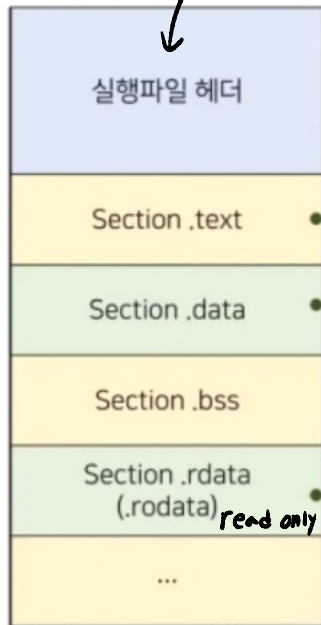


section



실행파일 헤더

Section .text

Section .data

Section .bss

Section .rdata
(.rodata)
read only

...

별 비트 실행파일이고, 언제 만들었기에 관한
기본정보

foo()의 기계어 코드

..... 함수 기계어 코드

..... 전역 변수, static 지역변수 초기값
0x11223344, "abcdefg", 0xAABBCCDD

..... 상수
"hello, world\n"

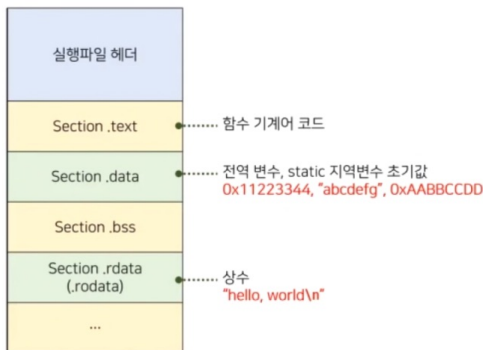
printf 안에 상수문자열이 있다

Windows에서는 PE

Portable Executable file format

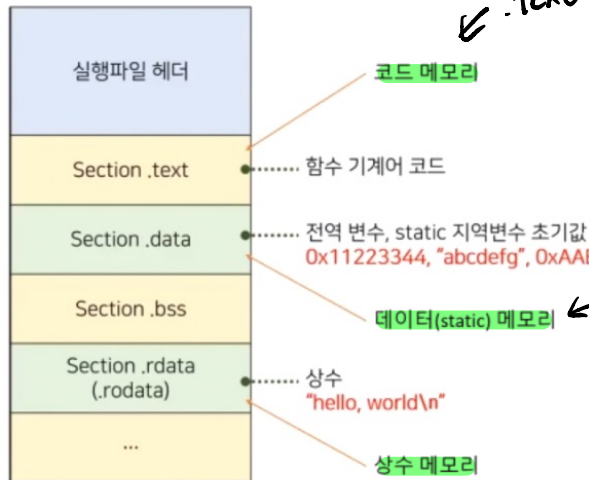
Linux에서는 ELF

Executable Linking format



↓
이 실행파일이 메모리로 올라간다

⑥ 코드 메모리, 데이터 메모리, 상수 메모리는 실행파일이다.

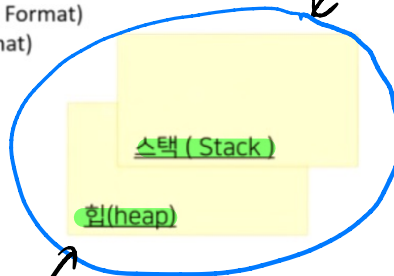


PE (Portable Executable File Format)
ELF (Executable Linking Format)

