


제3장 C 프로그래밍 환경

학습 목표

- 문서 편집 : vi, geidt
- C 컴파일러 사용: gcc
- 컴파일 자동화: make
- 디버깅: gdb
- 통합개발환경: Eclipse
- 라이브러리 관리: ar
- 소스 관리: ctags
- 형상 관리: CVS, SVN, git





3.4 라이브러리



라이브러리 작성

- 라이브러리

- 오브젝트 파일들의 모임
- 재사용의 가능성이 많은 코드들을 라이브러리로 생성
- `/usr/lib` : 컴파일러가 제공하는 표준 라이브러리

- ar

- ar : Library Archives
- 오브젝트 모듈을 라이브러리로 보관하여 필요 시에 사용
- 사용법 : `ar [option] archive [member-file]`
- 라이브러리 파일 형식
`liblibrary_name.a`



ar 명령어 옵션

| 옵 션 | 설 명 |
|-----|------------------------------------------------------------------------------------------|
| -r | 아카이브(Archive)에 있는 특정 목적 파일(Object)을 수정하거나 신규로 추가 ar -r libmy.a func1.o func2.o |
| -d | 아카이브(Archive)에 있는 특정 목적 파일(Object)을 삭제 한다. ar -d libmy.a func1.o |
| -t | 아카이브(Archive)에 있는 목적 파일(Object)의 목록 을 보여준다. ar -t libmy.a |
| -v | 아카이브(Archive)에 있는 목적 파일(Object)의 정보를 자세히 보여준다. (-t옵션과 같이 사용) |
| -s | 아카이브 내 목적 파일의 참조 속도를 높이기 위해 인덱스 를 만들어 준다. |



라이브러리 작성

```
vi ex7-a.c
```

```
#include <stdio.h>
void a(void)
{
    printf("func a\n");
}
```

```
vi ex7-b.c
```

```
#include <stdio.h>
void b(void)
{
    printf("func b\n");
}
```

```
[sugar@hussein bit]$ gcc -c ex7-a.c ex7-b.c
```

```
[sugar@hussein bit]$ ls
```

```
ex7-a.o ex7-b.o ex7-a.c ex7-b.c
```

```
[sugar@hussein bit]$ ar rcv libex7.a ex7-a.o ex7-b.o
```

```
a - ex7-a.o          cv (x)
```

```
a - ex7-b.o          rv (o)
```

```
[sugar@hussein bit]$ ls
```

```
ex7-a.o ex7-b.o libex7.a ex7-a.c ex7-b.c
```

```
[sugar@hussein bit]$ ar t libex7.a
```

```
ex7-a.o
```

```
ex7-b.o
```

```
[sugar@hussein bit]$
```



작성된 라이브러리 사용

-L<directory>

현재 디렉토리(.)를 라이브러리를 찾는 경로에 추가

vi ex7.h

```
void a(void);  
void b(void);
```

vi ex7.c

```
#include <stdio.h>  
#include "ex7.h"  
int main(int argc, char ** argv)  
{  
    a();  
    b();  
    return 0;  
}
```

[sugar@hussein bit]\$ **gcc -L. ex7.c -lex7**

[sugar@hussein bit]\$./a.out libex7.a 라이브러리를 찾아 프로그램에 링크
func a -l<library_name>
func b -lex7은 libex7.a (정적 라이브러리)

[sugar@hussein bit]\$



오브젝트 파일 순서

```
vi ex7-a2.c
#include <stdio.h>
void a(void)
{
    printf("func a2\n");
}
```

```
[sugar@hussein bit]$ gcc -c ex7-a2.c
[sugar@hussein bit]$ ar qv libex7.a ex7-a2.o
a - ex7-a2.o      //제일 마지막에 추가
[sugar@hussein bit]$ ar t libex7.a
ex7-a.o
ex7-b.o
ex7-a2.o
[sugar@hussein bit]$ gcc -L. ex7.c -lex7
[sugar@hussein bit]$ ./a.out
func a
func b
```

