

## 1.3.21版本之后启动时的一些说明

版本: V0.1

最后修改时间: 2023-7-10

### 一、Makefile中的一些说明

```
41  #added by sundong 2023.3.21
42  #用于区分是使用grub chainloader
43  #可选值为 true false
44  USING_GRUB_CHAINLOADER = false
45  #选择启动分区的文件系统格式, 目前仅支持fat32和orangesfs
46  BOOT_PART_FS_TYPE= fat32
47
```

`BOOT_PART_FS_TYPE` 变量用于表示启动分区的文件系统类型, 共有两种类型可选分别是 `fat32` 和 `orangesfs`, 默认是 `fat32`.

`USING_GRUB_CHAINLOADER` 用于表示是否使用GRUB引导程序, 默认是`false`, 不使用`grub`引导。选择使用`grub`引导时请在linux上提前安装 `grub-install`程序。

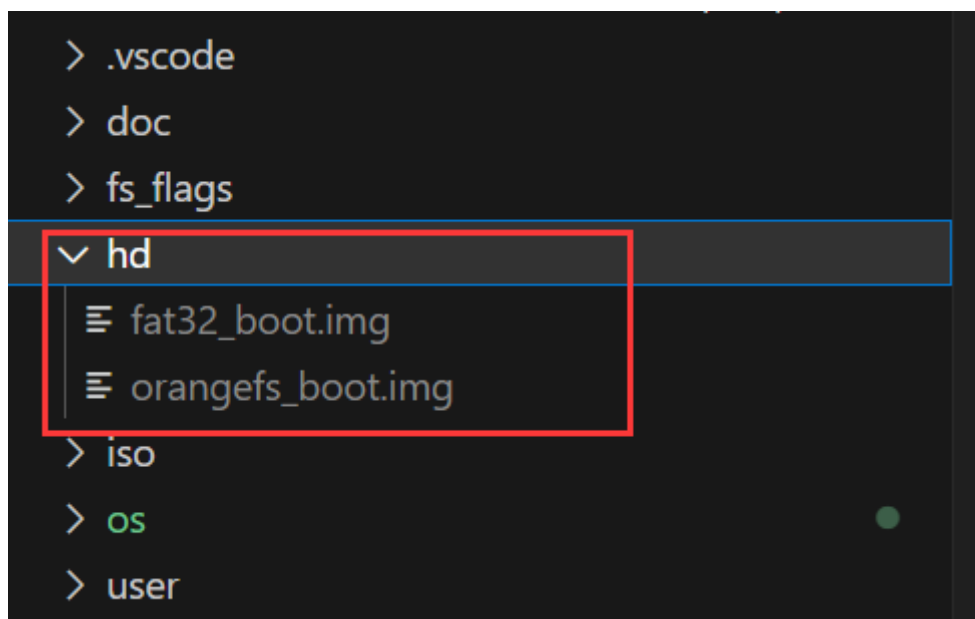
### 二、关于启动镜像的一些说明

目前启动镜像单独放在了一个仓库中仓库地址是: [http://202.117.249.26:2080/minios/minios\\_image](http://202.117.249.26:2080/minios/minios_image)

使用MiniOS v1.3.21及以后的版本时, 请将下图中的两个镜像下载下来, 放在Minios代码的hd文件夹下。

添加了V1.3.21和1.3.22版本中使用的镜像 Sun Dong authored 12 minutes ago			c228484d
main	minios_image / V1.3.21及以后的版本	History	Find file
			Web IDE
			Clone
Name	Last commit	Last update	
..			
fat32_boot.img	添加了V1.3.21和1.3.22版本中使用的镜像	13 minutes ago	
orangesfs_boot.img	添加了V1.3.21和1.3.22版本中使用的镜像	13 minutes ago	

镜像的放置的目录如下:



`fat32_boot.img`的分区格式如下:

Device	Boot	Start	End	Sectors	Size	Id	Type
hd/fat32_boot.img1	*	2048	4095	2048	1M	83	Linux
hd/fat32_boot.img2		4096	106495	102400	50M	83	Linux
hd/fat32_boot.img3		106496	204799	98304	48M	5	Extended
hd/fat32_boot.img5		108544	204799	96256	47M	83	Linux

orangeefs\_boot.img的分区格式如下:

Device	Boot	Start	End	Sectors	Size	Id	Type
hd/orangeefs_boot.img1		2048	4095	2048	1M	83	Linux
hd/orangeefs_boot.img2	*	4096	106495	102400	50M	83	Linux
hd/orangeefs_boot.img3		106496	204799	98304	48M	5	Extended
hd/orangeefs_boot.img5		108544	204799	96256	47M	83	Linux

这两个启动镜像的区别在于启动分区标志不同。

镜像中分区1和分区2为主分区。分区1格式化成为fat32文件系统，分区2格式化为OrangeFS文件系统。这两个分区都可以作为启动分区，可在Makefile中通过设置