① .text section이 메모리로 올라가면 그걸 코드 메모리라고 부른다

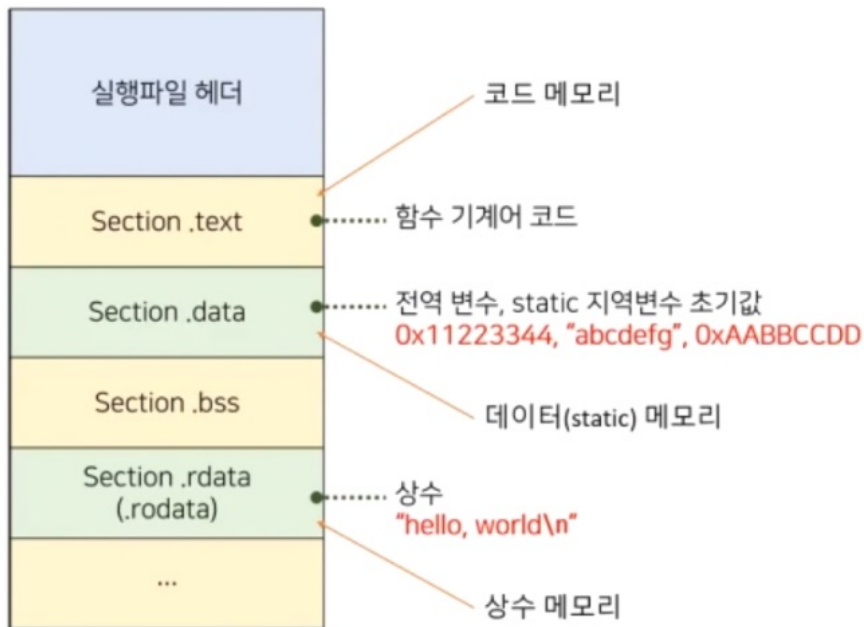
② ← 파괴되지 않는다.
실행중에 크기가 변경되지 않는다

③ 시스템에 의해 할당된다.



④ stack: 지역 변수
foo() 안에 | x는
함수 로컬 끝이면 파괴된다.
stack은 실행할때 크기가
계속 바뀐다.

⑤ malloc에 의한 할당



● PVIEW 로 실행파일 포맷 조사

- <http://wjrdburn.com/software/>
- PVIEW.exe 실행 후 "File -> Open" 메뉴 선택 후 first.exe 선택
- Linux 환경에서는 **readelf** 와 **objdump** 유틸리티로 확인 가능

PE (Portable Executable File Format)

ELF(Executable Linking Format)

스택 (Stack)

힙(heap)

개서

	pfile	Raw Data
IMAGE_DOS_HEADER	00000000 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00	
MS-DOS Stub Program	00000010 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00	
IMAGE_NT_HEADERS	00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
IMAGE_SECTION_HEADER .text	00000030 00 00 00 00 00 00 00 00 00 00 00 80 00 00 00 00	
IMAGE_SECTION_HEADER .data	00000040 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68	
IMAGE_SECTION_HEADER .rdata	00000050 69 73 20 70 72 6F 67 72 61 60 20 63 61 6E 6E 6F	
IMAGE_SECTION_HEADER	00000060 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20	
IMAGE_SECTION_HEADER .bss	00000070 60 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00	
IMAGE_SECTION_HEADER .idata	00000080 50 45 00 00 4C 01 0D 00 42 83 0C 5C 00 70 00 00	
IMAGE_SECTION_HEADER .CRT	00000090 D7 01 00 00 E0 00 07 01 0B 01 02 1C 00 2C 00 00	
IMAGE_SECTION_HEADER .tls	000000A0 00 46 00 00 00 02 00 00 E0 12 00 00 00 10 00 00	
IMAGE_SECTION_HEADER	000000B0 00 40 00 00 00 00 40 00 00 10 00 00 00 02 00 00	
IMAGE_SECTION_HEADER	000000C0 04 00 00 00 01 00 00 00 04 00 00 00 00 00 00 00	
IMAGE_SECTION_HEADER	000000D0 00 10 01 00 00 04 00 00 61 02 01 00 03 00 00 00	
IMAGE_SECTION_HEADER	000000E0 00 00 20 00 00 10 00 00 00 00 10 00 00 10 00 00	
IMAGE_SECTION_HEADER	000000F0 00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00	
SECTION .text	00000100 00 80 00 00 BC 05 00 00 00 00 00 00 00 00 00 00 00	
SECTION .data	00000110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
SECTION .rdata	00000120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
SECTION	00000130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
SECTION .idata	00000140 04 A0 00 00 18 00 00 00 00 00 00 00 00 00 00 00	
SECTION .CRT	00000150 00 00 00 00 00 00 00 00 28 81 00 00 D8 00 00 00	
SECTION .tls	00000160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
SECTION	00000170 00 00 00 00 00 00 00 00 2E 74 65 78 74 00 00 00	
SECTION	00000180 84 2B 00 00 00 10 00 00 00 2C 00 00 00 04 00 00	