변하지않음



오브젝트 파일 순서

```
[sugar@hussein bit]$ ar mv libex7.a ex7-a.o //해당 .o file을 제일 마지막으로 이동  
m - ex7-a.o
```

```
[sugar@hussein bit]$ ar t libex7.a  
ex7-b.o
```

```
ex7-a2.o
```

```
ex7-a.o // 제일 마지막으로 이동
```

```
[sugar@hussein bit]$ gcc -L. ex7.c -lex7
```

```
[sugar@hussein bit]$ ./a.out
```

```
func a2
```

```
func b
```

```
[sugar@hussein bit]$
```

결과가

func a

func b

가 아니고

func a2

func b

가 된 이유는?



오브젝트 파일 추출

```
[sugar@hussein bit]$ ar t libex7.a
```

```
ex7-b.o
```

```
ex7-a2.o
```

```
ex7-a.o
```

```
ar x libex7.a  
모든 파일 추출
```

추출

```
yu22312072@acslab-146:~/8$ ls -l ex7-a.o  
-rw-r--r-- 1 yu22312072 yu22312072 1480 12월  3 20:20 ex7-a.o  
yu22312072@acslab-146:~/8$ ar xo libex7.a ex7-a.o  
yu22312072@acslab-146:~/8$ ls -l ex7-a.o  
-rw-r--r-- 1 yu22312072 yu22312072 1480  1월  1 1970 ex7-a.o
```

```
[sugar@hussein bit]$ ar x libex7.a ex7-a.o
```

```
[sugar@hussein bit]$ date
```

```
Wed Jun  7 16:22:31 KST 2000
```

```
[sugar@hussein bit]$ ls -l ex7-a.o
```

```
-rw-r--r--  1 sugar  users          908 Jun  7 16:22 ex7-a.o [추출된 시각-현재]
```

ar xo: 아카이브 내부에 저장된 원래 생성 시간(아카이브에 추가될 당시의 시간)을 유지합니다.

```
[sugar@hussein bit]$ ar xo libex7.a ex7-a.o
```

```
[sugar@hussein bit]$ ls -l ex7-a.o
```

```
-rw-r--r--  1 sugar  users          908 Jun  7 15:55 ex7-a.o [목적 파일이 생성된 시각-과거]
```

```
[sugar@hussein bit]$ gcc -c ex7-a.c //ex7-a.c 갱신 후 아래에서 재첨가
```

```
[sugar@hussein bit]$ ar ru libex6.a ex7-a.o [ru 기존 아카이브 파일에 오브젝트 추가]
```

오브젝트 파일의 순서관계

```
[sugar@hussein bit]$ ar rcv libex7-2.a ex7-a.o
```

```
a - ex7-a.o
```

```
[sugar@hussein bit]$ ar rcv libex7-2.a ex7-b.o
```

```
a - ex7-b.o
```

```
[sugar@hussein bit]$ ar rav ex7-a.o libex7-2.a ex7-a2.o
```

```
a - ex7-a2.o
```

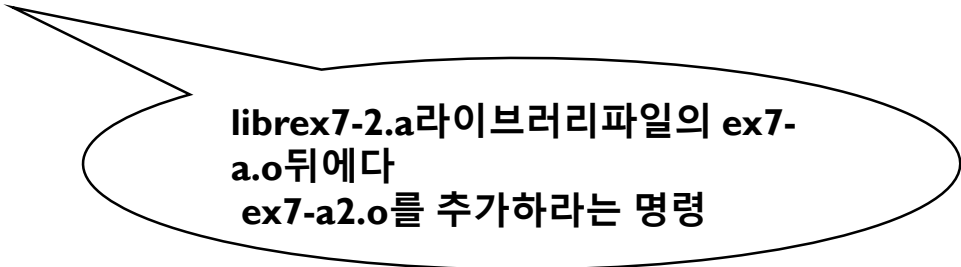
```
[sugar@hussein bit]$ ar t libex7-2.a
```

```
ex7-a.o
```

```
ex7-a2.o
```

```
ex7-b.o
```

```
[sugar@hussein bit]$
```



