Linux内核远控

残夜
canye@whitecell-club.org

PART 01

背景介绍

PART 02

内核启动过程

PART 03

实验过程

PART 04

实验结果

目 录

PART
01

背景介绍

背景介绍

Android是一种基于Linux的自由及开放源代码的操作系统，主要使用于移动设备，如智能手机和平板电脑，并且逐渐扩展到平板电脑及其他领域上，如电视、数码相机、游戏机等。不管是在哪里，安卓的设备随处可见，由于其一直上升的使用率和系统开源的特性，安卓安全也越来越受到人们的重视。

APP

安卓应用程序

NDK

ROM

ROM包

Root

获取最高权限

内核

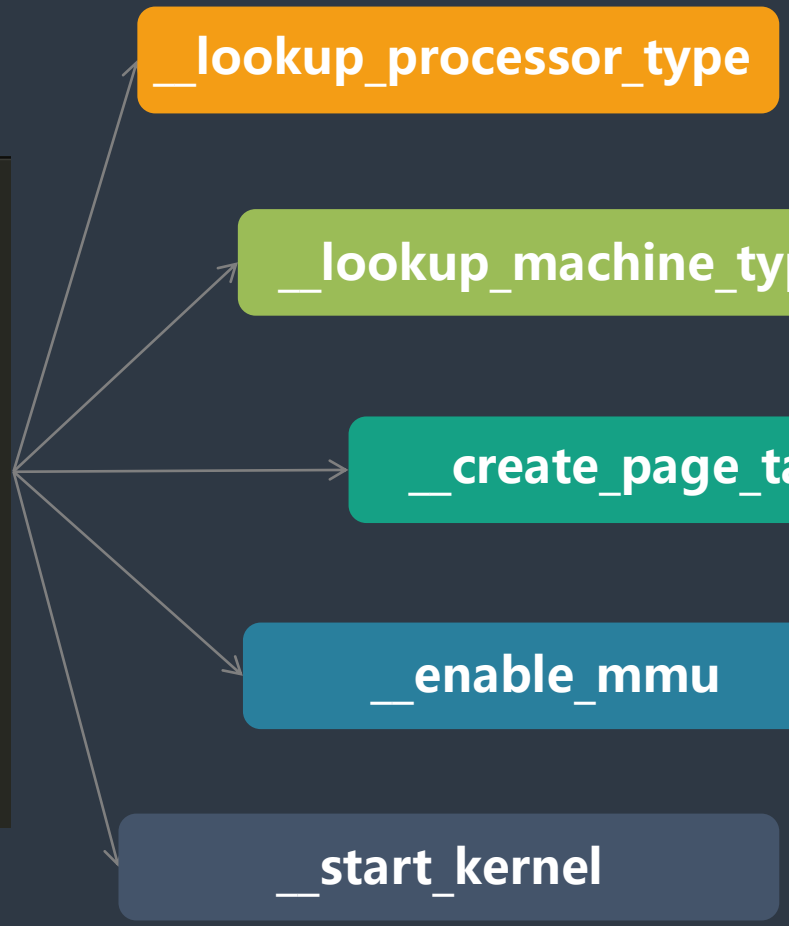
安卓内核

PART
02

内核启动过程

内核启动过程

```
__lookup_processor_type ->
__lookup_machine_type ->
__create_page_tables ->
__enable_mmu ->
start_kernel
    +--- ...
    |
    +---rest_init
    +---kernel_init (kernel_thread)
    | +--- ...
    | +---init_post
    | +--- ...
    | +---run_init_process
    |
    +---kthreadd (kernel_thread)
```



确定处理器类型。

确定机器类型。

开始创建内核使用的段页表,只映射了内核本身和内核本身所在的页表的基址PA及启动参数所在的内存。

激活MMU硬件，将域访问寄存器和页表指针值加载到ARM处理器的寄存器。

开始执行第一个 C函数。

内核启动过程

__enable_mmu

arch/arm/kernel/head-common.S

```
80 __mmap_switched:
81 adr r3, __mmap_switched_data
82
83 ldmia r3!, {r4, r5, r6, r7}
…
90 mov fp, #0 @ Clear BSS (and zero fp)
91 1: cmp r6, r7
92 strcc fp, [r6],#4
93 bcc 1b
94 ...
103 b start_kernel
104 ENDPROC(__mmap_switched)
105
106 .align 2
107 .type __mmap_switched_data, %object
108 __mmap_switched_data:
109 .long __data_loc @ r4
110 .long _sdata @ r5
111 .long __bss_start @ r6
112 .long _end @ r7
```