실행파열과 메모

○ PEView 로 실행

• <http://wjradb>

• PEView.exe

> preview a.exe

>

	pFile	Raw Data
IMAGE_SECTION_HEADER .text	00000400 83 EC 1C 8B 44 24 20 8B 00 8B 00 3D 91 00 00 C0	
IMAGE_SECTION_HEADER .data	00000410 77 4E 3D 8D 00 00 C0 73 60 3D 05 00 00 C0 0F 85	
IMAGE_SECTION_HEADER .rdata	00000420 CC 00 00 00 C7 44 24 04 00 00 00 00 C7 04 24 0B	
IMAGE_SECTION_HEADER	00000430 00 00 00 E8 F0 29 00 00 83 F8 01 0F 84 48 01 00	
IMAGE_SECTION_HEADER .bss	00000440 00 85 C0 0F 85 E7 00 00 00 8D B4 26 00 00 00 00	
IMAGE_SECTION_HEADER .idata	00000450 31 C0 83 C4 1C C2 04 00 90 8D B4 26 00 00 00 00	
IMAGE_SECTION_HEADER .CRT	00000460 3D 94 00 00 C0 74 49 3D 96 00 00 C0 0F 84 89 00	
IMAGE_SECTION_HEADER .tls	00000470 00 00 3D 93 00 00 C0 75 D7 C7 44 24 04 00 00 00	
IMAGE_SECTION_HEADER	00000480 00 C7 04 24 08 00 00 00 E8 9B 29 00 00 83 F8 01	
IMAGE_SECTION_HEADER	00000490 0F 84 AD 00 00 00 85 C0 74 B6 C7 04 24 08 00 00	
IMAGE_SECTION_HEADER	000004A0 00 FF D0 B8 FF FF FF FF E8 AB 8D B6 00 00 00 00	
IMAGE_SECTION_HEADER	000004B0 C7 44 24 04 00 00 00 00 C7 04 24 08 00 00 E8	
IMAGE_SECTION_HEADER	000004C0 64 29 00 00 83 F8 01 75 CD C7 44 24 04 01 00 00	
SECTION .text	000004D0 00 C7 04 24 08 00 00 00 E8 4B 29 00 00 B8 FF FF	
SECTION .data	000004E0 FF FF E9 6B FF FF FF 89 F6 8D BC 27 00 00 00 00	
SECTION .rdata	000004F0 3D 10 00 00 C0 0F 85 55 FF FF FF C7 44 24 04 00	
SECTION	00000500 00 00 00 C7 04 24 04 00 00 00 E8 19 29 00 00 83	
SECTION .idata	00000510 F8 01 74 59 85 C0 0F 84 34 FF FF FF C7 04 24 04	
SECTION .CRT	00000520 00 00 00 FF D0 B8 FF FF FF FF E9 23 FF FF FF 90	
SECTION .tls	00000530 C7 04 24 0B 00 00 00 FF D0 B8 FF FF FF FF E9 0F	
SECTION	00000540 FF FF FF C7 44 24 04 01 00 00 00 C7 04 24 08 00	
SECTION	00000550 00 00 E8 D1 28 00 00 C7 04 24 00 00 00 00 E8 BD	
SECTION	00000560 0E 00 00 B8 FF FF FF FF E9 E5 FE FF FF C7 44 24	
SECTION	00000570 04 01 00 00 00 C7 04 24 04 00 00 00 E8 A7 28 00	
SECTION	00000580 00 83 C8 FF E9 C9 FE FF FF C7 44 24 04 01 00 00	