libex7-2.a 라이브러리 파일의 ex7-a.o 뒤에다
ex7-a2.o를 추가하라는 명령

ar 명령어의 전체 옵션 참조
<https://linux.die.net/man/1/ar>



라이브러리 사용

```
vi ex5.c
```

```
#include <stdio.h>
```

```
#include <math.h>
```

```
int main(int argc, char **argv)
```

```
{
```

```
    double a=16.0, b;
```

```
    b = sqrt(a);
```

```
    printf("sqrt(16.0)=%f\n",b);
```

```
    return 0;
```

```
}
```

```
[sugar@hussein bit]$ gcc ex5.c
```

```
/tmp/ccwlSS3d.o: In function `main':
```

```
/tmp/ccwlSS3d.o(.text+0x1e): undefined reference to `sqrt'
```

```
collect2: ld returned 1 exit status
```

```
[sugar@hussein bit]$ ls /usr/lib/libm.a
```

```
/usr/lib/libm.a
```

수학 라이브러리 : libm

```
[sugar@hussein bit]$ gcc -lm ex5.c
```

```
[sugar@hussein bit]$ ./a.out
```

```
sqrt(16.0)=4.000000
```

```
[sugar@hussein bit]$ cp /usr/lib/libm.a libmym.a
```

```
[sugar@hussein bit]$ gcc -L. ex5.c -lmym
```

```
[sugar@hussein bit]$ a.out
```

→ -Ldir, -llibname



라이브러리 바인딩

- 정적(static) 라이브러리
 - 라이브러리의 모든 코드를 실행파일에 포함
 - 코드의 크기가 크지만, 실행 속도가 빠르다.
- 동적(dynamic) 라이브러리
 - 기본(default) 설정
 - 공유 라이브러리 사용(Shared Library)
 - 코드의 크기가 작지만, 실행 속도가 느리다.
- 리눅스에서는 두개의 라이브러리가 공존할 경우, 동적 라이브러리를 우선 호출함.



라이브러리 바인딩

- 동적바인딩과 정적바인딩의 비교

```
[sugar@hussein bit]$ gcc hello.c
```

```
[sugar@hussein bit]$ ls -l
```

```
total 16
```

```
-rwxr-xr-x  1 sugar  users    11703 Jun  7 15:14 a.out
```

```
-rw-r--r--  1 sugar  users      72 Jun  7 14:12 hello.c
```

```
[sugar@hussein bit]$ gcc -static hello.c
```

```
[sugar@hussein bit]$ ls -l
```

```
total 896
```

```
-rwxr-xr-x  1 sugar  users   909299 Jun  7 15:14 a.out
```

```
-rw-r--r--  1 sugar  users      72 Jun  7 14:12 hello.c
```

```
[sugar@hussein bit]$
```



표준 헤더 파일 디렉토리

vi ex4.h

```
#define VALUE 1999
```

vi ex4-1.c

```
#include <stdio.h>
```

```
#include "ex4.h"
```

```
int main(int argc, char **argv)
```

```
{
```

```
    printf("Hello World!%d \n",VALUE);
```

```
        return 0;
```

```
}
```

vi ex4-2.c

```
#include <stdio.h>
```

```
#include <ex4.h>
```

```
int main(int argc, char **argv)
```

```
{
```

```
    printf("Hello World!%d \n",VALUE);
```

```
        return 0;
```

```
}
```

```
[sugar@hussein bit]$ gcc ex4-1.c
```

```
[sugar@hussein bit]$ gcc ex4-2.c
```

```
ex4-2.c:2: ex4.h: No such file or directory
```

```
[sugar@hussein bit]$ gcc -I. ex4-2.c (include)
```

```
[sugar@hussein bit]$ ./a.out
```

```
Hello World! 1999
```

-I[include_directory] i의 대문자

-L[library_directory]

