在Granite2 and Gemstone2 LSP中，SD card作为启动设备，在u-boot给kernel的启动参数上有

root=/dev/mmcblk1p2

表示rootfs在mmc block device的"mmcblk1p2"分区上。

in init/do_mounts.c

```c
static char __initdata saved_root_name[64];

static int __init root_dev_setup(char *line)
{
        strlcpy(saved_root_name, line, sizeof(saved_root_name));
        return 1;
}

__setup("root=", root_dev_setup);
```

root=xxxx,path不能太长，saved_root_name[64]限制了大小。

saved_root_name = "/dev/mmcblk1p2"

prepare_namespace()用于mount rootfs!

```
/*
 * Prepare the namespace - decide what/where to mount, load ramdisks, etc.
 */
void __init prepare_namespace(void)
{
        int is_floppy;

        if (root_delay) {
                printk(KERN_INFO "Waiting %d sec before mounting root device...\n",
                       root_delay);
                ssleep(root_delay);
        }

        /*
         * wait for the known devices to complete their probing
         *
         * Note: this is a potential source of long boot delays.
         * For example, it is not atypical to wait 5 seconds here
         * for the touchpad of a laptop to initialize.
         */
        wait_for_device_probe();
                                                                    ①

        md_run_setup();

        if (saved_root_name[0]) {
                                                                  ②
                root_device_name = saved_root_name;
                if (!strncmp(root_device_name, "mtd", 3) ||
                    !strncmp(root_device_name, "ubi", 3)) {
                        mount_block_root(root_device_name, root_mountflags);
                        goto out;
                }
                ROOT_DEV = name_to_dev_t(root_device_name);
              ③
                if (strncmp(root_device_name, "/dev/", 5) == 0)
                        root_device_name += 5;
        }

        if (initrd_load())

       ④
                goto out;

        /* wait for any asynchronous scanning to complete */
        if ((ROOT_DEV == 0) && root_wait) {
                                            ⑤
                printk(KERN_INFO "Waiting for root device %s...\n",
                        saved_root_name);
                while (driver_probe_done() != 0 ||
                                                  ⑥
                        (ROOT_DEV = name_to_dev_t(saved_root_name)) == 0)
```

```
          ⑦
                        msleep(100);
                async_synchronize_full();
        }

        is_floppy = MAJOR(ROOT_DEV) == FLOPPY_MAJOR;

        if (is_floppy && rd_doload && rd_load_disk(0))
                ROOT_DEV = Root_RAM0;

        mount_root();
                ⑧
out:
        devtmpfs_mount("dev");

        sys_mount(".", "/", NULL, MS_MOVE, NULL);
        sys_chroot(".");
}
```

①

在mount rootfs之时，不能有driver正在initialization。

②

在G2 LSP中，saved_root_name[] = "/dev/mmcblk1p2"。这是由root=/dev/mmcblk1p2 kernel parameter决定的。

③

ROOT_DEV = name_to_dev_t("/dev/mmcblk1p2");

由于mmc driver在初始化时(probe阶段)会把各个partition注册进gendisk.c。

这里name_to_dev_t()会向gendisk.c 查询"mmcblk1p2"对应的device number。如果能查到，说明mmc driver已经初始化完毕，否则就还没初始化。如果没有初始化，就需要下面第⑤⑥步的等待了。rootfs在MMC disk上，但MMC driver还未初始化完毕，自然要等待了。