text 영역

data 영역

	pFile	Raw Data
IMAGE_SECTION_HEADER .text	00003000 00 00 00 00 44 33 22 11 61 62 63 64 65 66 67 00	
IMAGE_SECTION_HEADER .data	00003010 DD CC BB AA 02 00 00 00 FD FF FF FF 00 40 00 00	
IMAGE_SECTION_HEADER .rdata	00003020 80 3B 40 00 FF FF FF FF 00 00 00 00 00 00 00 00	
IMAGE_SECTION_HEADER .bss	00003030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
IMAGE_SECTION_HEADER .idata	00003040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
IMAGE_SECTION_HEADER .CRT	00003050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
IMAGE_SECTION_HEADER .tls	00003060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
IMAGE_SECTION_HEADER	00003070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
IMAGE_SECTION_HEADER	00003080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
IMAGE_SECTION_HEADER	00003090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
IMAGE_SECTION_HEADER	000030A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
IMAGE_SECTION_HEADER	000030B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
SECTION .text	000030C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
SECTION .data	000030D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
SECTION .rdata	000030E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
SECTION	000030F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
SECTION .idata	00003100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
SECTION .CRT	00003110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
SECTION .tls	00003120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
SECTION	00003130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
SECTION	00003140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
SECTION	00003150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
SECTION	00003160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
SECTION	00003170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	
SECTION	00003180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	

```
int g1 = 0x11223344;
char g2[] = "abcdefg";
```

```
void foo(void)
{
```

```
    static int sx = 0xAABBCCDD;
```

	pFile	Raw Data	Value
IMAGE_SECT	00003000	00 00 00 00 44 33 22 11 61 62 63 64 65 66 67 00D3".abcdefg
IMAGE_SECT	00003010	DD CC BB AA 02 00 00 00 FD FF FF FF 00 40 00 000.

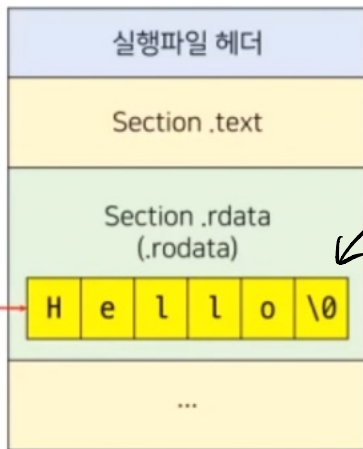
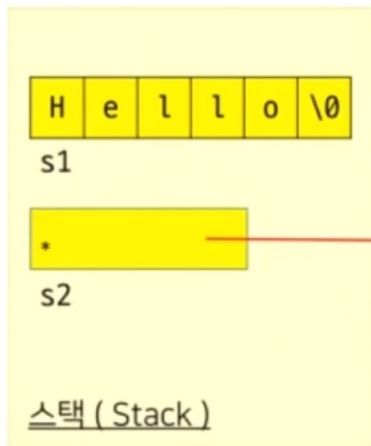
rdata 영역