__start_kernel

```
801 static noinline int init_post(void)
802 {
803 /* need to finish all async __init code before freeing the memory */
804 async_synchronize_full();
805 free_initmem();
806 mark_rodata_ro();
807 system_state = SYSTEM_RUNNING;
808 numa_default_policy();
```

```
static void run_init_process(const char *init_filename)
{
  argv_init[0] = init_filename;
  kernel_execve(init_filename, argv_init, envp_init);
}
```

run_init_process 只是对 kernel_execve 的一层封装。而 kernel_execve 则与execve 一样最终将调用 do_execve。

fork/exec 模型

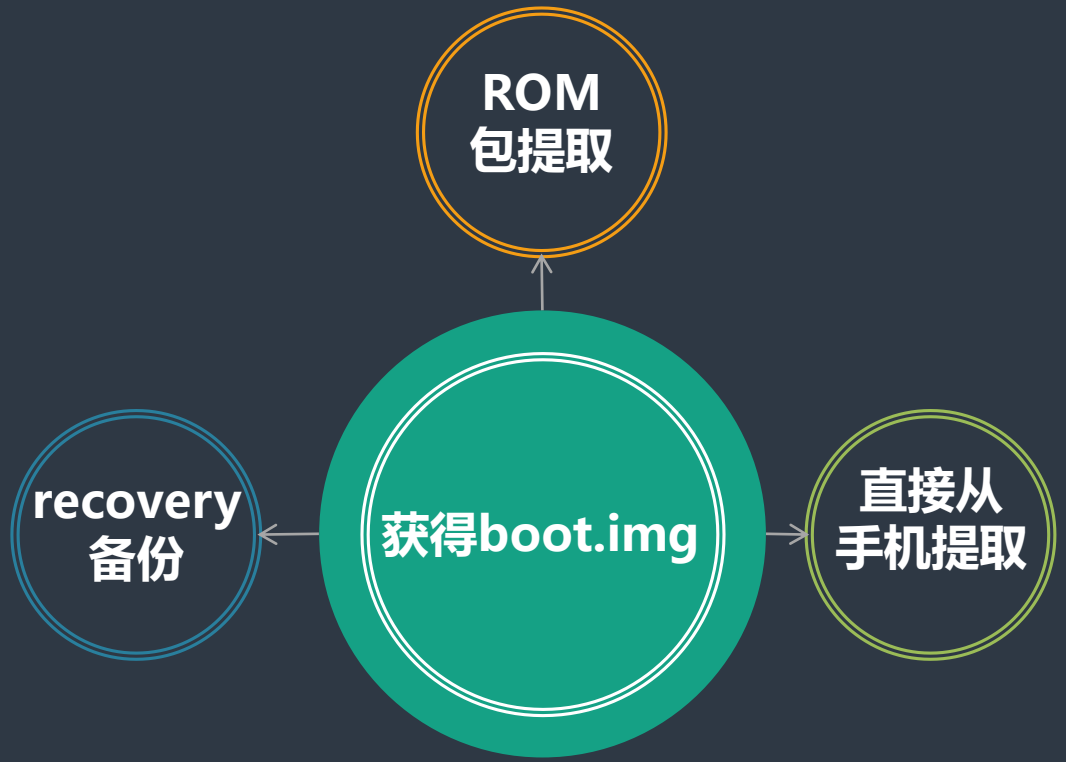
PART 04

实验过程

目 录

boot. img

获得boot.img



boot. img

boot.img格式

```
** +-----------------+
** | boot header     | 1 page
** +-----------------+
** | kernel          | n pages
** +-----------------+
** | ramdisk         | m pages
** +-----------------+
** | second stage    | o pages
** +-----------------+
**
** n = (kernel_size + page_size - 1) / page_size
** m = (ramdisk_size + page_size - 1) / page_size
** o = (second_size + page_size - 1) / page_size
```