in drivers/mmc/card/block.c

```c
static struct mmc_driver mmc_driver = {
        .drv            = {
                .name   = "mmcblk",
        },
.probe          =mmc_blk_probe,
        .remove         = mmc_blk_remove,
        .suspend        = mmc_blk_suspend,
        .resume         = mmc_blk_resume,
        .shutdown       = mmc_blk_shutdown,
};


static int mmc_blk_probe(struct mmc_card *card)
{
        struct mmc_blk_data *md, *part_md;
        char cap_str[10];

        /*
         * Check that the card supports the command class(es) we need.
         */
        if (!(card->csd.cmdclass & CCC_BLOCK_READ))
                return -ENODEV;

        mmc_fixup_device(card, blk_fixups);

        md = mmc_blk_alloc(card);
        if (IS_ERR(md))
                return PTR_ERR(md);

        string_get_size((u64)get_capacity(md->disk) << 9, STRING_UNITS_2,
                        cap_str, sizeof(cap_str));
        pr_info("%s: %s %s %s %s\n",
                md->disk->disk_name, mmc_card_id(card), mmc_card_name(card),
                cap_str, md->read_only ? "(ro)" : "");

        if (mmc_blk_alloc_parts(card, md))
                goto out;

        mmc_set_drvdata(card, md);

        if (mmc_add_disk(md))
                goto out;

        list_for_each_entry(part_md, &md->part, part) {
                if (mmc_add_disk(part_md))
                        goto out;
        }

        pm_runtime_set_autosuspend_delay(&card->dev, 3000);
        pm_runtime_use_autosuspend(&card->dev);

        /*
         * Don't enable runtime PM for SD-combo cards here. Leave that
```

```
	 * decision to be taken during the SDIO init sequence instead.
	 */
	if (card->type != MMC_TYPE_SD_COMBO) {
		pm_runtime_set_active(&card->dev);
		pm_runtime_enable(&card->dev);
	}

	return 0;

 out:
	mmc_blk_remove_parts(card, md);
	mmc_blk_remove_req(md);
	return 0;
}

static int mmc_add_disk(struct mmc_blk_data *md)
{
	int ret;
	struct mmc_card *card = md->queue.card;

add_disk(md->disk);
	md->force_ro.show = force_ro_show;
	md->force_ro.store = force_ro_store;
	sysfs_attr_init(&md->force_ro.attr);
	md->force_ro.attr.name = "force_ro";
	md->force_ro.attr.mode = S_IRUGO | S_IWUSR;
	ret = device_create_file(disk_to_dev(md->disk), &md->force_ro);
	if (ret)
		goto force_ro_fail;

	if ((md->area_type & MMC_BLK_DATA_AREA_BOOT) &&
	     card->ext_csd.boot_ro_lockable) {
		umode_t mode;

		if (card->ext_csd.boot_ro_lock & EXT_CSD_BOOT_WP_B_PWR_WP_DIS)
			mode = S_IRUGO;
		else
			mode = S_IRUGO | S_IWUSR;

		md->power_ro_lock.show = power_ro_lock_show;
		md->power_ro_lock.store = power_ro_lock_store;
		sysfs_attr_init(&md->power_ro_lock.attr);
		md->power_ro_lock.attr.mode = mode;
		md->power_ro_lock.attr.name =
					"ro_lock_until_next_power_on";
		ret = device_create_file(disk_to_dev(md->disk),
				&md->power_ro_lock);
		if (ret)
			goto power_ro_lock_fail;
	}
	return ret;

power_ro_lock_fail:
	device_remove_file(disk_to_dev(md->disk), &md->force_ro);
```

```
force_ro_fail:
        del_gendisk(md->disk);

        return ret;
}
```

in block/gendisk.c

```c
/**
 * add_disk - add partitioning information to kernel list
 * @disk: per-device partitioning information
 *
 * This function registers the partitioning information in @disk
 * with the kernel.
 *
 * FIXME: error handling
 */
voidadd_disk(struct gendisk *disk)
{
        struct backing_dev_info *bdi;
        dev_t devt;
        int retval;

        /* minors == 0 indicates to use ext devt from part0 and should
         * be accompanied with EXT_DEVT flag.  Make sure all
         * parameters make sense.
         */
        WARN_ON(disk->minors && !(disk->major || disk->first_minor));
        WARN_ON(!disk->minors && !(disk->flags & GENHD_FL_EXT_DEVT));

        disk->flags |= GENHD_FL_UP;

        retval = blk_alloc_devt(&disk->part0, &devt);
        if (retval) {
                WARN_ON(1);
                return;
        }
        disk_to_dev(disk)->devt = devt;

        /* ->major and ->first_minor aren't supposed to be
         * dereferenced from here on, but set them just in case.
         */
        disk->major = MAJOR(devt);
        disk->first_minor = MINOR(devt);

        disk_alloc_events(disk);

        /* Register BDI before referencing it from bdev */
        bdi = &disk->queue->backing_dev_info;
        bdi_register_dev(bdi, disk_devt(disk));

        blk_register_region(disk_devt(disk), disk->minors, NULL,
                            exact_match, exact_lock, disk);
        register_disk(disk);
        blk_register_queue(disk);

        /*
         * Take an extra ref on queue which will be put on disk_release()
         * so that it sticks around as long as @disk is there.
         */
        WARN_ON_ONCE(!blk_get_queue(disk->queue));

        retval = sysfs_create_link(&disk_to_dev(disk)->kobj, &bdi->dev->kobj,
                                   "bdi");
        WARN_ON(retval);

        disk_add_events(disk);
}
EXPORT_SYMBOL(add_disk);
```