	Raw Data	Value
IMAGE_SECTION_HEADER	69 62 67 63 63 5F 73 5F 64 77 32 2D 31 2E 64	libgcc_s_dw2-1.d
IMAGE_SECTION_HEADER	6C 00 5F 5F 72 65 67 69 73 74 65 72 5F 66 72	ll.__register_fr
IMAGE_SECTION_HEADER	6D 65 5F 69 6E 66 6F 00 5F 5F 64 65 72 65 67	ame_info.__dereg
IMAGE_SECTION_HEADER	73 74 65 72 5F 66 72 61 6D 65 5F 69 6E 66 6F	ister_frame_info
IMAGE_SECTION_HEADER	6C 69 62 67 63 6A 2D 31 36 2E 64 6C 6C 00 5F	.libgcj-16.dll._
IMAGE_SECTION_HEADER	76 5F 52 65 67 69 73 74 65 72 43 6C 61 73 73	Jv_RegisterClass
IMAGE_SECTION_HEADER	73 00 00 68 65 6C 6C 6F 20 77 6F 72 6C 64 00	es..hello world.
IMAGE_SECTION_HEADER	1A 40 00 4D 69 6E 67 77 20 72 75 6E 74 69 6D	..@.Mingw runtim
IMAGE_SECTION_HEADER	20 66 61 69 6C 75 72 65 3A 0A 00 20 20 56 69	e failure... Vi
IMAGE_SECTION_HEADER	74 75 61 6C 51 75 65 72 79 20 66 61 69 6C 65	rtualQuery faile
IMAGE_SECTION_HEADER	20 66 6F 72 20 25 64 20 62 79 74 65 73 20 61	d for %d bytes a
IMAGE_SECTION_HEADER	20 61 64 64 72 65 73 73 20 25 70 00 00 00 00	t address %p....
SECTION .text	20 55 6E 6B 6E 6F 77 6E 20 70 73 65 75 64 6F	Unknown pseudo
SECTION .data	72 65 6C 6F 63 61 74 69 6F 6E 20 70 72 6F 74	relocation prot
SECTION .rdata	63 6F 6C 20 76 65 72 73 69 6F 6E 20 25 64 2E	ocol version %d.
SECTION	00 00 00 20 20 55 6E 6B 6E 6F 77 6E 20 70 73 Unknown ps
SECTION .idata	75 64 6F 20 72 65 6C 6F 63 61 74 69 6F 6E 20	eudo relocation
SECTION .CRT	69 74 20 73 69 7A 65 20 25 64 2E 0A 00 00 00	bit size %d.....
SECTION .tls	00 67 6C 6F 62 2D 31 2E 30 2D 6D 69 6E 67 77	..glob-1.0-mingw
SECTION	32 00 00 00 00 2E 00 00 00 00 47 43 43 3A	32.....GCC:
SECTION	28 47 4E 55 29 20 36 2E 33 2E 30 00 00 00 00	(GNU) 6.3.0....
SECTION	43 43 3A 20 28 47 4E 55 29 20 36 2E 33 2E 30	GCC: (GNU) 6.3.0
SECTION	00 00 00 47 43 43 3A 20 28 4D 69 6E 47 57 2EGCC: (MinGW
SECTION	72 67 20 47 43 43 2D 36 2E 33 2E 30 2D 31 29	org GCC-6.3.0-1)
SECTION	36 2E 33 2E 30 00 00 47 43 43 3A 20 28 47 4E	6.3.0..GCC: (GN

구조가
✓

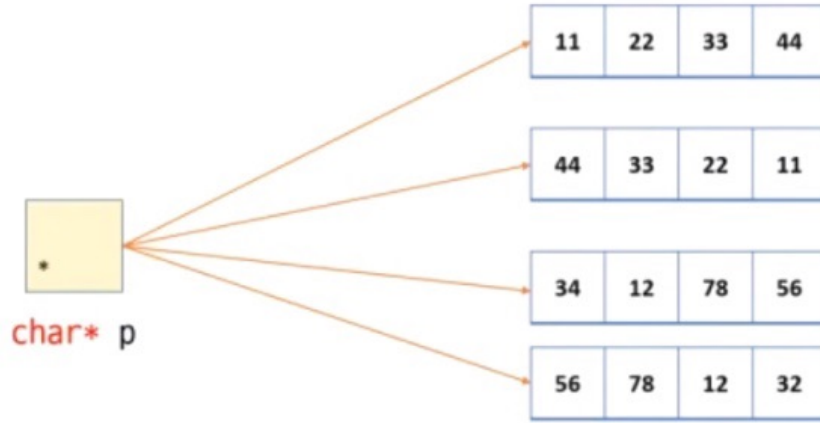
경피워러마다 약간리글 수 있다.

1-14

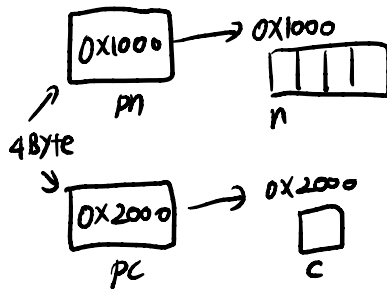
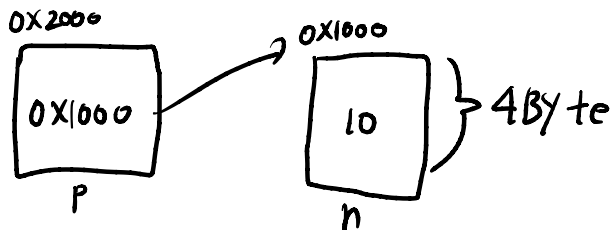


읽을수밖에 없다.

