H9

1

1)

我直接复制 H2 的解答过来了:

- 用 `man gcc` 来查询, 发现这个参数的含义是

  > Do not search the standard system directories for header files. Only the directories you have specified with -I options (and the directory of the current file, if appropriate) are searched.

- 用中文解释一下也就是, 编译的时候不去搜索头文件的标准系统路径, 这样一来自然就找不到 `stdio.h` 的包含路径, 从而引发编译错误.

一般是 `-E` 的预处理阶段就检查.

2)

检查大小

```
rabbit@DESKTOP-CK1FK5P:~$ gcc -o link-s -static link2.c link1.c
rabbit@DESKTOP-CK1FK5P:~$ gcc -o link link2.c link1.c
rabbit@DESKTOP-CK1FK5P:~$ du -sh link*
32K     link
1.2M    link-s
```

objdump 和 nm 的结果

- link-s: 显然它把所有涉及的函数都静态地加到了文件里, 因此它符号也特别多
  - 好处:
    - 在没有动态库的情况下也能运行
  - 坏处:
    - 占的空间大
- link: 则以动态加载库的方式进行调用, 只有几个关键的符号
  - 好处:
    - 所占空间小
  - 坏处

    - 没有动态库则无法运行

3)

`_start` 是一个程序的起始标签, 链接时当然要找到它. 这是在 `链接` 的阶段发生的(printf那个也是)

4)

64 位

段错误.
原本 buf 位置是一个数组, 其第一个元素值为 100.
现在 `link2.c` 单纯地去引用 `buf` 这个符号, 就会以为 `buf` 是个指针, 其值为 100, 这时候去访问地址为 100 的空间一般来说是访问到了内核态的内存, 因此会报段错误

32 位

使用 `嵌入式系统设计方法` 课程上给的 32位 arm 开发板, 依然得到段错误. 其原因相同.(输出 buf 而非 *buf 即可得到 100)

2

可以得到可执行程序, 运行时也不报错. 但输出结果为 0.

原因是:

- `short` 类型是 2 字节的, 而 32768 * 2 = 2 ** 16, 这意味着它恰好是最高位为1, 低16位为0的值,
- 而我的系统又是小尾端的, 这意味着 `short.c` 文件中的变量 `i` 的范围内都是 0,

因此最终输出结果为 0

3

经检验 (a), (b), (c) 都不会报错.
输出结果是 `258` , 也就是 `0000 0001 0000 0010` , 其中 `0000 0001` 来自 `j` , `0000 0010` 来自 `i`

- 这是由于我的系统上的 gcc 会把 `file1.c` 的 k 和 j 连续分配,
- 且和上题一样, 小尾端
- `short` 又是 2 字节, 而 `char` 是 1 字节

因此在 `file2.c` 的 `k` 就把 `file1.c` 的 `j` 和 `k` 合起来看了.

所以就输出了 `258`