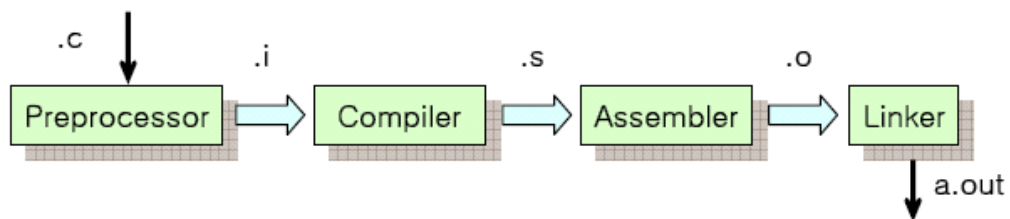


GCC – 4th Lab

1. gcc?

- FSF (Free Software Foundation)의 C/C++ 컴파일러
- Compiler, Assembler, Linker의 역할



- 위 과정이 gcc 명령의 호출로 실행
- gcc 4.1.2 in martini

- simple example

```
$ vi hello.c
#include <stdio.h>

int main()
{
    printf("hello world\n");
    return 0;
}

$ gcc hello.c
$ ./a.out
hello world
```

2. gcc Basic Options

- -v (verbose)
 - 컴파일러의 버전과 각 단계에서 실행하는 자세한 사항 출력
 - 컴파일 하는 과정에서 어떤 옵션이 사용되는지 알고자 할 때 사용

```
$ gcc -v
$ gcc -v hello.c
```

- -o
 - 실행파일(Executable File)명을 지정함
 - -o 옵션을 명시하지 않을 경우 실행파일명은 a.out

4190.102A Computer Programming
(2007 Winter)

```
$ gcc -o hello hello.c
```

- -c
 - 컴파일은 하되 링크는 하지 않음
 - .o (Object Module) 파일 생성

```
$ gcc -c hello.c  
$ gcc hello hello.o
```

- -D
 - 컴파일러 매크로: 컴파일러가 파일을 제어할 때 사용하는 정보
 - -D 옵션은 매크로를 정의함 (소스코드의 #define문의 역할)

```
(test.c)  
#include <stdio.h>  
int main()  
{  
    #ifdef DEBUG  
        printf("Debug message\n");  
    #endif  
  
    #ifdef INFO_FILE  
        printf("%s\n", INFO_FILE);  
    #endif  
  
    return 0;  
}  
  
$ gcc -DINFO_FILE= \"infofile\" test.c  
(#define INFO_FILE \"infofile\")  
  
$ gcc -DDEBUG test.c  
(#define DEBUG)
```

- -I
 - 헤더파일의 위치를 지정
 - 비 표준 라이브러리를 위한 헤더 파일의 디렉토리를 지정

```
#include <stdio.h>          - (1)  
#include "my_header.h"    - (2)
```

- (1)의 경우 : 시스템 표준 헤더 디렉토리인 /usr/include를 기준으로 헤더파일을 찾아서 include한다.
- (2)의 경우 : 지금 컴파일러가 실행되고 있는 현재 디렉토리를 기준으로 헤더파일을 찾아서 include한다.
- 헤더파일이 위 두 디렉토리 아닌 디렉토리에 있다면 아래와 같이 해당 디렉토리를 -I 옵션을 이용해 지정한다.

```
$ gcc -c -I../include test.c
```

4190.102A Computer Programming (2007 Winter)

- -l
 - 링크할 라이브러리를 지정
 - 표준 라이브러리를 담고 있는 디렉터리(/lib, /usr/lib)에서 lib`name`.a를 찾음
 - -l 옵션은 소스 파일 뒤에 위치

```
$ gcc -o solve main.o -lm
$ gcc -o thread-create thread-create.c -lpthread
```

- -L
 - 라이브러리를 찾는 디렉토리 목록에 사용자가 지정하는 디렉토리를 추가

```
$ gcc -o solve main.o -L../lib -lusermath -lm
```

3. gcc Other Options

■ Warning Message Control

- -w
 - 모든 경고 메시지를 나오지 않게 함
- -W
 - 합법적이지만 다소 모호한 코딩에 대하여 부가적인 경고 메시지 출력
- -Wall
 - 모호한 코딩에 대하여 자세한 경고 메시지

■ Optimization

- 프로그램의 수행 속도를 컴파일러가 최적화 함
- -O
 - -O1과 같음
- -O0
 - 최적화 하지 않음 (Default)
- -O1
 - 코드 크기와 실행시간을 줄여줌
- -O2
 - 더 많은 최적화를 수행함

■ Debugging and Profiling

- Profiling : 프로그램의 어느 부분에서 가장 많은 실행 시간을 소비되고 있는 지에 대한 정보 및 프로그램 내에 있는 함수들간의 호출 관계에 대한 정보를 알 수가 있다.
- -g
 - gdb에서 쓸 수 있는 확장된 심볼 테이블을 생성
- -p
 - prof에서 profiling할 수 있는 데이터를 생성하는 프로그램으로 컴파일
- -pg
 - gprof에서 profiling할 수 있는 데이터를 생성하는 프로그램으로 컴파일

4190.102A Computer Programming (2007 Winter)

■ Library Designation

- -static
 - 공유 라이브러리가 아닌 정적 라이브러리와 링크
- -shared (default)
 - 가능한 한 공유 라이브러리와 링크하고 공유 라이브러리가 없는 경우에만 정적 라이브러리와 링크

4. Static vs. Shared (Dynamic) Libraries

- Static Libraries
 - Collections of Object Files That Are Linked into the Program during the Linking Phase of Compilation, and Are Not Relevant during Runtime.
- Shared(Dynamic) Libraries
 - Collections of Object Files That Are Linked into the Program at Runtime.

➤ Example

usermath.c

```
unsigned long pow2(int x, int y)
{
    int i;
    unsigned long result = 1;
    for(i = 0; i < y; i++)
        result *= (unsigned long)x;
    return result;
}
```

pow.c

```
main(int argc, char *argv[])
{
    int a, b;
    unsigned long c;
    if(argc < 3){
        printf("Usage: pow [base] [exponent]\n");
        exit(1);
    }
    a = atoi(argv[1]);
    b = atoi(argv[2]);
    c = pow2(a, b);
    printf("%d ^ %d = %u\n", a, b, c);
    exit(0);
}
```

➤ Shared Library 생성 (.so)

```
$ gcc -shared -o libusermath.so usermath.c
```

➤ Shared Library 이용 컴파일

- -shared는 default 옵션

4190.102A Computer Programming (2007 Winter)

```
$ gcc [-shared] -o pow_shared pow.c -L./ -lusermath
```

- Shared Dependency 확인

```
$ ldd pow_shared
```

- Shared Library 이용 실행

- usermath 라이브러리가 LD_LIBRARY_PATH에 존재하는지 확인
- PATH지정 후 export

```
$ env | grep 'LD_LIBRARY_PATH'  
$ LD_LIBRARY_PATH=`pwd`:LD_LIBRARY_PATH  
$ export LD_LIBRARY_PATH  
$ ./pow_shared 2 3
```

- Static Library 생성 (.a)

```
$ gcc -c usermath.c  
$ ar rscv libusermath2.a usermath.o
```

- Static Library 이용 컴파일

```
$ gcc [-static] -o pow_static pow.c -L./ -lusermath2
```

- Static Library 이용 실행

```
$ ./pow_static 2 3
```

5. 참고자료

- <http://wiki.kldp.org/wiki.php/%B0%B3%B9%DF%C0%DA%C4%DA%B3%CA#s-3.1>