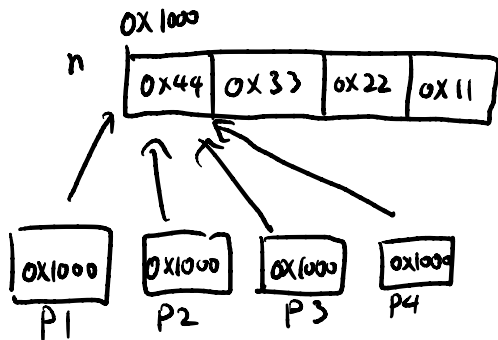
1-16



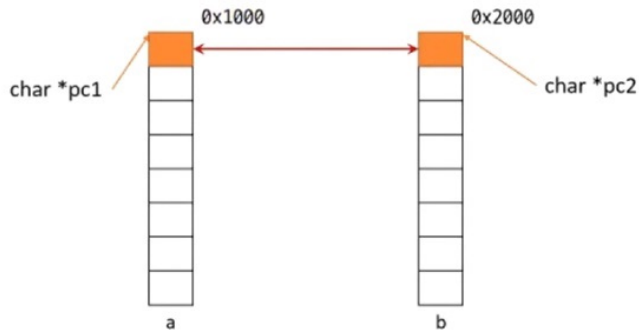
2-1 ①



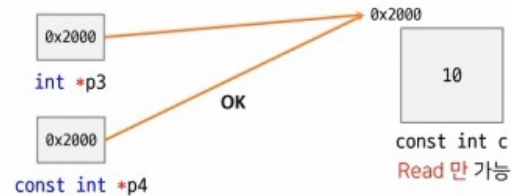
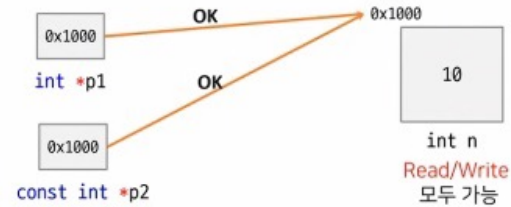
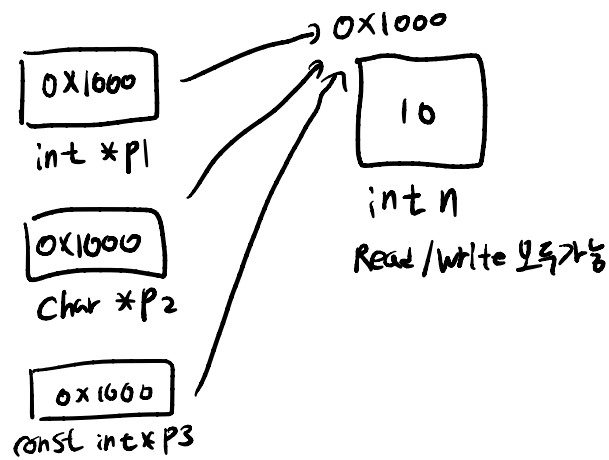
2-2



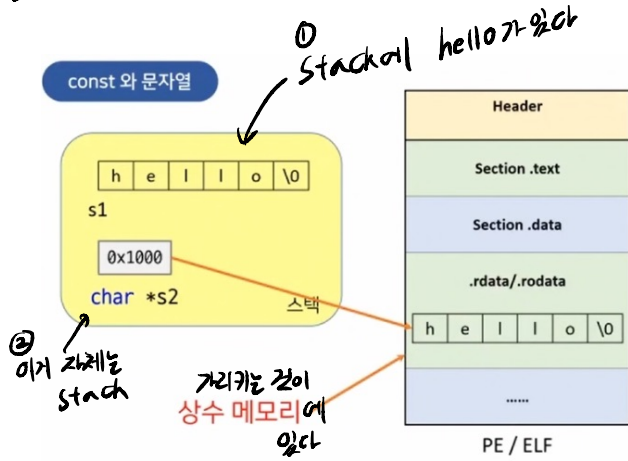
2-4



2-6



2-10



3-1

```
int *p3[3];
```

[] 연산자를 먼저 해석 한다.