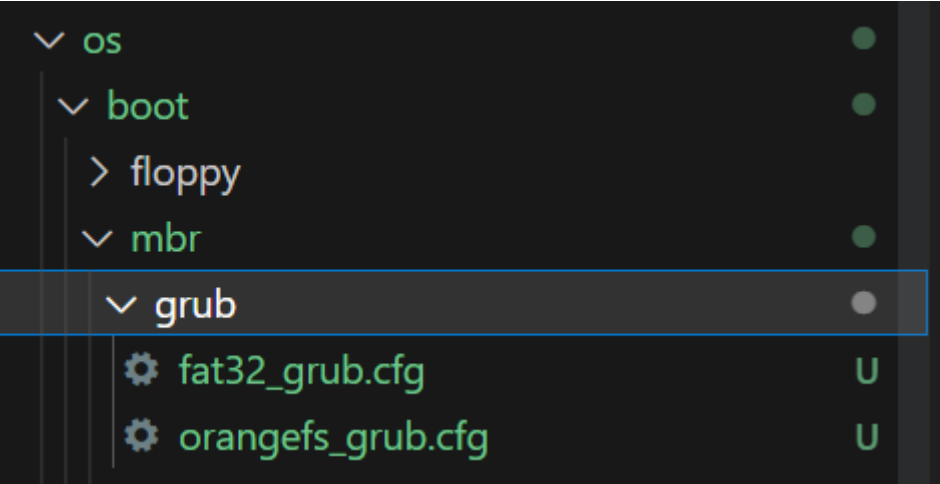
`BOOT_PART_FS_TYPE` 变量来指定启动的分区的文件系统格式来选择不同的启动分区。

分区3为扩展分区，分区5为扩展分区中的一个逻辑分区，格式化成fat32文件系统，用于安装grub。

### 三、关于使用GRUB引导的一些说明

若使用grub引导，则需要在编译之前在Linux上安装grub-install。（默认情况下是不使用grub引导的）

1.3.21版本及以后的版本中，使用grub引导时不再需要根据启动分区的文件系统类型来修改grub.cfg文件的内容。现在已经为不同的启动分区设置了对应的grub配置文件，如下图所示



编译时Makefile文件会复制对应.cfg文件到镜像的grub安装目录中。

### 四、1.3.21版本及以后的版本在MiniOS shell中运行用户程序的注意事项

当启动分区是fat32时（v1.3.21版本及以后的版本默认为fat32），用户程序都放在分区1的fat32文件系统中，分区1在MiniOS中会被挂载到/fat0目录上。因此当启动分区为fat32时，在MiniOS的shell中应输入测试程序的绝对路径，比如 `fat0/ptest1.bin`

```
QEMU
-----Kernel Initialization Begins-----
ABAR: 0xFEBF1000 interrupt: 0x10B
SATA drive found at port :0x0Message queue initialization done.
Initializing root file system... Superblock Address:0xC01DBCAB
-----Processes Begin-----
main:total_mem_size=0x1BDFB00
[TTY #0]
init:toatal_mem_size=1BCDAF4
miniOS:/ $ fat0/pptest1.bin
main pid = 18 [pthread success:fat0/pptest1.bin]
test1 pid = 19 [pthread success:fat0/pptest1.bin]
test2 pid = 20
exit_status:0
miniOS:/ $
```

当启动分区为Orangefs时，所有的用户程序都被放在分区2的Orangefs文件系统中，分区2的Orangefs文件系统作为MiniOS的根文件系统，因此在MiniOS的shell程序中直接输入测试程序的名称即可，比如直接输入 `pptest1.bin`。

## 五、当启动分区的文件系统类型为Orangefs时需要对源码进行的修改

当启动分区的文件系统类型为Orangefs时，init.bin和shell\_0.bin都放在了分区2的orangefs文件系统中，MiniOS启动后分区2的orangefs文件系统作为根文件系统，因此需要变更一下init.bin和shell\_0.bin的路径，这些路径被硬编码在kernel的程序中，具体的修改如下。

在os/kernel/ktest.c中，将execve("/fat0/init.bin",NULL,NULL)改为execve("/init.bin",NULL,NULL);

```
//orangefs_test();
//while (1); 改为 "/init.bin"

execve("/fat0/init.bin",NULL,NULL);
//execve("fat0/test_0.bin");
//sys_execve("fat0/init.bin"); //modified by mingxuan 2021-4-6

while (1)
;
```

将user/init/init.c中的execve("fat0/shell\_0.bin",NULL,NULL);改为execve("/shell\_0.bin",NULL,NULL);

```

void main(int arg,char *argv[])
{
    int stdin = open("/dev/tty0",O_RDWR);
    int stdout= open("/dev/tty0",O_RDWR);
    int stderr= open("/dev/tty0",O_RDWR);

    printf("init:toatal_mem_size=%x\n",total_mem_size());

    //char filename[30] = "fat0/shell_0.bin";
    //printf("hello world!\n");
    if(0!=fork())
    {//father
        while(1);
    }
    else
        改为 "shell_0.bin"
    {//child
        // execve("fat0/shell_0.bin");
        execve("fat0/shell_0.bin",NULL,NULL);

        //execve(filename);
    }
}

```

上述两处修改是目前MiniOS中需要修复的问题，问题出现的原因在于MiniOS的kernel无法知道当前是从哪个启动分区启动的而且init进程执行的代码存在对路径的硬编码。