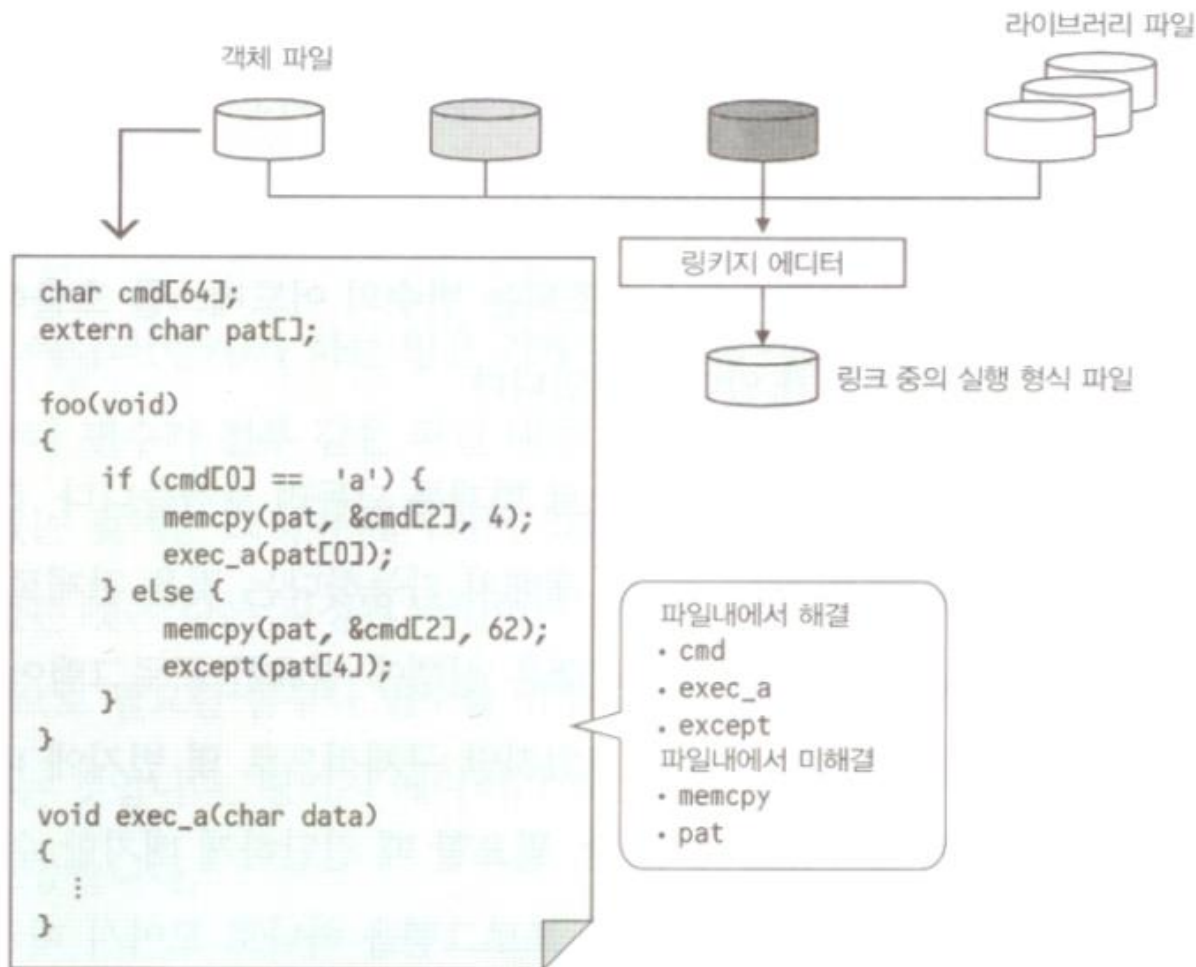
-l[library_name] L의 소문자



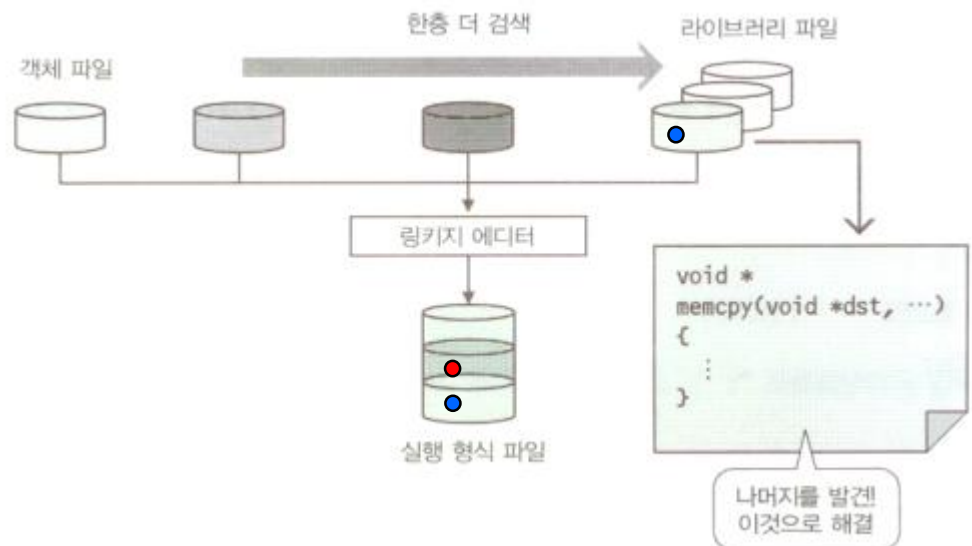
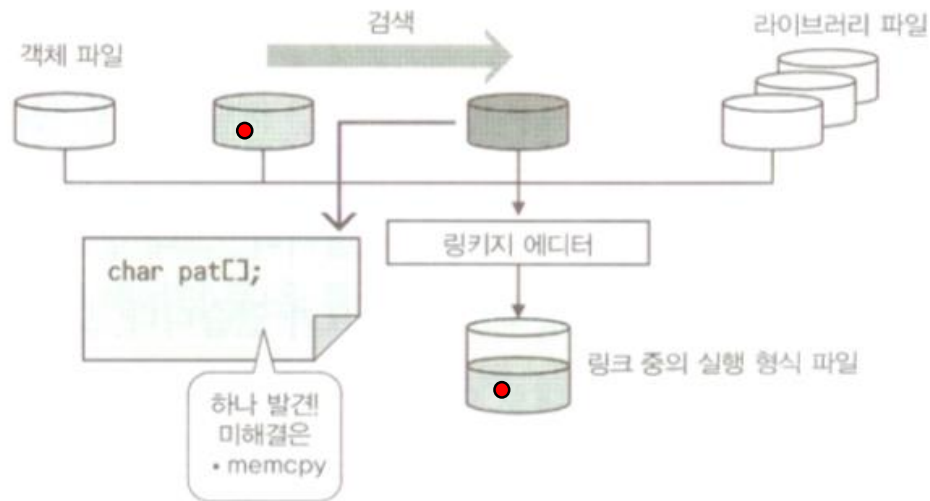
(review) gcc 컴파일러 옵션 요약

| 옵 션 | 설 명 |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -O | . 컴파일 결과로 생성되는 파일의 이름을 명시적으로 지정한다. . -o 옵션 뒤에 파일명을 지정한다. . 일반적으로 소스 파일의 확장자를 제외한 부분을 실행 파일 이름으로 사용 |
| -C | . 목적(Object) 파일을 생성한다. . filename.o 파일이 생성됨 |
| -I (i 대문자) | . 헤더 파일의 디렉토리 위치 를 명시적으로 지정한다. . #include "my.h"와 같은 코드를 #include <my.h>로 쓰고 싶을 때, 컴파일시 옵션으로 지정 (참고) "file.h"는 현재 디렉토리에서 찾음. <file.h> 는 /usr/include에서 찾음 헤더 파일이 저장된 디렉토리의 경로 . gcc -o filename file.c -Idirname |
| -l (L 소문자) | . 컴 파일에 사용되는 라이브러리를 명시적으로 지정한다. . Lib prefix 와 .a 확장자명을 제외하고 사용 |
| -L | . 라이브러리의 디렉토리 위치를 명시적으로 지정한다. . 사용자가 별도의 라이브러리를 생성할 경우, /usr/lib에 삽입하는 것보다 별도의 디렉토리로 관리하는 것이 바람직하다. -l<name> 라이브러리를 링크하도록 . gcc -o filename filename.c -lmy -L. |
| -D | . 매크로를 지정한다. |