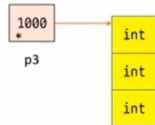
p3는 크기가 3인 배열, 각 요소는 int*

32비트 환경에서 sizeof(p3) => 12



포인터 배열 (Array of pointers)

```
int (*p3)[3];
```



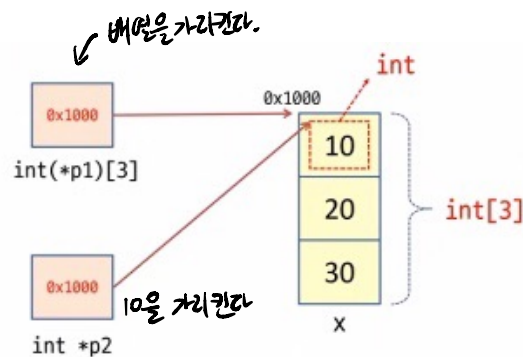
* 연산자를 먼저 해석한다.

p3는 포인터 변수, 배열(int[3]) 을 가리킨다.

32비트 환경에서 sizeof(p3) => 4

배열을 가리키는 포인터 (Pointer to array)

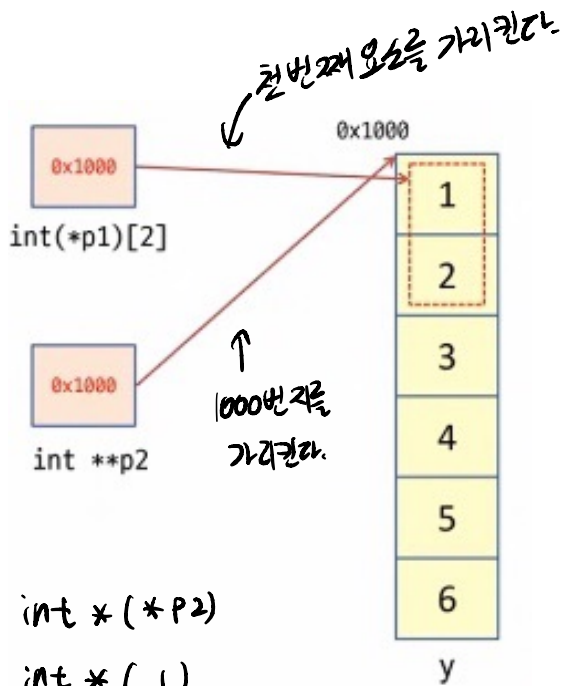
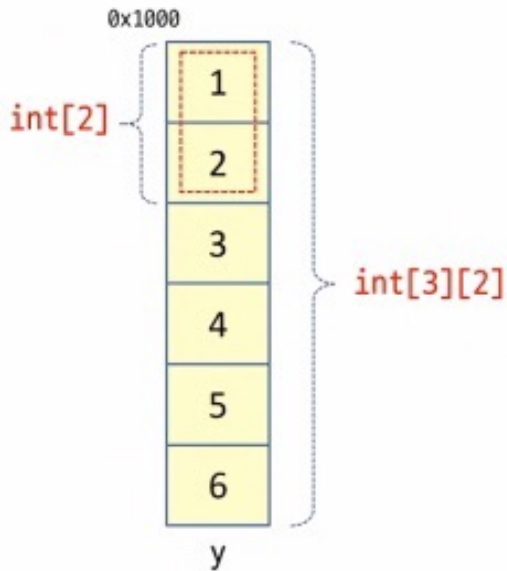
3-2



⦿ "Array to pointer conversion" 문법

배열은 배열의 "첫 번째 요소의 주소"로 암시적 형 변환 된다.

3-3



`int * (*p2)`

`int * ()`

결국 1번리를 가리킴