Lab 0 实验过程的一些知识点的记录
ls 命令：列出文件
cd 命令：切换到某文件夹
pwd 命令：显示当前目录
mkdir 命令：创建目录
gcc-v:查看 gcc 版本
gdb:进入 gdb,输入 quit，退出

到 labcodes_answer/lab1_result 文件夹下，先 make clean,然后 make 就可以重新编译了。
在这时,bin 目录下会有 kernel 这样一个文件，利用 file bin/kernel 命令查看该文件的属性，可以发现该文件是一个可执行文件，它可以把我们的 bootloader 给加载进去。而 bootloader 在 bin 目录下的 ucore.img 镜像文件之中，这个磁盘镜像包含了 bootloader 以及 ucore 的 kernel。

执行 make qemu 就执行了刚才的镜像文件，通过串口进行输出到屏幕上。按 ctrl+c 就可以终止刚才程序的执行。

以上是用字符方式执行的，我们也可以用图形化的界面，Eclipse CDT 环境
先 Clean project，再 build project，可以看到这个项目就编译完成了

对该项目进行 Debug 的方法：点击 debug 下的 debug configuration，到 zylin native 那项，debugger 的 stop on startup 项关掉
在 commands 栏 initialize commands 里敲入命令：
target remote :1234　　//用于完成对 qemu 的连接
file /home/moocos/moocos/ucore_lab/labcodes_answer/lab1_result/obj/bootblock.o
//加载 bootloader 符号，即 bootloader 的主体执行程序
break bootmain //在 bootmain 所对应的虚拟地址里面设置了一个断点
#file /home/moocos/moocos/ucore_lab/labcodes_answer/lab1_result/bin/kernel
#break kern_init
#break print_kerninfo

Run commands 里填：continue

在 debugger 的 gdb debugger 中键入 gdb(原来是 arm-elf-gdb)

点击带个红点的运行按钮，下面有个 External tools configuration:
location 里填的是：/usr/bin/make
Arguments 处填的是：gdb
Working directory 处填的是：/home/moocos/moocos/ucore_lab/labcodes_answer/lab1_result

点击红点运行下的 lab1_ans，可以看到与 qemu 建立了联系。
然后双击 bootmain.c 文件的 88 行，设置断点
然后，首次对该项目 debug，要选择 debug 按钮下的 debug configuration 中的 debug 以后该项目就自动出现在 debug 列表中了，这时就对该项目进行 debug 的操作了。

然后点击 Step over 按钮开始 Debug

这里注意，如果我们要调试 ucore,为此我们需要把 ucore 的符号信息加载进来，这时我们进入 console 里面键入 file bin/kernel
这里面有一个起止函数 kern_init，这是 ucore 最开始跑的一个 c 程序的函数，我们在 console 中设置中断：
break kern_init
这时将提示：
Breakpoint 3 at 0x100000: file kern/init/init.c, line 17.
这时点击按钮 step return,然后点击 step over 以继续运行程序，可以发现此时便跑完了 bootloader 的工作，同时把控制权交给了 ucore 另外一个程序：程序跳到了刚才设置的 kern/init/init.c 文件的 kern_init 函数处

此时再尝试单步运行几次，然后按 resume，使得程序正常运行。
输入 meld lab1_result/ lab2_result/ 可以比较两次实验中代码的区别。

可以用 understand 工具更好的分析代码