④

u-boot启动kernel时

bootm 0x400000 - 0xf00000

这里的"-"就是在没有ramdisk的情况下的占位符。

在G2 LSP中没有使用ramdisk,所以这里为什么都不做。

⑤

in init/do_mounts.c

```
static int root_wait;

static int __init rootwait_setup(char *str)

{

        if (*str)

                return 0;

        root_wait = 1;

        return 1;

}
```

in kernel-parameters.txt

```
rootwait    [KNL] Wait (indefinitely) for root device to show up.
            Useful for devices that are detected asynchronously
            (e.g. USB and MMC devices).
```

在LSP中u-boot启动kernel时带有rootwait参数

在boot parameter中添加"debug"参数后可以看到如下log

```
Waiting for root device /dev/mmcblk1p2...
mmc1: new high speed SDHC card at address aaaa
mmcblk1: mmc1:aaaa SL16G 14.8 GiB
 mmcblk1: p1 p2
EXT3-fs (mmcblk1p2): recovery required on readonly filesystem
EXT3-fs (mmcblk1p2): write access will be enabled during recovery
kjournald starting.  Commit interval 5 seconds
EXT3-fs (mmcblk1p2): recovery complete
EXT3-fs (mmcblk1p2): mounted filesystem with ordered data mode
VFS: Mounted root (ext3 filesystem) readonly on device 179:34.
devtmpfs: mounted
```

由于这里返回的ROOT_DEV为零，也就是因为MMC driver还未初始化，所以name_to_dev_t()无法作转换，只能返回0。

kernel只能在这里等待。一直等到MMC driver的初始化。从上面的log中可清晰的看到这个过程。

⑥

表示有driver还在initialization，见《怎样确定系统当前是否有driver处于初始化中?》笔记。

⑦

还无法从"mmcblk1p2"得到device number。表示mmc device driver还未辨认出SD card的partition。

mmc device driver的初始化在之前已经初始化完毕，但mmc driver要辨认出SD card，这还是需要时间的。由于mount SD partition与辨认出SD partition是并行的,而且mount动作又依赖于mmc正确辨认SD partition，所以这里要有等待动作。

⑧

运行到这里，表示mmc driver初始化已经完成，可以把mount "mmcblk1p2" partition为rootfs了。

in init/do_mounts.c

```c
void __init mount_root(void)
{
#ifdef CONFIG_ROOT_NFS
        if (ROOT_DEV == Root_NFS) {
                if (mount_nfs_root())
                        return;

                printk(KERN_ERR "VFS: Unable to mount root fs via NFS, trying floppy.\n");
                ROOT_DEV = Root_FD0;
        }
#endif
#ifdef CONFIG_BLK_DEV_FD
        if (MAJOR(ROOT_DEV) == FLOPPY_MAJOR) {
                /* rd_doload is 2 for a dual initrd/ramload setup */
                if (rd_doload==2) {
                        if (rd_load_disk(1)) {
                                ROOT_DEV = Root_RAM1;
                                root_device_name = NULL;
                        }
                } else
                        change_floppy("root floppy");
        }
#endif
#ifdef CONFIG_BLOCK
        create_dev("/dev/root", ROOT_DEV);
        mount_block_root("/dev/root", root_mountflags);
#endif
}
```

实际就两行

create_dev("/dev/root", ROOT_DEV); **(A)**

mount_block_root("/dev/root", root_mountflags); **(B)**

(A)

create device node, make "dev/root" ==> ROOT_DEV

(B)

见怎样《mount rootfs》note.