MiniOS C语言Loader说明文档

作者：孙东

版本：V0.1

时间：2023-4-27

一、C语言Loader简介

从MiniOS1.3.19开始loader程序就由原来的汇编实现改成了大部分由C语言实现，C语言的loader与汇编实现的loader相比，功能也更强大。

C语言Loader将如下功能进行了C语言的实现：

1. 输出函数。用C语言实现了`lprintf()`函数，loader的所有输出统一使用该函数，除此之外还实现了一个清屏函数，`clear_screen()`.
2. 启动分页。获取物理可用内存块是使用汇编语言调用int 15 Bios中断实现的，在获取内存块之后页表的建立是使用C语言实现的。
3. 读硬盘。C语言Loader与之前汇编语言的Loader相比最大的特点在于C语言Loader处于实模式下的代码较少，其中读硬盘操作在汇编语言Loader中是通过调用Bios中断实现的，该部分代码处于实模式，在C语言Loader中读硬盘操作针对IDE硬盘和SATA硬盘有个各自的驱动程序，不再使用实模式下的Bios中断。
4. 去启动分区加载kernel。加载Kernel操作是Loader最重要的功能，C语言Loader使用文件系统读kernel.bin文件这步操作也进行了C语言化，与汇编语言的Loader实现相比，C语言的Loader能支持fat32和orangefs两个类型的文件系统，并且能自动识别当前分区的文件系统类型，并调用对应文件系统的初始化函数和read函数。C语言loader中还对kernel.bin的bss段进行了清零操作。

二、C语言Loader实现的关键部分说明

2.1 启动分页实现中需要注意的点

具体页表的建立与汇编写的Loader没什么太大的区别，无非是建立页目录、建立页表、将页目录基地址赋值给CR3寄存器、将CR0寄存器31位标志位置为1。但是分页过程中的一些细节需要稍微注意下：

1. 页目录的放置在物理内存**0x400000**处，页表放置在**0x401000**处。这两个内存位置需要注意下，后续改动时避免修改这几处内存。
2. 分页时映射使用的是双映射，可用内存区域大小最大为32M。Loader的所有代码是运行在低地址处的，kernel的代码是运行在高地址处的，若仅仅建立高地址处的页表，那么在启动页表后Loader剩余未执行的代码将无法继续执行下去。因此采用了双映射方案，将物理内存的`0-32M`地址映射到线地址的`0-32M`和`3G-3G+32M`,当可用物理内存不足32M时会以实际内存的大小进行映射，当物理内存可用大小大于32M时将只映射物理地址的0-32M。Loader中建立的页表只在Loader和kernel最开始处使用，kernel开始时会重新建立页表，Loader建立的页表就无效了，因此Loader并未将所有可用的物理内存全映射一遍，32M的空间足够使用了。

2.2 读硬盘实现需要注意的点

1. 当前Loader暂时无法区别当前启动的硬盘是IDE设备还是SATA设备，也无法区分当前启动的硬盘是第几个IDE/SATA硬盘，因此C语言的Loader采取了以下执行逻辑：若SATA设备存在则将AHCI的0号端口的SATA设备视为启动硬盘，从中读扇区。若SATA设备不存在则使用第一个IDE设备作为启动硬盘。
2. 调用读硬盘函数（`readsect`等）之前应当先执行`AHCI_init`,扫描一遍SATA设备，从而决定是读IDE还是SATA。
3. 读硬盘函数参数的说明。`readsects(void* dst, u32 offset,u32 count)`为连续读多个扇区的函数，`dst`表示内存地址，`offset`表示物理扇区号（从0开始计数），`count`表示读取的连续扇区的数目，建议一次读取不要超过128个扇区，否则可能会出现读取失败的情况。
4. 读SATA设备的驱动程序没有使用中断方式去读SATA设备，而是使用的是阻塞的方式，等待SATA设备读完之后才执行下一步。
5. AHCI驱动程序运行时需要的一些内存空间是从`0x600000`处开始分配的，后面Loader如果添加新功能需要使用内存的话应当注意这个区域。

2.3 加载kernel实现中需要注意的一点

1. 初始化文件系统之前应当调用`find_act_part`，该函数主要是遍历硬盘分区表并找到启动分区，当有多个分区有活跃标志时则返回第一个带有活跃标志的分区。该函数还支持将扩展分区中的逻辑分区作为启动分区，但是os_boot暂时支持不了这一点，所以整个MiniOS还是无法从扩展分区启动的。
2. 初始化文件系统会识别去启动分区的0号或者1号扇区读取相应的文件系统标志，用于区分当前是FAT32还是Orangefs。
3. 去文件系统中查找kernel.bin这个文件时，不同文件系统存储的文件名是不一样的。kernel.bin文件FAT32中存储的文件名为`"KERNEL BIN"`,Orangefs文件系统中存储的文件名为`"kernel.bin"`,因此使用FAT32文件系统查找文件时需要注意这一点。C语言Loader中提供了`fat32_uppercase_filename(char*src,char*dst)`用以转换文件名，具体来说是将小写转为大写，将“.”替换为“ ”（空格）。
4. 在文件系统中加载到ELF格式的kernel.bin文件之后，会根据ELF头记录的信息将对应的段从文件中拷贝到内存相应的的位置。
5. 加载完kernel之后会根据elf头信息进行初始化bss段，此处确认bss段的方法在于比较每个section name的名称是不是`".bss"`，这个地方使用目前的编译器来说是没有问题的，不知道换编译器后bss段的名称是否还是`".bss"`。