p217.

5. (1) main.c中 强符号：x, z, main.

弱符号：y, proc1.

proc1.c中 强符号：proc1.

弱符号：x

(2) 调用前：

	0	1	2	3
&z	02	00		
&x	01	01	00	00

调用后：

	0	1	2	3
&z	00	00	F8	BF
&x	00	00	00	00

-1.5：0xBFF8 0000 0000 0000

打印结果为 x=0, z=0

改为 short y=1, z=2. 则 z：0xBFF8.

六、打印结果为：x=0, z=-16392.

(3) 在proc1.c中改成 static double x;

7. ∵ main在m1.c中是强符号 在m2.c中为弱符号.
∴ m2.c中打印main的1, main的时.
打印的是main函数中指令的机器码.

8. ∵ 可读写数据段由可重定位目标文件合并后的.data节和.bss节组成. .data节中总数据长度为0xe8. 所以后面的28字节是.bss中未初始化的全局变量.

9. (1) gcc -static -o p p.o libx.a liby.a p.o
(2) gcc -static -o p p.o libx.a liby.a libx.a
(3) gcc -static -o p p.o libx.a liby.a libx.a libz.a

10. 符号名: swap.
相对于.text节起始位置位移: 7. 在第6行.
按PC相对地址方式重定位.
重定位前内容为 fc ff ff ff
∴ init = -4.
∵ (0x8048386 + 0x12) - (0x8048386 + 7 + (-4)) = 7.
∴ 重定位后. 在位移量 7, 8, 9, a 处为 07 00 00 00.

11.

序号	符号	位移	指令所在行号	重定位类型	重定位前	重定位后
1	bufp1(.bss)	0x8	6w7	R_386_32	0x00000000	0x80496d0
2	buf(.data)	0xc	6w7	R_386_32	0x00000004	0x80495cc
3	bufp0(.data)	0x11	10	R_386_32	0x00000000	0x80495d0
4	bufp0(.data)	0x16	14	R_386_32	0x00000000	0x80495d0
5	bufp1(.bss)	0x21	16	R_386_32	0x00000000	0x80496d0
6	bufp1(.bss)	0x2a	20	R_386_32	0x00000000	0x80496d0

实现 readelf -h main.o 的功能

```
tjy@tjy-virtual-machine:~$ gcc readhead.c -o readhead
tjy@tjy-virtual-machine:~$ ./readhead a.o
ELF header:
Magic:7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
Type:1
Machine:3
Version:1
Entry point address:0x0
Start of program headers:0(bytes into file)
Start of section headers:496(bytes into file)
Flags:0x0
Size of this header:52
Size of program headers:0 (bytes)
Number of program headers:0
Size of section headers:40 (bytes)
Number of section headers:12
Section header string table index:9
tjy@tjy-virtual-machine:~$
```

```c
#include<stdio.h>
#define EI_NIDNET (16)
typedef unsigned short Elf32_Half;
typedef unsigned int Elf32_Word;
typedef unsigned int Elf32_Addr;
typedef unsigned int Elf32_Off;
typedef struct{
  unsigned char e_ident[EI_NIDNET];
  Elf32_Half e_type;
  Elf32_Half e_machine;
  Elf32_Word e_version;
  Elf32_Addr e_entry;
  Elf32_Off e_phoff;
  Elf32_Off e_shoff;
  Elf32_Word e_flags;
  Elf32_Half e_ehsize;
  Elf32_Half e_phentsize;
  Elf32_Half e_phnum;
  Elf32_Half e_shentsize;
  Elf32_Half e_shnum;
  Elf32_Half e_shstrndx;
}Elf32_Ehdr;

int read(FILE* fp, Elf32_Ehdr* ehdr){
  fread(&ehdr->e_ident, 16, 1, fp);
  fread(&ehdr->e_type, 2, 1, fp);
  fread(&ehdr->e_machine, 2, 1, fp);
  fread(&ehdr->e_version, 4, 1, fp);
  fread(&ehdr->e_entry, 4, 1, fp);
  fread(&ehdr->e_phoff, 4, 1, fp);
  fread(&ehdr->e_shoff, 4, 1, fp);
  fread(&ehdr->e_flags, 4, 1, fp);
  fread(&ehdr->e_ehsize, 2, 1, fp);
  fread(&ehdr->e_phentsize, 2, 1, fp);
  fread(&ehdr->e_phnum, 2, 1, fp);
  fread(&ehdr->e_shentsize, 2, 1, fp);
  fread(&ehdr->e_shnum, 2, 1, fp);
  fread(&ehdr->e_shstrndx, 2, 1, fp);
  fclose(fp);
}
```

```c
int out(Elf32_Ehdr* ehdr){
  printf("ELF header:\n");
  printf("Magic:");
  for(int i=0;i<16;i++){
    printf("%02x ", ehdr->e_ident[i]);
  } //Magic+class+Data。。。
  printf("\n");
  printf("Type:");
  printf("%x\n", ehdr->e_type);
  printf("Machine:");
  printf("%x\n", ehdr->e_machine);
  printf("Version:");
  printf("%x\n", ehdr->e_version);
  printf("Entry point address:");
  printf("0x%x\n", ehdr->e_entry);
  printf("Start of program headers:%u(bytes into file)\n", ehdr->e_phoff);
  printf("Start of section headers:%u(bytes into file)\n", ehdr->e_shoff);
  printf("Flags:0x%0x\n", ehdr->e_flags);
  printf("Size of this header:%u\n", ehdr->e_ehsize);
  printf("Size of program headers:%u (bytes)\n", ehdr->e_phentsize);
  printf("Number of program headers:%u\n", ehdr->e_phnum);
  printf("Size of section headers:%u (bytes)\n", ehdr->e_shentsize);
  printf("Number of section headers:%u\n", ehdr->e_shnum);
  printf("Section header string table index:%u\n", ehdr->e_shstrndx);
}

int main(int argc,char* argv[]){
  char* filename=argv[1];
  FILE *fp;
  Elf32_Ehdr ehdr;
  fp=fopen(filename, "r");
  if(fp==NULL){
    printf("fail\n");
    return 1;
  }
  read(fp, &ehdr);
  out(&ehdr);
  return 0;
}
```