

1) ddd(the Data Display Debugger) 디버거 소개

1. GUI 환경을 제공함
2. gdb 디버거 사용
3. 사용 가능 디버거
   - dbx, xdb, jdb 등

2 ddd 디버거

2) ddd 디버거 설치 및 실행

1 설치

- ddd 디버거는 기본적으로 설치가 되어 있지 않음
- 설치 명령 : # yum install ddd

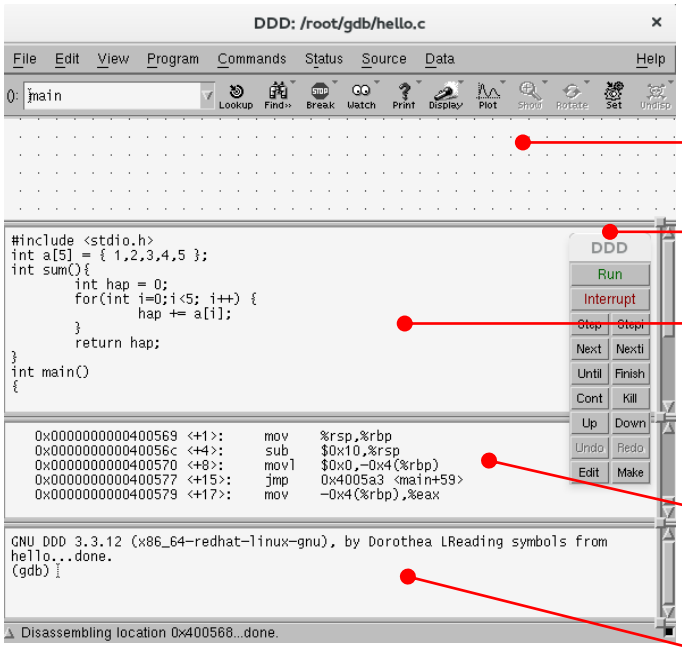
2 시작 방법

명령
ddd [실행 파일명]
ddd [실행 파일명] [core 파일명]
ddd [실행 파일명] [실행 중인 프로세스의 PID]

gdb/ddd 디버거

2 ddd 디버거

3) ddd 디버거 화면 구성



데이터창
- 변수나 데이터 표시창
- display 명령에 의해 표시됨

명령툴창
- 디버깅 명령 버튼창

소스창
- 소스가 출력되는 창
- 브레이크 포인트 설정

머신코드창
- 어셈블리 코드 출력창

디버깅 콘솔창
- gdb 명령창

gdb/ddd 디버거

2 ddd 디버거

4) ddd 디버거 명령툴 창 설명

명령	설명
Run	프로그램 실행
Interrupt	프로그램 실행 중지
Step	현재 행 실행, 함수 내부 들어감
Stepi	현재 인스트럭션 실행, 함수 내부 들어감
Next	현재 행 실행, 함수 내부 들어가지 않음
Nexti	현재 인스트럭션 실행 및 함수 내부로 들어가지 않음
Until	현재 loop 빠져 나감

명령	설명
Finish	현재 함수 실행 후 빠져 나감
Cont	다음 브레이크 포인트까지 실행
Kill	프로그램 실행 종료
Up	현재 함수를 호출한 함수로 소스창 이동
Down	현재 함수가 호출한 함수로 소스창 이동
Undo	명령 수행 전 상태로 돌아감
Redo	바로 전 수행 명령 재실행
Edit	소스 수정
Make	재컴파일 후 make 명령 실행

요점정리

gcc 컴파일러

- gcc(GUN C Compiler) 컴파일러 소개
  - GNU에서 만든 C 컴파일러
  - 지원 CPU 아키텍처 : ARM, DEC, AVR, i386, PPC, SPARC, M68xx 등
- GCC(GNU Compiler Collection)
  - 다양한 컴파일러를 포함하는 컴파일러 묶음
  - gcc, c++, Java, Ada, Fortran, Objective-C
- gcc 컴파일 가능 언어
  - C, Fortran, Ada, Java, Objective-C
- gcc 컴파일 과정
  - 소스 파일(*.c) - 전처리기(*.i) - 컴파일러(*.s) - 어셈블러(*.o) - 링커 - 실행 파일
- gcc 컴파일 명령
  - 옵션 없이 컴파일 : # gcc hello.c
  - 실행 파일 : a.out
- 컴파일 드라이버(Compile Driver)
  - 전처리기, 컴파일러, 어셈블러, 링커를 호출하는 역할
  - gcc : C 컴파일 드라이버
- gcc 옵션 분류
  - C 컴파일 드라이버 옵션, 전처리기 옵션, 컴파일러 옵션, 어셈블러 옵션, 링커 옵션

요점정리

gdb/ddd 디버거

- gdb 디버거 소개
  - GNU에서 만든 디버거
  - CLI(Command Line Interface)를 통해서 정보 제공
  - 제공 정보 : 메모리, 스택, 레지스터 등(임베디드 시스템 개발)
  - 다양한 언어 지원
  - 다양한 CPU 아키텍쳐 지원
- gdb 디버거를 위한 gcc 컴파일
  - 필수 옵션 : -g
  - 최적화 옵션 사용 안 함
- ddd(the Data Display Debugger) 디버거
  - GUI 환경 제공
  - gdb 디버거 사용
  - 사용가능 디버거 : dbx, xdb, jdb등
- ddd 디버거 설치
  - # yum install ddd

요점정리

POINT MANUAL

+ gdb 시작 명령

명령	설명
# gdb [실행 파일명]	# gdb hello
# gdb [실행 파일명] [core 파일명]	# gdb hello core.1356
# gdb [실행 파일명] [실행 중인 프로세스의 PID]	# gdb hello 2435

+ gdb 종료 명령

명령
(gdb) q
(gdb) [ctrl] + [d]