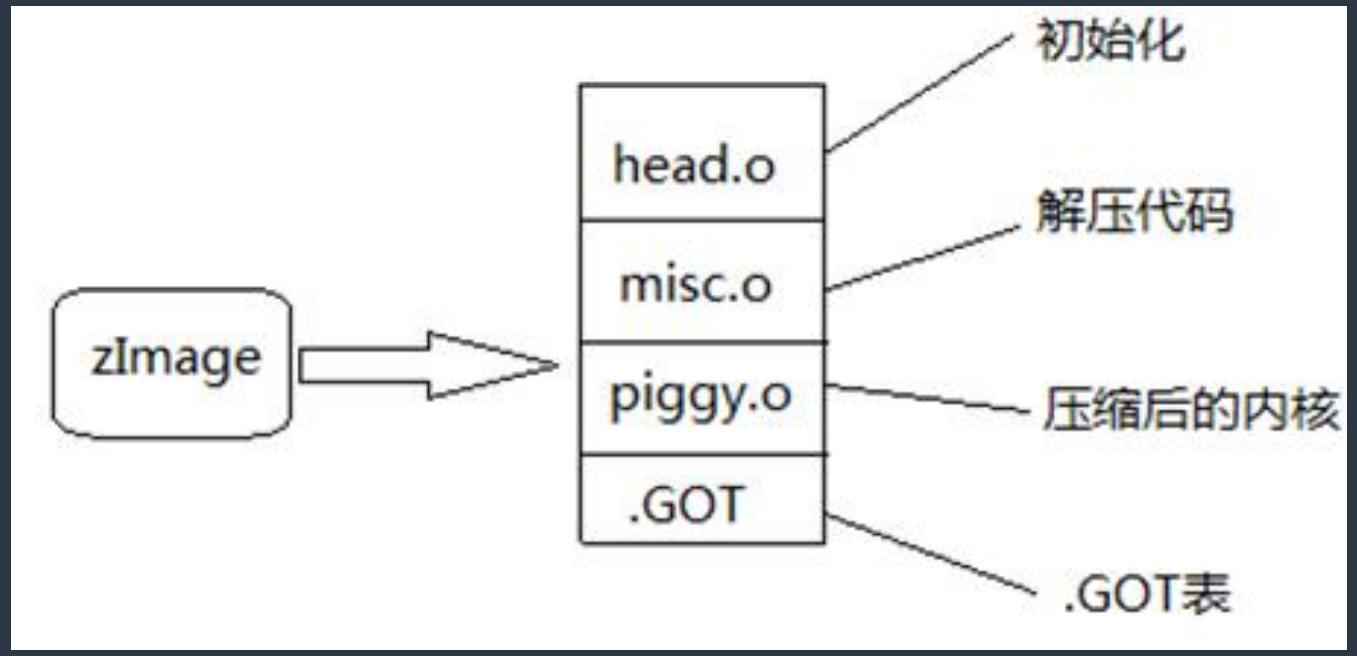
```
#define BOOT_MAGIC "ANDROID!"
#define BOOT_MAGIC_SIZE 8
#define BOOT_NAME_SIZE 16
#define BOOT_ARGS_SIZE 512
struct boot_img_hdr
{
    unsigned char magic[BOOT_MAGIC_SIZE];

    unsigned kernel_size;  /* size in bytes */
    unsigned kernel_addr;  /* physical load addr */

    unsigned ramdisk_size; /* size in bytes */
    unsigned ramdisk_addr; /* physical load addr */

    unsigned second_size;  /* size in bytes */
    unsigned second_addr;  /* physical load addr */

    unsigned tags_addr;    /* physical addr for kernel tags */
    unsigned page_size;    /* flash page size we assume */
    unsigned unused[2];    /* future expansion: should be 0 */

    unsigned char name[BOOT_NAME_SIZE]; /* asciiz product name */

    unsigned char cmdline[BOOT_ARGS_SIZE];
```

boot.img

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	41	4E	44	52	4F	49	44	21	40	42	5D	00	00	80	00	00	ANDROID!@B].....
0010h:	6C	DF	10	00	00	00	00	02	00	00	00	00	00	00	F0	00	l..............
0020h:	00	00	E0	01	00	08	00	00	00	50	0F	00	00	00	00	00	.........P.....
0030h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0040h:	63	6F	6E	73	6F	6C	65	3D	6E	75	6C	6C	20	61	6E	64	console=null and

