一. Ucore.img 的生成过程：

槽点 1：这个 makefile 对新手也太太太太太不友好了

实验代码给的 makefile 中包含了大量系统函数，编写的风格和手法也很高级。但这样的东西对我这个新手来说过于残忍，还要让我尽可能弄懂每一句在干嘛。所以只好碰到不会的看不懂的挨着百度，比如这些它自定义的函数：

```
include tools/function.mk

listf_cc = $(call listf,$(1),$(CTYPE))

# for cc
add_files_cc = $(call add_files,$(1),$(CC),$(CFLAGS) $(3),$(2),$(4))
create_target_cc = $(call create_target,$(1),$(2),$(3),$(CC),$(CFLAGS))

# for hostcc
add_files_host = $(call add_files,$(1),$(HOSTCC),$(HOSTCFLAGS),$(2),$(3))
create_target_host = $(call create_target,$(1),$(2),$(3),$(HOSTCC),$(HOSTCFLAGS))

cgtype = $(patsubst %.$(2),%.$(3),$(1))
objfile = $(call toobj,$(1))
asmfile = $(call cgtype,$(call toobj,$(1)),o,asm)
outfile = $(call cgtype,$(call toobj,$(1)),o,out)
symfile = $(call cgtype,$(call toobj,$(1)),o,sym)
```

很多东西真的头一次见，挨着去百度这些函数的用法，listf 之类的，费了半天功夫终于明白此处调用 listf 的返回结果为 libs 目录下的所有.c 和.S 文件。由于 lab1 的 libs 目录下只有.h 和.c 文件，因此最终返回.c 文件。

虽然本校课程 PPT 上说只写出槽点和踩到的坑就行，但还是稍微简述一下 ucore.img 的完整生成过程吧（行数指 makefile 文件中的行数）：

1. 编译 libs 和 kern 目录下所有的.c 和.S 文件，生成.o 文件，并链接得到 bin/kernel 文件（117-151 行）
2. 编译 boot 目录下所有的.c 和.S 文件，生成.o 文件，并链接得到 bin/bootblock.out 文件（155-168 行）
3. 编译 tools/sign.c 文件，得到 bin/sign 文件（172-174 行）
4. 利用 bin/sign 工具将 bin/bootblock.out 文件转化为 512 字节的 bin/bootblock 文件，并将 bin/bootblock 的最后两个字节设置为 0x55AA（166 行，0x55AA 的设置需参考 sign.c 文件）
5. 为 bin/ucore.img 分配 5000MB 的内存空间，并将 bin/bootblock 复制到 bin/ucore.img 的第一个 block，紧接着将 bin/kernel 复制到 bin/ucore.img 第二个 block 开始的位置（182 行）

关于更多的细节我以中文注释的形式加入了 makefile 文件中，一并提交到 github 上

二、一个被系统认为是符合规范的硬盘主引导扇区的特征是什么？

槽点 2：bootblock 和主引导扇区的区别？

看完了 makefile 文件并没有发现啥是主引导扇区，直觉感觉和 bootblock 有关，有道翻译 bootblock 的结果是“刷鞋匠”，百度了一下告诉我

bootblock 编辑 讨论 上传视频

本词条缺少概述图，补充相关内容使词条更完整，还能快速升级，赶紧来编辑吧！

BOOTBLOCK是BIOS中一段特定的区域，包含有用于引导的最小指令集。

外文名	bootblock	类　型	计算机学
定　义	BIOS中一段特定的区域	隶属系统	BIOS

主板上的引导块Boot Block

BOOTBLOCK是BIOS中一段特定的区域，包含有用于引导的最小指令集，正常的BIOS升级操作不能消除这段信息。如果BIOS升级失败，可以利用BOOTBLOK来重新恢复，具体步骤如下：

然后又百度主引导扇区：

简介

主引导扇区位于整个硬盘的0磁头0柱面1扇区，包括硬盘主引导记录MBR（Master Boot Record）和分区表DPT（Disk Partition Table）。其中主引导记录的作用就是检查分区表是否正确以及确定哪个分区为引导分区，并在程序结束时把该分区的启动程序，也就是操作系统引导扇区调入内存加以执行。

MBR结构

在总共512字节的MBR扇区中，由四部分组成：

引导程序

引导程序在0号扇区的开始位置，共占用440字节。

Windows磁盘签名

Windows磁盘签名占用引导程序后的4个字节，是windows系统对硬盘初始化时写入的一个硬盘标签。

分区表

分区表占用64字节，是MBR中非常重要的一个结构。

结束标志

扇区最后两个字节“55AA”是MBR的结束标志。

完全没整明白到底是啥。但是注意到一个细节，百度说 55AA 是 MBR 结束标志，而在分析 makefile 时已经发现 bootblock 的最后两个字节就是 0x55AA，所以盲猜这俩应该说的就是一回事吧。为啥不直接告诉我这个就是在说“刷鞋匠”？
那么关于这个 bootblock(MBR?)的特性应该就是大小为 512 字节以及最后两个字节是 55AA。

```
buf[510] = 0x55;
buf[511] = 0xAA;
FILE *ofp = fopen(argv[2], "wb+");
size = fwrite(buf, 1, 512, ofp);
```

sign.c 文件中的设置代码片段