

in init/main.c

```

1.      /* Open the /dev/console on the rootfs, this should never fail */
2.      if (sys_open((const char __user *) "/dev/console", O_RDWR, 0) < 0)
3.          pr_err("Warning: unable to open an initial console.\n");
4.
5.      (void) sys_dup(0);
6.      (void) sys_dup(0);
7.      /*
8.       * check if there is an early userspace init.  If yes, let it do all
9.       * the work
10.      */
11.
12.      if (!ramdisk_execute_command)
13.          ramdisk_execute_command = "/init";
14.
15.      if (sys_access((const char __user *) ramdisk_execute_command, 0) != 0) {
16.          ramdisk_execute_command = NULL;
17.          prepare_namespace();
18.      }
19.
20.      /*
21.       * Ok, we have completed the initial bootup, and
22.       * we're essentially up and running. Get rid of the
23.       * initmem segments and start the user-mode stuff..
24.      */
25.
26.      /* rootfs is available now, try loading default modules */
27.      load_default_modules();

```

③处会mount rootfs，但在有rootfs以前，却...

①处的打开"/dev/console"是什么意思？ "/"都还没有呢？

②在没有定义"rdinit=XXX"的情况下，去access("/init") 是什么意思？没有"/"，难道access "/init"有可能成功吗？

Answer:

in kernel config file

```
CONFIG_BLK_DEV_INITRD=y
```

```
CONFIG_INITRAMFS_SOURCE=""
```

```
CONFIG_RD_GZIP=y
```

在u-boot启动kernel时的bootcmd如下：

```
bootcmd: mmc dev 1;ext2load mmc 1:2 0x400000 /boot/ulmage;ext2load mmc 1:2 0xf00000  
/boot/mv6220-toc.dtb;setenv bootargs $bootargs root=/dev/mmcblk1p2  
uio_pdrv_genirq.of_id=generic-uio rootwait;bootm 0x400000 - 0xf00000
```

```
"bootm 0x400000 - 0xf00000"
```

这里 "-"就是initramfs的placeholder,由于没有用户提供的initramfs,所以这里用 "-"代替。

G2 LSP没有使用initramfs，**但kernel总有一个"initramfs"(错，可以没有)**。如果用户有自己的，那就用用户指定的，如果没有，那么就用kernel自己生成的最简单的default initramfs。

在build的kernel的usr directory

```
1. walterzh@walterzh-ThinkPad-T440p:~/gerrit/linux/3.18.7+gitAUTOINC+e2438e08f1-r0/linux-granite2-standard-build/usr$ ls -l
2. 总用量 36
3. -rw-r--r-- 1 walterzh walterzh 399 12月 17 17:49 built-in.mod.c
4. -rw-r--r-- 1 walterzh walterzh 1000 12月 17 17:49 built-in.o
5. -rwxr-xr-x 1 walterzh walterzh 19292 12月 17 17:49 gen_init_cpio
6. -rw-r--r-- 1 walterzh walterzh 134 12月 17 17:49 initramfs_data.cpio.gz
7. -rw-r--r-- 1 walterzh walterzh 912 12月 17 17:49 initramfs_data.o
8. -rw-rw-r-- 1 walterzh walterzh 0 12月 17 17:53 modules.builtin
9. -rw-rw-r-- 1 walterzh walterzh 0 12月 17 17:53 modules.order
```

initramfs_data.cpio.gz就是在用户没有指定生成initramfs的情况 (CONFIG_INITRAMFS_SOURCE="")下生成的默认的initramfs.

```
1. walterzh@walterzh-ThinkPad-T440p:~/tmp$ cpio -ivmd < initramfs_data.cpio
2. dev
3. cpio: dev/console: 函数 mknod 失败: 不允许的操作
4. dev/console
5. root
6. 1 块
```

