U201614702 CS1607 樊俊超

PA2 必答题

Q1：

- 编译与链接 在 `nemu/include/cpu/rtl.h` 中，你会看到由 `static inline` 开头定义的各种RTL指令函数. 选择其中一个函数，分别尝试去掉 `static` ，去掉 `inline` 或去掉两者，然后重新进行编译，你可能会看到发生错误. 请分别解释为什么这些错误会发生/不发生？你有办法证明你的想法吗？

A1：

不加-werror 时同时去掉时会报错，单个去掉不影响。加上-werror 时会在单独去掉 static 时报错（报错为警告：定义但未使用过）。inline 跟宏定义类似，要想被多个文件使用，只有放在头文件中实现，加上 static 是在强调作用域为本文件空间。同时去掉 static inline 相当于在多个引用了该函数同时又引用了 rtl.h 文件的文件中重复定义了一个函数。

Q2：

- 编译与链接
  1. 在 `nemu/include/common.h` 中添加一行 `volatile static int dummy;` 然后重新编译NEMU. 请问重新编译后的NEMU含有多少个 `dummy` 变量的实体？你是如何得到这个结果的？
  2. 添加上题中的代码后，再在 `nemu/include/debug.h` 中添加一行 `volatile static int dummy;` 然后重新编译NEMU. 请问此时的NEMU含有多少个 `dummy` 变量的实体？与上题中 `dummy` 变量实体数目进行比较，并解释本题的结果.
  3. 修改添加的代码，为两处 `dummy` 变量进行初始化: `volatile static int dummy = 0;` 然后重新编译NEMU. 你发现了什么问题？为什么之前没有出现这样的问题？(回答完本题后可以删除添加的代码.)

A2：

1、一个

2、一个，因为是在头文件中定义的，同时 debug.h 引用了 common.h 文件，debug.h 中的定义被合并。

3、发现了重复定义的问题，这是因为在预编译时，debug.h 中相当于重复了 dummy=0 这一操作。

Q3:

- 了解Makefile 请描述你在 `nemu/` 目录下敲入 `make` 后，`make` 程序如何组织.c和.h文件，最终生成可执行文件 `nemu/build/nemu` .(这个问题包括两个方面: `Makefile` 的工作方式和编译链接的过程.) 关于 `Makefile` 工作方式的提示:
  - `Makefile` 中使用了变量，包含文件等特性
  - `Makefile` 运用并重写了一些implicit rules
  - 在 `man make` 中搜索 `-n` 选项，也许会对你有帮助
  - RTFM

A3:make 执行过程：

1、读取当前目录下的 GNUmakefile/makefile/Makefile 文件；

2、一次读取 makefile 文件中使用``include``包含的文件；

3、查找重建所有已读取的 makefile 文件的规则（如果存在一个目标是当前读取的某一个 makefile 文件，则执行此规则重建此 makefile 文件，完成以后从第一步开始重新执行）；

4、初始化变量值并展开那些需要立即展开的变量和函数并根据预设条件确定执行分支；

5、根据“终极目标”以及其他目标的依赖关系建立依赖关系链表；

6、执行除“终极目标”以外的所有的目标的规则（规则中如果依赖文件中任一个文件的时间戳比目标文件新，则使用规则所定义的命令重建目标文件）；

7、执行“终极目标”所在的规则。