zImage

zImage文件



主要完成一些初始化工作，为解压内核的代码准备好运行环境

负责把内核解压到指定的内存里

生成的Image经过gzip压缩后得到文件

主要是在访问全局变量和全局函数采用位置无关的重定位方法

解压内核

确认压缩格式和标志方法从而确定内核的起始位置

用zImage的大小搜索

用gzip解压。

修改内核

内核修改步骤

1. 添加编译后的代码
2. 修改部分变量的数值，如内核的大小
3. 压缩内核
4. 修改boot.img部分变量的数值

远控软件

远控软件功能设计

数据库操作

文件操作

PART
04

实验结果

实验结果

运行界面

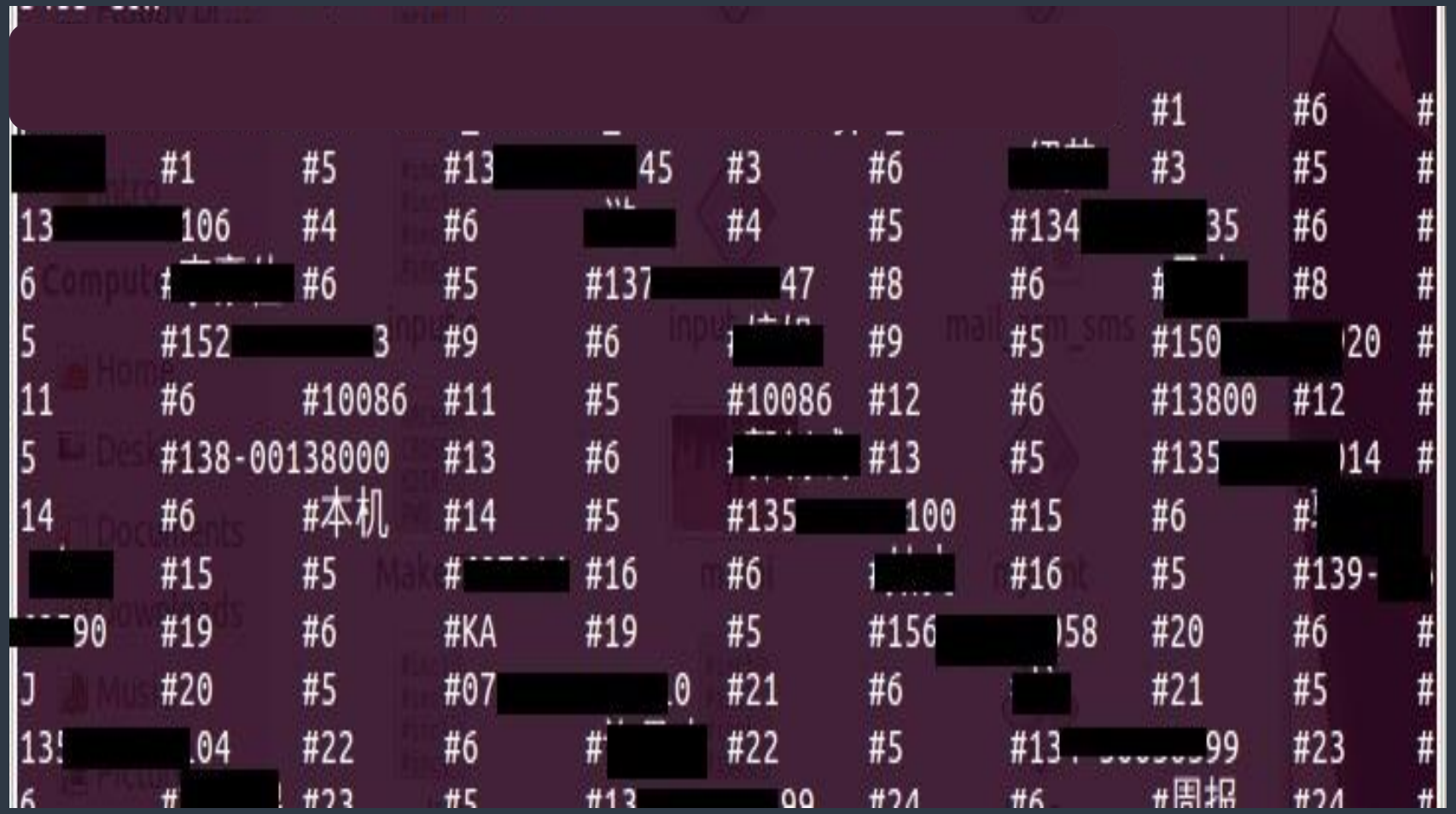
实验结果

获取短信信息


00客户，您当月个人流量套餐内流量共100M，截至01日16时09分，已用18.58M，剩余81.42M，其中：国
通用流量余81.42M，更多流量详情请点击 gd.10086.cn/app24 下载“广东移动10086”APP 查询 #42

实验结果

获取联系人信息



实验结果

修改数据库信息

```
sms set body='look at your back,i am watching you!' where address='1
lenth:107AXLINE];
```

THANKS