虽然运行cpio command fail，但大致也知道了在default initramfs中有点什么东西。

/root

/root/dev

/root/dev/console

就这么点东西！

in init/initramfs.c

```

1. static int __init populate_rootfs(void)
2. {
3.     char *err = unpack_to_rootfs(__initramfs_start, __initramfs_size);
4.     if (err)
5.         panic("%s", err); /* Failed to decompress INTERNAL initramfs */
6.     if (initrd_start) {
7. #ifdef CONFIG_BLK_DEV_RAM
8.         int fd;
9.         printk(KERN_INFO "Trying to unpack rootfs image as initramfs...\n
10. ");
11.         err = unpack_to_rootfs((char *)initrd_start,
12.                                initrd_end - initrd_start);
13.         if (!err) {
14.             free_initrd();
15.             goto done;
16.         } else {
17.             clean_rootfs();
18.             unpack_to_rootfs(__initramfs_start, __initramfs_size);
19.         }
20.         printk(KERN_INFO "rootfs image is not initramfs (%s)"
21.                "; looks like an initrd\n", err);
22.         fd = sys_open("/initrd.image",
23.                       O_WRONLY|O_CREAT, 0700);
24.         if (fd >= 0) {
25.             ssize_t written = xwrite(fd, (char *)initrd_start,
26.                                     initrd_end - initrd_start);
27.
28.             if (written != initrd_end - initrd_start)
29.                 pr_err("/initrd.image: incomplete write (%zd != %
30. ld)\n",
31.                        written, initrd_end - initrd_start);
32.
33.             sys_close(fd);
34.             free_initrd();
35.         }
36.     done:
37. #else
38.         printk(KERN_INFO "Unpacking initramfs...\n");
39.         err = unpack_to_rootfs((char *)initrd_start,
40.                                initrd_end - initrd_start);
41.         if (err)
42.             printk(KERN_EMERG "Initramfs unpacking failed: %s\n", err
43. );
44.         free_initrd();
45. #endif
46.
47.     /*
48.      * Try loading default modules from initramfs. This gives
49.      * us a chance to load before device_initcalls.
50.      */
51.     load_default_modules();
52. }
53. return 0;
54. }

```

```
51. rootfs_initcall(populate_rootfs);
```

populate_rootfs()就用于把gz压缩格式的cpio文件展开。

__initramfs_start symbol is defined in vmlinux.lds

```
1. __initramfs_start = .; *(.init.ramfs) . = ALIGN(8); *(.init.ramfs.info)
```

and __initramfs_size is defined in usr/initramfs_data.S

```
1. .section .init.ramfs,"a"  
2. __irf_start:  
3. .incbin __stringify(INITRAMFS_IMAGE)  
4. __irf_end:  
5. .section .init.ramfs.info,"a"  
6. .globl VMLINUX_SYMBOL(__initramfs_size)  
7. VMLINUX_SYMBOL(__initramfs_size):  
8. #ifdef CONFIG_64BIT  
9.     .quad __irf_end - __irf_start  
10. #else  
11.     .long __irf_end - __irf_start  
12. #endif
```

__initramfs_start表示了initramfs的开始处，__initramfs_size表示了initramfs的size。

而rootfs_initcall(populate_rootfs);

表示populate_rootfs() function将在initcall level "rootfs"被调用。

in include/linux/init.h

```

1.  #define rootfs_initcall(fn)                __define_initcall(fn, rootfs)
2.
3.  in include/asm-generic/vmlinux.lds.h
4.
5.  #define INIT_CALLS                                \
6.          VMLINUX_SYMBOL(__initcall_start) = .;    \
7.          *(.initcallearly.init)                  \
8.          INIT_CALLS_LEVEL(0)                      \
9.          INIT_CALLS_LEVEL(1)                      \
10.         INIT_CALLS_LEVEL(2)                      \
11.         INIT_CALLS_LEVEL(3)                      \
12.         INIT_CALLS_LEVEL(4)                      \
13.         INIT_CALLS_LEVEL(5)                      \
14.         INIT_CALLS_LEVEL(rootfs)                  \
15.         INIT_CALLS_LEVEL(6)                      \
16.         INIT_CALLS_LEVEL(7)                      \
17.         VMLINUX_SYMBOL(__initcall_end) = .;

```

initcall level "rootfs"夹在level 5和6之间，但反正会在init/main.c中的do_basic_setup()中被do_initcalls()调用。

也就是当kernel初始化期间运行到①时，虽然真正的SD card上的rootfs,也就是root=/dev/mmcblk1p2中的分区还没有被mount，但由kernel生成的default initramfs的rootfs已经是存在了。虽然这个rootfs极其简单，简单到几乎什么都没有，就是为了①code successfully,但②处是肯定失败的(也就是sys_access system call返回non-zero),因为在initramfs_data.cpio中就根本没有/init这个文件。但不要紧，正是由于②的失败才会运行③，而在③里会mount真正的SD card上的rootfs。