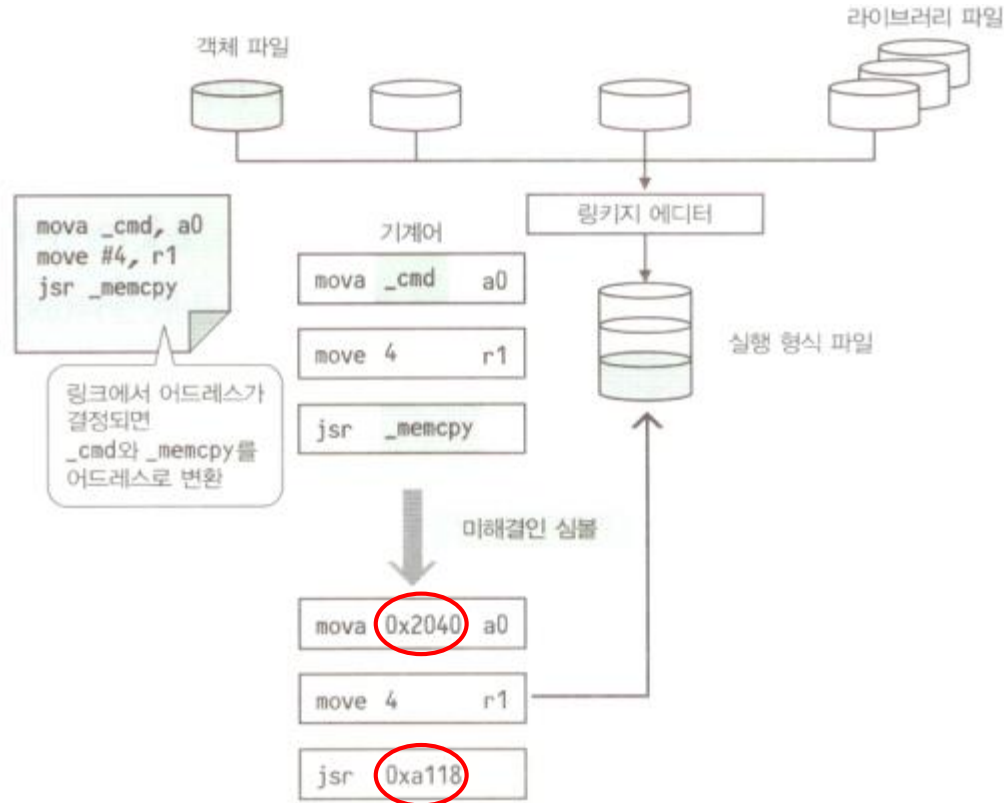
링키지 에디터(링커)의 필요성 및 사용



링키지 에디터(링커)의 필요성 및 사용



링키지 에디터(링커)의 필요성 및 사용



링키지 에디터를 통해 미해결 심볼의 실제 값을 기재한 후, 최종 실행 파일 생성

(참고) 동적 라이브러리 작성

- 동적 라이브러리 (shared library) 생성
 - `cc -fPIC -c source.c`
 - PIC : Position Independent Code
 - `cc -shared -o lib_output_name.so lib1.o, lib2.o`
- 동적 라이브러리 등록
 - Linux Kernel 2.4
 - `/etc/ld.so.conf` 파일에 shared library가 있는 절대 경로 입력
 - Linux Kernel 2.6
 - `/etc/ld.so.conf.d/lib_output_name.conf` 생성
 - 해당 파일에 shared library가 있는 절대 경로 입력
 - `ldconfig` 명령 실행 (운영체제에게 경로 통지)

